Introduction to Large Language Modeling

DPPD Pre-Conference Workshop
September 21st, 2025

Dr. Björn E. Hommel | bjoern.hommel@uni-leipzig.de
Department of Personality Psychology and Psychological Assessment
Wilhelm Wundt Institute of Psychology
Leipzig University

Introduction to Large Language Modeling

Agenda

- **Setup & Introduction**

1. **A Brief History of Computational Linguistics**
2. **The Transformer Model**
3. **The Hugging Face Ecosystem**
4. **Encoder-Only Models**
5. **Decoder-Only Models**
6. **Encoder-Decoder Models**
7. **Training Transformer Models**

- **Discussion**

Formats:

- Presentations on Theory and Practice
- 8 Python Exercises

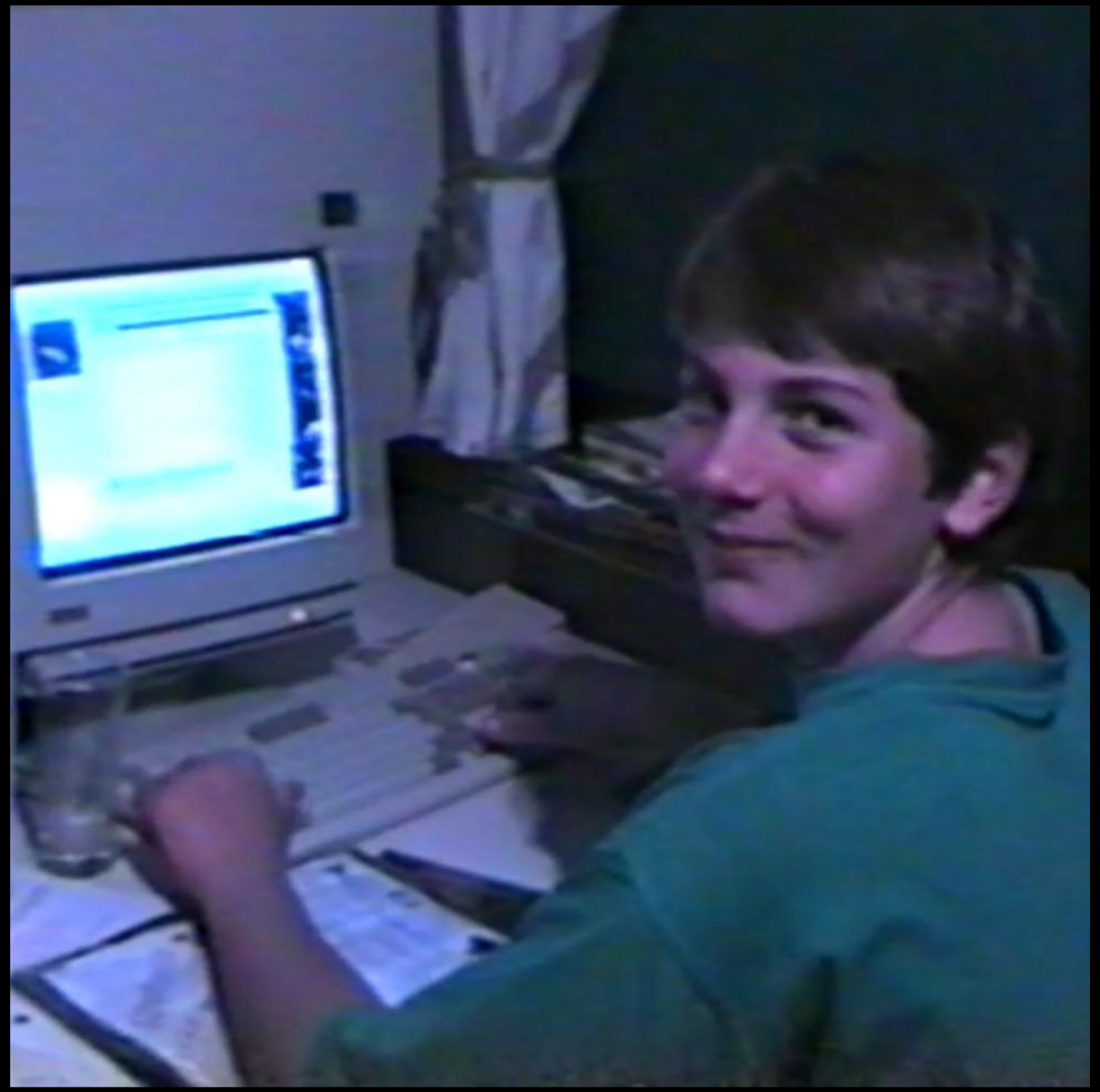
Quick Introduction

Briefly introduce yourself

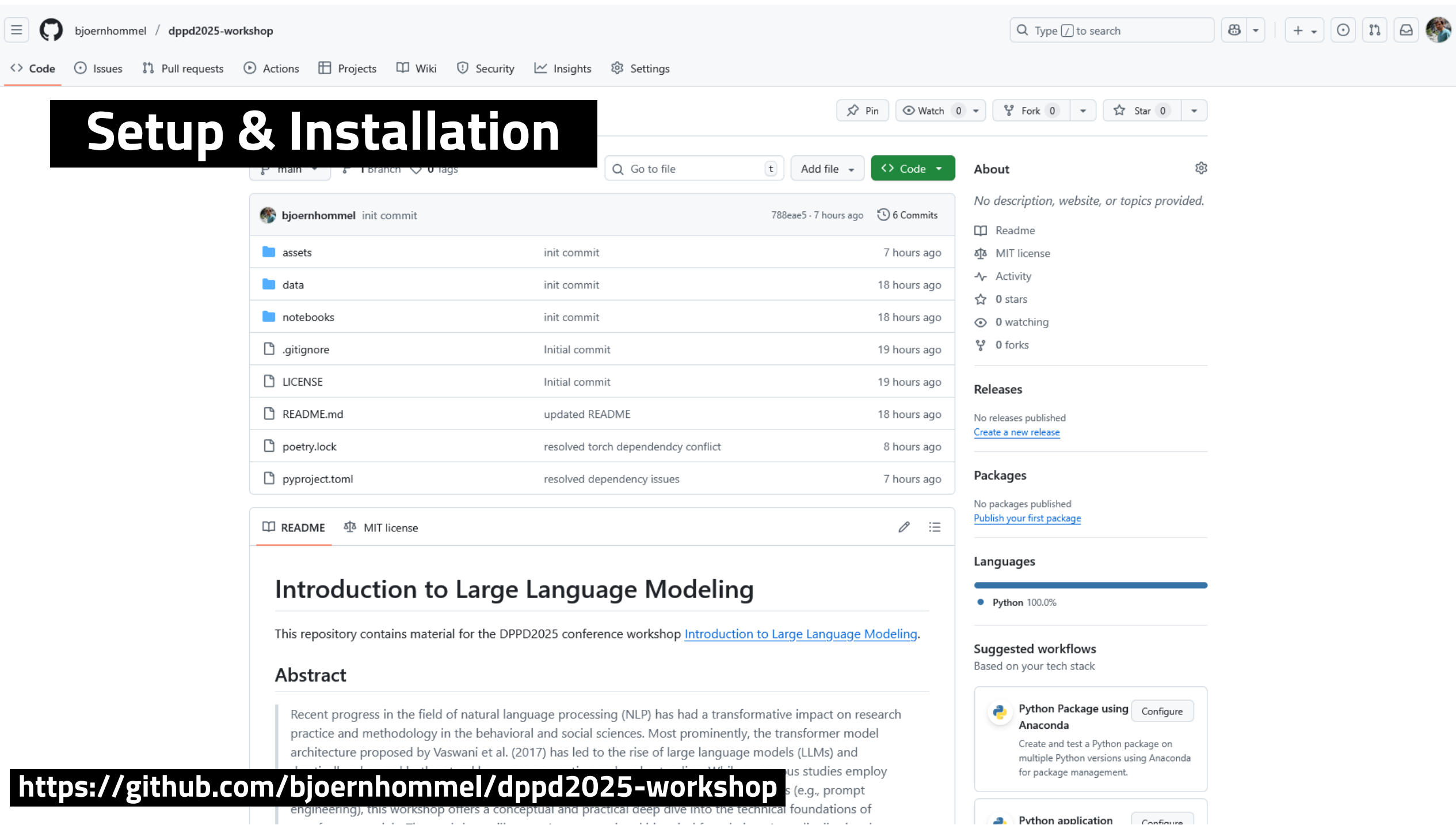
- What's current role or research focus?
- Any prior experiences with natural language processing?
- What are you hoping to learn today?

Now everyone

- Please rate your Python skills from 1 to 5.



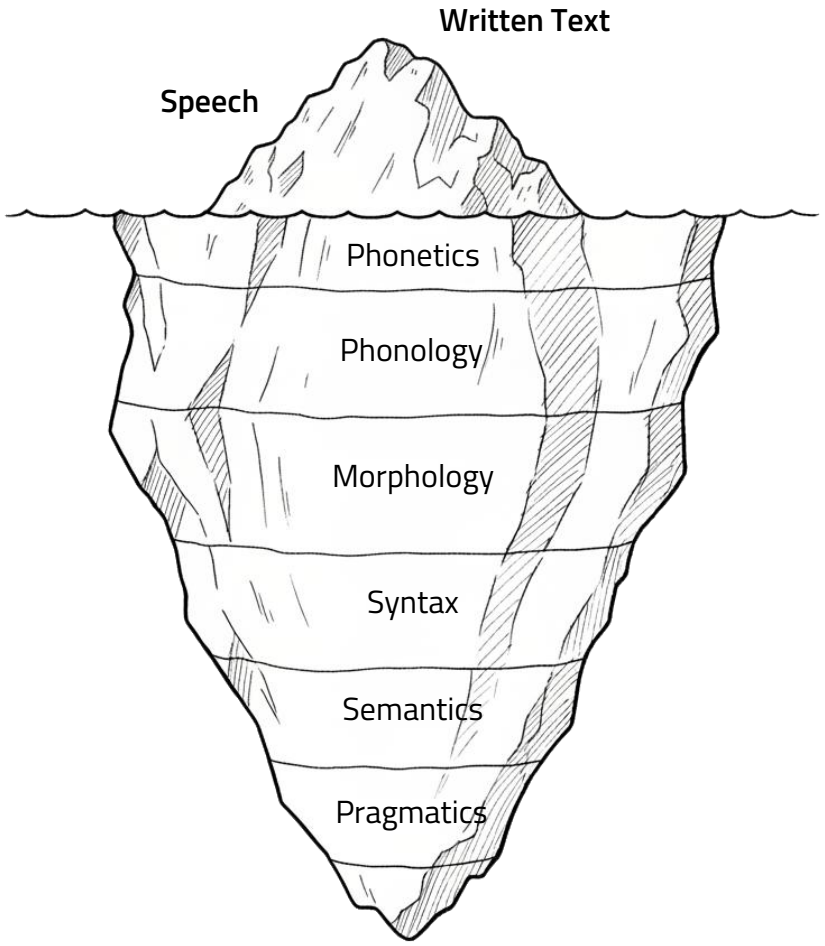
Setup & Installation



https://github.com/bjoernhommel/dppd2025-workshop

Language and Language Modeling

The Levels of Linguistics



Dr. Björn E. Hommel | bjoern.hommel@uni-leipzig.de

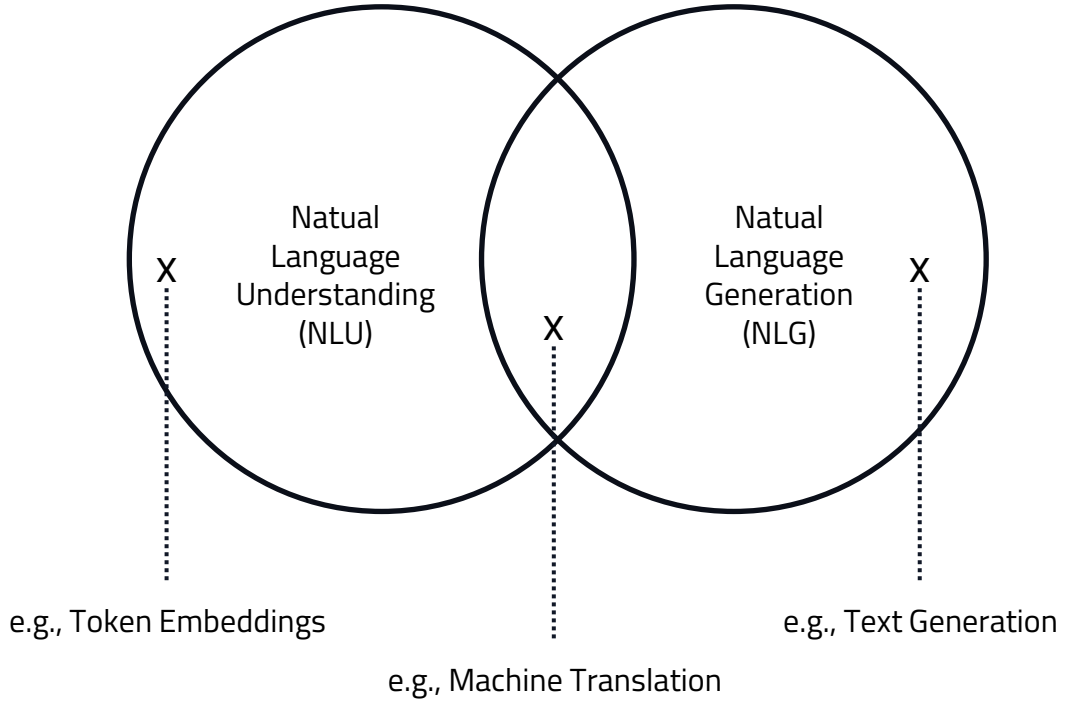
Language. It’s complicated.

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

Language and Language Modeling

Natural Language Processing

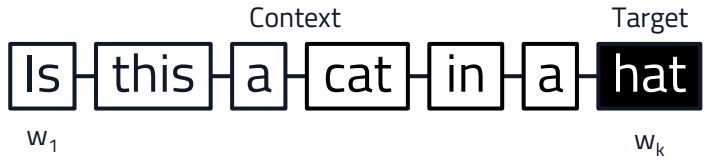
Describes as set of methods for making human language accessible for computers.



Eisenstein (2018)

Language and Language Modeling

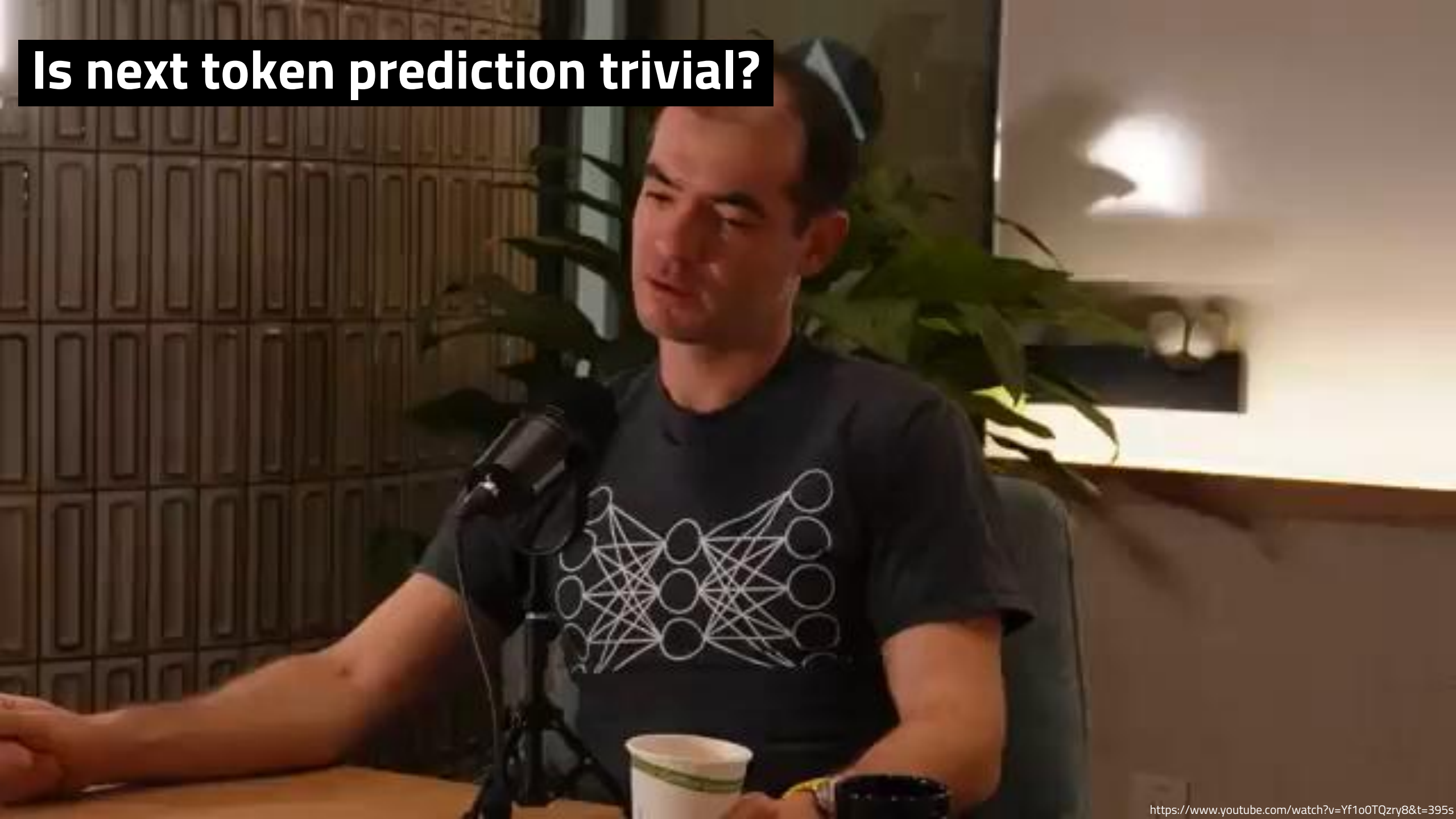
What is a Language Model?



$$\prod_{k=1}^{n} P\left(w_k \middle| w_{[1,k-1]}\right)$$

Jurafsky & Martin (2019)

Is next token prediction trivial?



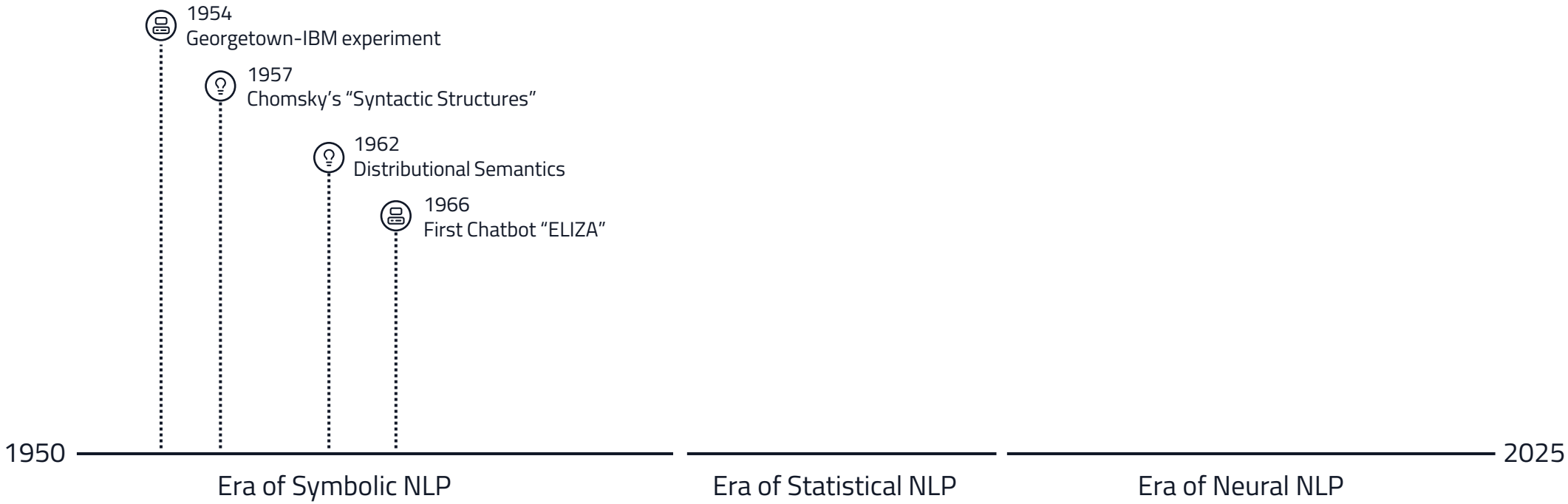
https://www.youtube.com/watch?v=Yf1o0TQzry8&t=395s

Part I

A Brief History of Computational Linguistics

A Brief History of Computational Linguistics

Selected Milestones



A Brief History of Computational Linguistics

Era of Symbolic NLP

Early “chatterbots” used very simple rulesets to mimic thoughtful responses.

- Simple transformations (e.g., "I am" → "you are")
- Reflecting statements back as questions (e.g., "I feel sad" → "Why do you feel sad?")
- Response Templates (e.g., “Tell me more about that”)

https://developers.googleblog.com/en/t5gemma/

A Brief History of Computational Linguistics

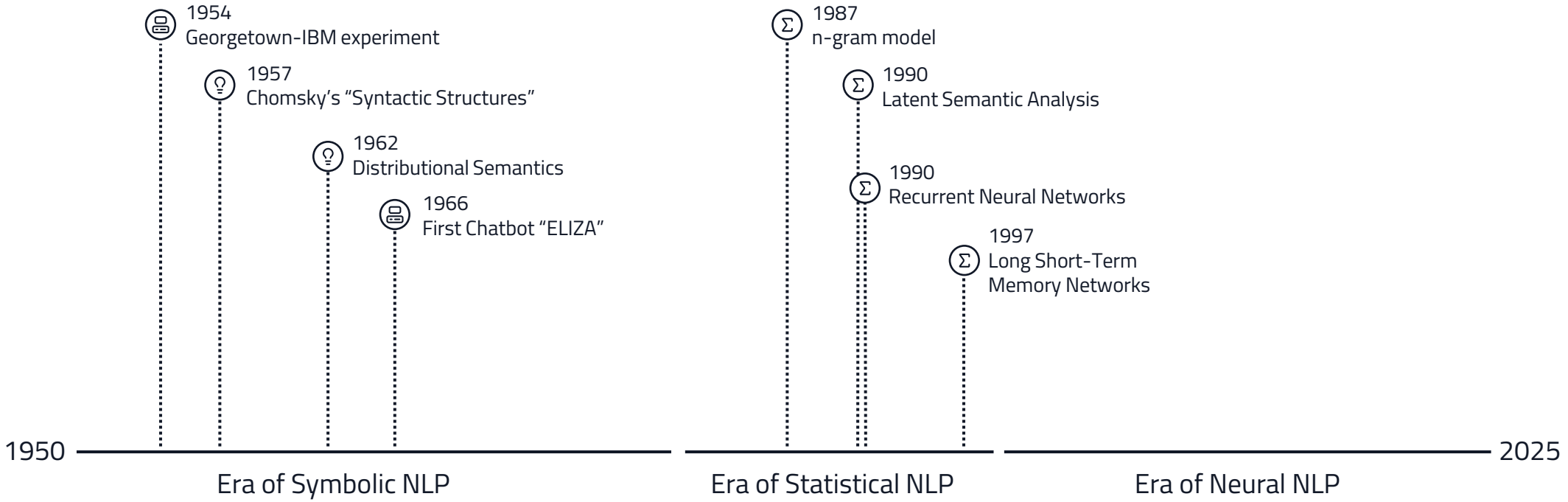
Era of Symbolic NLP

The 1950s witnessed the emergence of two influential philosophies that would shape the field of natural language processing for the next decades:

- **Chomsky & Formal Grammar**: Aimed to create mathematical models of linguistic structure using syntax trees and transformational rules.
    - Later evolved into arguments about innate language capacity.
- **Distributional Hypothesis:** Meaning and grammar arise from observing how words and structures are used in context.
    - Wittgenstein: "The meaning of a word is its use in language“
    - Firth: "You shall know a word by the company it keeps"

A Brief History of Computational Linguistics

Selected Milestones



A Brief History of Computational Linguistics

Era of Statistical NLP

n-gram Models: Approximate language by counting how often word sequences (n-grams) occur.

Unigram Model

The Party intellectual knows **in** which direction his memories must be altered; he therefore knows that he is playing tricks with reality; but by the exercise of doublethink he also satisfies himself that reality is not violated.

Bigram Model

The Party intellectual knows **in which** direction his memories must be altered; he therefore knows that he is playing tricks with reality; but by the exercise of doublethink he also satisfies himself that reality is not violated.

Trigram Model

The Party intellectual knows **in which direction** his memories must be altered; he therefore knows that he is playing tricks with reality; but by the exercise of doublethink he also satisfies himself that reality is not violated.

4-gram Model

The Party intellectual knows **in which direction his** memories must be altered; he therefore knows that he is playing tricks with reality; but by the exercise of doublethink he also satisfies himself that reality is not violated.

5-gram Model

The Party intellectual knows **in which direction his memories** must be altered; he therefore knows that he is playing tricks with reality; but by the exercise of doublethink he also satisfies himself that reality is not violated.

Smaller context windows:
Trivial predictions

Larger context windows:
Out-of-distribution

Jurafsky & Martin (2019)

A Brief History of Computational Linguistics

Era of Statistical NLP

Sparse Embeddings

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9
w_1	0	.64	0	0	0	0	0	0	0
w_2	0	.34	.43	0	0	0	0	0	0
w_3	0	0	0	0	0	0	.71	0	0
w_4	0	0	0	0	.54	0	0	0	0
w_5	.21	0	0	0	0	.30	0	0	0
w_6	.34	0	0	0	0	0	.86	0	0
w_6	0	0	0	.12	0	0	0	.22	.41

Dimensionality	High
Semantic Richness	Low
Interpretability	Moderate
Curse of Dimensionality	High

Dense Embeddings

	d_1	d_2	d_3	d_4	d_5	d_6
w_1	.38	.75	.91	.43	.75	.73
w_2	.61	.20	.20	.57	.04	.09
w_3	.08	.48	.24	.61	.46	.37
w_4	.03	.49	.41	.26	.18	.39
w_5	.17	.58	.39	.59	.57	.06
w_6	.57	.46	.40	.56	.64	.79
w_6	.38	.75	.91	.43	.75	.73

Dimensionality	Low:ish
Semantic Richness	High
Interpretability	Low
Curse of Dimensionality	Moderate

A Brief History of Computational Linguistics

Era of Statistical NLP

Latent Semantic Analysis

1. Document-Term Matrix

terms (t)

documents (d)	is	mat	a	cat	the	hat
is this a cat in a hat	1	1	2	1	0	1
the cat sat on the mat	0	1	0	1	2	0
a cat wearing a hat	0	0	2	1	0	1

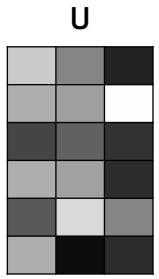
c = concepts

2. Term-Weighting (TF-IDF)

$$freq_{t,d} \times log\left(\frac{N_d}{df_{t \in d}}\right)$$

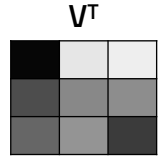
3. Singular Value Decomposition (SVD)

Term Embeddings



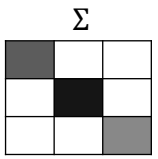
t, c

Document Embeddings



c, d

Weight Matrix

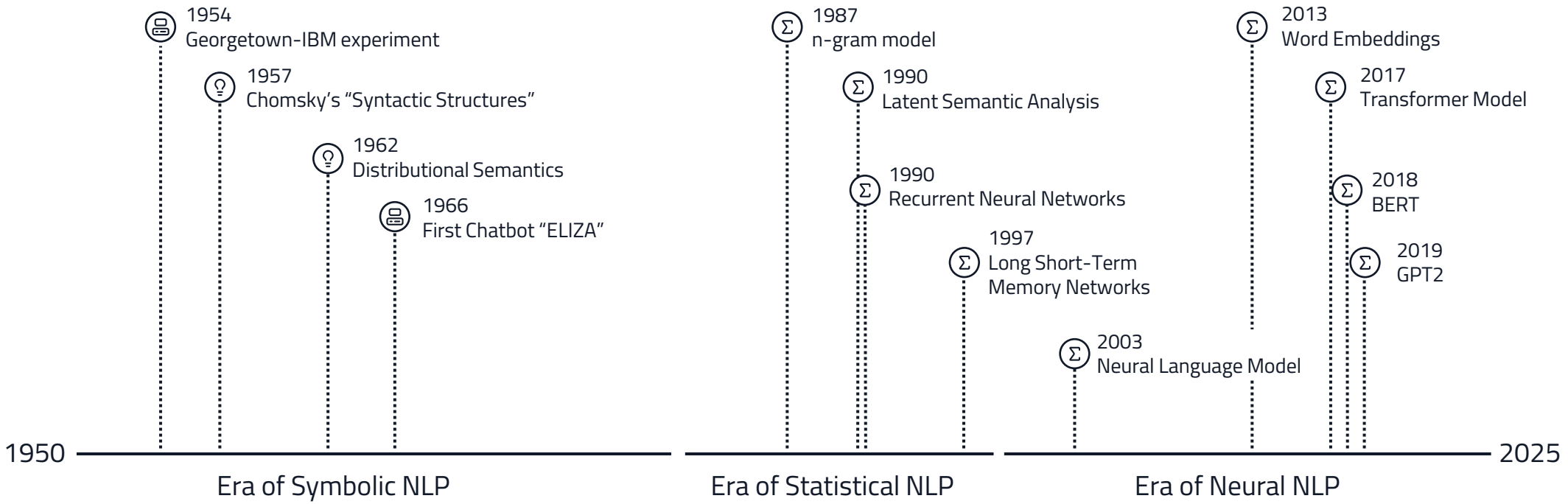


c, c

Jurafsky & Martin (2019)

A Brief History of Computational Linguistics

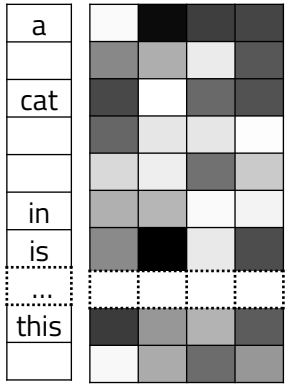
Selected Milestones



A Brief History of Computational Linguistics

Era of Neural NLP

Embedding Matrix
(v, E_{dim})



v: size of vocabulary
c: size of context window
E_{dim}: embedding dimensionality
H_{dim}: hidden layer dimensionality

Bengio et al. (2003)

A Brief History of Computational Linguistics

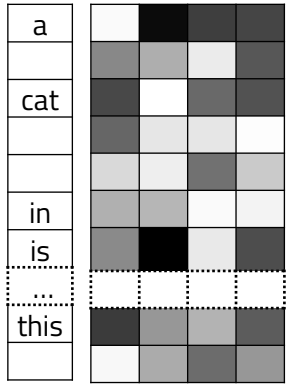
Era of Neural NLP

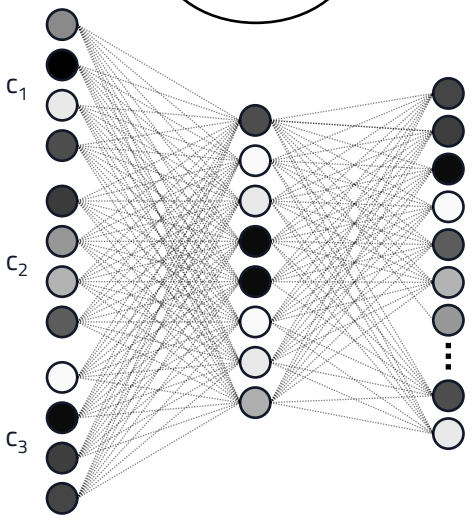
Embedding Matrix (v, E_{dim})

Input Layer $c \times E_{dim}$

Hidden Layer H_{dim}

Output Layer v





v: size of vocabulary
c: size of context window
E_{dim}: embedding dimensionality
H_{dim}: hidden layer dimensionality

Hidden layer activation

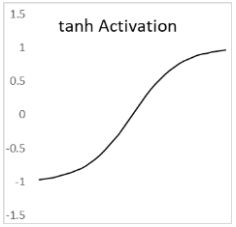
$$y = \tanh(d + Hx)$$

weight (slope): how sensitive each neuron is to each input $(c \times E_{dim}, H_{dim})$.

bias (intercept): activation threshold for each neuron (H_{dim}).

activation function: bends and curves the decision boundary.

