

Natural Language Processing and Large Language Models

Corso di Laurea Magistrale in Ingegneria Informatica



Lesson 11

From Transformers to LLMs

Nicola Capuano and Antonio Greco


DIEM – University of Salerno



Outline

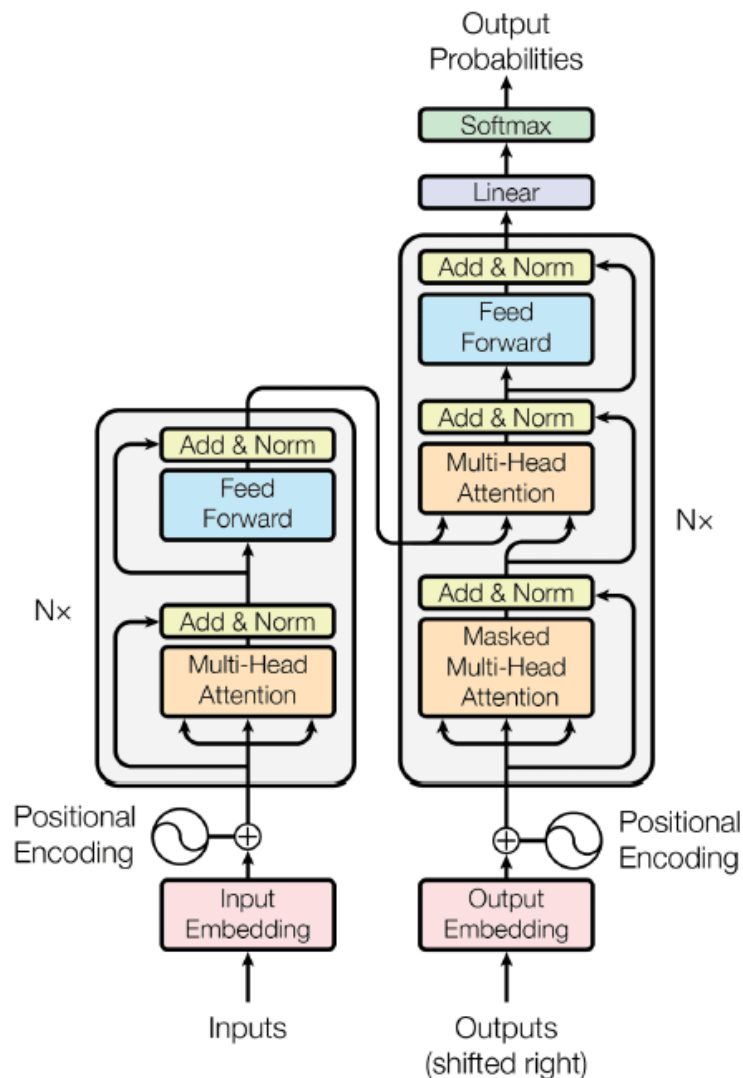
- Transformers for text representation and generation
- Paradigm shift in NLP
- Pre-training of LLMs
- Datasets and data pre-processing
- Using LLMs after pre-training





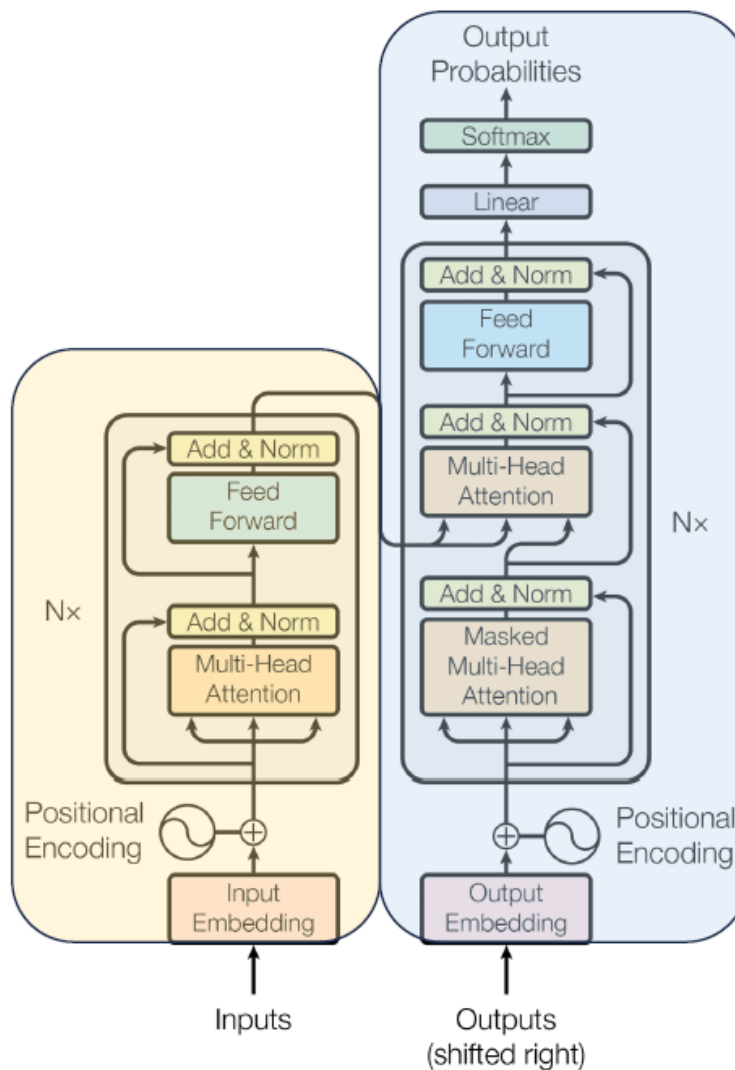
Transformers for text representation and generation

