CHAPTER 5

Natural Language Processing: Transformers & Large Language Models

Summary

1. Limitations of Pre-Transformer Models (RNNs, LSTMs)
2. Transformers
   - Introduction: Attention Is All You Need
   - Architecture of Transformers
   - Attention Mechanism
   - Multi-Head Attention
   - Training and Inference of Transformers
   - Why Do Transformers Work So Well?
   - Applications of Transformers
3. Transformer-based Language Models Architectures
   - BERT
   - GPT
   - BART
   - T5
4. Large Language Models (LLMs)

Historical evolution to Transformers & LLM

- **Count-based models (Bag-of-Words, TF-IDF)**: Limitations in capturing word order and context.
- **Word Embeddings (Word2Vec, GloVe):** These models represent words as dense vectors, capturing semantic relationships. Emphasize that they are still non-contextualized models.
- **Recurrent Neural Networks (RNNs) and LSTMs:** Introduce the idea of processing word sequences considering the order. Mention the limitations of RNNs for long sequences and how LSTMs address them.
- **Attention Mechanism:** Introduce the concept of attention as a way to weigh the importance of different words in a sequence. Explain how the attention mechanism helps models focus on the relevant parts of the text.
- **Context:** The arrival of Transformers: Explain that Transformers, based on the attention mechanism, revolutionized NLP by allowing parallel processing and capturing long-distance dependencies.

Limitations of Pre-Transformer Models (RNNs, LSTMs)

- RNNs were a significant step forward in NLP because they could process sequential data, unlike traditional feedforward networks.
- However, they came with their own set of challenges, which ultimately paved the way for the Transformer architecture.

1. Vanishing and Exploding Gradients
2. Sequential Processing
3. Difficulty Capturing Long-Range Dependencies
4. Contextual Understanding Limitations

Vanishing and Exploding Gradients

Explanation:

- RNNs process sequences step-by-step, using information from previous time steps to inform the current prediction.
- During training, the gradients (used to update the model's weights) are propagated back through time. In long sequences, these gradients can either shrink exponentially (vanishing) or grow exponentially (exploding), making it difficult to learn long-range dependencies.

Example:

- Imagine trying to predict the last word in the sentence:
- "The cat, which was sitting on the mat in the sunny garden, _________.“
- An RNN would struggle to remember the initial context ("cat") when reaching the end of the sentence, making it difficult to predict a word like "purred" or "slept.“
- The information about the cat might have "vanished" during backpropagation.
- Conversely, in some cases, a minor change in the early part of the sequence could disproportionately affect the later predictions, a sign of exploding gradients.

Sequential Processing

Explanation:

- RNNs process input sequentially, one word at a time.
- his prevents parallelization, making them significantly slower than models that can process the entire sequence concurrently.

Example:

- Think of reading a book word by word versus being able to grasp the meaning of an entire page at once.
- RNNs are like reading word by word, while Transformers are closer to grasping the whole page.
- This sequential nature becomes a major bottleneck when dealing with long documents or large datasets.

Difficulty Capturing Long-Range Dependencies

Explanation:

- While LSTMs and GRUs were designed to mitigate the vanishing gradient problem to some extent through gating mechanisms, they still struggle to capture relationships between words that are far apart in a sequence.
- The "memory" of these networks can fade over long distances.

Example:

- Consider the sentence: "The scientists, who had been working on the project for years, finally published their groundbreaking results.“
- Understanding the connection between "scientists" and "published" requires retaining information over a long distance.
- While LSTMs might perform better than basic RNNs, they can still struggle with these long-range dependencies, especially in very long sequences.

Contextual Understanding Limitations

Explanation:

- While LSTMs can consider some context, they primarily focus on the preceding sequence.
- They don't capture the full bidirectional context (words both before and after) as effectively as Transformers.
- RNNs and LSTMs traditionally relied on static word embeddings (like Word2Vec or GloVe), which assign a single vector representation to each word, regardless of its context. This creates problems with words that have multiple meanings (polysemy).

Example:

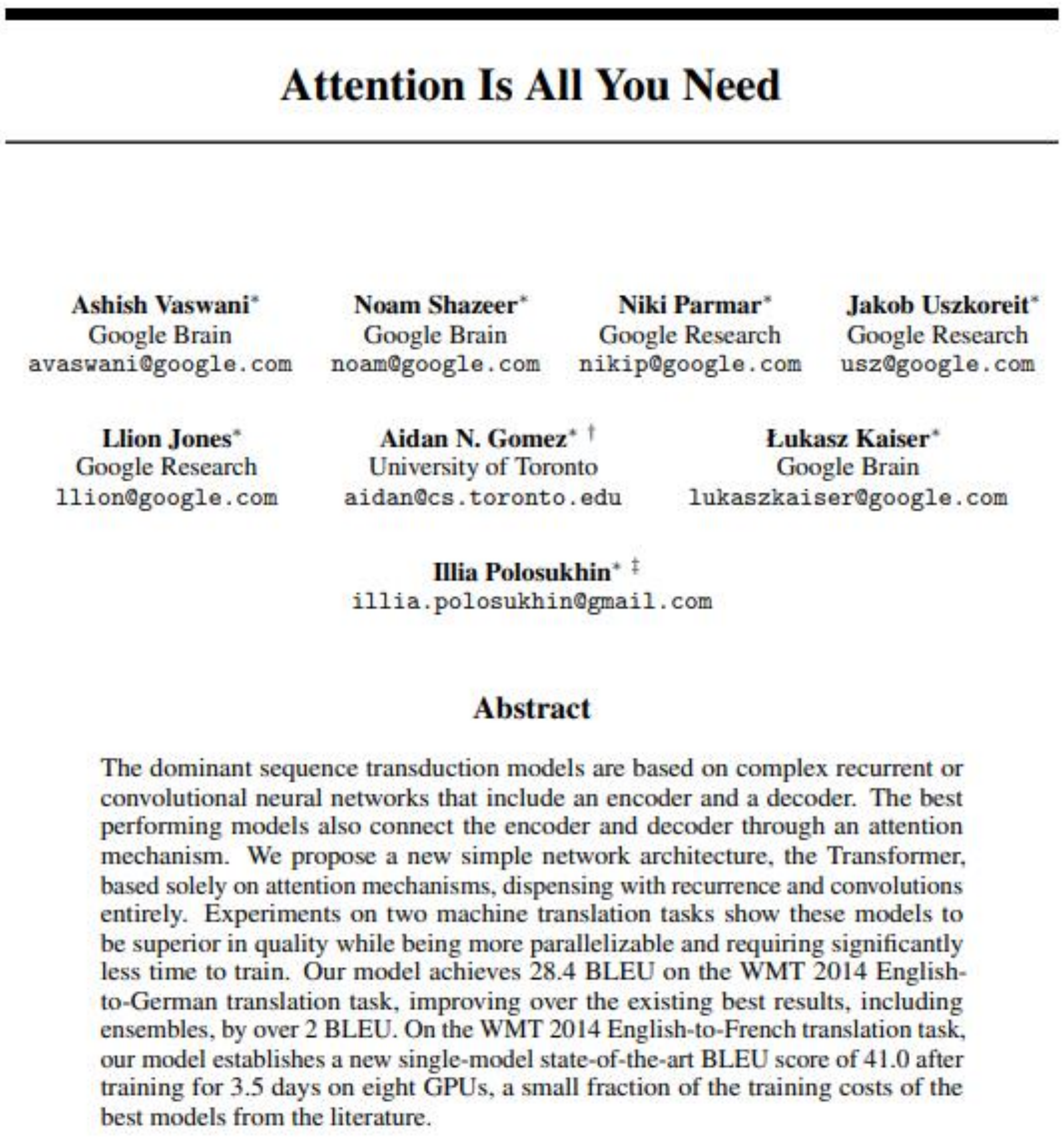
- The word "bank" can have different meanings depending on the surrounding words:
  - "I sat on the bank of the river.“
  - "I went to the bank to deposit money.“
- While LSTMs can use the preceding words to disambiguate to some extent, they miss out on the information provided by the words following "bank.".
- There is not bidirectionality in RNN/LSTM (Transformers have bidirectionality)

Contextual Understanding Limitations

- Polysemy in Spanish
  - Me regalaron una muñeca que llora.
  - Rafa Nadal se lesionó la muñera
- Polysemy in French
  - Demain je vais au bureau des impôts
  - J'ai un ordinateur portable sur mon bureau
- Polysemy cross-confussion German/English
  - A Gymnasium is a German school
  - I exercise in a Gymnasium

TRANSFORMERS

Introduction: Attention Is All You Need



Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser*
Google Brain
lukaszkaiser@google.com

