Reminders

- PRA2 due 2/8
- Course project progress report 1 due 2/27
- Come to OH for course project discussion!

Artificial Intelligence Methods for Social Good

Lecture 8

Basics of Natural Language Processing

17-537 (9-unit) and 17-737 (12-unit)

Instructor: Fei Fang

feifang@cmu.edu

Outline

- Sentiment Analysis
- Topic Modeling
- Transformer → BERT → GPT → ChatGPT
- Revisit
  - Social Bot Detection
  - Food Rescue Difficulty Prediction
- Discussion

Sentiment Analysis

- Analyze the sentiment of given text manually is easy:
  - "Great service!"
  - "The food was terrible!"
  - "It is Tuesday today."
- How to do it automatically?

Sentiment Analysis

- A simple approach:
  - Manually construct a list of "positive" and "negative" words, give each word a polarity score in $[-1,1]$
  - Given the text to be analyzed, extract the words and average their polarity score

"Great service!"
"The food was terrible!"
"It is Tuesday today."

Sentiment Analysis

- A simple learning-based approach if data is available:
  - Data: labeled with "positive", "negative"
  - Represent given text as a feature vector
    - E.g., Use bag-of-words with tf-idf
  - Train a classifier
    - E.g., decision tree

TF-IDF (term frequency–inverse document frequency)

- Importance of a word t in document D within corpus D
  - $\text{tfidf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D)$
  - $\text{tf}(t, d) = \dfrac{\#\text{word } t \text{ in document } d}{\#\text{words in document } d}$
  - $idf(t, D) = \log \dfrac{\#\text{documents in } D}{\#\text{documents in } D \text{ that contains word } t}$

```
>>> from sklearn.feature_extraction.text import TfidfVectorizer
>>> corpus = [
...     'This is the first document.',
...     'This document is the second document.',
...     'And this is the third one.',
...     'Is this the first document?',
... ]
>>> vectorizer = TfidfVectorizer()
>>> X = vectorizer.fit_transform(corpus)
>>> vectorizer.get_feature_names_out()
array(['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third',
       'this'], ...)
>>> print(X.shape)
(4, 9)
```


https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

Sentiment Analysis: Example Code

```
from textblob import TextBlob

# Components of an article on AI for social good
article_title = "Leveraging AI for Social Good: A New Frontier"

# Function to analyze sentiment
def analyze_sentiment(text):
    blob = TextBlob(text)
    # sentiment polarity ranges from -1 (very negative) to 1 (very positive)
    return blob.sentiment.polarity

# Analyzing sentiment of each component
title_sentiment = analyze_sentiment(article_title)

# Printing the sentiment scores
print(f"Sentiment Polarity of Article Title: {title_sentiment}")
```

```
Sentiment Polarity of Article Title: 0.2898989898989899
```


https://github.com/sloria/TextBlob/blob/dev/textblob/en/sentiments.py

Outline

- Sentiment Analysis
- Topic Modeling
- Transformer → BERT → GPT → ChatGPT
- Revisit
  - Social Bot Detection
  - Food Rescue Difficulty Prediction
- Discussion

Topic Modeling

- An unsupervised learning task: Given a collection of documents, discover "topics" among them
  - A document can be part of multiple topics

LOCAL NEWS ›

Carnegie Mellon University hit by cyberattack possibly impacting more than 7,000 people

KDKA NEWS

By **Michael Guise**
January 19, 2024 / 6:25 PM EST / CBS Pittsburgh

January 11, 2024

All Seven CMU Colleges Send Iris to Space

Interdisciplinary effort across Carnegie Mellon University crucial to student-led space project

Tech

Millions of hacked toothbrushes used in Swiss cyber attack, report says

Army of infected devices reportedly caused millions of euros of damage

Anthony Cuthbertson • 14 hours ago • Comments

Topic Modeling

- Intuition: If a document is under topic "Wildlife", which one of the two words will be in the document with a higher probability, "Tiger" or "Christmas"?

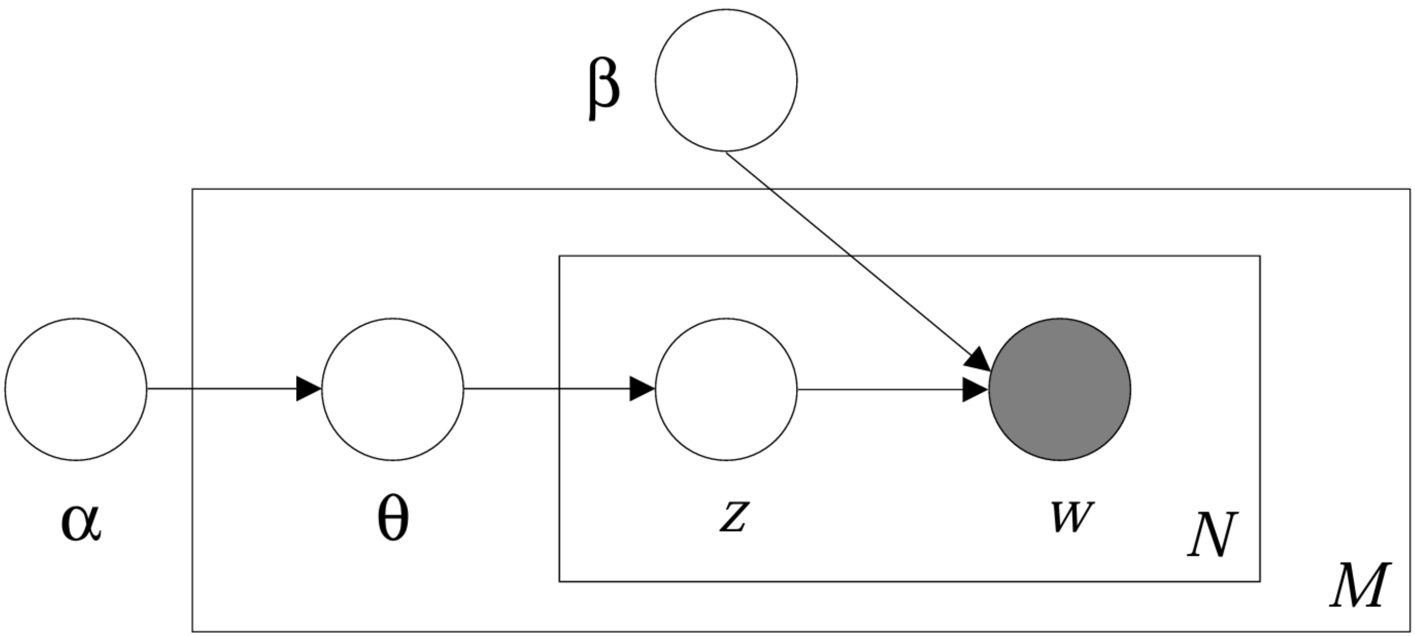
	Wildlife	Football	Cyber Security
Tiger	0.3	0.01	0.01
Christmas	0.01	0.1	0.07
Movie	0.1	0.17	0.2
Loss	0.05	0.2	0.3
Win	0.02	0.3	0.08

