Large Language Models

D. Psarras, Prof. Ioannis Pitas
Aristotle University of Thessaloniki
pitas@csd.auth.gr
www.aiia.csd.auth.gr
Version 2.0

Artificial Intelligence &
Information Analysis Lab

Large Language Models

- **Introduction**
- LLM Building Blocks
- Encoder-only LLMs
- Decoder-only LLMs
- Encoder – Decoder LLMs
- LLM tasks

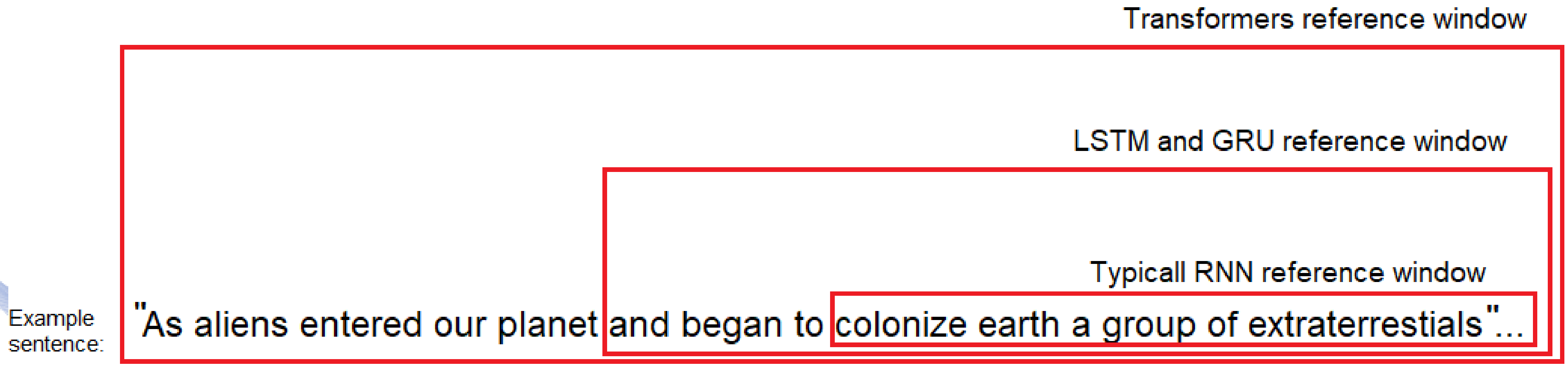
Introduction

Large Language Models (**LLMs**) are machine learning models equipped to handle Natural Language Processing (**NLP**) use cases.

In contrast to traditional NLP algorithms state of the art LLMs have an infinite reference window because of the Transformers based architecture they use.

Introduction

- In theory, transformers have an ***infinite reference window***.



Hypothetical reference window of RNNs, LSTMs and Transformers.

Introduction

The main NLP tasks handled by LLMs are [YAN2023] :

- **Natural Language Understanding (NLU)**. Uses generalization of the LLMs on out of distribution data or cases with few training data. Taks of NLU include:
  - **text categorization**
  - **content analysis**
  - **sentiment analysis**
- **Knowledge-intensive tasks**. Tasks requiring domain specific expertise or general world knowledge.

Introduction

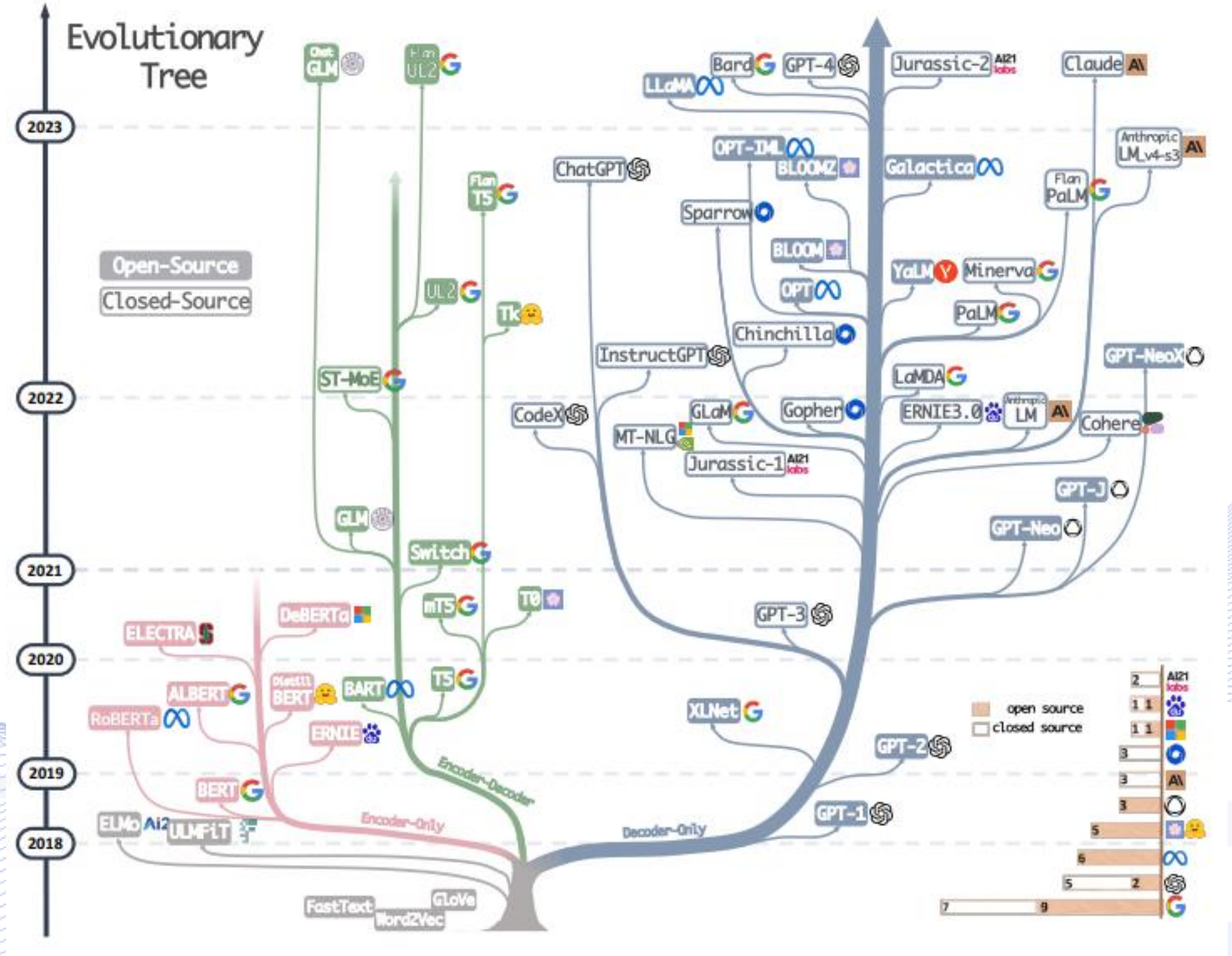
The main NLP tasks handled by LLMs are [YAN2023] :

- **Natural Language Generation (NLG)**. Creation of coherent, contextually relevant and high-quality text. Includes **question answering**, **text summarization**, **machine translation**, and **chatbots**.
- **Reasoning ability**. Perform decision making and problem solving in different contexts.

Introduction

State of the art LLMs are categorized into three types [YAN2023]:

- Encoder-only LLMs
- Decoder-only LLMs
- Encoder-Decoder LLMs



The evolutionary tree of modern LLMs traces the development of language models in recent years [YAN2023].

Large Language Models

- Introduction
- **LLM Building Blocks**
- Encoder-only LLMs
- Decoder-only LLMs
- Encoder – Decoder LLMs
- LLM tasks

LLM Building Blocks

LLM Building Blocks [AJI2023]:

- ***Tokenization***: Text compression in order to minimize the size of the encoded token while retaining the ability to represent text sequences. ***Byte Pair Encoding*** (***BPE***) and ***WordPiece*** are the main algorithms used.
- ***Embedding***: Representation of tokens to vectors capturing the semantic meaning in high-dimensional space. The embeddings are processed by the NN and are learned during the training.

LLM Building Blocks

- ***Attention***: Self attention mechanism used in Transformers assigns different weights to input tokens capturing long-range dependencies by focusing on the relevant information.
- ***Pre-Training***: LLM ***unsupervised*** or ***self-supervised training*** on large datasets. Allows fine-tuning on smaller task-specific labeled dataset.
- ***Transfer Learning*** allows LLM fine-tuning on smaller task-specific dataset to achieve high-performance.

LLM Building Blocks

Tokenization [PHU2022]:

A tokenizer breaks the unstructured data of text and creates discrete elements of chucks of information. The raw text is converted to a sequence of integers according to a vocabulary through an iterative process, such as the Byte Pair Encoding (BPE).

- First step in language modeling
- Applied on the corpus $\mathcal{C}$ to obtain tokens.
- The tokens are used to create the vocabulary $\mathcal{V}$, which is the set of unique tokens in the corpus.

LLM Building Blocks

Byte-Pair Encoding (***BPE***) tokenization [PHU2022]:

- Step 0: Define the alphabet set $\mathcal{A}\,\{a_i\}$ of the N characters in the corpus and map them through an injective function to an initial vocabulary set $\mathcal{V}_0$ consisting of all 256 bytes. New corpus $\mathcal{C}_0 \coloneqq \left(\mathcal{E}(a_i)\right)_{i\in I}$ where I is the interval $[0, \ldots, N-1]$.
- Step 1: Add the most frequent bigram (a pair of consecutive written units such as letters, syllables, or words) (b, b') of the corpus $\mathcal{C}_0$ in the set $\mathcal{V}_0$ as $s = bb'$ and replace every (b, b') with s in $\mathcal{C}_0$. New corpus $\mathcal{C}_1$ and vocabulary set $\mathcal{V}_1 = \mathcal{V}_0 \cup \{bb'\}$.
- Step 2: Repeat step 1 until the size of the vocabulary set $\mathcal{V}$ is n_v.

