Introduction and Organization

IE686 Large Language Models and Agents

UNIVERSITY OF MANNHEIM
Data and Web Science Group

Hello

- About me:
  - M.Sc. Wi-Inf Ralph Peeters
  - Graduate Research Associate
  - Research Interests
    - Sustainable LLM-Agents
    - Entity Matching using Deep Learning
    - Data Integration
  - Office: B6, 26 - C 1.04
  - eMail: ralph.peeters@uni-mannheim.de

- I will teach the lectures, exercises and coach the student projects.

Hello

- About me:
  - M.Sc. Aaron Steiner
  - Graduate Research Associate
  - Research Interests
    - Entity Matching using LLMs
    - Data Integration
  - Office: B6, 26 - C 1.04
  - eMail: aaron.steiner@uni-mannheim.de

- I will co-teach the exercises and coach the student projects.

Hello

- About me:
  - Prof. Dr. Christian Bizer
  - Professor for Information Systems V
  - Research Interests:
    - Web-based Systems
    - Large-Scale Data Integration
    - Data and Web Mining
  - Room: B6, 26 - B1.15
  - eMail: christian.bizer@uni-mannheim.de

- Contributed to the design of the course and will contribute to the final assessment.

Outline

- **Course Organization**
- What is a Language Model?
- Language Representations
- Common Language Tasks
- The Transformer Architecture
- Pre-trained Language Models

The First Half of the Course

- Lectures
  - Foundational knowledge for understanding LLMs
  - Architectures, training, prompt engineering patterns, fine-tuning, agents, evaluation, …
  - **Goal:** Learn and understand concepts and methods

- Exercises
  - Practical applications of the lecture concepts
  - Introduce tools: LangGraph and AutoGen
  - **Goal:** Learn to apply concepts in practice and prepare you for the group projects

The Second Half of the Course

- Group Projects
  - Work in teams of five students on the practical application of LLMs to a problem
  - **Goal:** Understand a problem domain and apply the learned concepts and tools to solve the problem!
  - Teams write a 12-page report about their project and present their results during a presentation at the end of the semester

- Course Grading
  - **The project is the basis for your grading, there is no exam!**
  - 3 ECTS (70% written project report, 30% presentation of results)

Course Schedule

Day	Topic
13.02	**Lecture:** Introduction to Language Models
20.02	**Lecture:** Instruction Tuning and RLHF
27.02	**Lecture:** Prompt Engineering and Efficient Adaptation
06.03	**Lecture:** LLM Agents and Tool Use
13.03	**Exercise:** Introduction to LangGraph 1
20.03	**Exercise:** Introduction to LangGraph 2
27.03	**Project:** Introduction to Student Projects
03.04	**Exercise:** Introduction to Autogen
10.04	**Project:** Project Coaching
30.04	**Project:** Project Coaching
08.05	**Project:** Project Coaching
15.05	**Project:** Project Coaching
22.05	**Project:** Project Coaching
28.05	**Project:** Presentation of Project Results

Course Organization

- Course Webpage
  - https://www.uni-mannheim.de/dws/teaching/course-details/courses-for-master-candidates/ie-686-large-language-models-and-agents/
  - The lecture slides are published on this webpage

- Time and Location
  - Every Thursday, 15:30 to 17:00
    Room: B6 A101
  - Starting 13.02.2025
  - **Important:** Slots on 30.04 and 28.05 are a Wednesday instead!
    Room and Time TBA

Course Organization

- Course ILIAS
  - https://ilias.uni-mannheim.de/ilias.php?baseClass=ilrepositorygui&ref_id=159725
  - Need to be accepted for the course to access
- Contains:
  - **Forum**
    - For online communication/discussion during the course
    - Not just with me, but also amongst each other!
    - Don’t be shy to post questions and bringing your unique knowledge to discussions, so we can all learn from each other!
  - **Literature Links**
    - Relevant textbooks/papers
    - Further reading not covered in the course

Literature and Credits

UNIVERSITY OF MANNHEIM
Data and Web Science Group

- Some supporting literature for the course
  - Daniel Jurafsky & James H. Martin: Speech and Language Processing. (3rd edition)
  - Zhao et al.: A Survey of Large Language Models. 2024. arXiv:2303.18223
  - Wang et al.: A Survey on Large Language Model based Autonomous Agents. 2024. Frontiers of Computer Science.
  - Zhou et al.: A Comprehensive Survey on Pretrained Foundation Models: A History from BERT to ChatGPT. 2024. International Journal of Machine Learning and Cybernetics.
- The slide set of this lecture builds on slides from:
  - Jiaxin Huang
  - Mrinmaya Sachan
  - Danqi Chen
  - Daniel Jurafsky & James H. Martin
  - Many thanks to all of you!

Tools

- LangChain
  - Library for easy interaction with various hosted and local LLMs
  - Many tools for prompt and embedding orchestration
- AutoGen/LangGraph
  - Libraries for orchestrating agentic workflows
  - You are free to use whatever fits your project use-case better

Project API and Hardware Usage

- bwUniCluster 2.0
  - All students can register
  - Provide compute servers with modern GPUs (up to 8 per machine)
  - Uses a job queuing system for distributing resources
  - **Good and free option** for locally hosting open-source LLMs
- APIs for hosted LLMs
  - **We cannot reimburse you** for API costs, e.g. OpenAI, Anthropic, …
  - You may consider using cheap but powerful models like GPT4o-mini
  - Check for free tier offers from different providers (e.g. Groq API)
  - I may have more tips for you once we reach the exercises…
- Google Colab
  - Could be useful for **prototyping** with small open-source models

Outline

- Course Organization
- **What is a Language Model?**
- Language Representations
- Common Language Tasks
- The Transformer Architecture
- Pre-trained Language Models

What is a Language Model?

- The classic definition of a language model (LM) is a probability distribution over each possible token sequence $[w_1,w_2,\ldots,w_3]$, independent of it making any sense:
  – Sally fed my cat with meat: P(Sally, fed, my, cat, with, meat) = 0.03
  – My cat fed Sally with meat: P(My, cat, fed, Sally, with, meat) = 0.005
  – fed cat meat my my with: P(fed, cat, meat, my, my, with) = 0.0001

- A **good** language model ideally assigns a high probability to sequences that make sense given …
  – the structure of the actual language (English in this case)
  – any additional context in which a sentence is uttered, if available

Our Focus: Autoregressive LMs

- A type of language model based on the chain rule of probability:

  $P(w_1,w_2,w_3,\ldots,w_n) = P(w_1)*p(w_2|w_1)*p(w_3|w_1,w_2)*\ldots*p(w_n|w_1,w_2,\ldots,w_{n-1})$

- P(Sally, fed, my, cat, with, meat) = P(Sally)
  * P(fed | Sally)
  * P(my | Sally, fed)
  * P(cat | Sally, fed, my)
  * P(with | Sally, fed, my, cat)
  * P(meat | Sally, fed, my, cat, with)

➔ Conditional probability

- We will also see some other types of language models later

Language Generation

- Assume we have a good language model and a given text prompt $w_{[1:n]}$
  - Now we want to generate a good completion for this prompt with some length L
  - How to find $w_{[n+1:n+L]}$ with the highest probability?
  - Enumerate over all possible combinations? 😳

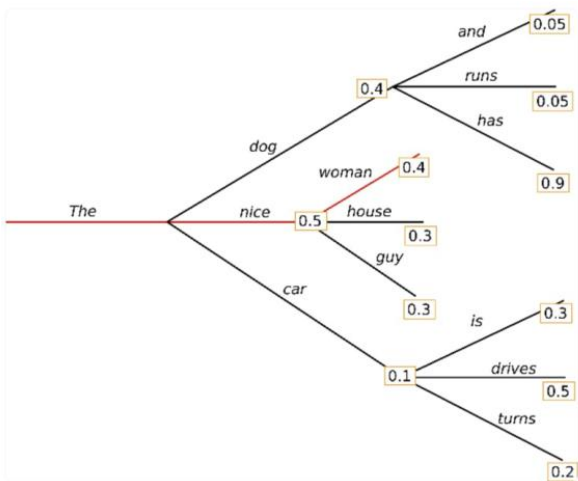
➔ Next token prediction

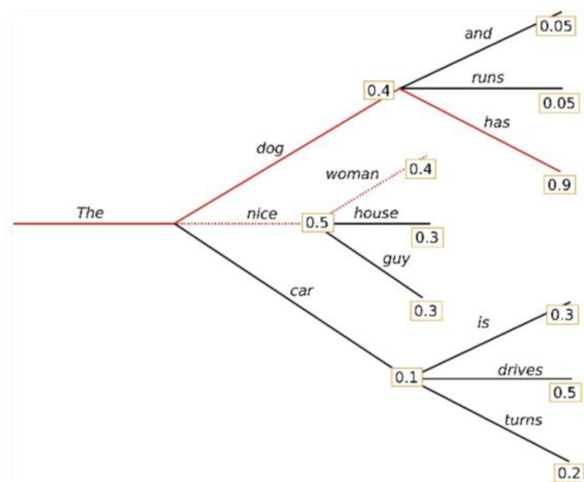
  - Generate tokens step by step starting from w_{n+1} using $p(w_{n+1}|w_{[1:n]})$
  - For selecting the next token with $p(w_{n+1}|w_{[1:n]})$, there are different decoding approaches

Decoding Approaches

- **Greedy decoding:** At each step, always select w_t with the highest $p(w_t / w_{[1:t-1]})$.

- **Beam Search:** Keep track of k possible paths at each step instead of just a single one. Reasonable beam size k: 5-10





Decoding Approaches

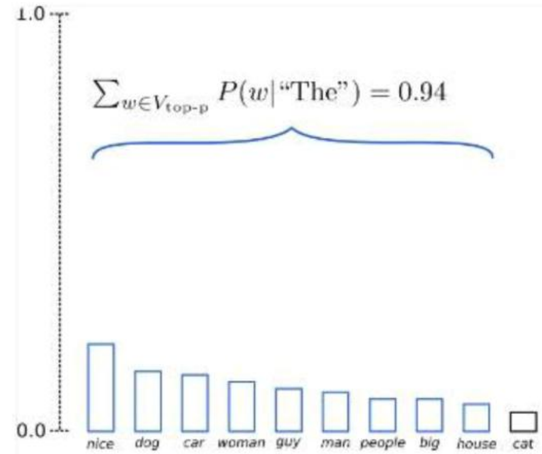
- **Top-k sampling:** At each step, randomly sample the next token from $p(w_t|w_{[1:t-1]})$, but restrict to only the k most probable tokens.