Latent Dirichlet Allocation (LDA)

- Main idea:
  - Build a parametric model of how the words in the documents are generated, with "topics" represented in the model
  - Find the parameter values that best fit the data

Latent Dirichlet Allocation (LDA)

- The model (with parameters α and β):
  - Sample a distribution over topics (θ)
  - Sample a topic (z)
  - Generate a word (w) based on z according to β



Blei, D.M., Ng, A.Y. and Jordan, M.I., 2003. Latent dirichlet allocation. *Journal of machine Learning research*, *3*(Jan), pp.993-1022.

Topic Modeling: Example Code

```python
from sklearn.decomposition import LatentDirichletAllocation
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np

# Placeholder corpus: list of documents
corpus = [
    "Document 1 text ...",
    "Document 2 text ...",
    # Add more documents as needed, e.g., WHS-CORP and INFRACORP dataset
]

# Preprocess & vectorize the text data
vectorizer = TfidfVectorizer(stop_words='english')
X = vectorizer.fit_transform(corpus)

# Train the LDA model
lda = LatentDirichletAllocation(n_components=50, random_state=0)
lda.fit(X)

# Extracting the topic distribution for each document
doc_topic_distributions = lda.transform(X)

# Displaying the topic distribution vector for each document
for doc_idx, topic_distribution in enumerate(doc_topic_distributions):
    print(f"Document #{doc_idx} Topic Distribution:")
    print(topic_distribution)
```

Outline

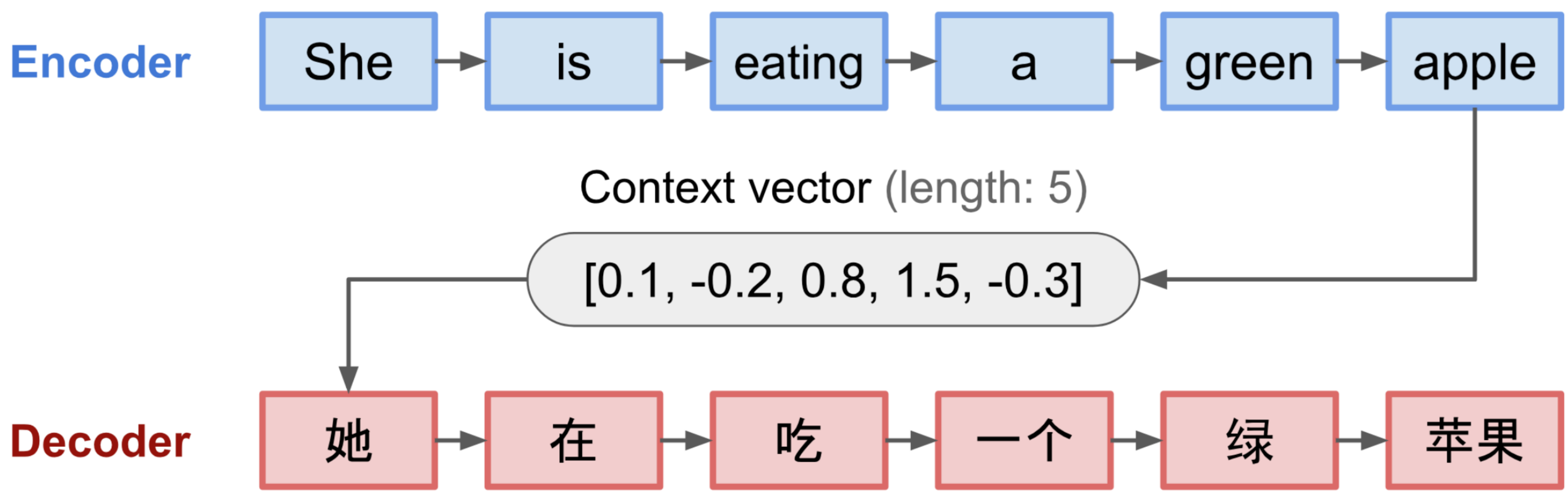
- Sentiment Analysis
- Topic Modeling
- Transformer → BERT → GPT → ChatGPT

- Revisit
  - Social Bot Detection
  - Food Rescue Difficulty Prediction

- Discussion

Sequence to Sequence (Seq2seq) Model

- Transform an input sequence (source) to a new one (target) and both sequences can be of arbitrary lengths
- Usually use encoder-decoder architecture
- Incapable of remembering long sentences



Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." NeurIPS 2014.