LLM Building Blocks

Word embedding [PHU2022]:
The embedding learns to represent each vocabulary element as a vector in $\mathcal{R}^{d_m}$.

Input: $\mathbf{X} \in \mathcal{R}^{L \times n_v}$, token IDs
Output: $\mathbf{X}' \in \mathcal{R}^{L \times d_m}$, vector representations of the token.
Parameters: $\mathbf{W}_E \in \mathcal{R}^{d_m \times n_v}$, the token embedding matrix.

LLM Building Blocks

Word embedding:

- The vocabulary $\mathcal{V}$ is ***one-hot encoded*** resulting in a set of one-hot tokens $\sigma(\mathcal{V}) \subset \mathcal{R}^{n_v}$.
- The model takes as input parts of the training dataset of size L called context window.

Using the vocabulary, the one-hot encoding and the context window parameter a string S of real text of $L(S) = L$ forms to a matrix $\mathbf{X} \in \mathcal{R}^{L \times n_v}$.

LLM Building Blocks

Word embedding:

The matrix $\mathbf{X} \in \mathcal{R}^{L \times n_v}$ is embedded to a smaller vector space through a $(d_m \times n_v)$ projection matrix:

$$\mathbf{X} \rightarrow \mathbf{X}' = \mathbf{X}\mathbf{W}_{\mathbf{E}}^T, \qquad \mathbf{W}_E: \mathcal{R}^{n_v} \longrightarrow \mathcal{R}^{d_m}$$

In ***unembedding***, a $(n_v \times d_m)$ projection matrix projects the output of the model on $\mathcal{R}^{n_v}$:

$$\mathbf{X}' \rightarrow \mathbf{X} = \mathbf{X}'\mathbf{W}_{\mathbf{U}}^T, \qquad \mathbf{W}_U: \mathcal{R}^{d_m} \longrightarrow \mathcal{R}^{n_v}$$

Artificial Intelligence & Information Analysis Lab

LLM Building Blocks

Attention:

Computes a single masked self- or cross- attention head.

Input: Vector representation of primary sequence $\mathbf{X}' \in \mathbb{R}^{L\times d}$ and context sequence $\mathbf{Y}' \in \mathbb{R}^{L'\times d}$.

Output: $\mathbf{X}'' \in \mathbb{R}^{L\times d_{out}}$ updated representations of tokens in $\mathbf{X}'$ combining information from tokens in $\mathbf{Y}'$.

Parameters: Consisting of:

$$\mathbf{W}_Q \in \mathbb{R}^{d_m\times d_k}, \mathbf{b}_Q \in \mathbb{R}^{d_k}$$

$$\mathbf{W}_K \in \mathbb{R}^{d_m\times d_k}, \mathbf{b}_K \in \mathbb{R}^{d_k}$$

$$\mathbf{W}_V \in \mathbb{R}^{d_m\times d_{out}}, \mathbf{b}_V \in \mathbb{R}^{d_{out}}$$

Hyperparameters: Mask of dimensions $L \times L'$.

LLM Building Blocks

Attention:

Step 1: Compute Query $\mathbf{Q} \in \mathbb{R}^{L \times d_k}$, Key $\mathbf{K} \in \mathbb{R}^{L' \times d_k}$ and Value $\mathbf{V} \in \mathbb{R}^{L' \times d_{out}}$ matrices

$$\mathbf{Q} = \mathbf{X}'\mathbf{W}_Q + \mathbf{1}_{L \times 1}\mathbf{b}_Q$$
$$\mathbf{K} = \mathbf{Y}'\mathbf{W}_K + \mathbf{1}_{L' \times 1}\mathbf{b}_K$$
$$\mathbf{V} = \mathbf{Y}'\mathbf{W}_V + \mathbf{1}_{L' \times 1}\mathbf{b}_V$$

Step 2: Calculate the scores $\boldsymbol{S} \in \mathbb{R}^{L \times L'}$: $\boldsymbol{S} = \mathbf{Q}\mathbf{K}^{\mathrm{T}}$

Step 3: Apply the Mask

Step 4: Calculate the attention: $\mathbf{X}'' = \mathrm{softmax}\left(\mathbf{S}/\sqrt{d_k}\right)\mathbf{V}$.

Step 5: For Multi-head attention the $\mathbf{X}''$ results from the linear projection of the $\mathbf{X}_i''$ concatenation: $\mathbf{X}'' = [\mathbf{X}_1'', \dots, \mathbf{X}_H'']\mathbf{W}_\mathbf{0}$

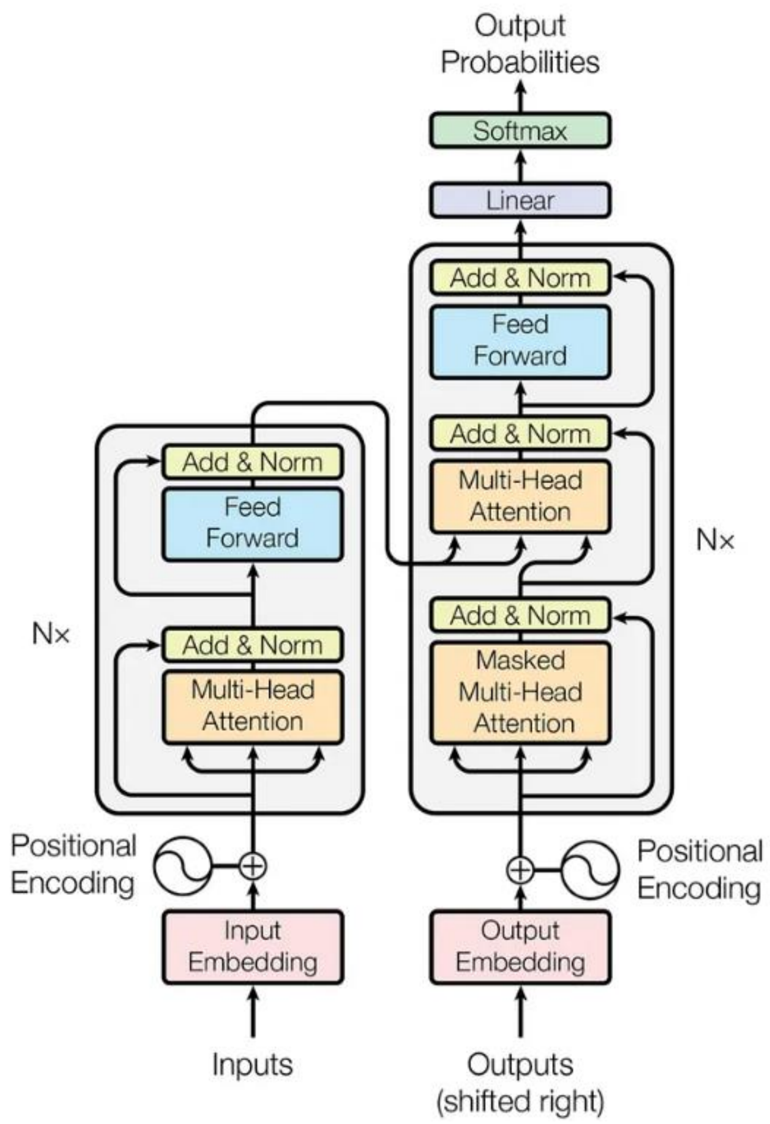
Artificial Intelligence & Information Analysis Lab

LLM Building Blocks

Attention:

State of the art LLMs are based upon the Transformer architecture.

- The basic building block of a Transformer architecture is the multi-head scaled dot-product attention unit.
- The remaining blocks of the overall architecture consist of normalization and point-wise, fully connected layers.



Transformer architecture [VAS2017].

Artificial Intelligence & Information Analysis Lab

LLM Building Blocks

Attention:

Transformer models usually consist of an ***encoder-decoder*** architecture, with several encoder/decoder layers stacked on top of each other.

- The Encoder consists of two sub-layers: a ***multi-head self attention module*** and a ***position-wise*** fully connected ***feed-forward*** network.
- The Decoder consists of three sub-layers: a ***multi-head self attention module***, a ***position-wise*** fully connected ***feed-forward*** network and a ***multi-head cross-attention module***.

LLM Building Blocks

Multi-head Attention:

- Counteracts the reduced effective resolution due to averaging attention-weighted positions in single attention.
- Multi-head attention provides multiple low-scale featured map compared to a single map obtained by single attention.
- Multiple attention head are analogous to multiple kernels in a single layer in a CNN.

LLM Building Blocks

Multi-head Attention:

- Jointly attend information from different representation subspaces at different positions capturing richer interpretations (various patterns and dependencies).
- Redundancy is introduced making the model more resilient to noise or errors in individual heads (robustness).

Large Language Models

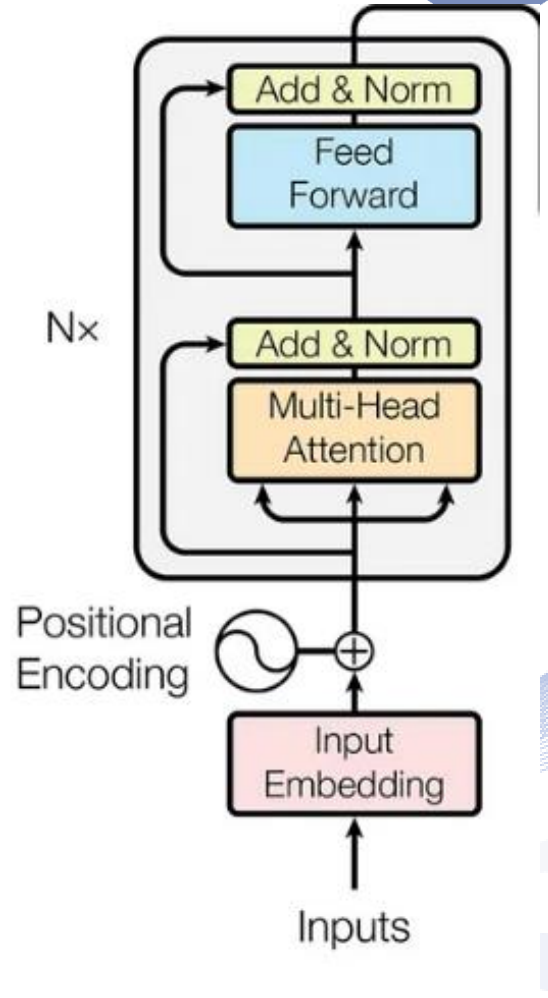
- Introduction
- LLM Building Blocks
- **Encoder-only LLMs**
- Decoder-only LLMs
- Encoder – Decoder LLMs
- LLM tasks

Encoder-only LLMs

Encoder-only LLMs: BERT (Google), RoBERTa (META) and DeBERTa (Microsoft).