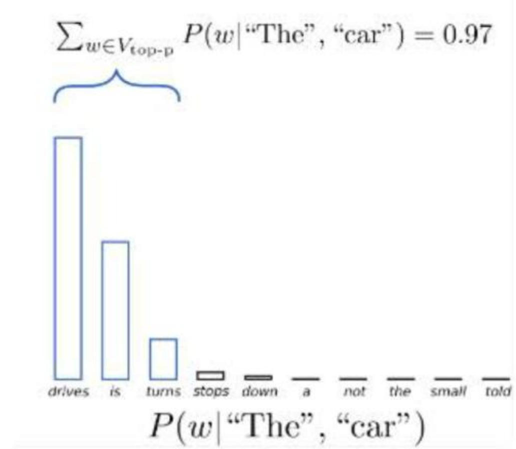
➔ Allows to **control diversity**:
  - Increasing k leads to more creative outputs with an increasing risk of getting bad outputs
  - Decreasing k gives “safer” but less creative outputs
  - Problem: fixed k can cover wildly different amount of probability mass

- **Top-p sampling:** At each step, randomly sample the next token from $p(w_t|w_{[1:t-1]})$, but restrict to the set of tokens with a cumulative probability of p

➔ Throw away long-tail tokens

- Top-k and Top-p can be used together!





Decoding Approaches

- **Temperature sampling**
  - Reshape the distribution instead of truncating it
  - Inspired by thermodynamics
    - System at high temperature is flexible and can explore many possible states
    - System at lower temperature is likely to explore a subset of lower energy (better) states
- Low-temperature sampling ($\tau \leq 1$)
  - increases the probability of the most probable words
  - decreases the probability of the rare words

$$\cancel{\mathbf{y} = \text{softmax}(u)}$$

$$\mathbf{y} = \text{softmax}(u/\tau)$$

Temperature sampling

$$\mathbf{y} = \text{softmax}(u/\tau) \qquad 0 \leq \tau \leq 1$$

- Why does it work?
  - When τ is close to 1 the distribution does not change much
  - The lower τ is, the larger the scores being passed to the softmax
  - Softmax pushes high values toward 1 and low values toward 0
  - Large inputs pushes high-probability words higher and low probability words lower, making the distribution more greedy
  - As τ approaches 0, the probability of the most likely word approaches 1

Sounds great but…

Question: How do we actually train a good language model?

Sounds great but…

Question: How do we actually train a good language model?

Answer: By maximizing the language model probability over an observed large corpus of text.

N-gram Language Models

- For example: Bi-gram language models based on co-occurrence of two words
- $P(w_N|w_{N-1}) = C(w_{N-1},w_N) / C(w_{N-1})$
  - P(to|want) = 608/927 = 65.59%
  - P(spend|want) = 1/927 = 0.11%

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Curse of Dimensionality

- Limitations of N-gram models
  - **Limited context length**: N-grams have a finite context window of length N, which means they cannot capture long-range dependencies or context beyond the previous N-1 words
  - **Sparsity**: As N increases, the number of possible N-grams grows exponentially, leading to sparse data and increased computational demands
    - Suppose the vocabulary size is V, the number of possible N-grams increases to V^N
  - Usually V is larger than ten thousand. Representing each word as a one-hot vector is **inefficient** and **ignores word semantics**
    - “Dogs” and “cats” are more similar than “dogs” and “rectangular”

Outline

- Course Organization
- What is a Language Model?
- **Language Representations**
- Common Language Tasks
- The Transformer Architecture
- Pre-trained Language Models

Language Representations

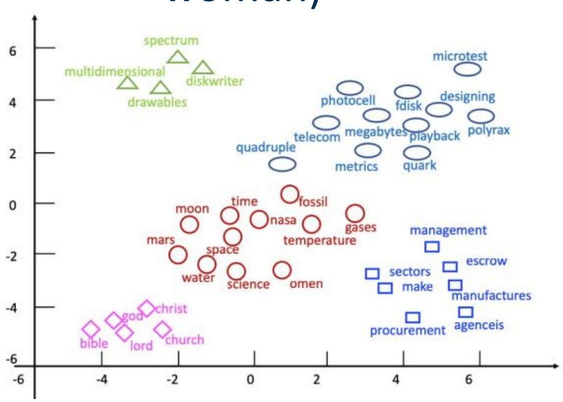
- Ludwig Wittgenstein: "*The meaning of a word is its use in the language*"
  – How to represent words while accounting for their meaning?

1. Words are defined by their **environment** (the words around them)
   – "*If A and B have almost identical environments, we say that they are synonyms*" – Zellig Harris (1954)
2. Words are defined by **different dimensions**
   – Which can be represented as a point in a multi-dimensional space
   – E.g. 3 affective dimensions of words (Osgood et al. 1957)

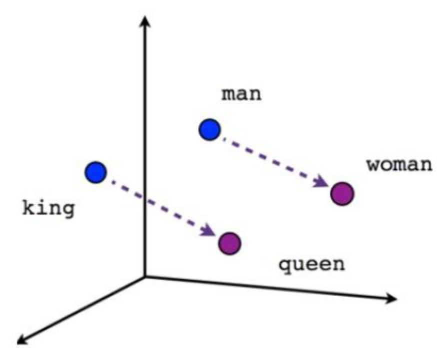
	Word	Score		Word	Score
Valence	love	1.000		toxic	0.008
	happy	1.000		nightmare	0.005
Arousal	elated	0.960		mellow	0.069
	frenzy	0.965		napping	0.046
Dominance	powerful	0.991		weak	0.045
	leadership	0.983		empty	0.081

Distributed Representations

- A milestone in NLP and ML:
  - Unsupervised learning of text representations – no supervision needed
  - Embed previously one-hot vectors into lower dimensional space
  - → **Word embeddings** have fixed dimensions
  - → Addresses sparsity of bag-of-words model (curse of dimensionality)
- Embeddings capture relevant properties of word semantics
  - Word similarity: Words with similar meaning are embedded closer
  - Word analogy: Linear relationships between words (e.g. king - queen = man - woman)



Word Similarity



Word Analogy

Distributed Representations

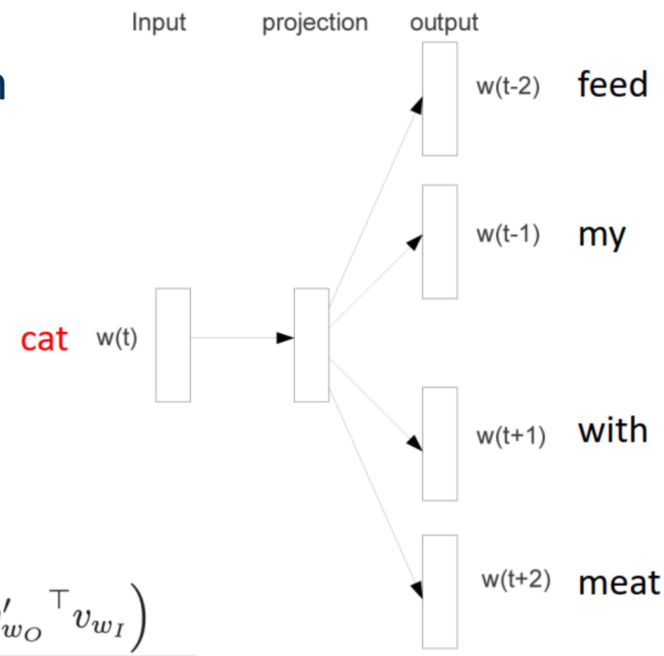
University of Mannheim
Data and Web Science Group

Intuition: Why embeddings?

- With **words**, a feature is a word identity (e.g. early bag-of-words models, see Data Mining lecture videos)
  - For example Feature 5: “terrible” with no information about context
  - Requires exact same word to be in training and test sets
- With **embeddings**:
  - Feature is a vector in a semantic space
  - Feature 5: [35,22,17,…]
  - In the test set there might be a similar vector [34,21,14,…]
  - It is possible to generalize to **similar but unseen** words!

Word2Vec

- Assumption: If two words have similar contexts, then they have similar semantic meanings!
- Training objective: Learn to predict word(s) in nearby context
  - Skip-gram: predict context from center word
  - CBOW: predict center word from context



Co-occurring words in a **local context window**

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(\boxed{w_{t+j}|w_t})$$

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^{\top} v_{w_I}\right)}{\sum_{w=1}^{W}\exp\left({v'_w}^{\top} v_{w_I}\right)}$$

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J., 2013. Distributed Representations of words and Phrases and their Compositionality. *Advances in Neural Information Processing Systems*, *26*.

Extension to Subwords: fastText

- fastText improves on Word2Vec by incorporating subword information into word embeddings

Tri-gram extraction

<where> ⇒ <wh, whe, her, ere, re>

- Words are represented by aggregating their n-gram embeddings

➔ Allows to also embed words unseen during training

Word2Vec probability expression

$$p(w_O|w_I) = \frac{\exp\left(\boxed{{v'_{w_O}}^{\top} v_{w_I}}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^{\top} v_{w_I}\right)} \dashrightarrow \sum_{g \in \mathcal{G}_w} \boxed{\mathbf{z}_g^{\top}} \mathbf{v}_c .$$

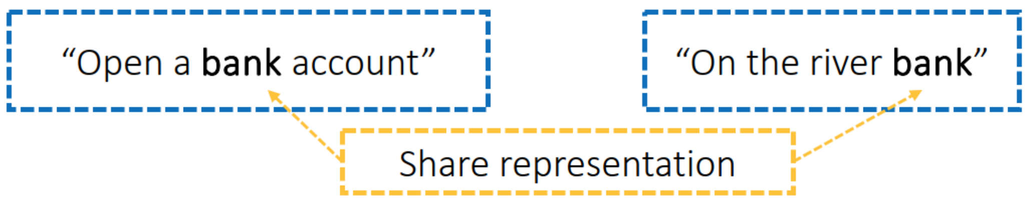
Represent a word by the sum of the vector representations of its n-grams

N-gram embedding

Bojanowski, P., Grave, E., Joulin, A. and Mikolov, T., 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, *5*, pp.135-146.

