Language Modeling

Introduction to NLP — MSc. DH EdC-PSL

Noé Durandard (noe.durandard@psl.eu)

October 29, 2025

0.1 Before we start

0.1.1 Class Overview and Objectives

Introduction to modern NLP models and methods.

1. Language Modeling
    - Foundations, n-grams, Transformer, Pre-trained Language Models
2. Discovering Structure
    - Semantic Spaces, Retrieval, Topic Modeling
3. Inferring Patterns
    - Fine-tuning, Inference, Classification
4. Generative LLMs*
    - Prompting, Automating, TBD

0.1.2 Material

Contact: noe.durandard@psl.eu

GitHub Repository: d-noe/NLP_DH_PSL_Fall2025

Hands-On: Google Colab, Local Machines, Binder, …

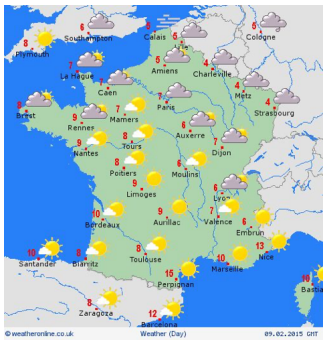
Table of Contents

1 Introduction to Language Modeling

1.1 Foundations and Objectives

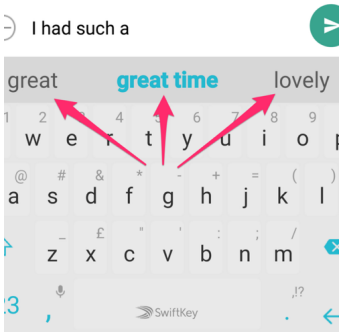
1.1.1 Modeling

- What do you expect from a **model**?




```
- simulate behaviour of the real world
- understand which events are in better agreement
with the world
- predict the next event given a description of
"context" (current state)
```

- What about **language**?




```
- Same!
- But what are these *events*?
```

1.1.2 Modeling Langague

The *linguistic events* used in Language Models are **linguistic unit**: text, sentence, word, token, character, …

> ⓘ **Ambiguous Definitions?**
>
> In the context of this lecture, you can consider these *atomic units*, or *tokens*, as words, but keep in mind the most appropriate unit can depend on the application, and that:
>
> - the choice of this 'unit' may depend on the application.
> - definitions of units (e.g. words) are ambiguous and also context-dependent.
> - /!\ always be sure all your NLP modules use same notion of *tokens* /!\

1.1.3 Language Modeling Problem

Language Model

A Language Model (LM) estimates the probability of pieces of text.

Given a sequence of text $w_1, w_2, \cdots, w_S$, it answers the question:

What is $P(w_1, w_2, \cdots, w_S)$?

1.1.4 LMs play the role of …

- … a judge of **gramaticality**
  - e.g. *“The player runs.”* vs. *“The player run.”*
- … a judge of **semantic plausability**
  - e.g. *“The teacher spoke.”* vs. *“The blackboard spoke.”*
- … an enforcer of **stylistic consistency**
  - e.g. *“In conclusion, the results substantiate the proposed hypothesis.”* vs. *“In conclusion, the results totally back up the proposed hypothesis.”*
- … a repository of **knowledge (?)**
  - e.g. *“Zinedine Zidane played for Real Madrid.”*
  - /!\ Very difficult to guarantee!

1.2 Language Models in practice

1.2.1 LMs are everywhere

- Web Search Engines
- Translation Service
- Autocomplete
- Autocorrect
- Of course: Chat bots
- …

1.2.2 Common NLP Tasks (source: 🤗)

- Analysing words
  - POS-tagging, Named Entity Recognition, **Word Sense Disambiguation**, …
- Analysing sentences (to documents)
  - Sentiment classification, **Topic Modeling**, Natural Language Inference …
- Retrieving Information
  - Extracting information from document, **Ranking similar documents**, …
- Generating text content
  - **Completing Prompt, Filling blanks**, …
- Sequence to sequence generation
  - Machine Translation, Text Summarization, …

1.2.3 DH Applications

- Analyze style, authorship, ideology across corpora.
- Track semantic change over time.
- Support historical text restoration or OCR correction.
- Build metadata enrichment pipelines.
- Use models as *tools for exploration*, rather than oracles.

… but also the other way around: **use DH to study LMs**.

1.2.4 How to get there?

Recall:

> **Language Model**
>
> A Language Model (LM) estimates the probability of pieces of text.

How to compute: $P(w_1, w_2, \cdots, w_S)$?

- Gain knowledge from corpora.
- **Different formalisms to compute P**

What is the most probable piece of text?

1. *I like Digital Humanities*
2. *Humanities like I Digital*
3. *I likes Digital Humanities*
4. *I bike Digital Humanities*

→ quite intuitive, but how are machines supposed to understand it?

2 From n-grams to Neural LMs

2.1 n-gram models

2.1.1 Reminder: Probabilities

Conditional Probabilities:

$$P(B|A) = P(A,B)/P(A) \;\Leftrightarrow\; P(A,B) = P(A)P(B|A)$$

The Chain Rule in general:

$$\begin{aligned} P(x_1, x_2, x_3, \cdots x_S) &= P(x_1)P(x_2|x_1)P(x_3|x_1, x_2) \\ &\quad \cdots P(x_S|x_1, x_2, x_3, \cdots x_{S-1}) \\ &= \prod_{i=1}^{S} P(x_i|x_1 \cdots x_{i-1}) \end{aligned}$$

2.1.2 $(n-1)$ Markov Assumption

State *only* depends on $(n-1)$ preceeding states:

$$P(x_i | x_1 \cdots x_{i-1}) \approx P(x_i | x_{i-n+1} \cdots x_{i-1})$$

$$\Rightarrow P(x_1, x_2, x_3, \cdots x_S)$$

$$\approx \prod_{i=1}^{S} P(x_i | x_{i-n+1} \cdots x_{i-1})$$

$\rightarrow$ reduces "context" and allows simple (count-based) computation of n-gram probabilities.

