HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Large Language Models and Generative AI for NLP

Dr. Aarne Talman

Machine Learning & Data Science Lead, EMEA Centre for Advanced AI at Accenture
Visiting Researcher and Lecturer at University of Helsinki

Contents

Contents

Teachers

Aarne Talman, Accenture
Weeks 1, 2, 4 & 7

Dmitry Kan, TomTom
Weeks 5 & 6

Jussi Karlgren, Silo AI
Week 3

Course description

This hands-on course explores Large Language Models (LLMs) and their applications in Natural Language Processing (NLP)

You will learn:

- What are LLMs and what is Generative AI?
- How LLMs work?
- How to fine-tune them for specific tasks?
- How to use them for various NLP applications?

Syllabus

Week 1: Introduction to Generative AI and Large Language Models (LLM)

- Introduction to Large Language Models (LLMs) and their architecture
- Overview of Generative AI and its applications in NLP
- Hands-on lab: Getting started with LLMs using popular libraries

Week 2: Using LLMs and Prompting-based approaches

- Understanding prompt engineering and its importance in working with LLMs
- Exploring different prompting techniques for various NLP tasks
- Hands-on lab: Experimenting with different prompts and evaluating their effectiveness

Week 3: Evaluating LLMs

- Understanding the challenges and metrics involved in evaluating LLMs
- Exploring different evaluation frameworks and benchmarks
- Hands-on lab: Evaluating LLMs using different metrics and benchmarks

Week 4: Fine-tuning LLMs

- Understanding the concept of fine-tuning and its benefits
- Exploring different fine-tuning techniques and strategies
- Hands-on lab: Fine-tuning an LLM for a specific NLP task

Week 5: Retrieval Augmented Generation (RAG)

- Understanding the concept of RAG and its advantages
- Exploring different RAG architectures and techniques
- Hands-on lab: Implementing a RAG system for a specific NLP task

Week 6: Use cases and applications of LLMs

- Exploring various real-world applications of LLMs in NLP
- Discussing the potential impact of LLMs on different industries
- Hands-on lab: TBD

Week 7: Final report preparation

- Students work on their final reports, showcasing their understanding of the labs and the concepts learned.

Prerequisites

- Python coding experience
- Basics of machine Learning
    - e.g. Machine Learning for Linguists (LDA-T317, KIK-LG210)

Weekly Schedule

- Lectures:
  - Time: Tuesday at 2:15 PM – 3:45 PM
  - Location: Metsätalo, B308 (sali 12), Unioninkatu 40 (Metsätalo)

- Labs:
  - Time: Thursday at 2:15 PM – 3:45 PM
  - Location: Kielikeskus, sh.405, Language Centre, Fabianinkatu 26

Evaluation

- Students will need to submit **a final report** that covers all the labs:
  - What was done in each lab?
  - What was the motivation behind your solutions?
  - What did you learn?
  - Challenges you encountered?
- You will also be required to submit **a link to your code in Github** that covers all the labs
- Final report submission deadline
  - 31st December 2024

Contents

Outline

- What are language models?
- Popular language modeling algorithms
- The Transformer Architecture
- Core Components of the Transformer Architecture
- Variants of the Transformer Architecture
- Decoder Only LLMs (“Generative models”)
- Recent developments: Mixture of Experts (MoE) Models
- Recent developments: Multimodal models

What are language models?

- **Language modeling** is the task of **estimating the probability distribution of sequences of words or tokens in a given language**.
- More formally, given a sequence of words $w_1 w_2 \dots w_n$, a language model assigns a probability $P(w_1 w_2 \dots w_n)$ to the entire sequence.
- This can also be expressed as the **conditional probability of the next word given the previous words**:

$$P(w_n \mid w_1 w_2 \dots w_{n-1})$$

- In essence, **language modeling aims to capture the statistical patterns and structures of a language**, allowing it **to predict the likelihood of a given sequence of words** or to **generate new text** that is coherent and contextually relevant.