Limitations of Word2Vec

- The embeddings are context-free
  - Each (sub-)word is mapped to only one vector
  - Polysemous words with wildly different meaning have same vector



- The embeddings do not consider the order of words

- Every word is weighted the same, regardless of importance

Outline

- Course Organization
- What is a Language Model?
- Language Representations
- **Common Language Tasks**
- The Transformer Architecture
- Pre-trained Language Models

Common Language Tasks

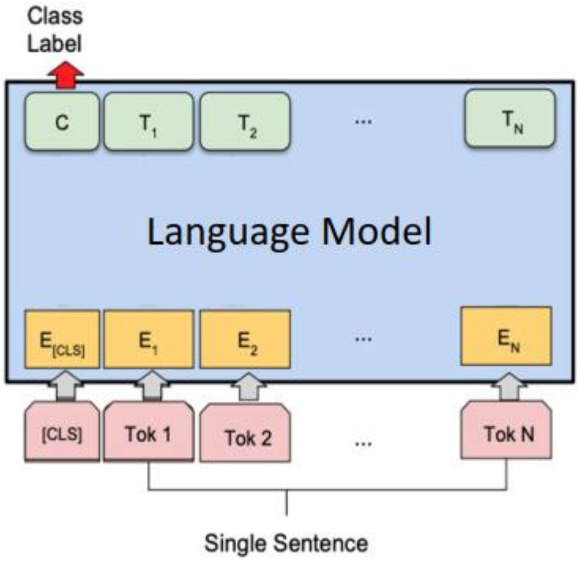
- Sentence-level tasks:
  - Single sentence classification tasks: text classification, sentiment analysis, …
  - Sentence-pair classification tasks: sentence entailment, …
  - Sentence generation tasks: machine translation, question answering, …

- Token-level tasks:
  - Part-of-speech tagging
  - named entity recognition
  - …

Single-Sentence Tasks

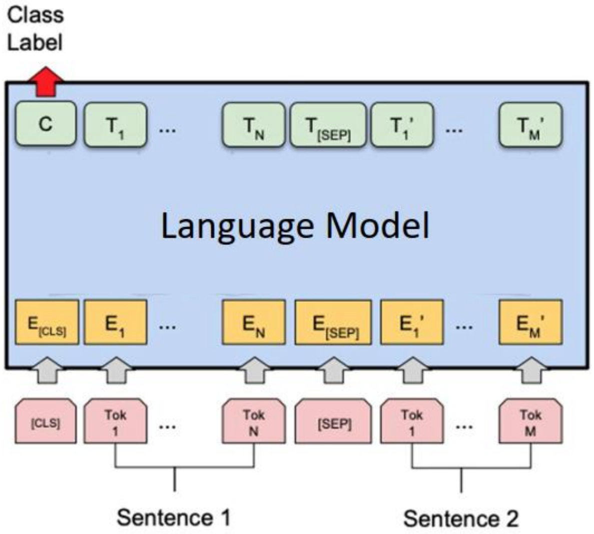
UNIVERSITY OF MANNHEIM
Data and Web Science Group

- Text classification tasks
  – Input:
    The bike is too small and I want to return it.
  – Output:
    <refund, return, check_status>
- Sentiment Analysis
  – Input:
    The restaurant is crowded and I waited for my food for thirty minutes!
  – Output:
    <positive, negative>



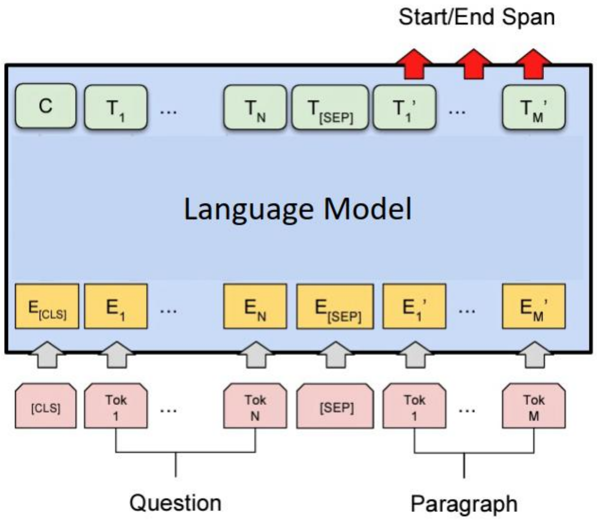
Sentence-Pair Tasks

- Sentence entailment
  - Input:
    Sentence 1: Our Large Language Model course meets on Thursdays at the University of Mannheim
    Sentence 2: There is a large language model course at the University of Mannheim
  - Output:
    <entailment, contradiction, neutral>



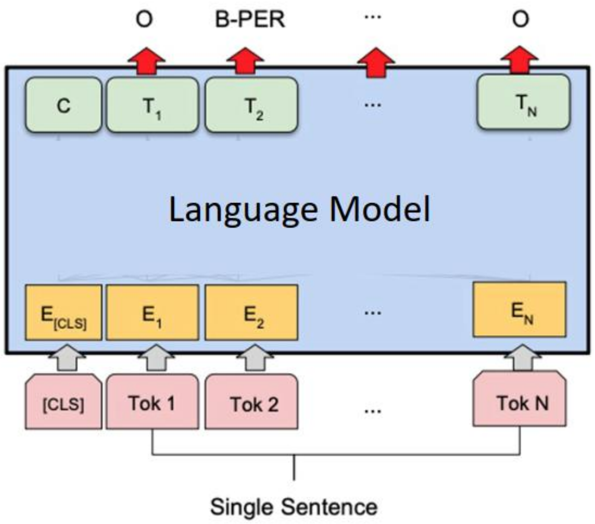
Sentence Generation Tasks

- Machine Translation
  - Input:
    English: This is good. German:
  - Output:
    Das ist gut.



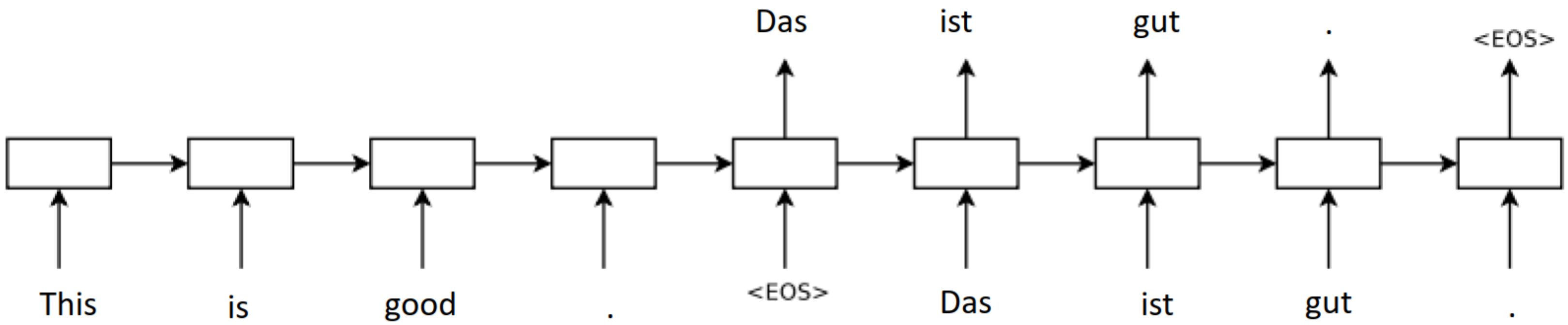
Token-level Tasks

- Named Entity Recognition
  - Input:
    St. Louis is located in the state of Missouri.
  - Output:
    <Begin-Location><Inside-location> O O O O O O <Begin-Location> O



Encoder and Decoder Models

- Language tasks can be broadly categorized into language understanding and language generation
- Encoder models are generally used to **understand** input sentences
- Decoder models are generally used to **generate** sentences



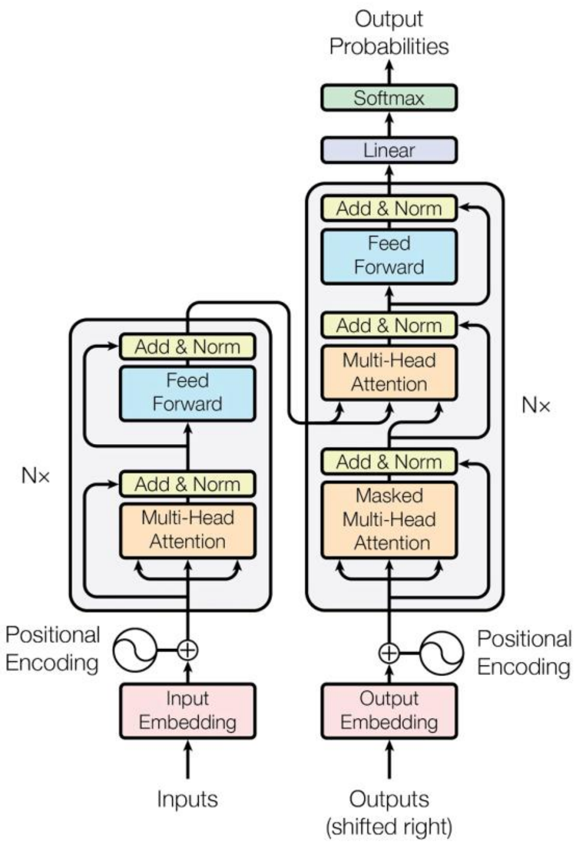
Outline

- Course Organization
- What is a Language Model?
- Language Representations
- Common Language Tasks
- **The Transformer Architecture**
- Pre-trained Language Models

The Transformer Architecture

UNIVERSITY OF MANNHEIM
Data and Web Science Group

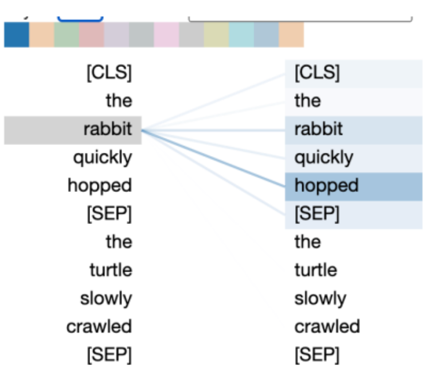
- Input Embedding
- Positional Encoding
- 12 Transformer layers
  - 6 encoder layers
  - 6 decoder layers
- Linear + Softmax layer for next word prediction