Transformers for text representation and generation



Transformers for text representation and generation

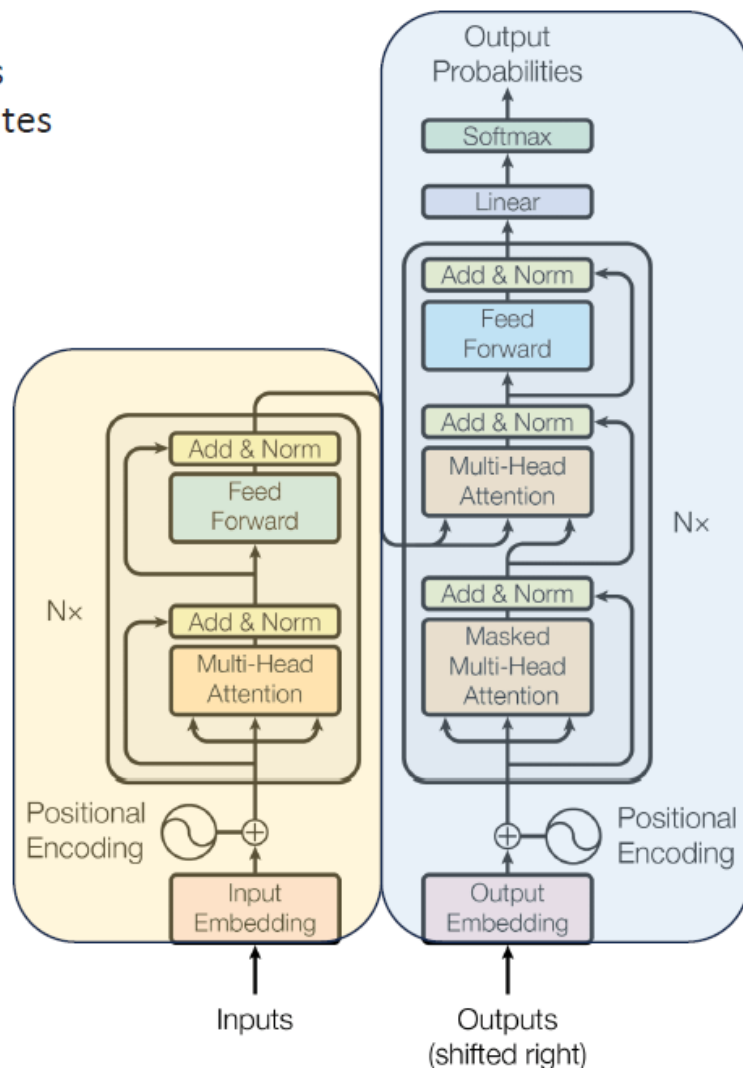
Representation



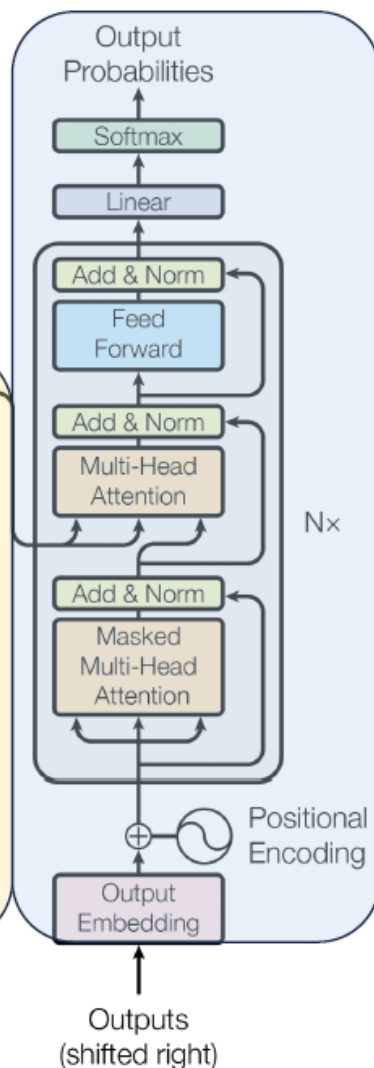
Generation

Transformers for text representation and generation

Input – input tokens
Output – hidden states



Input – output tokens and hidden states*
Output – output tokens



Representation

Generation

Transformers for text representation and generation

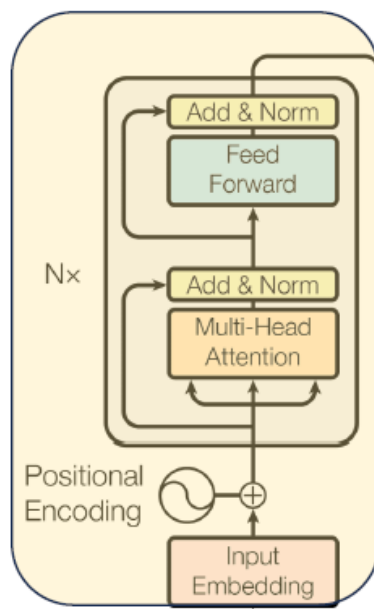
Input – input tokens

Output – hidden states

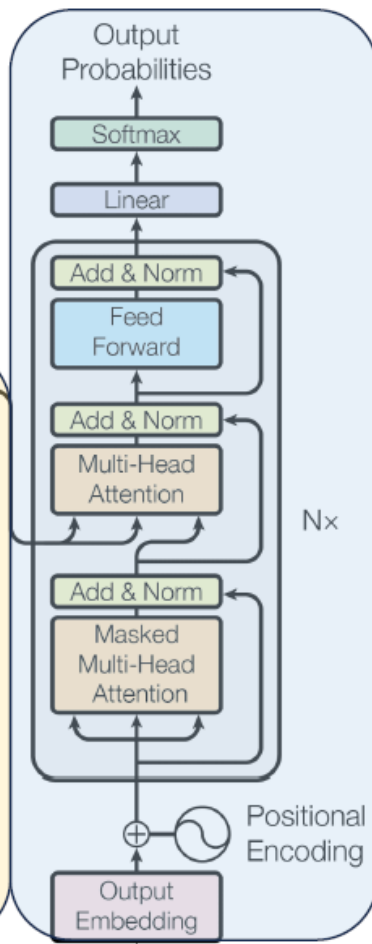
Model can see all timesteps

Does not usually output tokens, so no inherent auto-regressivity

Representation



Inputs



Outputs
(shifted right)

