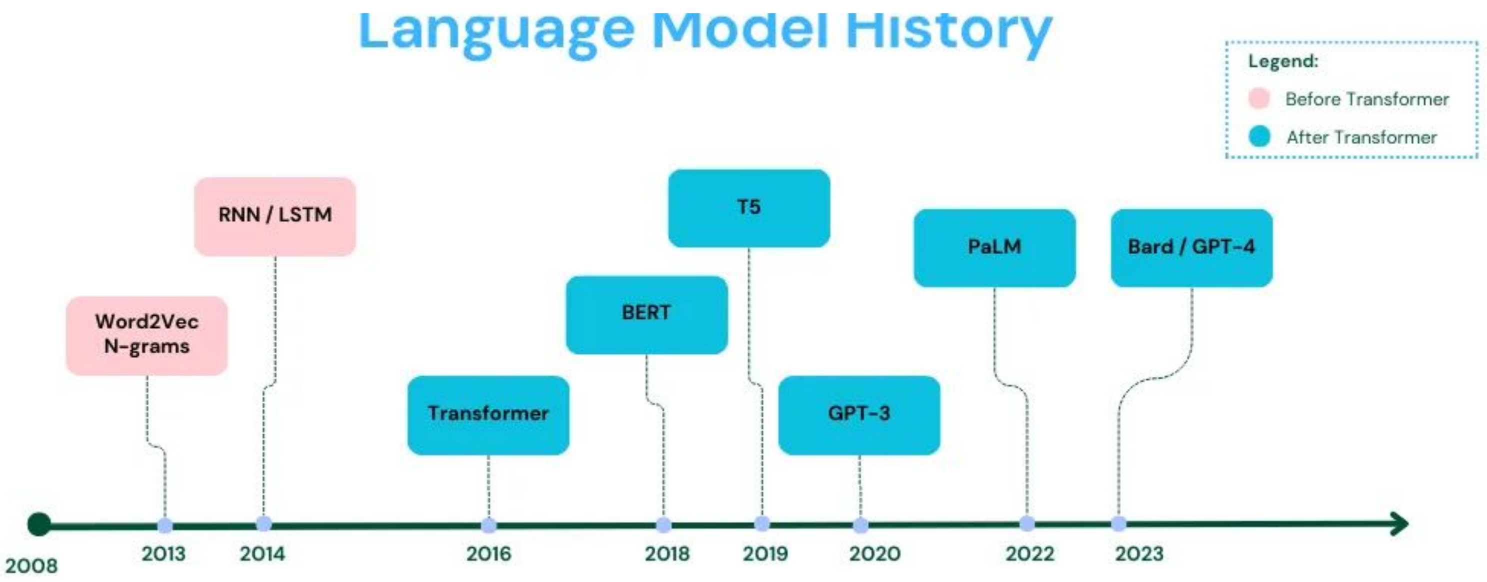
Cornell Bowers C·IS
College of Computing and Information Science

Transformers

How to handle text data?

Language Modelling History



https://medium.com/@kirudang/language-model-history-before-and-after-transformer-the-ai-revolution-bedc7948a130

Language Modeling: predict the next word

Assign probabilities to text.

Given a sequence $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$, we want to **maximize** $P(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$.

$$P(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T) = P(\mathbf{x}_1)P(\mathbf{x}_2|\mathbf{x}_1)P(\mathbf{x}_3|\mathbf{x}_1, \mathbf{x}_2)P(\mathbf{x}_4|\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)\ldots P(\mathbf{x}_T|\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_{T-1})$$

P(I like cats because they look cute) = P(I) P(like | I) P(cats | I like) P(as | I like cats) P(they | I like cats because)

P(look | I like cats because they) P(cute | I like cats because they look)

Predict the next word given current text!

n-Gram Language Model

n-Gram: chunk of *n* consecutive words

Uni-gram: "I" "like" "cats" "as" "they" "look" "cute"

Bi-gram: "I like" "like cats" "cats as" "as they" …

Tri-gram: "I like cats" "like cats as" "cats as they" …

Count the frequency of each *n*-grams and predict next word!

Assume each word only depends on previous *n - 1* words.

$$P(\mathbf{x}_t|\mathbf{x}_1,\ldots,\mathbf{x}_{t-1}) = P(\mathbf{x}_t|\mathbf{x}_{t-n+1},\ldots,\mathbf{x}_{t-1})$$
$$= \frac{\text{count}(\mathbf{x}_{t-n+1},\ldots,\mathbf{x}_{t-1},\mathbf{x}_t)}{\text{count}(\mathbf{x}_{t-n+1},\ldots,\mathbf{x}_{t-1})}$$

In *bi-gram* LM

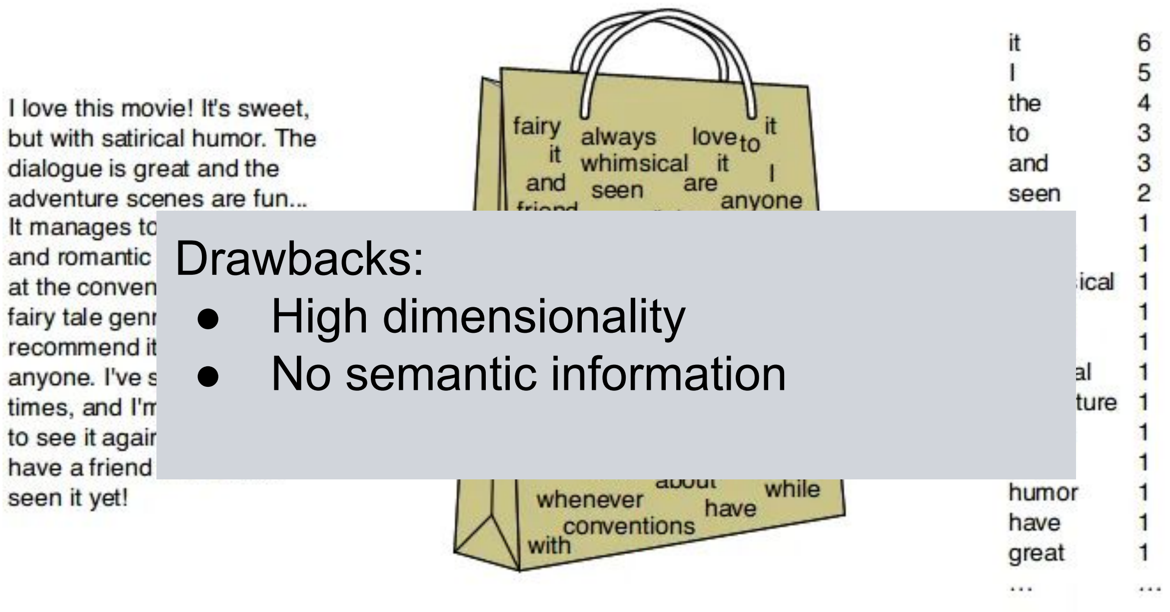
P(I like cats as they look cute) = P(I) P(like | I) P(cats | like) P(as | cats) P(they | because) P(look | they) P(cute | look)

Discuss: Do you want to have a large n or a small n in a n-gram model?

Bag of Words

https://koushik1102.medium.com/nlp-bag-of-words-and-tf-idf-explained-fd1f49dce7c4

Bag of Words



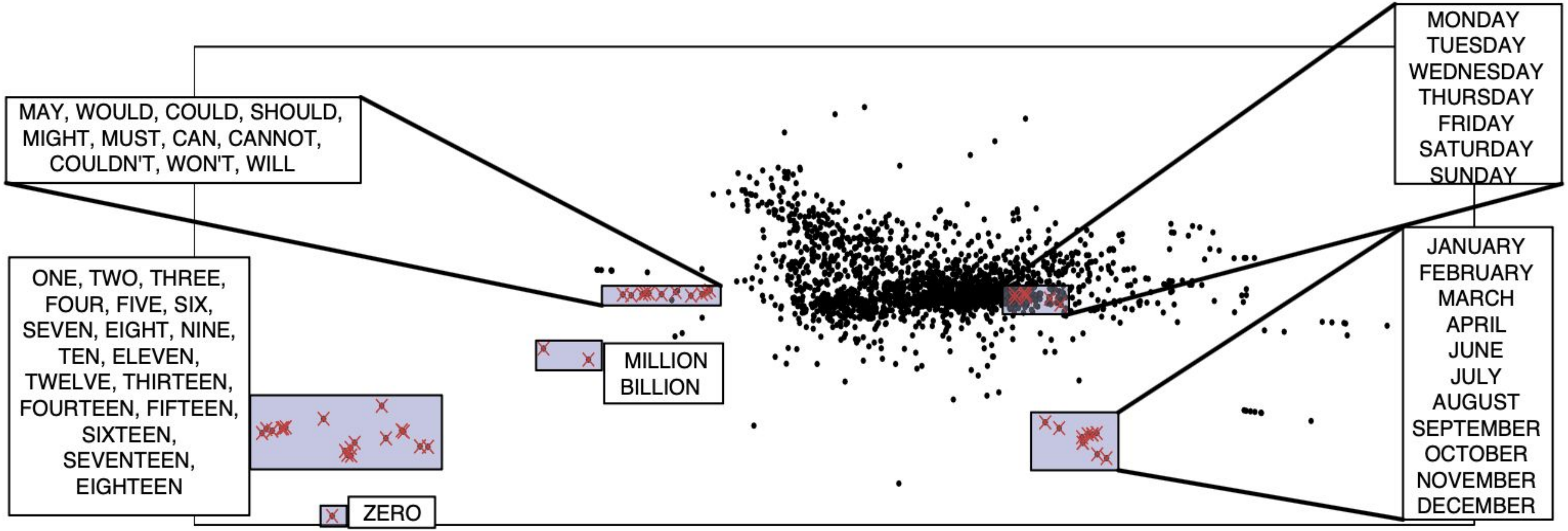
Drawbacks:

- High dimensionality
- No semantic information

https://koushik1102.medium.com/nlp-bag-of-words-and-tf-idf-explained-fd1f49dce7c4

Cornell Bowers C·IS

Word Embeddings



Blitzer et al. 2004

What does ong choi mean?

Suppose you see these sentences:

- Ong choi is delicious sautéed with garlic.
- Ong choi is superb over rice
- Ong choi leaves with salty sauces

And you've also seen these:

- ...spinach sautéed with garlic over rice
- Chard stems and leaves are delicious
- Collard greens and other salty leafy greens

Generative AI is experimental. Learn more

Ong choy is **a leafy green vegetable with long, hollow stems and slender leaves**. It's also known as Chinese water spinach, Chinese water spinach, or hollow stem spinach.

The Seasoned Wok
Ong Choy (Water Spinach) Recipe with Fermented Bean...
Nov 3, 2022 — What is Ong Choy, Rau Muong or Water Spinach? Ong choy i...