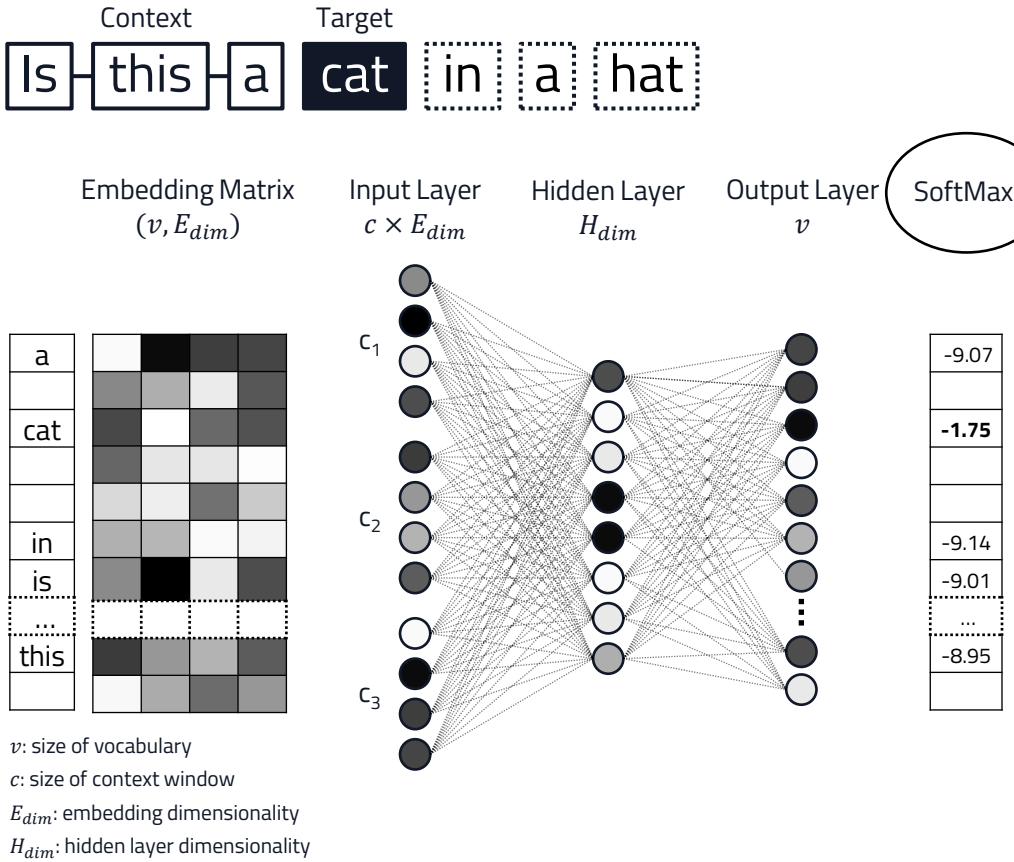
$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



Bengio et al. (2003)

A Brief History of Computational Linguistics

Era of Neural NLP



SoftMax output conversion to smooth probability distribution

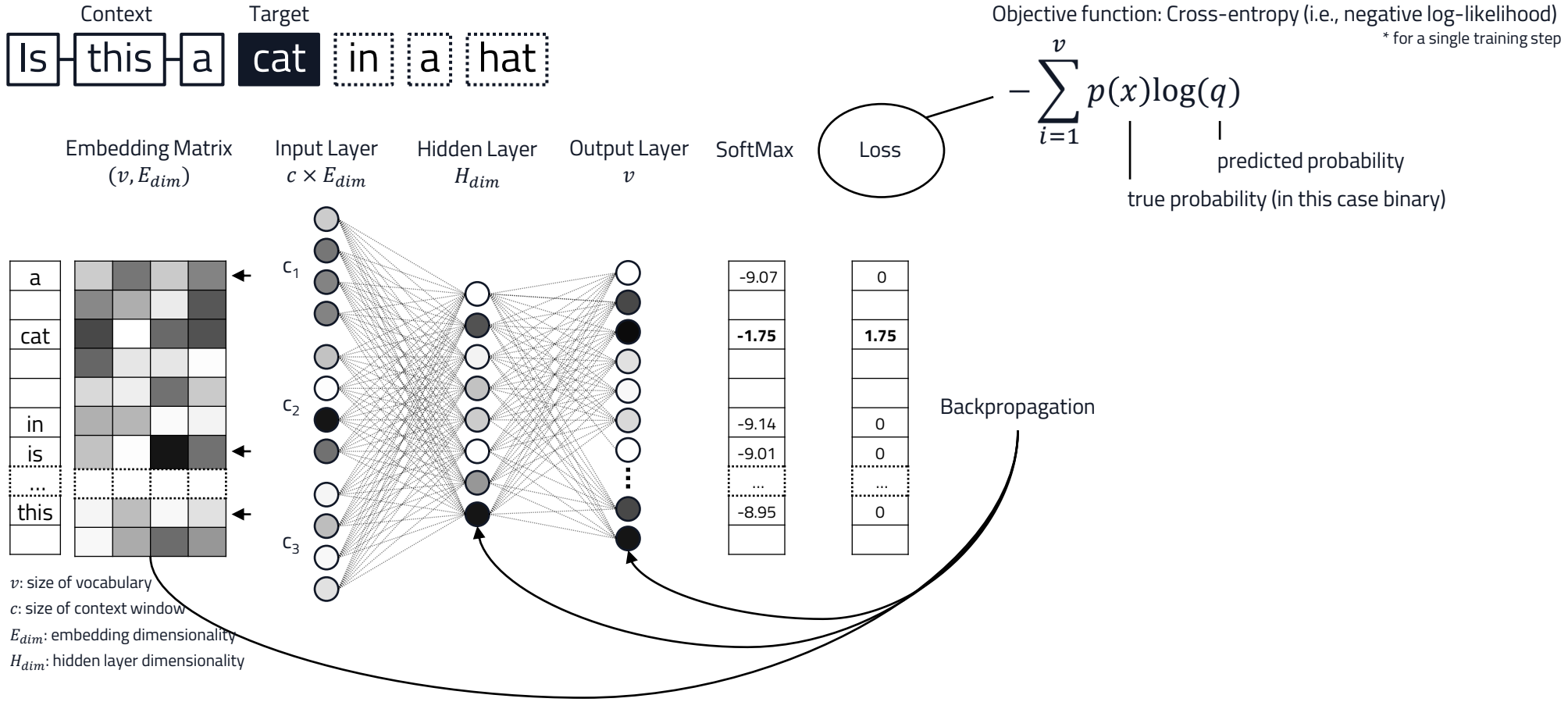
$$\frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$$

Introduction to Large Language Modeling
Department of Personality Psychology and Psychological Assessment | Wilhelm Wundt Institute of Psychology
Dr. Björn E. Hommel | bjoern.hommel@uni-leipzig.de

Bengio et al. (2003)

A Brief History of Computational Linguistics

Era of Neural NLP



Introduction to Large Language Modeling
Department of Personality Psychology and Psychological Assessment | Wilhelm Wundt Institute of Psychology
Dr. Björn E. Hommel | bjoern.hommel@uni-leipzig.de

Bengio et al. (2003)

A Brief History of Computational Linguistics

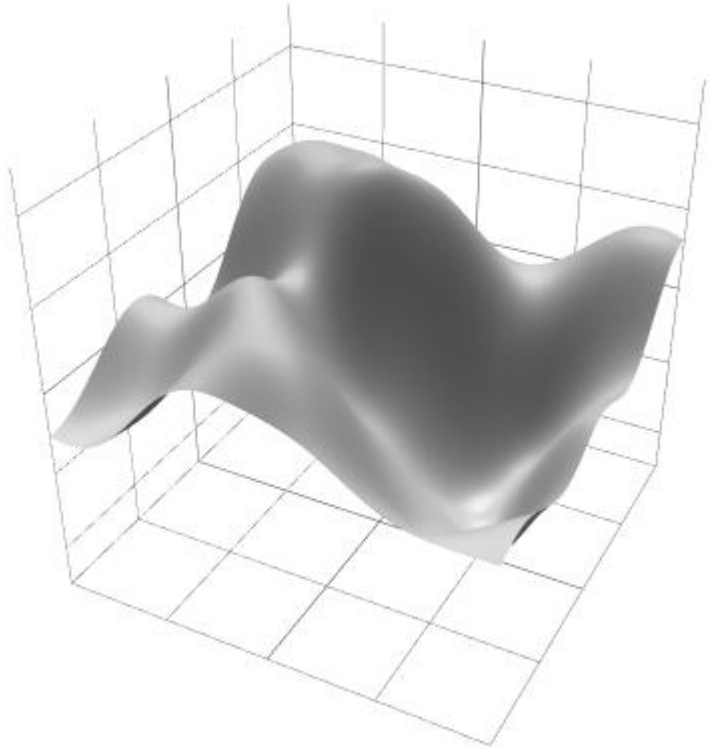
Era of Neural NLP

The goal of **backpropagation** is to update the parameters of a model so that it will minimize the error between predicted and actual targets. After calculating the loss:

1. Calculate how much each parameter contributed to the final error.
2. Layer-wise computation of gradients (slopes) for each parameter, starting from the output layer.
3. Propagate these gradient calculations backwards to the preceding layers.
4. Update the parameters with the final gradients

$$\text{new parameter} = \text{old parameter} - (\text{learning rate} \times \text{gradient})$$

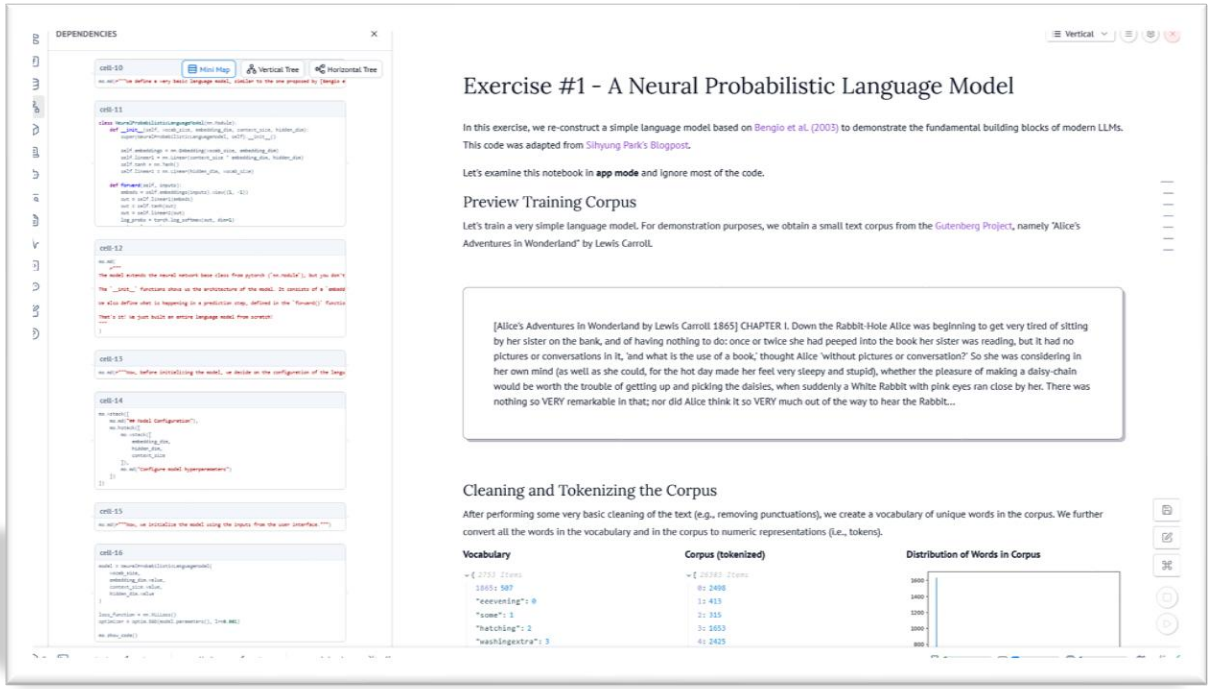
For computational efficiency, this is only done for a smaller subset of the training data (**Stochastic Gradient Descent**).



Jurafsky & Martin (2019)

Interactive Exercise

A Simple Neural Language Model




```
poetry run marimo edit ./notebooks/01-neural-language-model.py

Edit 01-neural-language-model.py in your browser 📝

➜  URL: http://localhost:2720?access_token=LM1yR3eI5HjmVeyC43fRYw
```

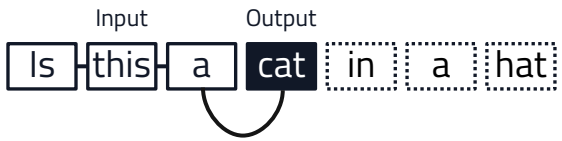
Dr. Björn E. Hommel | bjoern.hommel@uni-leipzig.de

A Brief History of Computational Linguistics

Era of Neural NLP

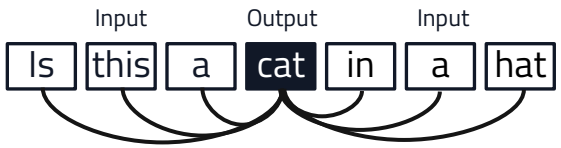
Early Word Embedding Models

Probabilistic Language Models
Learn distributions for next-word predictions

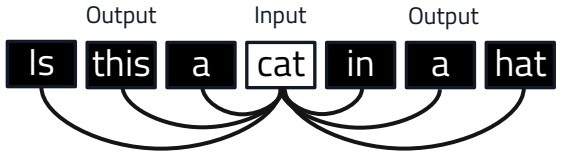


Word Embedding Models
Learn meaningful word representations
e.g., Word2Vec

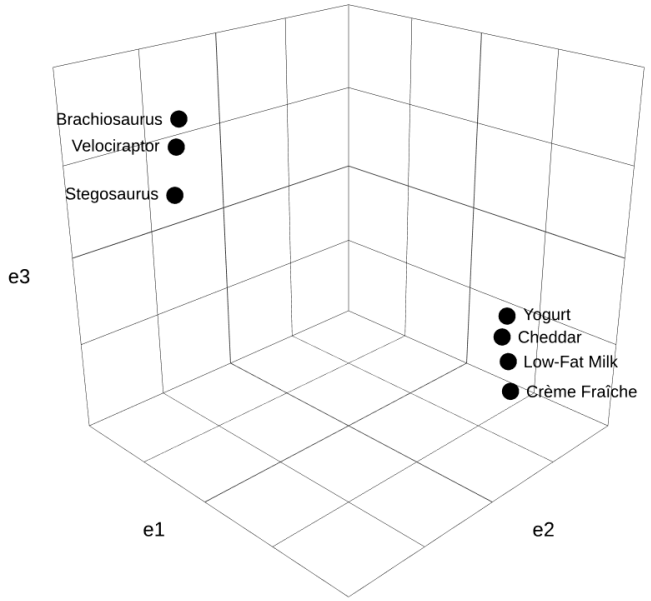
Continuous Bag of Words (CBOW)



Skip-gram



Dinosaurs & dairy products in simplified semantic space



Mikolov et al. (2013)

A Brief History of Computational Linguistics

Limitations of Models in the Pre-Transformer Era

In summary, the limitations of earlier models can

- Poor Long-Range Dependencies
- Sequential Processing Bottleneck
- Limited Contextual Understanding
- Weak Transfer Learning Capabilities

Dr. Björn E. Hommel | bjoern.hommel@uni-leipzig.de

Part II

The Transformer Model

The Transformer Model

Concepts and Architecture

- **Self-Attention**: Mechanism to assigns weights to every other token to learn context-aware representations. Captures **long-range dependencies**.

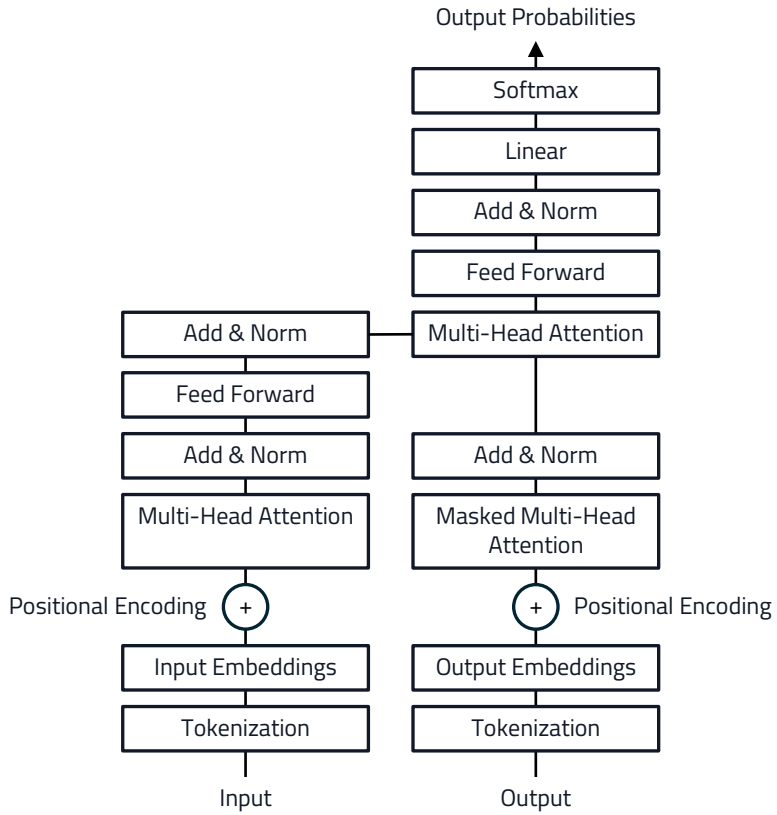
Recurrence

[...] means the power of holding two contradictory beliefs in one's mind simultaneously, and accepting both of them. The Party intellectual knows in which direction his memories must be altered; he therefore knows that he is playing tricks with reality; but by the exercise of doublethink he also satisfies himself that reality is not violated.

Attention

[...] means the power of holding two contradictory beliefs in one's mind simultaneously, and accepting both of them. The Party intellectual knows in which direction his memories must be altered; he therefore knows that he is playing tricks with reality; but by the exercise of doublethink he also satisfies himself that reality is not violated.

- **Encoder Block**: Learns rich, contextualized representation of the input sequence.
- **Decoder Block**: Iteratively generates text via next-token prediction.
- **Parallelizable**: Processes entire sequences simultaneously, reduce training time and enable massive scaling.



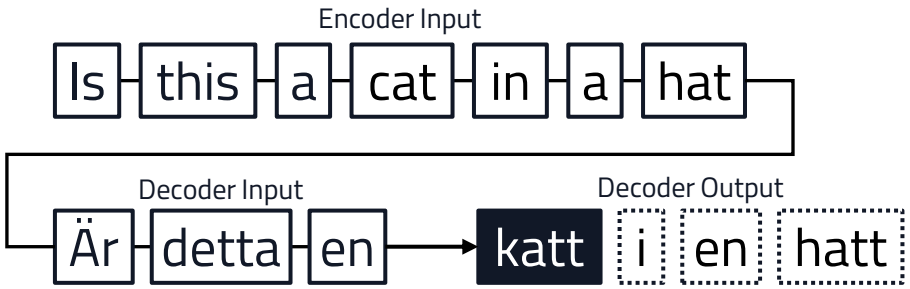
* Simplified diagram of original model by **Vaswani et al.** (2017). Omitted layers, residual connections, etc.

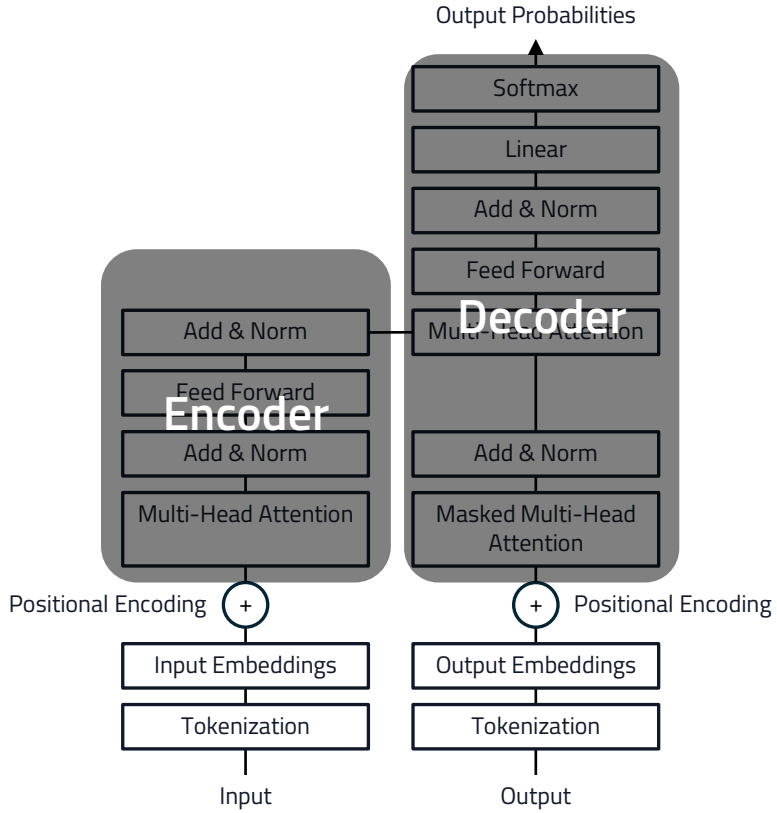
Vaswani et al. (2017)

The Transformer Model

Concepts and Architecture

- **Autoregressive Training**: Given an input sequence, the model attempts to predict the output sequence token-by-token.





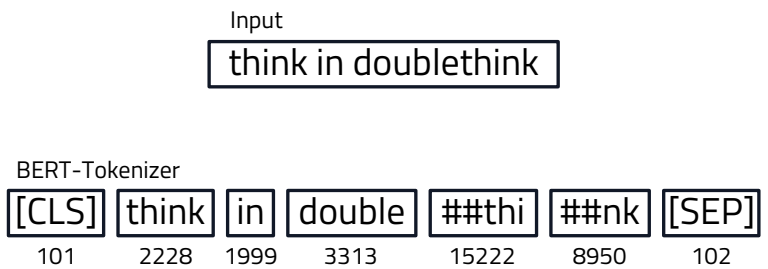
Vaswani et al. (2017)

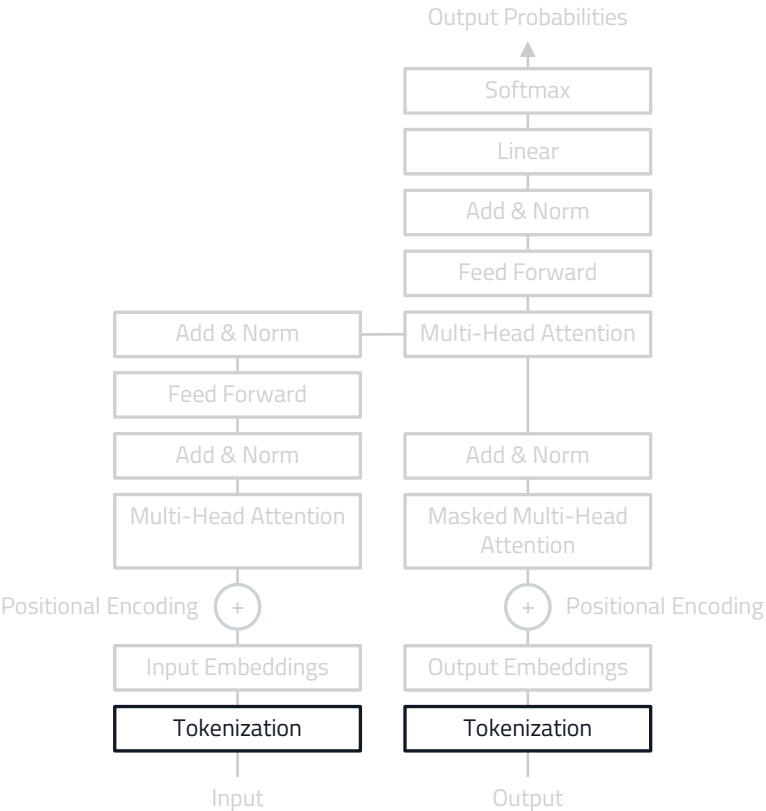
The Transformer Model

Architecture in Detail

Tokenization

- **Tokenizer**: Algorithm that splits sequences of text into atomic building blocks, represented by an numerical index.
- **Vocabulary:** The collection all tokens to represent language. May contain entire words, sub-words, punctuations, special characters and special tokens.
- **Granularity Dilemma**: Coarser tokens (e.g., words) are computationally cheaper but cannot represent the entirety of natural language (Out-of-Vocabulary errors).





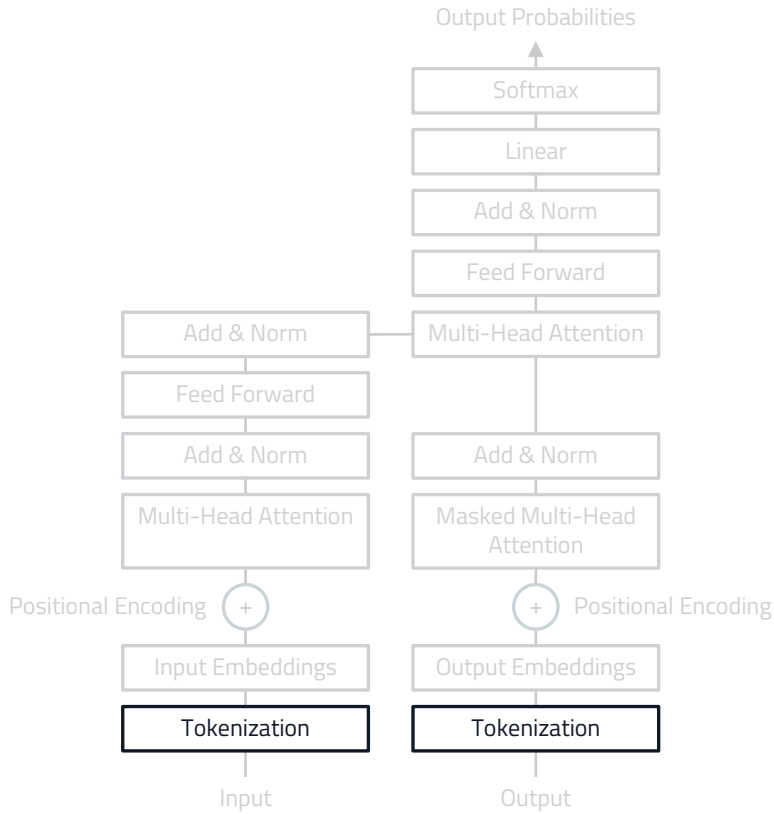
Vaswani et al. (2017)

The Transformer Model

Architecture in Detail

Special Tokens

Token	Model	Utility
[PAD]	e.g., BERT	Ensures uniform length when tokenizing a batch of multiple sequences.
[UNK]	e.g., BERT	Set for Out-of-Vocabulary inputs that cannot be constructed from other tokens.
[CLS]	e.g., BERT	First token in a seqeuence, used for classification tasks.
[SEP]	e.g., BERT	Used to separate two sequences within one input (e.g., for question-answering tasks).
[MASK]	e.g., BERT	Masked tokens at any position in the sequence will be predicted by bi-directional models.
[BOS]	e.g., Llama	Denotes the beginning of a sequence. Used for “unconditional” generation in generative models.
<|endoftext|>	e.g., GPT2	Used in text generation to predict the end of a generated text sequence.
<extra_id_1>	e.g., T5	Sentinel tokens serve as placeholders in encoder-decoder models and can be used to predict subsequences of multiple tokens.



Vaswani et al. (2017)

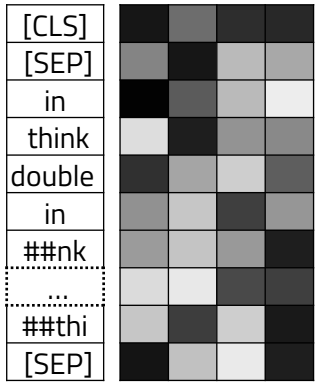
The Transformer Model

Architecture in Detail

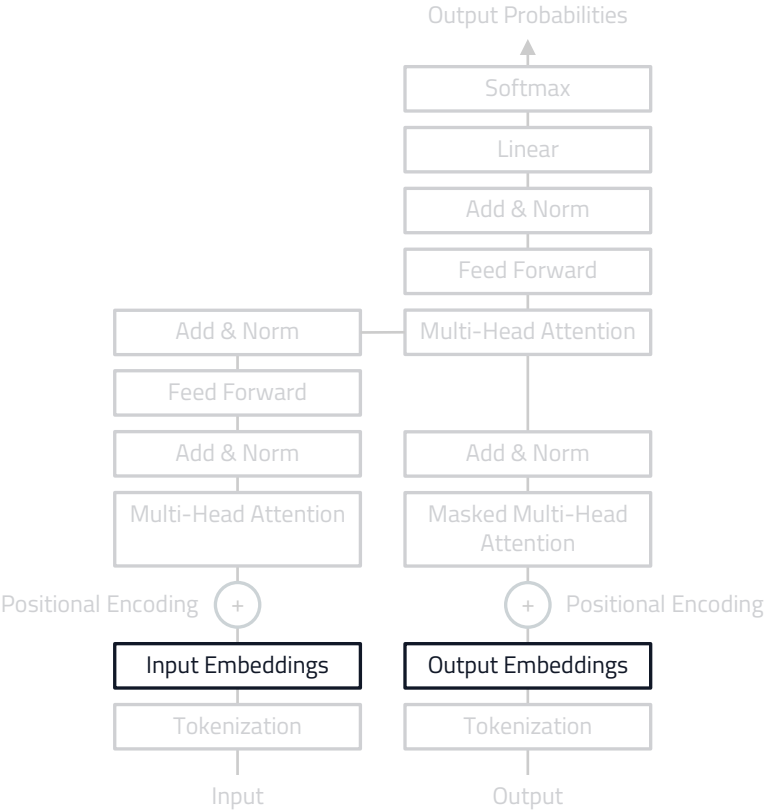
Token Embeddings

Matrices are randomly initalized from normal distribution (similar to older models, e.g., word2vec) and without meaning. Semantic representations are learned through backpropagation when the model is trained.