Input – output tokens and hidden states*

Output – output tokens

Model can only see previous timesteps

Model is auto-regressive with previous timesteps' outputs

Generation

Output
Probabilities

Softmax

Linear

Add & Norm

Feed
Forward

Add & Norm

Multi-Head
Attention

Add & Norm

Masked
Multi-Head
Attention

Output
Embedding

$N \times$

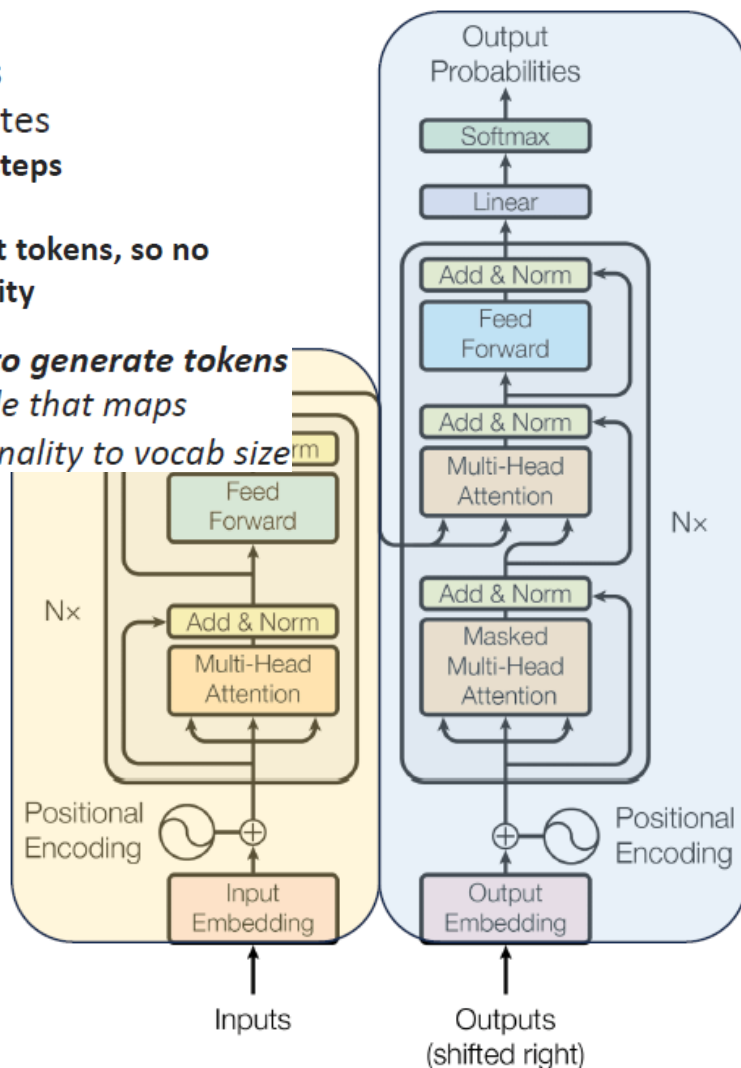
Transformers for text representation and generation

Input – input tokens
Output – hidden states
Model can see all timesteps

Does not usually output tokens, so no inherent auto-regressivity

Can also be adapted to generate tokens by appending a module that maps hidden state dimensionality to vocab size

Representation



Input – output tokens and hidden states*
Output – output tokens
Model can only see previous timesteps

Model is auto-regressive with previous timesteps' outputs

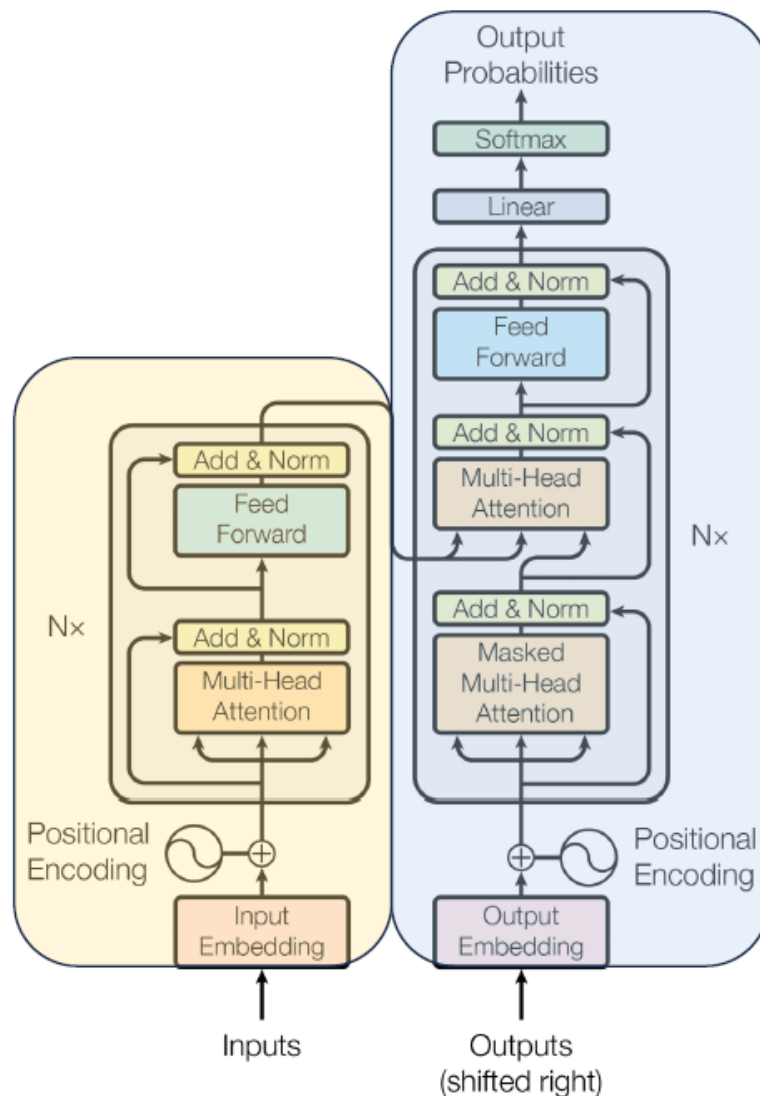
Can also be adapted to generate hidden states by looking before token outputs