The Woks of Life
Ong Choy with XO sauce - The Woks of Life
May 2, 2017 — Ong Choy is a popular Chinese leafy green vegetable that's...

Onolicious Hawai'i
Garlic and Fish Sau - Onolicious Hawai
Jan 7, 2021 — What is C
Hawaii everyone know

Word2Vec

- We want vectors for words so that the context of a word can suggest the vector of this word, and vice versa
- Idea: **Similar words appear in similar contexts**

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

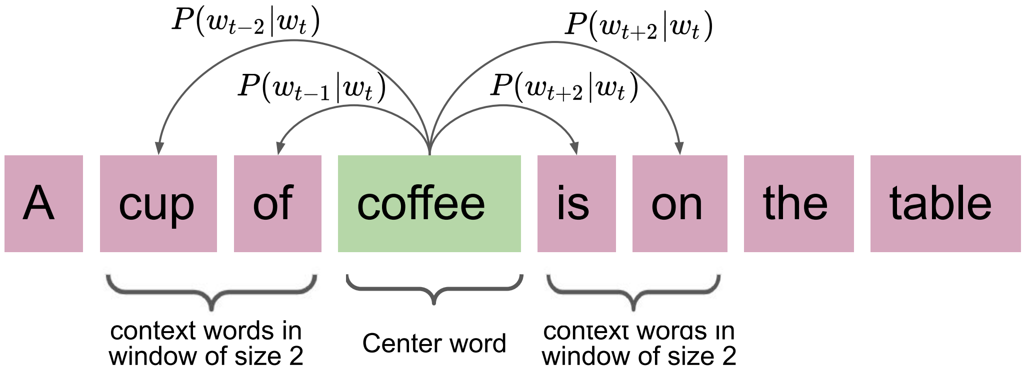
Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

Abstract

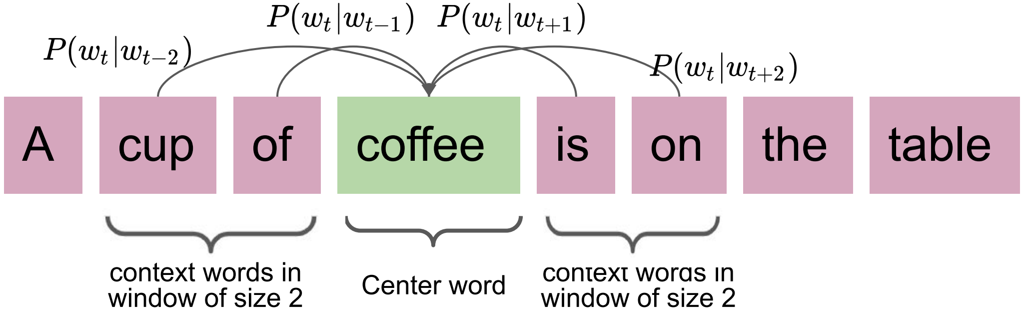
We propose two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity task, and the results are compared to the previously best performing techniques based on different types of neural networks. We observe large improvements in accuracy at much lower computational cost, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion words data set. Furthermore, we show that these vectors provide state-of-the-art performance on our test set for measuring syntactic and semantic word similarities.

Word2Vec - Training

SkipGram



cBow



Looking closer…

Context word

Target word

$$p(\mathbf{w}_c|\mathbf{w}_t) = \frac{\exp(\mathbf{w}_c^\top \mathbf{w}_t)}{\sum_j \exp(\mathbf{w}_c^\top \mathbf{w}_j)}$$

Other words

(One option is to have two embeddings for each word, one as target and one as context.)

Cross-Entropy Loss:

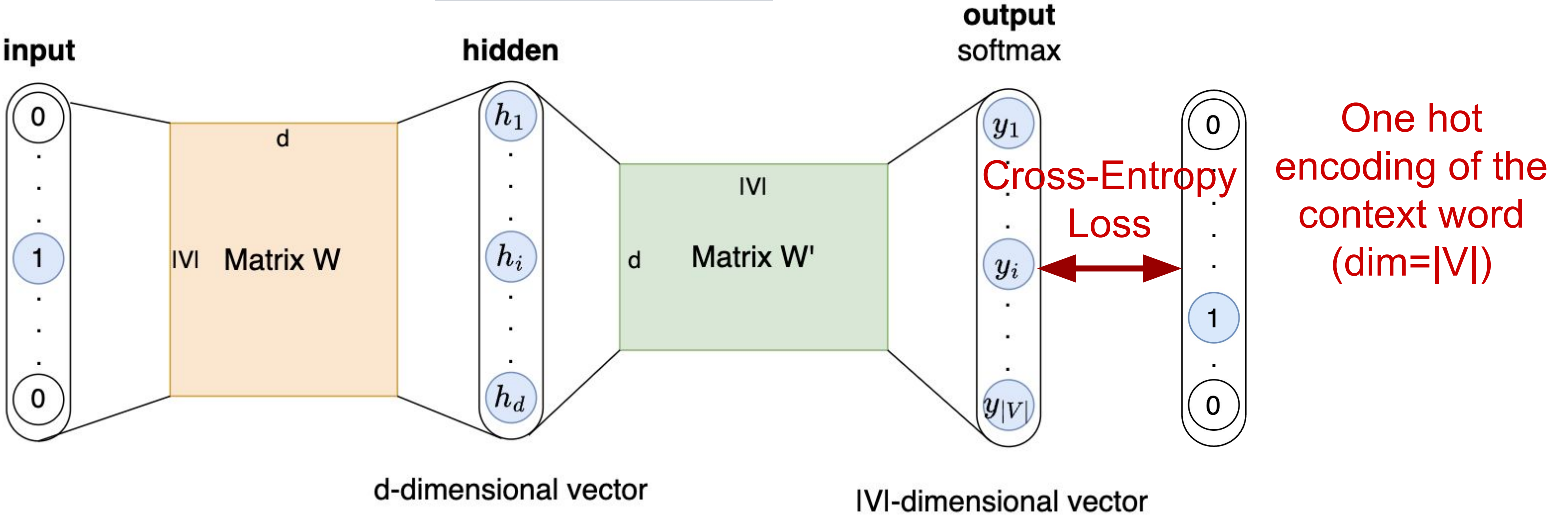
$$\mathcal{L}_{\mathbf{W}} = - \sum_{(c,t)\in D} \log\left(p(\mathbf{w}_c|\mathbf{w}_t)\right)$$

Cornell Bowers C·IS

Word2Vec Architecture - SkipGram

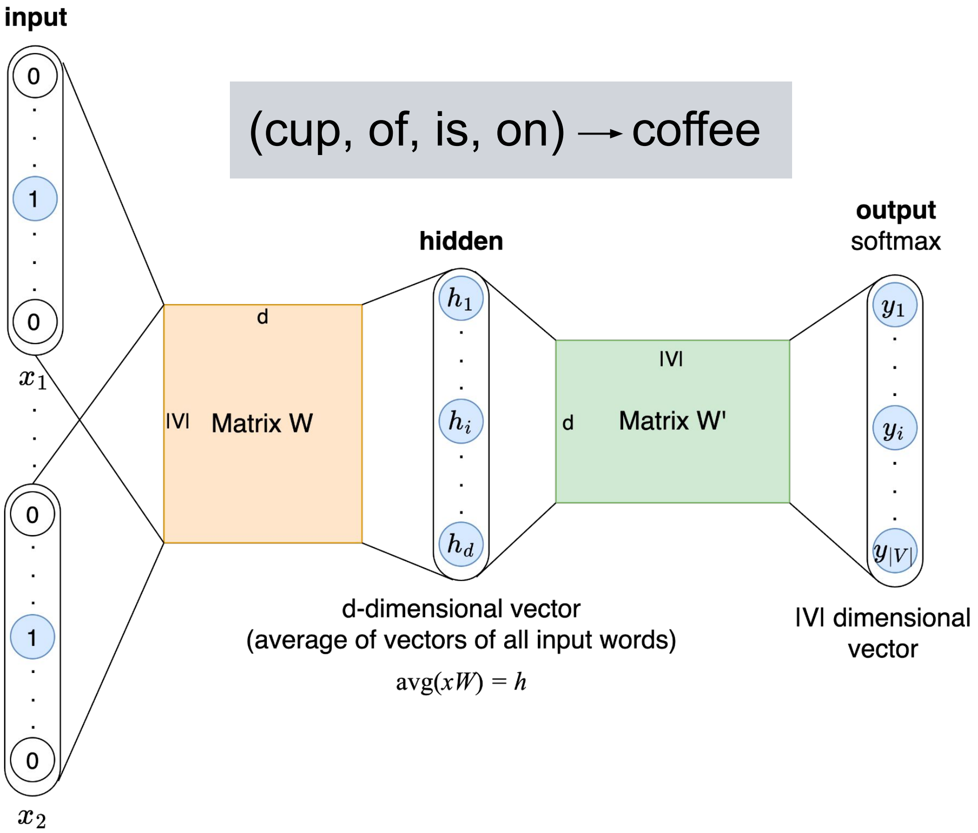
Predict context word from target word!

coffee ⟶ cup



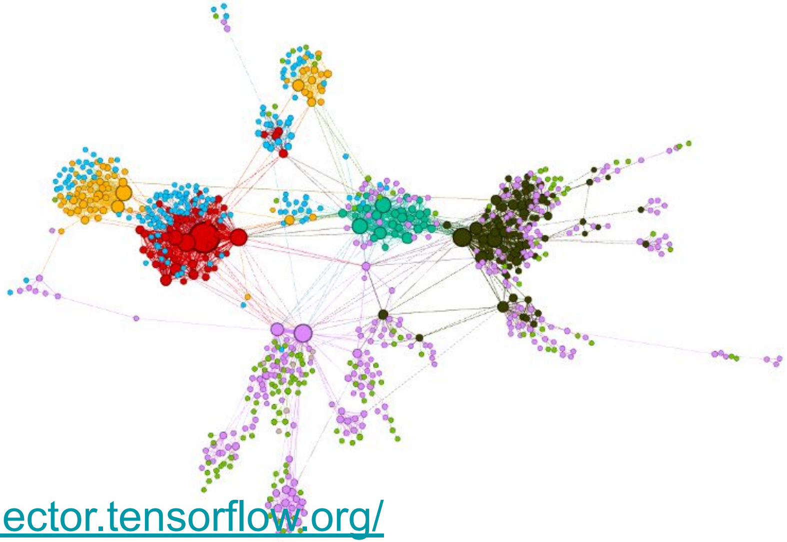
Cornell Bowers C·IS

Word2Vec Architecture - CBOW (continuous bag of words)



X 2 vec

- Generate vector representations (embeddings) for various data types
- Examples:
  - Word2Vec
  - Doc2Vec
  - Node2Vec
  - Item2Vec
  - Sent2Vec
  - Gene2Vec