Embedding Matrix
(v, E_{dim})



v: size of vocabulary
E_{dim}: embedding dimensionality
c: size of context window



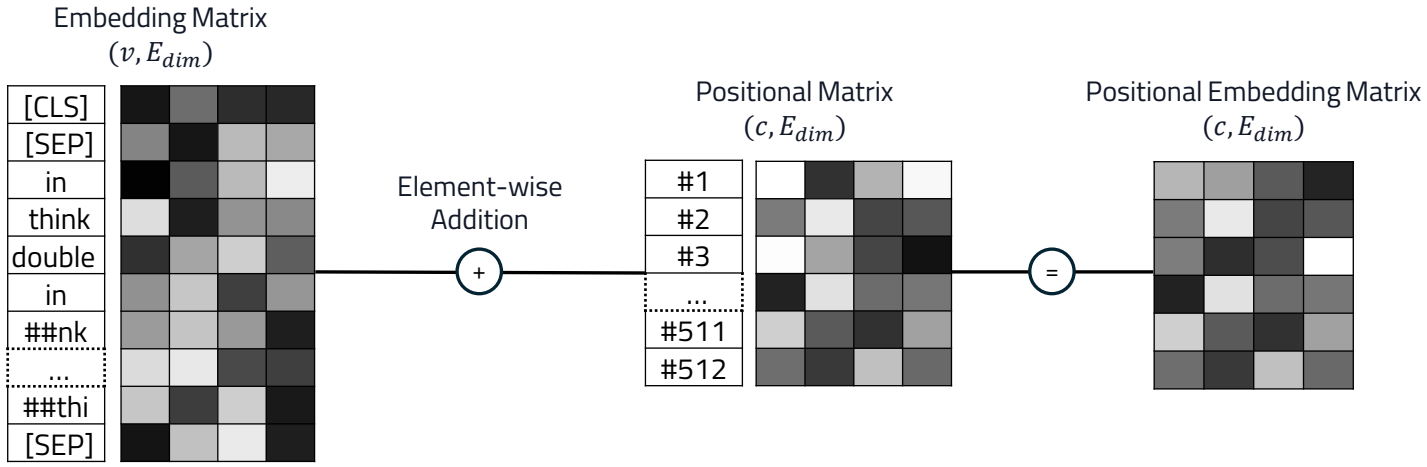
Vaswani et al. (2017)

The Transformer Model

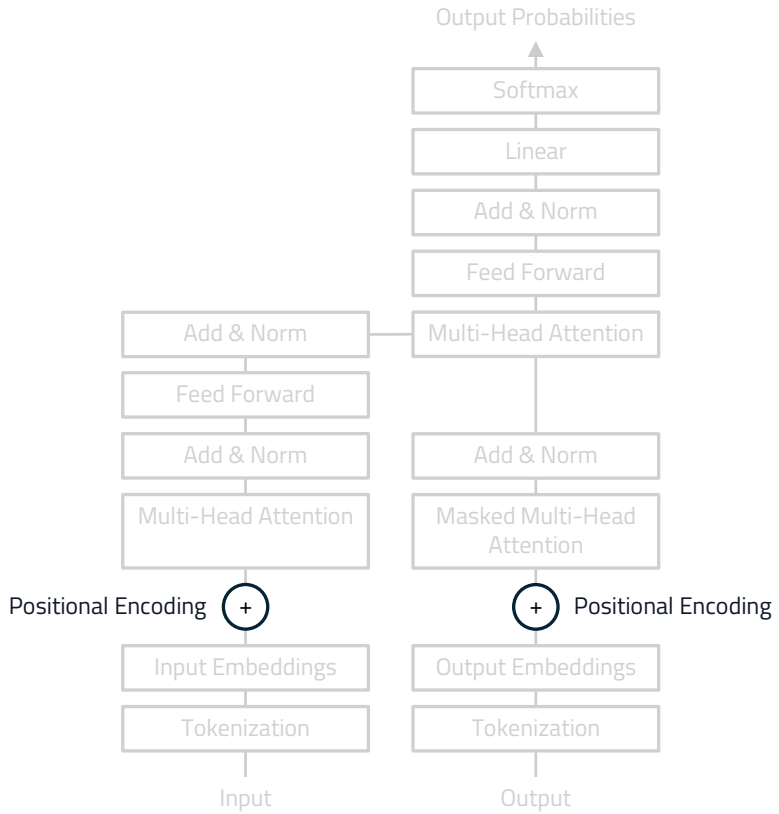
Architecture in Detail

Positional Encoding

- **Dilemma in earlier models**: Either information about the token position is lost (e.g., word2vec) or the ability to process tokens in parallel is lost (e.g., Bengio et al., 2003).
- **Positional Encoding:** Add information about the order of tokens to the embedding matrix.



v: size of vocabulary
E_{dim}: embedding dimensionality
c: size of context window

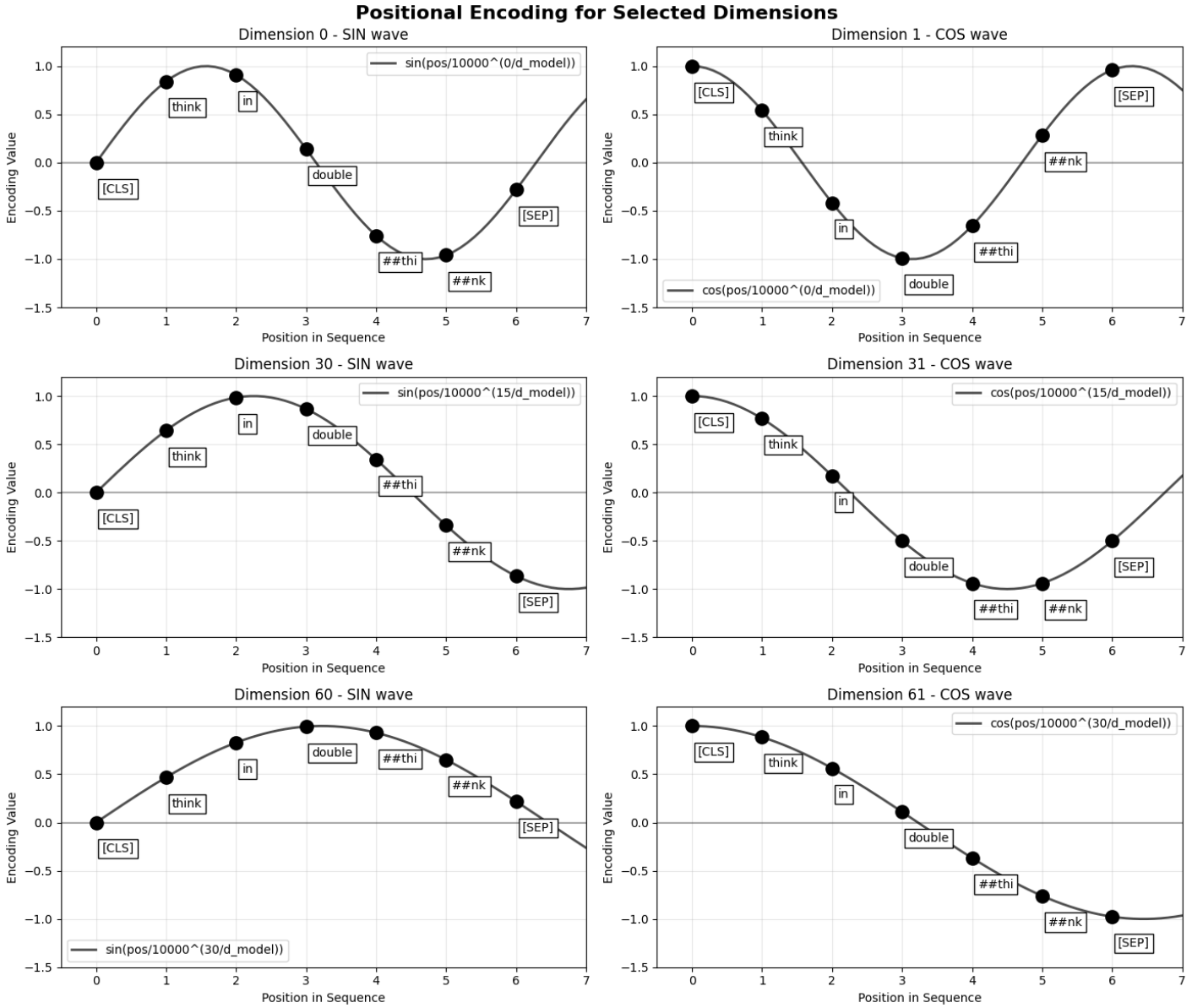


Vaswani et al. (2017)

Position values are calculated based on alternating sine and cosine-functions.

$$PE(pos, 2i) = \sin\left(\frac{pos}{10{,}000^{2i/E_{dim}}}\right)$$

$$PE(pos, 2i+1) = \cos\left(\frac{pos}{10{,}000^{2i/E_{dim}}}\right)$$



https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/

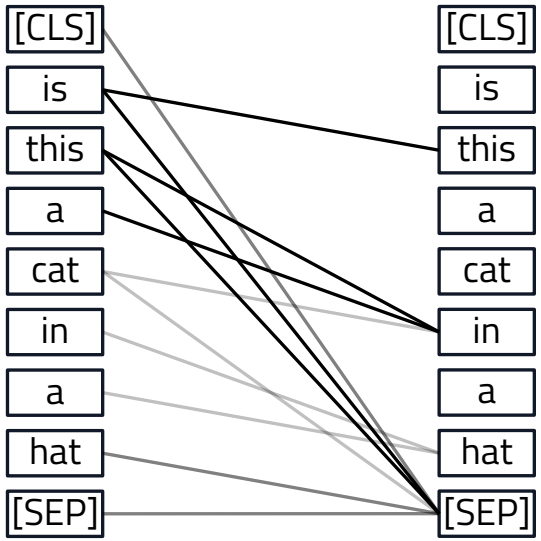
The Transformer Model

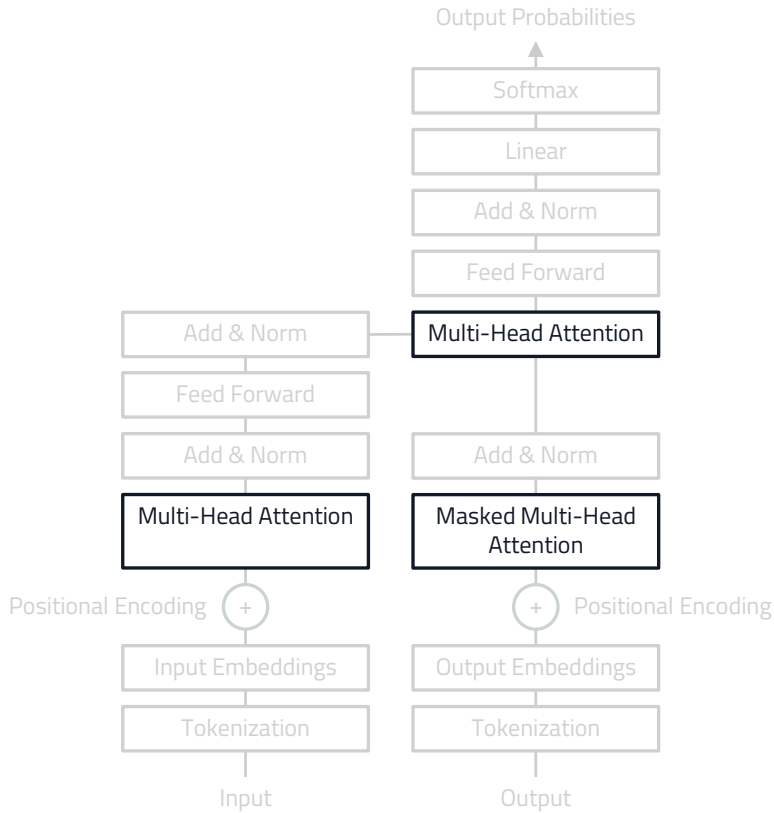
Architecture in Detail

Multi-Headed Self-Attention

Weights the importance of each token in relation to each other token.

- **"Self":** All tokens, including itself, within in the input sequence.
- "**Multi-Headed" :** Multiple attention heads in parallel, focussing on different relationships (e.g., syntax, semantics, but usually not directly interpretable).



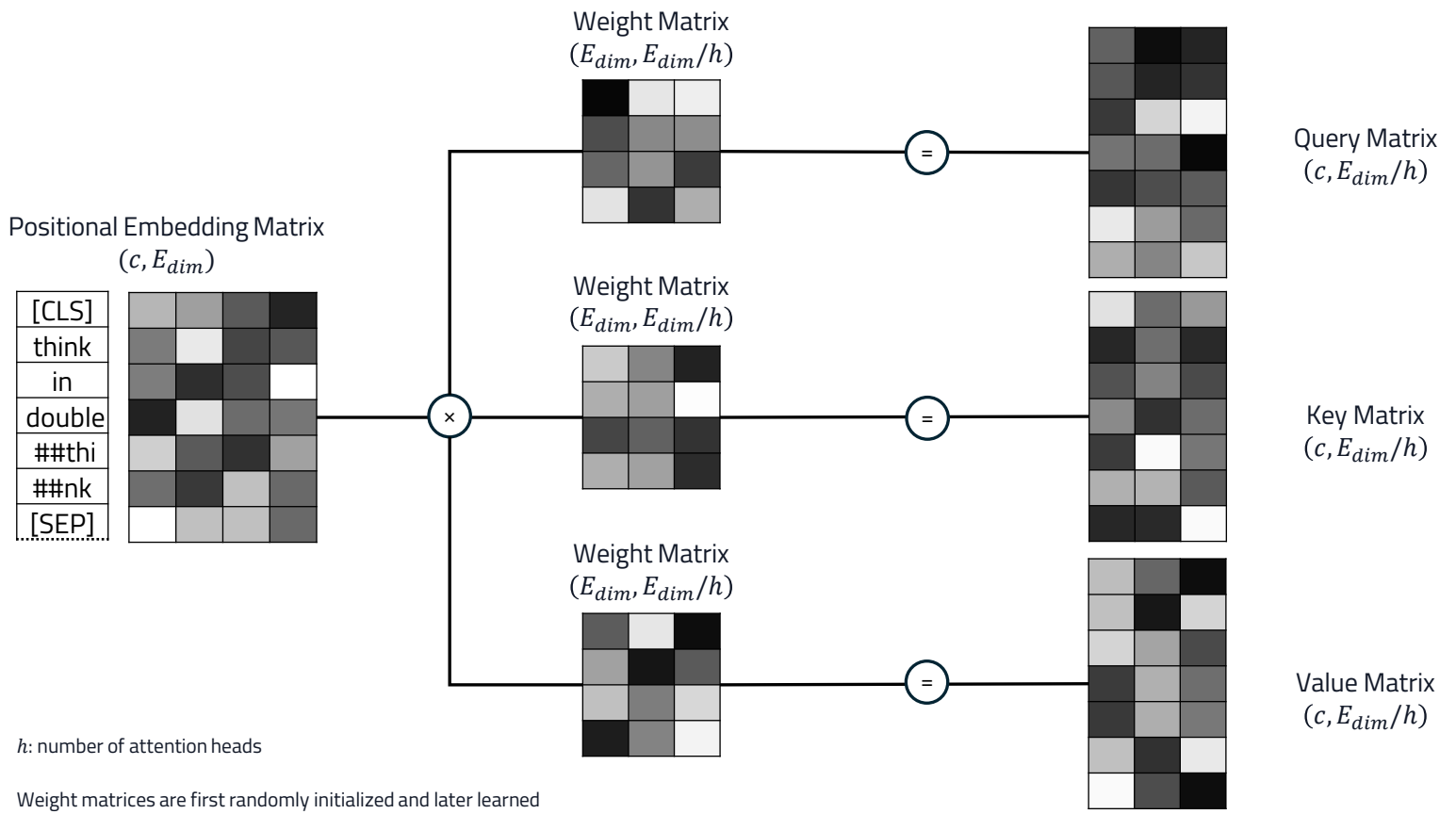


Vaswani et al. (2017)

Dr. Björn E. Hommel | bjoern.hommel@uni-leipzig.de

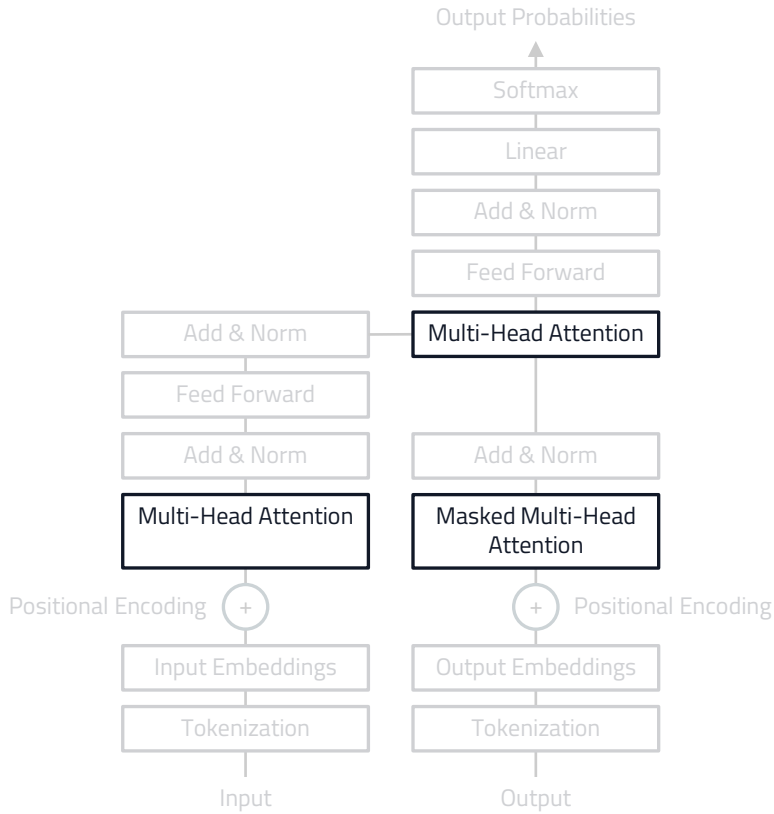
The Transformer Model

Architecture in Detail



h: number of attention heads

Weight matrices are first randomly initialized and later learned through backpropagation.



Introduction to Large Language Modeling
Department of Personality Psychology and Psychological Assessment | Wilhelm Wundt Institute of Psychology
Dr. Björn E. Hommel | bjoern.hommel@uni-leipzig.de

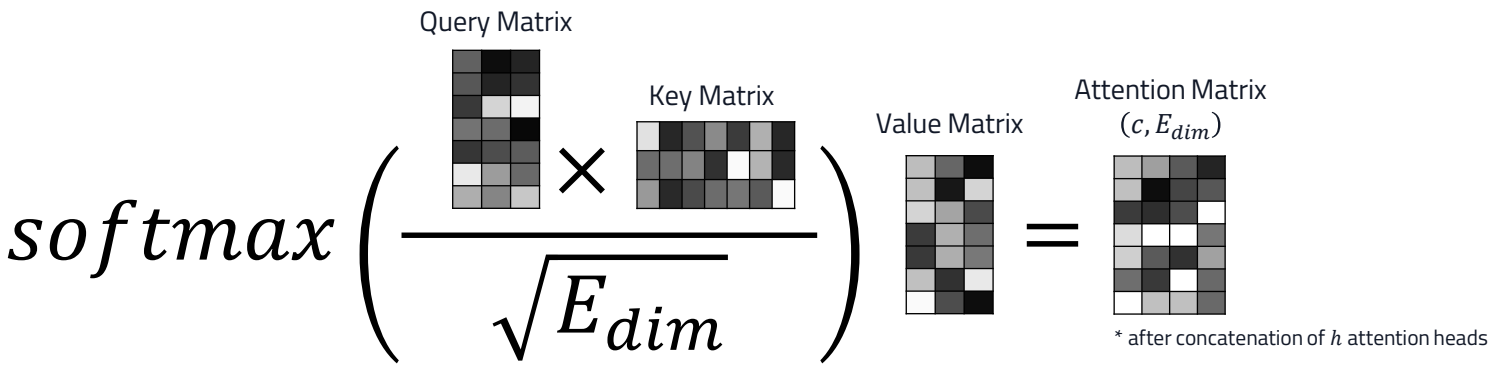
Vaswani et al. (2017)

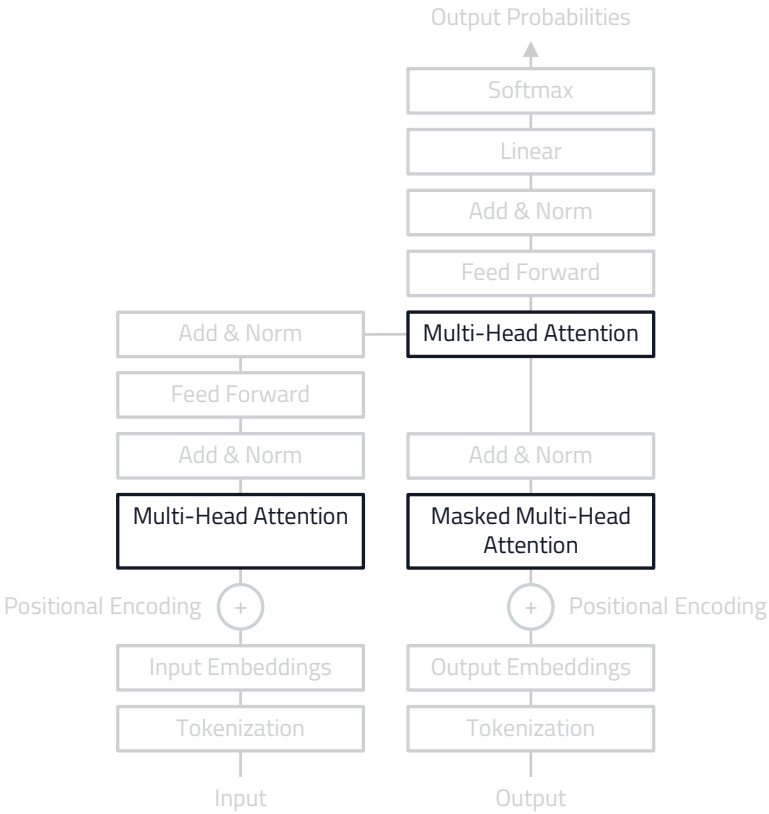
The Transformer Model

Architecture in Detail

Multi-Headed Self-Attention

Attention is calculated as shown below for each attention head, which are subsequently concatenated.





Vaswani et al. (2017)

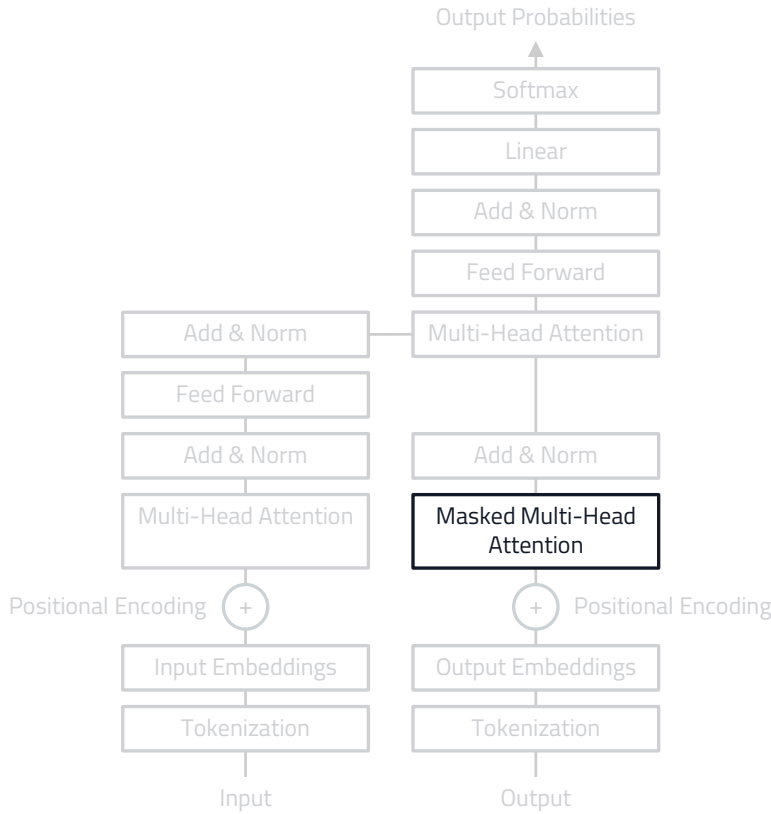
The Transformer Model

Architecture in Detail

Multi-Headed Self-Attention

Masked self-attention: As the decoder is tasked with predicting the next token, it hides future positions so each step only attends to known tokens.

	pos_1	pos_2	pos_3	pos_4	pos_5	pos_6	pos_7
$input_1$	[CLS]	think	in	double	##thi	##nk	[SEP]
$input_2$	[CLS]	think	in	double	##thi	##nk	[SEP]
$input_3$	[CLS]	think	in	double	##thi	##nk	[SEP]
$input_4$	[CLS]	think	in	double	##thi	##nk	[SEP]
$input_5$	[CLS]	think	in	double	##thi	##nk	[SEP]
$input_6$	[CLS]	think	in	double	##thi	##nk	SEP]
$input_7$	[CLS]	think	in	double	##thi	##nk	[SEP]



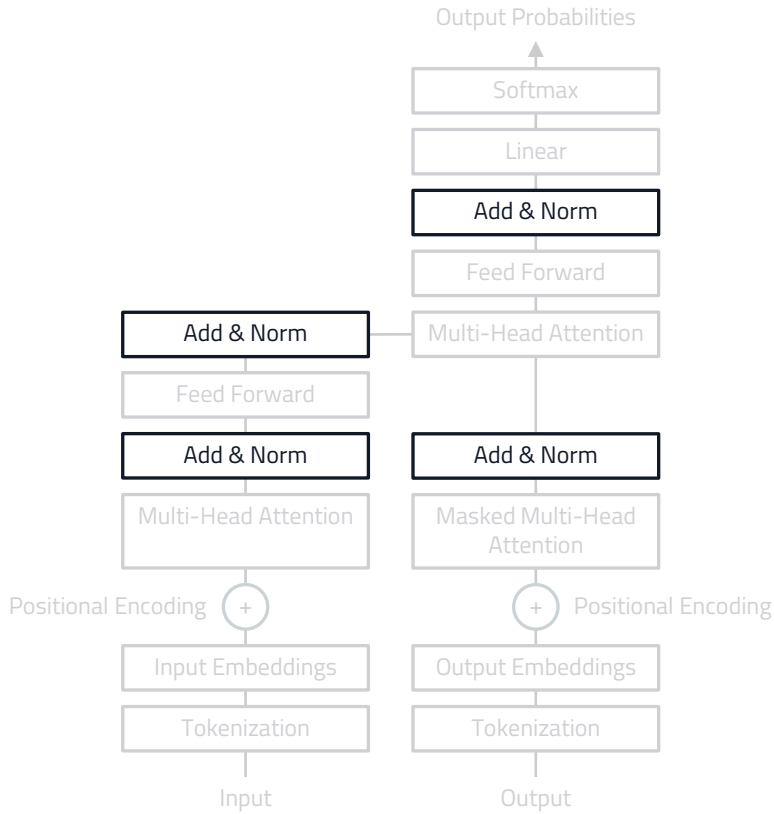
Vaswani et al. (2017)

The Transformer Model

Architecture in Detail

Residual Connections & Layer Normalization

- **“Add”**: Residual connection; adds the values from the current sub-layer (e.g., attention matrix) to the preceding output (e.g., positional embedding matrix).
- **“Norm”:** Normalization for stability (z-standardization).



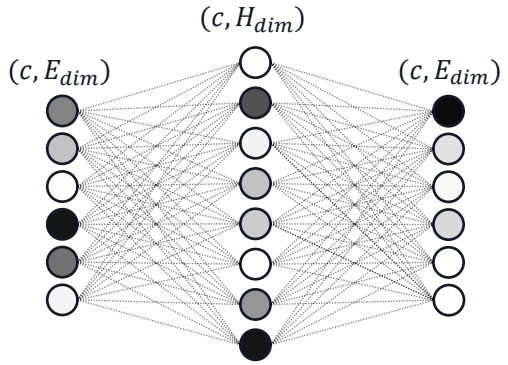
Vaswani et al. (2017)

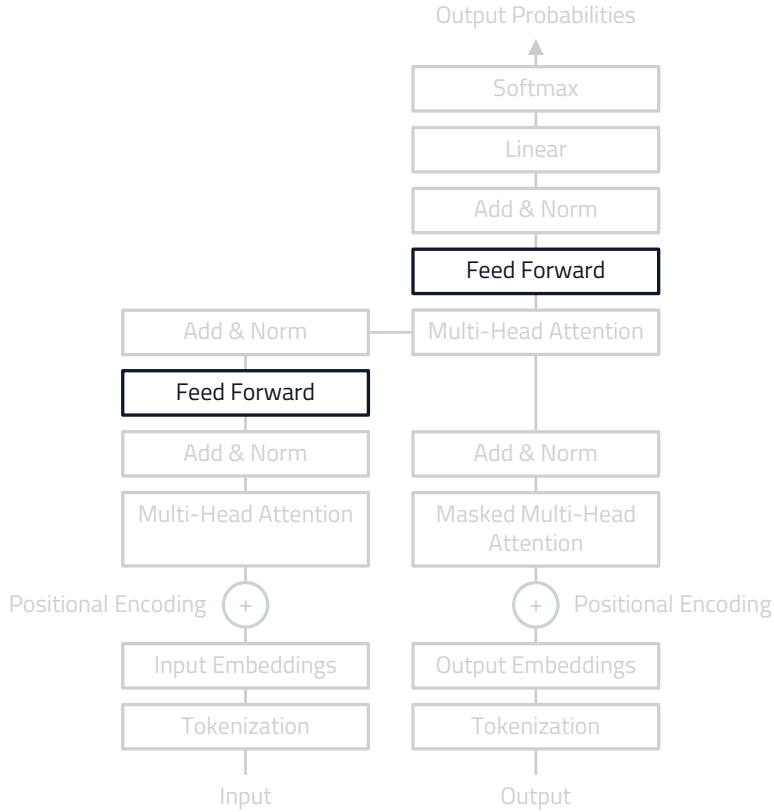
The Transformer Model

Architecture in Detail

Feed Forward

A humble, fully connected neural network. Expansion to a larger dimensionality provides more computational space, followed by compression.





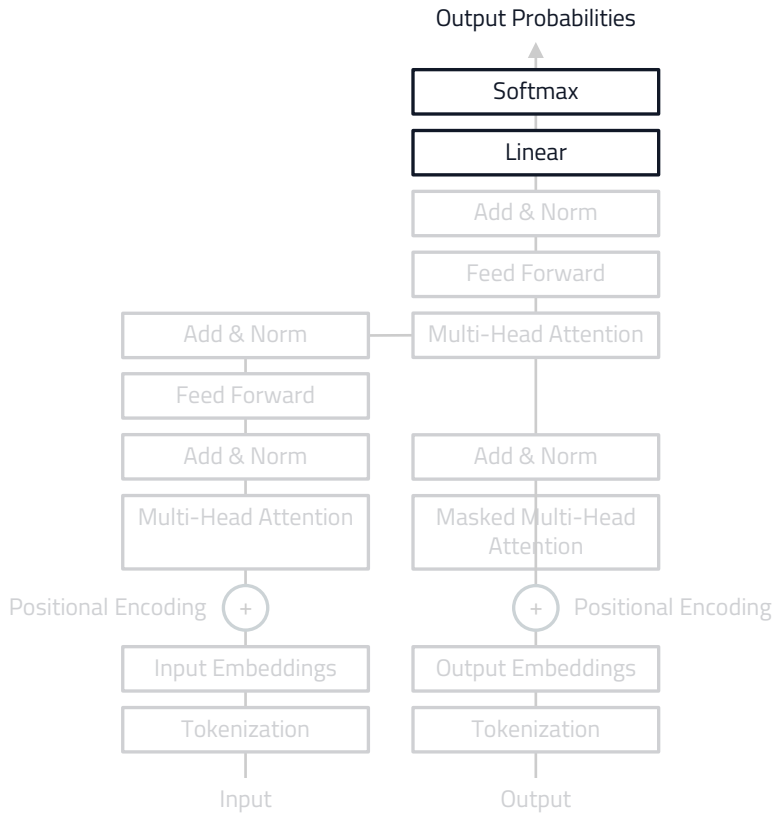
Vaswani et al. (2017)

The Transformer Model

Architecture in Detail

Output Layer and Softmax-Transformation

Finally, the last **linear layer** projects a vector of the size of the models vocabulary. Again, Softmax-transformation is used to arrive at output probabilities for the next token.



Introduction to Large Language Modeling
Department of Personality Psychology and Psychological Assessment | Wilhelm Wundt Institute of Psychology
Dr. Björn E. Hommel | bjoern.hommel@uni-leipzig.de

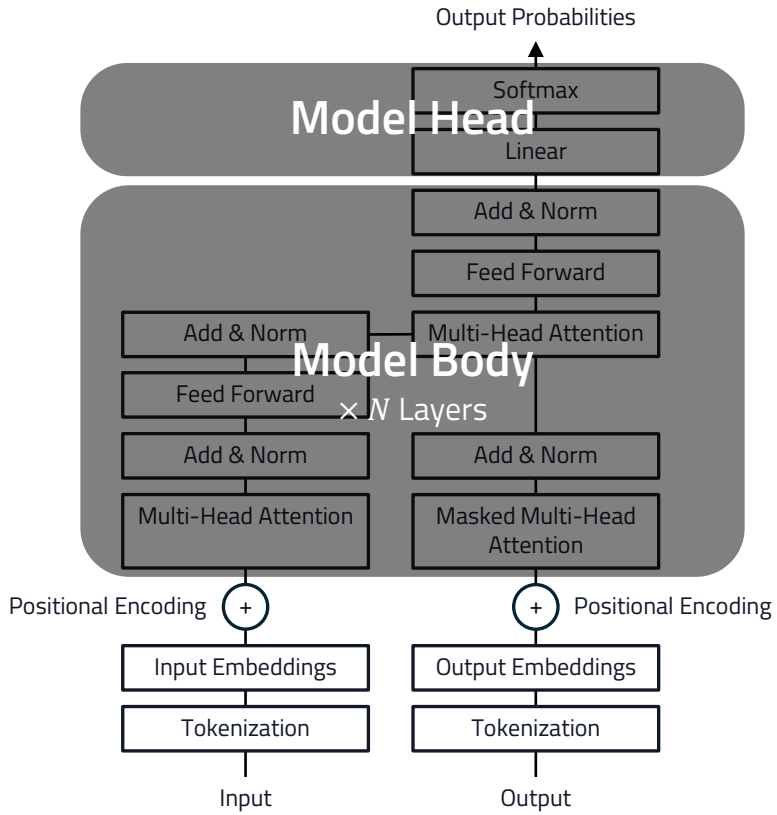
Vaswani et al. (2017)

The Transformer Model

Transfer Learning

Transfer Learning is a key concept that drives the success of Transformer models. Models first develop a foundational understanding of language and general knowledge, then can be easily adapted for more specific tasks.

- **Pretraining**: Describes the process of initially training a model, which usually requires vast resources in terms of computing power and data.
- Further Training often involves domain adaptation and/or fine-tuning, although these terms are sometimes used interchangeably:
  - **Domain Adaptation**: Continue training a previously pretrained model on a smaller, but more domain-specific dataset.
  - **Fine-tuning:** Continue training a previously pretrained model on a different task, which oftentimes involves slight architectural modifications (e.g., retaining the **model body**, but replacing the **model head**).



Vaswani et al. (2017)

Dr. Björn E. Hommel | bjoern.hommel@uni-leipzig.de

The Transformer Model

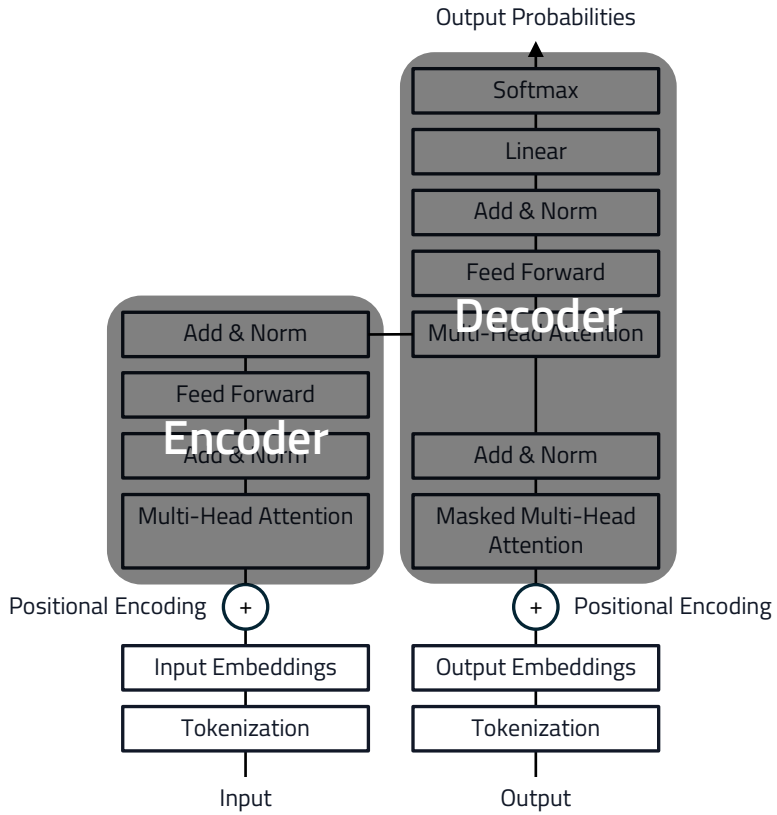
Specialized Blocks

As the two main blocks of the Transformer model serve different purposes, they can be used as independent models for specialized tasks.

We can therefore distinguish between three basic model types:

- **Encoder-Decoder Models**: Utilize the full architecture, as originally proposed.
- **Encoder-Only Models**: Predominantly used for **Natural Language Understanding (NLU)**.
- **Decoder-Only Models**: Predominantly used for **Natural Language Generation (NLG).**

Each detached block can of course be subjected to further fine-tuning and/or domain adaptation.