Attention

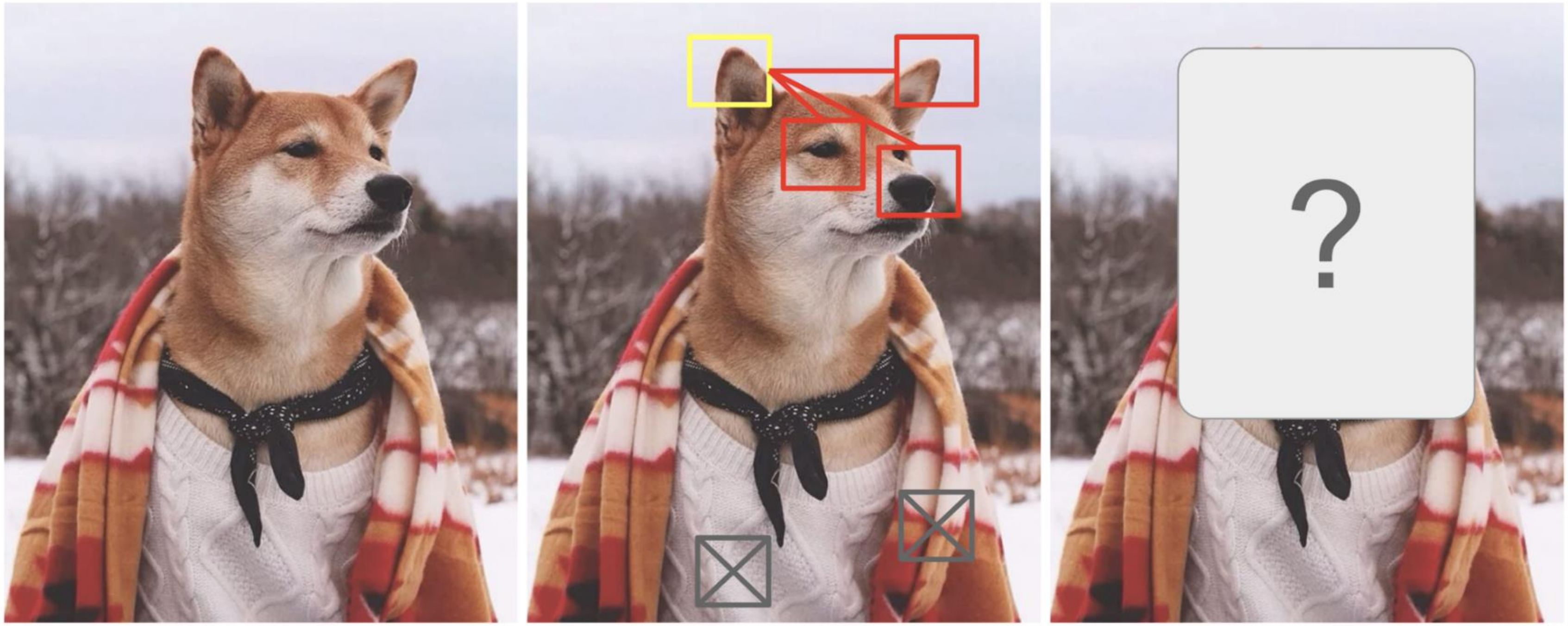
@mensweardog

Look at this picture for 10s.

Tell your neighbor which part of the picture your eyes stayed on for more than 1s

Vaswani et al. Attention is All You Need. 2017
https://lilianweng.github.io/posts/2018-06-24-attention/

Attention



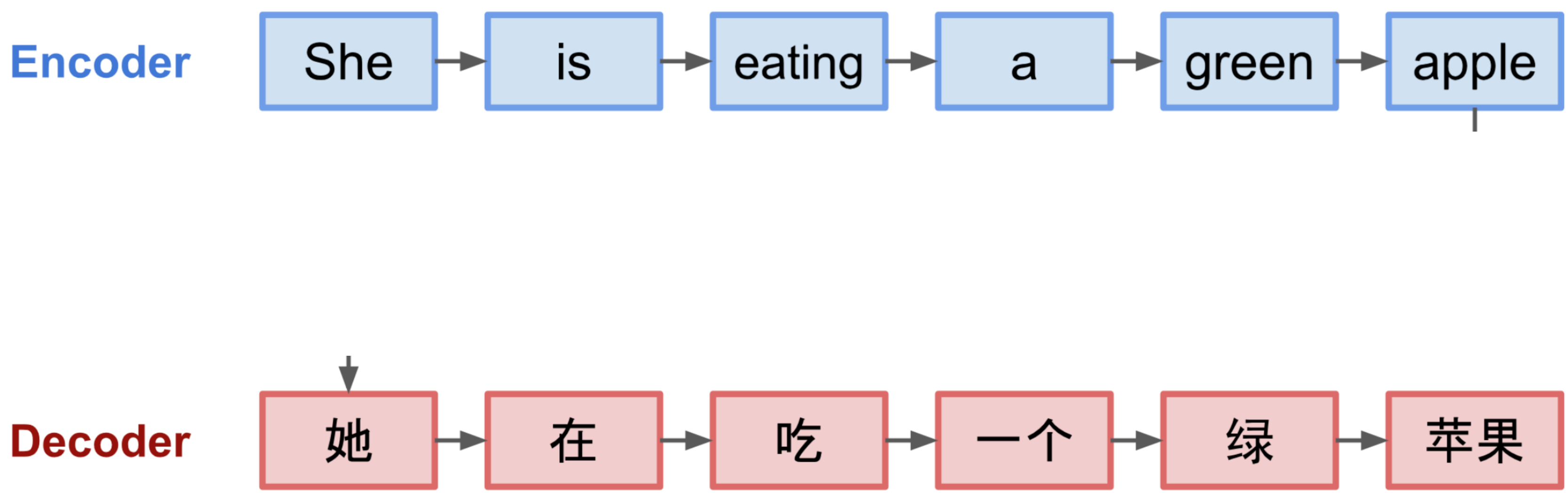
@mensweardog

Attention

Attention can be broadly interpreted as a vector of importance weights

How can Attention Improve over Seq2seq Model?

- Each output token depends on input tokens differently
- Intuitively, calculate the importance weight for each of the source token for current predicting token



How can Attention Improve over Seq2seq Model?