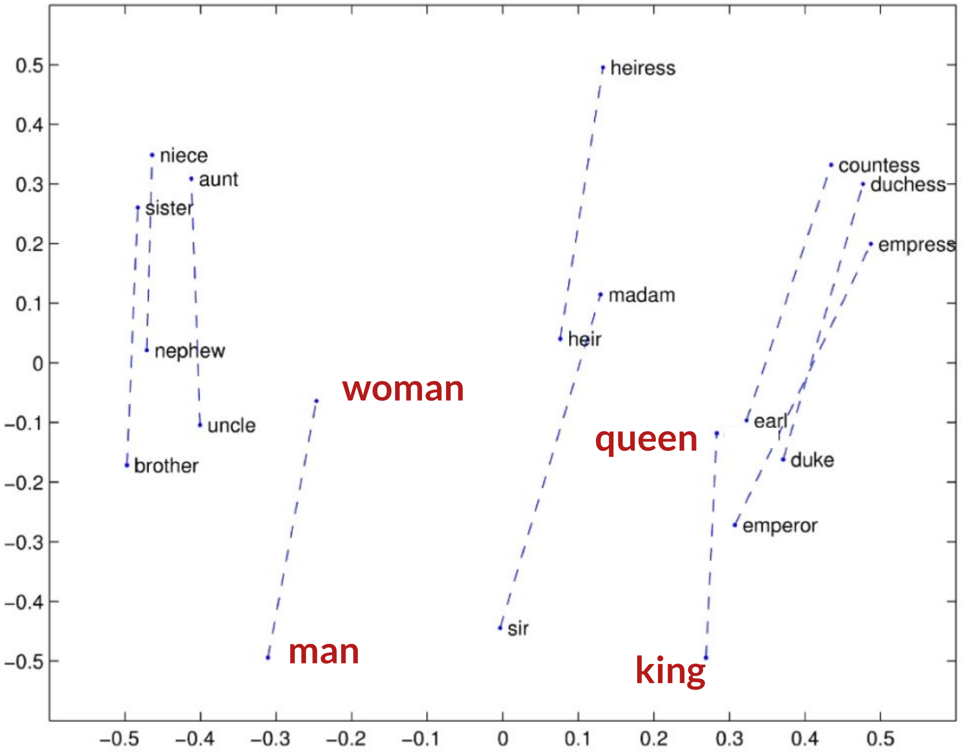


Visualize: https://projector.tensorflow.org/

Explore: http://epsilon-it.utu.fi/wv_demo/

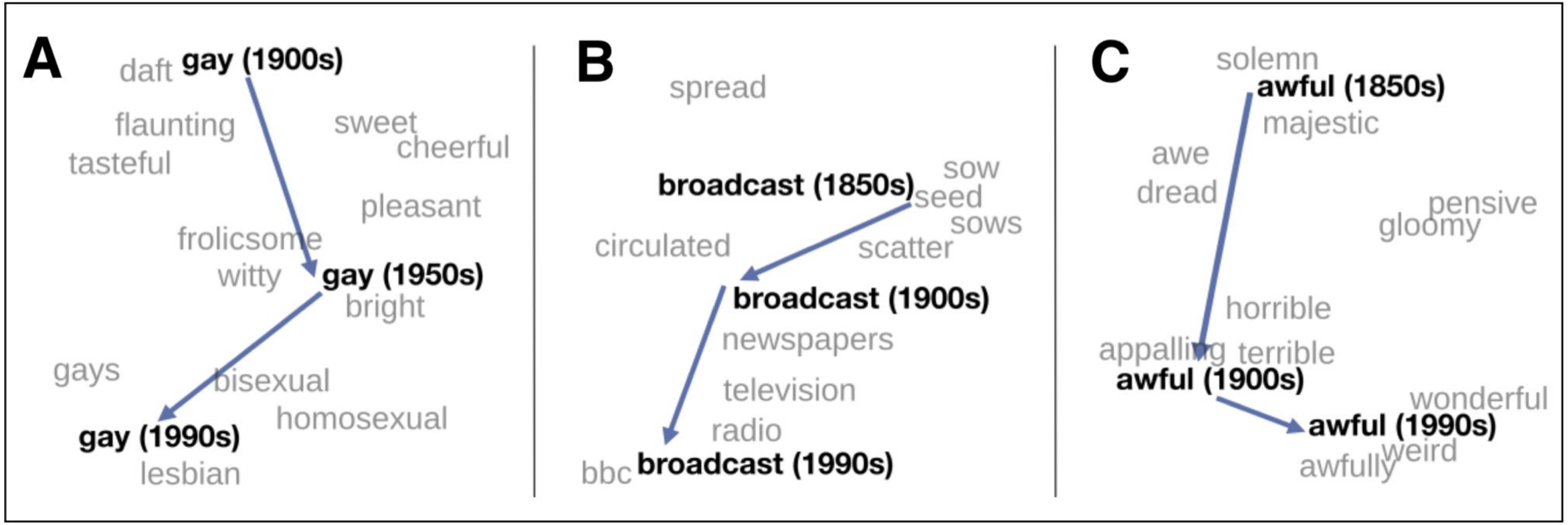
Cornell Bowers C·IS

In vector space…



Word embeddings are time-dependent (why?)

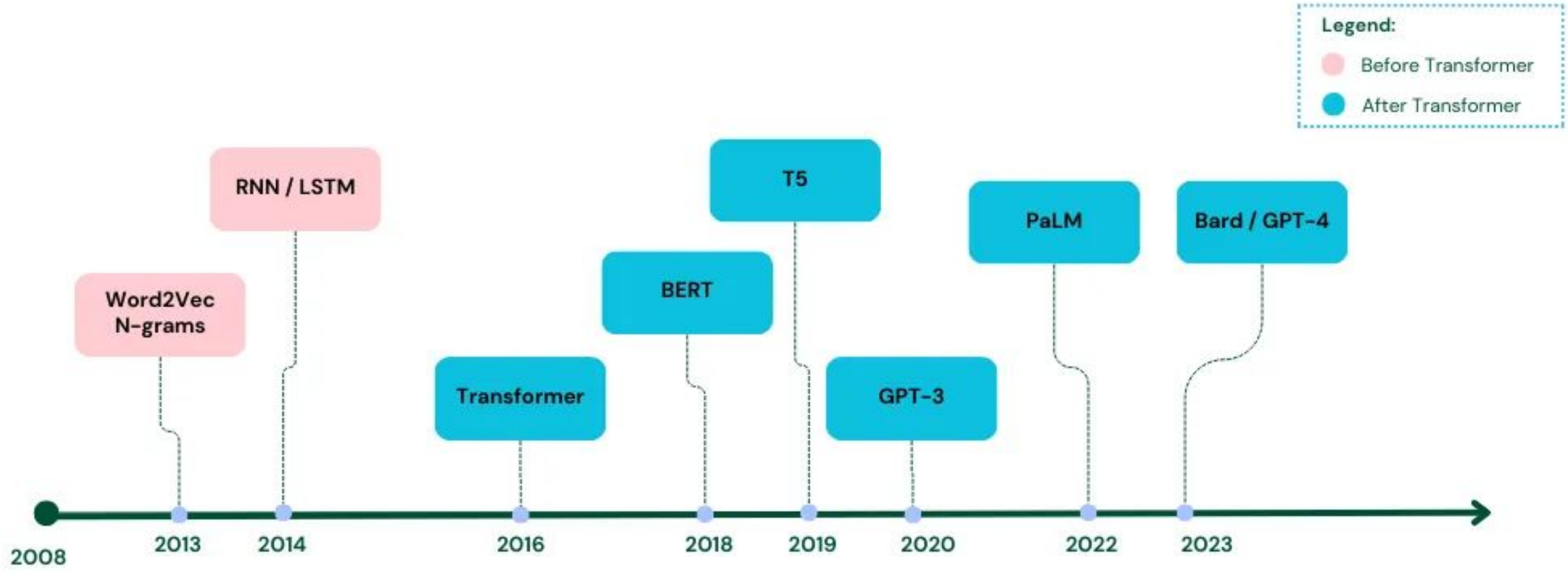
- Semantic similarity of words depends on *time.*



Problems with word2vec

- Words with multiple meanings only have one representation
  - eg. **bank** of river or **bank** of money
  - Need contextual information
- Limited Context
  - only trained on words within the context window

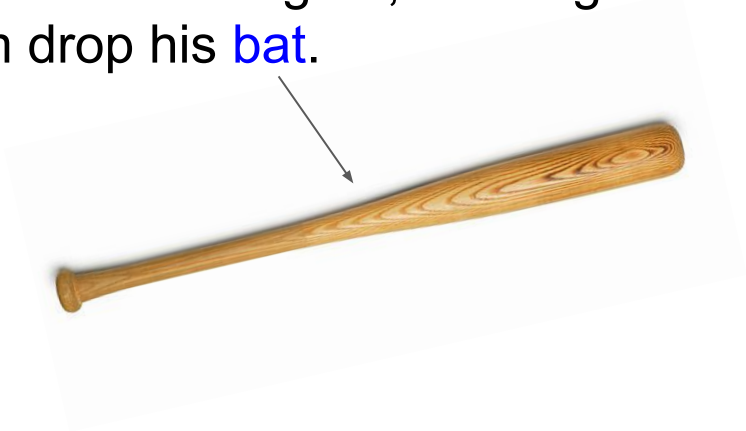
Language Modelling History



https://medium.com/@kirudang/language-model-history-before-and-after-transformer-the-ai-revolution-bedc7948a130

Self-Attention

A bat flew out of the dugout, startling the baseball player and making him drop his bat.



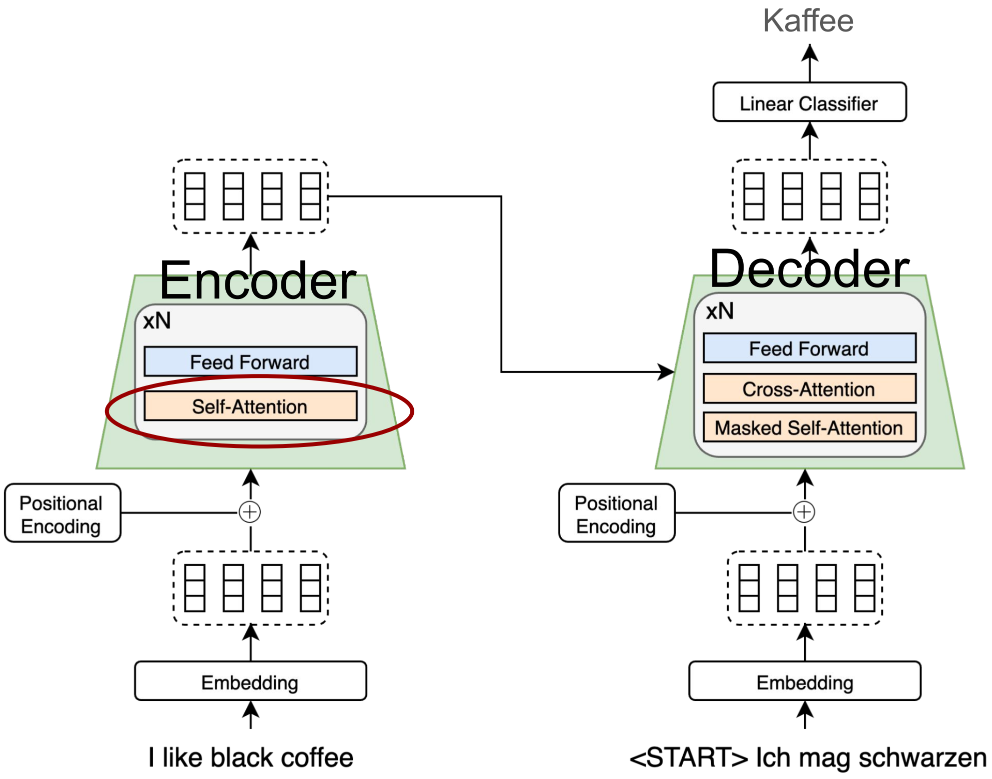
Self-Attention

Each word pays attention to the words around it, focusing more on the words which provide important information about its meaning.