The parameters of the model are:
$P(x_i \cdots x_{i+n-1})$, which can be estimated on some corpus.

2.1.3 Unigram

$P(w_1, w_2, \cdots, w_S) = \prod_i P(w_i) = P(w_1)P(w_2) \cdots P(w_S)$

→ What is the most probable sequence?

$P(\text{the a the a the a})$

$= P(\text{the})P(\text{a})P(\text{the})P(\text{a})P(\text{the})P(\text{a})$

$P(\text{the cat sat on the mat})$

$= P(\text{the})P(\text{cat})P(\text{sat})P(\text{on})P(\text{the})P(\text{mat})$

2.1.4 Bigram

$P(w_1, w_2, \cdots, w_S) = \prod_i P(w_i|w_{i-1}) = P(w_1)P(w_2|w_1) \cdots P($

→ What is the most probable sequence?

$P(\text{the a the a the a})$

$= P(\text{the})P(\text{a}|\text{the})P(\text{the}|\text{a})P(\text{a}|\text{the})P(\text{the}|\text{a})P(\text{a}|\text{the})$

$P(\text{the cat sat on the mat})$

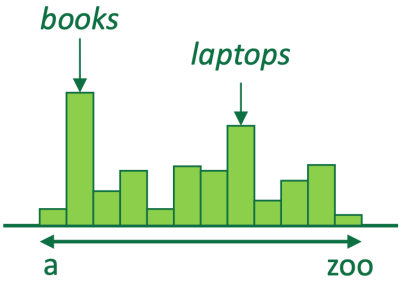
$= P(\text{the})P(\text{cat}|\text{the})P(\text{sat}|\text{cat})P(\text{on}|\text{sat})P(\text{the}|\text{on})P(\text{mat}|\text{the})$

2.1.5 Generation with n-gram

- n-gram model *knows* (/has learned) distribution of sizes ≤ n
  - e.g. unigram: $P(\text{the})$, bigram: + $P(\text{the cat})$
- next-word probability distribution over vocabulary can be inferred: $p(w_{S+1}|w_{S-n+1} \cdots w_S)$

The students opend their [?]

→ 4-gram model:
$p(?|\text{students opened their})$



Sample from distribution:
→ *The students opend their books*

2.1.6 Simple but limited paradigm

Why n-grams?

- Simple and fast.
- Effective for short frequent sequences.
- Clear paradigm introducing important issues for (L)LMs.
  - Training, evaluation, sampling, etc. (we'll come back to this)

> **Note**
>
> All modern neural NLP techniques actually focus on n-grams, estimating various kinds of related probabilities.

Limitations:

- Data sparsity.
  - Scales badly with n.
- Rigid context window.
  - can't handle long dependencies.
- No sense of meaning.

2.2 Neural Language Models

2.2.1 Timeline

100+ years of LMs in 30s:

- **1906:** Markov – Statistical modeling of sequences
- **1948:** Shannon – Information theory and early ideas of statistical prediction of text
- **1980s–1990s:** Back-off and smoothing, probabilistic grammars
- **2003:** *Neural Probabilistic Language Model* (Bengio et al. 2003)
  - First to learn **distributed representations (embeddings)** jointly with next-word prediction
- **2010s:** Recurrent Neural Networks (RNNs), LSTMs, GRUs
  - Allowed *dynamic* context windows
- **2017:** *Attention is All You Need* (Vaswani 2017)
  - self-attention mechansim, drastically reduces training costs
- **2018—:** Pre-trained LLMs + scaling
  - BERT (Devlin et al. 2019)
  - GPT (Radford et al. 2018)

2.2.2 Different paradigm

Language Modeling problem

How to compute $P(w_1, w_2, \cdots, w_S)$ *or* $P(w_i|w_1 \cdots w_{i-1})$*?*

Count-based models: rely on explicit co-occurrence counts.

$$P(w_i|w_1 \cdots w_{i-1}) \approx \frac{\mathrm{C}(w_{i-n+1} \cdots w_{i-1} w_i)}{\mathrm{C}(w_{i-n+1} \cdots w_{i-1} w_i)}$$

Neural models: learn a function f_Θ that models NL

$$P(w_i|w_1 \cdots w_{i-1}) = f_\Theta(w_1, w_2, \cdots, w_{i-1}$$

$\rightarrow$ + based on continuous vector embedding (make semantic emerge).

2.2.3 Neural Probabilistic LM (Bengio et al. 2003)

$$P(w_t|w_{t-n+1:t-1}) = \text{softmax}(g_\theta(e(w_{t-n+1}), \ldots, e(w_{t-1})))$$

- Core ideas:
  - Represent each word as an **embedding vector**.
  - Concatenate embeddings of previous $(n-1)$ words.
  - Feed to a feedforward neural network to predict next word probability.