- Say we have a source sequence of length n and try to output a target sequence of length m
  - $x = [x_1, x_2, \dots, x_m], y = [y_1, y_2, \dots, y_m]$
  - Let h_i be the encoder state at the ith position in the source sequence
  - Let $s_t = f(s_{t-1}, y_{t-1}, c_t)$ be the decoder hidden state for the output word at position t
  - Context vector c_t is a weighted sum of h_i

$$c_t = \sum_i \alpha_{t,i} h_i$$

where the importance weight $\alpha_{t,i} = \dfrac{\exp\left(score(s_{t-1}, h_i)\right)}{\sum_{i'} \exp\left(score(s_{t-1}, h_{i'})\right)}$

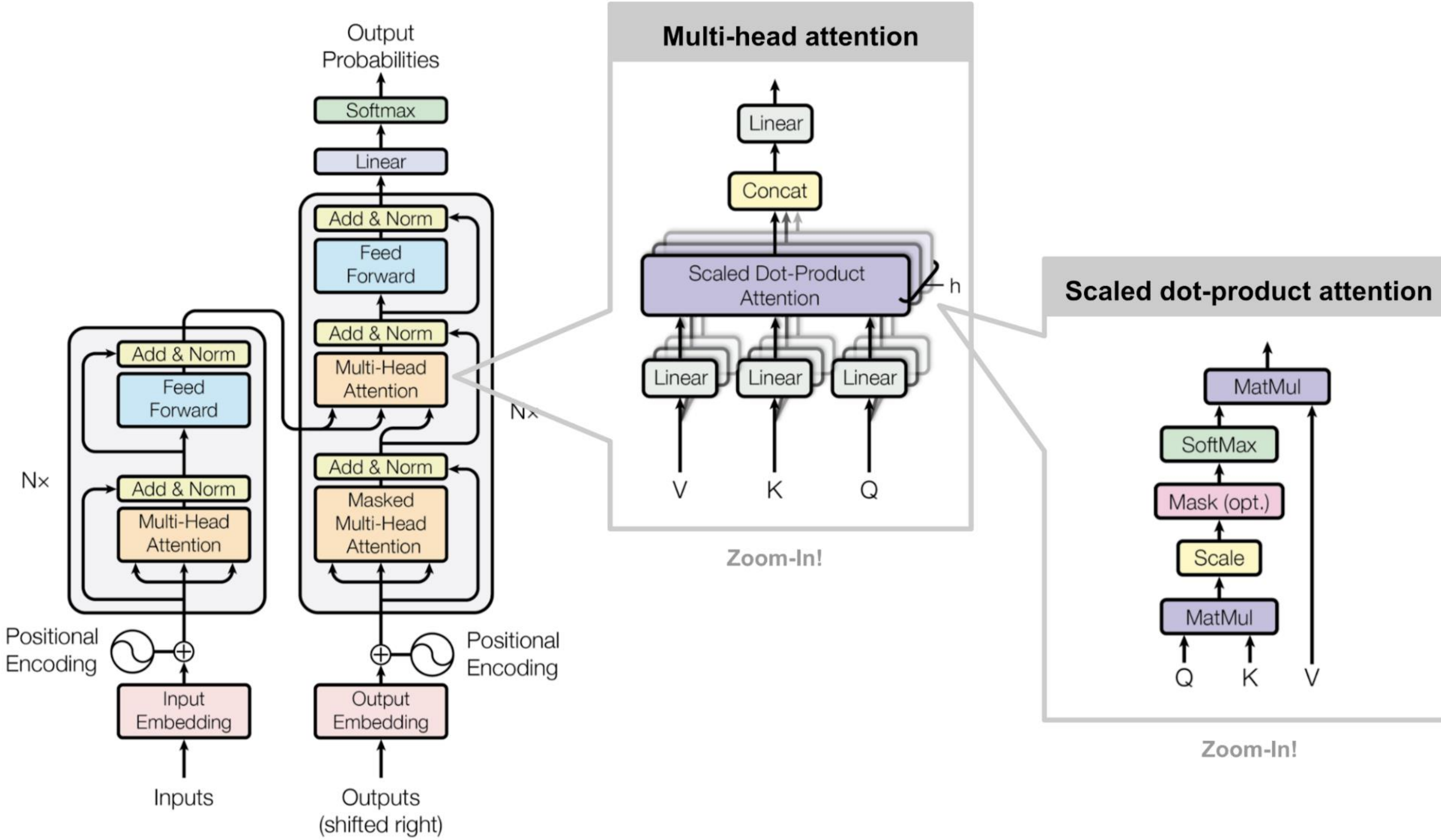
Scaled Dot-Product Attention

$$c_t = \sum_i \alpha_{t,i} h_i, \text{ where } \alpha_{t,i} = \frac{\exp(score(s_{t-1}, h_i))}{\sum_{i'} \exp(score(s_{t-1}, h_{i'}))}$$

- There are many choices of the score function
- Scaled Dot-Product Attention: $score(s_t, h_i) = \frac{s_t^T h_i}{\sqrt{n}}$