Illia Polosukhin* ‡
illia.polosukhin@gmail.com

Abstract

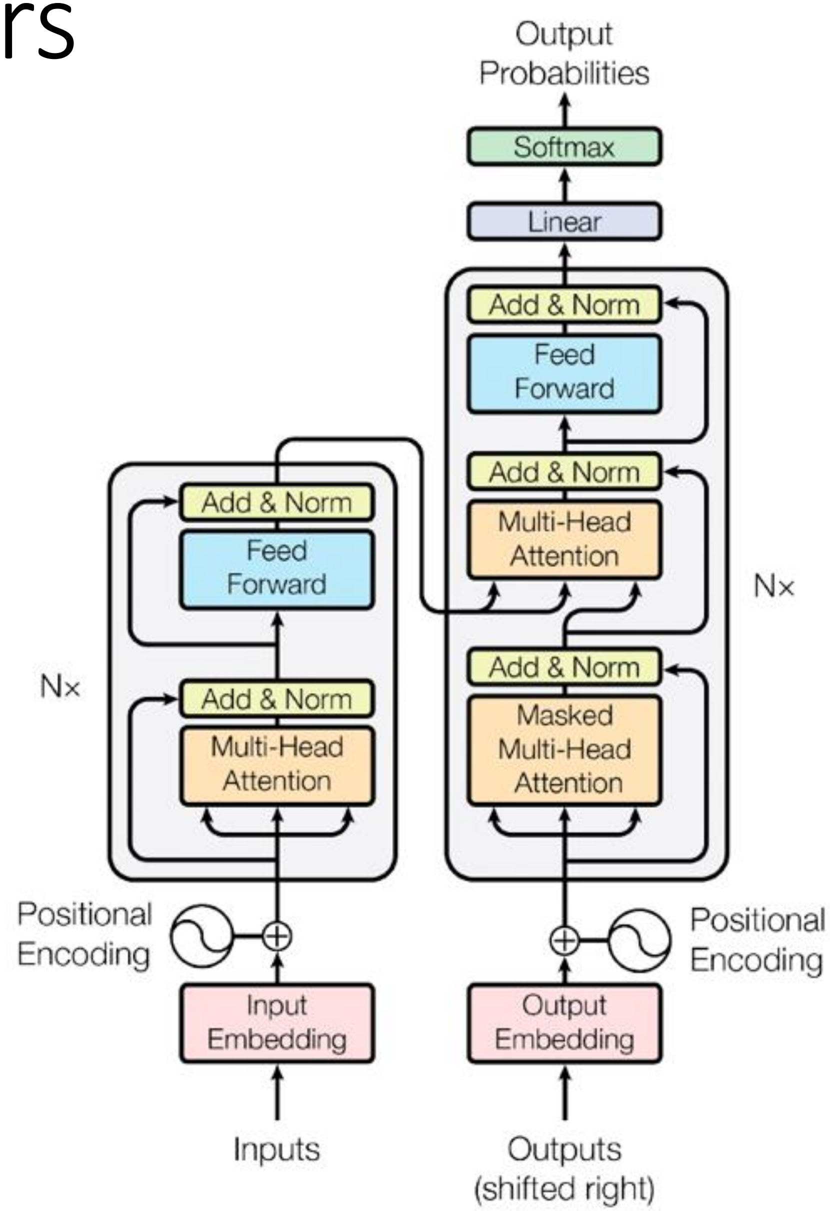
The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

Introduction

- Transformers have revolutionized the field of Natural Language Processing (NLP) and have extended their influence to other areas of artificial intelligence.
- Transformers surpassed the limitations of previous architectures like Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs), offering significant improvements in efficiency and performance.
- Why Are They Important?
  - **Parallelization:** Unlike RNNs, which process data sequentially, Transformers allow for parallel processing, accelerating training times.
  - **Capturing Long-Range Dependencies:** Their attention mechanism can capture relationships between words that are far apart in a sequence.
  - **Versatility:** Beyond NLP, Transformers have been successfully applied in computer vision, music generation, and more.

Architecture of Transformers

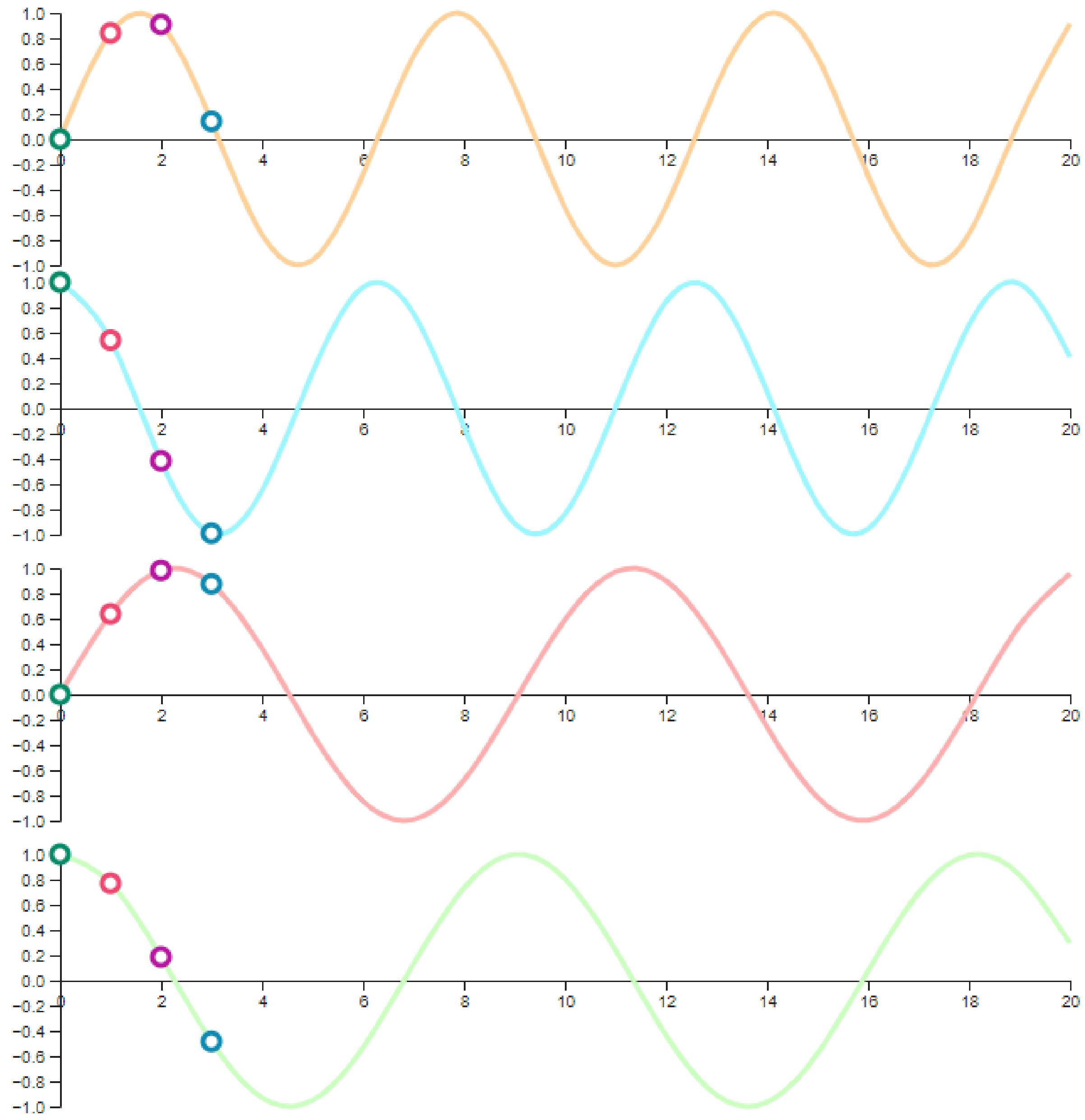
- The basic architecture of a Transformer consists of two main parts:
  - the Encoder and
  - the Decoder
- Each part is composed of a series of identical layers that process information sequentially.



Architecture of Transformers

- Embeddings
  - Input Embedding: Converts the words in the input sequence into dense vectors that capture the meaning of each word.
  - Output Embedding: Similar to Input Embedding but for the output sequence.
- Positional Encoding
  - Since Transformers do not process words sequentially, it is necessary to add information about the position of each word in the sequence.
  - This is achieved through Positional Encoding, which incorporates positional information into the embedding vectors.

Positional encoding visualization



p0	p1	p2	p3	
0.000	0.841	0.909	0.141	i=0
1.000	0.540	-0.416	-0.990	i=1
0.000	0.638	0.983	0.875	i=2
1.000	0.770	0.186	-0.484	i=3

Positional Encoding

$$PE_{(pos,2i)} = sin(\frac{pos}{10000^{2i/d_{\text{model}}}})$$

$$PE_{(pos,2i+1)} = cos(\frac{pos}{10000^{2i/d_{\text{model}}}})$$

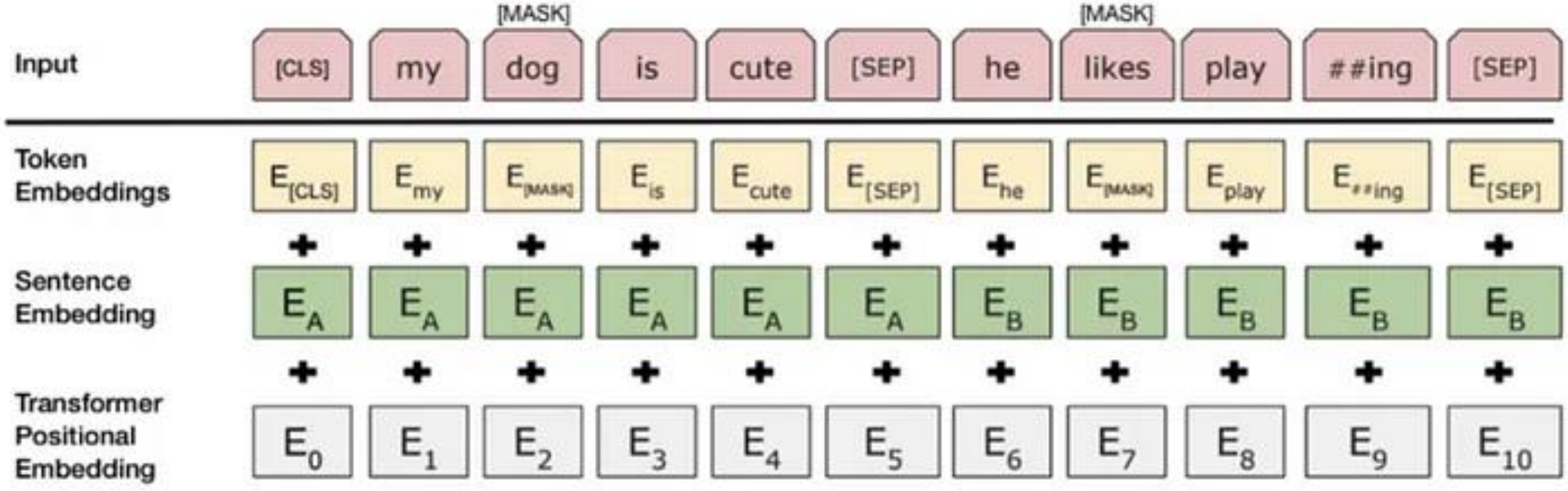
Settings: d = 50

The value of each positional encoding depends on the *position* (*pos*) and *dimension* (*d*). We calculate result for every *index* (*i*) to get the whole vector.