Popular language modeling algorithms (1/5)

N-gram models

- Core Idea: Predict the next word based on the frequencies of sequences of n previous words (n-grams) in the training corpus.
- Key Reference:
  - Shannon, C. E. (1948). A mathematical theory of communication. Bell System Technical Journal

Hidden Markov Models (HMMs)

- Core Idea: Model language as a sequence of hidden states with probabilistic transitions between them. Each state emits words with certain probabilities.
- Key Reference:
  - Baum, L. E., & Petrie, T. (1966). Statistical inference for probabilistic functions of finite state Markov chains. The Annals of Mathematical Statistics

Popular language modeling algorithms (2/5)

Neural Language Models (NLMs)

- Core Idea: Use feed-forward neural networks to directly predict the probability distribution of the next word given the previous context.
- Key Reference:
    - Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A neural probabilistic language model. Journal of Machine Learning Research

Word Embeddings (e.g., Word2Vec, GloVe)

- Core Idea: Represent words as dense vectors in a continuous space, capturing semantic relationships between words.
- Key References:
    - Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space.

Popular language modeling algorithms (3/5)

(Feed-forward) Neural Language Models (NLMs)

- Core Idea: Use feed-forward neural networks to directly predict the probability distribution of the next word given the previous context.
- Key Reference:
  - Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A neural probabilistic language model.

Word Embeddings (e.g., Word2Vec, GloVe)

- Core Idea: Represent words as dense vectors in a continuous space, capturing semantic relationships between words.
- Key References:
  - Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space.
  - Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation.

Popular language modeling algorithms (4/5)

Recurrent Neural Networks (RNNs) with Language Modeling objective

- Core Idea: Use sequential processing to capture dependencies between words in a sentence, allowing for variable-length input and potentially capturing long-range context.
- Key Reference:
  - Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory

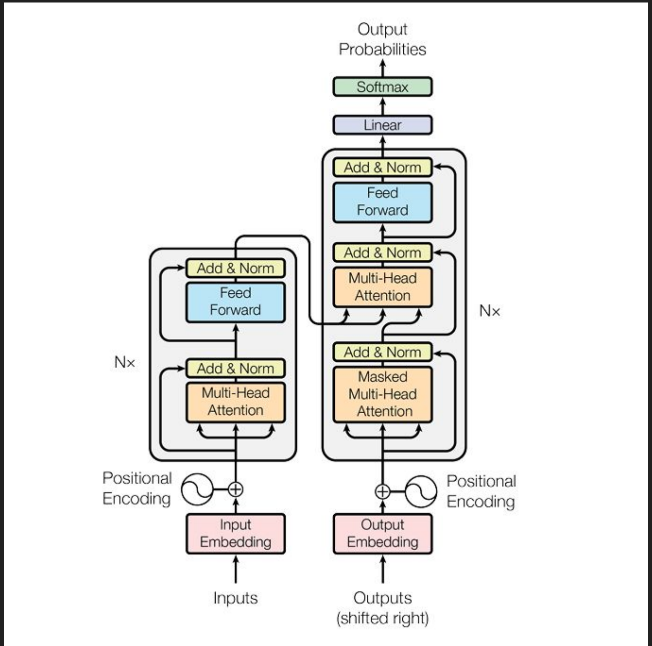
Variational Autoencoders (VAEs) for Text Generation

- Core Idea: Learn a latent representation of text that captures the underlying meaning and style. Generate new text by sampling from this latent space.
- Key Reference:
  - Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., & Bengio, S. (2015). Generating sentences from a continuous space.