Generation

Transformers for text representation and generation

BERT
Oct 2018

Representation



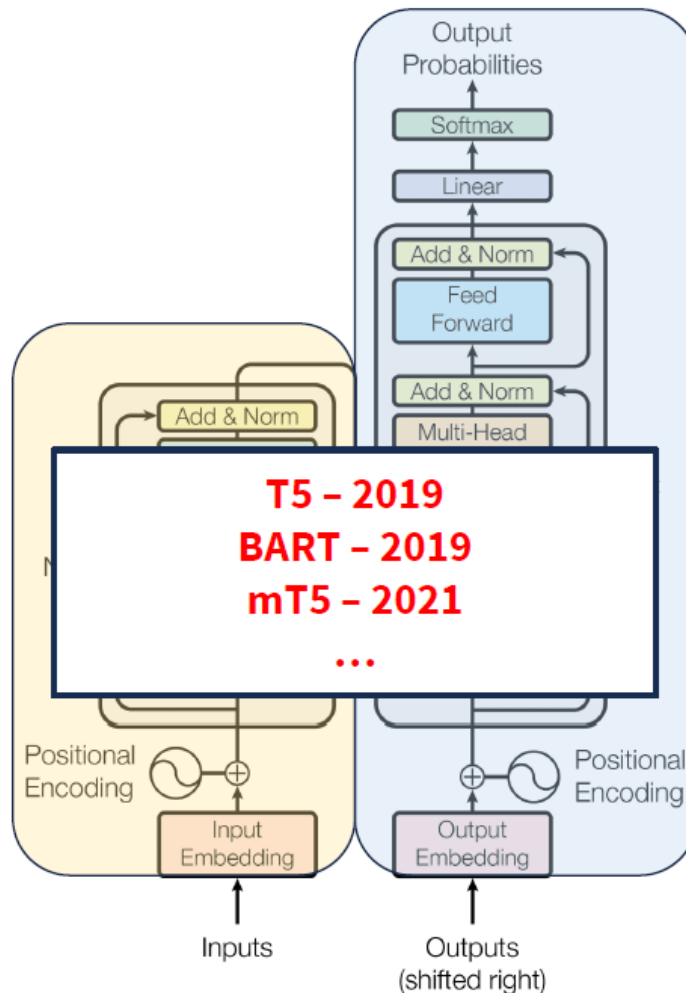
GPT
Jun 2018

Generation

Transformers for text representation and generation

BERT – 2018
DistilBERT – 2019
RoBERTa – 2019
ALBERT – 2019
ELECTRA – 2020
DeBERTa – 2020
...

Representation



GPT – 2018
GPT-2 – 2019
GPT-3 – 2020
GPT-Neo – 2021
GPT-3.5 (ChatGPT) – 2022
LLaMA – 2023
GPT-4 – 2023
...

Generation



Paradigm shift in NLP

Paradigm shift in NLP

Before LLMs

- **Feature Engineering**
 - How do we design or select the best features for a task?

Since LLMs

Paradigm shift in NLP

Before LLMs

- **Feature Engineering**
 - How do we design or select the best features for a task?
- **Model Selection**
 - Which model is best for which type of task?

Since LLMs

Paradigm shift in NLP

Before LLMs

- **Feature Engineering**
 - How do we design or select the best features for a task?
- **Model Selection**
 - Which model is best for which type of task?
- **Transfer Learning**
 - Given scarce labeled data, how do we transfer knowledge from other domains?

Since LLMs

Paradigm shift in NLP

Before LLMs

- **Feature Engineering**
 - How do we design or select the best features for a task?
- **Model Selection**
 - Which model is best for which type of task?
- **Transfer Learning**
 - Given scarce labeled data, how do we transfer knowledge from other domains?
- **Overfitting vs Generalization**
 - How do we balance complexity and capacity to prevent overfitting while maintaining good performance?

Since LLMs

Paradigm shift in NLP

Before LLMs

- **Feature Engineering**
 - How do we design or select the best features for a task?
- **Model Selection**
 - Which model is best for which type of task?
- **Transfer Learning**
 - Given scarce labeled data, how do we transfer knowledge from other domains?
- **Overfitting vs Generalization**
 - How do we balance complexity and capacity to prevent overfitting while maintaining good performance?

Since LLMs

- **Pre-training and Fine-tuning**
 - How do we leverage large scales of unlabeled data out there previously under-leveraged?

Paradigm shift in NLP

Before LLMs

- **Feature Engineering**
 - How do we design or select the best features for a task?
- **Model Selection**
 - Which model is best for which type of task?
- **Transfer Learning**
 - Given scarce labeled data, how do we transfer knowledge from other domains?
- **Overfitting vs Generalization**
 - How do we balance complexity and capacity to prevent overfitting while maintaining good performance?

Since LLMs

- **Pre-training and Fine-tuning**
 - How do we leverage large scales of unlabeled data out there previously under-leveraged?
- **Zero-shot and Few-shot learning**
 - How can we make models perform on tasks they are not trained on?

Paradigm shift in NLP

Before LLMs

- **Feature Engineering**
 - How do we design or select the best features for a task?
- **Model Selection**
 - Which model is best for which type of task?
- **Transfer Learning**
 - Given scarce labeled data, how do we transfer knowledge from other domains?
- **Overfitting vs Generalization**
 - How do we balance complexity and capacity to prevent overfitting while maintaining good performance?

Since LLMs

- **Pre-training and Fine-tuning**
 - How do we leverage large scales of unlabeled data out there previously under-leveraged?
- **Zero-shot and Few-shot learning**
 - How can we make models perform on tasks they are not trained on?
- **Prompting**
 - How do we make models understand their task simply by describing it in natural language?

Paradigm shift in NLP

Before LLMs

- **Feature Engineering**
 - How do we design or select the best features for a task?
- **Model Selection**
 - Which model is best for which type of task?
- **Transfer Learning**
 - Given scarce labeled data, how do we transfer knowledge from other domains?
- **Overfitting vs Generalization**
 - How do we balance complexity and capacity to prevent overfitting while maintaining good performance?

Since LLMs

- **Pre-training and Fine-tuning**
 - How do we leverage large scales of unlabeled data out there previously under-leveraged?
- **Zero-shot and Few-shot learning**
 - How can we make models perform on tasks they are not trained on?
- **Prompting**
 - How do we make models understand their task simply by describing it in natural language?
- **Interpretability and Explainability**
 - How can we understand the inner workings of our own models?