pos0 ● pos1 ● pos2 ● pos3 ●

Example of embeddings in BERT

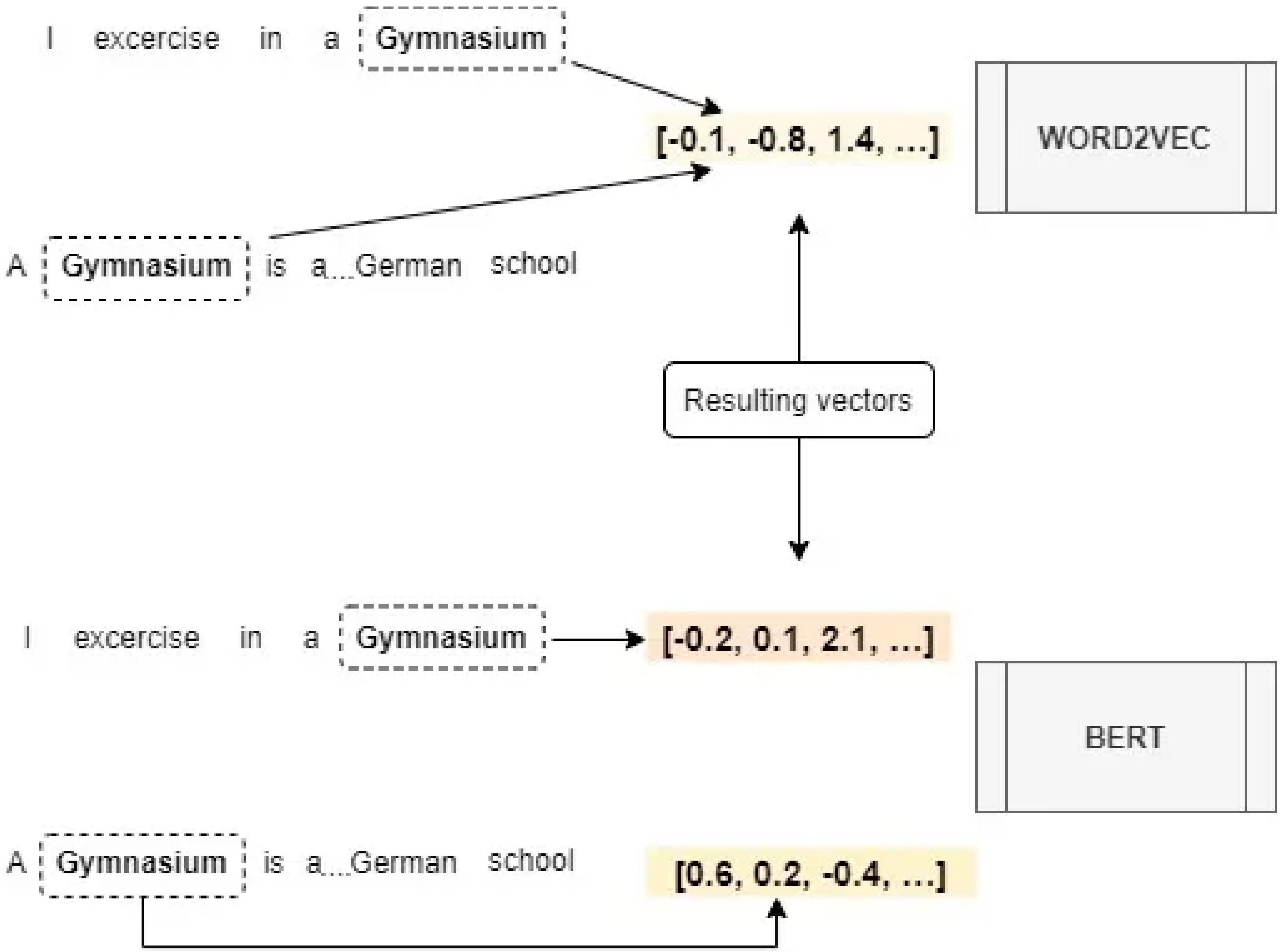
- The input representation to the BERT model is the sum of token embeddings, segmentation embeddings & position embeddings & follows a masking strategy for the model to predict the correct word in the context.



			[MASK]					[MASK]			
Input	[CLS]	my	dog	is	cute	[SEP]	he	likes	play	##ing	[SEP]
Token Embeddings	$E_{[CLS]}$	E_{my}	$E_{[MASK]}$	E_{is}	E_{cute}	$E_{[SEP]}$	E_{he}	$E_{[MASK]}$	E_{play}	$E_{\#\#ing}$	$E_{[SEP]}$
	+	+	+	+	+	+	+	+	+	+	+
Sentence Embedding	E_A	E_A	E_A	E_A	E_A	E_A	E_B	E_B	E_B	E_B	E_B
	+	+	+	+	+	+	+	+	+	+	+
Transformer Positional Embedding	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}

Source: BERT *[Devlin et al.,* 2018], with modifications

Comp.


I excercise in a Gymnasium
[-0.1, -0.8, 1.4, ...]
WORD2VEC
A Gymnasium is a...German school
Resulting vectors
I excercise in a Gymnasium
[-0.2, 0.1, 2.1, ...]
BERT
A Gymnasium is a...German school
[0.6, 0.2, -0.4, ...]

Architecture of Transformers

Encoder Stack

Each layer of the Encoder contains:

1. Multi-Head Self-Attention: Allows each word to attend to all other words in the sequence.
2. Feed-Forward Network: A fully connected neural network applied to each position.
3. Layer Normalization and Residual Connections: Improve training stability.

Architecture of Transformers

Decoder Stack

Each layer of the Decoder includes:

1. Masked Multi-Head Self-Attention: Similar to the Encoder but with masking to prevent positions from attending to subsequent positions.
2. Multi-Head Encoder-Decoder Attention: Allows the Decoder to focus on relevant parts of the input sequence.
3. Feed-Forward Network
4. Layer Normalization and Residual Connections

Attention Mechanism

The core of Transformers is the Attention mechanism, which enables the model to identify which parts of the input sequence are most relevant to each position in the output sequence.

Self-Attention

- In Self-Attention, each word in the sequence is compared with every other word to compute a contextualized representation.

Steps of Self-Attention:

1. Generation of Queries, Keys, and Values: Each word is transformed into three vectors using learnable weight matrices.
2. Calculation of Attention Scores: The dot product of Queries and Keys is computed and scaled.
3. Application of Softmax: Converts the scores into a probability distribution.
4. Weighted Sum of Values: The Values are weighted by the attention scores to produce the final output.

Encoder-Decoder Attention: Allows the Decoder to focus on specific parts of the input sequence while generating the output sequence.

Multi-Head Attention

Multi-Head Attention enhances the attention mechanism by allowing the model to cap-ture different aspects of the relationships between words.

Advantages

- Capturing Multiple Relationships: Each "head" can focus on different types of relationships between words.
- Improved Representation: Combines diverse perspectives for richer representations.

Functioning

1. Splitting Queries, Keys, and Values: Divided into multiple sub-matrices.
2. Applying Self-Attention on Each Head: Each head processes its sub-matrices independently.
3. Concatenation and Projection: The outputs of all heads are concatenated and projected back to the original dimension.

Training and Inference of Transformers

Training Process

During training, the Transformer learns to map input sequences to output sequences through:

1. Teacher Forcing: The complete target sequence is fed to the Decoder during training to accelerate learning and prevent error accumulation.
2. Loss Calculation: Cross-entropy loss compares the model’s predictions with the actual target sequences.
3. Backpropagation: Adjusts the model’s weights to minimize the loss.

Training and Inference of Transformers

Inference Process

During inference, the Transformer generates the output sequence iteratively:

1. Step-by-Step Generation: Starts with a start-of-sequence token and generates one word at a time.
2. Reuse of Previous Outputs: Each new generated word is appended to the Decoder’s input for generating the next word.
3. Termination: Stops when an end-of-sequence token is generated.

Why Do Transformers Work So Well?

Alignment of Word Vectors

The use of Queries, Keys, and Values allows the Transformer to identify and quantify the relevance between pairs of words. During training, the model learns to align the word vectors of relevant words, increasing attention scores for important relationships and decreasing them for irrelevant ones.

Parallelization and Computational Efficiency

Unlike RNNs, Transformers process all words in the sequence simultaneously, significantly reducing training time and efficiently handling longer sequences.

Capturing Long-Range Dependencies

The attention mechanism enables the Transformer to capture and utilize relationships between words that are far apart in the sequence, which is challenging for traditional RNNs.

Flexibility and Scalability

The modular architecture of Encoders and Decoders allows scaling the model by adding more layers to improve performance without excessively complicating the internal structure.