A bat flew out of the dugout, startling the baseball player and making him drop his bat.

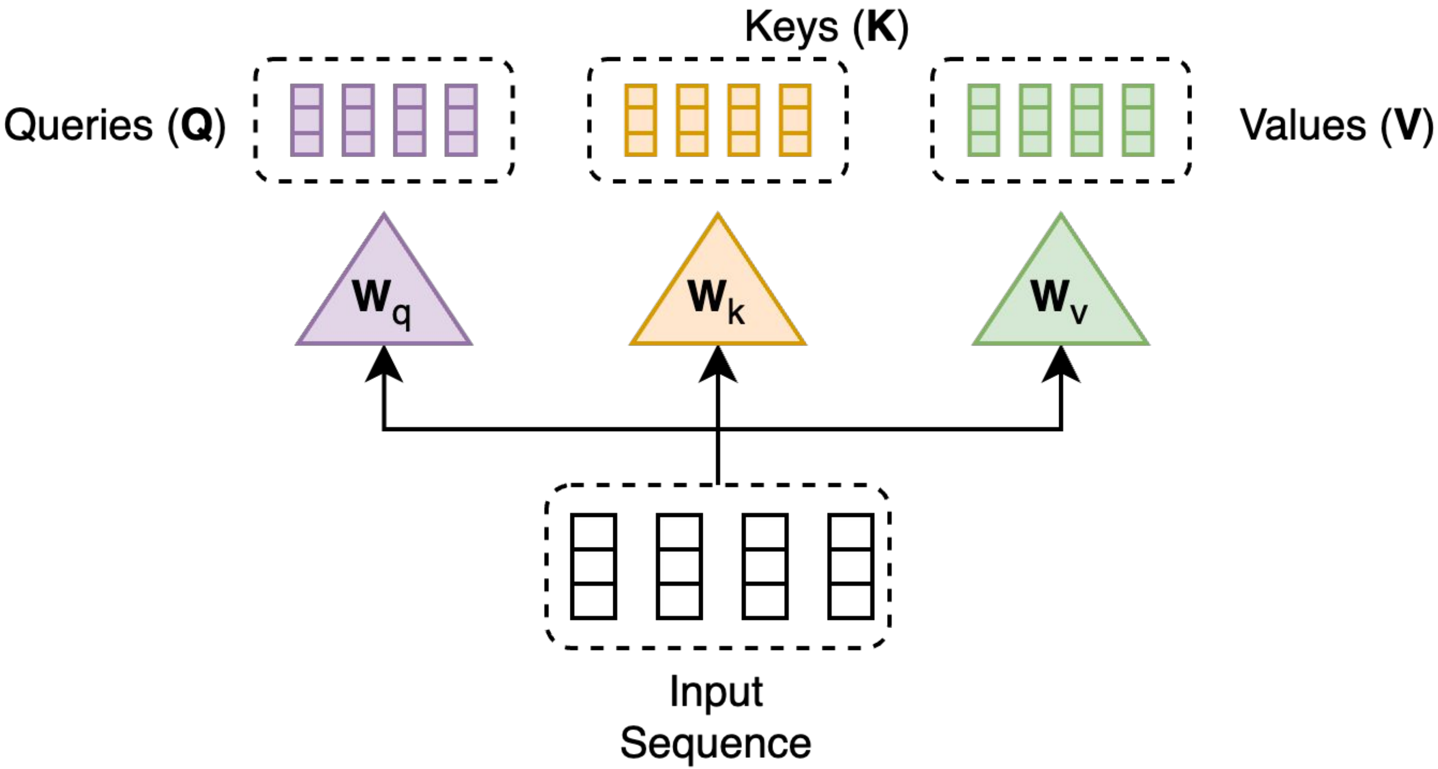
Transformer Architecture

Encoder:
Computes **contextual word embeddings** from context.

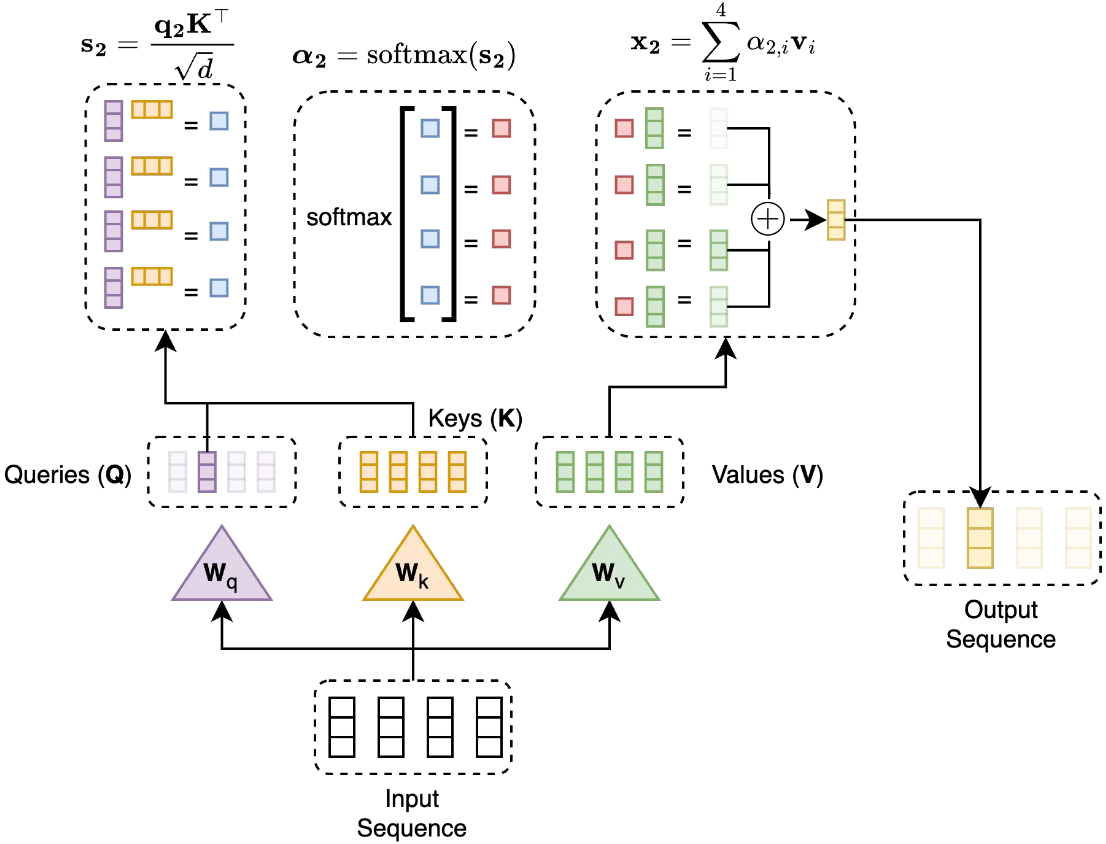


Decoder:
Predicts the next word given context.

Self-Attention



Self-Attention



Discuss:

- Q, K, V are all (n x d) matrices.

- What is the shape of QK^T?
  - What does this matrix represent?
- What is the shape of the final output?
  - What does this matrix represent?

$$Attention(Q, K, V) = Softmax(\frac{QK^T}{d_k})V$$

Multi-Head Attention

What if I want to pay attention to different things at the same time!?

Content-based	This is my big red dog, Clifford.
Description-based	This is my big red dog, Clifford.
Reference-based	This is my big red dog, Clifford.

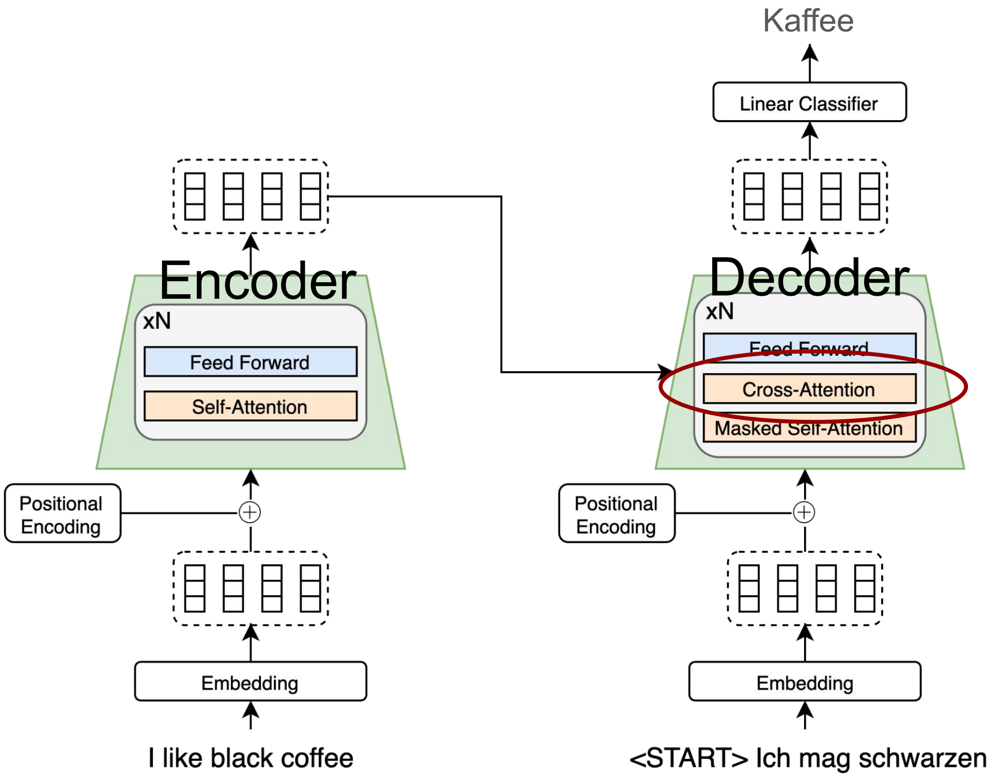
What’s useful depends on the task. How do I pick what to do?