Paradigm shift in NLP

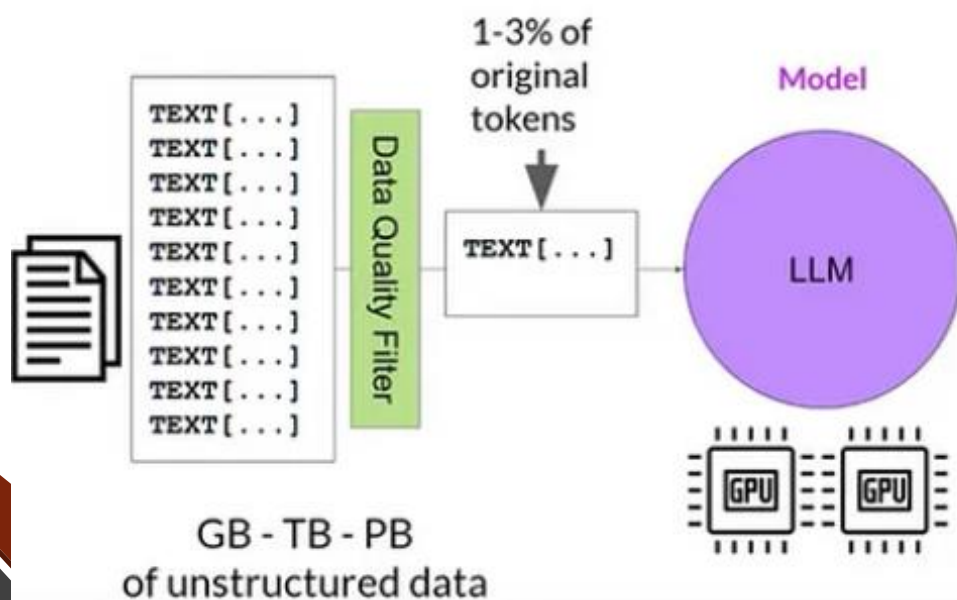
- What has caused this paradigm shift?
 - **Problem in recurrent networks**
 - Information is effectively lost during encoding of long sequences
 - Sequential nature disables parallel training and favors late timestep inputs
 - **Solution: Attention mechanism**
 - Handling long-range dependencies
 - Parallel training
 - Dynamic attention weights based on inputs



Pre-training of LLMs

Self supervised pre-training

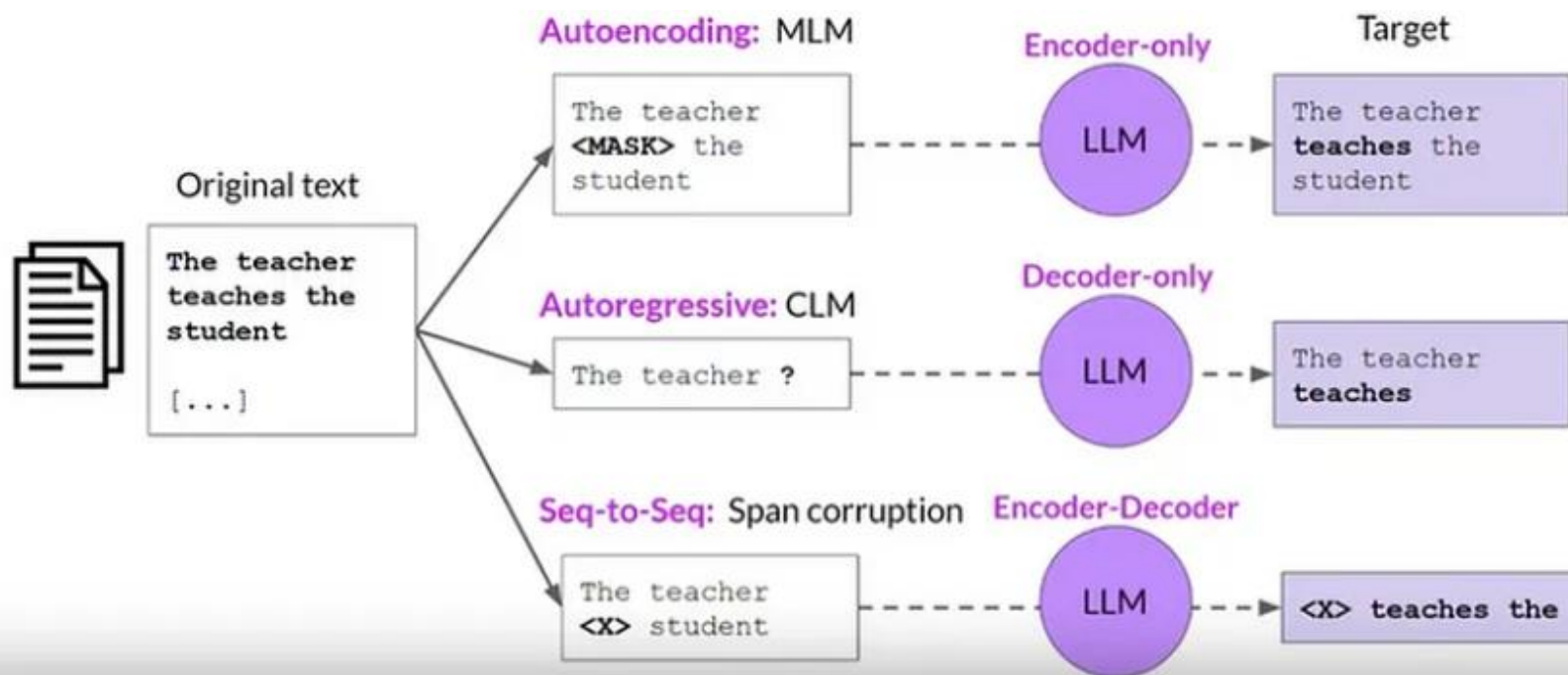
- Pre-training a large language model is done in a self-supervised way, meaning it is trained with unlabeled data which is just text from the internet.
- There is no need to assign labels on the dataset. No supervision? Create supervised tasks and solve them.



Token String	Token ID	Embedding / Vector Representation
'_The'	37	[-0.0513, -0.0584, 0.0230, ...]
'_teacher'	3145	[-0.0335, 0.0167, 0.0484, ...]
'_teaches'	11749	[-0.0151, -0.0516, 0.0309, ...]
'_the'	8	[-0.0498, -0.0428, 0.0275, ...]
'_student'	1236	[-0.0460, 0.0031, 0.0545, ...]
...