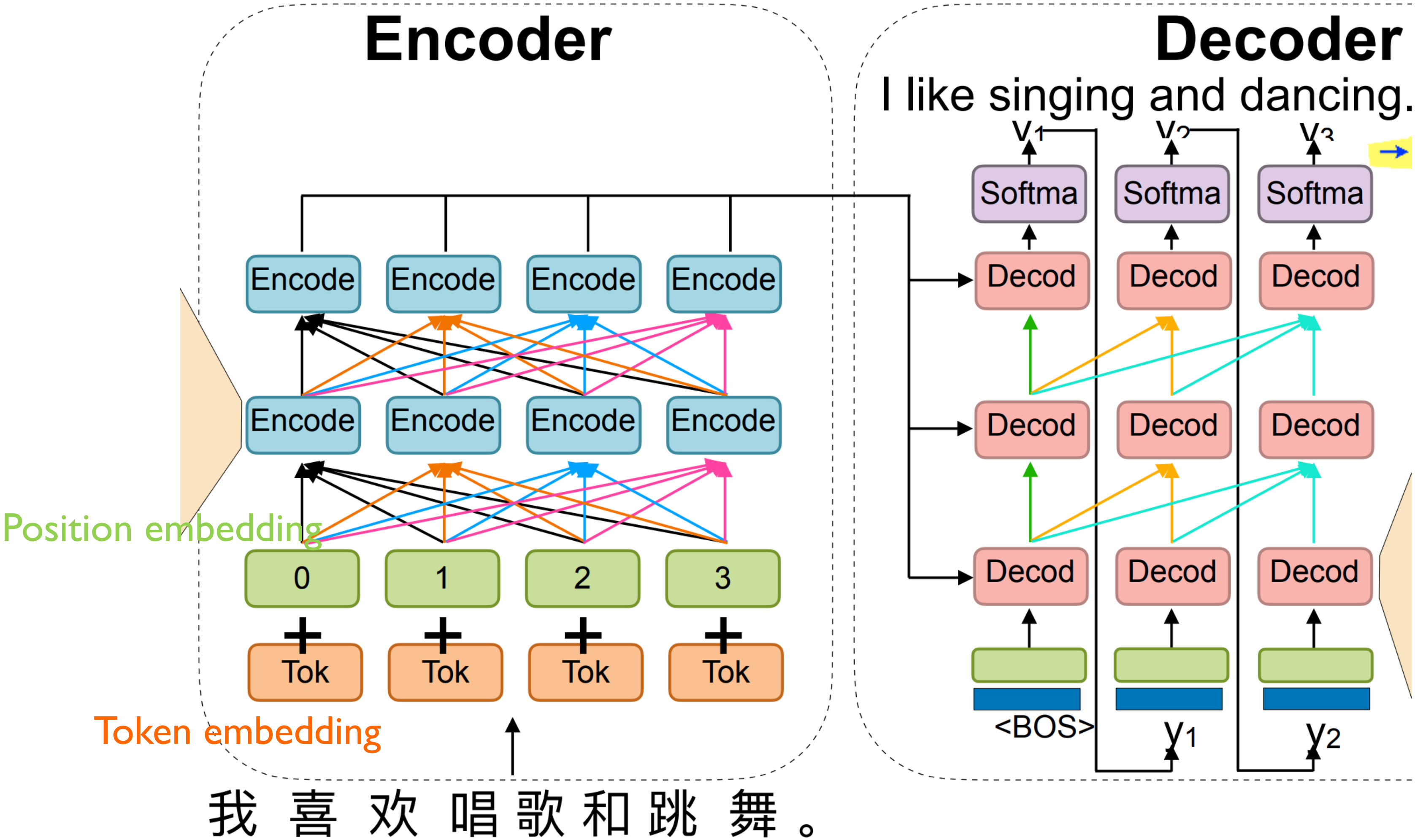
Transformer Architecture Overview

Key: Use Multi-head self-attention



Vaswani et al. Attention is All You Need. 2017

Transformer Architecture Overview



Token Embedding

- A pre-defined fixed set of tokens (vocabulary)
- Each token is represented by a fixed length vector, i.e., token embedding
  - Typically use $d = 512$(base) or $d = 1024$ (large)
- Such embedding (vectors for tokens) is to be learned

abandon	$[0.2, 0.3, 0.5, 0.1]$
abash	$[0.1, 0.4, 0.7, 0.9]$
abate	$[0.3, 0.6, 0.5, 0.3]$
abbreviate	$[0.8, 0.3, 0.9, 0.6]$

Position embedding

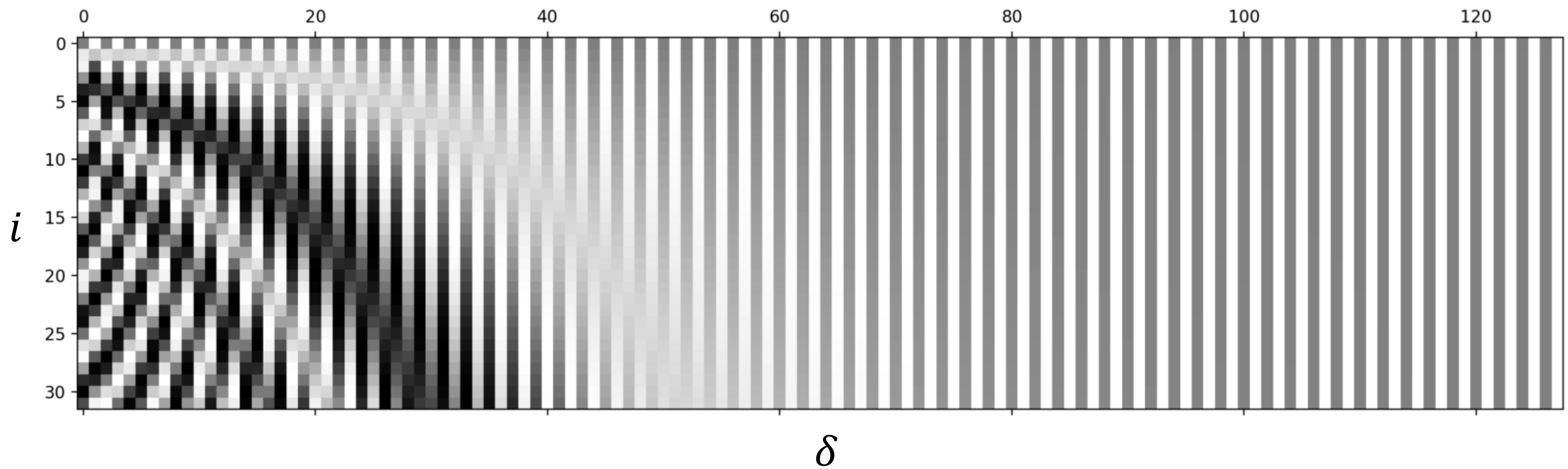
- To distinguish words in different position, map position labels to a vector whose dimension is the same as token embedding (so that they can add up)

- A simple position embedding
  - Given token position $i = 1, \dots, L$ and the dimension $\delta = 1, \dots, d$

$$\mathrm{PE}(i, \delta) = \begin{cases} \sin\left(\frac{i}{10000^{2\delta'/d}}\right) & \text{if } \delta = 2\delta' \\ \cos\left(\frac{i}{10000^{2\delta'/d}}\right) & \text{if } \delta = 2\delta' + 1 \end{cases}$$