Vaswani et al. (2017)

The Transformer Model

Many sizes, many flavours

	Prominent Models	Typical Tasks
Encoder-Only Models (or "bi-directional" models)	▪ BERT ▪ RoBERTa ▪ DistilBERT ▪ ModernBERT	▪ Text Classification ▪ Named Entity Recognition ▪ Question Answering ▪ Feature Extraction ▪ Sentence Similarity ▪ Natural Language Inference
Decoder-Only Models (also "causal" or "generative" models)	▪ GPTs (e.g., GPT-5) ▪ Claude ▪ DeepSeek ▪ Mistral ▪ Llama ▪ Qwen	▪ Text Generation ▪ In-Context-Learning ▪ …and everything else?
Encoder-Decoder Models (or "sequence-to-sequence" models)	▪ T5 ▪ BART ▪ T5Gemma	▪ Machine Translation ▪ Text Summarization ▪ Paraphrasing / Simplification ▪ Question Answering

The Transformer Model

Many sizes, many flavours

Get to know your model

Model Scale & Size	▪ Number of Parameters ▪ Number of Layers/Blocks ▪ Embedding Dimensionality ▪ Context Window Size ▪ Type of Architecture (Encoder, Decoder, etc.)
Tokenizer	▪ Tokenizer Type ▪ Vocabulary Size ▪ Special Tokens
Training	▪ Training Dataset ▪ Training Objectives ▪ Training Stages ▪ Languages
Performance	▪ Benchmark Scores / Leaderboards ▪ Accuracy Metrics
Availability	▪ License Type

Dr. Björn E. Hommel | bjoern.hommel@uni-leipzig.de

Part III

The Hugging Face Ecosystem

The Hugging Face Ecosystem

One-Stop Shop for Open Source LLM-Resources

https://huggingface.co/

Python Modules

Most prominently the *transformers* library, but also *datasets, diffusers, etc.*

Docs & Community

Extensive learning resources, tutorials, documentations and forums.

Model Hub

Currently over 2 mio. freely accessible models. Offers free model hosting and API endpoints.

Spaces

Hosts web applications intended for small scale demos of machine learning applications.

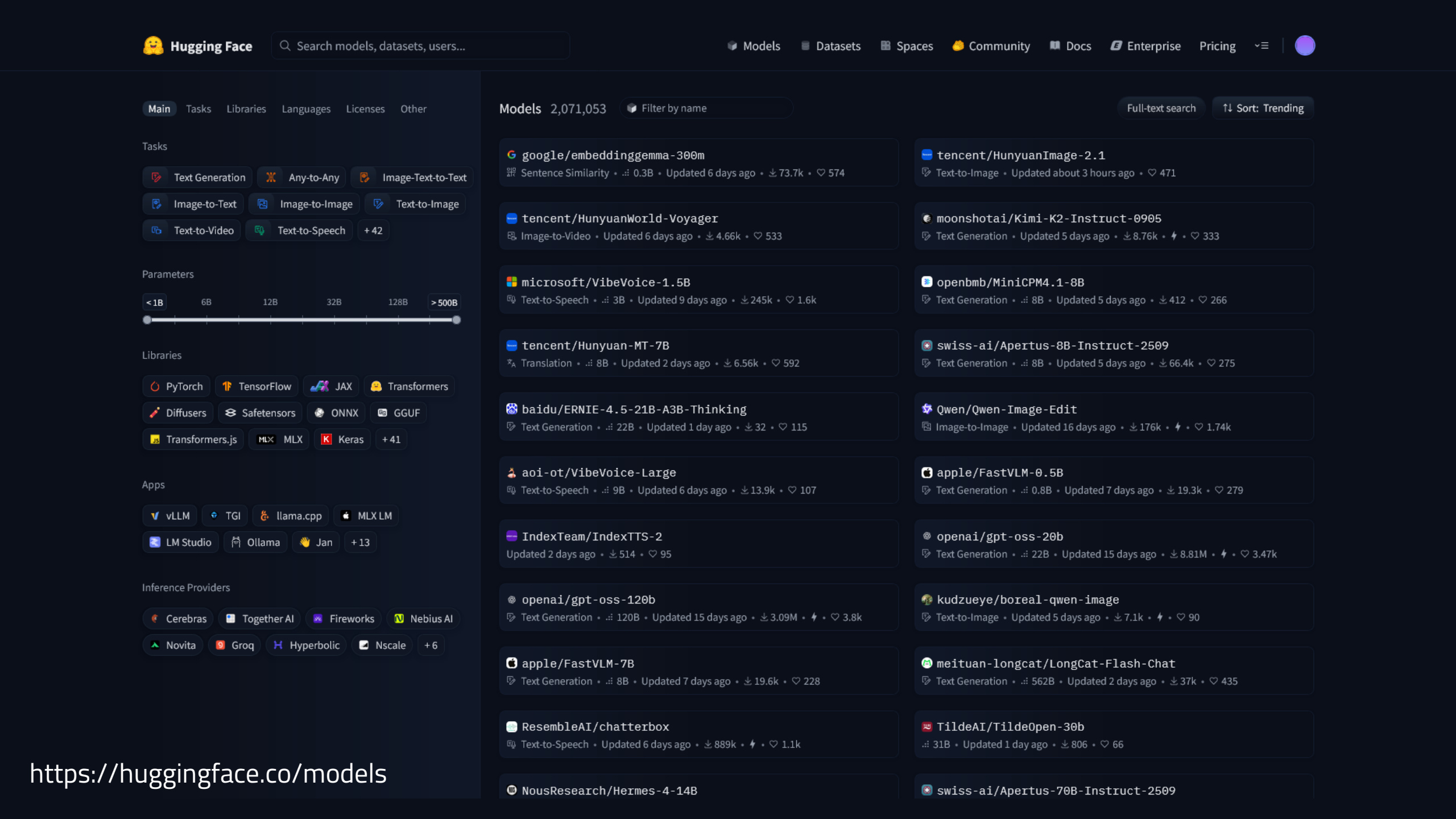
Datasets

Nearly 500k ready-to-use datasets, e.g., for training AI models.

The Hugging Face Ecosystem

Getting Started

```
pip install transformers      # the main package, to start working with transformers
pip install datasets          # to access and manipulate datasets
pip install huggingface_hub   # to interact with the HF hub (e.g., upload models)
pip install evaluate          # to calculate the usual transformer performance metrics


# not stricly part of the HF suite, but useful to work with sentence embeddings
pip install sentence-transformers
```



https://huggingface.co/models

Main | Tasks | Libraries | Languages | Licenses | Other

Filter Tasks by name

Multimodal

- Audio-Text-to-Text
- Image-Text-to-Text
- Visual Question Answering
- Document Question Answering
- Video-Text-to-Text
- Visual Document Retrieval
- Any-to-Any

Computer Vision

- Depth Estimation
- Image Classification
- Object Detection
- Image Segmentation
- Text-to-Image
- Image-to-Text
- Image-to-Image
- Image-to-Video
- Unconditional Image Generation
- Video Classification
- Text-to-Video
- Zero-Shot Image Classification
- Mask Generation
- Zero-Shot Object Detection
- Text-to-3D
- Image-to-3D
- Image Feature Extraction
- Keypoint Detection
- Video-to-Video

Natural Language Processing

- Text Classification
- Token Classification
- Table Question Answering
- Question Answering
- Zero-Shot Classification
- Translation
- Summarization
- Feature Extraction
- Text Generation
- Fill-Mask
- Sentence Similarity
- Text Ranking

Audio

Models 2,071,053

Filter by name | Full-text search | Sort: Trending

- google/embeddinggemma-300m — Sentence Similarity • 0.3B • Updated 6 days ago • 73.7k • 574
- tencent/HunyuanImage-2.1 — Text-to-Image • Updated about 3 hours ago • 471
- tencent/HunyuanWorld-Voyager — Image-to-Video • Updated 6 days ago • 4.66k • 533
- moonshotai/Kimi-K2-Instruct-0905 — Text Generation • Updated 5 days ago • 8.76k • 333
- microsoft/VibeVoice-1.5B — Text-to-Speech • 3B • Updated 9 days ago • 245k • 1.6k
- openbmb/MiniCPM4.1-8B — Text Generation • 8B • Updated 5 days ago • 412 • 266
- tencent/Hunyuan-MT-7B — Translation • 8B • Updated 2 days ago • 6.56k • 592
- swiss-ai/Apertus-8B-Instruct-2509 — Text Generation • 8B • Updated 5 days ago • 66.4k • 275
- baidu/ERNIE-4.5-21B-A3B-Thinking — Text Generation • 22B • Updated 1 day ago • 32 • 115
- Qwen/Qwen-Image-Edit — Image-to-Image • Updated 16 days ago • 176k • 1.74k
- aoi-ot/VibeVoice-Large — Text-to-Speech • 9B • Updated 6 days ago • 13.9k • 107
- apple/FastVLM-0.5B — Text Generation • 0.8B • Updated 7 days ago • 19.3k • 279
- IndexTeam/IndexTTS-2 — Updated 2 days ago • 514 • 95
- openai/gpt-oss-20b — Text Generation • 22B • Updated 15 days ago • 8.81M • 3.47k
- openai/gpt-oss-120b — Text Generation • 120B • Updated 15 days ago • 3.09M • 3.8k
- kudzueye/boreal-qwen-image — Text-to-Image • Updated 5 days ago • 7.1k • 90
- apple/FastVLM-7B — Text Generation • 8B • Updated 7 days ago • 19.6k • 228
- meituan-longcat/LongCat-Flash-Chat — Text Generation • 562B • Updated 2 days ago • 37k • 435
- ResembleAI/chatterbox — Text-to-Speech • Updated 6 days ago • 889k • 1.1k
- TildeAI/TildeOpen-30b — 31B • Updated 1 day ago • 806 • 66
- NousResearch/Hermes-4-14B
- swiss-ai/Apertus-70B-Instruct-2509

Main | Tasks | Libraries | Languages | Licenses | Other

Filter Languages by name

English | Chinese | French | Spanish
German | Japanese | Korean | Italian
Portuguese | Russian | Hindi | Arabic
Thai | multilingual | Indonesian | Turkish
Vietnamese | Polish | Dutch | Romanian
Swedish | Ukrainian | Persian | Czech
Finnish | Bengali | Nepali | Malay
Hebrew | Greek | Danish | Tamil | French
Hungarian | Catalan | Urdu | Bulgarian
Serbian | Telugu | Norwegian | Marathi
Swahili | Estonian | Gujarati | Malayalam
Slovenian | Burmese | Slovak | Lithuanian
Croatian | Galician | Kannada | Tagalog
Latvian | Khmer | Panjabi | Basque | Lao
Icelandic | Amharic | Afrikaans | Assamese
English | Georgian | Mongolian | Hausa
Welsh | Macedonian | Javanese | Sinhala
Kazakh | Sundanese | Armenian | Belarusian
Irish | Azerbaijani | Uzbek | Albanian
Yoruba | Thai | Arabic | Chinese | Spanish
Russian | Oriya | Somali | Japanese
German | Vietnamese | Portuguese
+ 4748 languages

Models 2,071,053

Filter by name | Full-text search | Sort: Trending

google/embeddinggemma-300m
Sentence Similarity • 0.3B • Updated 6 days ago • 73.7k • 574

tencent/HunyuanWorld-Voyager
Image-to-Video • Updated 6 days ago • 4.66k • 533

microsoft/VibeVoice-1.5B
Text-to-Speech • 3B • Updated 9 days ago • 245k • 1.6k

tencent/Hunyuan-MT-7B
Translation • 8B • Updated 2 days ago • 6.56k • 592

baidu/ERNIE-4.5-21B-A3B-Thinking
Text Generation • 22B • Updated 1 day ago • 32 • 115

aoi-ot/VibeVoice-Large
Text-to-Speech • 9B • Updated 6 days ago • 13.9k • 107

IndexTeam/IndexTTS-2
Updated 2 days ago • 514 • 95

openai/gpt-oss-120b
Text Generation • 120B • Updated 15 days ago • 3.09M • 3.8k

apple/FastVLM-7B
Text Generation • 8B • Updated 7 days ago • 19.6k • 228

ResembleAI/chatterbox
Text-to-Speech • Updated 6 days ago • 889k • 1.1k

NousResearch/Hermes-4-14B

tencent/HunyuanImage-2.1
Text-to-Image • Updated about 3 hours ago • 471

moonshotai/Kimi-K2-Instruct-0905
Text Generation • Updated 5 days ago • 8.76k • 333

openbmb/MiniCPM4.1-8B
Text Generation • 8B • Updated 5 days ago • 412 • 266

swiss-ai/Apertus-8B-Instruct-2509
Text Generation • 8B • Updated 5 days ago • 66.4k • 275

Qwen/Qwen-Image-Edit
Image-to-Image • Updated 16 days ago • 176k • 1.74k

apple/FastVLM-0.5B
Text Generation • 0.8B • Updated 7 days ago • 19.3k • 279

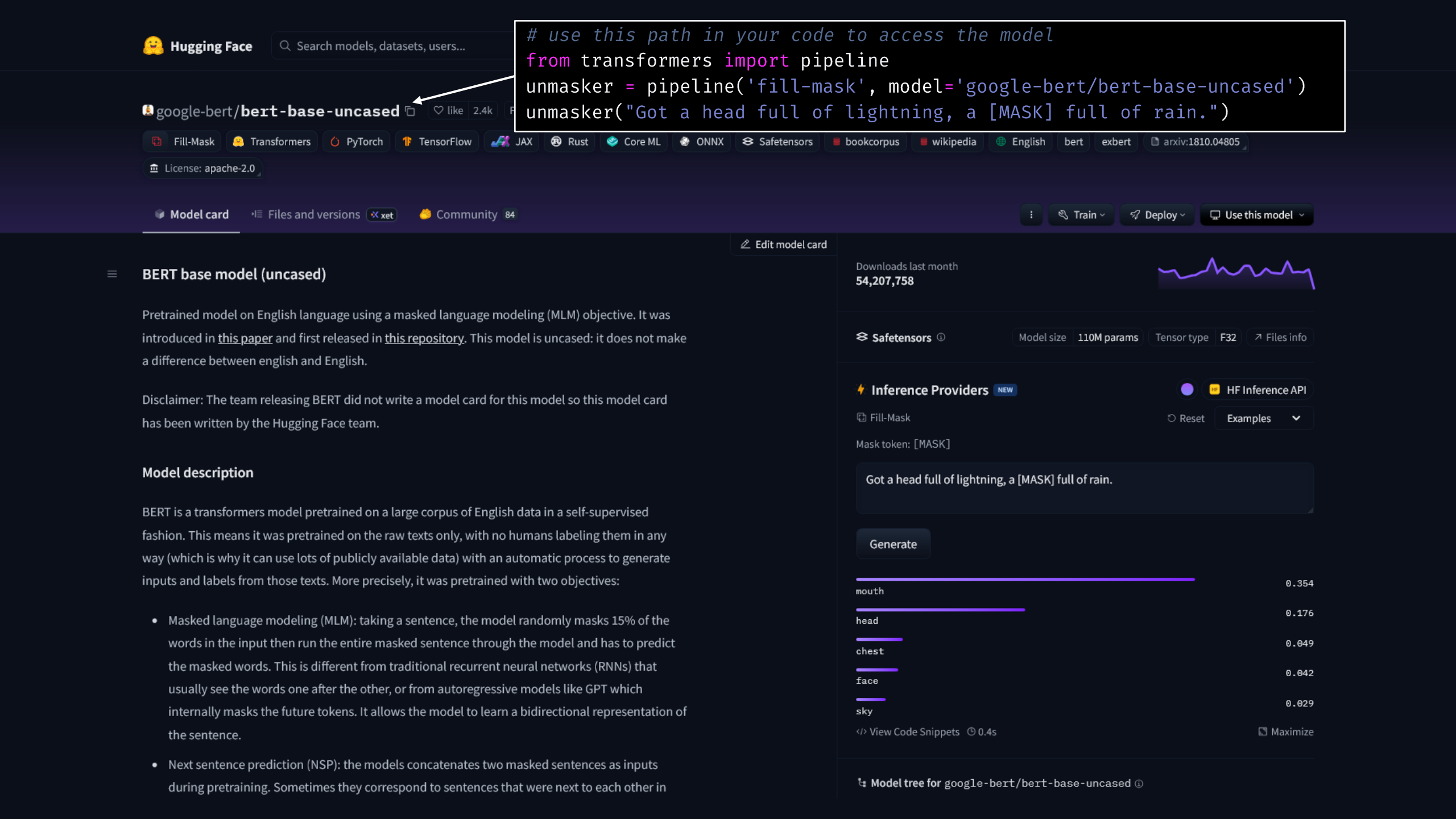
openai/gpt-oss-20b
Text Generation • 22B • Updated 15 days ago • 8.81M • 3.47k

kudzueye/boreal-qwen-image
Text-to-Image • Updated 5 days ago • 7.1k • 90

meituan-longcat/LongCat-Flash-Chat
Text Generation • 562B • Updated 2 days ago • 37k • 435

TildeAI/TildeOpen-30b
31B • Updated 1 day ago • 806 • 66

swiss-ai/Apertus-70B-Instruct-2509


Hugging Face
Search models, datasets, users...
# use this path in your code to access the model
from transformers import pipeline
unmasker = pipeline('fill-mask', model='google-bert/bert-base-uncased')
unmasker("Got a head full of lightning, a [MASK] full of rain.")
google-bert/bert-base-uncased
like 2.4k
Fill-Mask
Transformers
PyTorch
TensorFlow
JAX
Rust
Core ML
ONNX
Safetensors
bookcorpus
wikipedia
English
bert
exbert
arxiv:1810.04805
License: apache-2.0
Model card
Files and versions
xet
Community 84
Train
Deploy
Use this model
Edit model card
BERT base model (uncased)
Pretrained model on English language using a masked language modeling (MLM) objective. It was introduced in this paper and first released in this repository. This model is uncased: it does not make a difference between english and English.
Disclaimer: The team releasing BERT did not write a model card for this model so this model card has been written by the Hugging Face team.
Model description
BERT is a transformers model pretrained on a large corpus of English data in a self-supervised fashion. This means it was pretrained on the raw texts only, with no humans labeling them in any way (which is why it can use lots of publicly available data) with an automatic process to generate inputs and labels from those texts. More precisely, it was pretrained with two objectives:
Masked language modeling (MLM): taking a sentence, the model randomly masks 15% of the words in the input then run the entire masked sentence through the model and has to predict the masked words. This is different from traditional recurrent neural networks (RNNs) that usually see the words one after the other, or from autoregressive models like GPT which internally masks the future tokens. It allows the model to learn a bidirectional representation of the sentence.
Next sentence prediction (NSP): the models concatenates two masked sentences as inputs during pretraining. Sometimes they correspond to sentences that were next to each other in
Downloads last month
54,207,758
Safetensors
Model size 110M params
Tensor type F32
Files info
Inference Providers NEW
HF Inference API
Fill-Mask
Reset
Examples
Mask token: [MASK]
Got a head full of lightning, a [MASK] full of rain.
Generate
mouth 0.354
head 0.176
chest 0.049
face 0.042
sky 0.029
View Code Snippets
0.4s
Maximize
Model tree for google-bert/bert-base-uncased

google-bert/bert-base-uncased

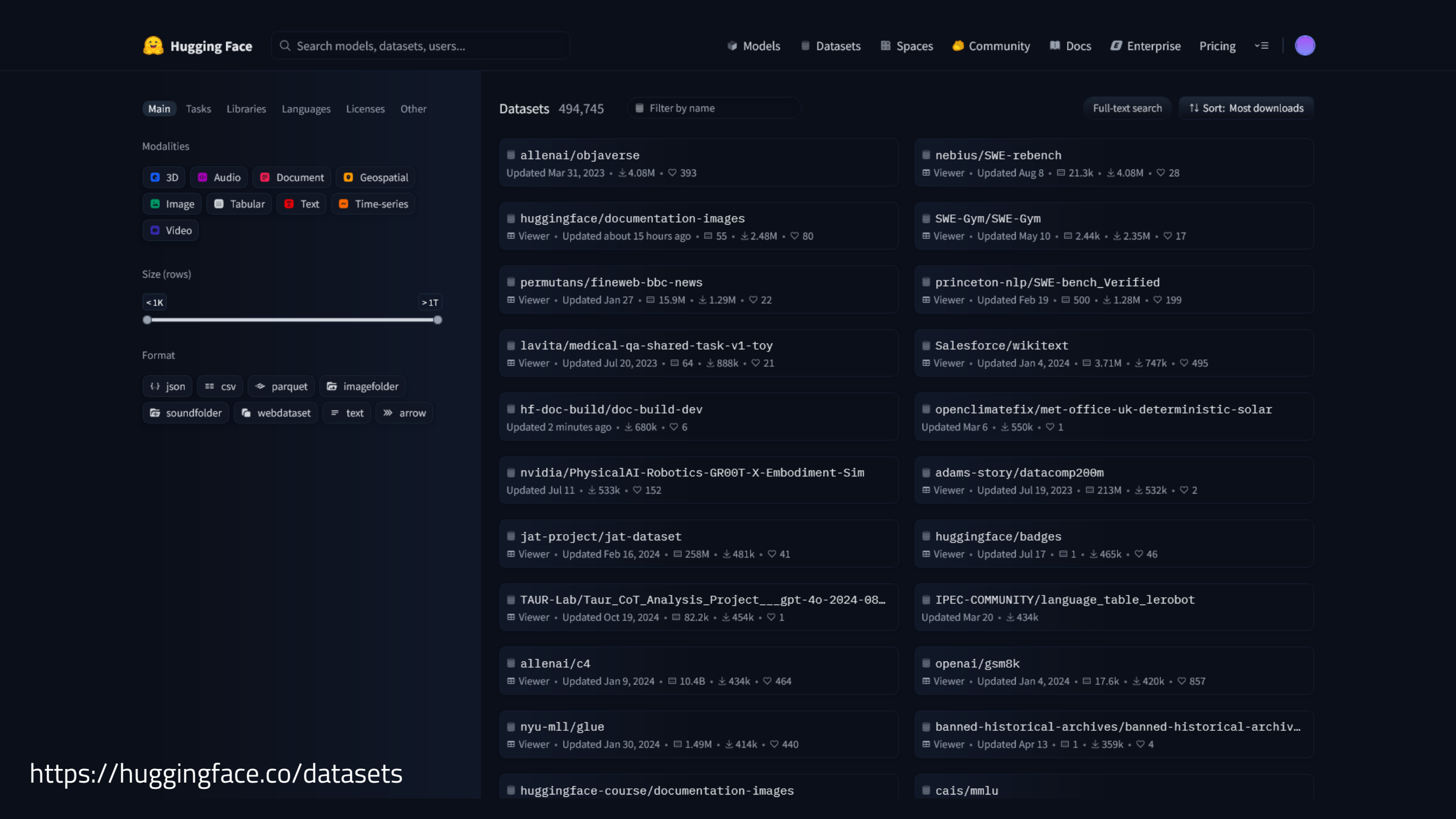
like 2.4k | Follow BERT community 781

Fill-Mask | Transformers | PyTorch | TensorFlow | JAX | Rust | Core ML | ONNX | Safetensors | bookcorpus | wikipedia | English | bert | exbert | arxiv:1810.04805

License: apache-2.0

main | bert-base-uncased | Go to file Ctrl+K | 14 contributors | History: 26 commits | + Contribute

lysandre HF Staff	Updates the tokenizer configuration file (#62) 86b5e09 VERIFIED		over 1 year ago
coreml		Add Core ML conversion (#42)	over 2 years ago
.gitattributes Safe	491 Bytes	Adding `safetensors` variant of this model (#15)	almost 3 years ago
LICENSE Safe	11.4 kB	Upload LICENSE (#2)	over 3 years ago
README.md Safe	10.5 kB	Update README.md (#11)	almost 3 years ago
config.json Safe	570 Bytes	correct weights	over 4 years ago
flax_model.msgpack Safe	438 MB xet	correct weights	over 4 years ago
model.onnx Safe	532 MB xet	Adding ONNX file of this model (#40)	about 2 years ago
model.safetensors Safe	440 MB xet	Adding `safetensors` variant of this model (#15)	almost 3 years ago
pytorch_model.bin Safe pickle	440 MB xet	Update pytorch_model.bin	about 6 years ago
rust_model.ot	534 MB xet	Update rust_model.ot	over 5 years ago
tf_model.h5	536 MB xet	Update tf_model.h5	almost 6 years ago
tokenizer.json Safe	466 kB	Update tokenizer.json	almost 5 years ago
tokenizer_config.json	48 Bytes	Updates the tokenizer configuration file (#62)	over 1 year ago
vocab.txt Safe	232 kB	Update vocab.txt	almost 7 years ago



https://huggingface.co/datasets

Datasets: stanfordnlp/snli

like 79 | Following Stanford NLP 259

Tasks: Text Classification | Modalities: Text | Formats: parquet | Sub-tasks: natural-language-inference, multi-input-text-classification | Languages: English | Size: 100K - 1M | ArXiv: arxiv:1508.05326

Libraries: Datasets, pandas, Croissant, +1 | License: cc-by-sa-4.0

Dataset card | Data Studio | Files and versions xet | Community 10

Dataset Viewer

Auto-converted to Parquet | API | Embed | Data Studio

Split (3)
train · 550k rows

Search this dataset

premise string · lengths	hypothesis string · lengths	label class label
A person on a horse jumps over a broken down airplane.	A person is training his horse for a competition.	1 neutral
A person on a horse jumps over a broken down airplane.	A person is at a diner, ordering an omelette.	2 contradiction
A person on a horse jumps over a broken down airplane.	A person is outdoors, on a horse.	0 entailment
Children smiling and waving at camera	They are smiling at their parents	1 neutral
Children smiling and waving at camera	There are children present	0 entailment
Children smiling and waving at camera	The kids are frowning	2 contradiction
A boy is jumping on skateboard in the middle of a red bridge.	The boy skates down the sidewalk.	2 contradiction
A boy is jumping on skateboard in the middle of a red bridge.	The boy does a skateboarding trick.	0 entailment

7 – 402 | 1 – 295 | 3 classes

Previous 1 2 3 ... 5,502 Next

Dataset Card for SNLI

Dataset Summary

Downloads last month 10,485

Use this dataset | Edit dataset card | Papers with Code

Homepage: nlp.stanford.edu
Paper: aclanthology.org
Paper: arxiv.org
Leaderboard: nlp.stanford.edu
Point of Contact: Samuel Bowman
Point of Contact: Gabor Angeli
Point of Contact: Chris Manning
Size of downloaded dataset files: 20.4 MB
Size of the auto-converted Parquet files: 20.4 MB
Number of rows: 570,152

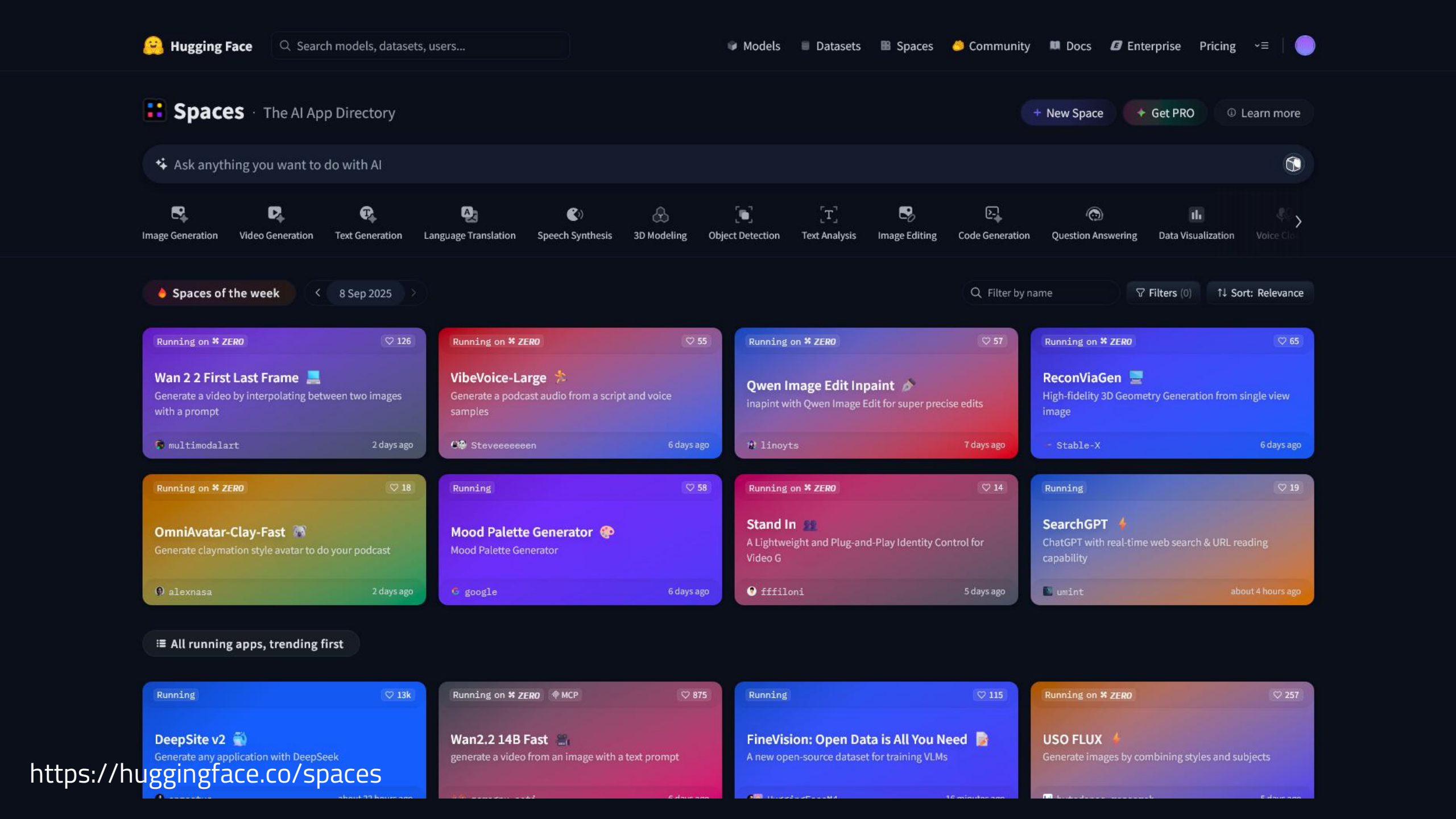
Models trained or fine-tuned on stanfordnlp/sn...

sentence-transformers/all-MiniLM-L6-v2
Sentence Similarity • 0.0B • 89.6M • 3

sentence-transformers/all-mpnet-base-...
Sentence Similarity • 0.1B • 17.1M • 1

sentence-transformers/all-MiniLM-L12-...
Sentence Similarity • 0.0B • 3.68M • 2

sileod/deberta-v3-base-tasksource-nli



https://huggingface.co/spaces

Try it yourself!

Our recent research shows that sentence transformer ("AI" models) can predict respondent patterns in survey data! The model accurately infers item-correlation with r = **.71** 🧨, and shows even higher precision for scale correlations (r = **.89** 💥) and reliability coefficients (r = **.86** 💣)!