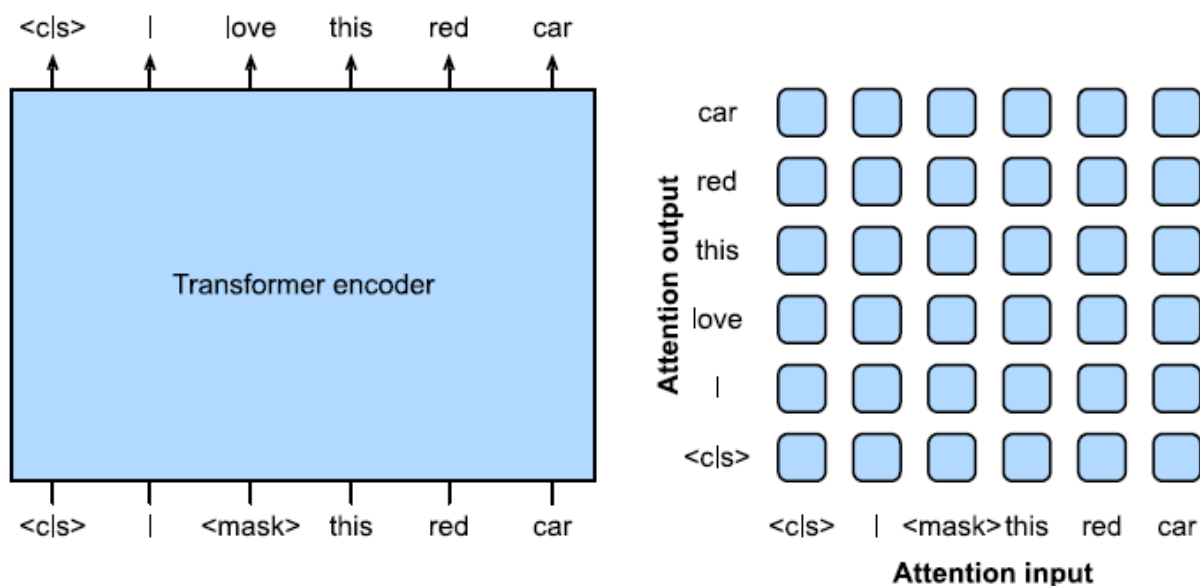
Self supervised pre-training

- Autoencoding models consist only of an encoder and typically predict the masked word from the every preceding and following words in the sentence, therefore it is bi-directional. The model has the knowledge of the entire context.



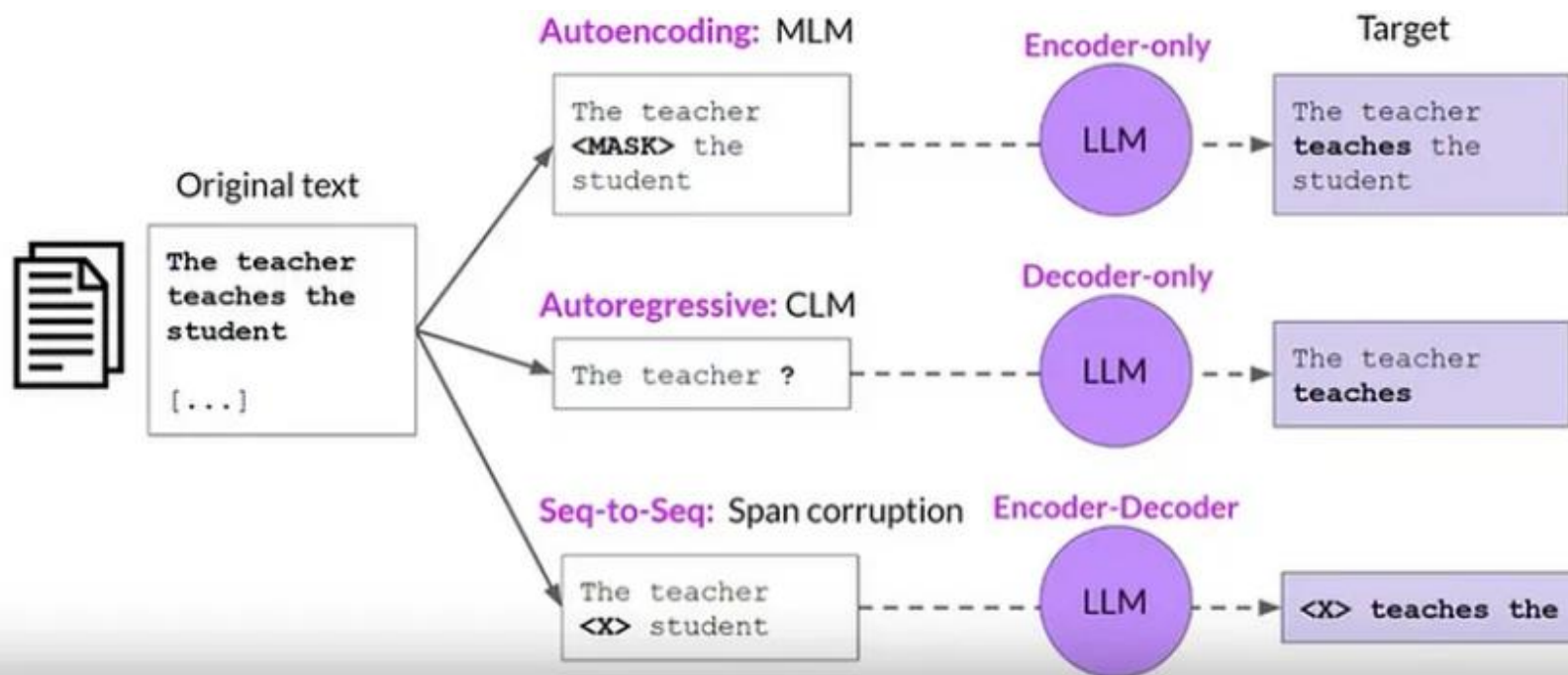
Masked Language Modeling

- Input text with randomly masked tokens is fed into a Transformer encoder to predict the masked tokens.
- As illustrated in the figure, an original text sequence "I", "love", "this", "red", "car" is prepended with the "<cls>" token, and the "<mask>" token randomly replaces "love"; then the cross-entropy loss between the masked token "love" and its prediction is to be minimized during pre-training.