BERT: A bidirectional transformer trained on the task of mask language modeling. It is a Discriminative model.

Mask modeling [PHU2022] : Given a text the goal is to correctly recover the masked-out tokens. Each input is replaced with a mask_token with probability p_{mask}.



Transformer encoder [VAS2017].

Encoder-only LLMs

Input: $\mathbf{X} \in \mathcal{R}^{L \times n_v}$, token IDs.

Output: $\mathbf{P} \in (0,1)^{L \times n_v}$, each column denotes a probability distribution over the vocabulary.

LLM parameter vector $\boldsymbol{\theta}$ containing:

- Token embedding/unembedding and positional matrices.
- Multi-head attention parameters for the lth layer.
- Layer-normalization parameters
- MLP weights parameters
- Final projection and layer-norm parameters.

Hyperparameters: $D, L, H, L_{mlp}, d_m, d_{mlp}, d_f \in \mathbb{N}$.

Encoder-only LLMs

Encoder-only model overview [PHU2022] :

Given a matrix $\mathbf{X} \in \mathbb{R}^{L \times n_v}$ of one-hot tokens, the full transformer Encoder-only model $\mathcal{T}$ first acts on $\mathbf{X}$ via the embedding, then via the encoder structure and then finally via unembedding:

$$\mathcal{T}: \mathcal{R}^{L \times n_v} \xrightarrow{Embedding} \mathcal{R}^{L \times d_m} \xrightarrow{Encoder} \mathcal{R}^{L \times d_m} \xrightarrow{\mathrm{GELU}(\mathrm{W_f X''} + \mathrm{b_f 1^T})} \mathcal{R}^{L \times d_m} \xrightarrow{Unembedding} \mathcal{R}^{L \times n_v}$$

Encoder-only LLMs

Encoder stack [PHU2022]:

- Let $\{H_i\}_{i=1}^{D}$ be a set of attention multi-heads, each H_i is a set of attention heads $\{h_i\}_{i=1}^{H}$ and let $\{m_i\}_{i=1}^{D}$ be a set of MLPs. Each multi-head has the same number of heads and the same dimensions and each MLP has L_{mlp} layers.
- The encoder stack is a composition of n blocks of $\{B_i(H_i, m_i)\}_{i=1}^{n}$ building hierarchical text representations to capture high level text features and dependencies:

$$\mathcal{R}^{n\times d} \xrightarrow{B_1(H_1,m_1)} \mathcal{R}^{n\times d} \xrightarrow{B_2(H_2,m_2)} \dots \xrightarrow{B_{D-1}(H_{D-1},m_{D-1})} \mathcal{R}^{n\times d} \xrightarrow{B_D(H_D,m_D)} \mathcal{R}^{n\times d}.$$

Encoder-only LLMs

Encoder stack stages [PHU2022]:
single attention H_i, Feed Forward (FF) model, multi-layer perceptron m_i:

$$H_i: \mathbf{X}'' = \mathbf{X}' + \sum_{h_j \in H_i} h_j(\mathbf{X}', Mask \equiv 1),$$

$$FF: \mathbf{X}''' = \mathbf{X}'' + m_i(\mathbf{X}''),$$

$$m_i(\mathbf{X}'') = \mathbf{W}_{mlp2}\mathrm{GELU}\left(\mathbf{W}_{mlp1}\mathbf{X}'' + \mathbf{b}_{mlp1}\mathbf{1}^T\right) + \mathbf{b}_{mlp2}\mathbf{1}^T,$$

$$B_i(\mathbf{X}') = \mathbf{X}'' + m\left(\mathbf{X}' + \sum_{h_j \in H_i} h_j(\mathbf{X}', Mask \equiv 1)\right).$$

Encoder-only LLMs

Bidirectional Encoder:

- ***Encoder-only LLMs use no masking*** hence the self-attention implementation is $h(\mathbf{X}, \mathrm{Mask} \equiv 1)$.
- As a result, given a sequence of token representations all tokens are treated as context $\mathrm{X} = \mathrm{Z}$.

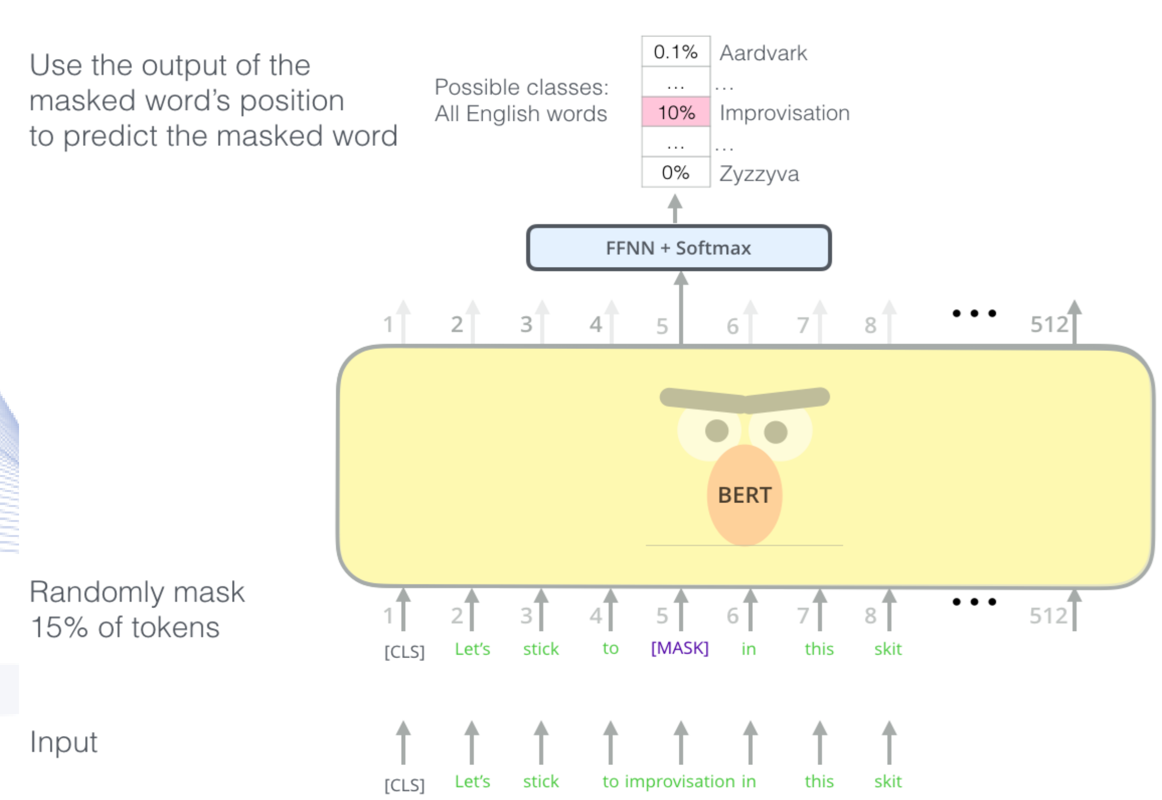
BERT Training

Bidirectional Encoder Representations from Transformers (***BERT***) architecture:

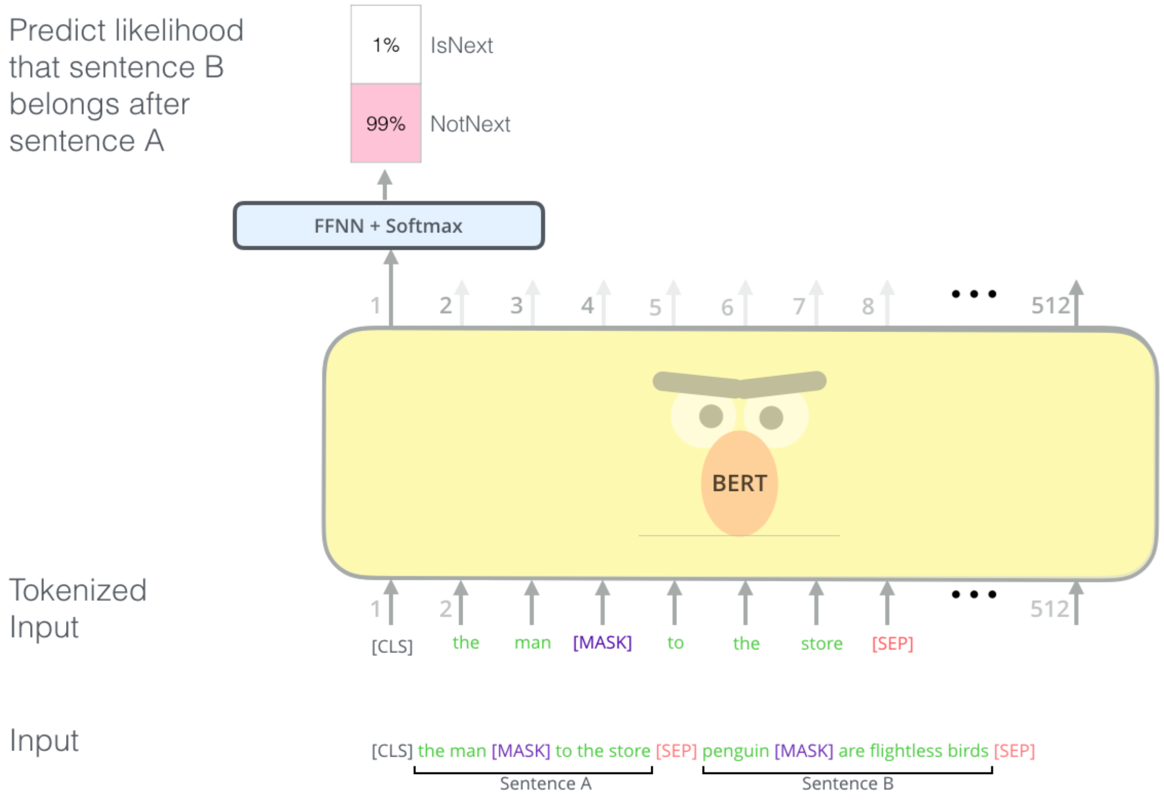
- Multi-layer bidirectional Transformer encoder.