Multi-Head Attention

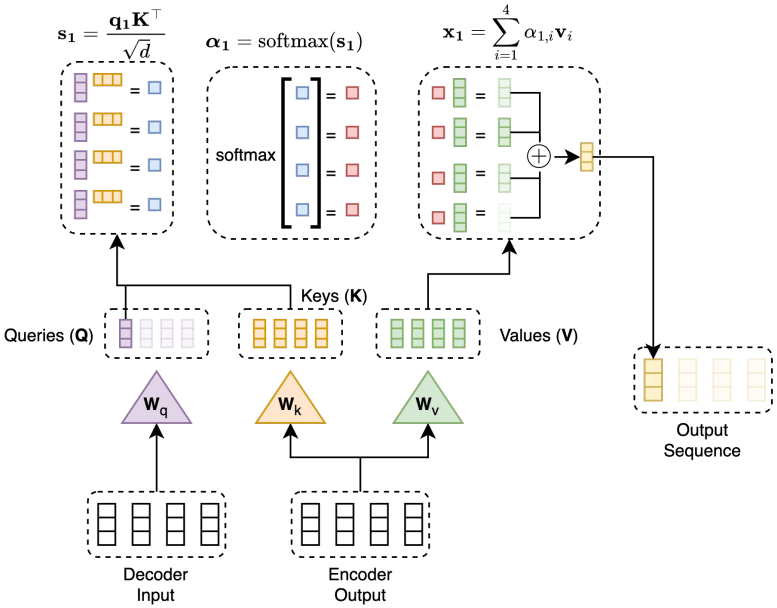
- The Scaled Dot-Product Attention attends to one or few entries in the input key-value pairs.
- Idea: apply Scaled Dot-Product Attention multiple times on the linearly transformed inputs.

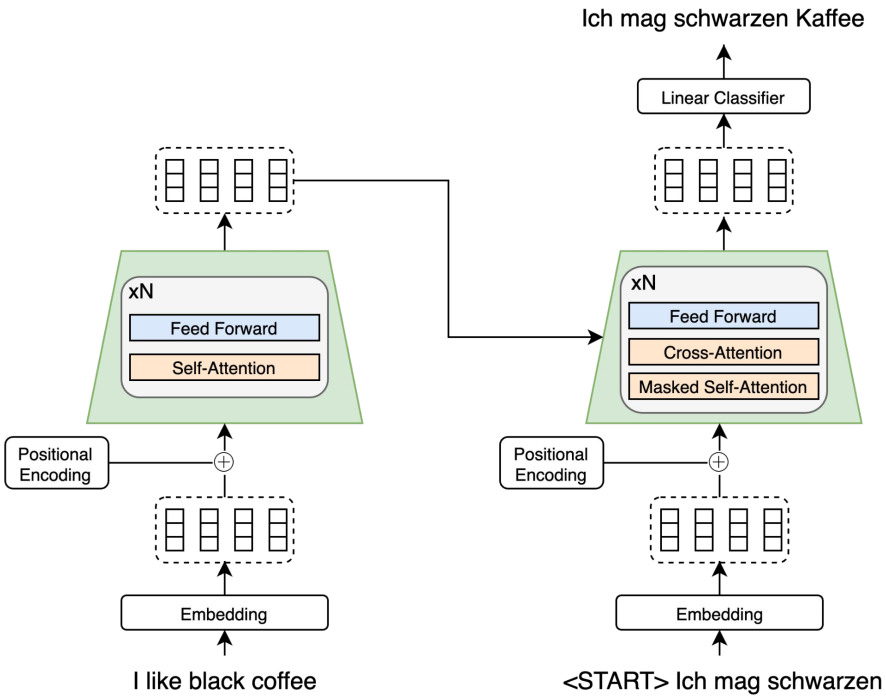
$$\begin{aligned}\mathrm{MultiHead}(Q, K, V) &= \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O \\ \text{where } \mathrm{head_i} &= \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)\end{aligned}$$