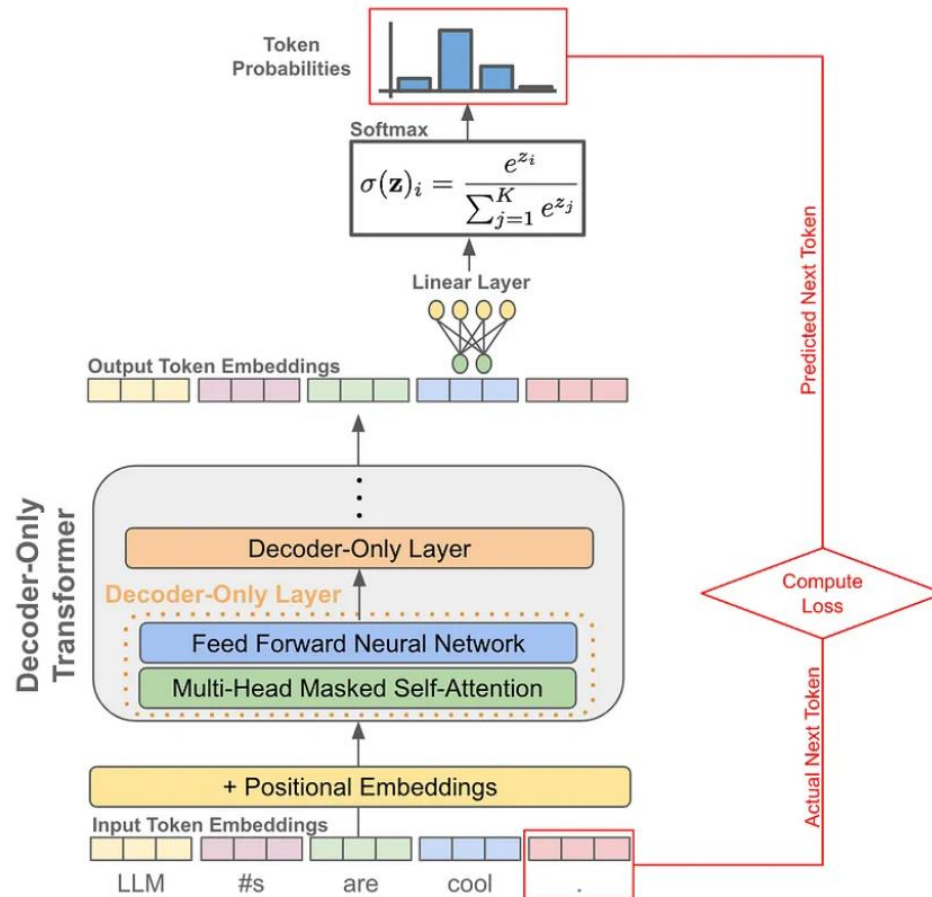
Self supervised pre-training

- Autoregressive models consist only of a decoder and predict the masked word from the preceding words. Thus, autoregressive models are great at auto-completing a sentence, which is what happens in text generation models.



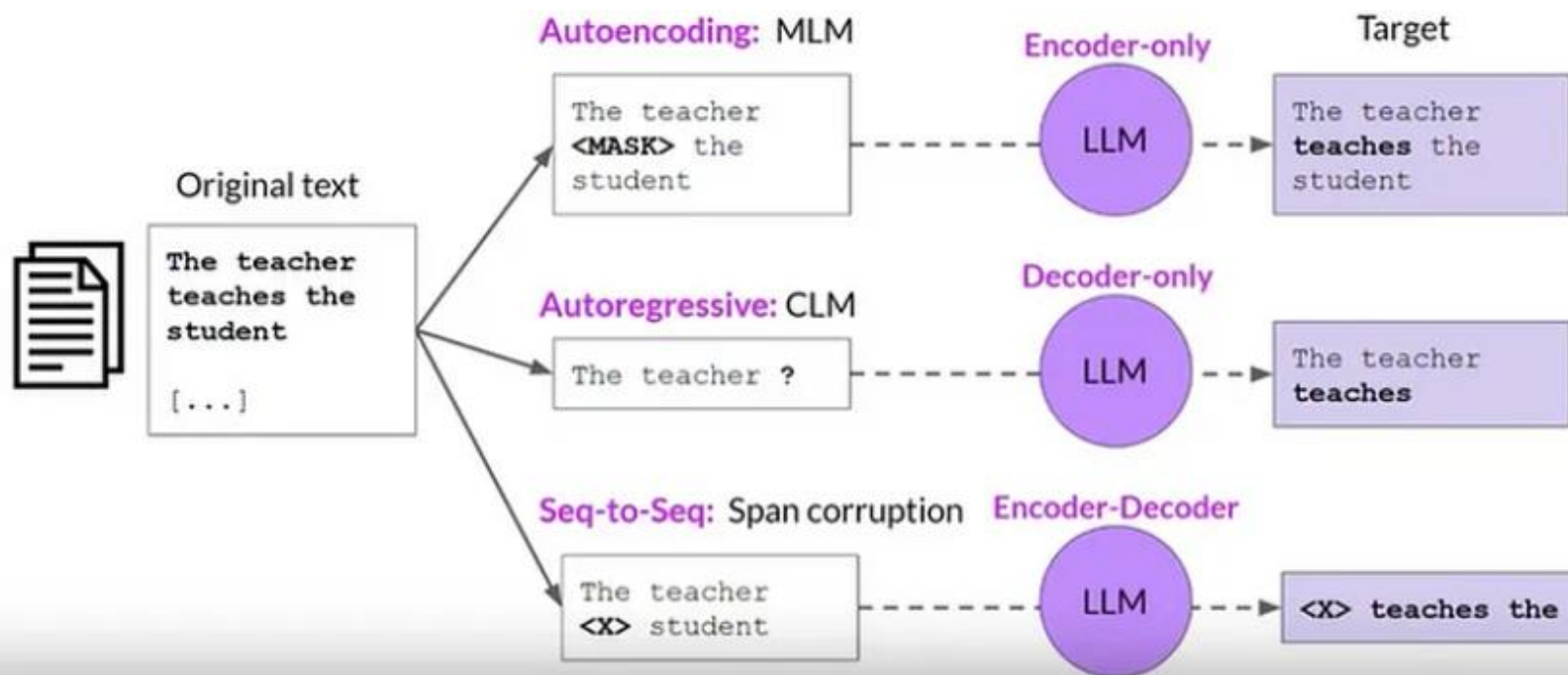
Next Token Prediction

- Any text can be used for this pre-training task, which only requires the prediction of the next word in the sequence.



Self supervised pre-training

- In training seq2seq models, random sequences of input are masked and replaced with a unique token sentinel. The output is the sentinel token followed by the predicted tokens. In summary, seq2seq models both need to understand the context and generate a text.



Span corruption

- In the original text, some words are dropped out with a unique sentinel token. Words are dropped out independently uniformly at random. The model is trained to predict basically sentinel tokens to delineate the dropped out text.

Original text

Thank you ~~for~~ ~~inviting~~ me to your party ~~last~~ week.

Inputs

Thank you <X> me to your party <Y> week.

Targets

<X> for inviting <Y> last <Z>

Summary on pre-training

- Pre-training tasks can be invented flexibly and effective representations can be derived from a flexible regime of pre-training tasks.
- Different NLP tasks seem to be highly transferable, producing effective representations that form a general model which can serve as the backbone for many specialized models.
- With Self-Supervised Learning the models seem to be able to learn from generating the language itself, rather than from any specific task.
- Language Model can be used as a Knowledge Base, namely a generatively pretrained model may have a decent zero-shot performance on a range of NLP tasks.