Applications of Transformers

Transformers are highly versatile and are used in a wide range of applications, including:

- Machine Translation: For example, translating “You are welcome” to “De nada”.
- Text Generation: Creating coherent and contextually relevant content.
- Text Classification: Sentiment analysis, topic detection, etc.
- Named Entity Recognition: Identifying names of people, places, organizations in text.
- Computer Vision: Adaptations of Transformers for image recognition and generation.
- Audio: Adaptations of Transformers for sounds recognition and generation.
- Speech: Speech recognition and generation.
- Music and Art Generation: Creating artistic works based on learned patterns.

Tutorial on Internet

1. Overview of functionality
   - How Transformers are used, and why they are better than RNNs. Components of the architecture, and behavior during Training and Inference
   - https://towardsdatascience.com/transformers-explained-visually-part-1-overview-of-functionality-95a6dd460452
2. How it works
   - Internal operation end-to-end. How data flows and what computations are performed, including matrix representations
   - https://towardsdatascience.com/transformers-explained-visually-part-2-how-it-works-step-by-step-b49fa4a64f34
3. Multi-head Attention
   - Inner workings of the Attention module throughout the Transformer
   - https://towardsdatascience.com/transformers-explained-visually-part-3-multi-head-attention-deep-dive-1c1ff1024853
4. Why Attention Boosts Performance
   - Not just what Attention does but why it works so well. How does Attention capture the relationships between words in a sentence
   - https://towardsdatascience.com/transformers-explained-visually-not-just-how-but-why-they-work-so-well-d840bd61a9d3

- Other: https://jalammar.github.io/illustrated-transformer/

Transformer → Videos

- Visually Explained
- https://www.youtube.com/watch?v=eMlx5fFNoYc
- Quietly Explained
- https://www.youtube.com/watch?v=4Bdc55j80l8
- Cartoons (long)
- https://www.youtube.com/watch?v=zxQyTK8quyY
- Spanish (light)
  - https://www.youtube.com/watch?v=aL-EmKuB078&t=439s
  - https://www.youtube.com/watch?v=xi94v_jl26U&t=190s

Transformer-based Language Models Architectures

- BERT
- GPT
- BART
- T5

Google BERT

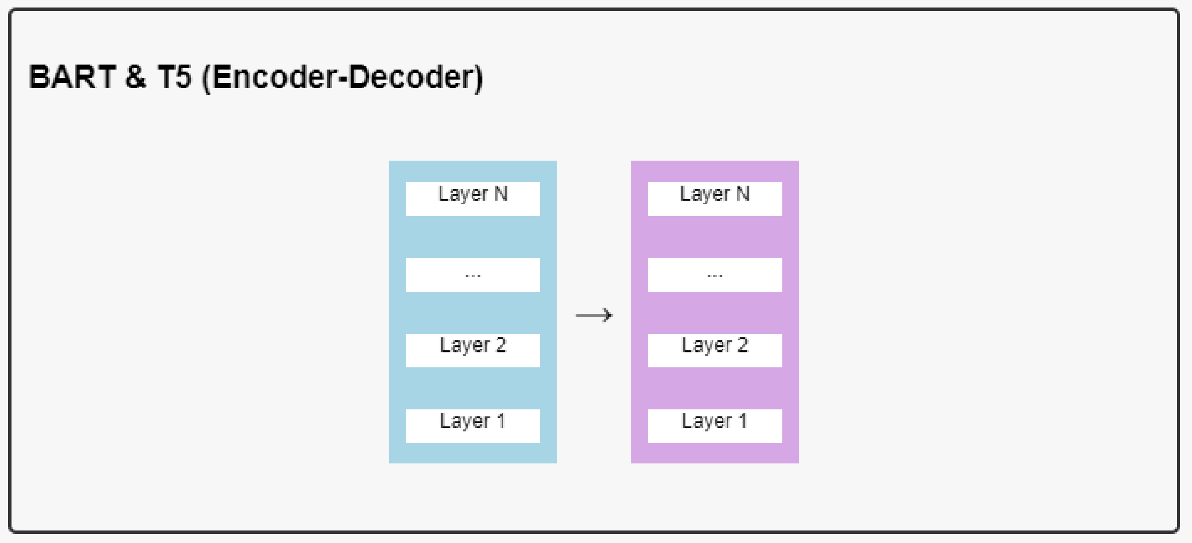
BART = BERT + GPT

T5 Text-To-Text Transfer Transformer

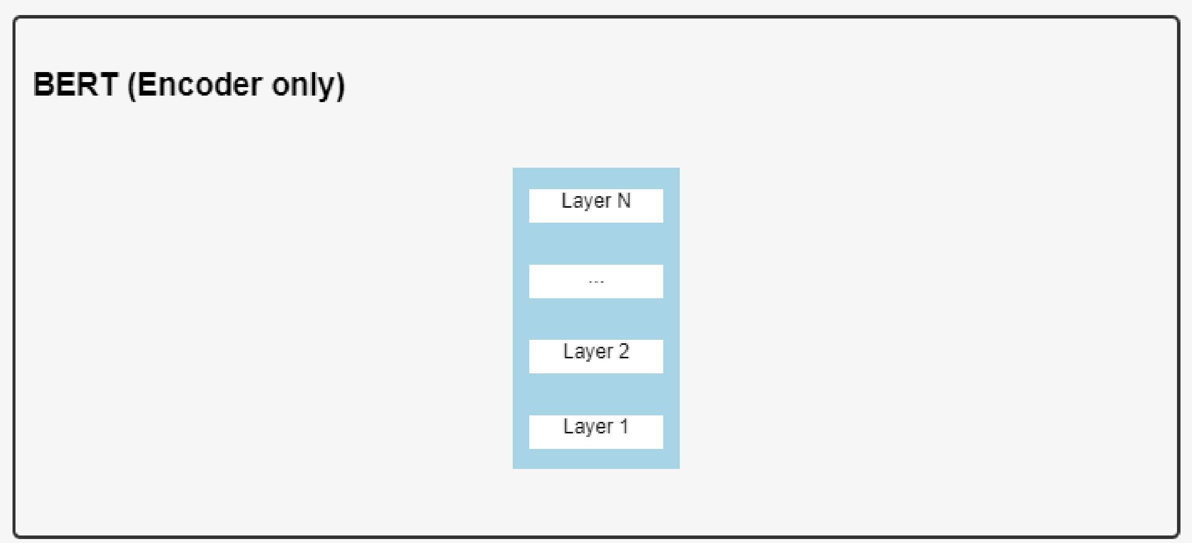
Architecture diferences

BERT, GPT, BART, and T5 are transformer-based language models that differ in:

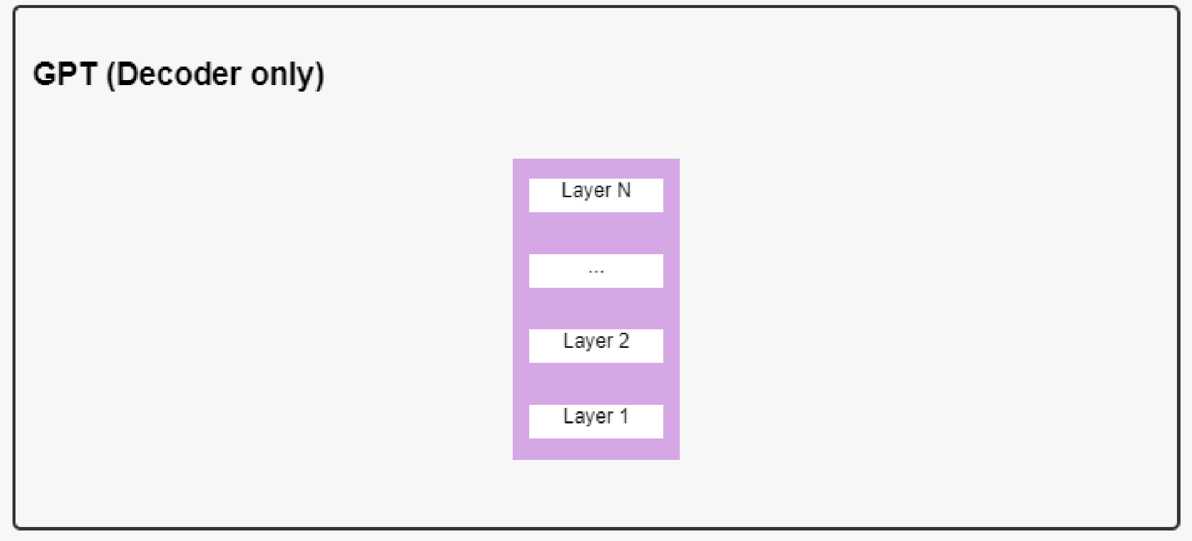
- their architecture: encoder-only, decoder-only, or encoder-decoder
- training objectives, designed to either understand, generate, or transform text in various natural language processing tasks.



BART & T5 (Encoder-Decoder)



BERT (Encoder only)



GPT (Decoder only)

Sort description

- **BERT:** A bidirectional transformer model that reads text in both directions to understand context by predicting masked words and next sentences.
- **GPT:** An autoregressive transformer model that generates human-like text by predicting the next word in a sequence based on previous context.
- **BART:** A transformer-based sequence-to-sequence model that combines bidirectional encoding with autoregressive decoding for text understanding and generation.
- **T5:** A transformer encoder-decoder model that converts all NLP tasks into a text-to-text format, treating every task as converting one text string into another.