Transformer Architecture

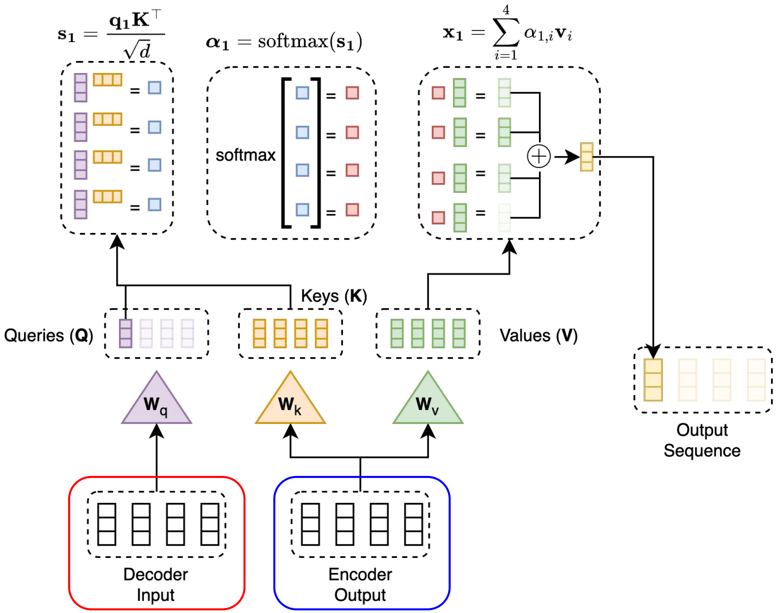


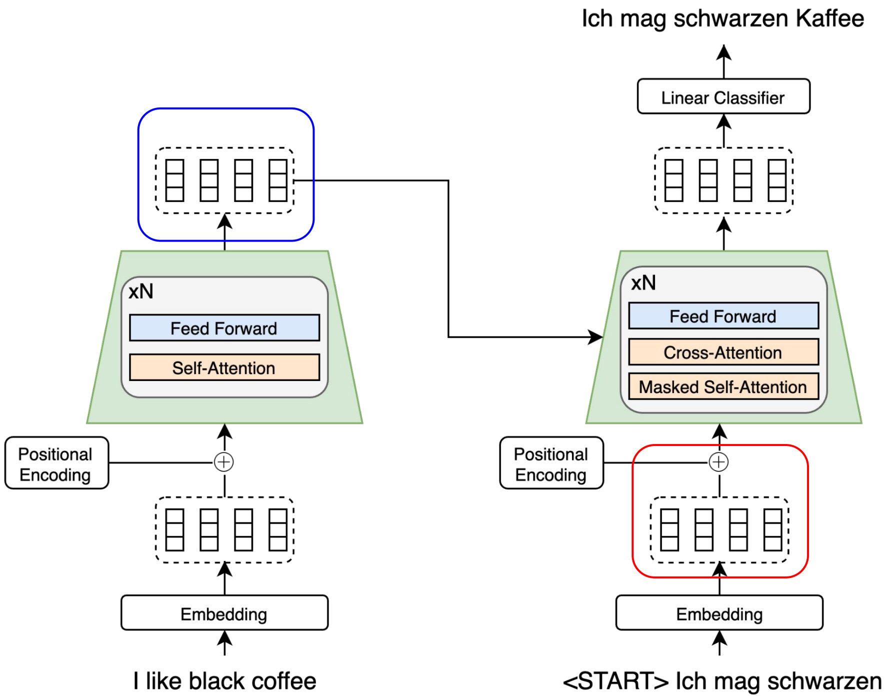
Cross-Attention



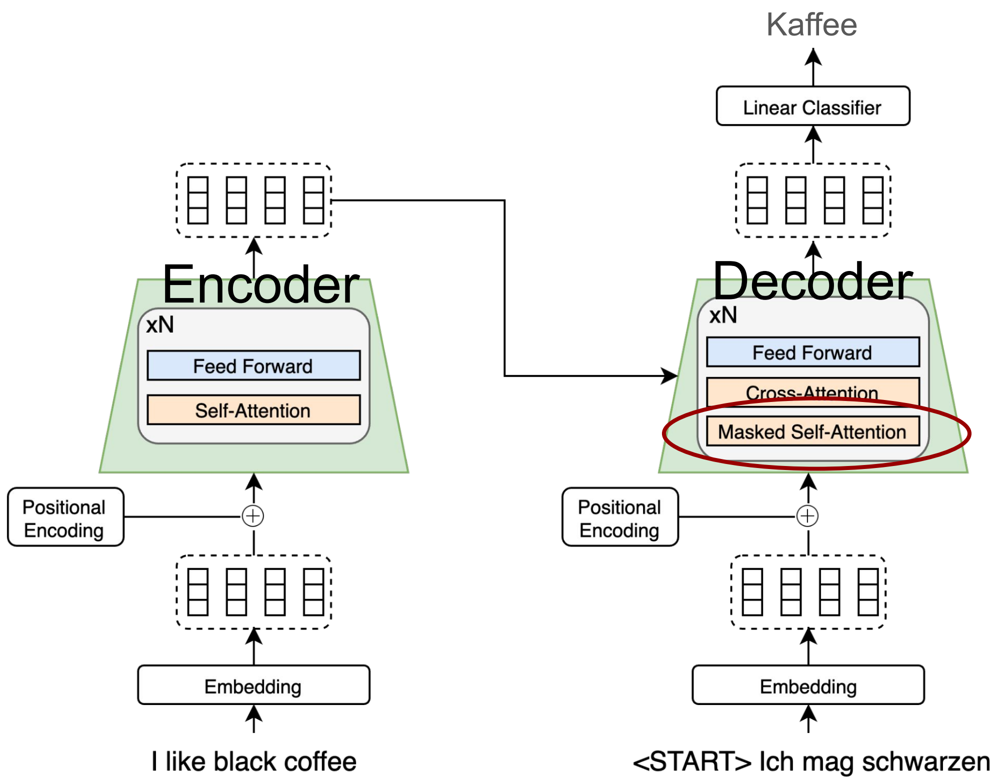


Cross-Attention





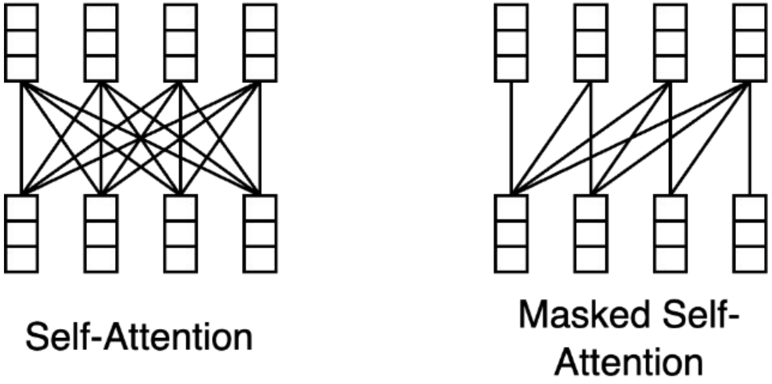
Transformer Architecture



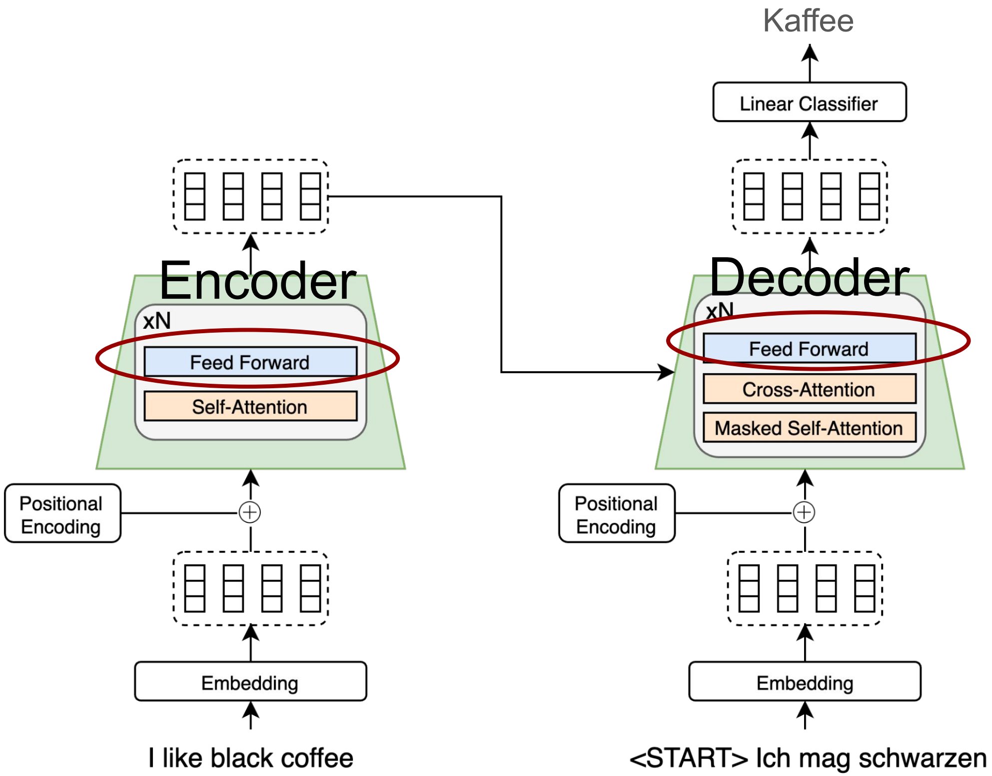
Self-Attention vs. Masked Self-Attention

In masked self-attention each word only pays attention to words from the past, but not the future.

Quiz: Why is this advantageous in the decoder?



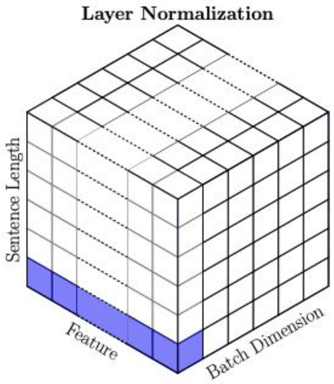
Transformer Architecture



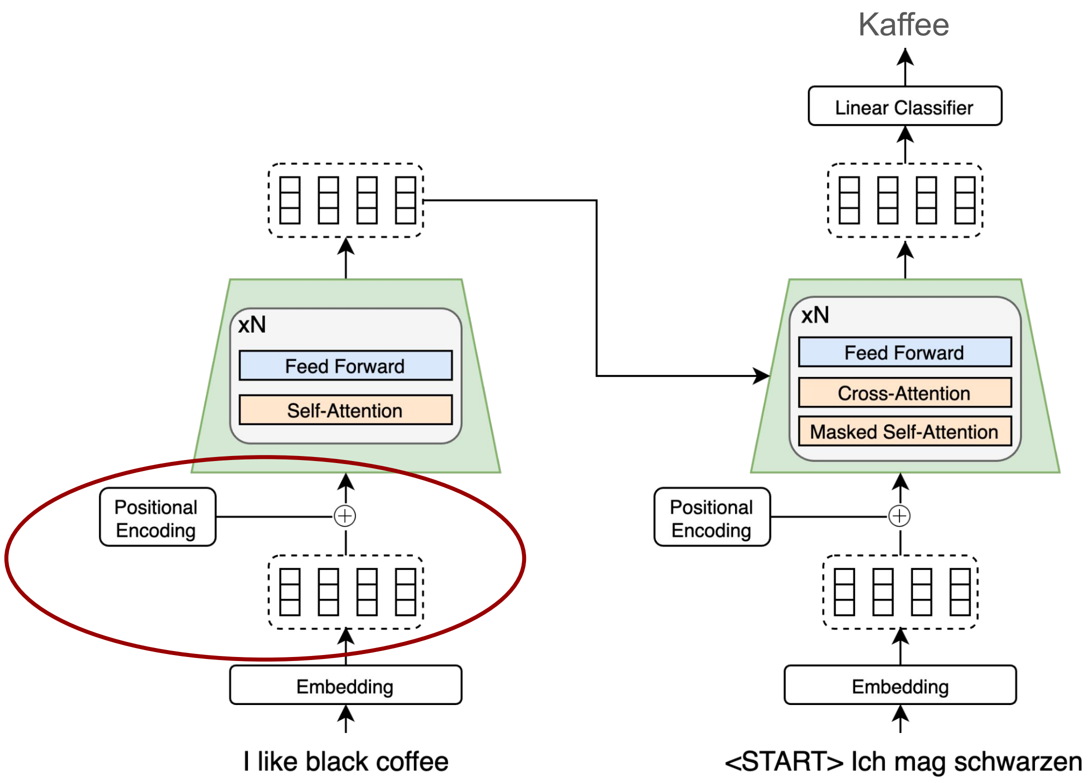
Cornell Bowers C·IS

Point-wise Feed-forward Networks

- Purpose
  - Applies non-linear transformations to the output of the attention layer
- Equation
  - $FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2$
    - where W and b are learned weights and biases
- These FFN is applied separately to each position
- Followed by Layer Normalization

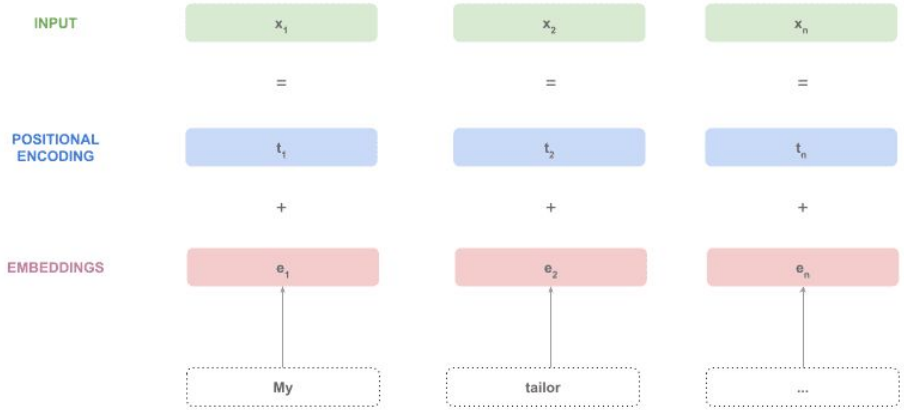


Transformer Architecture



Input Embeddings

- Replace tokens with continuous vectors
- Made up of two components:
  - token embeddings
  - positional encoding depends on position and dimension index as follows (next slide)
- Word order is important:
  - “Sally stole money from John”
  - “John stole money from Sally”



Positional Embedding

Encodes the position of each token in the sentence.

Token Position

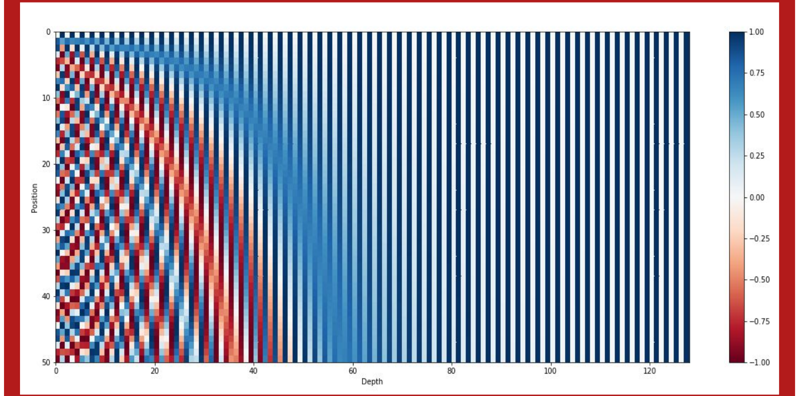
Integer	0	1	2	3	4	5	6	7
Positional encoding (binary)	0	0	0	0	1	1	1	1
	0	0	1	1	0	0	1	1
	0	1	0	1	0	1	0	1

Positional Embeddings

Token Position

Integer	0	1	2	3	4	5	6	7
Positional encoding	0.5	0.3	0.3	0.5	0.7	0.9	0.9	0.7
	0.5	0.5	0.7	0.7	0.5	0.5	0.7	0.7
	0	1	0	1	0	1	0	1