✅ Captures semantic similarity
🚫 Fixed-size context → still limited like n-gram

2.2.4 Recurrent Neural Networks (RNNs)

Motivation

- Fixed-size context of feedforward models is too restrictive.
- Need a model that can handle *arbitrary-length* context.

RNN idea:

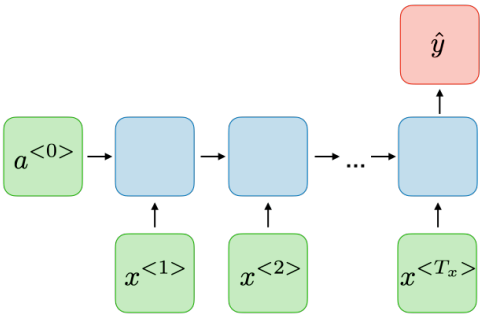
- Maintain a **hidden state** summarizing all past words.
- Update recurrently:

$$h_t = f(W_{xh}x_t + W_{hh}h_{t-1})$$

$$P(w_t|w_{1:t-1}) = \text{softmax}(W_{ho}h_t)$$

2.2.4.1 RNN Architecture

- The same weights are used at each time step → efficient and general.
- Capable to capture long-range dependencies.



RNN Architecture (from (Amidi and Amidi 2018)).

> **Note**
>
> Many variants have been developed, e.g. **Long Short-Term Memory (LSTM)** to control information flow (Hochreiter and Schmidhuber 1997) (+later bi-LSTM, context from both sides), Gated Recurent Units (GRUs) simplified approach (Cho et al. 2014). Both mitigate the vanishing gradients arising in traditional RNNs and allow longer dependencies.

2.2.4.2 Limitations of RNNs

- Difficult to capture very long dependencies
  - even with LSTMs.
- Memory bottleneck
  - all context must fit into one hidden vector.
- Slow to train and not prone to parallelization
  - Sequential, recurrent, nature.

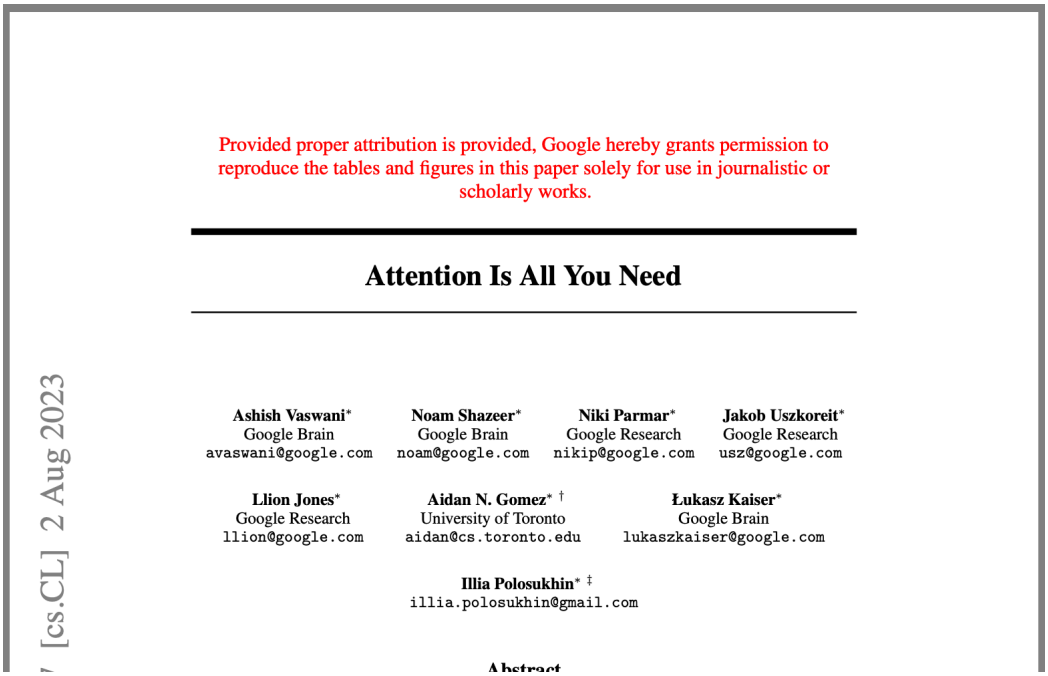
→ These challenges motivated **attention mechanisms** and the **Transformer** architecture.

3 Transformer

3.1 Motivation

- Need to use more information
  - Take advantage of full context
  - whole sentences available during training
  - why go through sentence word by word?
- Need to improve efficiency
  - Allows parallelization.
  - Based on matrices products.
  - Much faster to train than predecessors.
- Transformers (with variations) have became standard models for (any) LM tasks.

3.2 *Attention is All You Need* (Vaswani 2017)



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

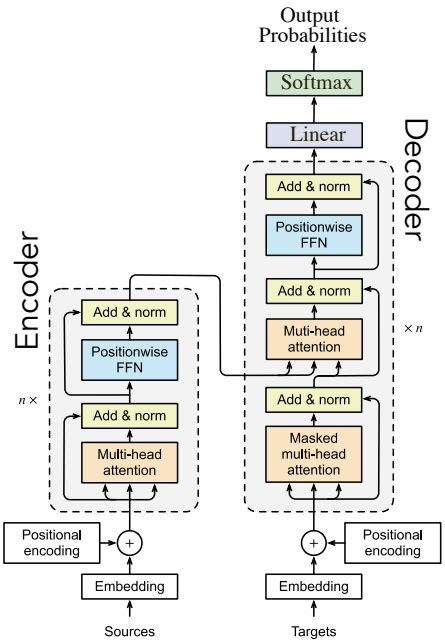
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