Position Embedding

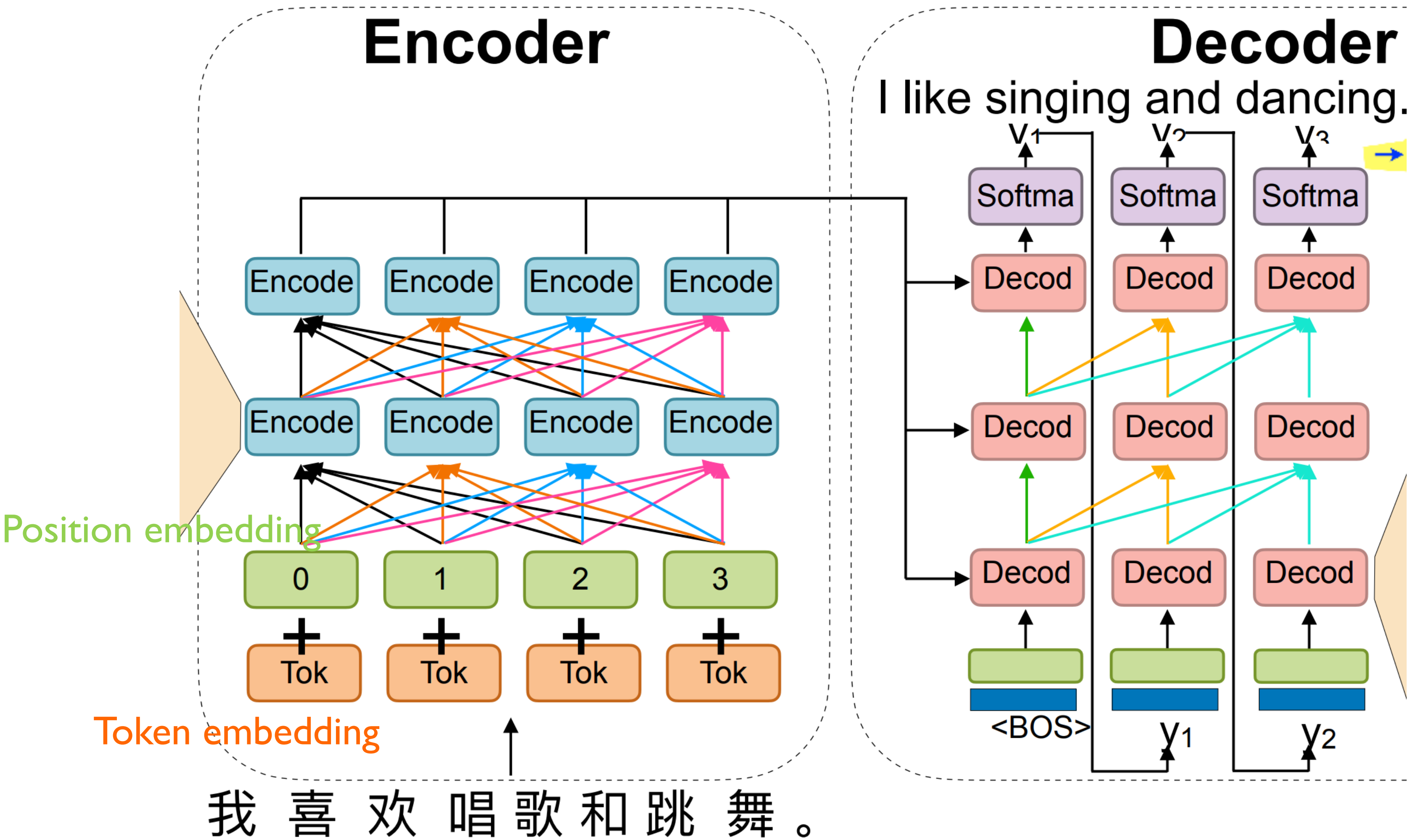
$$\mathrm{PE}(i,\delta) = \begin{cases} \sin\left(\frac{i}{10000^{2\delta'/d}}\right) & \text{if } \delta = 2\delta' \\ \cos\left(\frac{i}{10000^{2\delta'/d}}\right) & \text{if } \delta = 2\delta' + 1 \end{cases}$$