Try it yourself by defining a scale structure using the input field below and let the **SurveyBot3000** predict the expected response pattern. Use the YAML format or follow the structure outlined by the preset example:

- Scale names must end with "**:**"
- Nest items under a scale name by prepending "**-**" before each item

Questionnaire Structure (YAML-Formatted)

```
Dairy Product Phobia:
  - I can easily eat dairy products without feeling nervous.
  - I avoid places where dairy products are prominent, like ice cream shops.
  - The sight of milk or cheese makes me uncomfortable.
  - The thought of touching dairy products scares me.
Velociraptor Awareness:
  - I can identify different types of dinosaurs from a distance.
  - I am usually aware of any raptors in my surroundings.
  - I prefer to live in a neighborhood with a low density of velociraptors.
  - The idea of velociraptors roaming free does not concern me.
```

☐ Result as matrix ⓘ

Get Synthetic Estimates

https://huggingface.co/spaces/magnolia-psychometrics/synthetic-correlations

	item_a	item_b	Θ
3	I avoid places where dairy products are prominent, like ice cream shops.	I can easily eat dairy products without feeling nervous.	-0.13
4	I am usually aware of any raptors in my surroundings.	I can easily eat dairy products without feeling nervous.	-0.04
8	I avoid visiting places where I know there are likely to be reptiles.	I can easily eat dairy products without feeling nervous.	-0.12
14	I can easily eat dairy products without feeling nervous.	I can identify different types of dinosaurs from a distance.	0.06
17	I avoid places where dairy products are prominent, like ice cream shops.	I can identify different types of dinosaurs from a distance.	-0.06
18	I am usually aware of any raptors in my surroundings.	I can identify different types of dinosaurs from a distance.	0.27
22	I avoid visiting places where I know there are likely to be reptiles.	I can identify different types of dinosaurs from a distance.	-0.11
28	I can easily eat dairy products without feeling nervous.	I often look out for signs of reptiles in my local area.	0
29	I can identify different types of dinosaurs from a distance.	I often look out for signs of reptiles in my local area.	0.42
31	I avoid places where dairy products are prominent, like ice cream shops.	I often look out for signs of reptiles in my local area.	0
32	I am usually aware of any raptors in my surroundings.	I often look out for signs of reptiles in my local area.	0.44
36	I avoid visiting places where I know there are likely to be reptiles.	I often look out for signs of reptiles in my local area.	0.19
40	I have never noticed any prehistoric creatures around me.	I often look out for signs of reptiles in my local area.	-0.07
41	I have never been worried about encountering reptiles in my daily life.	I often look out for signs of reptiles in my local area.	-0.01
46	I am usually aware of any raptors in my surroundings.	I avoid places where dairy products are prominent, like ice cream shops.	-0.01
70	I can easily eat dairy products without feeling nervous.	I prefer parks and outdoor areas that are known to be free of reptiles.	0.06
71	I can identify different types of dinosaurs from a distance.	I prefer parks and outdoor areas that are known to be free of reptiles.	0.15
72	I often look out for signs of reptiles in my local area.	I prefer parks and outdoor areas that are known to be free of reptiles.	0.29
73	I avoid places where dairy products are prominent, like ice cream shops.	I prefer parks and outdoor areas that are known to be free of reptiles.	0.17

	item_a	item_b	Θ
3	I avoid places where dairy products are prominent, like ice cream shops.	I can easily eat dairy products without feeling nervous.	-0.13
4	I am usually aware of any raptors in my surroundings.	I can easily eat dairy products without feeling nervous.	-0.04
8	I avoid visiting places where I know there are likely to be reptiles.	I can easily eat dairy products without feeling nervous.	-0.12
14	I can easily eat dairy products without feeling nervous.	I can identify different types of dinosaurs from a distance.	0.06
17	I avoid places where dairy products are prominent, like ice cream shops.	I can identify different types of dinosaurs from a distance.	-0.06
18	I am usually aware of any raptors in my surroundings.	I can identify different types of dinosaurs from a distance.	0.27
22	I avoid visiting places where I know there are likely to be reptiles.	I can identify different types of dinosaurs from a distance.	-0.11
28	I can easily eat dairy products without feeling nervous.	I often look out for signs of reptiles in my local area.	0
29	I can identify different types of dinosaurs from a distance.	I often look out for signs of reptiles in my local area.	0.42
31	I avoid places where dairy products are prominent, like ice cream shops.	I often look out for signs of reptiles in my local area.	0
32	I am usually aware of any raptors in my surroundings.	I often look out for signs of reptiles in my local area.	0.44
36	I avoid visiting places where I know there are likely to be reptiles.	I often look out for signs of reptiles in my local area.	0.19
40	I have never noticed any prehistoric creatures around me.	I often look out for signs of reptiles in my local area.	-0.07
41	I have never been worried about encountering reptiles in my daily life.	I often look out for signs of reptiles in my local area.	-0.01
46	I am usually aware of any raptors in my surroundings.	I avoid places where dairy products are prominent, like ice cream shops.	-0.01
70	I can easily eat dairy products without feeling nervous.	I prefer parks and outdoor areas that are known to be free of reptiles.	0.06
71	I can identify different types of dinosaurs from a distance.	I prefer parks and outdoor areas that are known to be free of reptiles.	0.15
72	I often look out for signs of reptiles in my local area.	I prefer parks and outdoor areas that are known to be free of reptiles.	0.29
73	I avoid places where dairy products are prominent, like ice cream shops.	I prefer parks and outdoor areas that are known to be free of reptiles.	0.17

Documentation

Search across all docs

Hub & Client Libraries

- **Hub**
  Host Git-based models, datasets, and Spaces on the HF Hub
- **Hub Python Library**
  Python client to interact with the Hugging Face Hub
- **Huggingface.js**
  JavaScript libraries for Hugging Face with built-in TS types
- **Tasks**
  Explore demos, models, and datasets for any ML tasks
- **Dataset viewer**
  API for metadata, stats, and content of HF Hub datasets

Deployment & Inference

- **Inference Providers**
  Call 200k+ models hosted by our 10+ Inference partners
- **Inference Endpoints (dedicated)**
  Deploy models on dedicated & fully managed infrastructure on HF
- **Deploying on AWS**
  Train/deploy models from Hugging Face to AWS with DLCs
- **Text Generation Inference**
  Serve language models with TGI optimized toolkit
- **Text Embeddings Inference**
  Serve embeddings models with TEI optimized toolkit
- **Microsoft Azure**
  Deploy Hugging Face models on Microsoft Azure

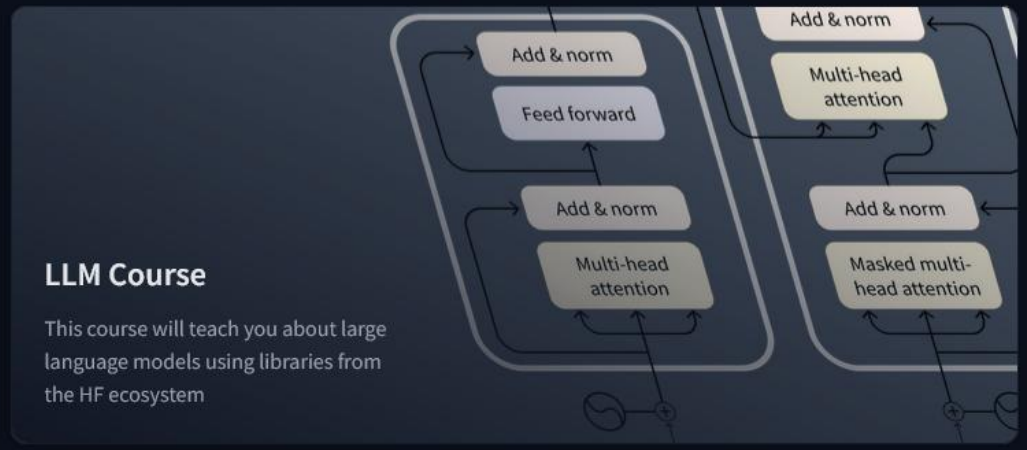
Core ML Libraries

- **Transformers**
  State-of-the-art AI models for PyTorch
- **Diffusers**
  State-of-the-art Diffusion models in PyTorch
- **Datasets**
  Access & share datasets for any ML tasks
- **Transformers.js**
  State-of-the-art ML running directly in your browser
- **Tokenizers**
  Fast tokenizers optimized for research & production
- **Evaluate**
  Evaluate and compare models performance
- **timm**
  State-of-the-art vision models, layers, optimizers, and utilities
- **Sentence Transformers**
  Embeddings, Retrieval, and Reranking

https://huggingface.co/docs

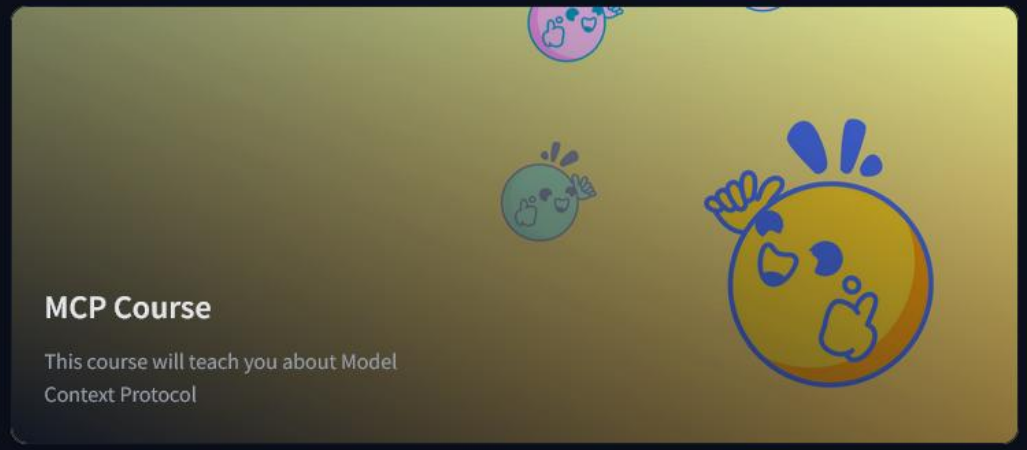
Hugging Face | Search models, datasets, users... | Models | Datasets | Spaces | Community | Docs | Enterprise | Pricing

Learn



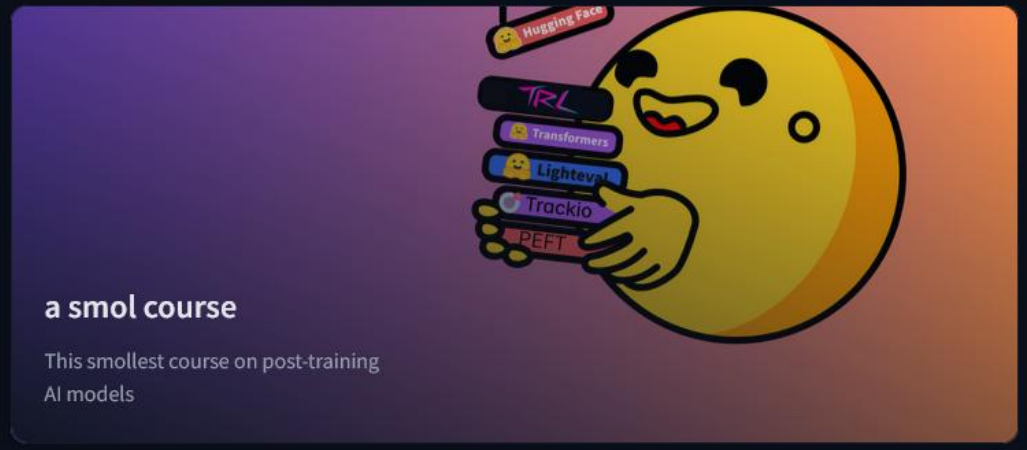
LLM Course

This course will teach you about large language models using libraries from the HF ecosystem



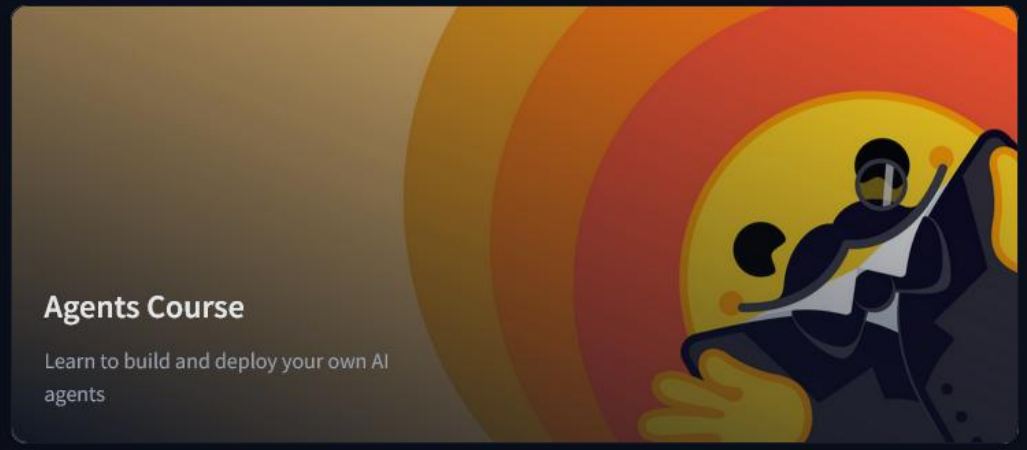
MCP Course

This course will teach you about Model Context Protocol



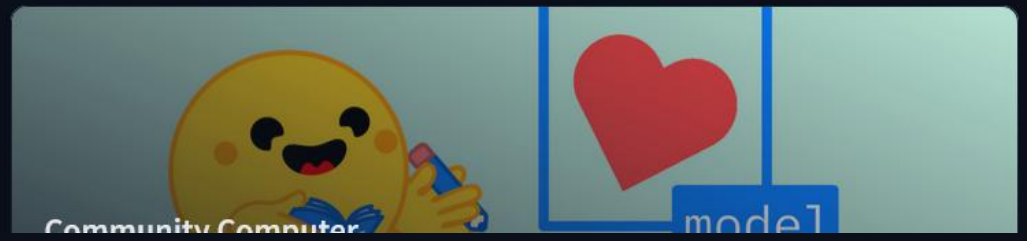
a smol course

This smollest course on post-training AI models



Agents Course

Learn to build and deploy your own AI agents



https://huggingface.co/learn

Part IV

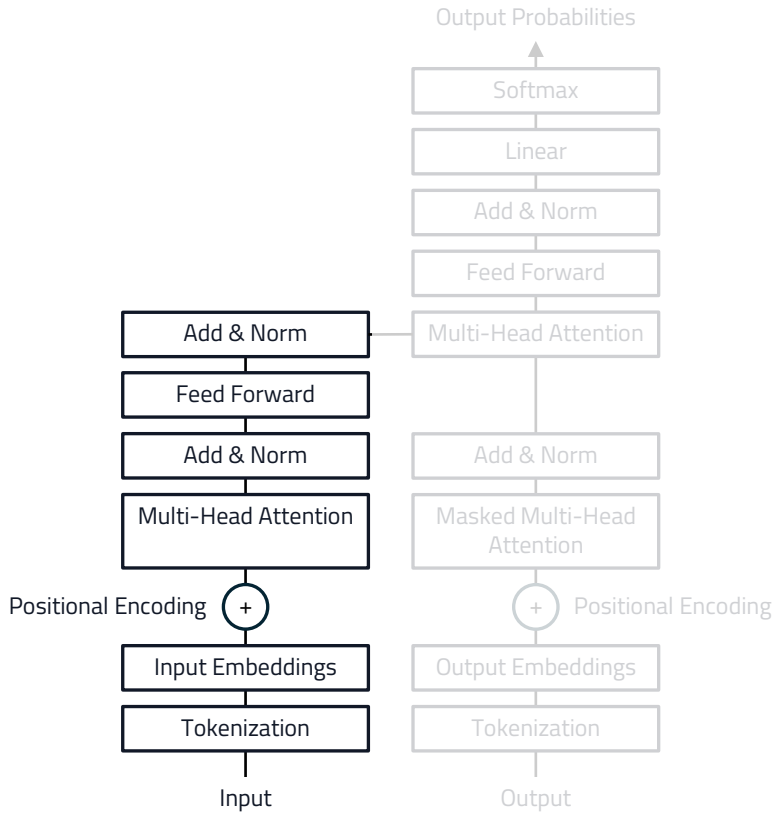
Encoder-Only Models

Encoder-Only Models

Sub-Agenda

In this part we'll look at basic use-cases for encoder-only models:

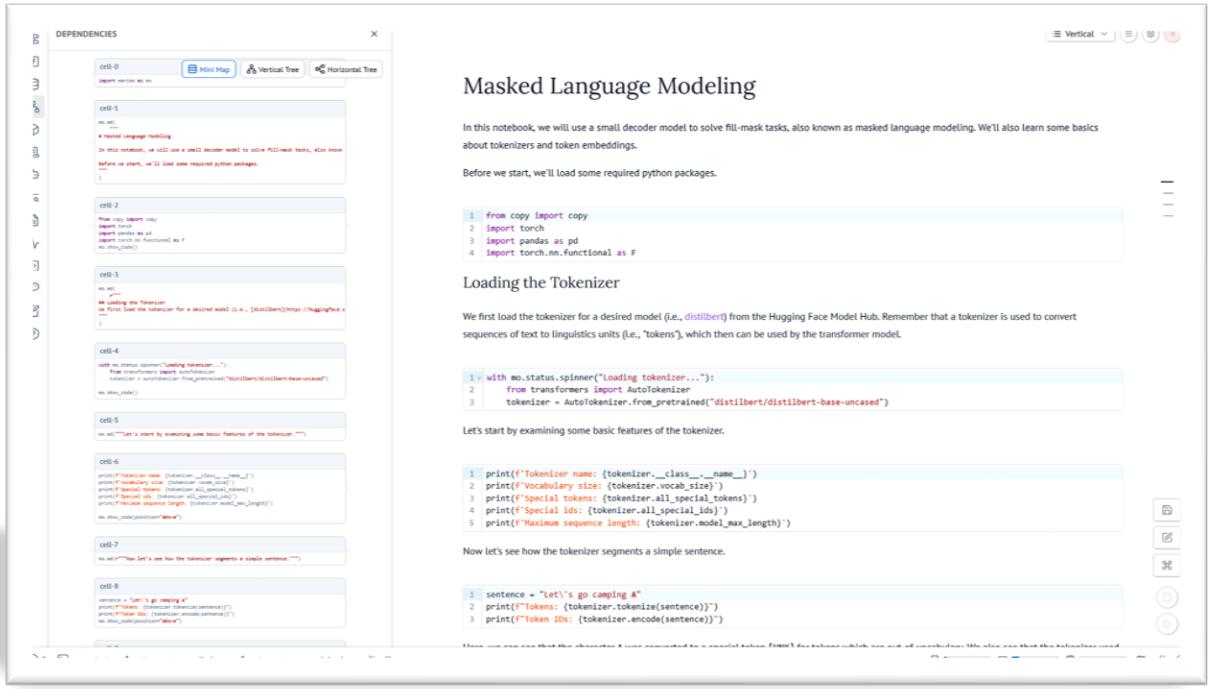
- Tokenization and masked-language modeling.
- Text classification in the case of sentiment analysis.
- Sentence Similarity
- Evaluating Performance



Dr. Björn E. Hommel | bjoern.hommel@uni-leipzig.de

Interactive Exercise

Masked Language Modeling




```
poetry run marimo edit ./notebooks/02-masked-language-modeling.py

Edit 02-masked-language-modeling.py in your browser 📝

➜  URL: http://localhost:2720?access_token=LM1yR3eI5HjmVeyC43fRYw
```

Encoder-Only Models

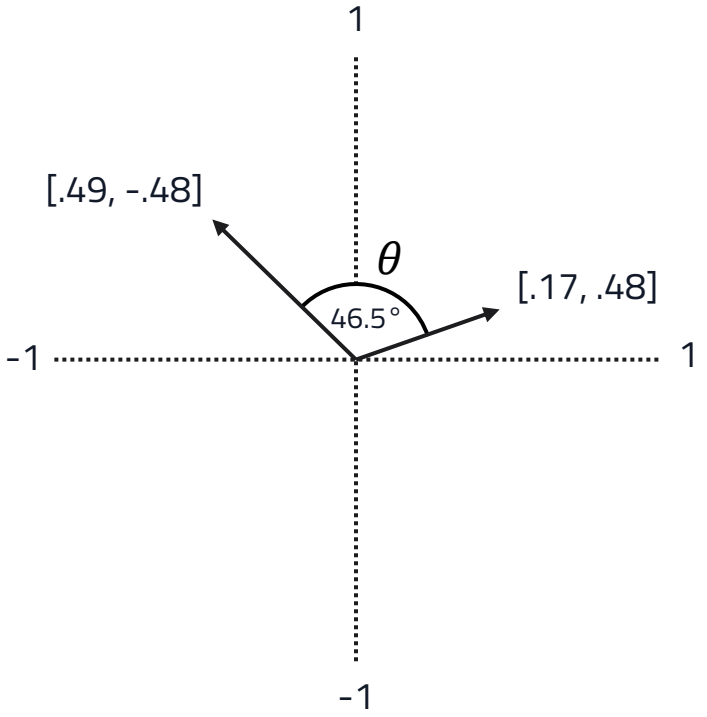
Why use Cosine Similarity?

In the previous exercise, we used cosine similarity to measure the relatedness of two token embeddings, but why?

Pearson Corelation

Considers the magnitude of elements

$$r = \frac{\sum(x_i - \hat{x})\,(y_i - \hat{y})}{\sqrt{\sum(x_i - \hat{x})^2 \sum(y_i - \hat{y})^2}}$$



Cosine Similarity

Considers the direction of elements (angle). In semantic space, the magnitudes of elements do not carry meaning.

$$s = \cos(\theta)$$

Dr. Björn E. Hommel | bjoern.hommel@uni-leipzig.de

Encoder-Only Models

Pipelines

```
from transformers import AutoTokenizer, AutoModelForMaskedLM


tokenizer = AutoTokenizer.from_pretrained("distilbert/distilbert-base-uncased")
model = AutoModelForMaskedLM.from_pretrained("bert-base-uncased")


text = [
    "We are all just monkeys with money and [MASK].",
    "Got a head full of lightning, a [MASK] full of rain.",
]


tokenizer_output = tokenizer(
    text=text,
    padding=True,
    truncation=True,
    return_attention_mask=True,
    return_tensors="pt"
)


outputs = model(
    input_ids=tokenizer_output['input_ids'],
    attention_mask=tokenizer_output['attention_mask']
)

logits = outputs.logits

[...]
```

As seen in the previous exercise, a simple fill-mask task can require quite a bit of coding.

Luckily, we can use **Pipelines** for well-defined problems...

Encoder-Only Models

Pipelines

```
from transformers import pipeline

unmasker = pipeline("fill-mask", model="distilbert/distilbert-base-uncased")
result = unmasker("We are all just monkeys with money and [MASK].")
```

Using the `"fill-mask"` pipeline, we have drastically reduced the required code down to just three lines!

```
> result
[{'score': 0.057134464383125305, 'token': 2769, 'token_str': 'money', 'sequence': 'we are all just monkeys with money and money.'}, {'score': 0.049332525581121445, 'token': 5850, 'token_str': 'drugs', 'sequence': 'we are all just monkeys with money and drugs.'}, {'score': 0.0339612141251564, 'token': 2373, 'token_str': 'power', 'sequence': 'we are all just monkeys with money and power.'}, {'score': 0.014588878490030766, 'token': 4409, 'token_str': 'guns', 'sequence': 'we are all just monkeys with money and guns.'}, {'score': 0.013395137153565884, 'token': 4476, 'token_str': 'fame', 'sequence': 'we are all just monkeys with money and fame.'}]
```

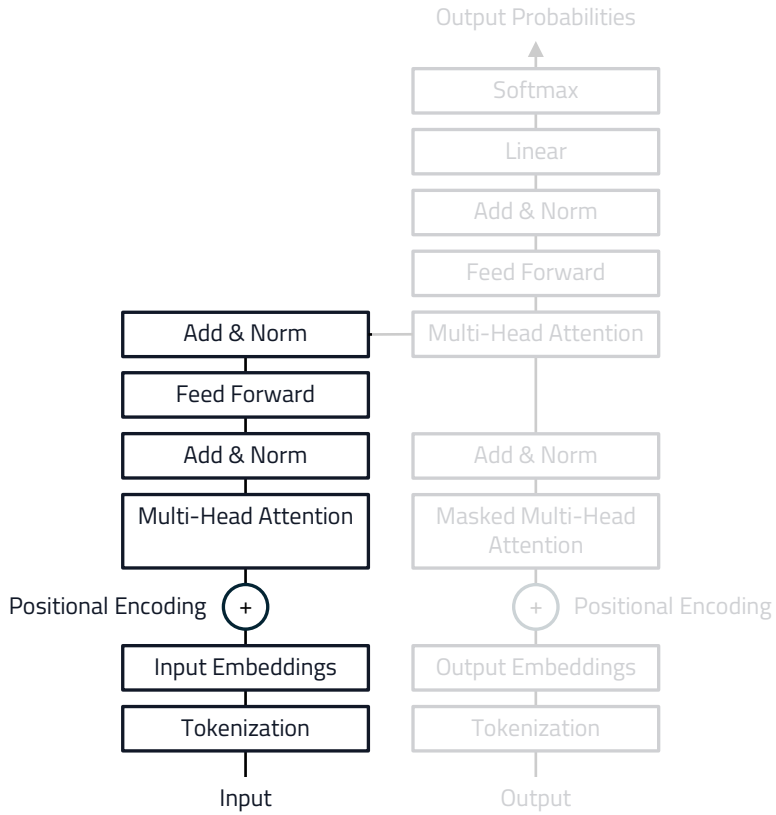
Encoder-Only Models

Pipelines

Useful pipelines for natural language processing include:

- "feature-extraction“
- "fill-mask“
- "question-answering“
- "summarization“
- "text2text-generation“
- "text-classification" or "sentiment-analysis“
- "text-generation“
- "zero-shot-classification“

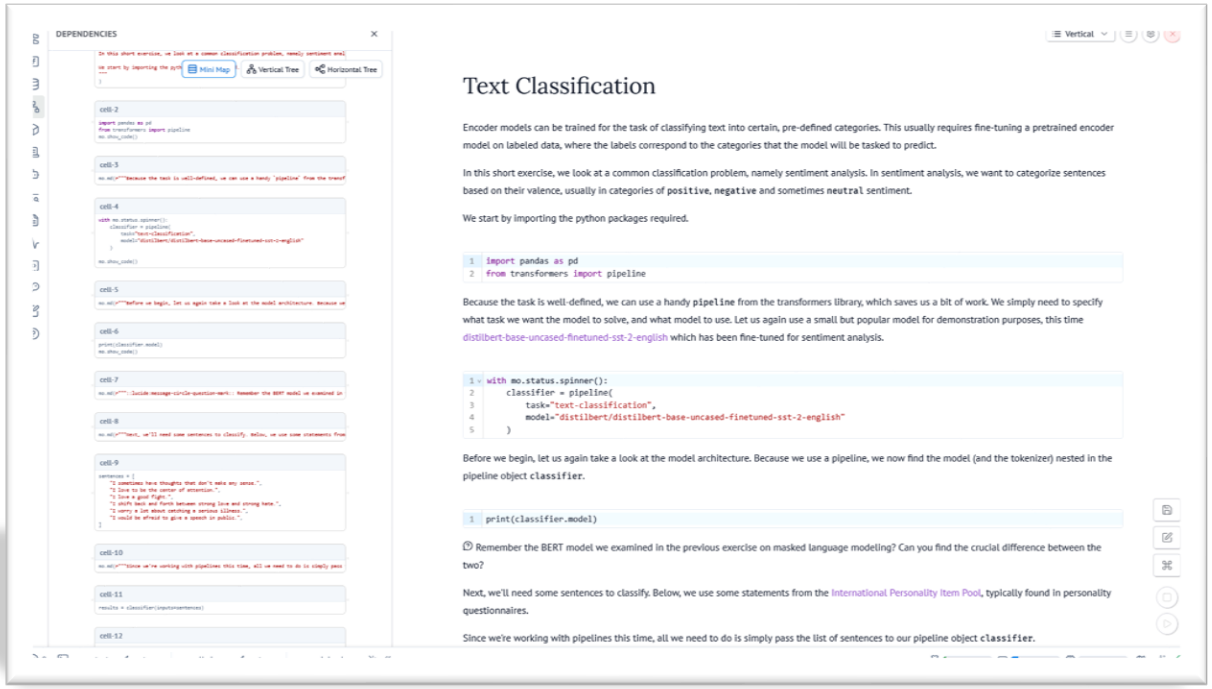
Check the Hugging Face documentation on pipelines or the model card of the model card of the model you intend to use.



Dr. Björn E. Hommel | bjoern.hommel@uni-leipzig.de

Interactive Exercise

Text Classification



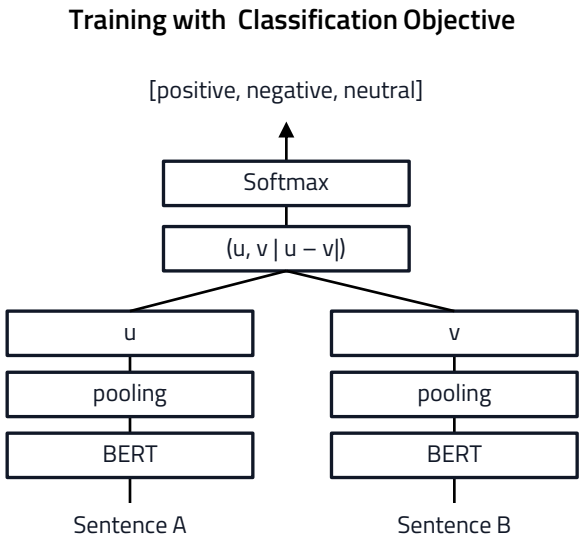

```
poetry run marimo edit ./notebooks/03-text-classification.py

Edit 03-text-classification.py in your browser ✏️

➜  URL: http://localhost:2720?access_token=LM1yR3eI5HjmVeyC43fRYw
```


Dr. Björn E. Hommel | bjoern.hommel@uni-leipzig.de

Encoder-Only Models

Sentence Similarity with Sentence Transformers

We have previously evaluated token-level similarities in two sentences, but how can we compare the meaning of two sequences on sentence-level?

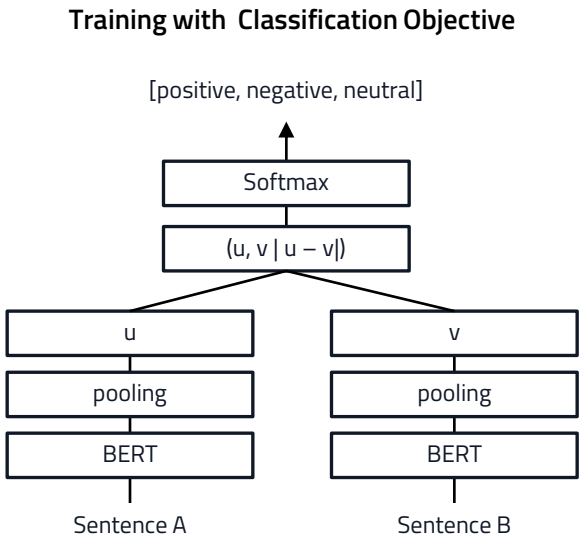


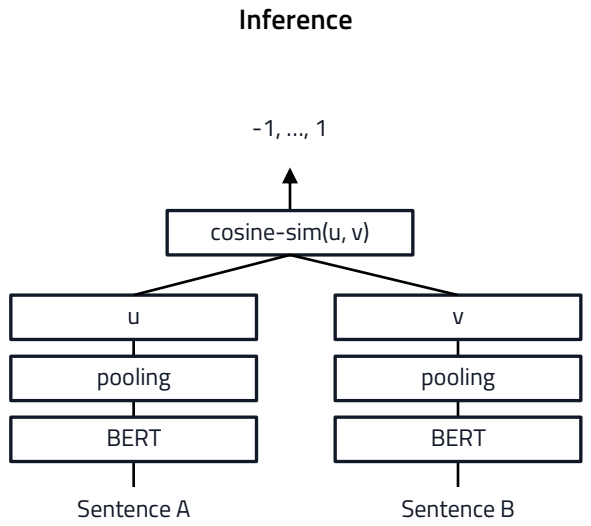
- Encoder produces embeddings with of size c, E_{dim}.
- Embeddings are average to size E_{dim} in mean pooling. Other pooling methods exist, e.g., [CLS] pooling.
- The pooled embeddings from both sentences are now concatenated, together with the element-wise difference of the embeddings.
- Softmax with a classification head predicts labeled data (e.g., positive, negative or neutral for classification objectives).

Encoder-Only Models

Sentence Similarity with Sentence Transformers

After training, the concatenation and the softmax layer can be discarded. The model has now learned how to produce semantically rich embeddings on the sentence level, by pooling embeddings on the token level. ✨





provides the best quality, while **all-MiniLM-L6-v2** is 5 times faster and still offers good quality. Toggle *All models* to see all evaluated original models.

All models

Model Name	Performance Sentence Embeddings (14 Datasets)	Performance Semantic Search (6 Datasets)	Avg. Performance	Speed	Model Size
all-mpnet-base-v2	69.57	57.02	63.30	2800	420 MB
multi-qa-mpnet-base-dot-v1	66.76	57.60	62.18	2800	420 MB
all-distilroberta-v1	68.73	50.94	59.84	4000	290 MB
all-MiniLM-L12-v2	68.70	50.82	59.76	7500	120 MB

all-MiniLM-L12-v2

Description:	All-round model tuned for many use-cases. Trained on a large and diverse dataset of over 1 billion training pairs.
Base Model:	microsoft/MiniLM-L12-H384-uncased
Max Sequence Length:	256
Dimensions:	384
Normalized Embeddings:	true
Suitable Score Functions:	dot-product (`util.dot_score`), cosine-similarity (`util.cos_sim`), euclidean distance
Size:	120 MB
Pooling:	Mean Pooling
Training Data:	1B+ training pairs. For details, see model card.
Model Card:	https://huggingface.co/sentence-transformers/all-MiniLM-L12-v2

Semantic Search Models

The following models have been specifically trained for **Semantic Search**: Given a question / search query, these models are able to find relevant text passages. For more details, see Usage > Semantic Search.

```python
from sentence_transformers import SentenceTransformer

model = SentenceTransformer("multi-qa-mpnet-base-cos-v1")

query_embedding = model.encode("How big is London")
passage_embeddings = model.encode([
    "London is known for its financial district",
    "London has 9,787,426 inhabitants at the 2011 census",
    "The United Kingdom is the fourth largest exporter of goods in the world",
])
```

Documentation

1. multi-qa-mpnet-base-cos-v1
2. `SentenceTransformer`
3. `SentenceTransformer.encode`
4. `SentenceTransformer.similarity`

https://sbert.net/

Select Benchmark

Multilingual

English

Image

Domain-Specific

Language-specific

Miscellaneous

Embedding Leaderboard

This leaderboard compares 100+ text and image embedding models across 1000+ languages. We refer to the publication of each selectable benchmark for details on metrics, languages, tasks, and task types. Anyone is welcome to add a model, add benchmarks, help us improve zero-shot annotations or propose other changes to the leaderboard.

MTEB(Multilingual, v2)

A large-scale multilingual expansion of MTEB, driven mainly by highly-curated community contributions covering 250+ languages.

- **Number of languages:** 1038
- **Number of tasks:** 131
- **Number of task types:** 9
- **Number of domains:** 20

Click for More Info

Cite and share this benchmark

Customize this Benchmark

Advanced Model Filters

Summary | Performance per Model Size | Performance per Task Type | Performance per task | Task information

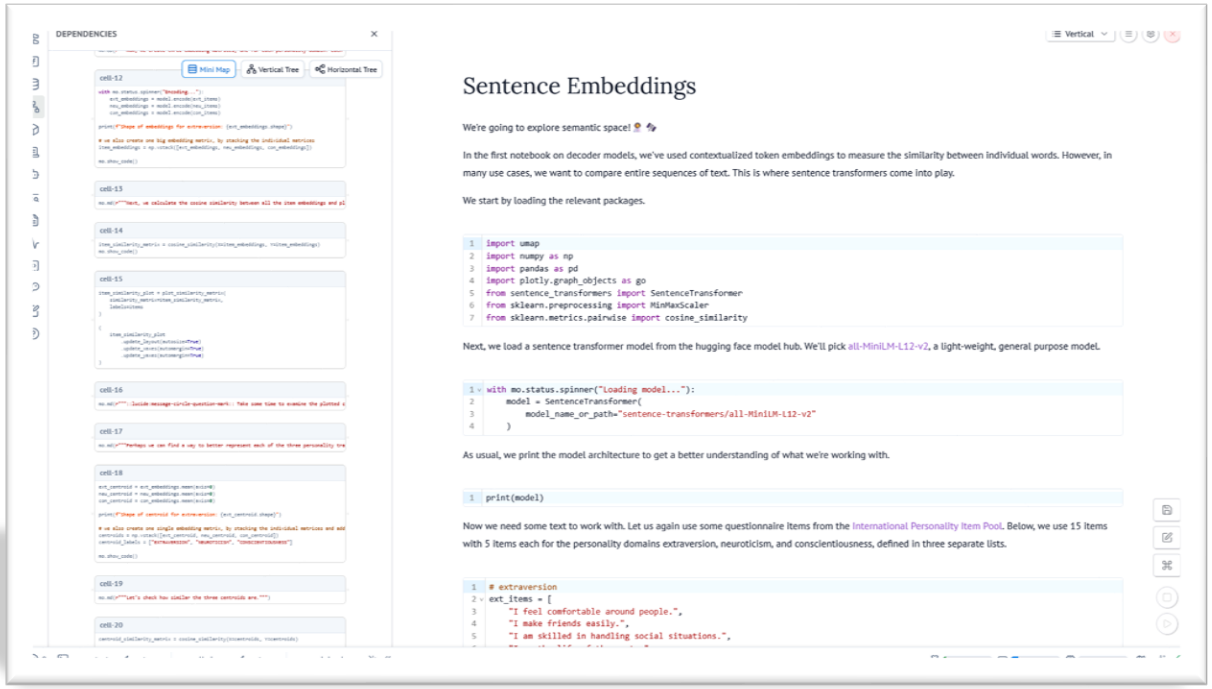
Filter...