3.3 The Transformer Architecture

3.3.1 Overview

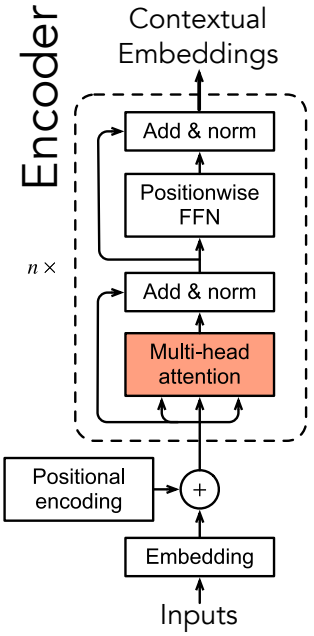
- Encoder / Decoder Model
  - Building blocks are ≈ similar
- Originally applied to machine translation
  - Foundational for many modern approaches
  - not limited to natural language! (vision, proteins, audio, …)
- Layered approach
  - → model sequentially builds intricate understanding.



Transformer Architecture, based on (Vaswani 2017), reworked from (Zhang et al. 2023).

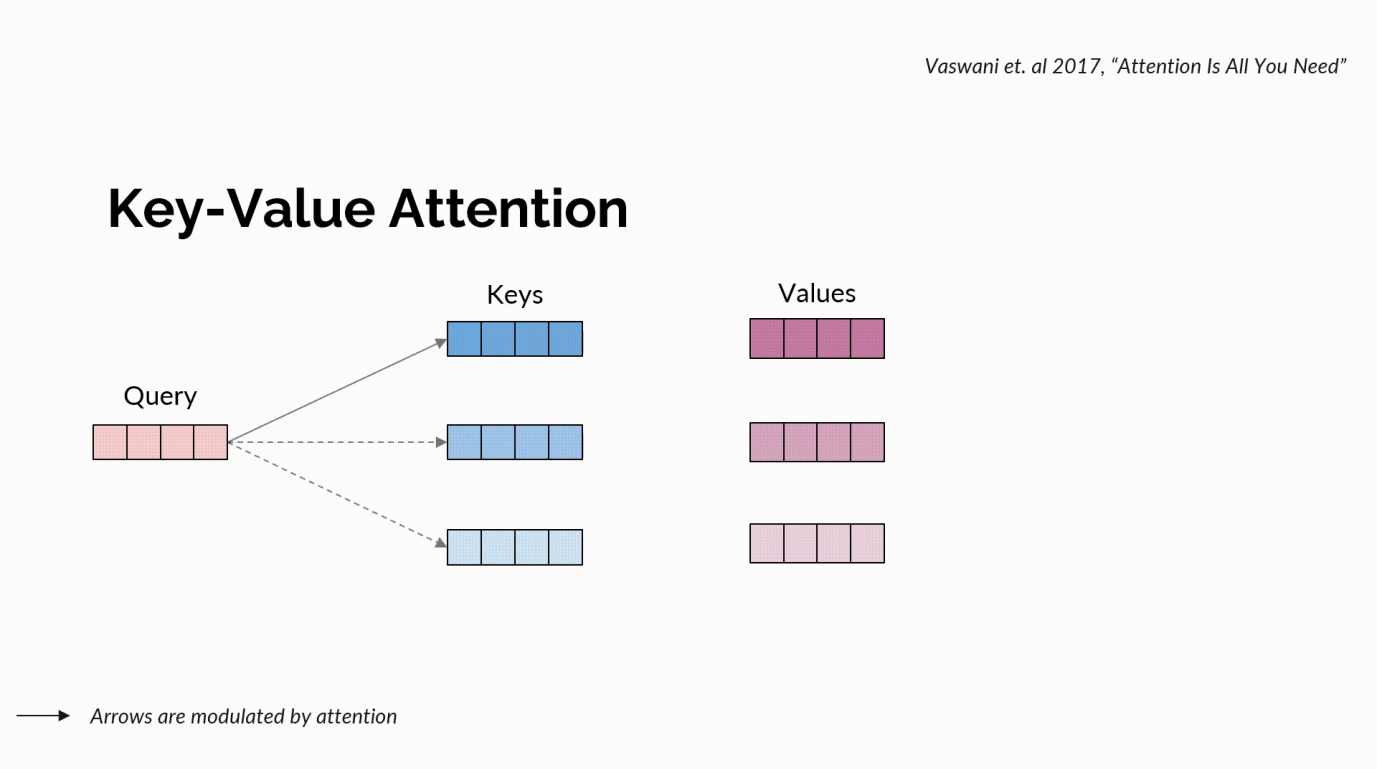
3.3.2 Self-attention Mechanism

- Capture relationships among tokens in a sequence.
- Split between Multi-Head Attention.



Transformer Architecture, based on (Vaswani 2017), reworked from (Zhang et al. 2023).

3.3.2.1 Intuition: Information Look-up



Extracted from (Mittal 2024).

3.3.2.2 Under the hood: Matrices Multiplications

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where, for a given input $X \in \mathbb{R}^{N \times d}$ and *learnable* projection matrices $W^Q \in \mathbb{R}^{d \times d_k}$, $W^K \in \mathbb{R}^{d \times d_k}$, and $W^V \in \mathbb{R}^{d \times d_v}$:

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V$$

→ can be parallelized on GPUs!