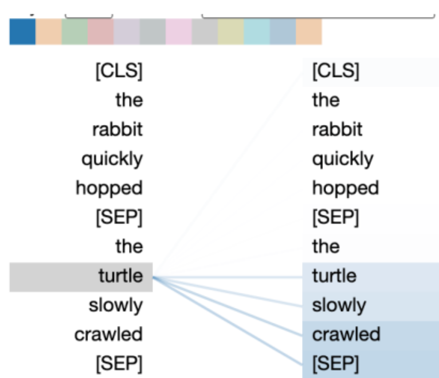


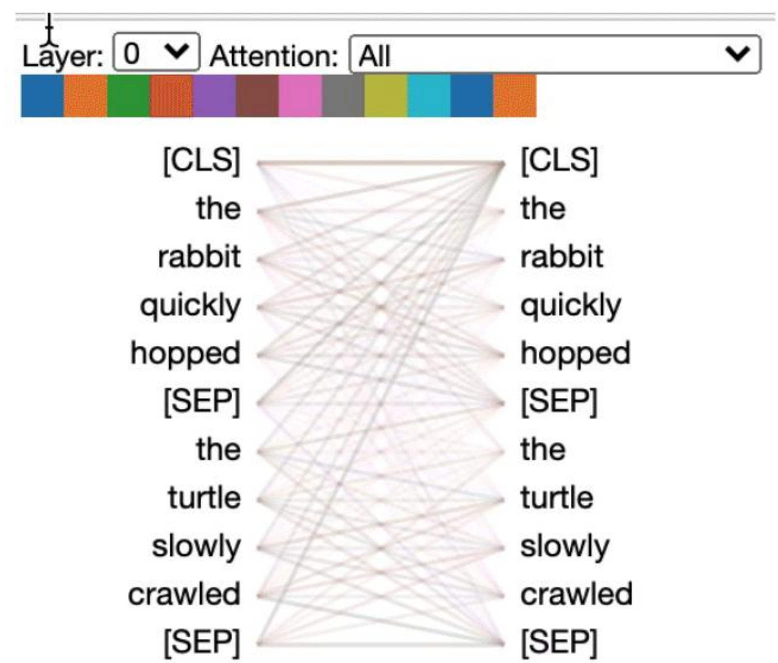
Vaswani, A., et al., 2017. Attention is All You Need. *Advances in Neural Information Processing Systems*.

Self-Attention

- Self-Attention: Each token attends to every other token in the sequence with differing weights
- Embeddings are **contextualized** based on surrounding words
- Demo for the BERT Transformer: https://github.com/jessevig/bertviz

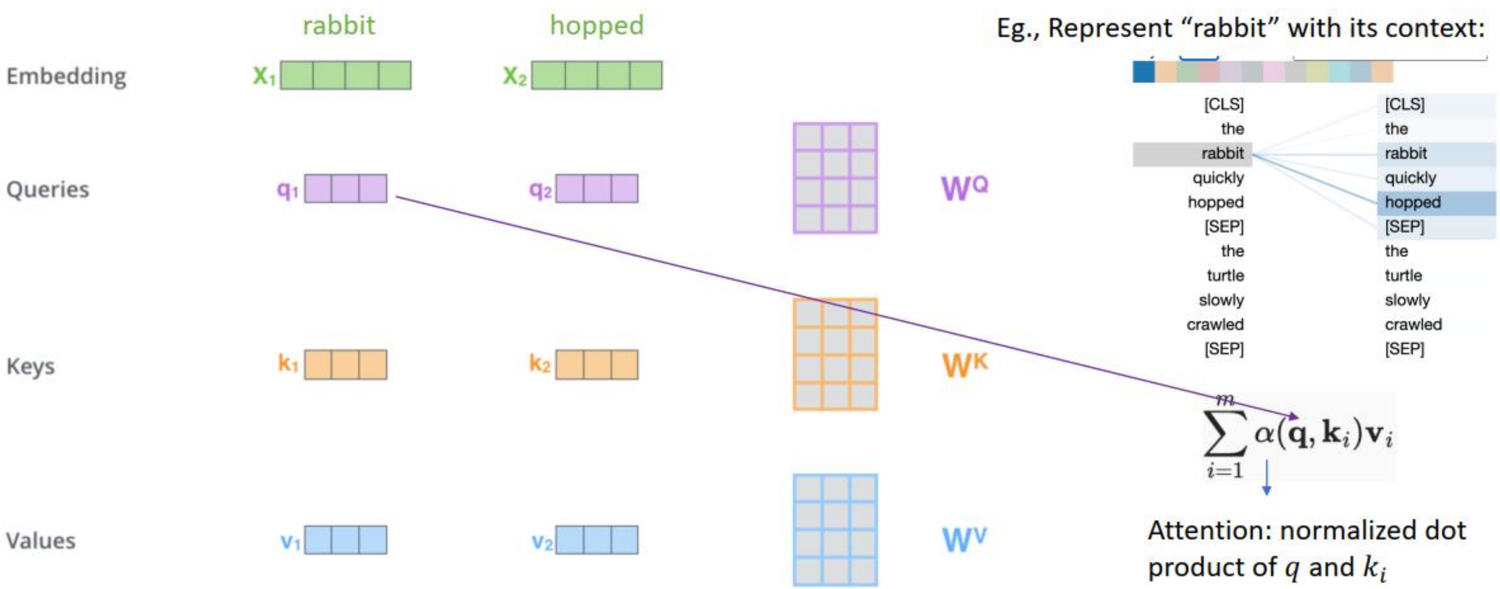




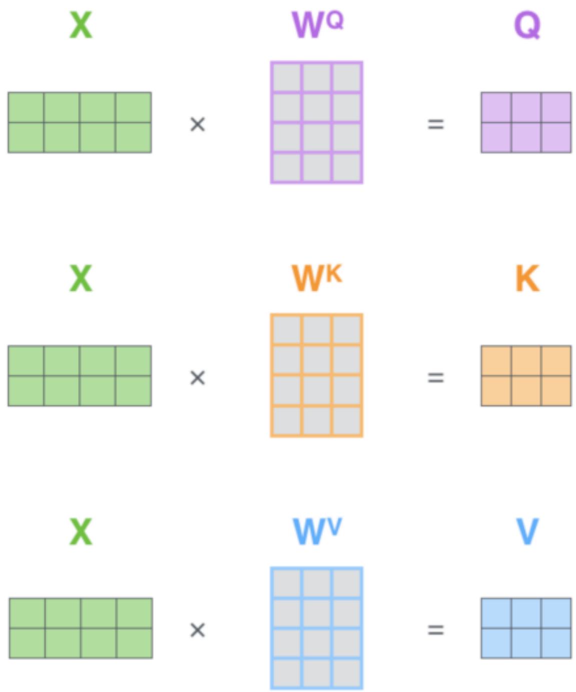


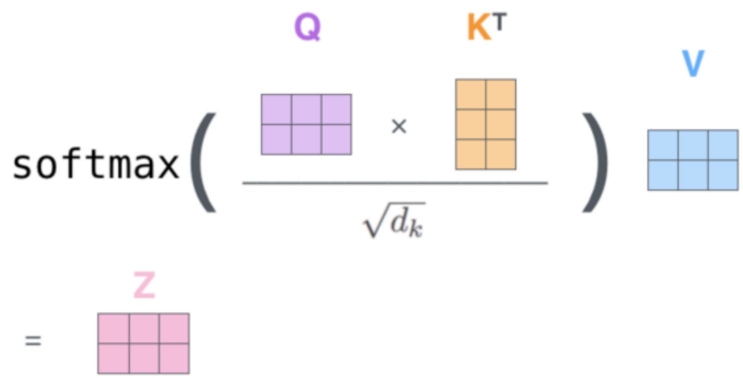
Self-Attention

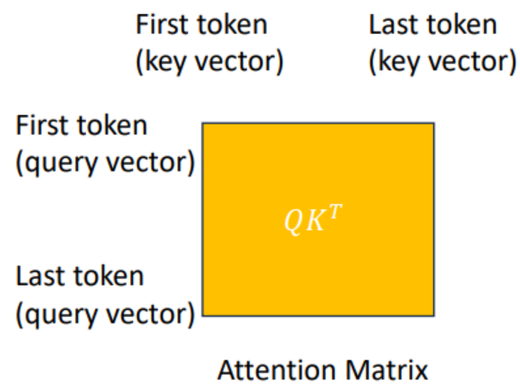
- Calculate the attention weight from a query word w_q (e.g. “rabbit”) to another word w_k o
- Each word is represented as a query, key and value vector.
- The vectors are obtained from the input embeddings multiplied by a weight matrix



Self-Attention: Matrix Calculation



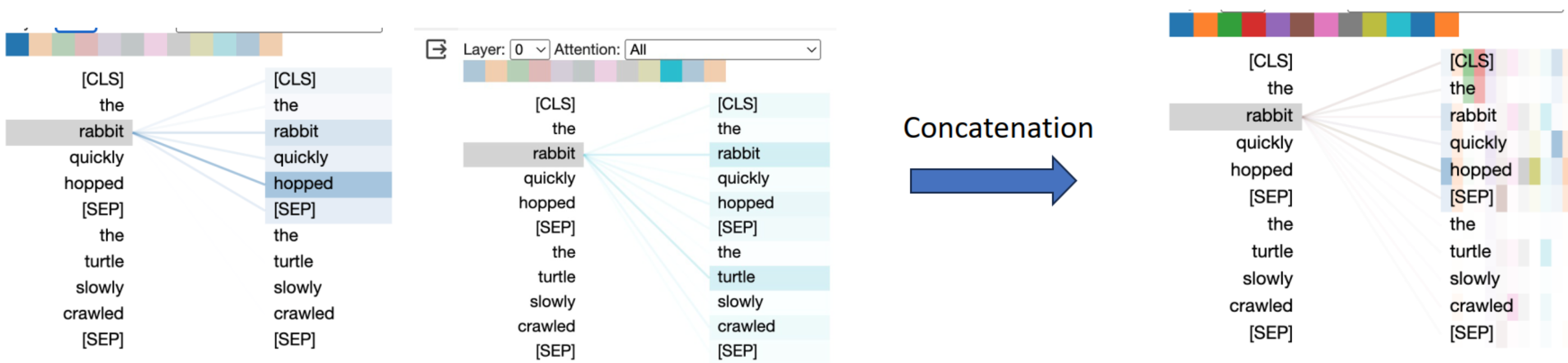




Multi-Head Attention

- Input: Multiple independent sets of query, key, value matrices
- Output: Concatenated outputs of all attention heads
- Advantage: Each attention head can focus on different patterns of the data