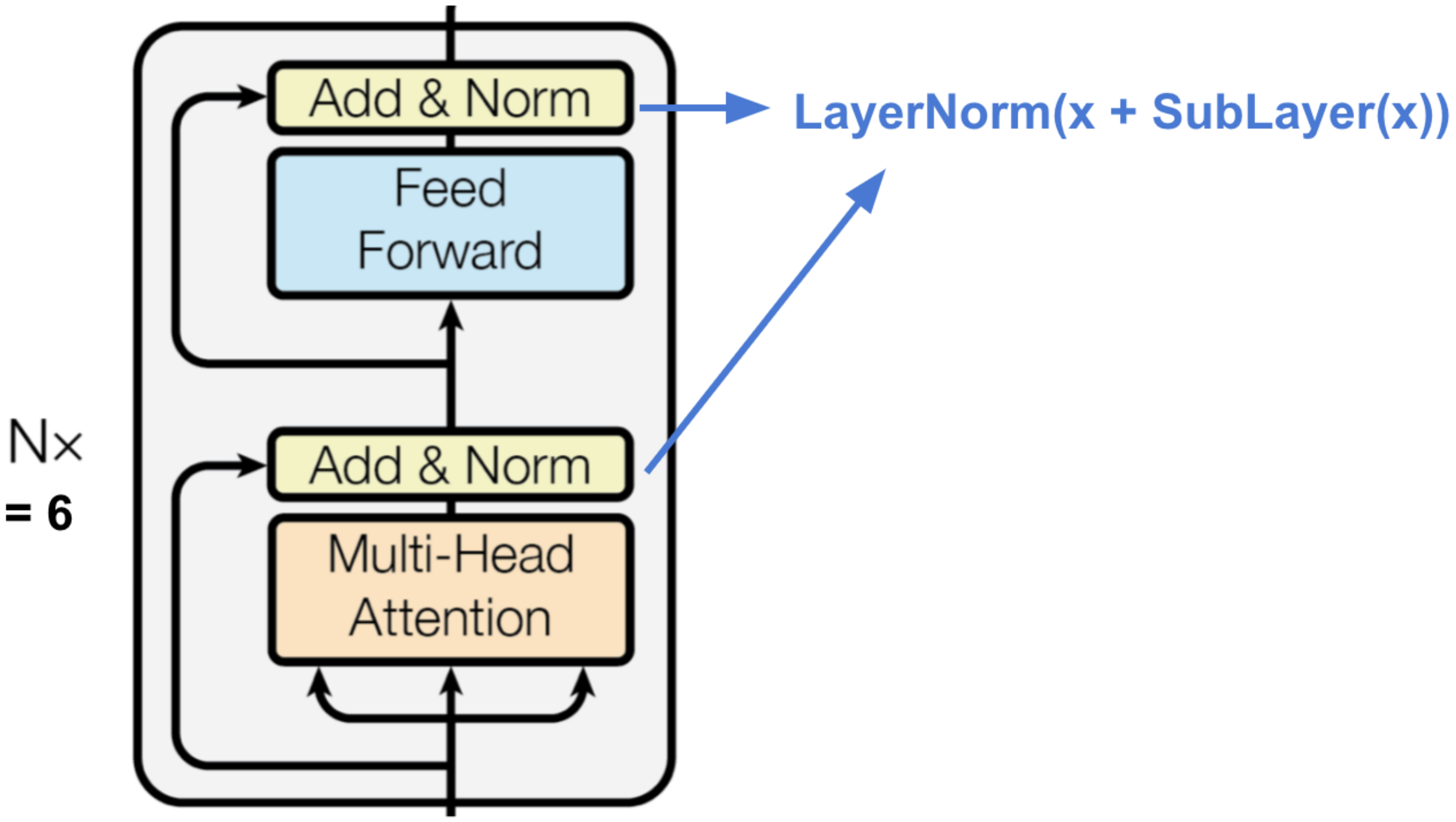
$$L = 32, d = 128$$

https://lilianweng.github.io/posts/2020-04-07-the-transformer-family/

Transformer Architecture Overview

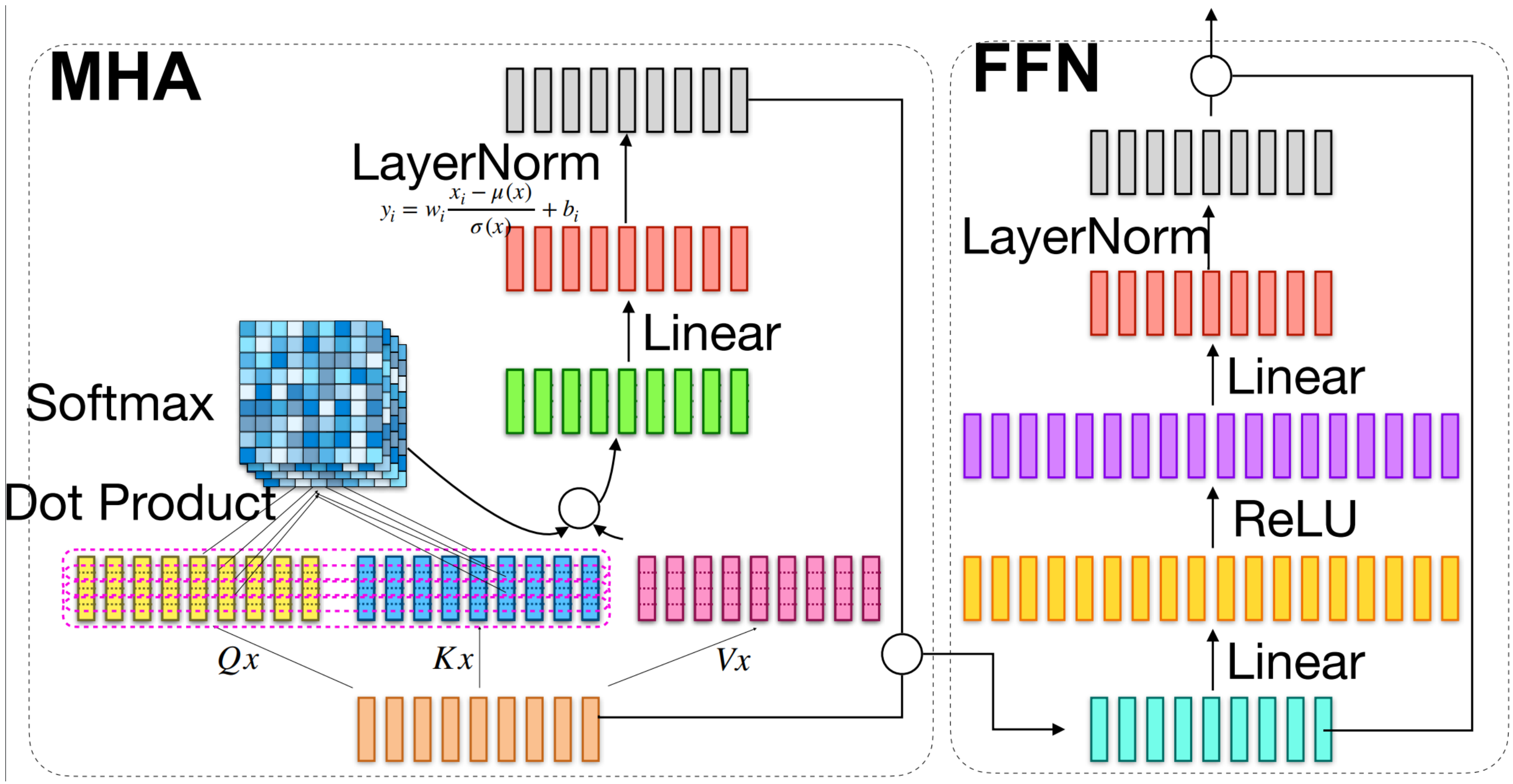


Transformer Encoder



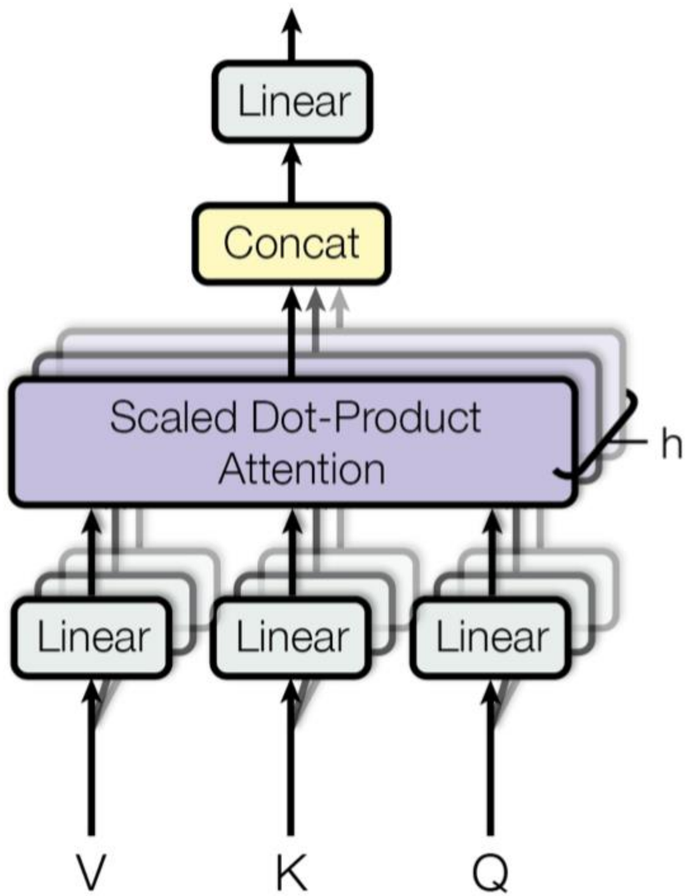
Vaswani et al. Attention is All You Need. 2017

Transformer Encoder