3.3.2.3 Query-Key-Value

- Q: asking information.
- K: presenting information.

→ QK^T : *attention weights* from one word to another.

- V: giving information.

→ information flow scaled by *attention weights*.

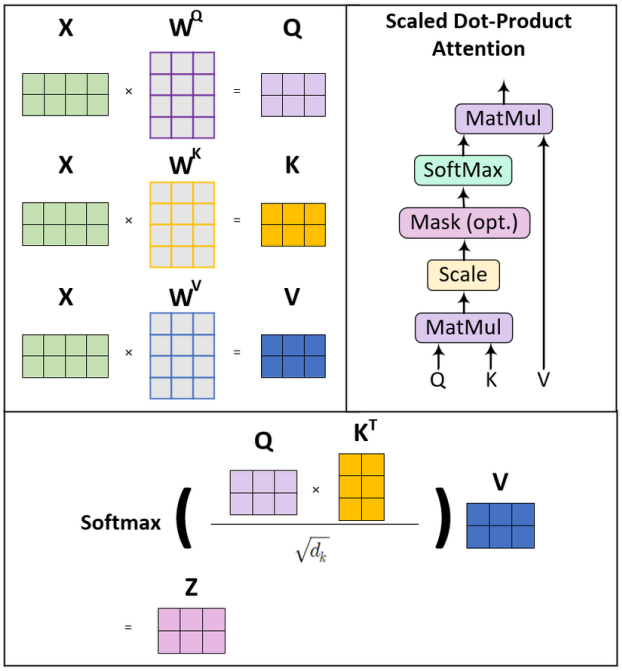
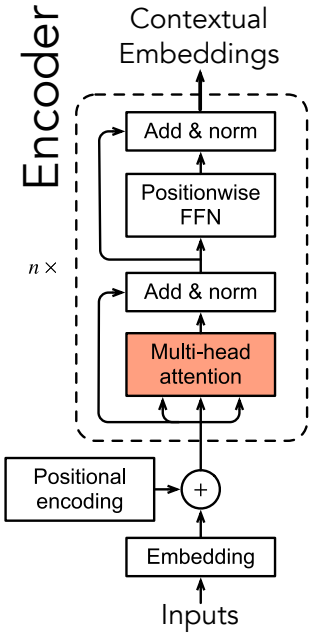


Diagram of the query, key, value, and self-attention mechanism (from (Hwang, Jeong, and Hwang 2025)).

3.3.3 Multi-Head Attention

- **Intuition:** independent "heads" can learn to capture different linguistic relationships.
  - e.g. syntactic, semantic, short-/long- range dependencies, …
- **Implementation:** several attention mechanisms then concatenated.

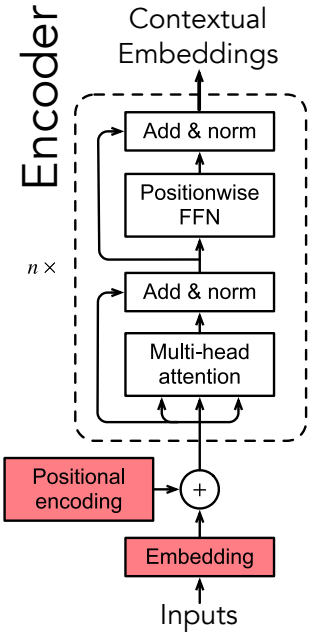
Think of: "The cat chased the mouse in the garden."



Transformer Architecture, based on (Vaswani 2017), reworked from (Zhang et al. 2023).

3.3.4 Input: Embedding + Positional encoding

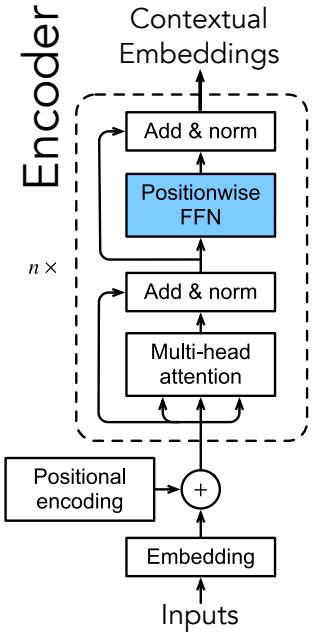
1. Tokenization
   - Breaking text into smaller units (from vocabulary).
2. Token Embedding
   - Associate token with dense vectors (ID → vector).
3. Positional Encoding
   - Encode information about token's positions.
   - 💡: No direct notion of word order in attention.
4. Combine: Semantic + Order



Transformer Architecture, based on (Vaswani 2017), reworked from (Zhang et al. 2023).

3.3.5 Feed Forward Network

- **Intuition:** self-attention captures relationships between tokens, FFN refines token-wise representations.
- **Implementation:** two linear layers with ReLU in-between:
  $$\text{FFN}(x) = max\,(0, xW_1 + b_1)\,W_2 + b_2$$
  - Adds non-linearity.
  - Enrichment through expansion-compression.

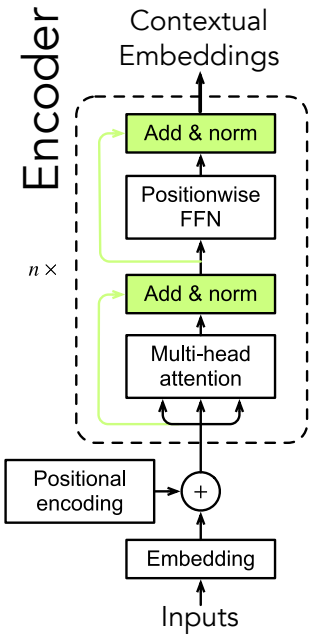


Transformer Architecture, based on (Vaswani 2017), reworked from (Zhang et al. 2023).