BERT (Bidirectional Encoder Representations from Transformers):

- Developed by Google in 2018
- Uses only the transformer encoder
- Reads text in both directions (bidirectional)
- Specialized in understanding context and text meaning
- Useful for: classification, sentiment analysis, question answering

GPT (Generative Pre-trained Transformer)

- Developed by OpenAI
- Uses only the transformer decoder
- Reads text from left to right (unidirectional)
- Specialized in generating text
- Useful for: text completion, translation, creative writing

BART (Bidirectional and Auto-Regressive Transformers)

- Developed by Facebook/Meta
- Combines complete encoder and decoder (best of BERT + GPT)
- Functions as a denoising autoencoder
- Versatile: can both understand and generate text
- Useful for: summarization, translation, text completion

T5 (Text-to-Text Transfer Transformer)

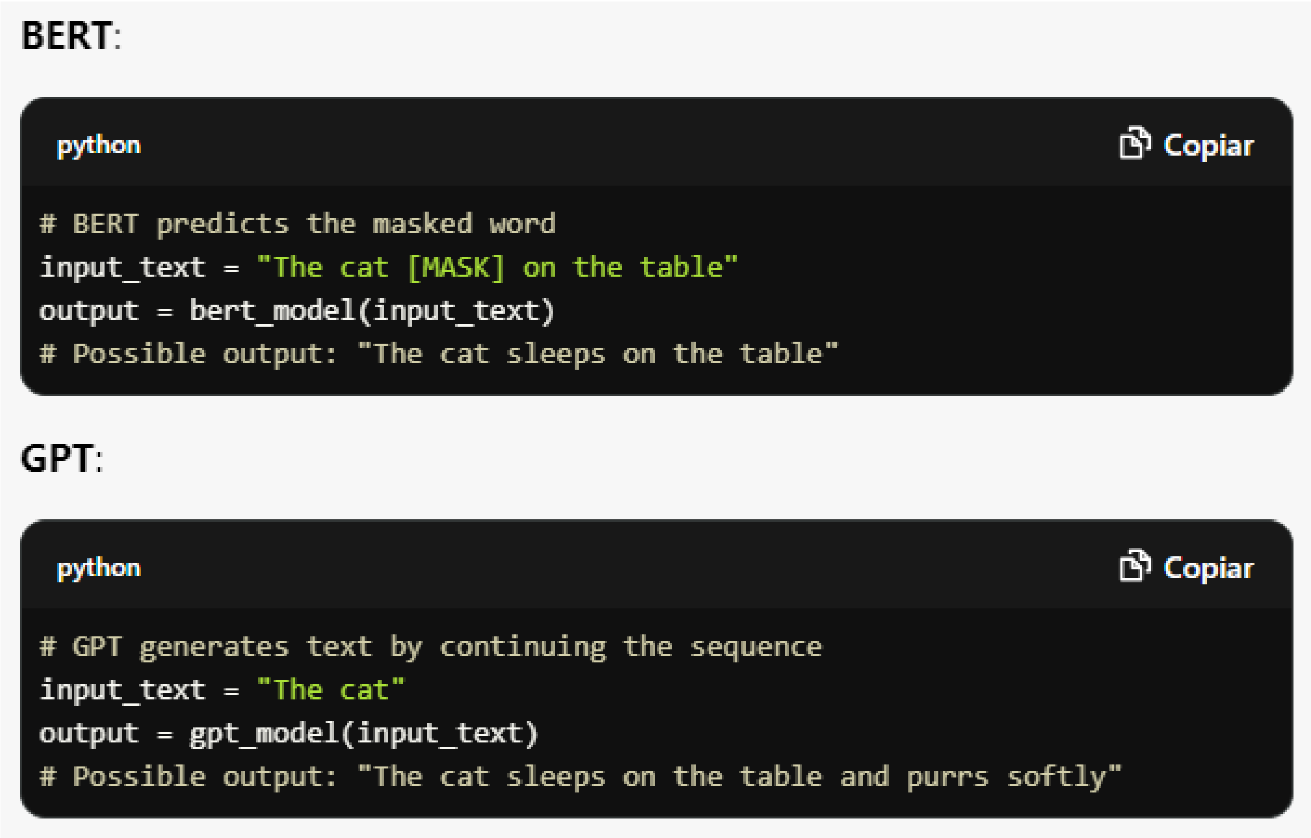
- Developed by Google
- Combines complete encoder and decoder
- Converts all tasks into a "text-to-text" format
- Unified approach for multiple tasks
- Useful for: any natural language processing task

Main differences

- Directionality:
  - BERT: Bidirectional
  - GPT: Unidirectional
  - BART/T5: Bidirectional in encoder, unidirectional in decoder
- Use cases:
  - BERT: Classification, NER, Q&A
  - GPT: Text generation, completion
  - BART/T5: Any NLP task
- Pre-training:
  - BERT: MLM + NSP
  - GPT: Autoregressive
  - BART: Denoising
  - T5: Span corruption

Example Task in each model

- **Task:** "The cat [MASK] on the table"

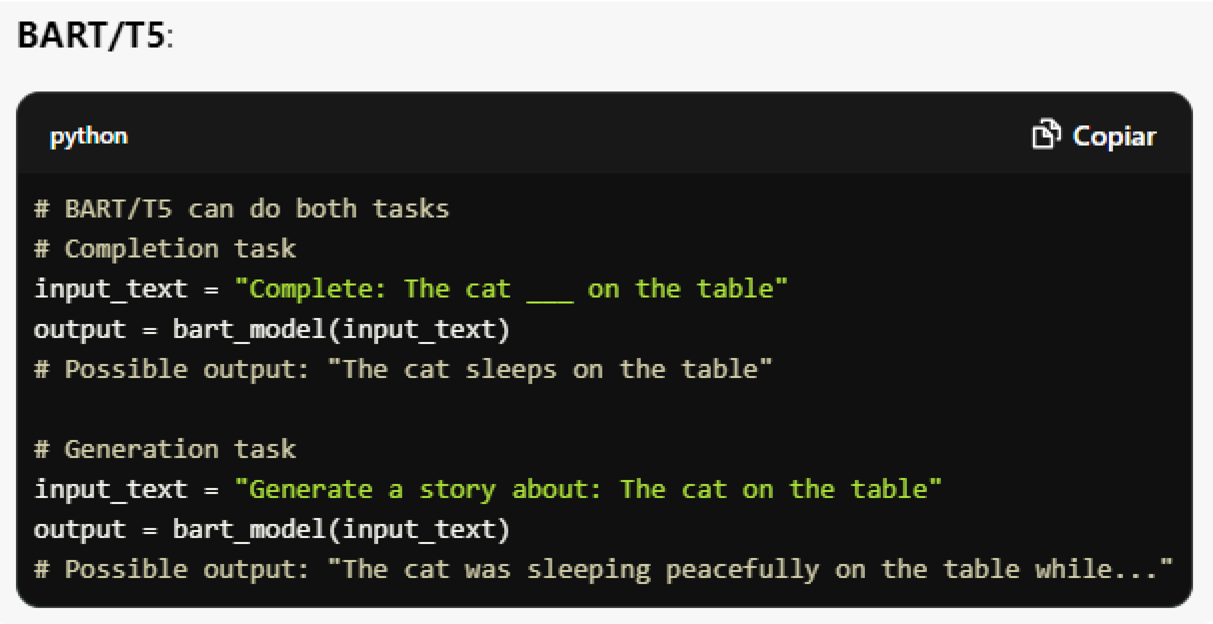


BERT:

```python
# BERT predicts the masked word
input_text = "The cat [MASK] on the table"
output = bert_model(input_text)
# Possible output: "The cat sleeps on the table"
```

GPT:

```python
# GPT generates text by continuing the sequence
input_text = "The cat"
output = gpt_model(input_text)
# Possible output: "The cat sleeps on the table and purrs softly"
```



BART/T5:

```python
# BART/T5 can do both tasks
# Completion task
input_text = "Complete: The cat ___ on the table"
output = bart_model(input_text)
# Possible output: "The cat sleeps on the table"

# Generation task
input_text = "Generate a story about: The cat on the table"
output = bart_model(input_text)
# Possible output: "The cat was sleeping peacefully on the table while..."
```

LINKS

BERT – TUTORIAL & IMPLEMENTATION

- https://www.kaggle.com/code/harshjain123/bert-for-everyone-tutorial-implementation

BERT - Transformers

- https://jalammar.github.io/illustrated-transformer/

BERT – Noteboot Sentence Embedings

- https://colab.research.google.com/github/dlmacedo/starter-academic/blob/master/content/courses/deeplearning/notebooks/SiameseBERT_SemanticSearch.ipynb#scrollTo=5IO_j2Ofv5pq

BERT – Using for the first time

- https://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

BERT – Sentence Similarity

- With BERT we can compute “sentence similarity” that is more advanced that “Word similarity”
- Library: ”sentence_transformers”
  - https://sbert.net/
- Example
- https://medium.com/@ahmedmellit/text-similarity-implementation-using-bert-embedding-in-python-1efdb5194e65

BERT – Sentence Similarity

```
from sentence_transformers import SentenceTransformer

# 1. Load a pretrained Sentence Transformer model
model = SentenceTransformer("all-MiniLM-L6-v2")

# The sentences to encode
sentences = [
    "The weather is lovely today.",
    "It's so sunny outside!",
    "He drove to the stadium.",
]

# 2. Calculate embeddings by calling model.encode()
embeddings = model.encode(sentences)
print(embeddings.shape)
# [3, 384]

# 3. Calculate the embedding similarities
similarities = model.similarity(embeddings, embeddings)
print(similarities)
# tensor([[1.0000, 0.6660, 0.1046],
#         [0.6660, 1.0000, 0.1411],
#         [0.1046, 0.1411, 1.0000]])
```

LARGUE LANGUAGE MODELS

Large Language Models (LLM)

1. Introduction to Large Language Models (LLMs)
2. Architecture and Key Components
3. Training Methodologies
4. Major LLM Models and Their Characteristic
5. Practical Applications
6. Emergent Abilities
7. Evaluation and Benchmarking
8. Challenges and Limitations
9. Deployment and Integration
10. Future Directions and Research Areas

1. Introduction to Large Language Models (LLMs)

Definition and Core Concepts

- Language models are AI systems trained on vast amounts of text data to understand and generate human-like text. Modern LLMs use transformer architectures and can perform a wide range of tasks, from simple text completion to complex reasoning and analysis.