$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$



Encoder Model

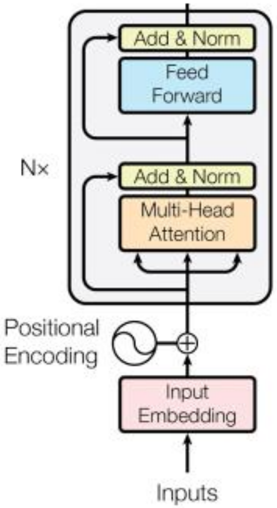
- Multi-head attention layer captures information from different patterns at different positions

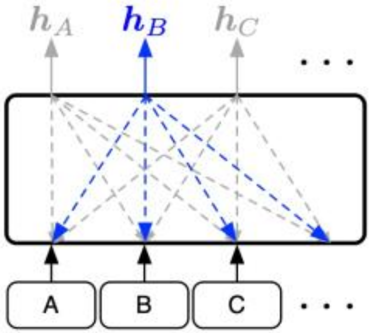
$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

- Feed-forward layer is applied to each token position without interaction with other positions

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

- **Bi-directional attention**: Every token attends to all other tokens

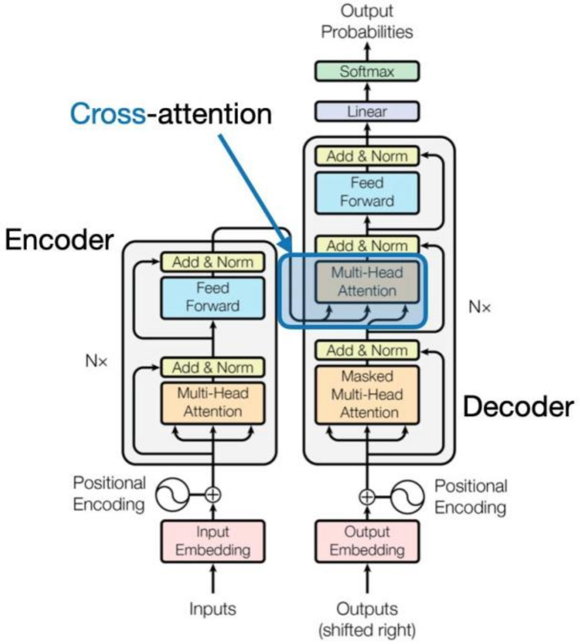




Decoder Model

- Multi-head self-attention: only allowed to attend to earlier positions (left side)
  - Q is from the generated tokens
  - K, V matrices are from the previously generated tokens

- Multi-head cross-attention: attend to the input sequence
  - Q is from the generated tokens
  - K, V matrices are from the input tokens

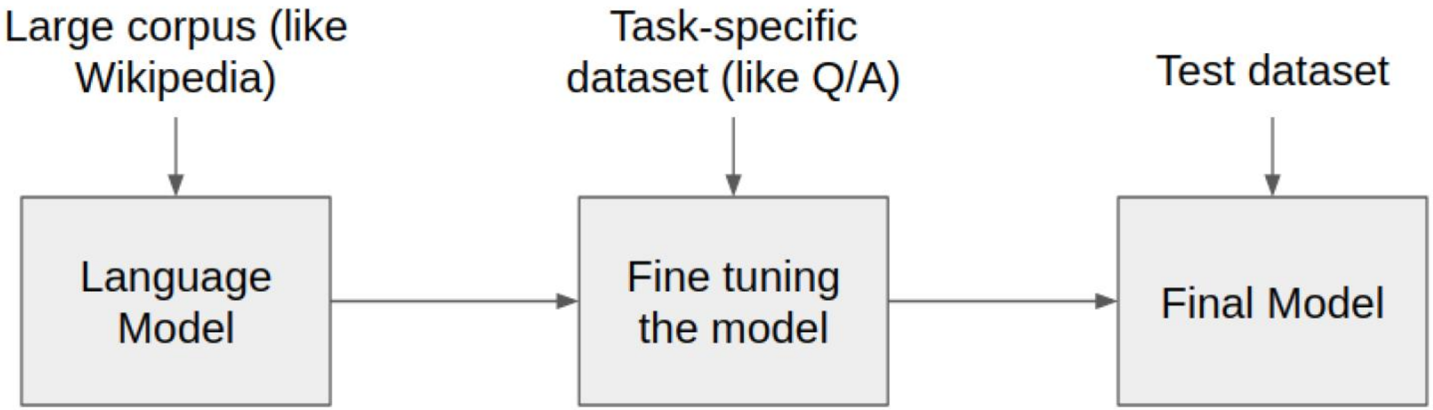


Outline

- Course Organization
- What is a Language Model?
- Language Representations
- Common Language Tasks
- The Transformer Architecture
- **Pre-trained Language Models**

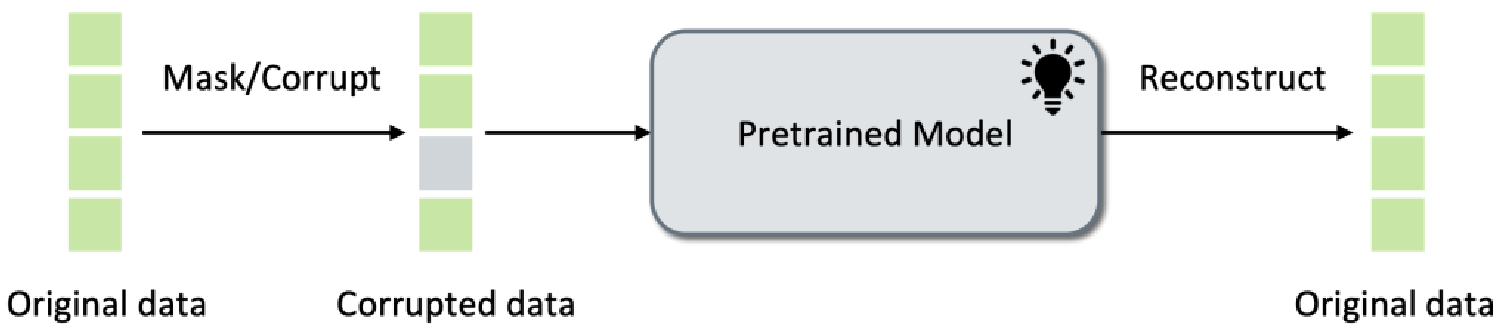
Pre-trained Language Models

- **Pre-training:** Train deep language models (usually Transformer-based) via self-supervised objectives on large-scale general-domain corpora
- **Fine-tuning:** Adapt the pre-trained language models (PLMs) to downstream tasks using task-specific data
- **Transfer Learning - The Power of PLMs:** Encode generic linguistic features and knowledge learned through large-scale pre-training, which can be effectively transferred to the target applications



General Pre-training Idea

- Self-supervised learning
- Make a part of the input unknown to the model
- Let the model predict that unknown part based on the known part



Different PLM Architectures

UNIVERSITY OF MANNHEIM
Data and Web Science Group

- **Decoder-only (Unidirectional)** PLM (e.g. GPT): Predict the next token based on previous tokens, usually used for **language generation tasks**

- **Encoder-only (Bidirectional)** PLM (e.g. BERT, XLNet, ELECTRA): Predict masked/corrupted tokens based on all other (uncorrupted) tokens, usually used for **language understanding/classification tasks**

- **Encoder-Decoder (Sequence-to-Sequence)** PLM (e.g., T5, BART): Generate output sequences given masked/corrupted input sequences, can be used for both **language understanding and generation tasks**

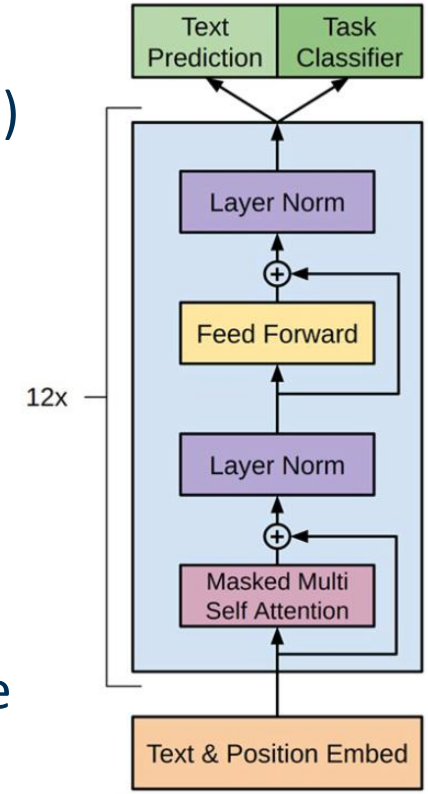
Generative Pre-training (GPT)

University of Mannheim Data and Web Science Group

- Architecture: multi-layer transformer decoder
- Leverages unidirectional context (usually left-to-right) for next token prediction (i.e., language modeling)

k previous tokens as context

$$\mathcal{L}_{\mathrm{LM}} = -\sum_i \log p(x_i \mid \boxed{x_{i-k}, \ldots, x_{i-1}})$$