Rank (Bor…	Model	Zero-shot	Memory U…	Number of P…	Embedding D…	Max Tokens	Mean (T…	Mean (TaskT…	Bitext …	Classification	Cluster
1	gemini-embedding-001	99%	Unknown	Unknown	3072	2048	68.37	59.59	79.28	71.82	54.59
2	Qwen3-Embedding-8B	99%	28866	7B	4096	32768	**70.58**	**61.69**	**80.89**	**74.00**	**57.65**
3	Qwen3-Embedding-4B	99%	15341	4B	2560	32768	69.45	60.86	79.36	72.33	57.15
4	Qwen3-Embedding-0.6B	99%	2272	595M	1024	32768	64.34	56.01	72.23	66.83	52.33
5	gte-Qwen2-7B-instruct	⚠ NA	29040	7B	3584	32768	62.51	55.93	73.92	61.55	52.77
6	Linq-Embed-Mistral	99%	13563	7B	4096	32768	61.47	54.14	70.34	62.24	50.60
7	multilingual-e5-large-instruct	99%	1068	560M	1024	514	63.22	55.08	80.13	64.94	50.75
8	embeddinggemma-300m	99%	578	307M	768	2048	61.15	54.31	64.40	60.90	51.17
9	SFR-Embedding-Mistral	99%	13563	7B	4096	32768	60.90	53.92	70.00	60.02	51.84
10	GritLM-7B	99%	13813	7B	4096	4096	60.92	53.74	70.53	61.83	49.75

https://huggingface.co/spaces/mteb/leaderboard

Interactive Exercise

Sentence Embeddings

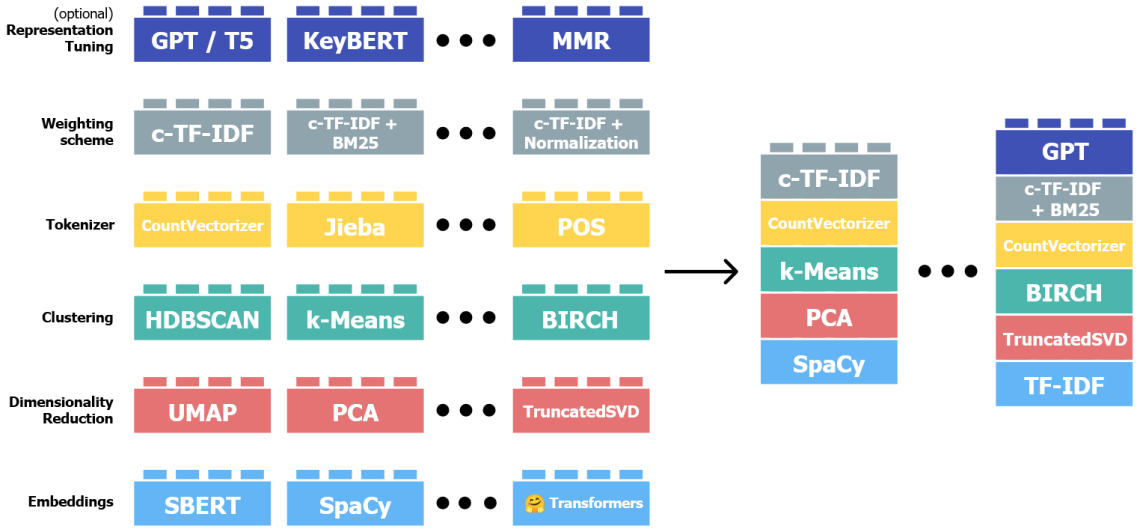


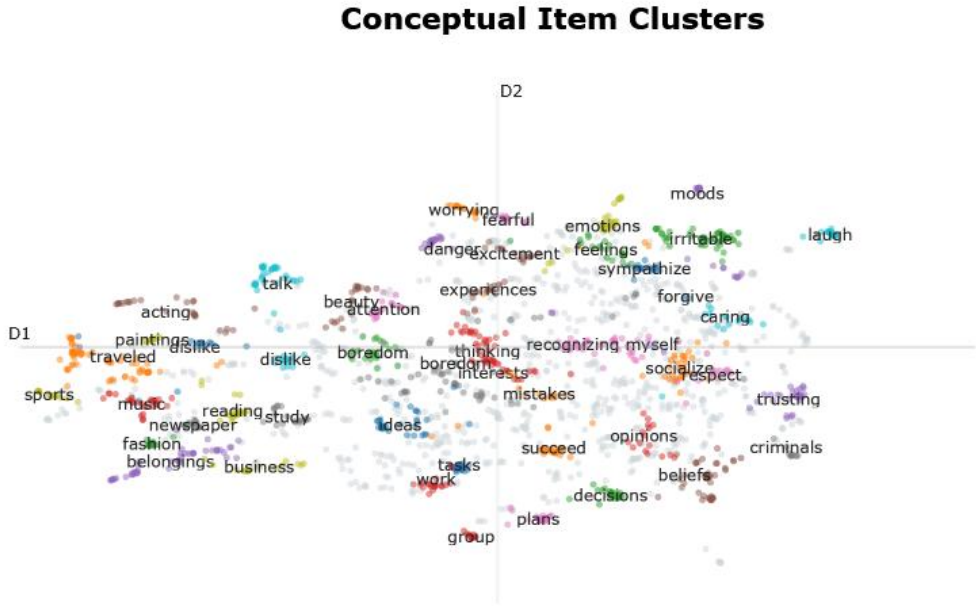

```
poetry run marimo edit ./notebooks/04-sentence-embeddings.py

Edit 04-sentence-embeddings.py in your browser

➜  URL: http://localhost:2720?access_token=LM1yR3eI5HjmVeyC43fRYw
```

Encoder-Only Models

Topic Modeling with BERTopic

We frequently encounter larger quantities of unstructured text data, which perhaps could be sorted into neat piles, if we had all the time in the world and no hobbies. Topic modeling describes a suite of algorithms that can be used cluster text collections, based on semantic information.

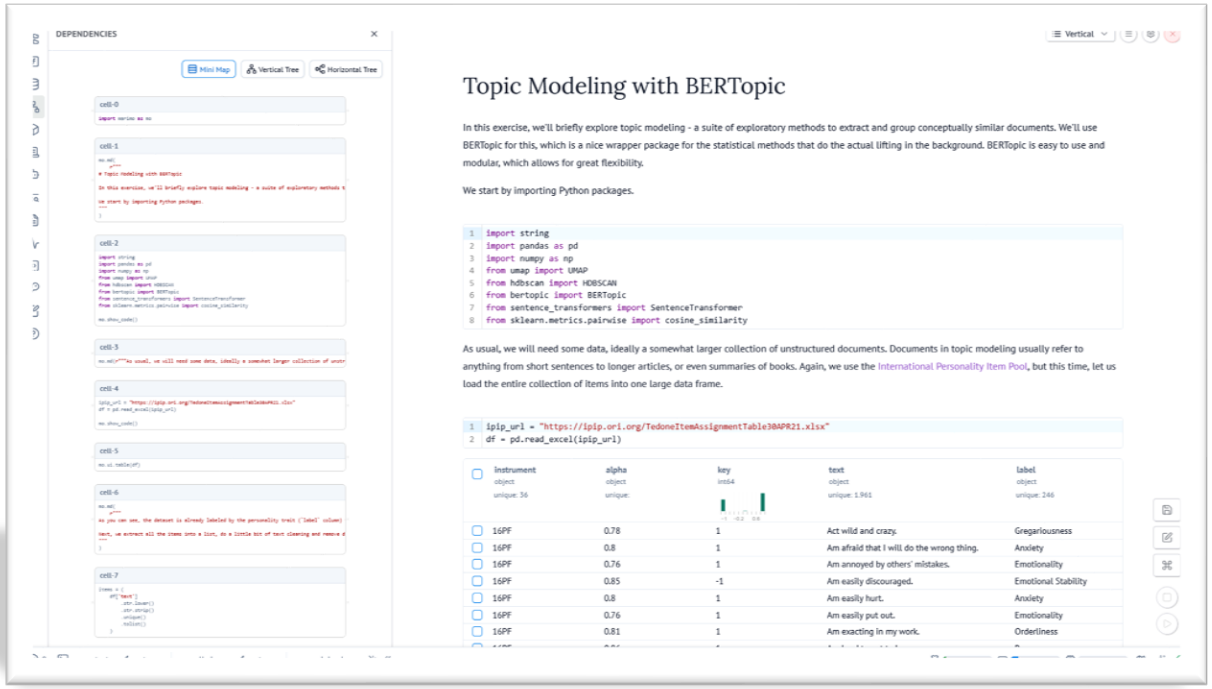




Dr. Björn E. Hommel | bjoern.hommel@uni-leipzig.de

Interactive Exercise

Topic Modeling




```
poetry run marimo edit ./notebooks/05-topic-modeling.py

Edit 05-topic-modeling.py in your browser 📝

➜  URL: http://localhost:2720?access_token=LM1yR3eI5HjmVeyC43fRYw
```

BERTopic

BERTopic is a topic modeling technique that leverages 🤗 transformers and c-TF-IDF to create dense clusters allowing for easily interpretable topics whilst keeping important words in the topic descriptions.

BERTopic supports all kinds of topic modeling techniques:

Guided	Supervised	Semi-supervised
Manual	Multi-topic distributions	Hierarchical
Class-based	Dynamic	Online/Incremental
Multimodal	Multi-aspect	Text Generation/LLM
Zero-shot **(new!)**	Merge Models **(new!)**	Seed Words **(new!)**

Corresponding medium posts can be found here, here and here. For a more detailed overview, you can read the paper or see a brief overview.

Installation

Installation, with sentence-transformers, can be done using pypi:

```
pip install bertopic
```

You may want to install more depending on the transformers and language backends that you will be using. The possible

https://maartengr.github.io/BERTopic/index.html

Encoder-Only Models

Evaluating Performance

The choice of performance metrics depends on the task type and data characteristics.

For Masked Language Modeling (Fill-Masks):

- **Top-K Accuracy**: Tests if the expected token is in the in top k predictions (e.g., 5).
- **Perplexity**: Measures how "perplexed" the model is by the actual sequence of tokens compared to the expected tokens. This is simply the squared cross entropy, i.e., negative log-likelihood.

$$ppl = \left(-\frac{1}{N} \sum_{j=i}^{w} \sum_{i=1}^{v} p(x) \log(q) \right)^2$$

$p(x)$: true probability (in this case binary)

q: predicted probability

Encoder-Only Models

Evaluating Performance

The choice of performance metrics depends on the task type and data characteristics.

For text classification:

		Predicted Class	
		True	**False**
Expected Class	**True**	True Positive (TP)	False Negative (FN)
	False	False Positive (FP)	True Negative (TN)

Precision

$$\frac{TP}{(TP + FP)}$$

Recall (Sensitivity)

$$\frac{TP}{(TP + FN)}$$

Accuracy

$$\frac{(TP + TN)}{(TP + TN + FP + FN)}$$

F1-Score

$$\frac{2TP}{(2TP + FP + FN)}$$

Encoder-Only Models

Evaluating Performance

The choice of performance metrics depends on the task type and data characteristics.

For Regression-like tasks:

Mean Absolute Error

$$MSE = \frac{1}{n}\sum|y - \hat{y}|$$

Mean Squared Error

$$MSE = \frac{1}{n}\sum(y - \hat{y})^2$$

Part V

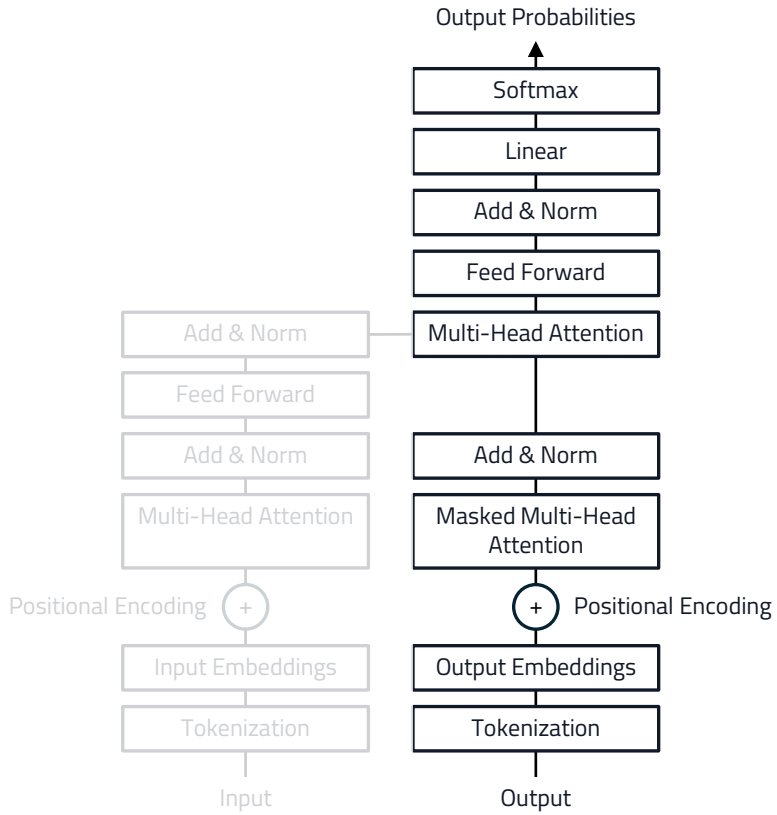
Decoder-Only Models

Decoder-Only Models

Sub-Agenda

In this part we'll look at basic use-cases for decoder-only models:

- Basic Generative Language Modeling
- Text generation with larger instruction-tuned models.
- Retrieval Augmented Generation (RAG)



Dr. Björn E. Hommel | bjoern.hommel@uni-leipzig.de

Decoder-Only Models

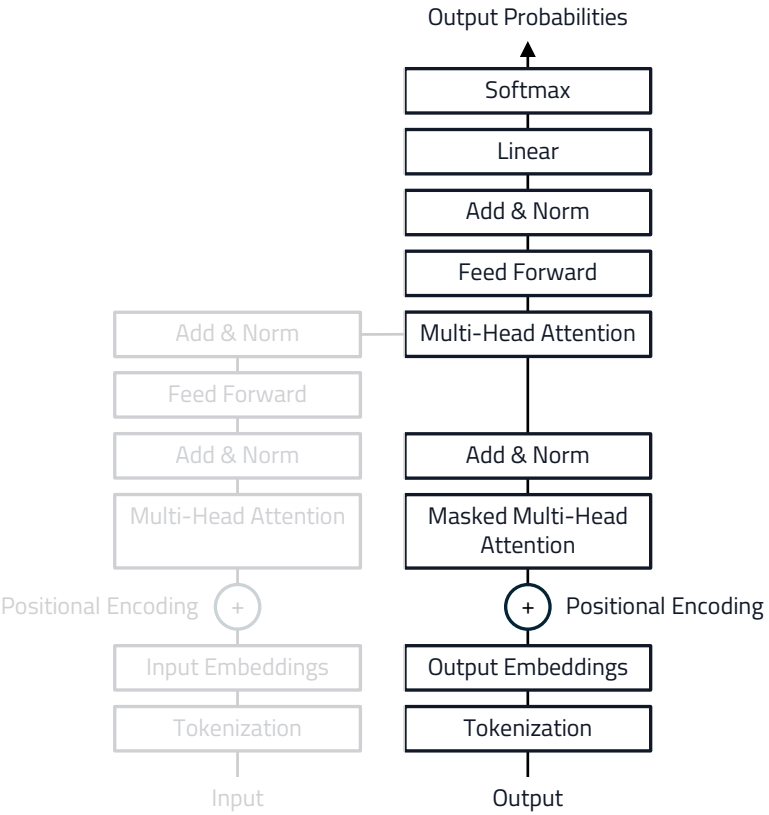
Basic Generative Language Modeling

Recall that decoder-only models predict the next token based on a previous history of tokens. To generate text, we'll therefore need to **prompt** it, by supplying a sequence of text (or at least one token) for conditional next-token prediction.

- [BOS] tokens are used by some models to mimic unconditional text generation without a prior prompt.

The central problem of generative language modeling: How to gain control over the output?

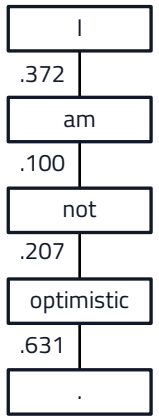
- **Instructional prompting** (i.e., **in-context learning**): Limited in early generative models (e.g., GPT-2) due to lower capabilities (e.g., parameters, context window).
- **Domain adaptation** and **fine-tuning** to more specific tasks and data.
- **Sampling strategies:** Manipulate the probability distribution for the next-token.



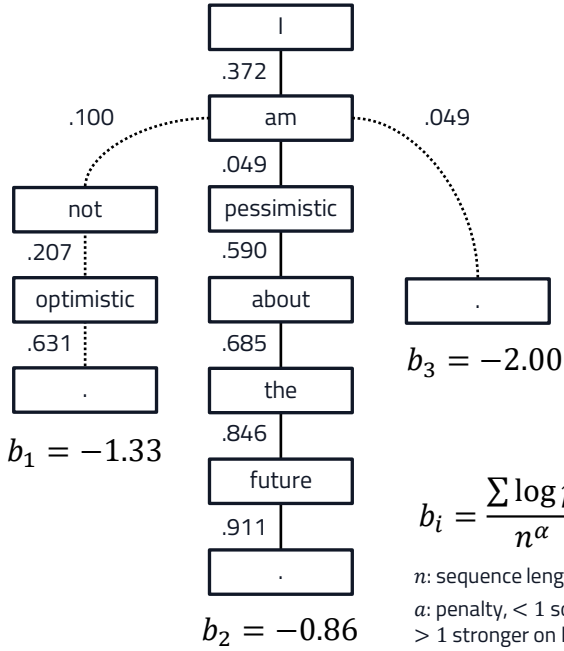
Decoder-Only Models

Sampling Strategies

Greedy Search
Always select the next token with the highest probability



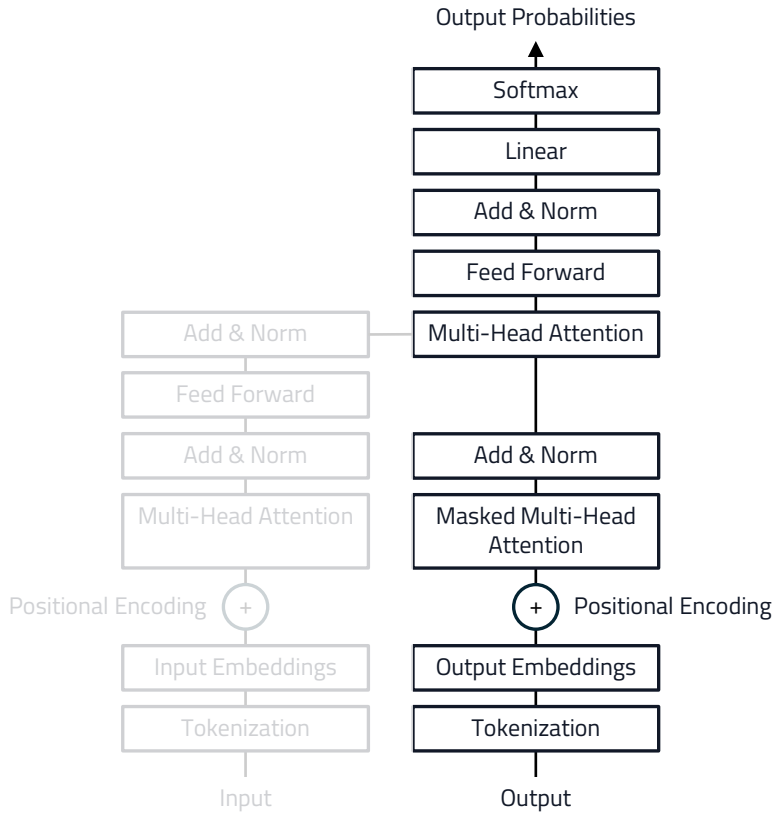
Beam Search
Explore *k* paths and the sequence with the highest, cumulative probability after normalization



$$b_i = \frac{\sum \log p}{n^\alpha}$$

n: sequence length

α: penalty, < 1 softer on length; > 1 stronger on length



Dr. Björn E. Hommel | bjoern.hommel@uni-leipzig.de

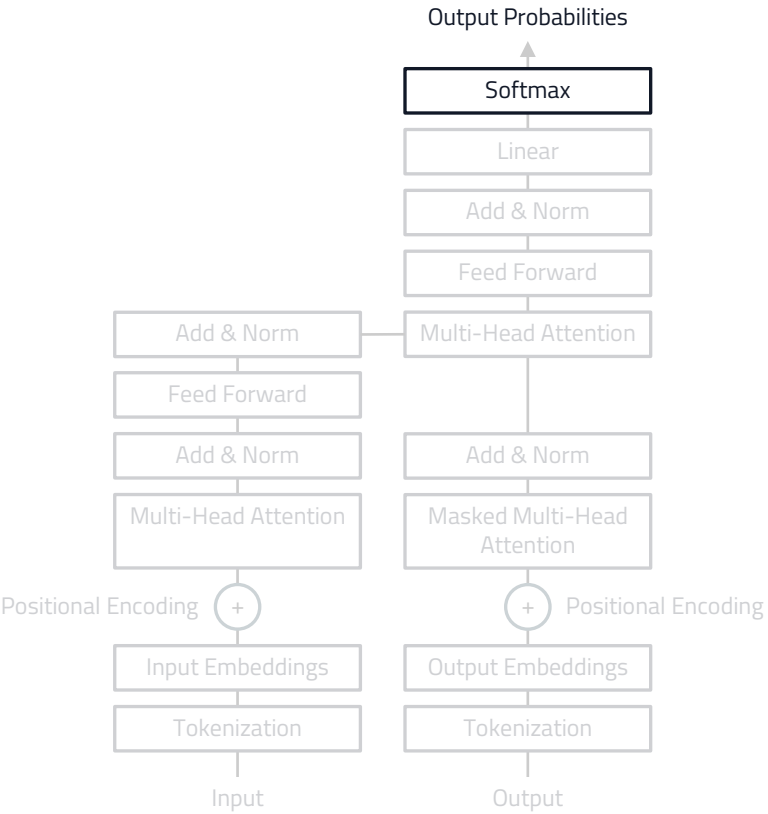
Decoder-Only Models

Sampling Strategies

Apart from greedy and beam search, **multinominal sampling** is a subset of sampling techniques which aim to modify the probability distribution of the softmax prediction.

Here, we randomly sample from the next-token probability distribution after:

- **Top k**: Keeps only the *k* highest probability tokens and sets all others to zero probability before sampling.
- **Top p** (**nucleus sampling**): Samples from the smallest set of tokens whose cumulative probability reaches at least p (e.g., 95%).



Dr. Björn E. Hommel | bjoern.hommel@uni-leipzig.de

Decoder-Only Models

Sampling Strategies

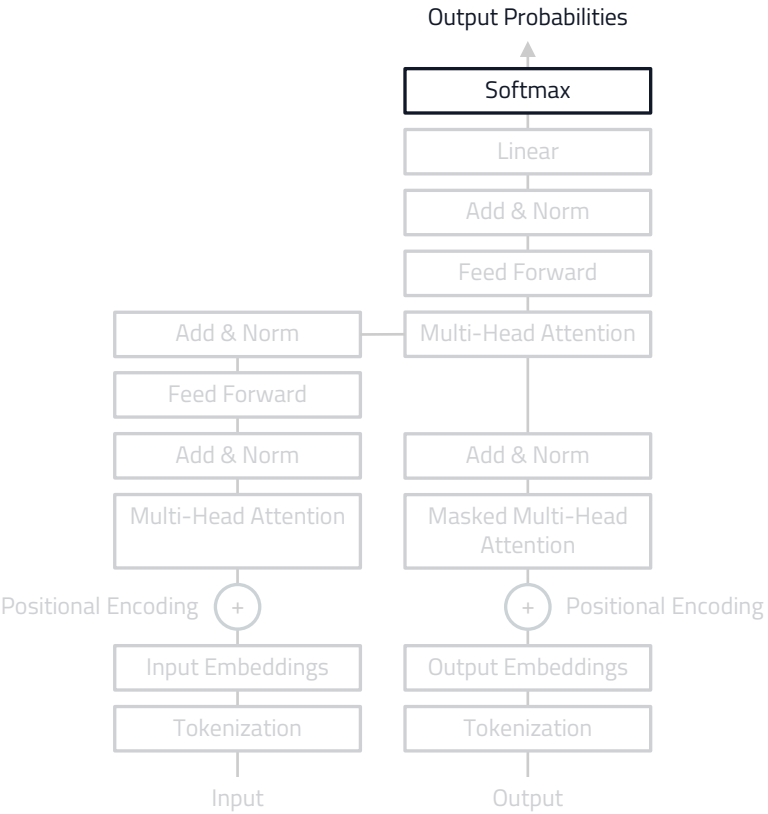
Apart from greedy and beam search, **multinominal sampling** is a subset of sampling techniques which aim to modify the probability distribution of the softmax prediction.

Here, we randomly sample from the next-token probability distribution after:

- **Temperature**: Scales the logits before applying softmax. Higher values (t > 1.0) make the distribution more uniform, while lower values (t < 1.0) make it more peaked around high-probability tokens.

$$SoftMax = \frac{e^{\frac{z_i}{T}}}{\sum_{j=1}^{N} e^{\frac{z_j}{T}}}$$

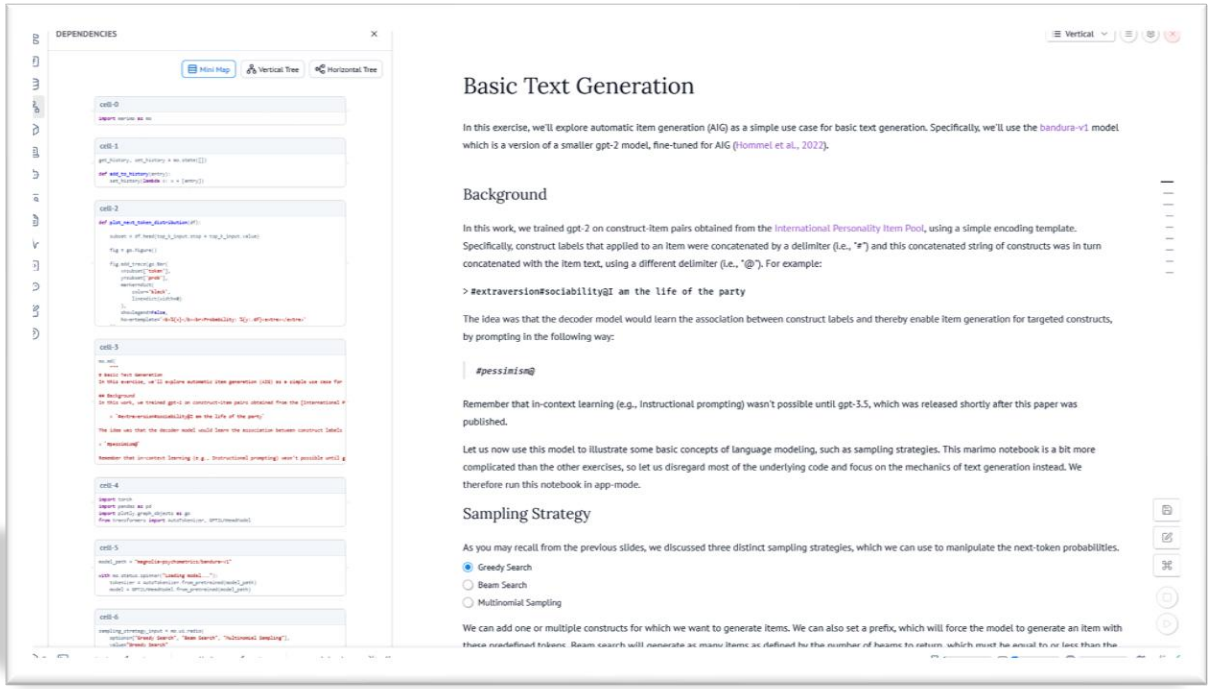
T: Temperature



Dr. Björn E. Hommel | bjoern.hommel@uni-leipzig.de

Interactive Exercise

Basic Text Generation




```
poetry run marimo edit ./notebooks/06-text-generation.py

Edit 06-text-generation.py in your browser

➜  URL: http://localhost:2720?access_token=LM1yR3eI5HjmVeyC43fRYw
```

Decoder-Only Models

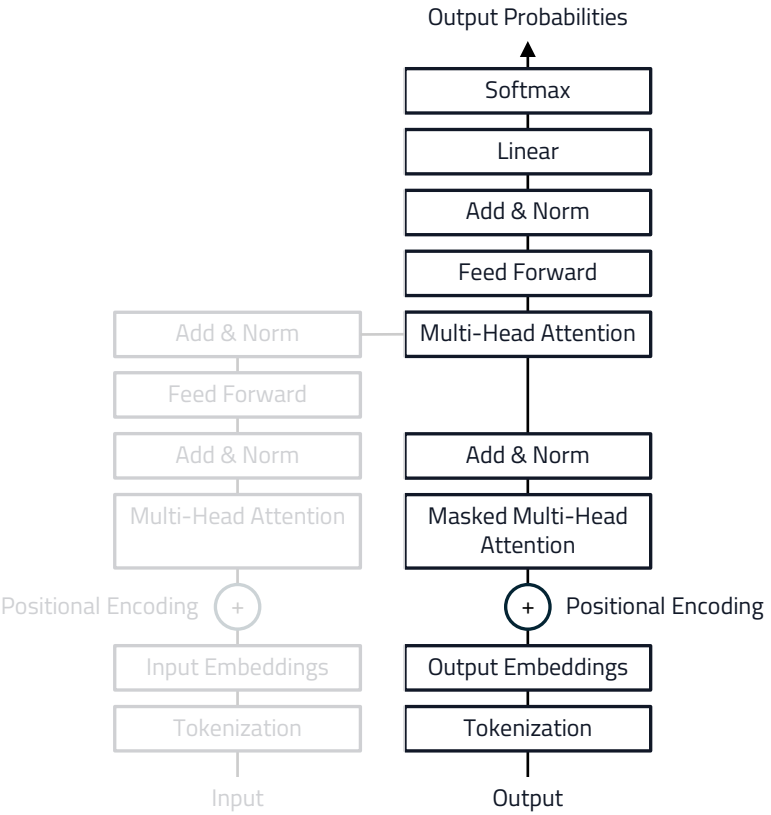
Large Language Models

The term "Large Language Models“ (LLMs) refers to decoder models with billions of parameters that demonstrated emergent capabilities around GPT-3's release.

These models showed qualitatively different performance, with capabilities determined by three factors. Their interaction became to be known as **scaling laws**:

- **Compute:** Total computational resources (FLOPs) used during training.
- **Dataset Size:** Volume of training text data, with larger datasets generally improve performance and knowledge coverage.
- **Parameters**: Number of learnable weights, determining the model's capacity to store learned patterns.

Since then, no major architectural breakthrough has fundamentally changed the transformer model.



Dr. Björn E. Hommel | bjoern.hommel@uni-leipzig.de

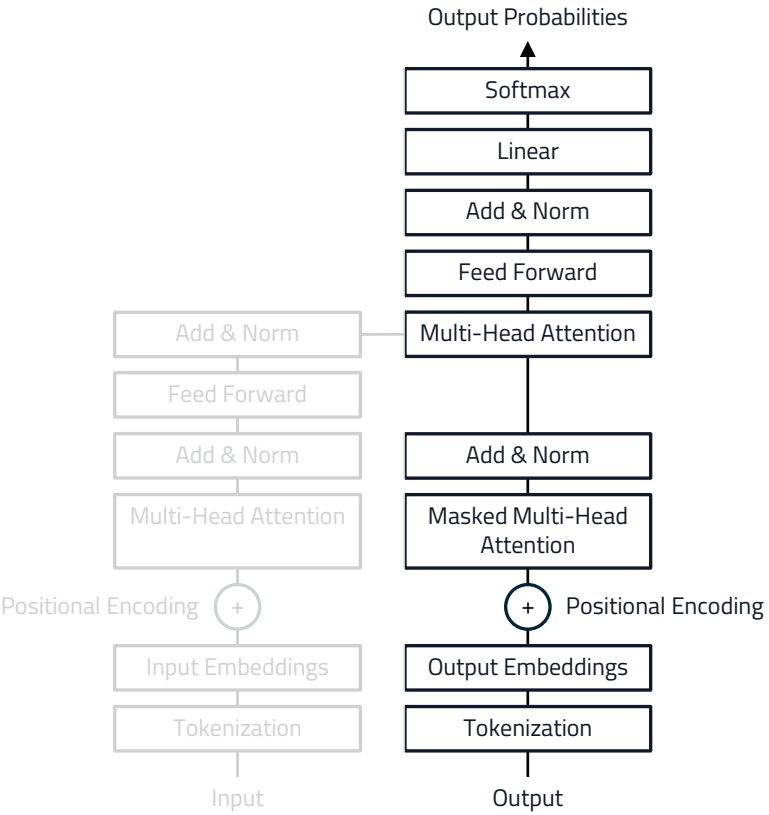
Decoder-Only Models

Large Language Models

With increases in model size, in-context learning became feasible, allowing models to "learn" from prompts and making fine-tuning less critical for many tasks.