- ***BERT unsupervised pre-training*** consists of two tasks:
  - ***Mask Language Model*** finds the masked/hidden words by looking at their context.
  - ***Next Sentence Prediction*** predicts the appearance order two input sentences A, B.

BERT Training



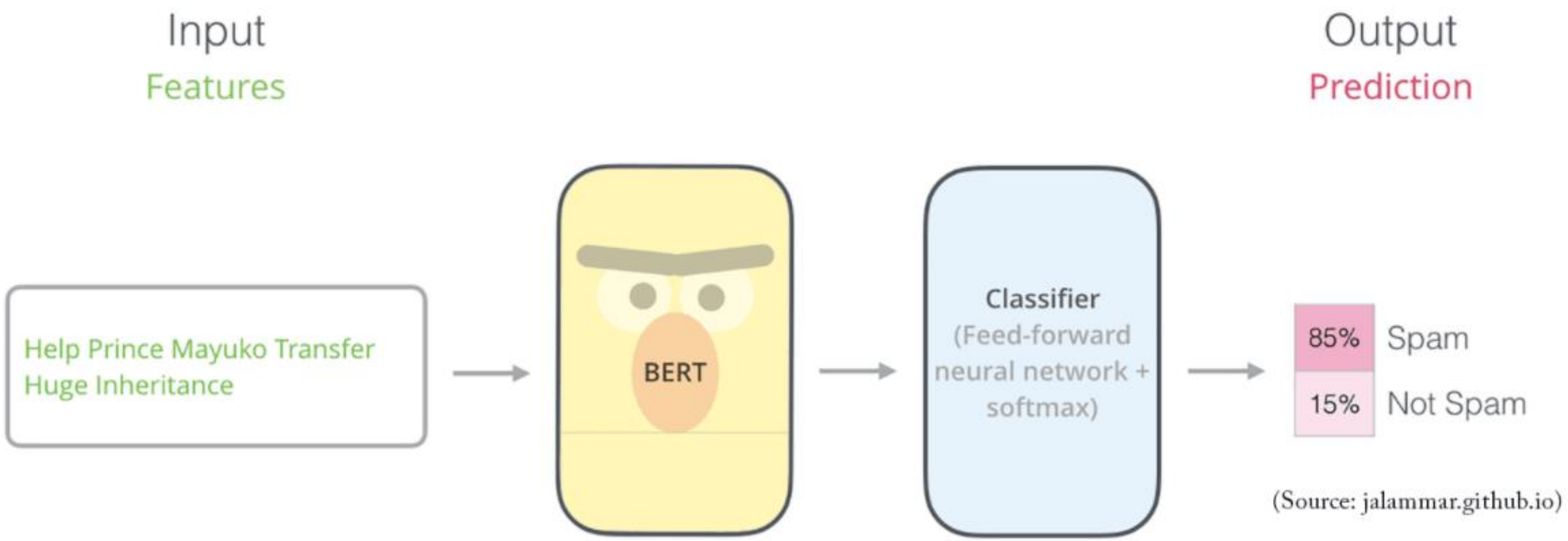
Masked Language Model.



Next sentence prediction.

BERT pre-training

BERT Training



Bert Fine-tuning: supervised training on a specific task.

BERT Training

BERT accuracy in different tasks.

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Average
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Leaderboards

TREND	DATASET	BEST METHOD	PAPER TITLE
	SST-2 Binary classification	T5-3B	Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer
	SST-5 Fine-grained classification	RoBERTa-large+Self-Explaining	Self-Explaining Structures Improve NLP Models
	IMDb	NB-weighted-BON + dv-cosine	Sentiment Classification Using Document Embeddings Trained with Cosine Similarity
	Yelp Binary classification	BERT large	Unsupervised Data Augmentation for Consistency Training
	Yelp Fine-grained classification	BERT large	Unsupervised Data Augmentation for Consistency Training
	MR	byte mLSTM7	A La Carte Embedding: Cheap but Effective Induction of Semantic Feature Vectors
	Amazon Review Polarity	BERT large	Unsupervised Data Augmentation for Consistency Training
	Amazon Review Full	BERT large	Unsupervised Data Augmentation for Consistency Training
	SemEval 2014 Task 4 Subtask 1+2	GRACE	GRACE: Gradient Harmonized and Cascaded Labeling for Aspect-based Sentiment Analysis
	CR	Block-sparse LSTM	GPU Kernels for Block-Sparse Weights
	Multi-Domain Sentiment Dataset	Distributional Correspondence Indexing	Revisiting Distributional Correspondence Indexing: A Python Reimplementation and New Experiments
	MPQA	STM+TSED+PT+2L	The Pupil Has Become the Master: Teacher-Student Model-Based Word Embedding Distillation with Ensemble Learning
	DBRD	RobBERT v2	RobBERT: a Dutch RoBERTa-based Language Model
	Twitter	AEN-BERT	Attentional Encoder Network for Targeted Sentiment Classification

Artificial Intelligence & Information Analysis Lab

Large Language Models

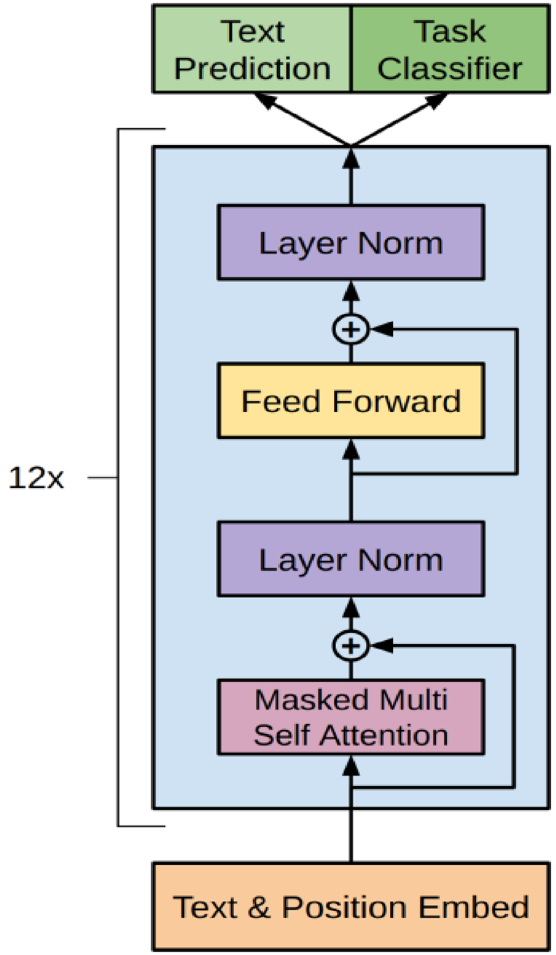
- Introduction
- LLM Building Blocks
- Encoder-only LLMs
- **Decoder-only LLMs**
- Encoder – Decoder LLMs
- LLM tasks

Decoder-only LLMs

Decoder-only LLMs: GPTs (1, 2, 3, Chat, 4 by OPEN-AI), LaMDA, PaLM and Bard (by Google) and OPT, LLaMA (by Meta)

Autoregressive language modeling [PHU2022] : Given an incomplete sentence the goal is to predict the next token. It is a Generative model.

Differences: In contrast to BERTs bidirectional attention Decoder-only LLMs use unidirectional attention and apply in a different order the layer-normalization.



GPT architecture [RAD2018].

Artificial Intelligence & Information Analysis Lab

Decoder-only LLMs

Input: $\mathbf{X} \in \mathcal{R}^{L \times n_v}$, token IDs.

Output: $\mathbf{P} \in (0,1)^{L \times n_v}$, where the t-th column of $\mathbf{P}$ represents the probability $\widehat{\mathbf{P}}_\theta(\mathbf{X}[L+1]|\mathbf{X}[0:L])$.

Parameter vector $\boldsymbol{\theta}$ comprises:

- Token embedding/unembedding and positional matrices.
- Multi-head attention parameters for the lth layer.
- Layer-normalization parameters
- MLP weights parameters
- Final layer-norm parameters.

Hyperparameters: $D, L, H, d_e, d_{mlp} \in \mathbb{N}$.

Decoder-only LLMs

Decoder-only model overview [PHU2022]:

Given a matrix $\mathbf{X} \in \mathbb{R}^{L \times n_v}$ of one-hot tokens, the full transformer Decoder-only model $\mathcal{T}$ first acts on $\mathbf{X}$ via the embedding, then via the decoder structure and then finally via unembedding:

$$\mathcal{T}\colon \mathcal{R}^{L \times n_v} \xrightarrow{Embedding} \mathcal{R}^{L \times d} \xrightarrow{Decoder} \mathcal{R}^{L \times d} \xrightarrow{Unembedding} \mathcal{R}^{L \times n_v}.$$

Decoder-only LLMs

Decoder stack [PHU2022] :

Let $\{H_i\}_{i=1}^{D}$ be a set of attention multi-heads, each H_i is a set of attention heads $\{h_i\}_{i=1}^{H}$ and let $\{m_i\}_{i=1}^{D}$ be a set of MLPs. Each multi-head has the same number of heads and the same dimensions and each MLP has L_{mlp} layers. The Decoder stack is a composition of n blocks of $\{B_i(H_i, m_i)\}_{i=1}^{n}$. The multiple stacked decoder blocks build hierarchical representations to capture high level features and dependencies.

$$\mathcal{R}^{n\times d} \xrightarrow{B_1(H_1,m_1)} \mathcal{R}^{n\times d} \xrightarrow{B_2(H_2,m_2)} \ldots \xrightarrow{B_{D-1}(H_{D-1},m_{D-1})} \mathcal{R}^{n\times d} \xrightarrow{B_D(H_D,m_D)} \mathcal{R}^{n\times d}.$$

Decoder-only LLMs

Decoder stack [PHU2022].