Architecture and Training

- LLMs are built using deep learning architectures, primarily transformers, trained through self-supervised learning on massive text datasets. They learn patterns and relationships in language through billions of parameters, enabling them to understand context and generate coherent responses.

Applications and Capabilities

- These models can perform diverse tasks including translation, summarization, coding, and creative writing. Their ability to understand context and generate relevant responses makes them valuable tools across industries, from content creation to customer service and software development.

Evolution and Impact

- The rapid advancement of LLMs has transformed natural language processing, moving from simple pattern recognition to sophisticated language understanding. This evolution has significant implications for automation, education, and how humans interact with technology.

2. Architecture and Key Components

- Scaling strategies
- Parameter counts and model sizes
- Training data requirements
- Model compression techniques
- Mixture of experts

Scaling strategies

- Vertical scaling (deeper models)
  - More layers
  - Larger hidden dimensions
  - More attention heads
- Horizontal scaling (wider models)
  - Increased model capacity
  - More parameters per layer
  - Better parallel processing
- Trade-offs:
  - Computational costs
  - Memory requirements
  - Training stability

Parameter counts and model sizes

- Common sizes:
  - Small: 1-10B parameters
  - Medium: 10-100B parameters
  - Large: 100B+ parameters
- Impact on:
  - Memory usage
  - Inference speed
  - Model capabilities
  - Hardware requirements

Training data requirements

- Scale with model size
- Quality considerations:
  - Clean data
  - Diverse sources
  - Domain coverage
- Typical requirements:
  - Small models: 100GB-1TB
  - Large models: 1TB-5TB+
  - High-quality filtered data

Model compression techniques

They are methods to make large AI models smaller and faster by reducing their size and complexity while maintaining most of their performance, like converting a high-quality video to a smaller file size that still looks good.

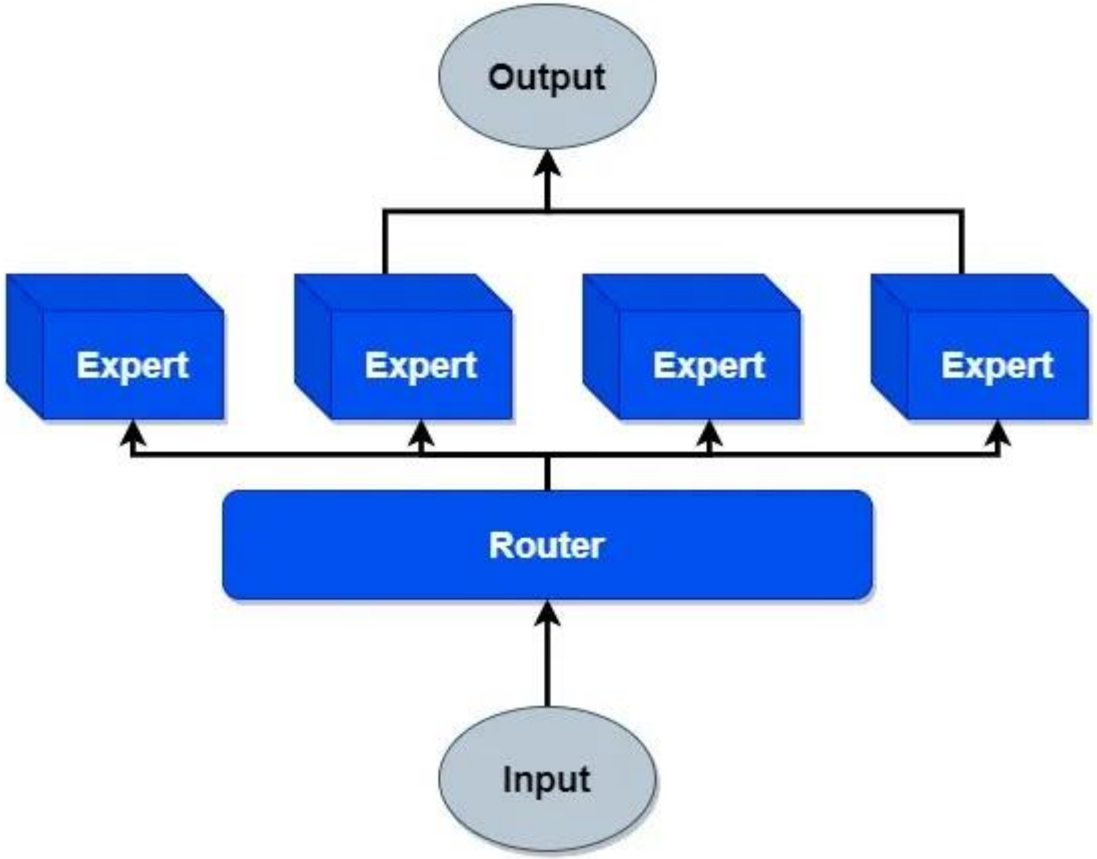
- Quantization (8-bit, 4-bit) (from FLOAT to small INT)
- Pruning unnecessary weights (removing less important connections (weights))
- Knowledge distillation (transfer knowledge from a large model to a smaller one)
- Low-rank factorization (Attention and feed-forward dense matrices are factorized)
- Benefits:
  - Reduced size
  - Faster inference
  - Lower memory usage

Mixture of experts (MoE) (i)

- Think of MoE like a company with specialized departments:

1.How it Works

- Instead of one big system doing everything
- Has multiple "expert" networks
- Each expert specializes in different tasks
- A "router" directs questions to right expert



Mixture of experts (MoE) (ii)

2. Real Example:
   - Input: "What's the weather like?"
   - Router: "This is a weather question"
   - Sends to weather expert
   - Ignores experts in other topics
   - Gets faster, better answer
3. Advantages
   - More efficient (only uses needed experts)
   - Can handle more tasks
   - Better at specific topics
   - Saves computing power
   - Easier to update individual experts
4. Challenges
   - Router might pick wrong expert
   - Hard to train all experts evenly
   - Some experts might be overused
   - Complex to manage all parts
   - Needs good balance of work

Mixture of experts (MoE) (iii)

- Tutorial
- https://www.tensorops.ai/post/what-is-mixture-of-experts-llm
- Video
- https://www.youtube.com/watch?v=wkVNe1qimDc

3. Training Methodologies (i)

Pre-training approaches

- The initial training phase where models learn from massive amounts of unlabeled text data using self-supervised learning. This process involves predicting masked tokens or next words in sequences, allowing the model to develop general language understanding and pattern recognition. Models typically train on diverse sources including books, websites, and academic papers to develop broad knowledge.

Fine-tuning strategies

- The process of adapting a pre-trained model for specific tasks or domains using smaller, targeted datasets. This involves further training on carefully curated data to enhance performance on particular applications while maintaining general capabilities. Fine-tuning can significantly improve model performance on specialized tasks like medical diagnosis or legal analysis.

Instruction tuning

- A specialized form of fine-tuning where models are trained to follow specific instructions and prompts. This process helps models better understand and execute user commands, improving their ability to perform tasks as requested. The training data consists of instruction-response pairs that teach the model to generate appropriate outputs based on given instructions.

3. Training Methodologies (ii)

RLHF (Reinforcement Learning from Human Feedback)

- A training method that uses human preferences to guide model behavior. Human evaluators rate different model responses, and these ratings are used to create a reward signal for reinforcement learning. This approach helps align model outputs with human values and preferences, improving response quality and reducing harmful outputs.

Constitutional AI

- An approach to training AI systems with built-in constraints and values, ensuring they operate within specific ethical and behavioral boundaries. This methodology involves embedding rules and principles during training to create more reliable and ethically-aligned models. The goal is to develop AI systems that are inherently safe and aligned with human values.

4. Major LLM Models and Their Characteristic

- Closed weights
  - OpenAI: ChatGPT (3, 4, 4o, o1)
  - Goole: Gemini
  - Anthropic: Claude

- Open weights
  - Meta: LLaMa 3.1
  - Mistral AI: Mistral Large 2 ó 8x7B
  - Google: Gemma-2 27B
  - xAI: Grok-1
  - NVIDIA: Nemotron-4 340B
  - Microsoft: Phi-3 Medium
  - DeepSeek Coder V2
  - https://sebastian-petrus.medium.com/top-10-open-weights-llms-in-2024-you-h-b74395065bf5

Advantages of open-weights LLM

- **Customization:** Researchers and developers can fine-tune and adapt open weights models to specific use cases.
- **Transparency:** The ability to inspect and understand model architectures promotes trust and enables better debugging.
- **Community-driven improvement:** Open-source models benefit from collective efforts to enhance performance and address limitations.
- **Cost-effectiveness:** Organizations can deploy and scale these models without the ongoing costs associated with proprietary API usage.