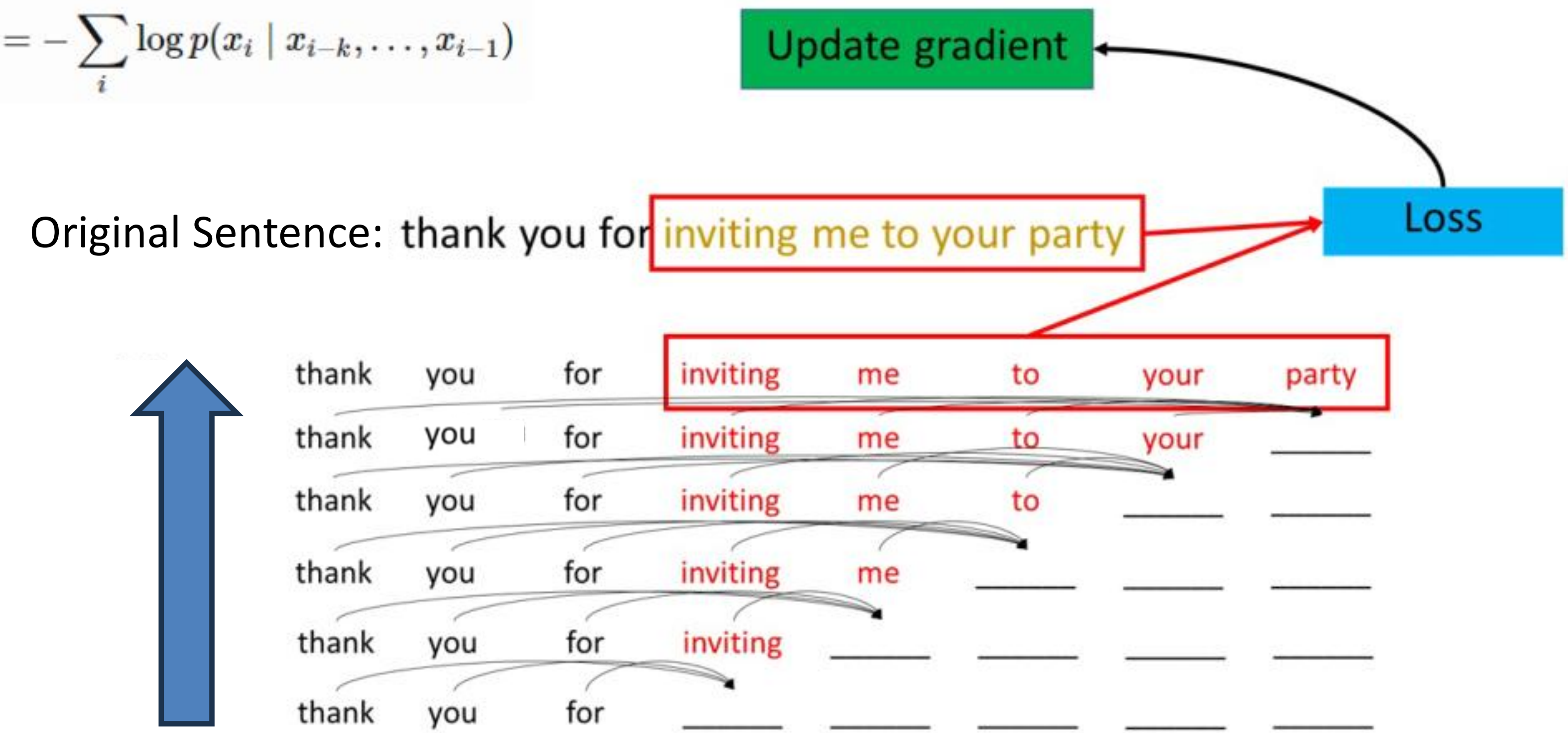


- The Transformer uses unidirectional attention masks (every token can only attend to previous tokens)
- Decoder architecture is the prominent choice in large language models

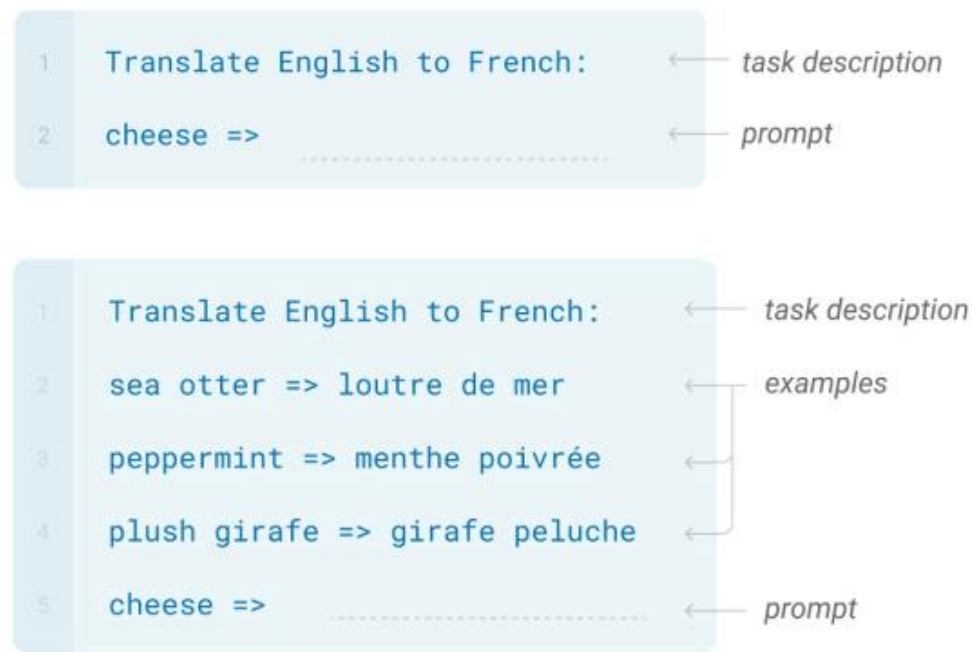
Radford et al., 2018. Improving Language Understanding by Generative Pre-Training
Radford et al., 2019. Language Models are Unsupervised Multitask Learners
Brown et al., 2020. Language Models are Few-shot Learners. In Proceedings of the 34th International Conference on Neural Information Processing Systems (pp. 1877-1901).

Decoder Pre-training

$$\mathcal{L}_{\text{LM}} = -\sum_i \log p(x_i \mid x_{i-k}, \ldots, x_{i-1})$$

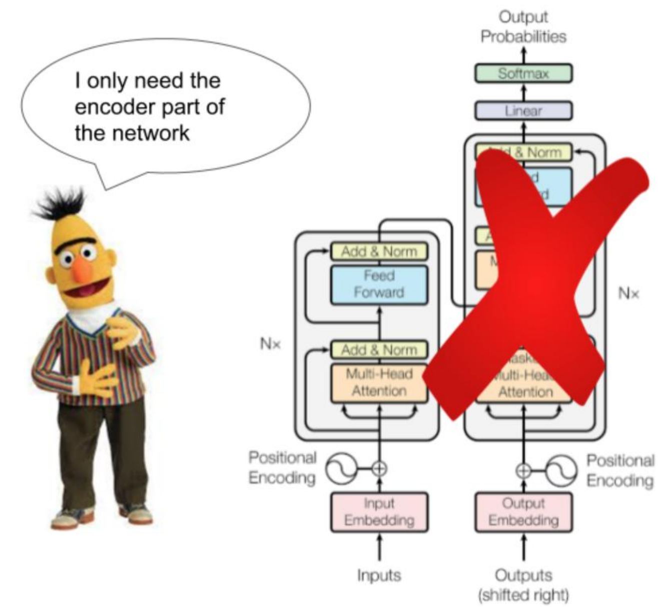


Usage of Decoder Models



The BERT Transformer

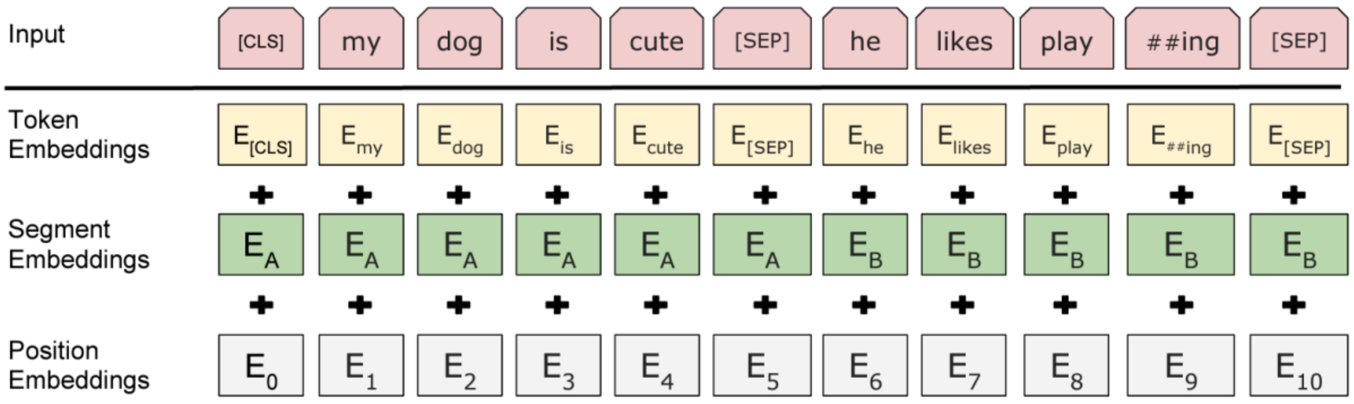
- Architecture: multi-layer transformer encoder
- Leverages bi-directional context
- Pre-training with
  - masked language modeling
  - next sentence prediction
- Pre-training corpus:
  - Wikipedia (2.5B)
  - BookCorpus (0.8B)
- Groundbreaking performance on a wide range of token-level and sentence-level tasks



Devlin et al., 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (pp. 4171-4186).

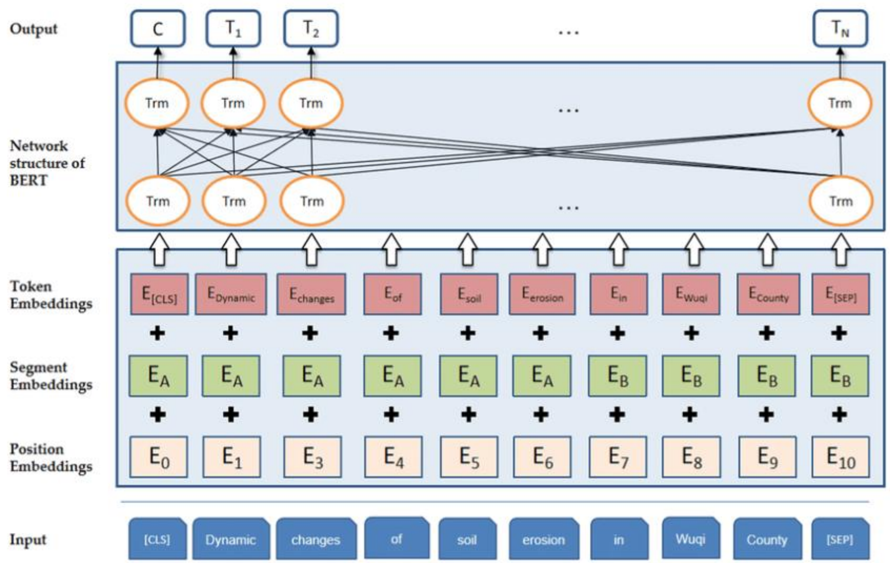
BERT Model Architecture

- Input: Sentence Pairs with special tokens [CLS] and [SEP]
  - Pair-wise tasks: question answering, translation, sentence entailment
  - [CLS]: beginning of a sentence
  - [SEP]: separation of two sentences
- WordPiece tokenization