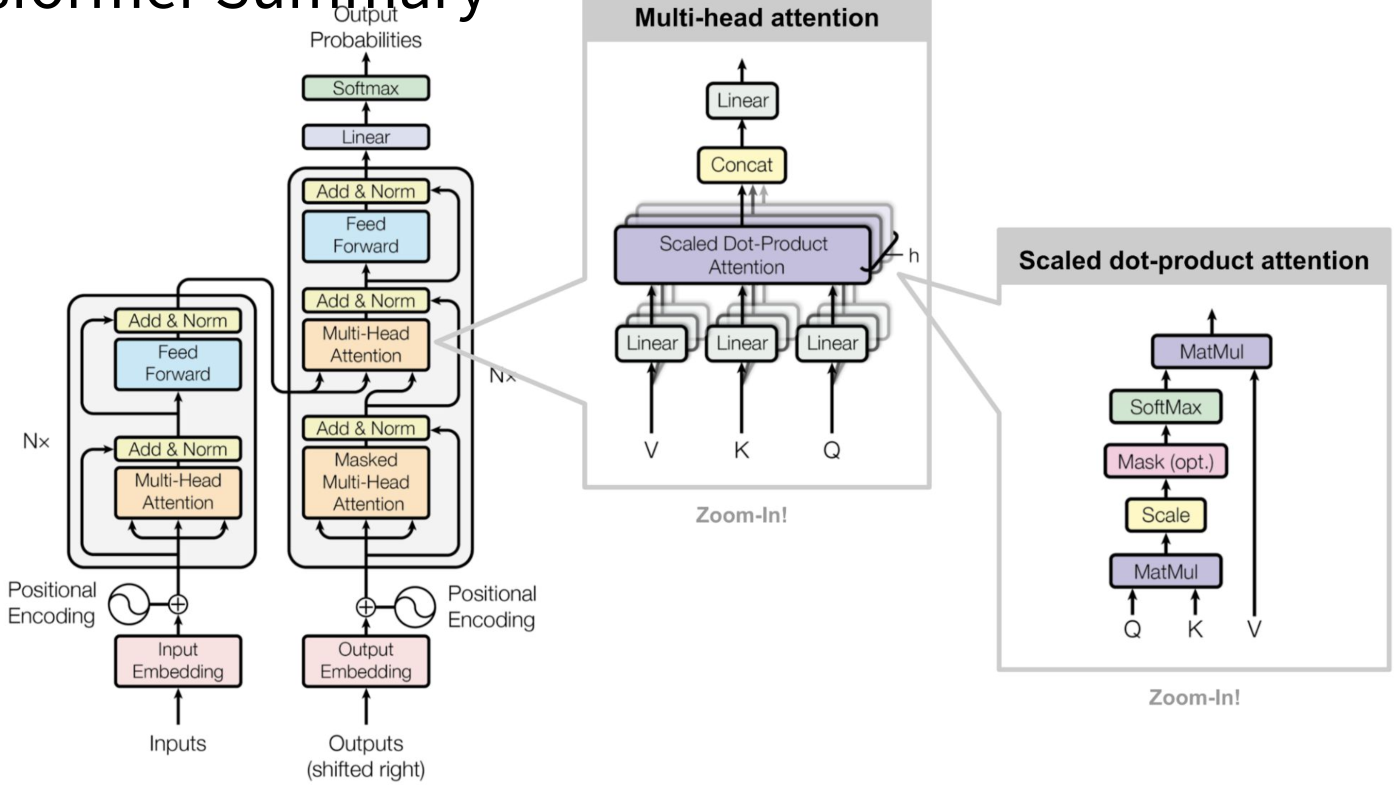
Positional Embeddings



$$PE_{(pos,2i)} = sin(\frac{pos}{10000^{2i/d_{\text{model}}}})$$

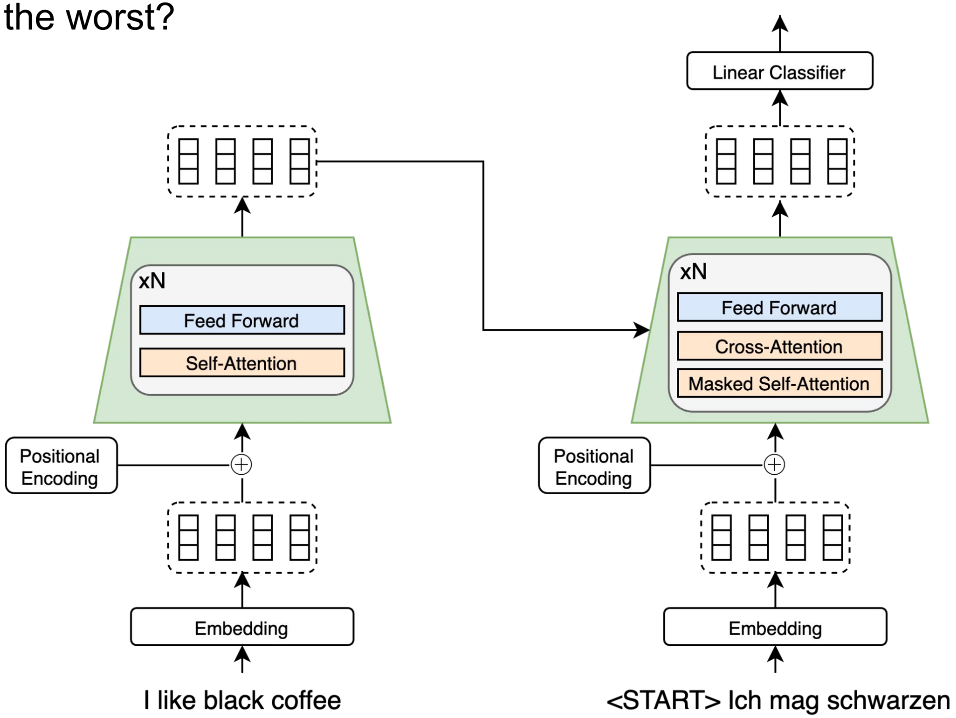
$$PE_{(pos,2i+1)} = cos(\frac{pos}{10000^{2i/d_{\text{model}}}})$$

Transformer Summary



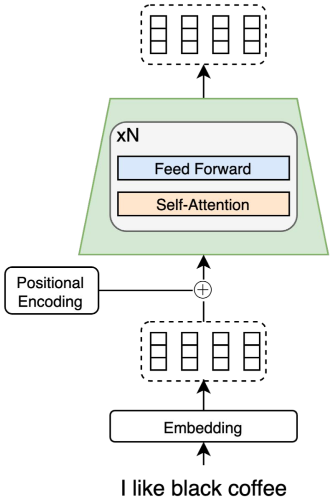
Discuss:

- How does the transformer scale with sequence length?
  - Which part of the transformer scales the worst?



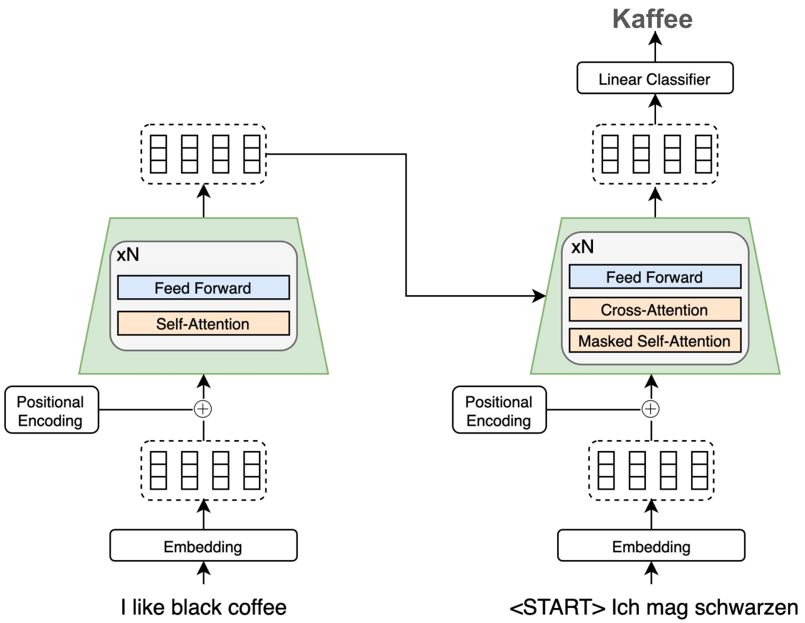
BERT

Encoder only



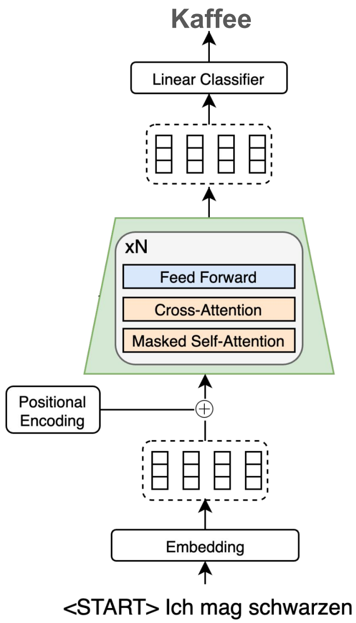
T5/BART

Encoder - Decoder



GPT

Decoder only



$$C = C_0 N D$$
$$L = \frac{A}{N^\alpha} + \frac{B}{D^\beta} + L_0$$

Scaling Laws

- Performance improves predictably with increased compute, data, and parameters
  - Can actually fit power laws!
  - Predict performance before training!