Single attention model H_i, the Feed Forward model FF, multi-layer perceptron m_i:

$$H_i: \quad \mathbf{X}'' = \mathbf{X}' + \sum_{h_j \in H_i} h_j\big(\mathbf{X}', Mask[t, t;] \equiv [[t \le t']]\big),$$

$$FF: \quad \mathbf{X}''' = \mathbf{X}'' + m_i(\mathbf{X}''),$$

$$m_i(\mathbf{X}'') = \mathbf{W}_{mlp2}\mathrm{GELU}\big(\mathbf{W}_{mlp1}\mathbf{X}'' + \mathbf{b}_{mlp1}1^{\mathrm{T}}\big) + \mathbf{b}_{mlp2}\mathbf{1}^{T},$$

$$B_i(\mathbf{X}') = \mathbf{X}'' + m\left(\mathbf{X}' + \sum_{h_j \in H_i} h_j\big(\mathbf{X}', Mask[t, t;] \equiv [[t \le t']]\big)\right).$$

Decoder-only LLMs

Decoder autoregressive masking [PHU2022]:
The Decoder-only LLMs implement unidirectional attention instead of bidirectional. Hence in self attention for each token only the preceding tokens are treated as context. The applied mask is an $(L \times L)$ matrix:

$$\mathbf{M} = \begin{pmatrix} 0 & -\infty & -\infty \\ 0 & 0 & -\infty \\ 0 & 0 & 0 \end{pmatrix}, \qquad m_{ij} = \begin{cases} 0, i \leq j \\ -\infty, else \end{cases}.$$

As a result of the unidirectional attention this causal autoregressive version can be used only for online prediction.

GPT Training stages

Unsupervised Pre-training stage:

- *Training dataset*: BooksCorpus [ZHU2015].
- *Objective*: Standard language modeling [RAD2018].

Fine-tuning stage:

- *Training dataset*: a labelled dataset corresponding to the fine-tuning task
- *Objective*: GPT model parameters adaptation to the supervised target task and language modeling [RAD2018].

In-context Learning

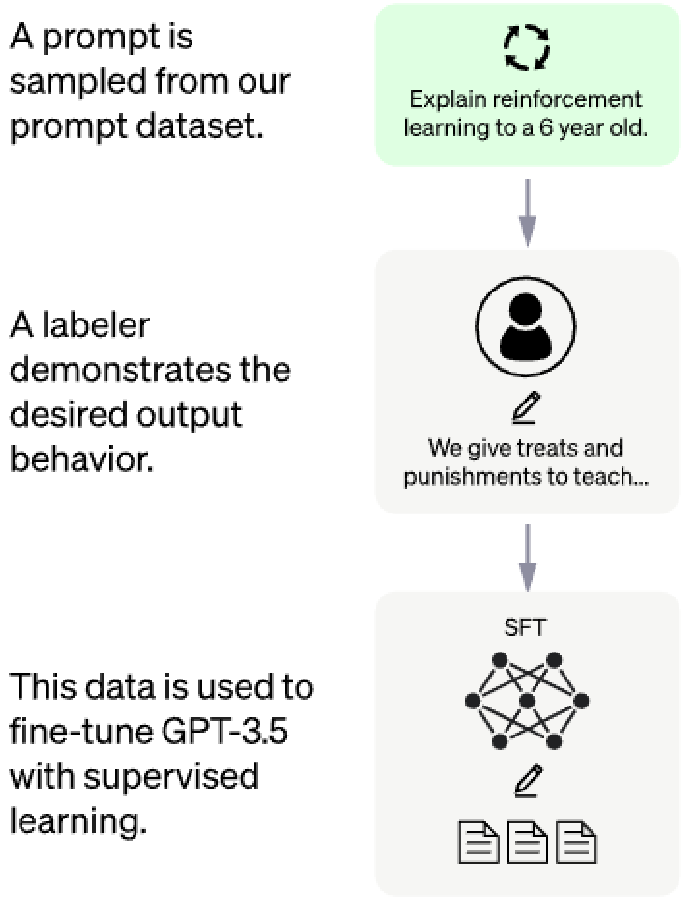
- ***Zero-shot learning***: GPT model input is: a) a task description b) prompt.
- Example: *Translate English to French* (task description), *cheese* (prompt).

- ***One-shot learning***: GPT model input is: a) task description and b) a single task example (from the training dataset).
- ***Few-shot learning***: GPT model input is: a) task description and b) few task examples (from the training dataset).

ChatGPT Fine-Tuning

A pre-trained 3rd generation GPT DNN for language tasks is acquired.

- Step 1: Fine-tune the pre-trained GPT DNN on a labelled dataset [OPE2023].



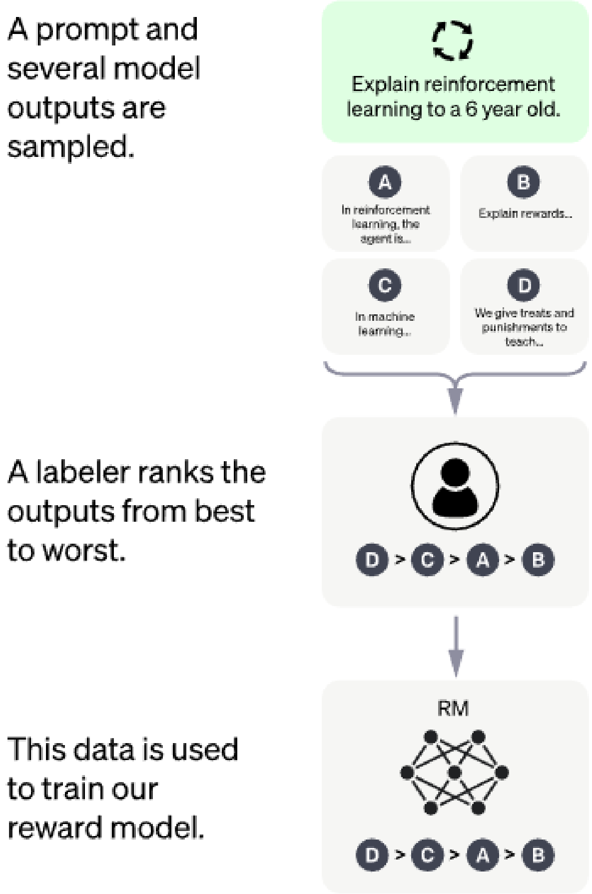
ChatGPT fine-tuning (step 1) [OPE2023].

Artificial Intelligence & Information Analysis Lab

ChatGPT Fine-Tuning

- Step 2: A reward model is trained with a scalar output.

- The output quantifies how good was the response of the fine-tuned GPT to a given prompt.



Step 2 of ChatGPT fine-tuning: reward model training [OPE2023].

ChatGPT Fine-Tuning

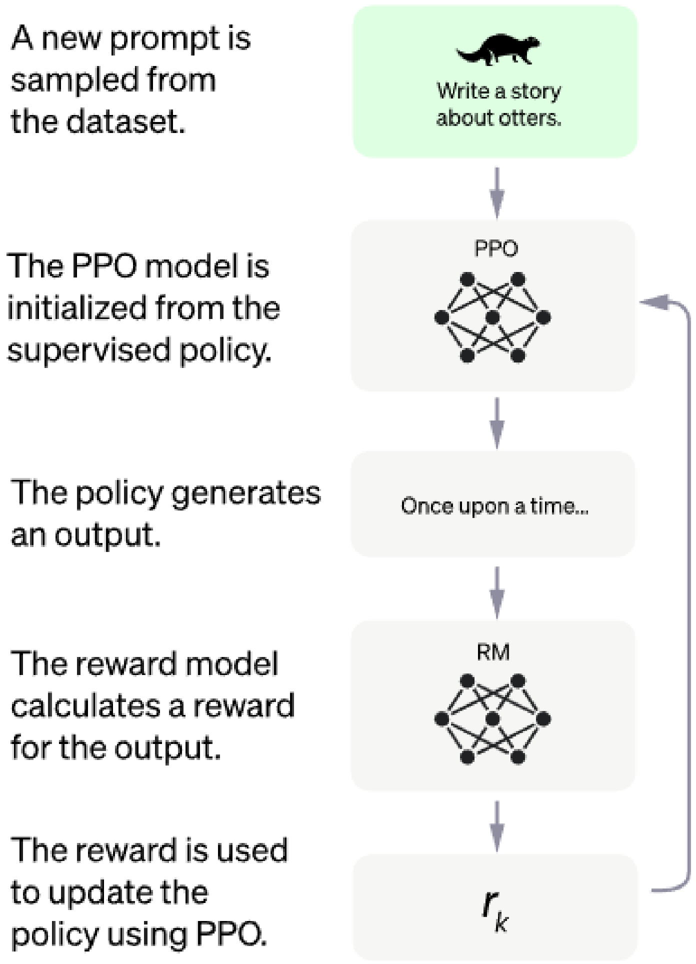
ChatGPT reward model:

- It is trained on a dataset oof responses returned by the fine-tuned GPT-3 for a given prompt [OPE2023].
- For each prompt the fine-tuned GPT outputs four responses according to a decoding strategy, sampling from responses with the highest probability.
- The responses are labelled by determining a reward proportional to the quality of each output.
- Non toxic and factual responses are given a higher reward.