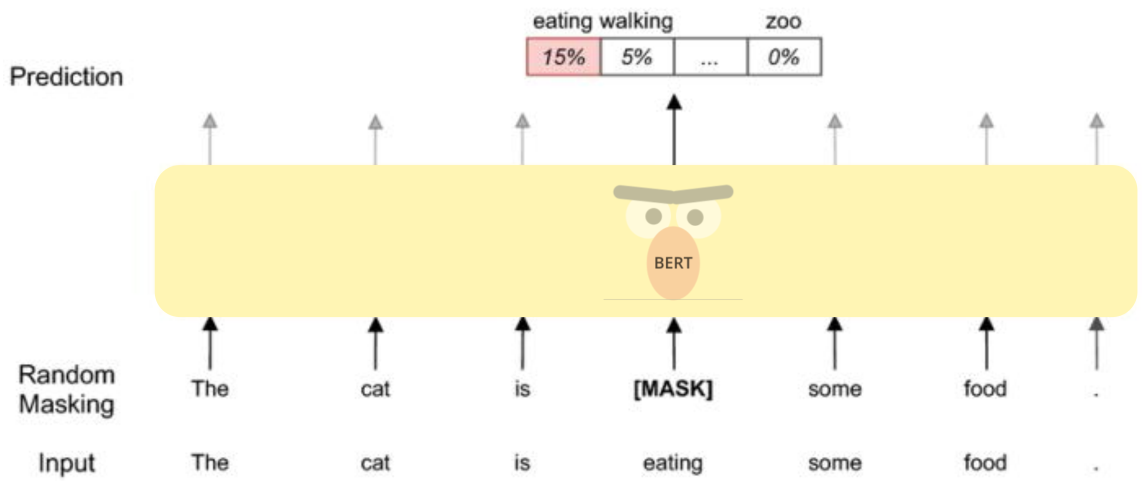
BERT Model Architecture

- Released in two versions
  - BERT-base: 12 layers, 768 hidden size, 12 attention heads (110M parameters)
  - BERT-large: 24 layers, 1024 hidden size, 16 attention heads (340M parameters)
- Bi-directional: Each token can attend to its left and right context



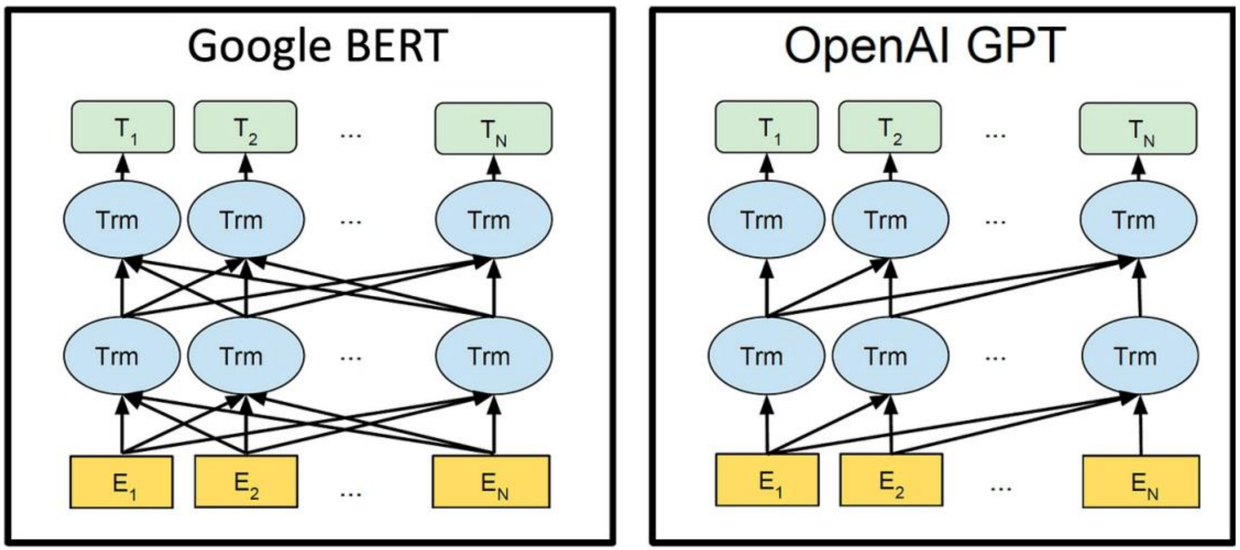
BERT Pre-training Objective: MLM

- Masked Language Modeling (MLM)
  - Randomly mask a few words in the original sentences
  - Predict the masked words using their left and right contexts
  - Masking ratio: 15%
  - Demo: https://huggingface.co/google-bert/bert-base-cased



Comparison with GPT

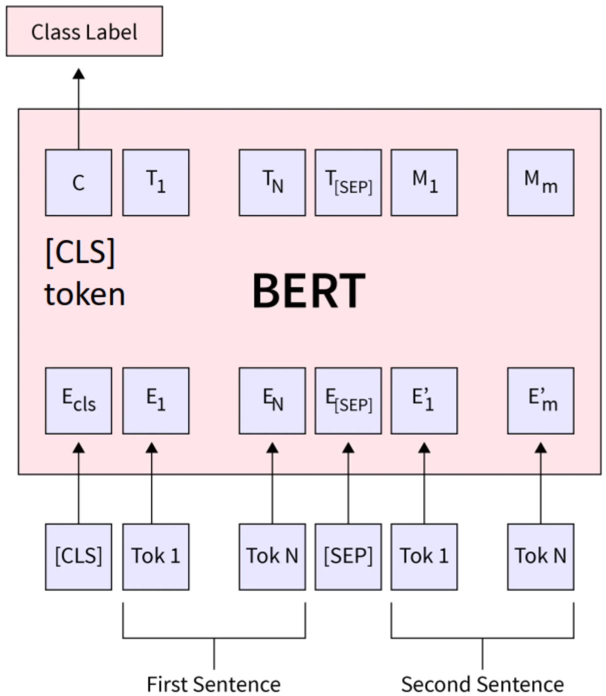
- Training objective:
  - Masked language modeling with bi-directional context (BERT)
  - Left-to-right next token prediction with left-only context (GPT)



BERT Pre-Training Objective: NSP

UNIVERSITY OF MANNHEIM
Data and Web Science Group

- Next Sentence Prediction (NSP)
  - Predict wether Sentence B is the next Sentence of Sentence A
  - Positive samples: two consecutive sentences in the corpus
  - Negative samples: sample a different sentence for A
  - Binary class labels: <is_next, not_next>



Variants of BERT

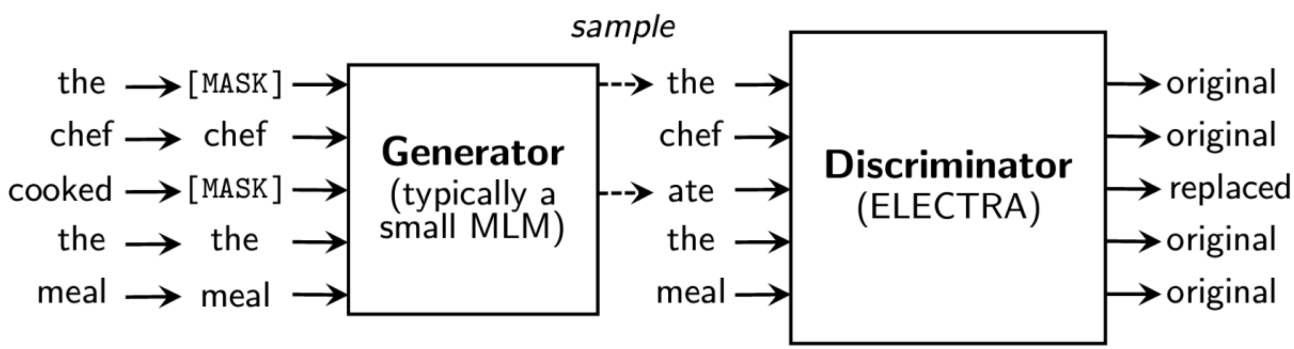
- **RoBERTa:** A Robustly Optimized BERT Pretraining Approach
  - Longer model training
  - On more data with bigger batches
  - Increased Vocabulary from ~30K to ~50K tokens
  - Next sentence prediction removed as it was experimentally found to not be useful
  - Dynamic changes to the [MASK] words in each epoch of training
- **DistilBERT:** a distilled version of BERT: smaller, faster, cheaper and lighter
  - Distilled from BERT-base
  - 40% less parameters, 60% faster, preserving 95% of performance based on the GLUE benchmark

Liu et al., 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*.
Sanh, V., 2019. DistilBERT, A Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. *arXiv preprint arXiv:1910.01108*.

Variants of BERT

- **ELECTRA:** Pre-training Text Encoders as Discriminators Rather than Generators
  - Replaced MLM objective by fist corrupting text sequences with an auxiliary small MLM model
  - ELECTRA model trained with binary objective
  - Class labels: <is_corrupted,not_corrupted>
  - No [MASK] tokens in input texts

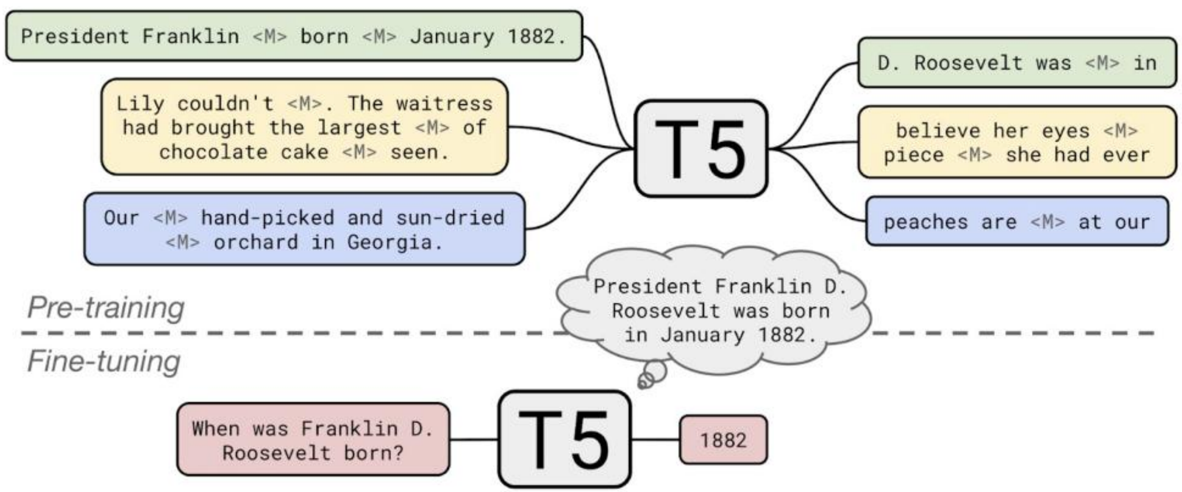
➔ Works better because no discrepancy between training and downstream task data



Clark et al., 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *International Conference on Learning Representations*.