Popular language modeling algorithms (5/5)

Transformer models

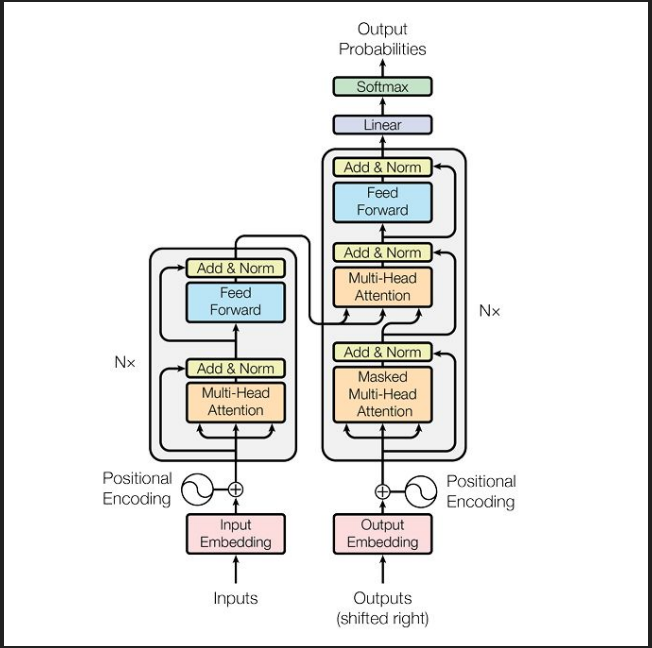
- Core Idea: Utilize self-attention mechanisms to weigh the importance of different words in a sentence, enabling the capture of long-range dependencies and contextual relationships effectively.
- Originally designed to be a machine translation model
- Key Reference:
  - Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need



Source: Vaswani et al. 2017

Transformer Architecture

- **Captures long-range dependencies:** The self-attention mechanism allows the model to effectively capture relationships between words that are far apart in the sequence, overcoming limitations of previous models like RNNs.
- **Parallel processing:** The Transformer architecture is highly parallelizable, enabling faster training and inference on modern hardware.
- **Strong performance on various NLP tasks:** Transformers have achieved state-of-the-art results on a wide range of NLP tasks, including machine translation, text summarization, question answering, and many others.



Source: Vaswani et al. 2017

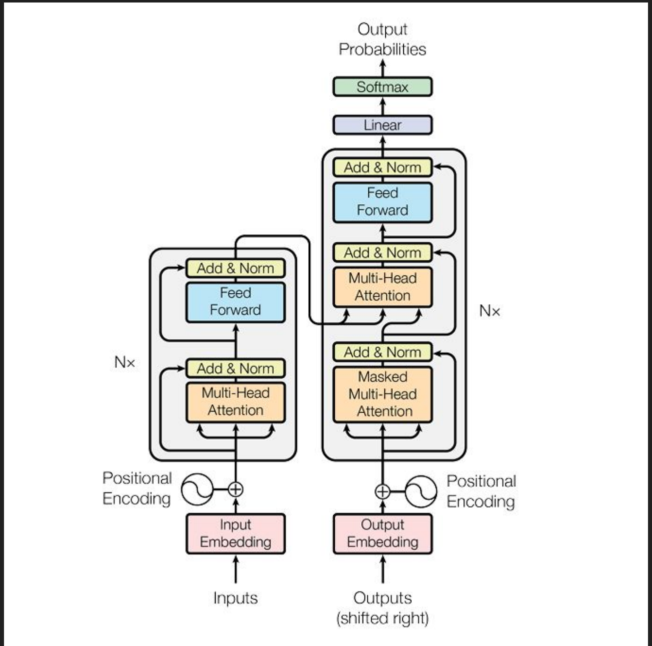
Core Components of the Transformer Architecture (1/3)

Self-Attention Mechanism:

- The heart of the Transformer, enabling it to **weigh the importance of different words in a sequence** when making predictions.
- **Each word in the input sequence attends to every other word**, calculating attention scores that represent the relative importance of each word for understanding the current word.
- This allows the model to **capture long-range dependencies** and contextual relationships effectively.

Multi-Head Attention:

- Employs **multiple attention heads in parallel**, each focusing on different aspects of the input sequence.
- This allows the model to learn a richer and more nuanced representation of the input.