ChatGPT Fine-Tuning

Reinforcement Learning with Human Feedback (***RLHF***).

- Step 3: The ***On-policy Proximal Policy Optimization*** (***PPO***) reinforcement learning algorithm is fine-tuned to optimize the scalar reward output of the reward model [OPE2023].



Step 3 of chatGPT fine-tuning using RLHF [OPE2023].

Artificial Intelligence & Information Analysis Lab

ChatGPT Reasoning

- Despite performing well on certain reasoning tasks, ChatGPT is unreliable, as its responses are inconsistent [BAN2023].
  - Its reasoning evaluation was performed via question answering.
- ChatGPT has acceptable performance in deductive. Abductive, temporal, causal and analogical reasoning [BAN2023].
- ChatGPT has weakness in inductive, spatial, mathematical, non-textual semantic and multi-hop reasoning [BAN2023].

ChatGPT Reasoning

Categories	Testset	Results
Deductive	ENTAILMENTBANK bAbI	28/30
Inductive	CLUTRR	13/30
Abductive	αNLI	26/30
Mathematical	Math	13/30
Temporal	Timedial (formatted)	26/30
Spatial	SpartQA StepGame (hard) StepGame (diagonal) StepGame (clock-direction)	12/30 7/30 11/20 5/20
Common sense	CommonsenseQA Pep-3k (Hard)	27/30 28/30
Causal	E-Care	24/30
Multi-hop	hotpotQA	8/30
Analogical	Letter string analogy	30/30

ChatGPT results on reasoning tasks [BAN2023].

ChatGPT Questionmarks

- ***Does ChatGPT have access to external resources?***
  - Knowledge graphs? Algebraic computations (Symbolic Algebra)?
  - If not, what is its ***knowledge storage capacity***?

- ***Does ChatGPT have explicit reasoning mechanisms***?
  - Texts contain many examples of reasoning.
  - Reasoning as a result of learning-by-examples?
  - ***Implicit/approximate reasoning***?

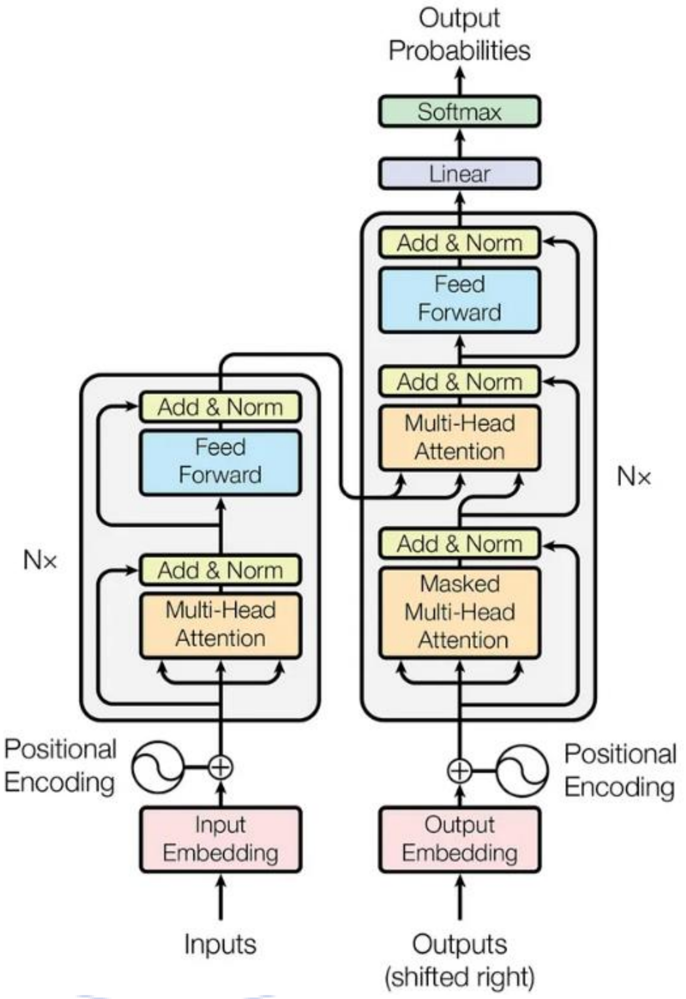
Large Language Models

- Introduction
- LLM Building Blocks
- Encoder-only LLMs
- Decoder-only LLMs
- **Encoder – Decoder LLMs**
- LLM tasks

Encoder – Decoder LLMs

Encoder-Decoder LLMs: Mainly T5, Flan T5, UL2 and Flan UL2 (Google).

This model is used for sequence-to-sequence tasks, such as machine translation.



Transformer architecture [VAS2017].

Encoder-Decoder LLMs

Input: $\mathbf{X}, \mathbf{Y} \in \mathcal{R}^{L \times n_v}$, token IDs.

Output: $\mathbf{P} \in (0,1)^{L \times n_v}$, where the t-th column of $\mathbf{P}$ represents the probability $\widehat{\mathbf{P}}_\theta(\mathbf{X}[L+1] | \mathbf{X}[0:L])$

Parameter vector $\boldsymbol{\theta}$ comprises:

- Token embedding/unembedding and positional matrices.
- Encoder: Multi-head attention parameters for each l layer, Layer-norm parameters and MLP weights parameters
- Decoder: Multi-head attention parameters for each l layer, Multi-head cross attention parameters for each l layer, Layer-norm parameters and MLP weights parameters.

Hyperparameters: $D_{enc}, D_{dec}, L_{enc}, L_{dec}, H, d_e, d_{mlp} \in \mathbb{N}$.

Encoder-Decoder LLMs

Encoder-Decoder model overview [PHU2022]:

Given two matrices $\mathbf{X} \in \mathbb{R}^{L_{enc} \times n_v}, \mathbf{Y} \in \mathbb{R}^{L_{dec} \times n_v}$ of one-hot tokens, the full transformer model $\mathcal{T}$ will be defined as first acting on the $\mathbf{X}$ context sequence via bidirectional multi-head attention, then on the $\mathbf{Y}$ primary sequence first via unidirectional multi-head attention and then combined with output of the encoder via multi-head cross attention.

$$\mathcal{T} = \begin{cases} \mathbf{X} \in \mathcal{R}^{L_{enc} \times n_{vocab}} \xrightarrow{Embedding} \mathcal{R}^{L_{enc} \times d} \xrightarrow{Encoder} \mathcal{R}^{L_{enc} \times d} \\ \mathbf{Y} \in \mathcal{R}^{L_{dec} \times n_v} \xrightarrow{Embedding} \mathcal{R}^{L_{dec} \times d} \xrightarrow[Encoder\ Output]{Decoder+} \mathcal{R}^{L_{dec} \times d} \xrightarrow{Unembedding} \mathcal{R}^{L_{dec} \times n_v} \end{cases}$$

Encoder-Decoder LLMs

Encoder:

The encoder is the same one as the one used in the Encoder-only LLMs with the difference than in the MLP instead of GELU activation the ReLU is used.

$$\mathcal{R}^{L_{enc}\times d} \xrightarrow{B_1(H_1,m_1)} \mathcal{R}^{L_{enc}\times d} \xrightarrow{B_2(H_2,m_2)} \dots \xrightarrow{B_{D_{enc}}(H_{D_{enc}},m_{D_{enc}})} \mathcal{R}^{L_{enc}\times d}.$$

Encoder-Decoder LLMs

Decoder:

The decoder in addition to the structure defined in the Decoder-only LLMs uses an extra multi-head cross attention and in the MLP instead of GELU activation the ReLU is used.

$$\mathcal{R}^{L_{dec}\times d} \xrightarrow{B_1(H_1,m_1)} \mathcal{R}^{L_{dec}\times d} \xrightarrow{B_2(H_2,m_2)} \dots \xrightarrow{B_{D_{dec}}\left(H_{D_{dec}},m_{D_{dec}}\right)} \mathcal{R}^{L_{dec}\times d}.$$

Artificial Intelligence & Information Analysis Lab

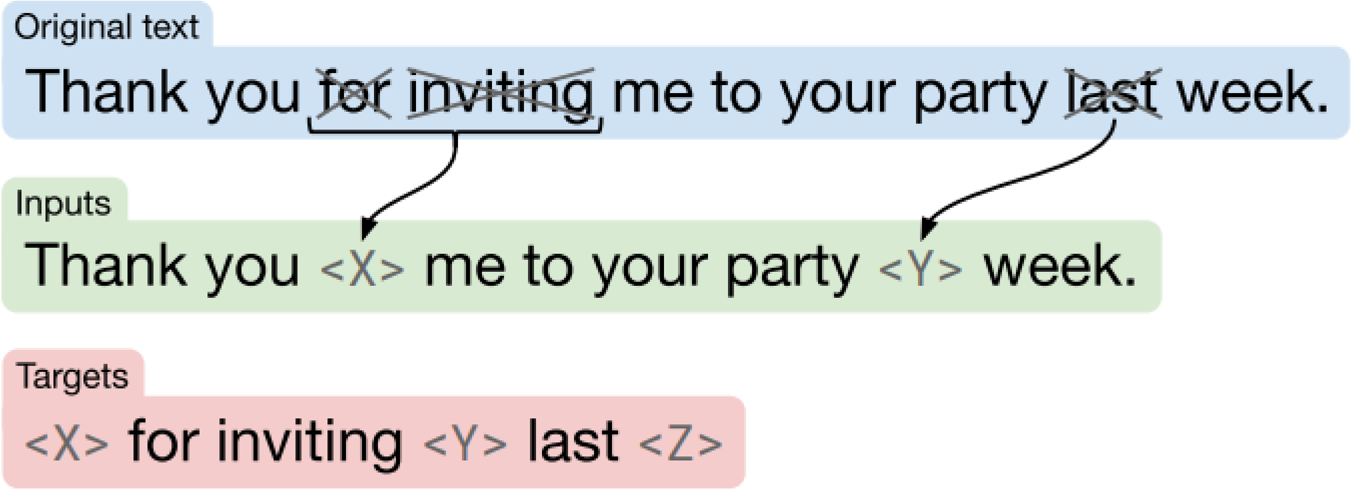
T5-Training

- ***T5 pre-training*** objective:
  - ***Mask Language Model*** finds the masked/hidden words by looking at their context. The difference from BERT is that multiple tokens are replaced by a single keyword.
  - The result is a trained LLM ***that inputs text and outputs text***, where the targets are a sequence, unlike BERT.