The T5 Transformer

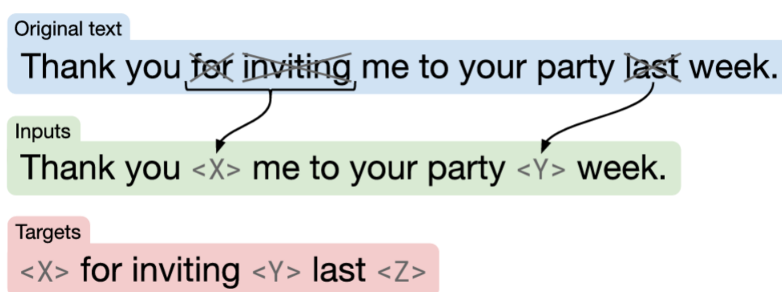
- Architecture: multi-layer transformer encoder-decoder
- How to predict a span of masked tokens within a sentence?
- T5: **T**ext-**t**o-**T**ext **T**ransfer **T**ransformer (60M-11B parameters)



Raffel et al., 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, *21*(140), pp.1-67.

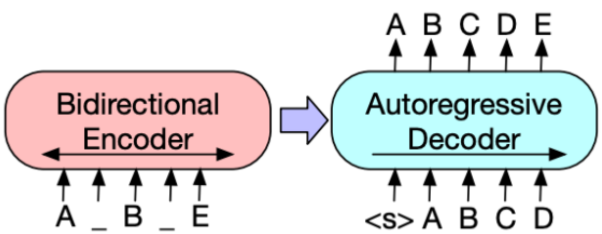
T5 Training Objectives

- Attention:
  - Full self-attention for input sequence in encoder
  - Left-to-right for decoder with cross-attention to full input in encoder
- Pre-training objective:
  - Mask out spans of texts …
  - Then generate the original spans
- Fine-tuning objective:
  - Convert any task into a sequence-to-sequence generation problem
  - No discrepancy between pre-training and fine-tuning

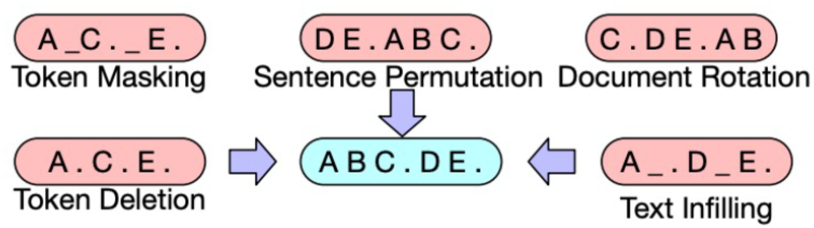


The BART Transformer

- **BART:** Denoising Autoencoder for Pre-training Sequence-to-Sequence Models
- Pre-training: Apply a series of noising schemes (e.g. masks, deletions, permutations…) to input sequences and train the model to recover the original sequences



BART architecture

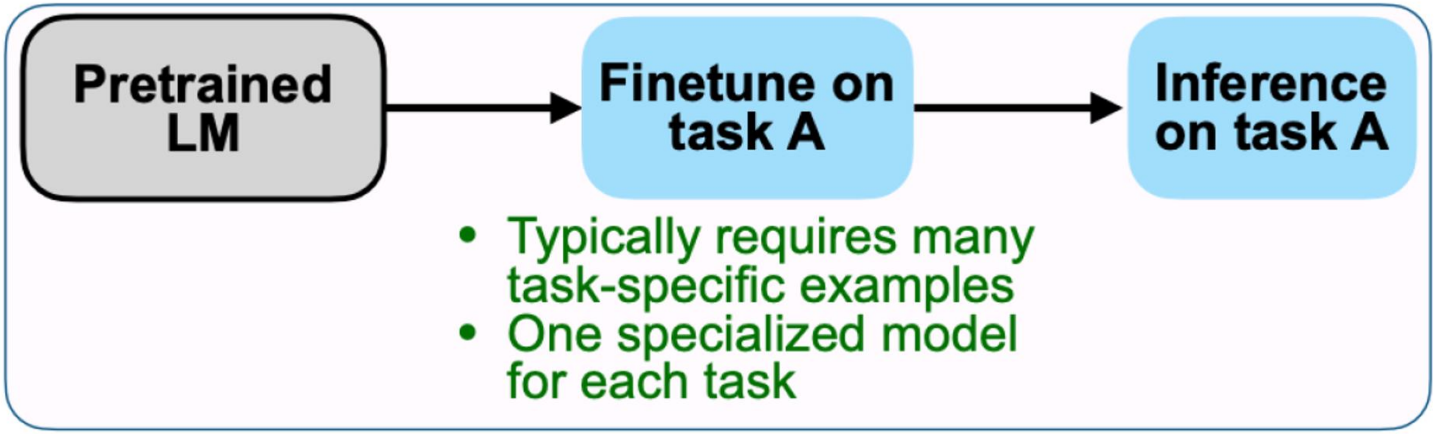


BART pretraining objectives

Lewis et al., 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 7871-7880).

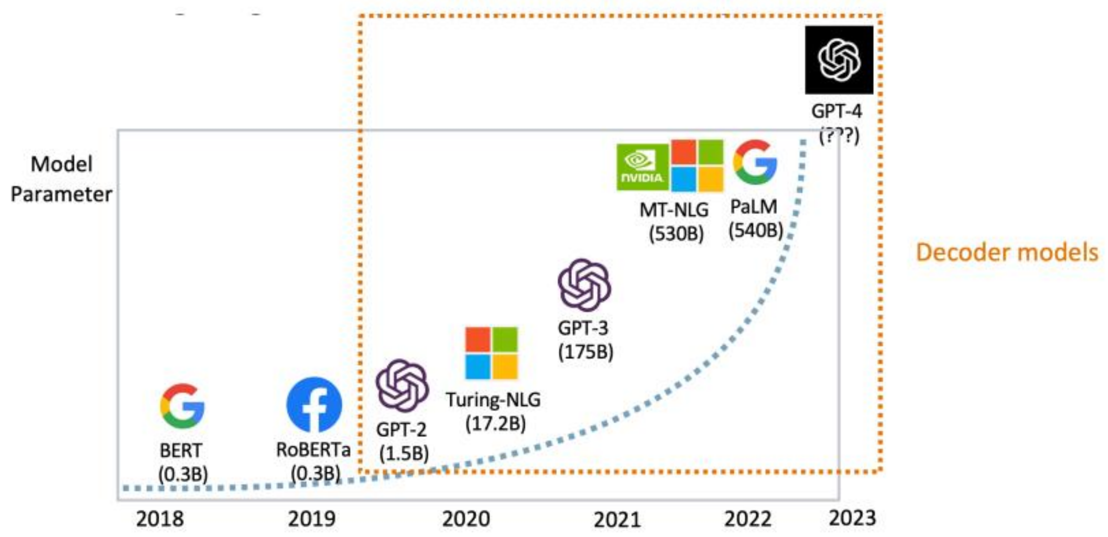
Fine-tuning PLMs

- The pre-training stage lets language models learn generic representations and knowledge from the corpus, but they are not fine-tuned on any form of user tasks.
- To adapt language models to a specific downstream task, we usually use task-specific datasets for fine-tuning



Scaling up Language Models

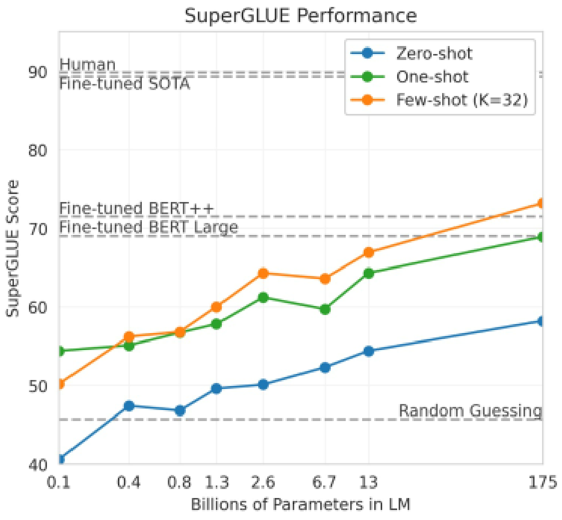
- Model size in terms of trainable parameters has increased significantly over the years, constantly increasing performance, especially for text generation models.
- The name **large language model** is *usually* applied to language models with more than 1B parameters (mostly decoder-only)



Performance of Zero-/Few-shot GPT 3

	SuperGLUE Average	BoolQ Accuracy	CB Accuracy	CB F1	COPA Accuracy	RTE Accuracy
Fine-tuned SOTA	**89.0**	**91.0**	**96.9**	**93.9**	**94.8**	**92.5**
Fine-tuned BERT-Large	69.0	77.4	83.6	75.7	70.6	71.7
GPT-3 Few-Shot	71.8	76.4	75.6	52.0	92.0	69.0

	WiC Accuracy	WSC Accuracy	MultiRC Accuracy	MultiRC F1a	ReCoRD Accuracy	ReCoRD F1
Fine-tuned SOTA	**76.1**	**93.8**	**62.3**	**88.2**	**92.5**	**93.3**
Fine-tuned BERT-Large	69.6	64.6	24.1	70.0	71.3	72.0
GPT-3 Few-Shot	49.4	80.1	30.5	75.4	90.2	91.1



See you next week!

- Next time: Training LLMs
  - Instruction tuning
  - Reinforcement learning from human feedback