Source: Vaswani et al. 2017

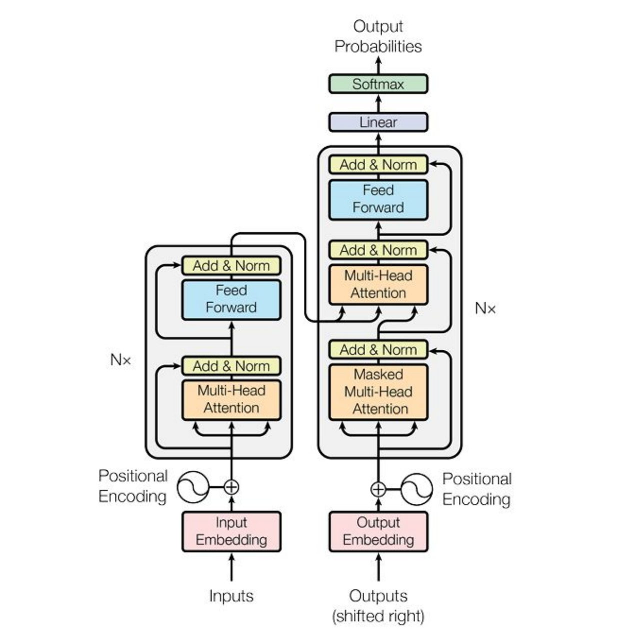
Core Components of the Transformer Architecture (2/3)

Positional Encoding:

- Since **self-attention is permutation-invariant**, it needs a way to incorporate word order information.
- Positional encodings are added to the input embeddings, providing the model with **information about the position of each word in the sequence**.

Encoder and Decoder:

- The Transformer typically consists of **an encode**r and a **decoder stack**, each **composed of multiple layers**.
- The e**ncoder processes the input sequence**, generating a contextualized representation for each word.
- The **decoder generates the output sequence**, attending to both the encoder output and its own previous outputs.



Source: Vaswani et al. 2017

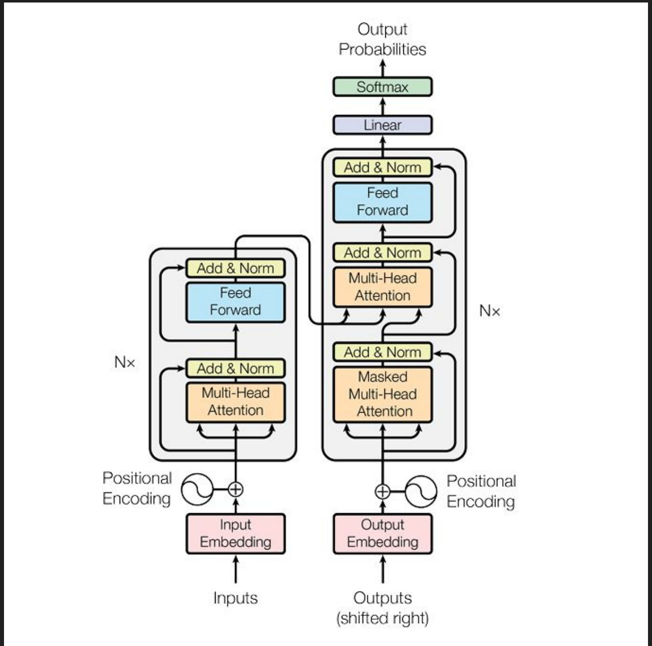
Core Components of the Transformer Architecture (3/3)

Feed-Forward Networks:

- Each layer in the encoder and decoder includes a **feed-forward network that applies non-linear transformations to the input**, further enhancing the model's representational power.

Layer Normalization and Residual Connections:

- **Layer normalization helps stabilize** training and improve the model's robustness.
- **Residual connections prevent vanishing gradients** by bypassing components of the network and allowing gradients to flow more easily during training, enabling the training of deeper networks.