- ***T5 fine-training*** tasks:
  - ***Language Translation, Summarization, Sentence Similarity, etc.***

Artificial Intelligence & Information Analysis Lab

T5-Training

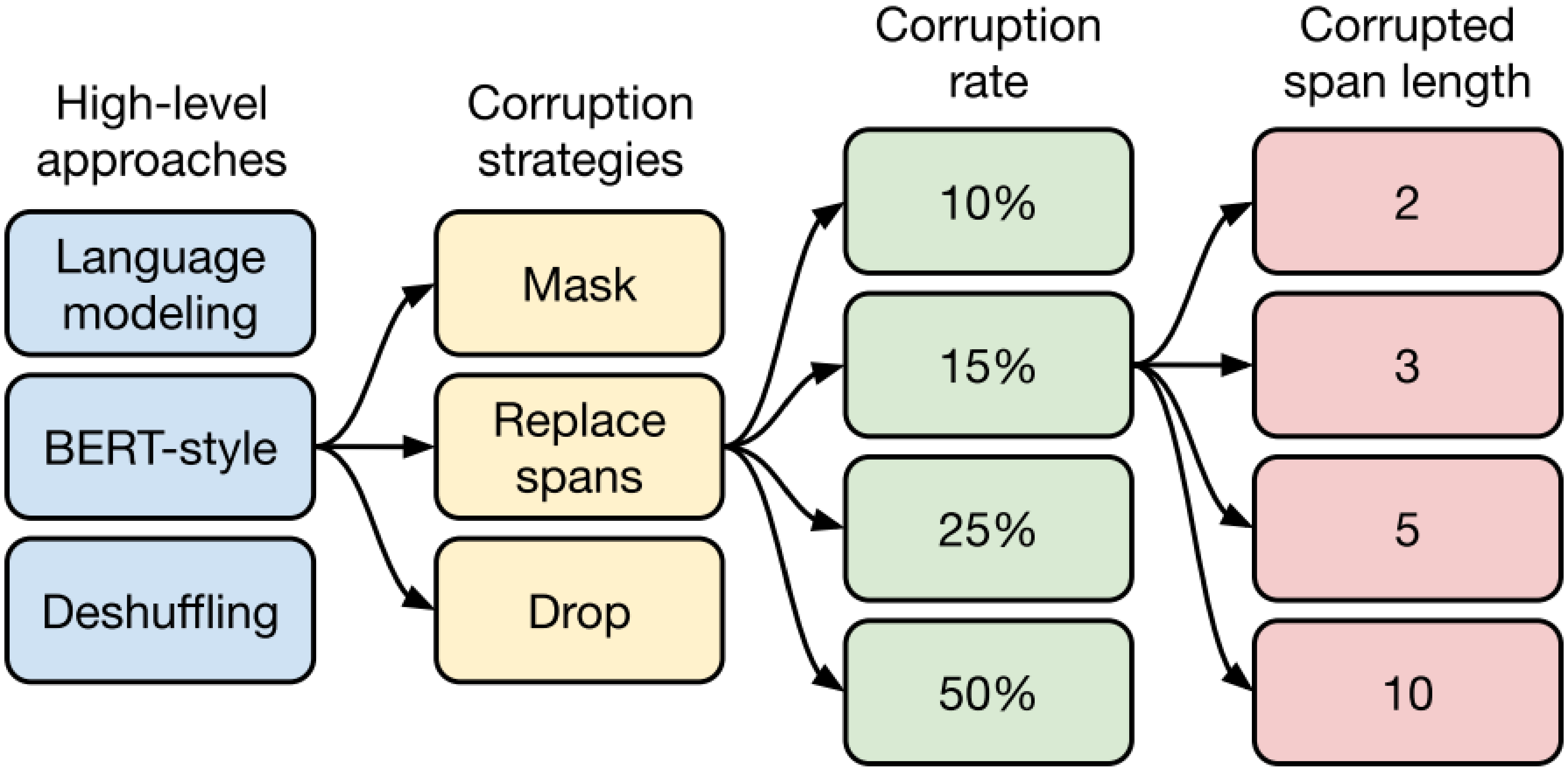


Schematic of the training objective [RAF2020].

T5-Training

- ***T5 hyper-parameter tuning***:
  - ***Pre-training style***: Autoregressive style language modeling, **BERT style Masked Language model objective** and Deshuffling denoising objective.
  - ***Corruption scheme***: Three strategies were used masking a random word**, a span** and dropping a word from input.
  - ***Corruption rate***: Same performance from all rates tested (15% slightly better).
  - ***Corruption length***: Different corruption span length were tested. Model performance degrades as length increases.

T5-Training



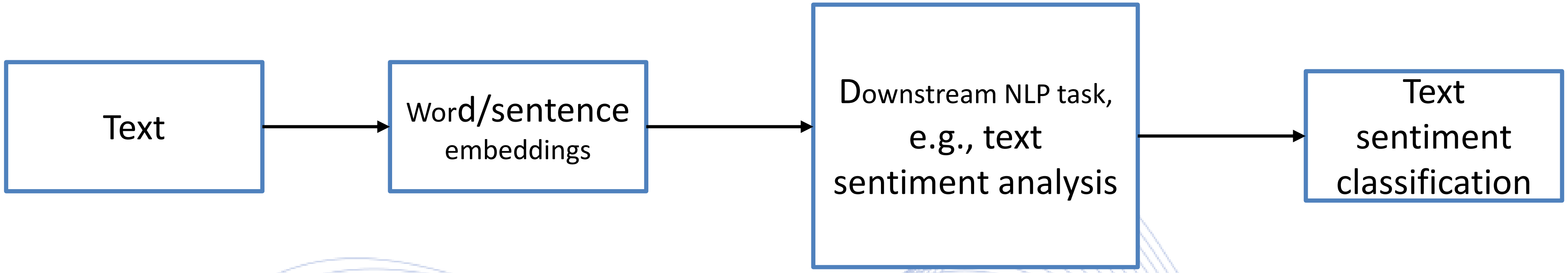
Flow chart of the experimentation on unsupervised objectives [RAF2020].

Artificial Intelligence & Information Analysis Lab

Large Language Models

- Introduction
- LLM Building Blocks
- Encoder-only LLMs
- Decoder-only LLMs
- Encoder – Decoder LLMs
- **LLM tasks**

LLM tasks

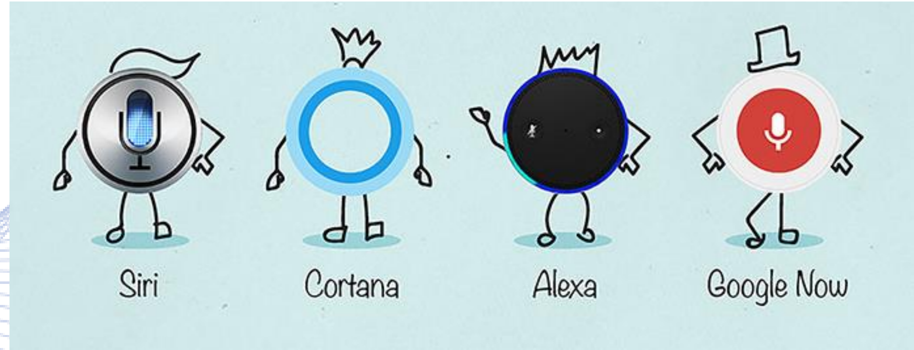


Word embeddings and downstream NLP tasks.

LLM tasks

Voice assistants are complex systems comprising:

- Speech recognition
- Speech to text
- Language analysis
- Dialogue processing
- Information retrieval
- Text to speech output.



LLM tasks

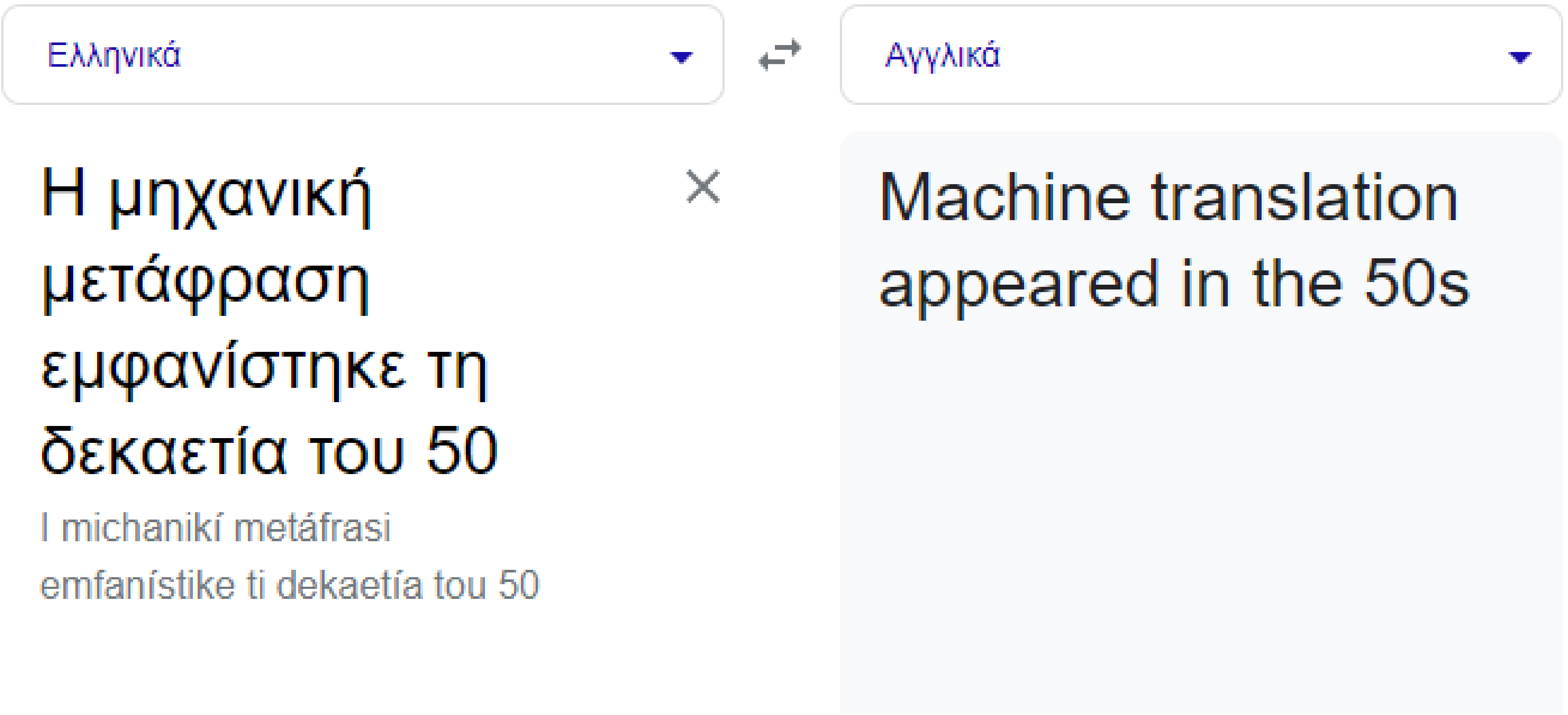
Question Answering.