Datasets and data pre- processing

Datasets

- Training LLMs require vast amounts of text data, and the quality of this data significantly impacts LLM performance.
- Pre-training on large-scale corpora provides LLMs with a fundamental understanding of language and some generative capability.
- Pre-training data sources are diverse, commonly incorporating web text, conversational data, and books as general pre-training corpora.
- Leveraging diverse sources of text data for LLM training can significantly enhance the model's generalization capabilities.

Datasets

Corpora	Type	Links
BookCorpus [65]	Books	https://github.com/soskek/bookcorpus
Gutenberg [66]	Books	https://www.gutenberg.org
Books1 [8]	Books	Not open source yet
Books2 [8]	Books	Not open source yet
CommonCrawl [67]	CommonCrawl	https://commoncrawl.org
C4 [68]	CommonCrawl	https://www.tensorflow.org/datasets/catalog/c4
CC-Stories [69]	CommonCrawl	Not open source yet
CC-News [70]	CommonCrawl	https://commoncrawl.org/blog/news-dataset-available
RealNews [71]	CommonCrawl	https://github.com/rowanz/grover/tree/master/realnews
RefinedWeb [72]	CommonCrawl	https://huggingface.co/datasets/tiiuae/falcon-refinedweb
WebText	Reddit Link	Not open source yet
OpenWebText [73]	Reddit Link	https://skylion007.github.io/OpenWebTextCorpus/
PushShift.io [74]	Reddit Link	https://pushshift.io/
Wikipedia [75]	Wikipedia	https://dumps.wikimedia.org/zhwiki/latest/

Datasets - Books

- Two commonly utilized books datasets for LLMs training are BookCorpus and Gutenberg.
- These datasets include a wide range of literary genres, including novels, essays, poetry, history, science, philosophy, and more.
- Widely employed by numerous LLMs, these datasets contribute to the models' pre-training by exposing them to a diverse array of textual genres and subject matter, fostering a more comprehensive understanding of language across various domains.
- Book Corpus includes 800 million words.

Datasets - CommonCrawl

- CommonCrawl manages an accessible repository of web crawl data, freely available for utilization by individuals and organizations.
- This repository encompasses a vast collection of data, comprising over 250 billion web pages accumulated over a span of 16 years.
- This continuously expanding corpus is a dynamic resource, with an addition of 3–5 billion new web pages each month.
- However, due to the presence of a substantial amount of low-quality data in web archives, preprocessing is essential when working with CommonCrawl data.

Datasets - Wikipedia

- Wikipedia, the free and open online encyclopedia project, hosts a vast repository of high-quality encyclopedic content spanning a wide array of topics.
- The English version of Wikipedia, including 2,500 million words, is extensively utilized in the training of many LLMs, serving as a valuable resource for language understanding and generation tasks.
- Additionally, Wikipedia is available in multiple languages, providing diverse language versions that can be leveraged for training in multilingual environments.

Datasets used in popular LLMs

LLMs	Datasets
GPT-3 [8]	CommonCrawl [67], WebText2 [8], Books1 [8], Books2 [8], Wikipedia [75]
LLaMA [9]	CommonCrawl [67], C4 [68], Wikipedia [75], Github, Books, Arxiv, StackExchange
PaLM [36]	Social Media, Webpages, Books, Github, Wikipedia, News (total 780B tokens)
T5 [68]	C4 [68], WebText, Wikipedia, RealNews
CodeGen [81]	the Pile, BIGQUERY, BIGPYTHON
CodeGeeX [82]	CodeParrot, the Pile, Github
GLM [37]	BooksCorpus, Wikipedia
BLOOM [38]	ROOTS
OPT [83]	BookCorpus, CCNews, CC-Stories, the Pile, Pushshift.io

Data pre-processing

- Once an adequate corpus of data is collected, the subsequent step is data preprocessing, whose quality directly impacts the model's performance and security.
- The specific preprocessing steps involve filtering low-quality text, including eliminating toxic and biased content to ensure the model aligns with human ethical standards.
- It also includes deduplication, removing duplicates in the training set, and excluding redundant content in the test set to maintain the sample distribution balance.
- Privacy scrubbing is applied to ensure the model's security, preventing information leakage or other privacy-related concerns.

Quality filtering

- Filtering low-quality data is typically done using heuristic-based methods or classifier-based methods.
- Heuristic methods involve employing manually defined rules to eliminate low-quality data. For instance, rules could be set to retain only text containing digits, discard sentences composed entirely of uppercase letters, and remove files with a symbol and word ratio exceeding 0.1, and so forth.
- Classifier-based methods involve training a classifier on a high-quality dataset to filter out low-quality datasets.

Deduplication


- Language models may sometimes repetitively generate the same content during text generation, potentially due to a high degree of repetition in the training data.
- Extensive repetition can lead to training instability, resulting in a decline in the performance of LLMs.
- Additionally, it is crucial to consider avoiding dataset contamination by removing duplicated data present in both the training and test set.

Privacy scrubbing

- LLMs, as text-generating models, are trained on diverse datasets, which may pose privacy concerns and the risk of inadvertent information disclosure.
- It is imperative to address privacy concerns by systematically removing any sensitive information. This involves employing techniques such as anonymization, redaction, or tokenization to eliminate personally identifiable details, geolocation, and confidential data.
- By carefully scrubbing the dataset of such sensitive content, researchers and developers can ensure that the language models trained on these datasets uphold privacy standards and mitigate the risk of unintentional disclosure of private information.

Filtering out toxic and biased text

- In the preprocessing steps of language datasets, a critical consideration is the removal of toxic and biased content to ensure the development of fair and unbiased language models.
- This involves implementing robust content moderation techniques, such as employing sentiment analysis, hate speech detection, and bias identification algorithms.
- By leveraging these tools, it is possible to systematically identify and filter out text that may perpetuate harmful stereotypes, offensive language, or biased viewpoints.



Using LLMs after pre-training

Using LLMs after pre-training

Fine-tuning

- Gradient descent on weights to optimize performance on *one task*.
- What to fine-tune?
 - Full network
 - Readout heads
 - Adapters

} Parameter efficient
fine-tuning

Change the model “itself”

Prompting

- Design special prompt to cue / condition the network into specific mode to solve any tasks.
- No parameter change. one model to rule them all.

Change the way to use it.

Natural Language Processing and Large Language Models

Corso di Laurea Magistrale in Ingegneria Informatica



Lesson 11

From Transformers to LLMs

Nicola Capuano and Antonio Greco

DIEM – University of Salerno