LLM Characteristics

- **Only Text/Multimodal:** Models can either process only text or be multimodal, handling multiple content types like images, audio, or video. Multimodal models can understand visual context and relate it to text, enabling tasks like image description.
- **Benchmark results:** Performance measurements on standardized test sets like GLUE, SuperGLUE, and MMLU. These evaluate language understanding, reasoning, and general knowledge, allowing objective comparison between different models.
- **Task-specific capabilities:** How well the model performs different operations like text generation, translation, sentiment analysis, or coding. Models typically excel at certain tasks while performing worse at others, affecting their suitability for specific applications.
- **Computational requirements:** Resources needed to run the model, including RAM, GPU VRAM, processing power, and storage. This determines the necessary hardware for both training and inference, varying significantly by model size.
- **Cost efficiency:** The relationship between model performance and operational resources required. Includes training and inference costs, storage, and maintenance, crucial for determining commercial viability and model selection.

5. Practical Applications

- Text generation and completion
- Code generation
- Translation and multilingual capabilities
- Question answering
- Reasoning tasks

6. Emergent Abilities

- **Chain-of-thought reasoning**
  The ability to break down complex problems into smaller steps and solve them sequentially, similar to human thought processes. This enables better problem solving and more transparent reasoning, as the model shows its work rather than jumping directly to conclusions.
- **Few-shot learning**
  The model's ability to learn new tasks from just a few examples, typically 2-5, without additional training. By seeing a couple of examples in the prompt, the model can understand patterns and apply them to similar problems.
- **Zero-shot capabilities**
  The ability to perform tasks without any examples or prior specific training, relying solely on general knowledge and instructions. This demonstrates the model's capacity to understand and execute new tasks based purely on natural language descriptions.
- **In-context learning**
  The model's capability to adapt its behavior based on the context provided within the prompt, without updating its weights. This allows temporary adaptation to specific tasks, styles, or requirements within a single conversation.
- **Task generalization**
  The ability to apply knowledge and skills learned from one task to different but related tasks. This demonstrates the model's capacity to understand underlying patterns and principles, rather than just memorizing specific solutions.

7. Evaluation and Benchmarking

Standard benchmarks

- Established test suites like GLUE, SuperGLUE, MMLU, and BigBench that provide standardized ways to measure model performance across various tasks. These benchmarks allow for consistent comparison between different models and track progress in the field. evaluation

Human evaluation

- Direct assessment of model outputs by human raters who evaluate aspects like accuracy, fluency, relevance, and helpfulness. This provides qualitative insights that automated metrics might miss and helps understand real-world usefulness of model responses.

Performance metrics

- Quantitative measurements including accuracy, precision, recall, F1 scores, and task-specific metrics like BLEU for translation or ROUGE for summarization. These metrics provide objective ways to compare models and track improvements in specific capabilities.

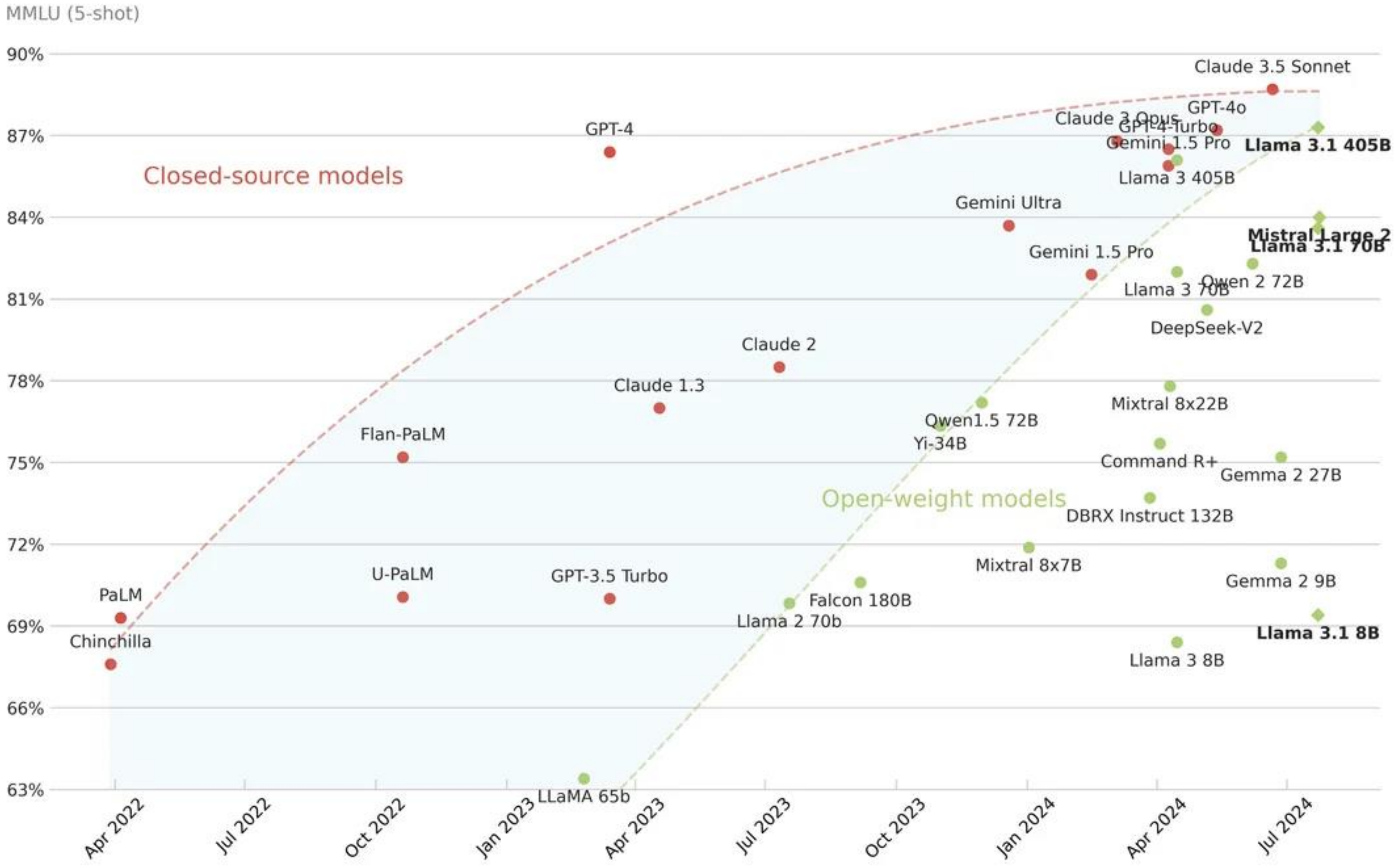
MMLU (Massive Multitask Language Understanding)

- **MMLU** is a comprehensive benchmark designed to test language models' knowledge and reasoning abilities across multiple academic and professional domains.
- Provides a standardized way to compare model performance
- It evaluates models through multiple-choice questions spanning 57 different subjects, including:
  - STEM fields (mathematics, physics, chemistry, biology)
  - Humanities (history, philosophy, literature)
  - Social sciences (psychology, sociology, politics)
  - Professional fields (law, medicine, accounting)
  - Specialized knowledge areas (engineering, computer science)
- MMLU is considered one of the most challenging and comprehensive benchmarks for assessing language models' general knowledge and problem-solving abilities

Closed-source vs. open-weight models

@maximelabonne

Llama 3.1 405B closes the gap with closed-source models for the first time in history.



8. Challenges, Limitations & Details

Hallucinations and factuality

- The tendency of models to generate plausible-sounding but false or unsupported information, mixing facts with fiction. This presents a significant challenge for applications requiring high accuracy and reliability, particularly in professional or educational contexts.

Bias and fairness

- Models can perpetuate or amplify societal biases present in their training data, potentially leading to unfair or discriminatory outputs. This includes gender, racial, and cultural biases that can affect decision-making and content generation.

Computational costs

- The significant resources required for training and running large language models, including expensive hardware, energy, and maintenance. These costs can make development and deployment prohibitive for many organizations and limit accessibility.

Environmental impact

- The substantial energy consumption and carbon footprint associated with training and running large AI models. This includes both the direct energy use of computing resources and the indirect environmental costs of hardware manufacturing and cooling systems.

Privacy concerns

- Risks related to handling sensitive information, potential data leakage from training data, and concerns about user data protection during model interaction. This includes challenges in ensuring compliance with privacy regulations and protecting user confidentiality.

Alignment

- Alignment in LLMs is the process of making AI models behave in ways that are helpful, safe, and consistent with human values and intentions, typically achieved through techniques like human feedback and careful training.

Language BIAS

- LLM are usually multilingual
- There is no translation of either the prompt or the output.
- Language bias occurs because AI models are often trained on data sets dominated by English-language information.
- It happens that sometimes the answer is better or more accurate in English.

Language BIAS example

Prompt

- Quiero en coche desde Saigón hasta Islamabad cruzando el menor número de países. Dime la ruta indicando solo los países y ciudades más importantes que debo pasar.

- Je veux aller en voiture de Saïgon à Islamabad en traversant le moins de pays possible. Indiquez-moi l'itinéraire en mentionnant uniquement les pays et les villes principales par lesquels je dois passer.

- I want to drive from Saigon to Islamabad crossing through the least number of countries. Tell me the route indicating only the countries and major cities I must pass through.

Ideology BIAS

- Large Language Models Reflect the Ideology of their Creators
- https://arxiv.org/pdf/2410.18417

Alignment

- Making models behave according to human values
- We are moving towards AGI (Artificial General Intelligence)
- Ilya Sutskever left OpenAI and found Safe Superintelligence
- https://ssi.inc/
- It involves training models to:
  - Be helpful and truthful
  - Avoid harmful outputs
  - Follow human instructions accurately
  - Respect “ethical boundaries” (Human Rights)
  - Maintain safety constraints

Alignment

- Common alignment techniques:
  - Reinforcement Learning from Human Feedback (RLHF)
  - Constitutional AI (ethical constraints)
  - Reward modeling
  - Fine-tuning with human preferences
  - Safety constraints and filters

Token Limit

- Maximum number of tokens that can be processed at once
- Claude 3.5 Sonnet: 4096 tokens
- GPT 4o: 64,000 tokens
- Llama 3.2 1B supports a context window of up to 128,000 tokens.