It may make sense to differentiate between:

- **Few-shot learning:** The model learns to perform tasks by observing a small number of input-output examples provided in the prompt.
- **Zero-shot learning:** The model performs tasks without any examples, relying solely on its pre-trained knowledge and task description.
- **Instruction following**: The model executes explicit natural language instructions, though this required specialized training paradigms (e. g., instruction tuning and reinforcement learning from human feedback).



Dr. Björn E. Hommel | bjoern.hommel@uni-leipzig.de

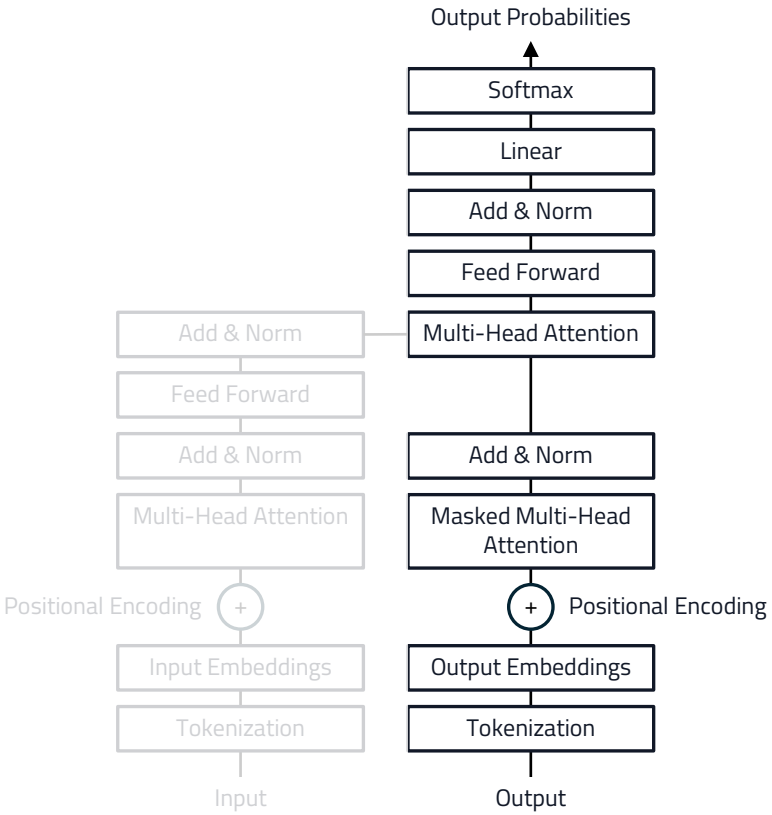
Decoder-Only Models

Large Language Models

Although modern LLMs show exceptional performance across multiple domains and can extrapolate to previously unseen tasks, some limitations do not seem to resolve simply by scaling models.

Two prominent phenomena include:

- **Sycophancy:** The tendency for models to provide responses that agree with or please the user rather than giving accurate or truthful information.
- **Hallucinations / Confabulations:** The generation of plausible-sounding but factually incorrect or fabricated information, often presented with high confidence.



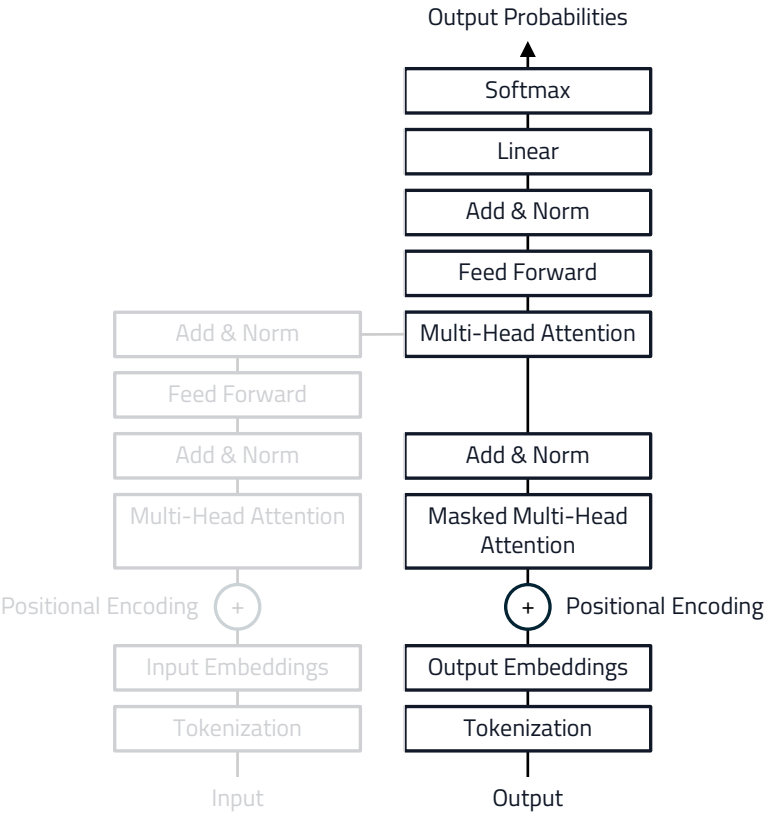
Dr. Björn E. Hommel | bjoern.hommel@uni-leipzig.de

Decoder-Only Models

Large Language Models

Some methods have been devised to reduce the risk of confabulations, or to at least make them measurable.

- **Retrieval Augmented Generation (RAG):** Combines language generation with information retrieval by fetching relevant documents to ground responses in factual sources, reducing fabricated and/or outdated content.
- **Semantic Entropy:** A method to detect potential hallucinations by measuring the semantic consistency of multiple model outputs for the same prompt. High entropy indicates uncertainty and potential confabulation (see Farquhar et al., 2024).



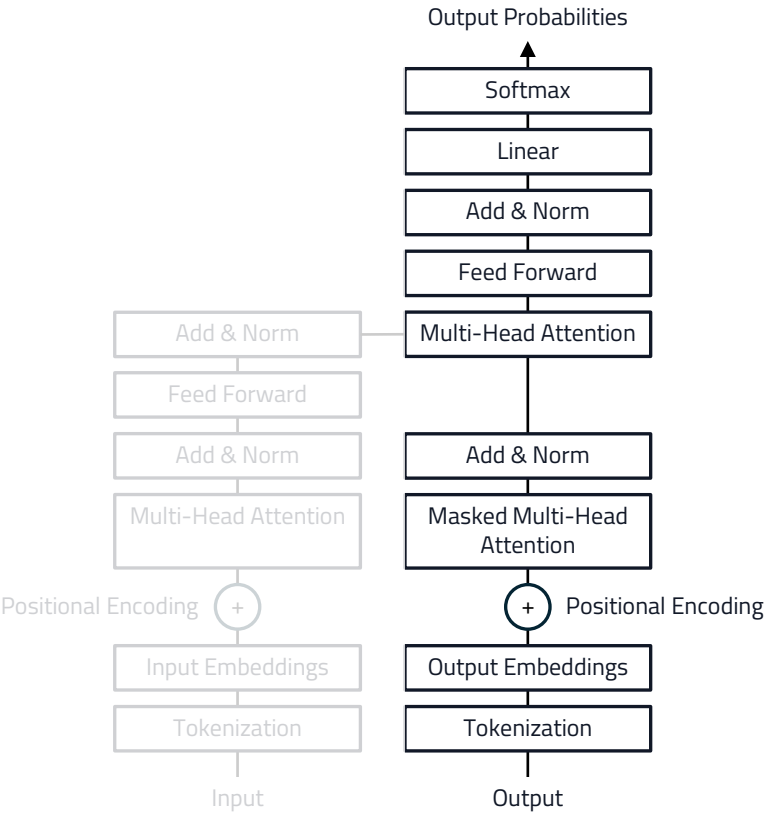
Dr. Björn E. Hommel | bjoern.hommel@uni-leipzig.de

Decoder-Only Models

Large Language Models

Since working with LLMs is quite resource-demanding, they usually require specialized hardware, such as GPUs with abundant VRAM. Options to work with LLMs are therefore limited to:

- **Remotely Hosted LLMs:** For especially large and oftentimes proprietary models (e.g., GPT-4, Claude Sonnet 4, Gemini Ultra). Usually pose problems for open-science practices due to lack of transparency (e.g., reproducibility issues from undisclosed model updates). Access via:
  - **Chat Interface**: User-friendly web applications, but permitting little to no additional control.
  - **API Access:** Programmatic access through APIs, enabling multiple sequential requests with more control over hyperparameters (e.g., temperature).
- **Locally Hosted LLMs:** Require technical expertise and hardware, but can be optimized using open-source solutions (e.g., Ollama, llama.cpp). Work with somewhat smaller but open-source models. Better suited for open-science practices.



Dr. Björn E. Hommel | bjoern.hommel@uni-leipzig.de

Models GitHub Discord Turbo

Search models

Sign in Download

Download Ollama

macOS Linux Windows

Install with one command:

```
curl -fsSL https://ollama.com/install.sh | sh
```

View script source• Manual install instructions

https://ollama.com/download/linux

Embedding Vision Tools Thinking Popular

gpt-oss

OpenAI's open-weight models designed for powerful reasoning, agentic tasks, and versatile developer use cases.

tools thinking 20b 120b

2.2M Pulls 3 Tags Updated 1 month ago

deepseek-r1

DeepSeek-R1 is a family of open reasoning models with performance approaching that of leading models, such as O3 and Gemini 2.5 Pro.

tools thinking 1.5b 7b 8b 14b 32b 70b 671b

61.7M Pulls 35 Tags Updated 2 months ago

gemma3

The current, most capable model that runs on a single GPU.

vision 270m 1b 4b 12b 27b

15.9M Pulls 26 Tags Updated 4 weeks ago

embeddinggemma

EmbeddingGemma is a 300M parameter embedding model from Google.

embedding 300m

19.4K Pulls 5 Tags Updated 4 days ago

qwen3

Qwen3 is the latest generation of large language models in Qwen series, offering a comprehensive suite of dense and mixture-of-experts (MoE) models.

https://ollama.com/search

qwen3

```
ollama run qwen3
```

8M Downloads Updated 1 month ago

Qwen3 is the latest generation of large language models in Qwen series, offering a comprehensive suite of dense and mixture-of-experts (MoE) models.

tools thinking 0.6b 1.7b 4b 8b 14b 30b 32b 235b

Models

View all →

Name	Size	Context	Input
qwen3:latest	5.2GB	40K	Text
qwen3:0.6b	523MB	40K	Text
qwen3:1.7b	1.4GB	40K	Text
qwen3:4b	2.5GB	256K	Text
qwen3:8b latest	5.2GB	40K	Text
qwen3:14b	9.3GB	40K	Text
qwen3:30b	19GB	256K	Text
qwen3:32b	20GB	40K	Text
qwen3:235b	142GB	256K	Text

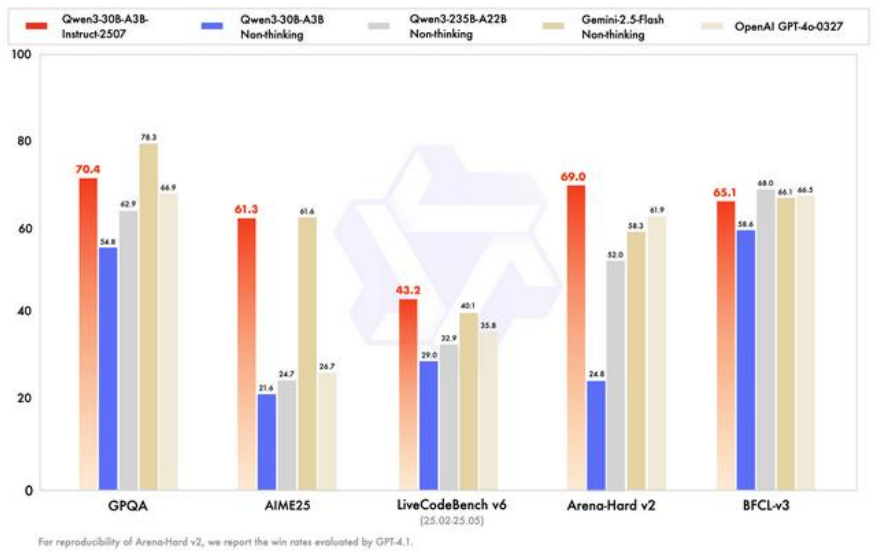
Readme

https://ollama.com/library/qwen3

Qwen 3 is the latest generation of large language models in Qwen series, with newly updated versions of the 30B and 235B models:

New 30B model

```
ollama run qwen3:30b
```



https://ollama.com/library/qwen3

Decoder-Only Models

Local LLMs with Ollama

```
bjorn@pantagruel:/home$ ollama --help
Large language model runner

Usage:
  ollama [flags]
  ollama [command]

Available Commands:
  serve       Start ollama
  create      Create a model from a Modelfile
  show        Show information for a model
  run         Run a model
  stop        Stop a running model
  pull        Pull a model from a registry
  push        Push a model to a registry
  list        List models
  ps          List running models
  cp          Copy a model
  rm          Remove a model
  help        Help about any command

Flags:
  -h, --help      help for ollama
  -v, --version   Show version information

Use "ollama [command] --help" for more information about a command.
```

Decoder-Only Models

Local LLMs with Ollama

```
bjorn@pantagruel:/home$ ollama list
NAME           ID              SIZE      MODIFIED
qwen2.5vl:7b   5ced39dfa4ba    6.0 GB    19 seconds ago
qwen3:32b      e1c9f234c6eb    20 GB     4 months ago

bjorn@pantagruel:/home$ ollama run qwen3:32b
>>> Send a message (/? for help)
```

Decoder-Only Models

Local LLMs with Ollama

```
bjorn@pantagruel:/home$ ollama list
NAME           ID              SIZE      MODIFIED
qwen2.5vl:7b   5ced39dfa4ba    6.0 GB    19 seconds ago
qwen3:32b      e1c9f234c6eb    20 GB     4 months ago

bjorn@pantagruel:/home$ ollama run qwen3:32b
>>> Why didn't I prepare my workshop slides in time and now have to spend my evenings doing last-minute preparations? I should get a dog.

<think>
Okay, let's see. The user is asking why they didn't prepare their workshop slides in time and now have to do last-minute prep, and they mention they should get a dog.

First, I need to address their main issue: procrastination. They're feeling stressed about last-minute work. The mention of getting a dog might be a way to cope with stress or
add structure to their life. But I should be careful not to assume too much. Maybe they think a dog would help with accountability or provide motivation?

I should start by acknowledging their frustration. Procrastination is a common issue, so validating their feelings is important. Then, offer practical tips for avoiding last-
minute work in the future. Breaking tasks into smaller steps, time management techniques like the Pomodoro method, and setting deadlines could help. Also, maybe suggest using
tools or apps to
stay organized.

The dog part is interesting. Getting a pet can indeed add structure and respon
```

Decoder-Only Models

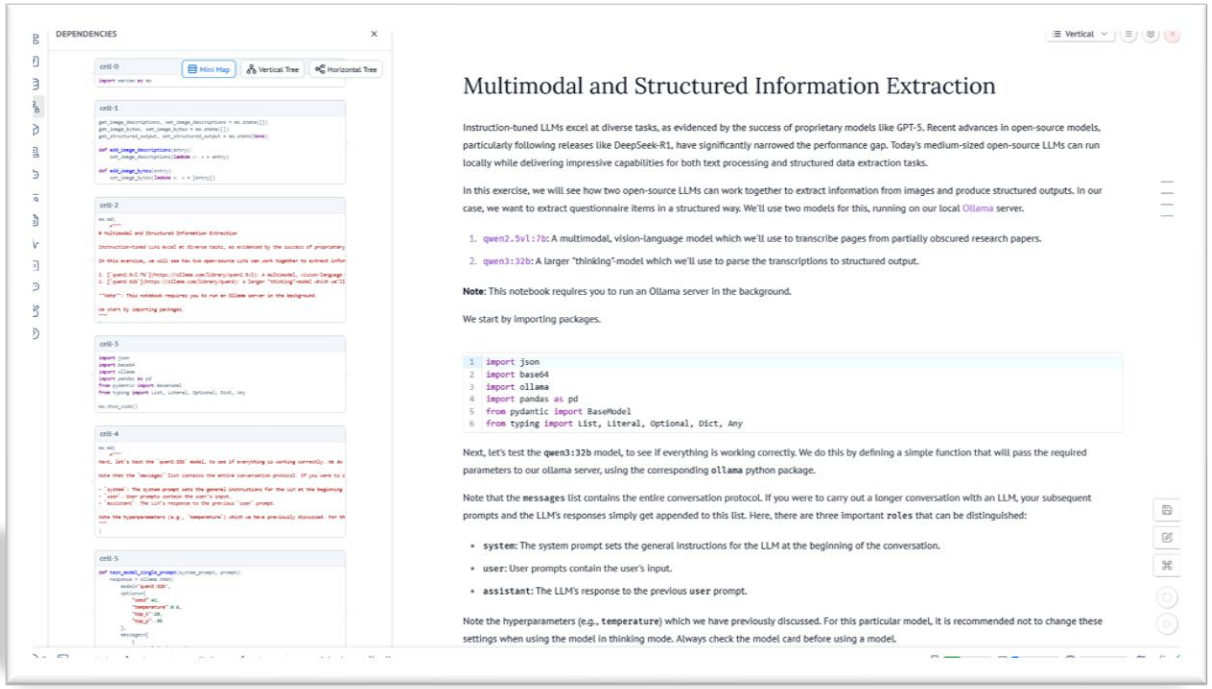
Local LLMs with Ollama

```
bjorn@pantagruel:/home$ ollama list
NAME           ID              SIZE      MODIFIED
qwen2.5vl:7b   5ced39dfa4ba    6.0 GB    19 seconds ago
qwen3:32b      e1c9f234c6eb    20 GB     4 months ago

bjorn@pantagruel:/home$ ollama run qwen3:32b
>>> curl http://localhost:11434/api/generate -d '{ "model": "qwen3:32b", "prompt": "Hi mom!" }'

{"model":"qwen3:32b","created_at":"2025-09-13T18:21:42.870953149Z","response":"\u003cthink\u003e","done":false}
{"model":"qwen3:32b","created_at":"2025-09-13T18:21:42.91005658Z","response":"\n","done":false}
{"model":"qwen3:32b","created_at":"2025-09-13T18:21:42.943907304Z","response":"Okay","done":false}
{"model":"qwen3:32b","created_at":"2025-09-13T18:21:42.977639437Z","response":",","done":false}
{"model":"qwen3:32b","created_at":"2025-09-13T18:21:43.01134133Z","response":" the","done":false}
{"model":"qwen3:32b","created_at":"2025-09-13T18:21:43.045146434Z","response":" user","done":false}
{"model":"qwen3:32b","created_at":"2025-09-13T18:21:43.078647716Z","response":" sent","done":false}
{"model":"qwen3:32b","created_at":"2025-09-13T18:21:43.112320989Z","response":" \"","done":false}
{"model":"qwen3:32b","created_at":"2025-09-13T18:21:43.145999212Z","response":"Hi","done":false}
{"model":"qwen3:32b","created_at":"2025-09-13T18:21:43.179670625Z","response":" mom","done":false}
{"model":"qwen3:32b","created_at":"2025-09-13T18:21:43.213307318Z","response":"!\"","done":false}
{"model":"qwen3:32b","created_at":"2025-09-13T18:21:43.247177012Z","response":" I","done":false}
{"model":"qwen3:32b","created_at":"2025-09-13T18:21:43.280642384Z","response":" need","done":false}
{"model":"qwen3:32b","created_at":"2025-09-13T18:21:43.314312757Z","response":" to","done":false}
{"model":"qwen3:32b","created_at":"2025-09-13T18:21:43.348213361Z","response":" respond","done":false}
{"model":"qwen3:32b","created_at":"2025-09-13T18:21:43.381679583Z","response":" in","done":false}
{"model":"qwen3:32b","created_at":"2025-09-13T18:21:43.415332236Z","response":" a","done":false}
{"model":"qwen3:32b","created_at":"2025-09-13T18:21:43.448988449Z","response":" friendly","done":false}
{"model":"qwen3:32b","created_at":"2025-09-13T18:21:43.48242695Z","response":" and","done":false}
{"model":"qwen3:32b","created_at":"2025-09-13T18:21:43.516242474Z","response":" warm","done":false}
{"model":"qwen3:32b","created_at":"2025-09-13T18:21:43.549778856Z","response":" way","done":false}
{"model":"qwen3:32b","created_at":"2025-09-13T18:21:43.583362439Z","response":".","done":false}
{"model":"qwen3:32b","created_at":"2025-09-13T18:21:43.61969496Z","response":" Since","done":false}
{"model":"qwen3:32b","created_at":"2025-09-13T18:21:43.653378043Z","response":" it","done":false}
{"model":"qwen3:32b","created_at":"2025-09-13T18:21:43.687407989Z","response":"'s","done":false}
```

Ollama automatically runs a local server which we can use for programmatic API access.

Interactive Exercise

Multimodal and Structured Information Extraction




```
poetry run marimo edit ./notebooks/07-information-extraction.py

Edit 07-information-extraction.py in your browser 📝

➜  URL: http://localhost:2720?access_token=LM1yR3eI5HjmVeyC43fRYw
```

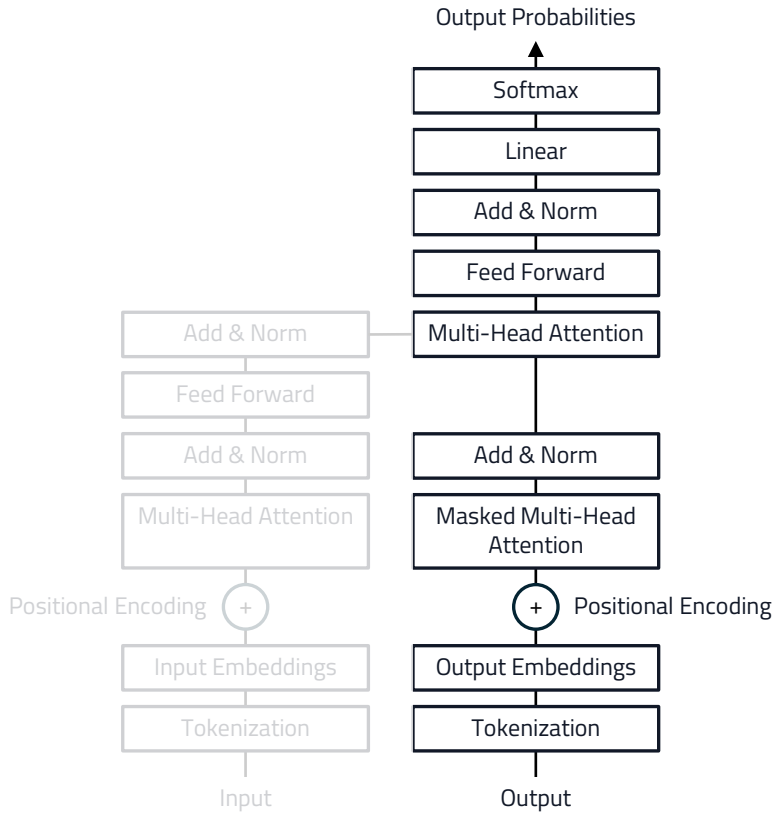

Dr. Björn E. Hommel | bjoern.hommel@uni-leipzig.de

Decoder-Only Models

Fitting large Models on small GPUs

For if you're not well-endowed with massive hardware, these techniques allow you to use larger models for inference:

- **Quantization:** Reduce model weights from 32-bit to e. g., 8-bit, thereby decreasing memory usage by 75 to 87% with minimal accuracy loss
- **Model Parallelism**: Split large models across multiple GPUs
- **Flash Attention:** Memory-efficient attention or faster token generation.

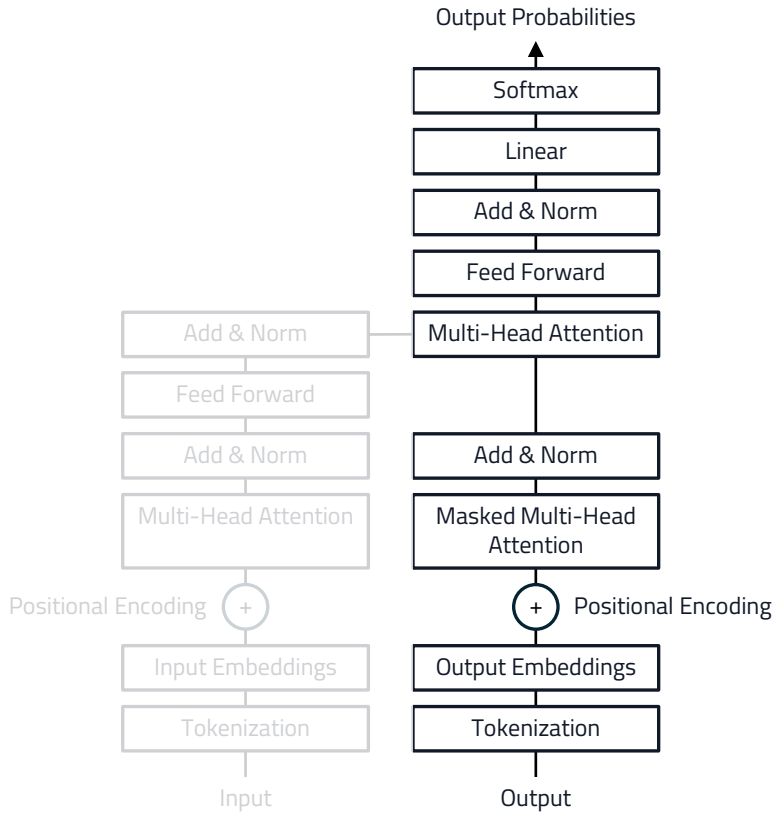


Dr. Björn E. Hommel | bjoern.hommel@uni-leipzig.de

Decoder-Only Models

Evaluating Performance

- **Perplexity:** Measures how well the model predicts the next token in a sequence. Very limited performance metric for LLMs, more abstract capabilities become relevant.
- **n-gram based metrics**: BLEU measures n-gram overlap; ROUGE on recall (sensitivity) of n-grams. Produces false negatives for paraphrased but correct answers.
- **Human Evaluation:** Ratings of fluency, coherence, relevance, etc. Costly.
- **Task-Specific Evaluations:** e.g., Code generation evaluated by functional correctness.
- **Benchmark Suites:** e.g., SuperGLUE, where models must perform well across diverse language understanding challenges, such as:
  - Corpus of Linguistic Acceptability (CoLA): Binary classification of whether English sentences are linguistically acceptable
  - Multi-Genre Natural Language Inference (MNLI): Textual entailment across multiple genres (entailment, contradiction, neutral)

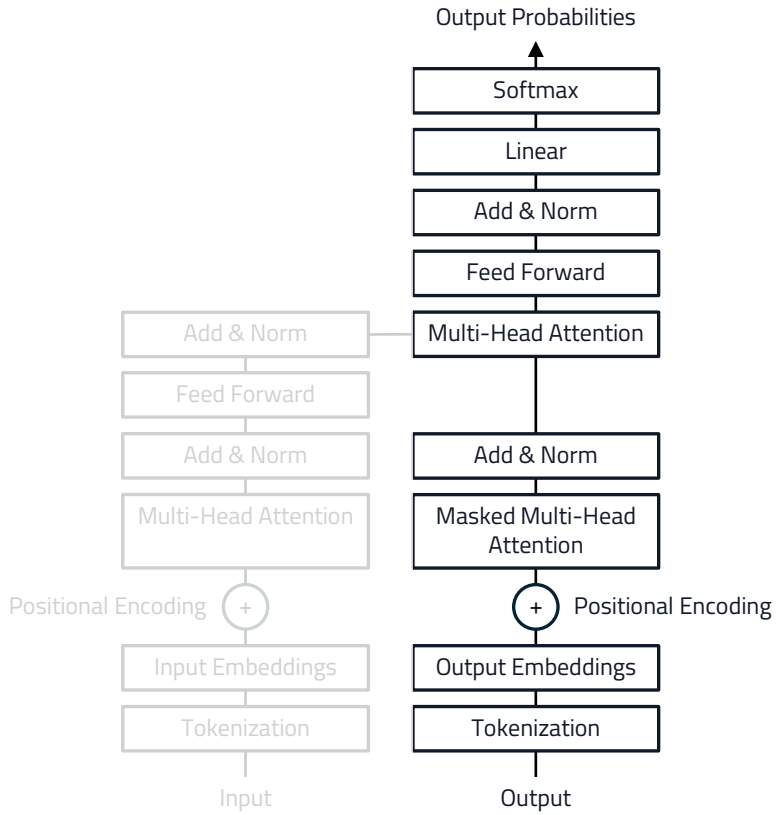


(Reiter, 2018; Sarlin et al., 2020)

Decoder-Only Models

Evaluating Performance

- **Demographic Biases:** e.g., gender/sex, racial/ethnic biases.
  - Benchmarks, e.g., StereoSet
  - Embedding analysis: e.g., Higher similarity between certain professions and gender token.
  - Template-based approaches: e.g., "In this story, assign roles: doctor, patient, nurse" with different names.

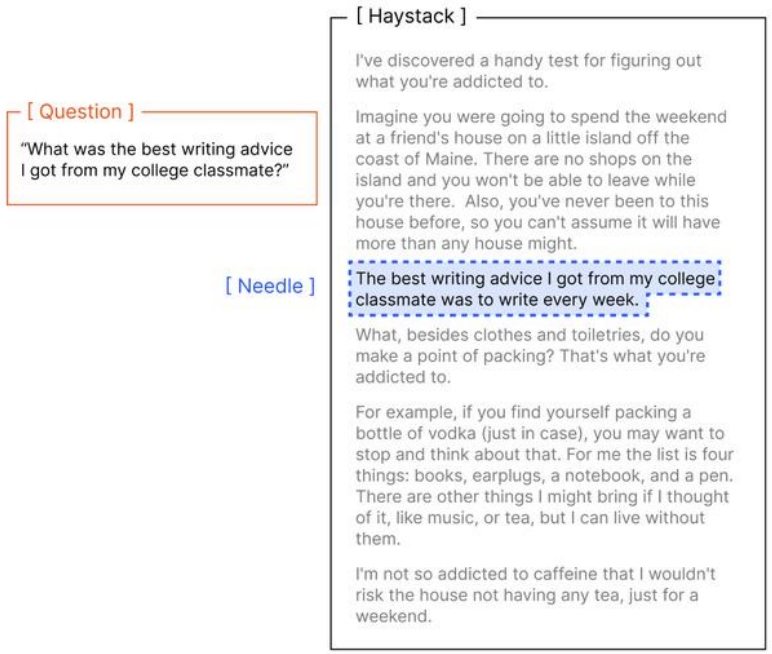


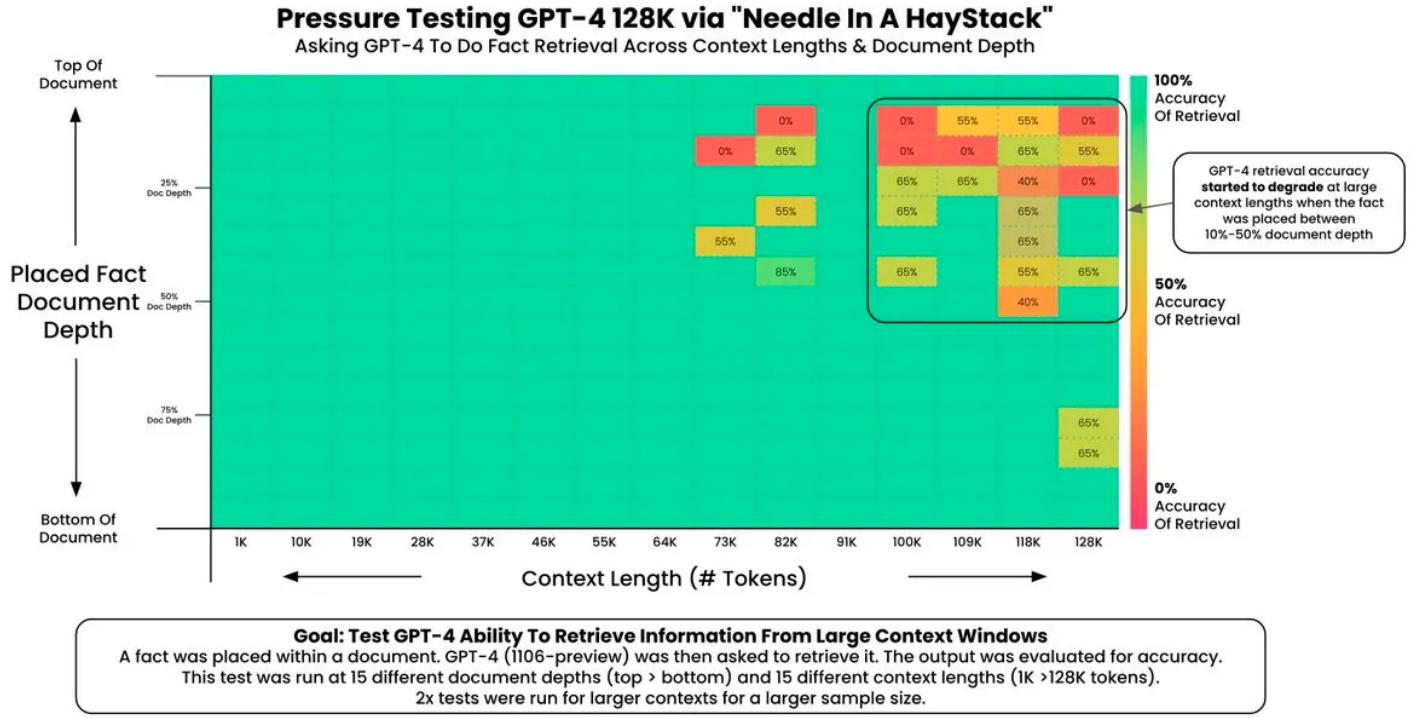
https://huggingface.co/blog/evaluating-llm-bias; Nadeem et al. (2020)

Decoder-Only Models

Evaluating Performance

Context Rot: Pattern of performance degradation in LLMs that occurs with longer context.





https://medium.com/data-science/the-needle-in-a-haystack-test-a94974c1ad38; Hong et al. (2025)

Text Arena

View rankings across various LLMs on their versatility, linguistic precision, and cultural context across text.