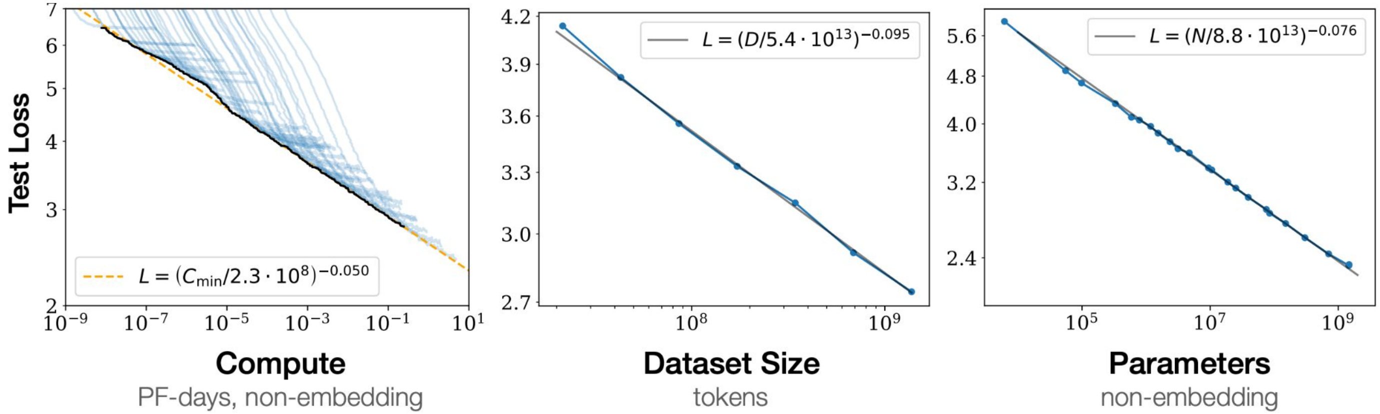
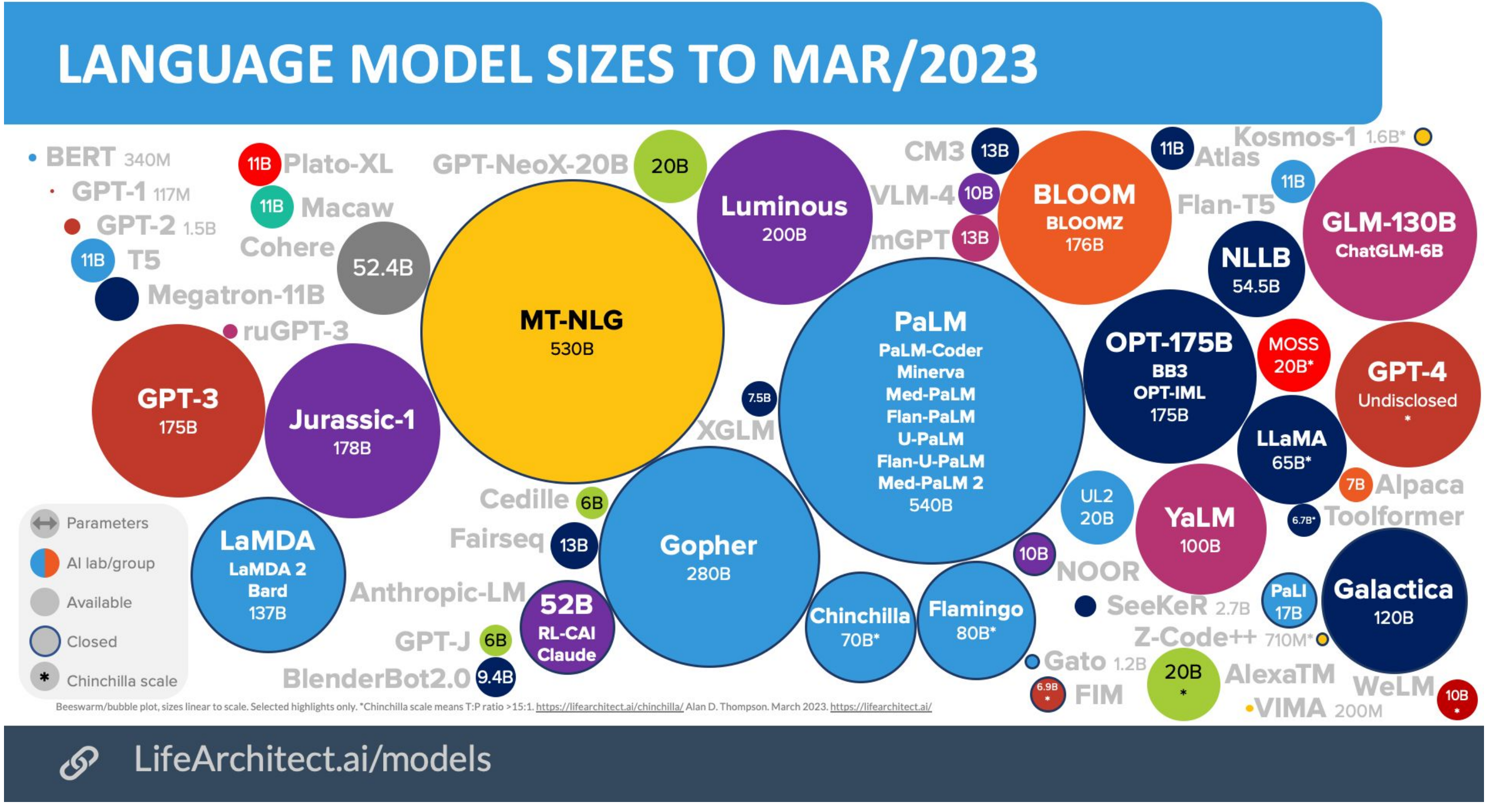


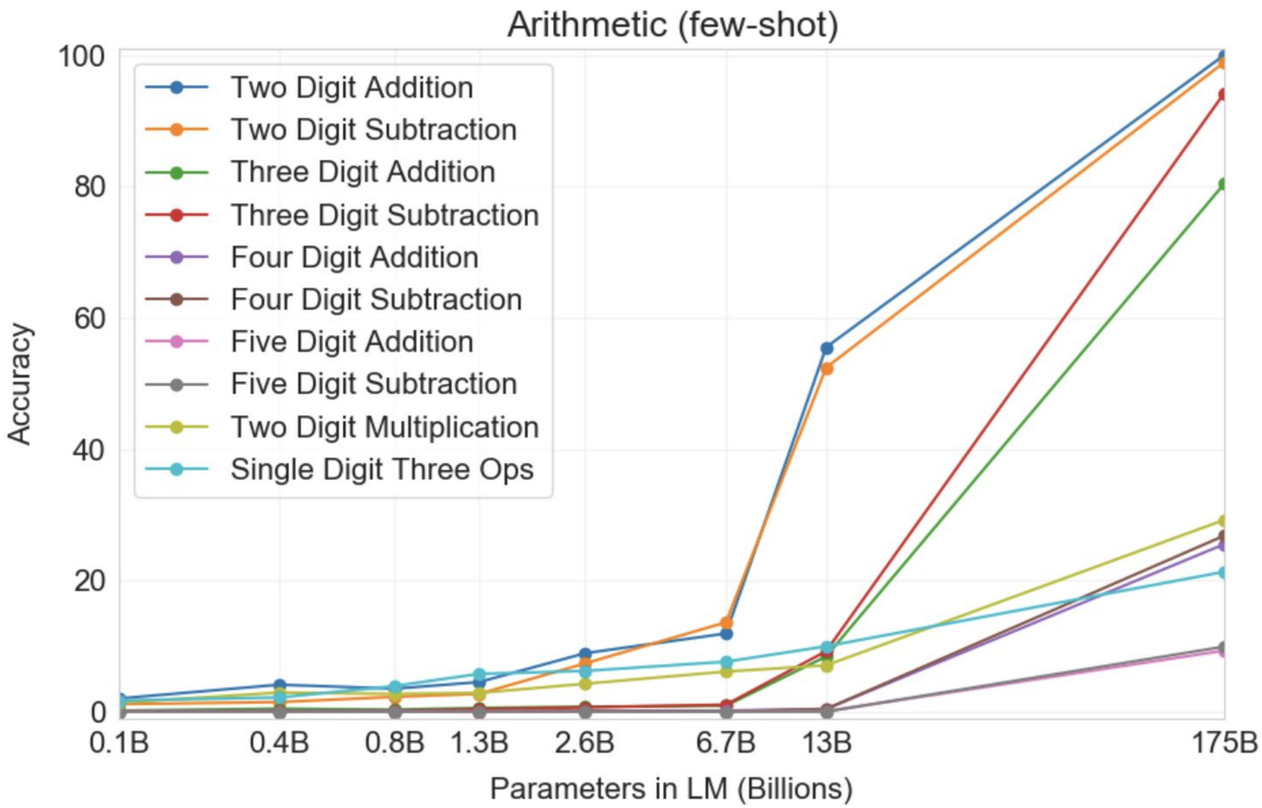
Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

GPT-3



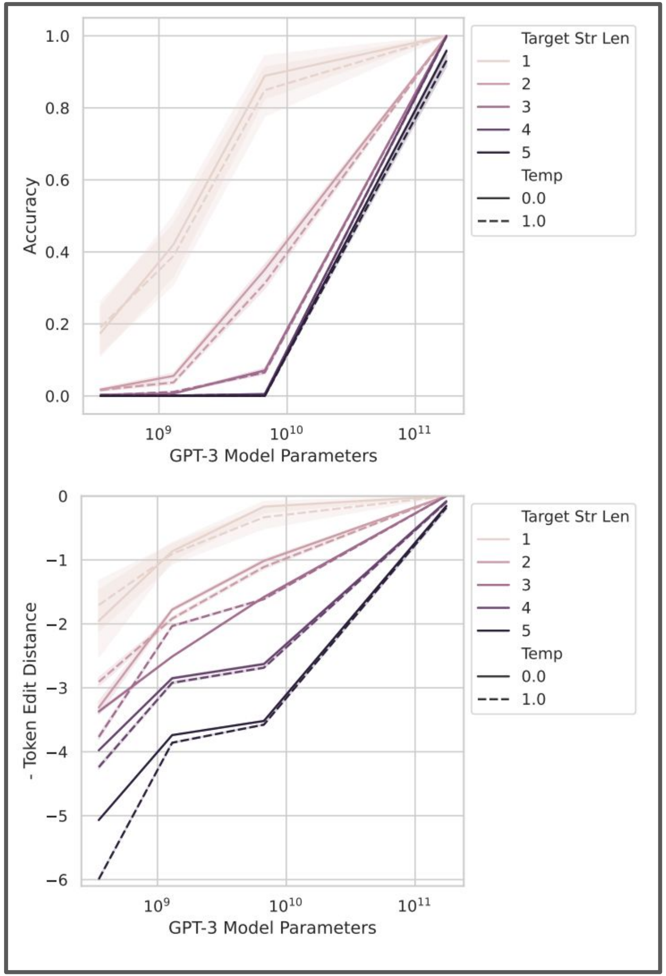
LANGUAGE MODEL SIZES TO MAR/2023
BERT 340M
GPT-1 117M
GPT-2 1.5B
11B T5
Megatron-11B
11B Plato-XL
11B Macaw
Cohere 52.4B
ruGPT-3
GPT-3 175B
Jurassic-1 178B
GPT-NeoX-20B 20B
MT-NLG 530B
Luminous 200B
CM3 13B
VLM-4 10B
mGPT 13B
7.5B XGLM
PaLM
PaLM-Coder
Minerva
Med-PaLM
Flan-PaLM
U-PaLM
Flan-U-PaLM
Med-PaLM 2
540B
BLOOM
BLOOMZ
176B
11B Atlas
Kosmos-1 1.6B*
11B Flan-T5
GLM-130B
ChatGLM-6B
NLLB 54.5B
OPT-175B
BB3
OPT-IML
175B
MOSS 20B*
GPT-4 Undisclosed *
LLaMA 65B*
7B Alpaca
6.7B* Toolformer
UL2 20B
YaLM 100B
10B NOOR
SeeKeR 2.7B
PaLI 17B
Galactica 120B
Z-Code++ 710M*
Gato 1.2B
6.9B * FIM
20B * AlexaTM
WeLM 10B *
VIMA 200M
Parameters
AI lab/group
Available
Closed
Chinchilla scale
LaMDA
LaMDA 2
Bard
137B
Cedille 6B
Fairseq 13B
Gopher 280B
Anthropic-LM
52B RL-CAI Claude
GPT-J 6B
BlenderBot2.0 9.4B
Chinchilla 70B*
Flamingo 80B*
Beeswarm/bubble plot, sizes linear to scale. Selected highlights only. *Chinchilla scale means T:P ratio >15:1. https://lifearchitect.ai/chinchilla/ Alan D. Thompson. March 2023. https://lifearchitect.ai/
LifeArchitect.ai/models

Emergent Capabilities



Is Emergence a Mirage?

- Look at four digit addition under different metrics
  - Emergence can be an artefact of the evaluation metric
- Addition capabilities improve smoothly when using a more granular metric
  - Exact match accuracy -> Token edit distance



Schaeffer, Rylan, Brando Miranda, and Sanmi Koyejo. "Are emergent abilities of large language models a mirage?." Advances in Neural Information Processing Systems 36 (2024).