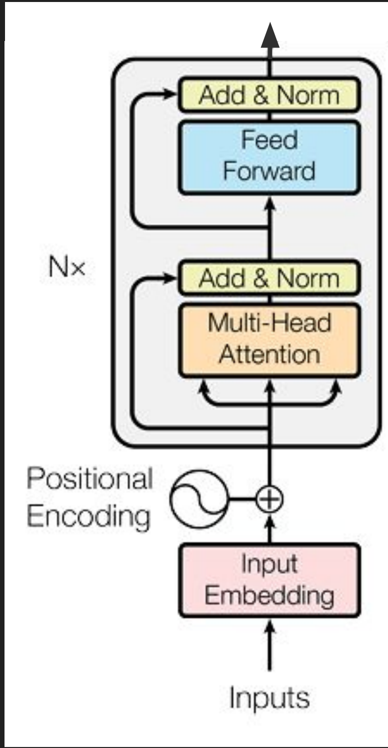


Source: Vaswani et al. 2017

Variants of the Transformer Architecture

Encoder-only Transformers ("Embedding models"):

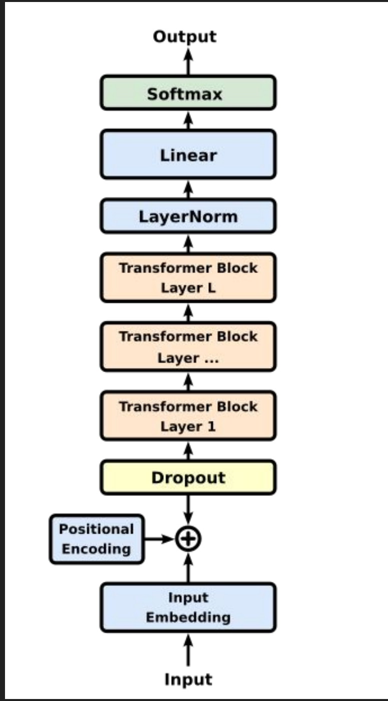
- Structure: **Only utilize the encoder stack** of the original Transformer architecture.
- Focus: Primarily designed **for** understanding and **representing the input text, capturing contextual relationships between words**.
- Common Applications:
  - Text classification: E.g. determining the sentiment or topic of a given text.
  - Named entity recognition: Identifying and classifying entities in text (e.g., names of people, organizations, locations).
- Examples: **BERT**, RoBERTa



Variants of the Transformer Architecture

Decoder-only Transformers ("Generative models"):

- Structure: Only utilize the decoder stack of the original Transformer architecture.
- Focus: Specialize in generating text, predicting one word at a time based on the previous context.
- Common Applications:
  - Text generation: Generating creative writing, completing sentences, or writing code.
  - Text summarization: Condensing a longer text into a shorter, informative summary.
- Examples: GPT family (GPT, GPT-2, GPT-3), LLaMa family

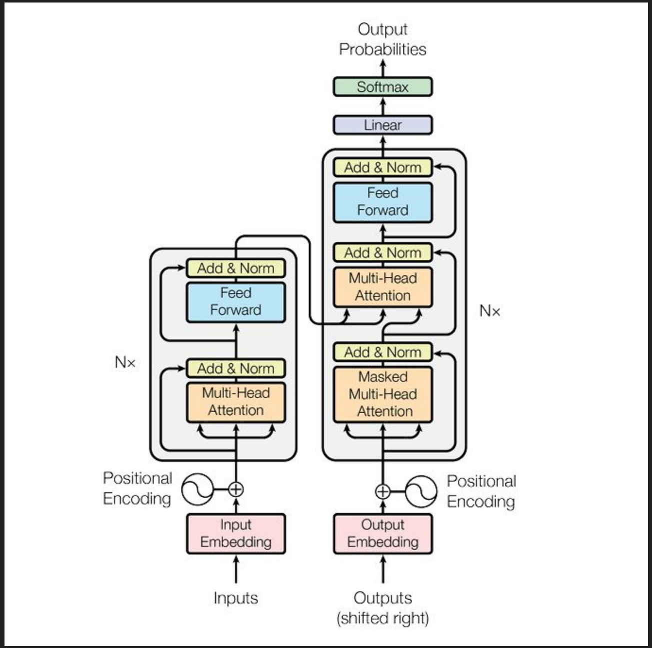


GPT-1

Variants of the Transformer Architecture

Encoder-Decoder Transformers:

- Structure: Utilize both the encoder and decoder stacks of the original Transformer.
- Focus: Combine the capabilities of understanding the input text (encoder) and generating output text (decoder).
- Common Applications:
    - **Machine translation:** More powerful for translating between languages with different structures.
    - **Dialogue systems:** Generating responses in a conversation based on the context of the dialogue.
    - **Summarization:** Generating summaries that are both informative and abstractive.
- Examples: T5, BART, Neural Machine Translation (NMT) models



Source: Vaswani et al. 2017

Decoder Only LLMs ("Generative LLMs")

GPT (Generative Pre-trained Transformer) (2018)

- Standard decoder-only Transformer with masked self-attention.
- Pre-trained on a large corpus of text data using the language modeling objective.
- Fine-tuned on specific tasks using supervised learning.

GPT-2 (2019)

- Scaled-up version of GPT with more layers and parameters.
- Demonstrated the power of larger models for language generation.

GPT-3 (2020)

- Massively scaled-up version of GPT with 175 billion parameters.
- Showcased the potential of few-shot learning, where the model can perform new tasks with just a few examples.

Decoder Only LLMs (“Generative LLMs”)

LLaMA (Large Language Model Meta AI) (2023)

- Focuses on efficiency and performance, achieving comparable results to much larger models with fewer parameters.
- Employs various optimizations and architectural choices to improve training and inference efficiency.

GPT-4 (2023)

- Details of the architecture are not fully disclosed, but it is expected to be a further scaled-up and improved version of the GPT-3 architecture.
- Demonstrates advancements in multimodality, handling both text and image inputs.

Mistral (2023)

- Mistral aims to achieve the best possible performance with the fewest parameters, making it more accessible and less computationally expensive to run than larger models.
- Open-weight and open source. Many of Mistral's models are open-weight, meaning the weights are publicly available, and some are even open source.

Recent developments: Mixture of Experts (MoE) Models

- Core Idea: A type of model architecture where **multiple "expert" networks specialize in different subtasks** or domains. A "gating" network dynamically routes inputs to the most relevant experts for processing.
- Key Differences from Standard Decoder-Only Models:
  - **Sparsity:** MoE models activate only a subset of experts for each input token, leading to more efficient computation and parameter usage compared to dense models that activate all parameters for every token.
  - **Specialization:** Experts can specialize in different linguistic patterns, topics, or styles, enabling the model to handle a wider range of tasks and generate more diverse and nuanced outputs.
  - **Adaptability: T**he gating network learns to dynamically route inputs based on their content, allowing the model to adapt to different contexts and tasks.
- Examples:
  - **GPT-4**: While not a fully MoE model, GPT-4 reportedly incorporates a Mixture of Experts layer in its architecture, potentially contributing to its improved capabilities.
  - **Mixtral**: A sparse mixture-of-experts model developed by Mistral AI, aiming for efficiency and strong performance across various tasks.

Recent developments: Multimodal models

- **Core Idea:** A type of language model that **can process and integrate information from multiple modalities**, such as text, images, audio, and video.
- Key Differences from Standard Decoder-Only Models:
  - **Multimodal Input Processing:** These models incorporate specialized encoders for each modality (e.g., vision encoders for images, audio encoders for sound) to extract meaningful representations. These representations are then fused and integrated with textual information.
  - **Cross-Modal Understanding:** Multimodal LLMs can reason across different modalities, for example, by answering questions about images, generating image captions, or even translating between modalities.
  - **Enhanced Contextual Awareness:** By combining information from multiple sources, these models gain a richer understanding of context, leading to more accurate and nuanced responses.
- Examples:
  - **GPT-4V:** An extension of GPT-4 with vision capabilities, allowing it to analyze images, answer questions about visual content, and generate detailed descriptions.
  - **Flamingo:** A visual language model from DeepMind that can connect information from images and text to perform tasks like visual question answering and image captioning.