3.3.6 Residual Connections & Normalization

More technicalities: facilitate information flow and convergence.

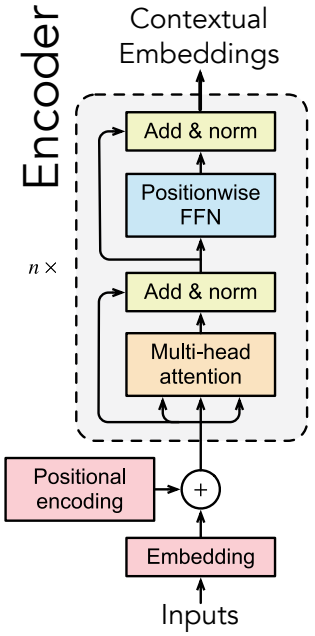
- **Residual Connections**
  - Ease gradient flow.
  - Allow stacking lots of layers.
- **Layer Normalization**
  - Control “flow” to next layer.
  - Improves convergence stability.



Transformer Architecture, based on (Vaswani 2017), reworked from (Zhang et al. 2023).

3.3.7 Summary

- Positional Encoding:
  - retains information of the word order.
  - enables parallelization and efficient model training.
- Self-Attention Mechanism:
  - global look-up in a sequence.
  - allows direct dependencies.
  - fully parallel.
- Multi-Head Attention:
  - jointly attend to information from different representation subspaces at different positions.
  - provide several perspectives.
  - enhances training stability.
  - effectiveness and optimality discussed in recent literature ((Liu, Liu, and Han 2021), (Mittal et al. 2022)).

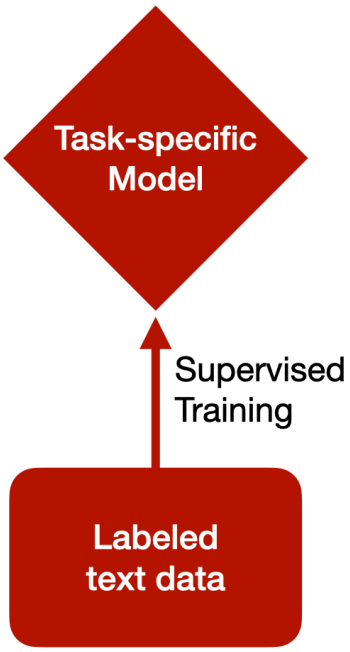


Transformer Architecture, based on (Vaswani 2017), reworked from (Zhang et al. 2023).