- What is the weather like today?
- Who is Noam Chomsky?
- How many hours are there in a year?
- Who won the 2022 US elections?

IBM's Watson competed against Jeopardy! champions.

LLM tasks

Machine Translation.



LLM tasks

Text summarization.

- Create an ***abstract*** of an article.
- Extract ***key phrases*** from large piece of text.
- Simplify and condense long documents.

LLM tasks

Information Extraction.

Extraction of ***meaning*** from unstructured data.

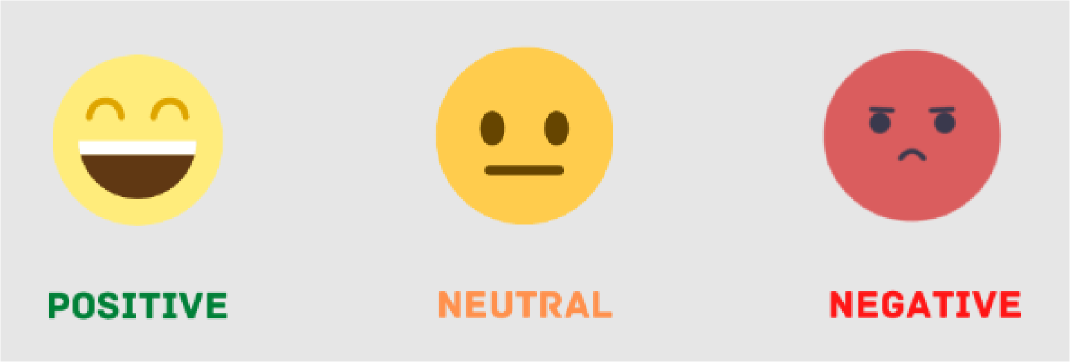
"AP: 45 yo m w/ESRD on HD asthma p/w significant hyperkaliemia"

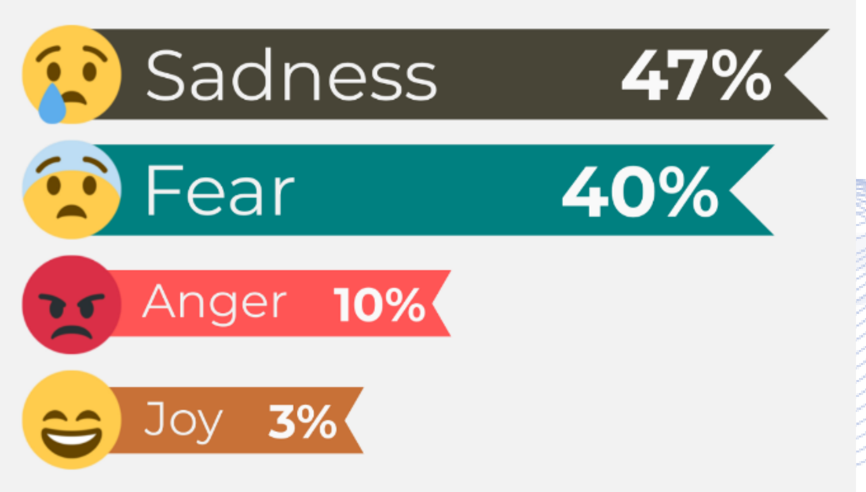
Attributes	Text
Age	45 years old
Sex	Male
Condition	Hypertensive disease
Symptom	Hyperkaliemia

LLM tasks

Text sentiment/emotion analysis.

- ***Text sentiment*** definition.
- Extract text polarity.





- Psychological theories on ***human affect/emotion/sentiment***.

- Useful for marketing and more.

LLM tasks

Association strengths between language models and downstream tasks [LIP2022].

Artificial Intelligence & Information Analysis Lab

LLM tasks

Model	Core differentiator	Pre-training objective	Para-meters	Access	Information Extraction	Text Classification	Conversa-tional AI	Summari-zation	Machine Translation	Content generation
BERT	First transformer-based LLM	AE	370M	Source code	Highly appropriate	Highly appropriate	Appropriate	Appropriate	Somewhat appropriate	Somewhat appropriate
RoBERTa	More robust training procedure	AE	354M	Source code	Highly appropriate	Highly appropriate	Appropriate	Appropriate	Somewhat appropriate	Somewhat appropriate
GPT-3	Parameter size	AR	175B	API	Somewhat appropriate	Somewhat appropriate	Highly appropriate	Appropriate	Appropriate	Highly appropriate
BART	Novel combination of pre-training objectives	AR and AE	147M	Source code	Appropriate	Somewhat appropriate	Appropriate	Highly appropriate	Appropriate	Highly appropriate
GPT-2	Parameter size	AR	1.5B	Source code	Somewhat appropriate	Somewhat appropriate	Highly appropriate	Appropriate	Appropriate	Appropriate
T5	Multi-task transfer learning	AR	11B	Source code	Appropriate	Appropriate	Somewhat appropriate	Highly appropriate	Appropriate	Highly appropriate
LaMDA	Dialogue; safety and factual grounding	AR	137B	No access	Appropriate	Somewhat appropriate	Highly appropriate	Somewhat appropriate	Somewhat appropriate	Appropriate
XLNet	Joint AE and AR	AE and AR	110M	Source code	Appropriate	Highly appropriate	Appropriate	Somewhat appropriate	Somewhat appropriate	Somewhat appropriate
DistilBERT	Reduced model size via knowledge distillation	AE	82M	Source code	Highly appropriate	Appropriate	Appropriate	Appropriate	Somewhat appropriate	Somewhat appropriate
ELECTRA	Computational efficiency	AE	335M	Source code	Highly appropriate	Appropriate	Appropriate	Appropriate	Somewhat appropriate	Somewhat appropriate
PaLM	Training infrastructure	AR	540B	No access	Somewhat appropriate	Appropriate	Highly appropriate	Highly appropriate	Highly appropriate	Highly appropriate
MT-NLG	Training infrastructure	AR and AE	530B	API	Somewhat appropriate	Highly appropriate	Highly appropriate	Appropriate	Somewhat appropriate	Highly appropriate
UniLM	Optimised both for NLU and NLG	Seq2seq, AE and AR	340M	Source code	Appropriate	Appropriate	Highly appropriate	Highly appropriate	Somewhat appropriate	Highly appropriate
BLOOM	Multilingual (46 languages)	AR	176B	Source code	Appropriate	Appropriate	Appropriate	Appropriate	Highly appropriate	Highly appropriate

AR = Autoregression

AE = Autoencoding

Seq2seq = Sequence-to-sequence

Highly appropriate

Appropriate

Somewhat appropriate

Summary of the features of the most popular LLMs [LIP2022].

Bibliography

[PHU2022] Phuong, Mary, and Marcus Hutter. "Formal algorithms for transformers." arXiv preprint arXiv:2207.09238 (2022).

[YAN2023] Yang, Jingfeng, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Bing Yin, and Xia Hu. "Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond." arXiv preprint arXiv:2304.13712 (2023).

[VAS2017] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." Advances in neural information processing systems 30 (2017).

[RAD2018] Radford, Alec, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. "Improving language understanding by generative pre-training." (2018).

[ZHU2015] Y. Zhu et al., "Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 2015, pp. 19-27, doi: 10.1109/ICCV.2015.11.

Bibliography

[AJI2023] Ajitesh Kumar,”Large language models: Concepts and examples”. Data Analytics. (2023, May 1). https://vitalflux.com/large-language-models-concepts-examples/#How_does_LLM_work_Key_Building_Blocks

[LIP2022] Lipenkova, J. “Choosing the right language model for your NLP use case.” Medium. (2022, September 29). https://towardsdatascience.com/choosing-the-right-language-model-for-your-nlp-use-case-1288ef3c4929

[RAF2020] Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. "Exploring the limits of transfer learning with a unified text-to-text transformer." The Journal of Machine Learning Research 21, no. 1 (2020): 5485-5551.

[OPE2023] OpenAI. “CHATGPT: Optimizing Language Models for Dialogue.” OpenAI, OpenAI, 2 Feb. 2023, https://openai.com/blog/chatgpt/.

[BAN2023] Bang, Yejin, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia et al. "A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity." arXiv preprint arXiv:2302.04023 (2023).

Q & A

Thank you very much for your attention!

More material in
http://icarus.csd.auth.gr/cvml-web-lecture-series/

Contact: Prof. I. Pitas
pitas@csd.auth.gr