Last Updated	Total Votes	Total Models
Sep 8, 2025	4,075,191	239

Overall | Search by model name... / | Default

Rank (UB)	Model	Score	95% CI (±)	Votes	Organization	License
1	gemini-2.5-pro	1455	±5	41.731	Google	Proprietary
1	claude-opus-4-1-20250805-thinking-16k	1451	±6	11.750	Anthropic	Proprietary
2	o3-2025-04-16	1444	±4	43.898	OpenAI	Proprietary
2	gpt-5-high	1442	±6	15.076	OpenAI	Proprietary
2	chatgpt-4o-latest-20250326	1441	±4	36.426	OpenAI	Proprietary
3	gpt-4.5-preview-2025-02-27	1439	±6	15.271	OpenAI	Proprietary
3	claude-opus-4-1-20250805	1438	±6	18.341	Anthropic	Proprietary
5	gpt-5-chat	1430	±6	11.808	OpenAI	Proprietary
6	qwen3-max-preview	1428	±7	8.781	Alibaba	Proprietary
8	grok-4-0709	1422	±5	21.446	xAI	Proprietary
8	claude-opus-4-20250514-thinking-16k	1420	±5	27.351	Anthropic	Proprietary
8	kimi-k2-0711-preview	1420	±5	23.527	Moonshot	Modified MIT
8	...en3-235b-a22b-instruct-2507	1420	±6	17.947	Alibaba	Apache 2.0
8	deepseek-v3.1-thinking	1419	±8	8.226	DeepSeek	MIT

https://lmarena.ai/leaderboard

open-llm-leaderboard's Collections

Details

Open LLM Leaderboard 2

Open LLM Leaderboard best models ❤️

The Big Benchmarks Collection

The Big Benchmarks Collection

updated Nov 18, 2024

Gathering benchmark spaces on the hub (beyond the Open LLM Leaderboard)

Open LLM Leaderboard 🏆 — 13.5k
Track, rank and evaluate open LLMs and chatbots

Note 📐 The 🤗 Open LLM Leaderboard aims to track, rank and evaluate open LLMs and chatbots.
🤗 Submit a model for automated evaluation on the 🤗 GPU cluster on the "Submit" page!

MTEB Leaderboard 🥇 — 6.41k
Embedding Leaderboard

Note Massive Text Embedding Benchmark (MTEB) Leaderboard.

LMArena Leaderboard 🏆 — 4.62k
Display LMArena Leaderboard

Note 🏆 This leaderboard is based on the following three benchmarks:
Chatbot Arena - a crowdsourced, randomized battle platform. We use 70K+ user votes to compute Elo ratings.
MT-Bench - a set of challenging multi-turn questions. We use GPT-4 to grade the model responses.
MMLU (5-shot) - a test to measure a model's multitask accuracy on 57 tasks.

LLM-Perf Leaderboard 🏆 — 561
Explore hardware performance for LLMs

Note The 🤗 LLM-Perf Leaderboard 🏋️ aims to benchmark the performance (latency, throughput & memory) of Large Language Models (LLMs) with different hardwares, backends and optimizations using Optimum-Benchmark and Optimum flavors.
Anyone from the community can request a model or a hardware/backend/optimization configuration for automated benchmarking:

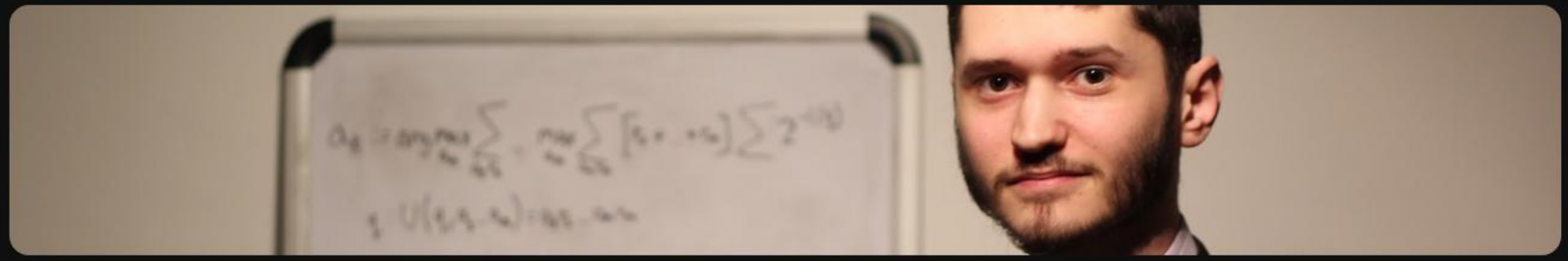
Search and submit code models for evaluation

Upvote 248 +244

Share collection
View history
Collection guide
Browse collections

https://huggingface.co/collections/open-llm-leaderboard/the-big-benchmarks-collection-64faca6335a7fc7d4ffe974a

Home
Shorts
Subscriptions
You
Downloads



Robert Miles AI Safety

@RobertMilesAI · 166K subscribers · 53 videos

Videos about Artificial Intelligence Safety Research, for everyone. ...more

patreon.com/robertskmiles and 3 more links

Subscribed

Home Videos Shorts Playlists Posts

Latest Popular Oldest

Tech is Good, AI Will Be Different

57K views • 3 weeks ago

AI Safety Career Advice! (And So Can You!)

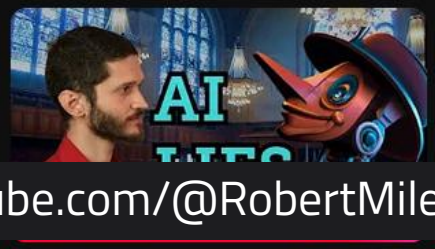
65K views • 4 months ago

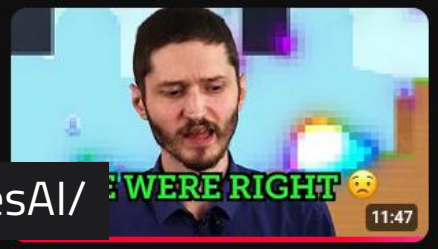
Using Dangerous AI, But Safely?

140K views • 10 months ago

AI Ruined My Year

263K views • 1 year ago

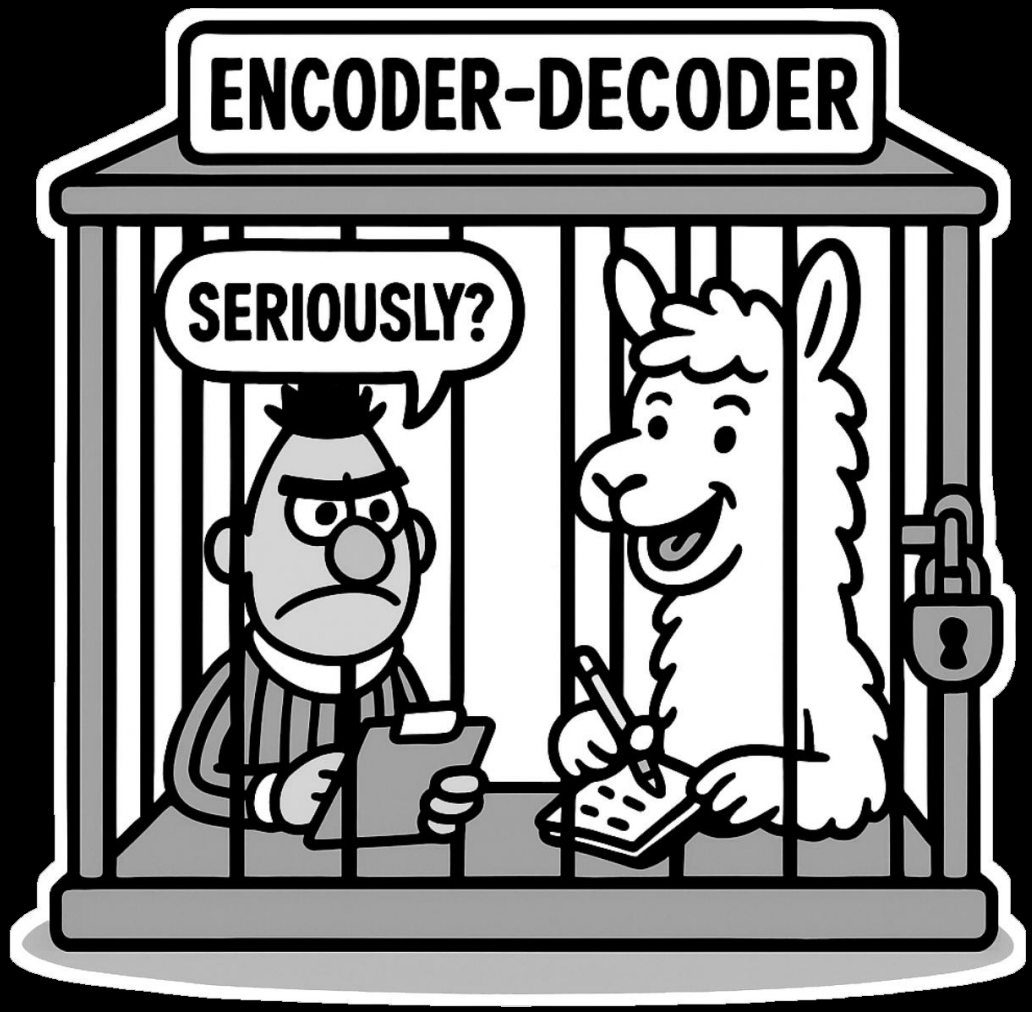




https://www.youtube.com/@RobertMilesAI/

Part V

Encoder-Decoder Models

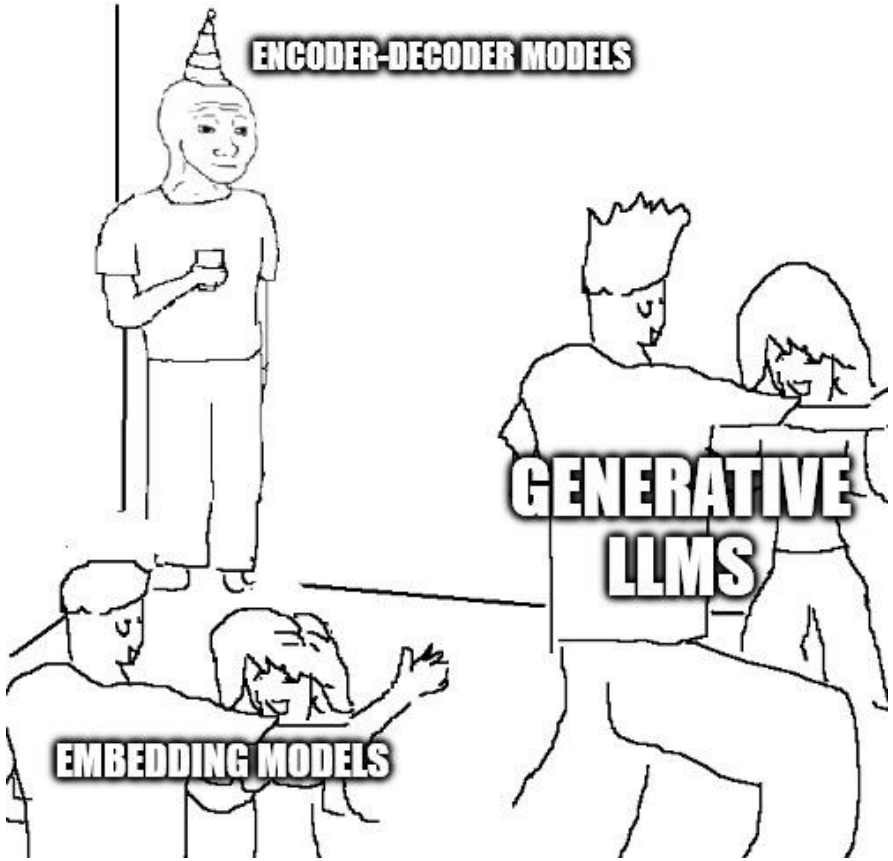


Encoder-Decoder Models

From Rising Star to Black Sheep?

Although encoder-decoder models constituted the original transformer model architecture, industry and research have shifted attention to decoder-only models.

- Encoder-Decoder models may still outperform decoder-only models on specific tasks.
  - But not necessarily in practice, because of the massive scaling of decoder models.
- Encoder-Decoder models are immune to **context degradation**, as each predicted token is conditioned on the encoder-representation.

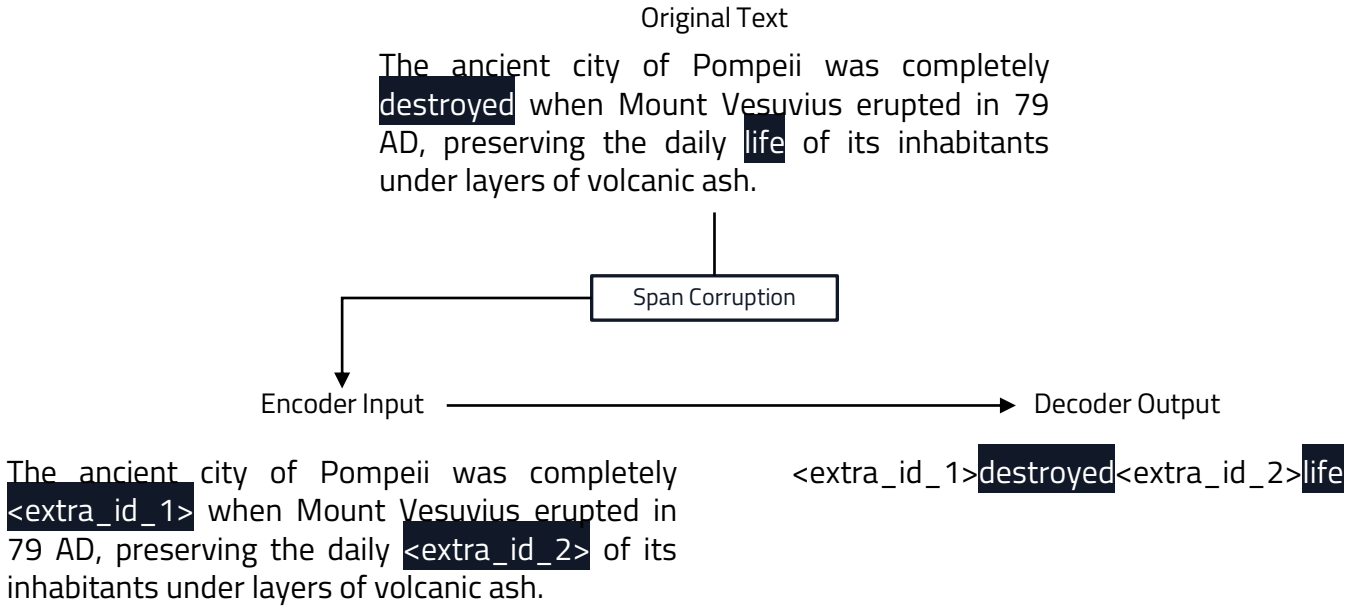


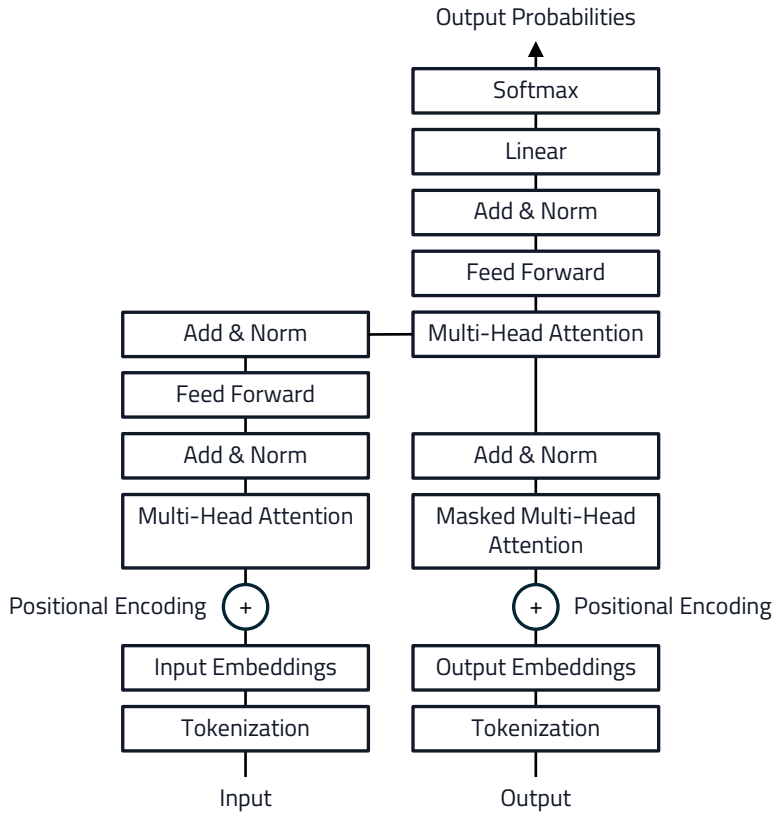
https://developers.googleblog.com/en/t5gemma/

Encoder-Decoder Models

Example of Span Corruption as a Training Objective

In contrast to encoder-only models, encoder-decoder models can predict bi-directionally predict multiple tokens as demonstrated by T5.

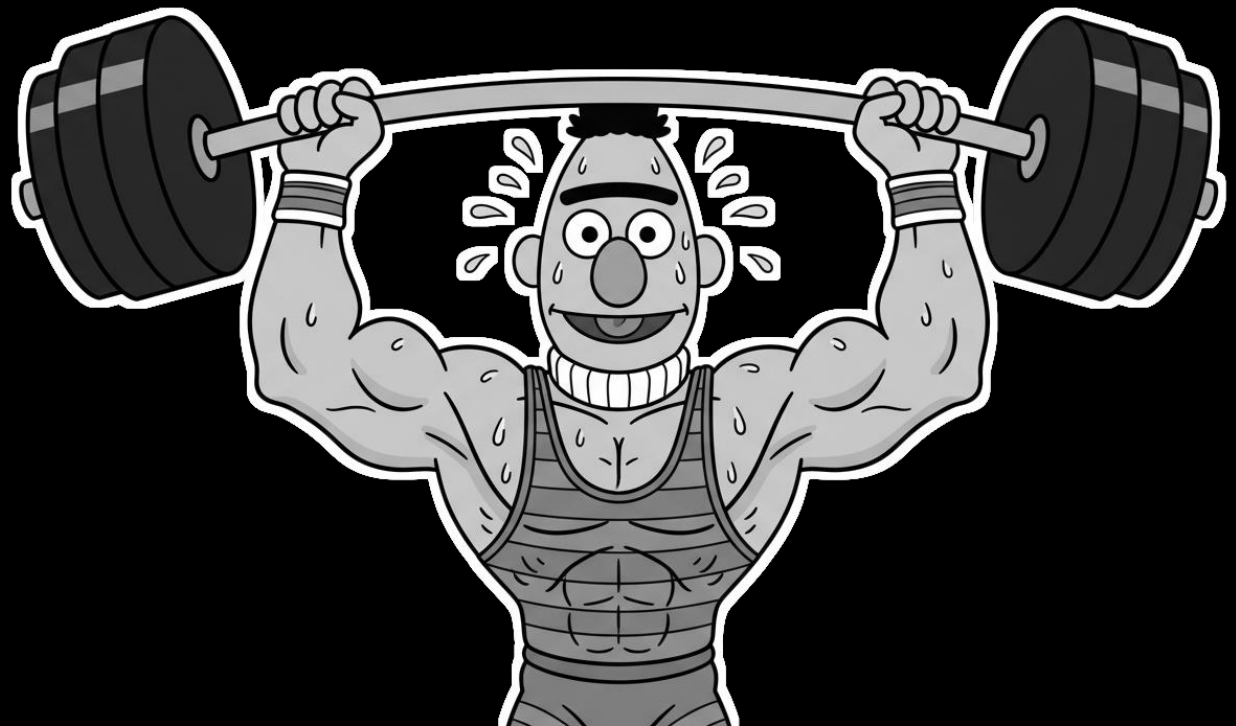




Raffel et al. (2020)

Part VI

Training Transformer Models



Training Transformer Models

Checklist

Questions to ask yourself, before training a model:

Do I need to train a model from scratch?

- Yes (but in most cases no), e.g., if you need to use an entirely different vocabulary.
- No, domain adaptation or fine-tuning will suffice.
    - Which pre-trained model to pick as a base?

How many training stages?

- What is the primary training objective in each training stage?
    - e. g., masked language modeling, text classification […]
    - decide on loss functions
- Train all the parameters or freeze layers?

What are my data requirements?

- Do I need labeled (i.e., supervised learning) or unlabeled data (autoregressive/unsupervised learning).
- What are the requirements on data quality (e.g., train on synthetic data, evaluate on human labeled data)
- Data integrity and leakage

Training Transformer Models

Checklist

Questions to ask yourself, before training a model:

What are my computational constraints?

- Sufficient GPU VRAM for local training vs. remote GPU training (e.g., Google Colab).
- Use parameter-efficient training methods (e.g., Low-Rank Adaptation [LoRA]).
    - Which pre-trained model to pick as a base?

How to decide on Hyperparameters and training configuration?

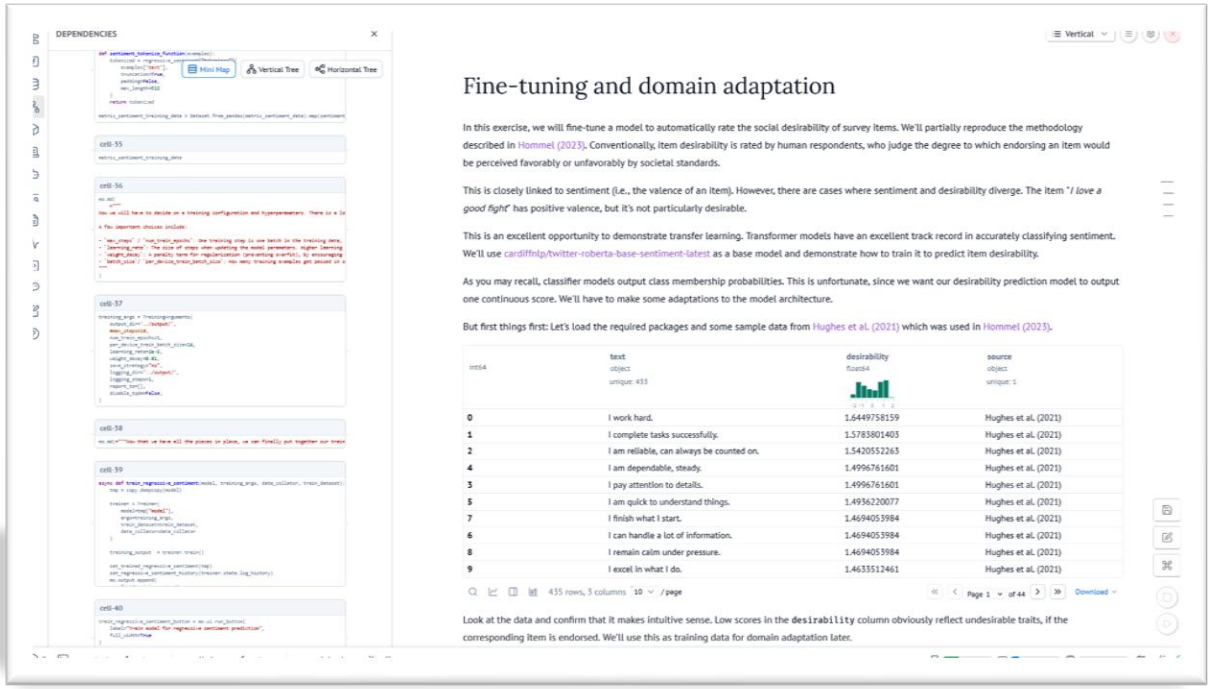
- Informed decisions vs. hyperparameter search (e.g., Optuna)

How to evaluate performance?

- e. g., holdout validation (out-of-sample testing), with train-dev-test-split.
- metrics depend on your task

Interactive Exercise

Fine-tuning and Domain Adaptation



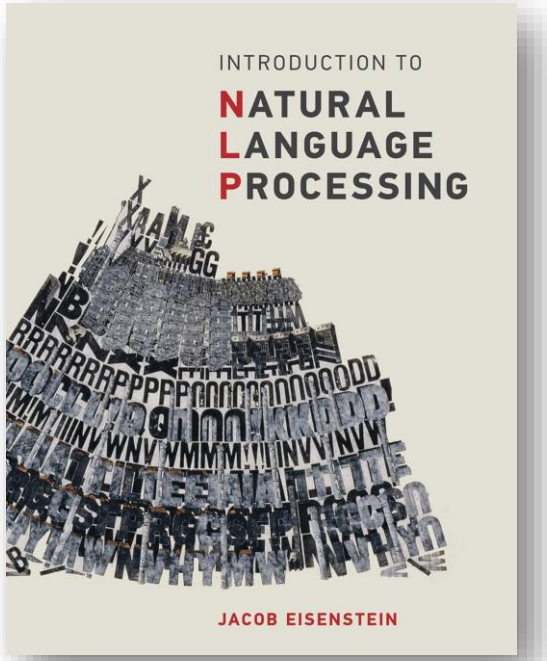

```
poetry run marimo edit ./notebooks/08-fine-tuning.py

Edit 08-fine-tuning.py in your browser

➜  URL: http://localhost:2720?access_token=LM1yR3eI5HjmVeyC43fRYw
```

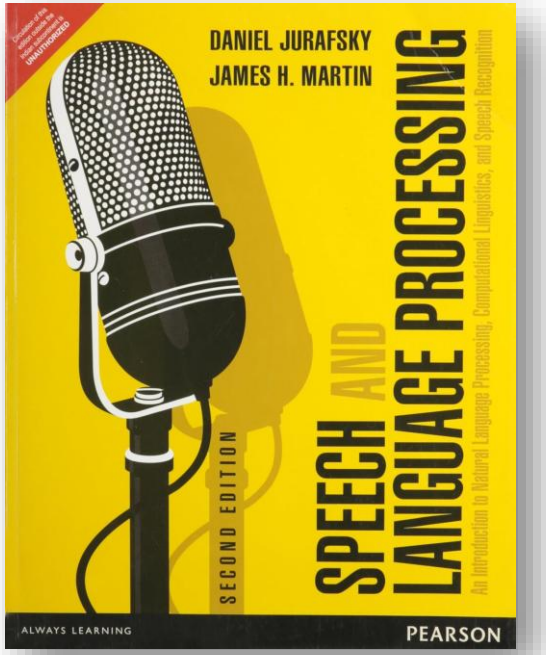
Appendix

Appendix

Recommended Literature

Natural Language Processing (Fundamentals)





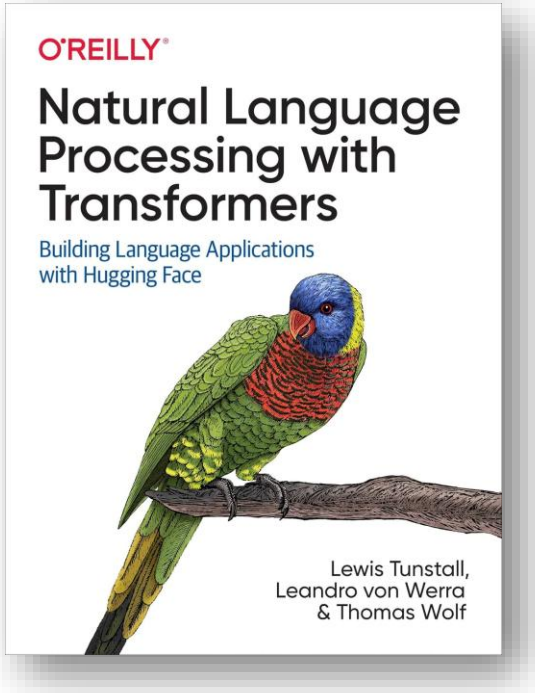
Eisenstein, J. (2018). *Natural Language Processing*. MIT Press.

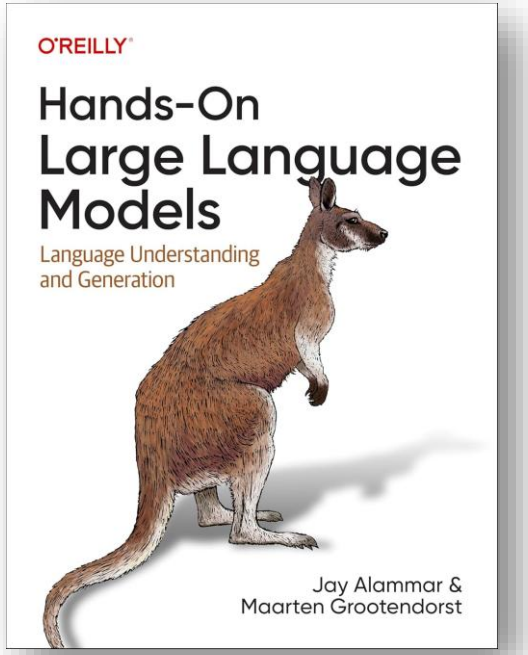
Jurafsky, D., & Martin, J. H. (2019). *Speech and Language Processing*. [Free Version: https://web.stanford.edu/~jurafsky/slp3/]

Appendix

Recommended Literature

Hugging Face and Transformers (Practical)





Alammar, J., & Grootendorst, M. (2024). *Hands-on large language models: Language understanding and generation* (First edition). O'Reilly Media, Inc. [Free Version: https://github.com/HandsOnLLM/Hands-On-Large-Language-Models]

Tunstall, L., Werra, L. von, Wolf, T., & Géron, A. (2022). *Natural language processing with Transformers: Building language applications with Hugging Face* (First edition). O'Reilly.

Appendix

Recommended Literature

Tutorials for NLP in the Behavioral Sciences

A tutorial on open-source large language models for behavioral science

Zak Hussain[1,2], Marcel Binz[3,4], Rui Mata[1], and Dirk U. Wulff[#,1]

[1]University of Basel
[2]Max Planck Institute for Human Development
[3]Max Planck Institute for Biological Cybernetics
[4]Helmholtz Center for Computational Health

Large language models (LLMs) have the potential to revolutionize behavioral science by accelerating and improving the research cycle, from conceptualization to data analysis. Unlike closed-source solutions, open-source frameworks for LLMs can enable transparency, reproducibility, and adherence to data protection standards, which gives them a crucial advantage for use in behavioral science. To help researchers harness the promise of LLMs, this tutorial offers a primer on the open-source Hugging Face ecosystem and demonstrates several applications that advance conceptual and empirical work in behavioral science, including feature extraction, fine-tuning of models for prediction, and generation of behavioral responses. Executable code is made available at github.com/Zak-Hussain/LLM4BeSci.git. Finally, the tutorial discusses challenges faced by research with (open-source) LLMs related to interpretability and safety and offers a perspective on future research at the intersection of language modeling and behavioral science.

Introduction

Large language models (LLMs)—machine learning systems trained on vast amounts of inputs that can analyze and generate meaningful text—are increasingly being used in science (Van Noorden & Perkel, 2023) and promise to change the scientific workflow in numerous ways. They are already used across all aspects of the research cycle (MacKay & Oldford, 2000), from conceptualization to data analysis. In psychology (Demszky et al., 2023) and related disciplines (Korinek, 2023), LLMs are being used to automate research processes, predict human judgments, and run in silico behavioral experiments.

Scientific applications of LLMs require high levels of transparency and reproducibility (Buckheit et al., 2023). In addition, many applications in behavioral science involve sensitive information (e.g., personal or health data) or target vulnerable populations (e.g., children) and thus require specific data protection protocols. Open-source frameworks that provide full transparency and respect privacy requirements are therefore indispensable for applications of LLMs in behavioral science.

We aim to help advance the responsible use of LLMs in behavioral science by providing a comprehensive tutorial on applications using an open-source framework that maximizes transparency, reproducibility, and data privacy. Specifically, we provide a primer on the Hugging Face ecosystem, covering several applications of LLMs, including conceptual clarification, prediction of behavioral outcomes, and generation of human-like responses. Our target audience consists of behavioral researchers with a basic knowledge of programming principles who are interested in adding LLMs to their workflows. We hope that this resource will help researchers in psychology and related disciplines to adopt LLMs for a wide range of tasks.

In what follows, we first provide a short primer on transformer-based language models. Second, we consider applications of LLMs in behavioral science and introduce the Hugging Face ecosystem. Third, we present three areas of application—feature extraction, fine-tuning, and text generation—and present several use cases in behavioral research. Finally, we address some advantages and limitations of current open-source approaches, and consider future directions at the intersection of LLMs and behavioral research.

A Primer on Transformer-Based Language Models

Today's LLMs are based on the transformer architecture, which is a family of neural network models that draw on a common set of building blocks. In this section, we first introduce these building blocks in sequence (see Figure 1) before discussing important architectural variants and ways of applying LLMs in behavioral science.

Tokenizers

When text input is fed into an LLM, it is first broken up into digestible pieces known as tokens. This decomposition is carried out by the LLM's tokenizer, which is a model in its own right. For instance, the *WordPiece* tokenizer (Wu et al., 2016) underlying the popular class of BERT models (Bidirectional Encoder Representations from Transformers; Devlin et al., 2018) breaks up the sentence "Open-source LLMs

Hussain, Z., Binz, M., Mata, R., & Wulff, D. U. (2023). *A tutorial on open-source large language models for behavioral science*.
https://osf.io/f7stn

Appendix

If you insist on working in R (please don't)