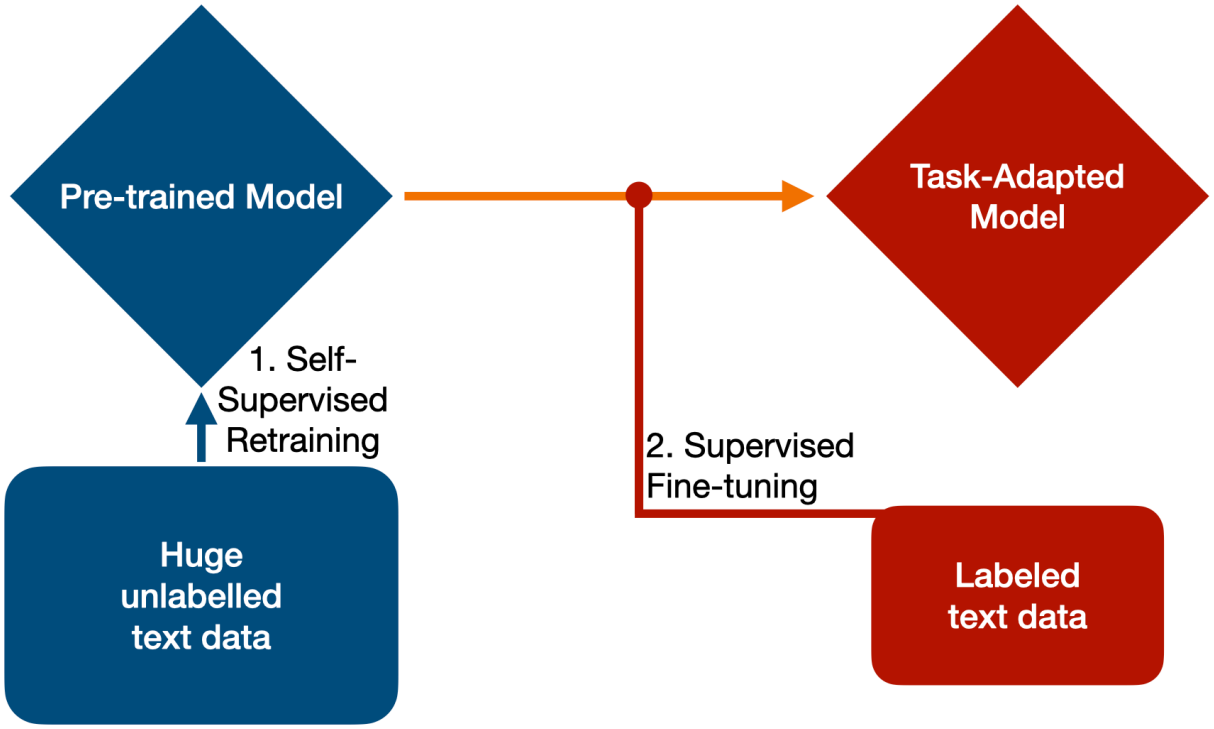
4 Pre-trained Transformer Models

4.1 NLP Pipeline

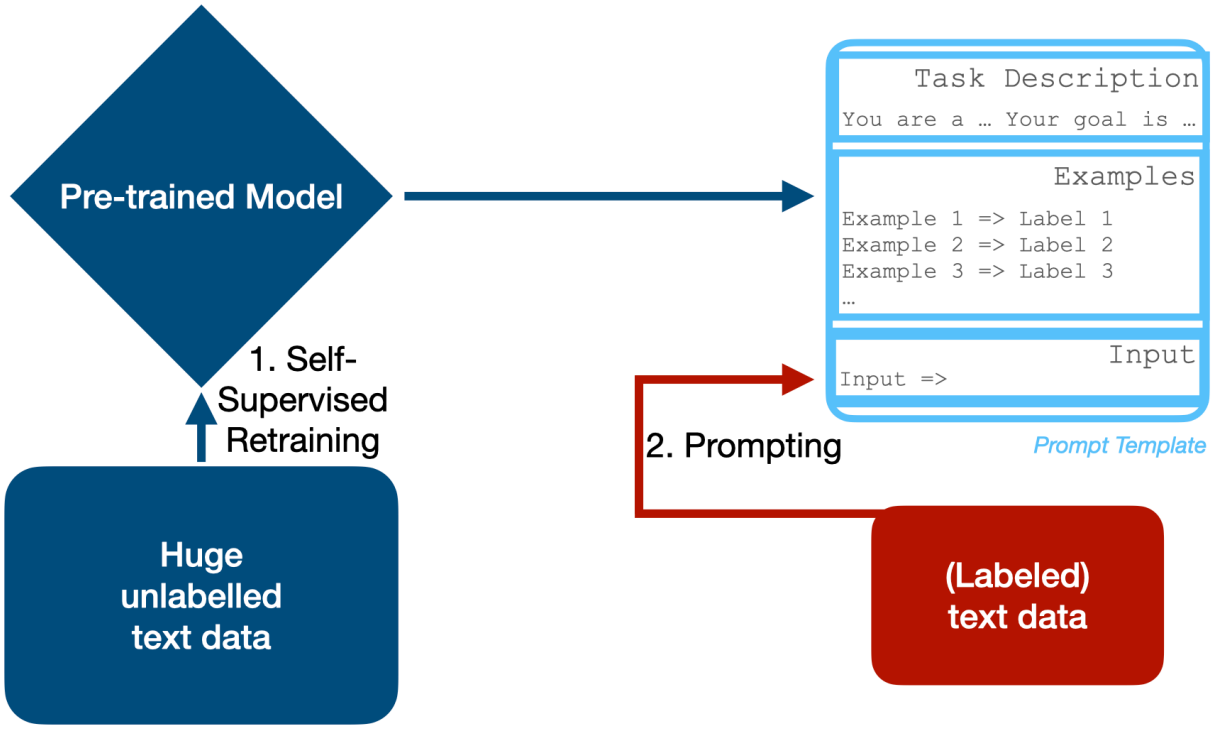
4.1.1 Back in the days



4.1.2 Adapting pre-trained models



4.1.3 Directly prompting



4.2 Types of pre-trained LMs

4.2.1 Variations in architectures and objectives

	Unidirectional language model	Bidirectional language model	Sequence-to-sequence model
Architecture	Transformer decoder	Transformer encoder	Transformer
Pre-training	Language modeling (2)	Mask language modeling (3)	Sequence-to-sequence learning
Tasks	Language generation	Language understanding	Sequence-to-sequence
Models	GPTs[3,25,26]	BERT,[8] RoBERTa,[17] ALBERT,[14] XLNet,[36] Electra[7]	BART,[15] T5[24]

Overview of pre-trained LM types (from (Li 2022)).

4.2.2 Masked Language Modeling

The goal is predict the masked token: *All the [MASK] best*

- Attends left- and right- context.
- Great for tasks that require a good **contextual understanding** of an entire sequence.
- Example: BERT-like models.