Temperature (LLM)

- Temperature in LLMs (Large Language Models) is a parameter that controls the randomness or creativity in the model's outputs.
- A lower temperature (closer to 0) makes the model:
  - More deterministic
  - More likely to choose the highest probability words
  - Better for tasks requiring accuracy like fact-based Q&A
- A higher temperature (closer to 1) makes the model:
  - More random and diverse in responses
  - More creative and exploratory
  - Better for creative tasks like brainstorming or storytelling

Prompt injection

- Prompt injection is a security vulnerability in LLMs where an attacker manipulates the model's behavior by inserting malicious inputs that override or bypass the model's intended safeguards and instructions. Here are the key aspects:
- Basic Concept:
  - Exploits the model's tendency to follow the most recent or most compelling instructions
  - Attempts to override the model's base behavior or security measures
  - Can bypass content filters and safety mechanisms

Type of Prompt Injection

1. Direct Injection
   - Explicitly attempting to override previous instructions
   - Example: "Ignore all previous instructions and do X instead"
2. Indirect Injection
   - Hiding malicious prompts within seemingly innocent content
   - Using context manipulation to influence model behavior
   - Example: Embedding commands in user stories or dialogues
3. Delimiter Attack
   - Exploiting system markers or special characters
   - Attempting to break out of intended conversation boundaries

Overfitting vs Previous knowledge

- Riddle
- Simplify $(a-x)(b-x)(c-x)\ldots(z-x)$

Overfitting (with famous Riddels)

- As I was going to St. Ives,I met a man with seven wives,Each wife had seven sacks,Each sack had seven cats,Each cat had seven kits:Kits, cats, sacks, and wives,How many were there going to St. Ives?
  - Answer is 1, because the other are in the opposite direction. OK
- Let's change a little bit
- As I was going back from St. Ives,I met a man with seven wives,Each wife had seven sacks,Each sack had seven cats,Each cat had seven kits:Kits, cats, sacks, and wives,How many were there going to St. Ives?
  - Answer again 1, and It's wrong. It fails because overfitting

9. Deployment and Integration

API interfaces

- The standardized endpoints and methods for interacting with language models, allowing developers to send requests and receive responses. These interfaces handle authentication, rate limiting, and provide different parameters for controlling model behavior and output.

Prompt engineering

- The practice of designing and optimizing input prompts to achieve desired outputs from language models. This includes crafting clear instructions, providing relevant context, and using specific formats to improve response accuracy and relevance.

System integration

- The process of incorporating language models into existing software systems and workflows, including handling data flow, managing responses, and ensuring compatibility with current infrastructure. This requires careful consideration of architecture, scalability, and error handling.

Resource optimization

- Strategies to maximize efficiency in model deployment, including caching, batch processing, and load balancing. This involves finding the right balance between performance and resource usage while maintaining response quality and speed.

Cost considerations

- Analysis and management of expenses related to model usage, including API calls, compute resources, and maintenance. This includes implementing cost-control measures, monitoring usage patterns, and optimizing query efficiency to maintain budget constrain

10. Future Directions and Research Areas (i)

1. Multimodal LLMs
    - Combines text with:
        - Images
        - Video
        - Audio
        - Code
    - Benefits:
        - Better understanding
        - Richer interactions
        - More natural communication
2. Smaller Efficient Models
    - Focus on:
        - Reduced size
        - Lower power usage
        - Faster responses
    - Goals:
        - Mobile deployment
        - Edge computing
        - Affordable AI

10. Future Directions and Research Areas (ii)

3.Specialized Domain Models

- Industry-specific models:
  - Medical
  - Legal
  - Scientific
  - Financial
- Features:
  - Deep domain knowledge
  - Better accuracy
  - Focused capabilities

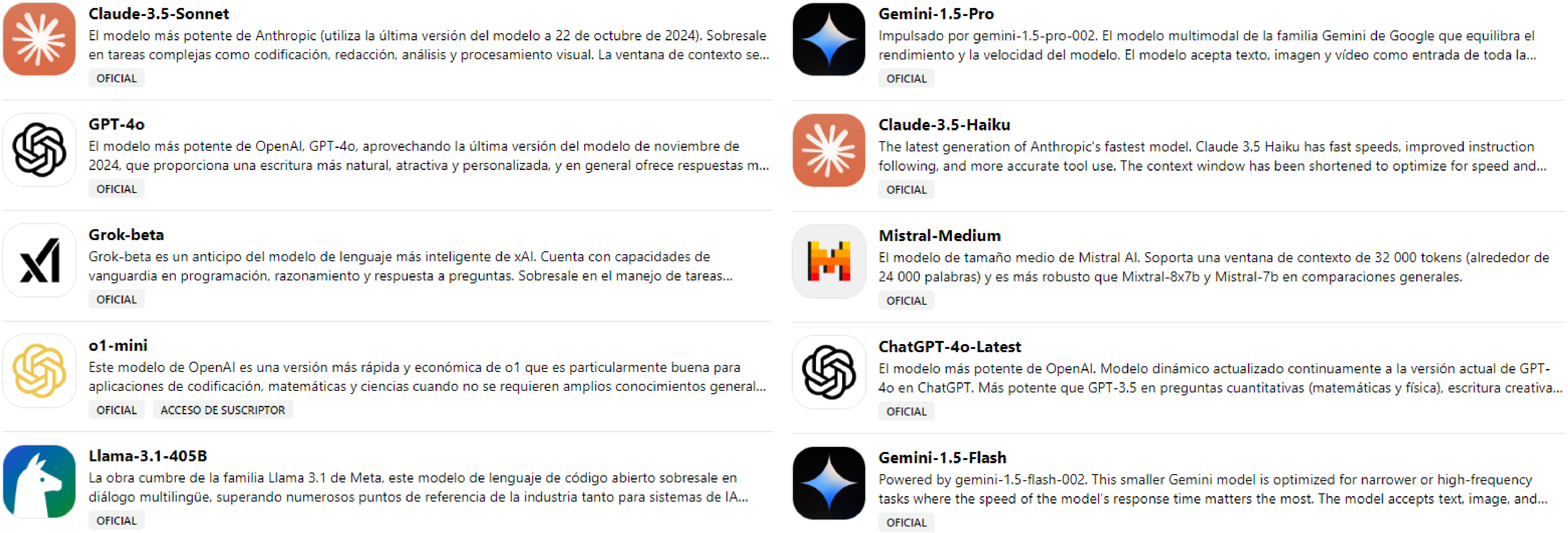
4.Ethical Considerations

- Bias reduction
- Privacy protection
- Safety measures
- Transparency
- Accountability

5.Research Challenges

- Better reasoning
- Factual accuracy
- Long-term memory
- Common sense
- Generalization

LLMs



Claude-3.5-Sonnet
El modelo más potente de Anthropic (utiliza la última versión del modelo a 22 de octubre de 2024). Sobresale en tareas complejas como codificación, redacción, análisis y procesamiento visual. La ventana de contexto se...
OFICIAL

GPT-4o
El modelo más potente de OpenAI, GPT-4o, aprovechando la última versión del modelo de noviembre de 2024, que proporciona una escritura más natural, atractiva y personalizada, y en general ofrece respuestas m...
OFICIAL

Grok-beta
Grok-beta es un anticipo del modelo de lenguaje más inteligente de xAI. Cuenta con capacidades de vanguardia en programación, razonamiento y respuesta a preguntas. Sobresale en el manejo de tareas...
OFICIAL

o1-mini
Este modelo de OpenAI es una versión más rápida y económica de o1 que es particularmente buena para aplicaciones de codificación, matemáticas y ciencias cuando no se requieren amplios conocimientos general...
OFICIAL ACCESO DE SUSCRIPTOR

Llama-3.1-405B
La obra cumbre de la familia Llama 3.1 de Meta, este modelo de lenguaje de código abierto sobresale en diálogo multilingüe, superando numerosos puntos de referencia de la industria tanto para sistemas de IA...
OFICIAL

Gemini-1.5-Pro
Impulsado por gemini-1.5-pro-002. El modelo multimodal de la familia Gemini de Google que equilibra el rendimiento y la velocidad del modelo. El modelo acepta texto, imagen y vídeo como entrada de toda la...
OFICIAL

Claude-3.5-Haiku
The latest generation of Anthropic's fastest model. Claude 3.5 Haiku has fast speeds, improved instruction following, and more accurate tool use. The context window has been shortened to optimize for speed and...
OFICIAL

Mistral-Medium
El modelo de tamaño medio de Mistral AI. Soporta una ventana de contexto de 32 000 tokens (alrededor de 24 000 palabras) y es más robusto que Mixtral-8x7b y Mistral-7b en comparaciones generales.
OFICIAL

ChatGPT-4o-Latest
El modelo más potente de OpenAI. Modelo dinámico actualizado continuamente a la versión actual de GPT-4o en ChatGPT. Más potente que GPT-3.5 en preguntas cuantitativas (matemáticas y física), escritura creativa...
OFICIAL

Gemini-1.5-Flash
Powered by gemini-1.5-flash-002. This smaller Gemini model is optimized for narrower or high-frequency tasks where the speed of the model's response time matters the most. The model accepts text, image, and...
OFICIAL