4.2.3 Causal Language Modeling

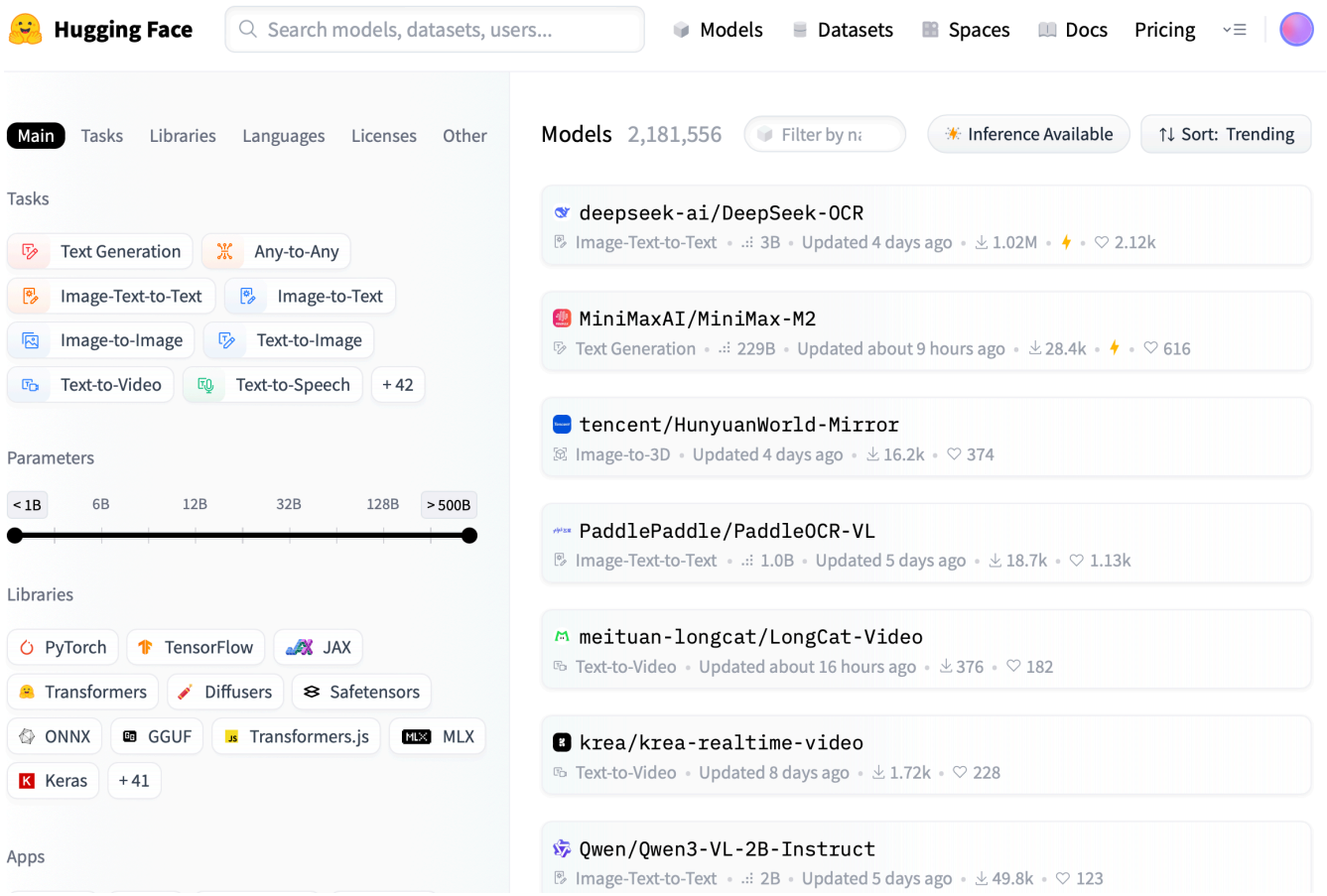
The goal is predict the next token: *All the very …*

- Attends to left-context only.
- Great for natural language **generation** (… and more?).
- Example: GPTs.

→ We'll come back to this later.

4.3 Where to find pre-trained LMs?

You can find, access (and share) open-weights LLMs on 🤗 HuggingFace.



Screenshot of https://huggingface.co.

5 Hands-on

References

Amidi, Afshine, and Shervine Amidi. 2018. “Recurrent Neural Networks Cheatsheet Star.” *CS 230 - Recurrent Neural Networks Cheatsheet*. https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks#.

Bengio, Yoshua, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. “A Neural Probabilistic Language Model.” *Journal of Machine Learning Research* 3 (Feb): 1137–55.

Cho, Kyunghyun, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. “On the Properties of Neural Machine Translation: Encoder-Decoder Approaches.” *arXiv Preprint arXiv:1409.1259*.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. “BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding.” https://arxiv.org/abs/1810.04805.

Geva, Mor, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022. “Transformer Feed-Forward Layers Build Predictions by Promoting Concepts in the Vocabulary Space.” In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, edited by Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, 30–45. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.emnlp-main.3.

Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. “Long Short-Term Memory.” *Neural Computation* 9 (8): 1735–80.

Hwang, Hae-Ji, Seon-Geun Jeong, and won-Joo Hwang. 2025. “Ultrawideband Non-Line-of-Sight Classification Using Transformer-Convolutional Neural Networks.” *IEEE Access* PP (January): 1–1. https://doi.org/10.1109/ACCESS.2025.3568830.

Li, Hang. 2022. “Language Models: Past, Present, and Future.” *Communications of the ACM* 65 (7): 56–63. https://doi.org/10.1145/3490443.

Liu, Liyuan, Jialu Liu, and Jiawei Han. 2021. “Multi-Head or Single-Head? An Empirical Comparison for Transformer Training.” https://arxiv.org/abs/2106.09650.

Mittal, Sarthak. 2024. “Compositional Attention: Disentangling Search and Retrieval.” *Mila*. Mila. https://mila.quebec/en/article/compositional-attention-disentangling-search-and-retrieval.

Mittal, Sarthak, Sharath Chandra Raparthy, Irina Rish, Yoshua Bengio, and Guillaume Lajoie. 2022. “Compositional Attention: Disentangling Search and Retrieval.” https://arxiv.org/abs/2110.09419.

Radford, Alec, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. “Improving Language Understanding by Generative Pre-Training.”

Vaswani, A. 2017. “Attention Is All You Need.” *Advances in Neural Information Processing Systems*. https://dl.acm.org/doi/10.5555/3295222.3295349.

Zhang, Aston, Zachary C. Lipton, Mu Li, and Alexander J. Smola. 2023. *Dive into Deep*