Transformer

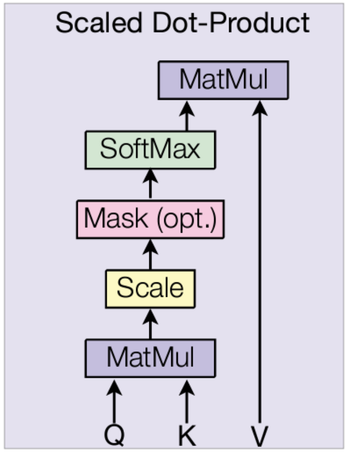
- Multi-head self-attention

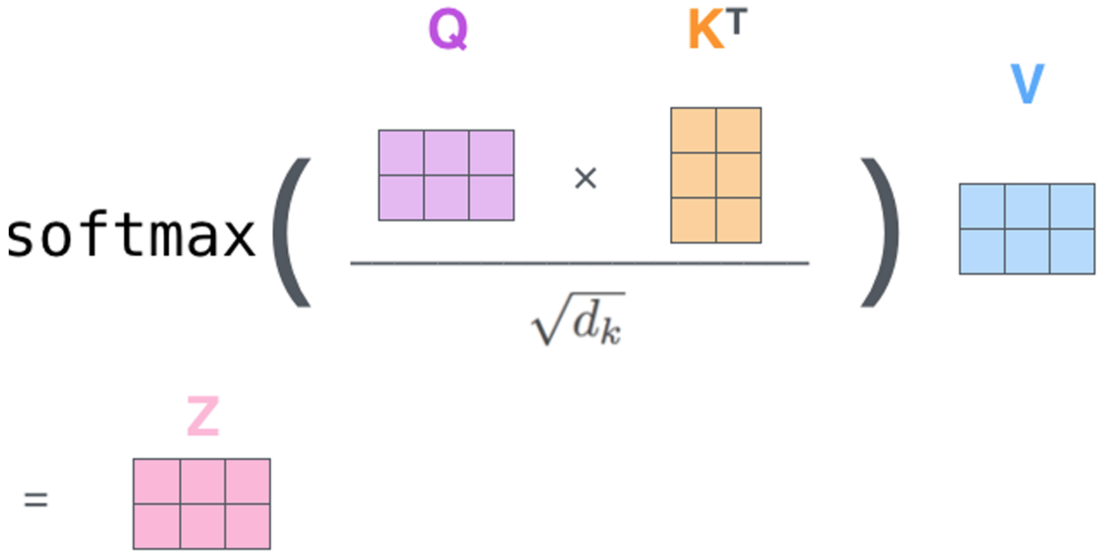


 Vaswani et al. Attention is All You Need. 2017

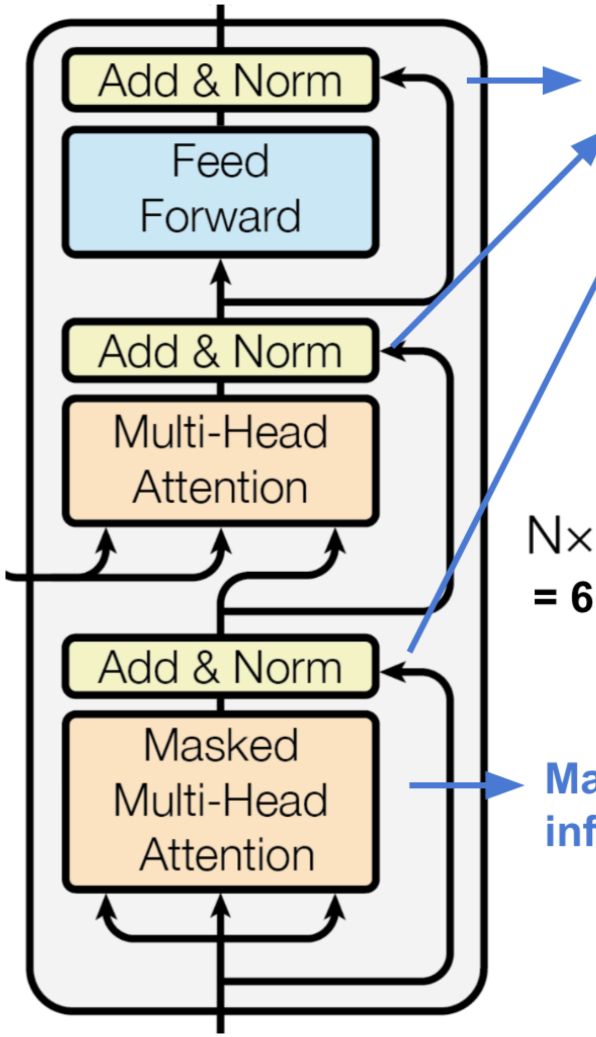
Scaled Dot-Product Attention in Transformer

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





Transformer Decoder



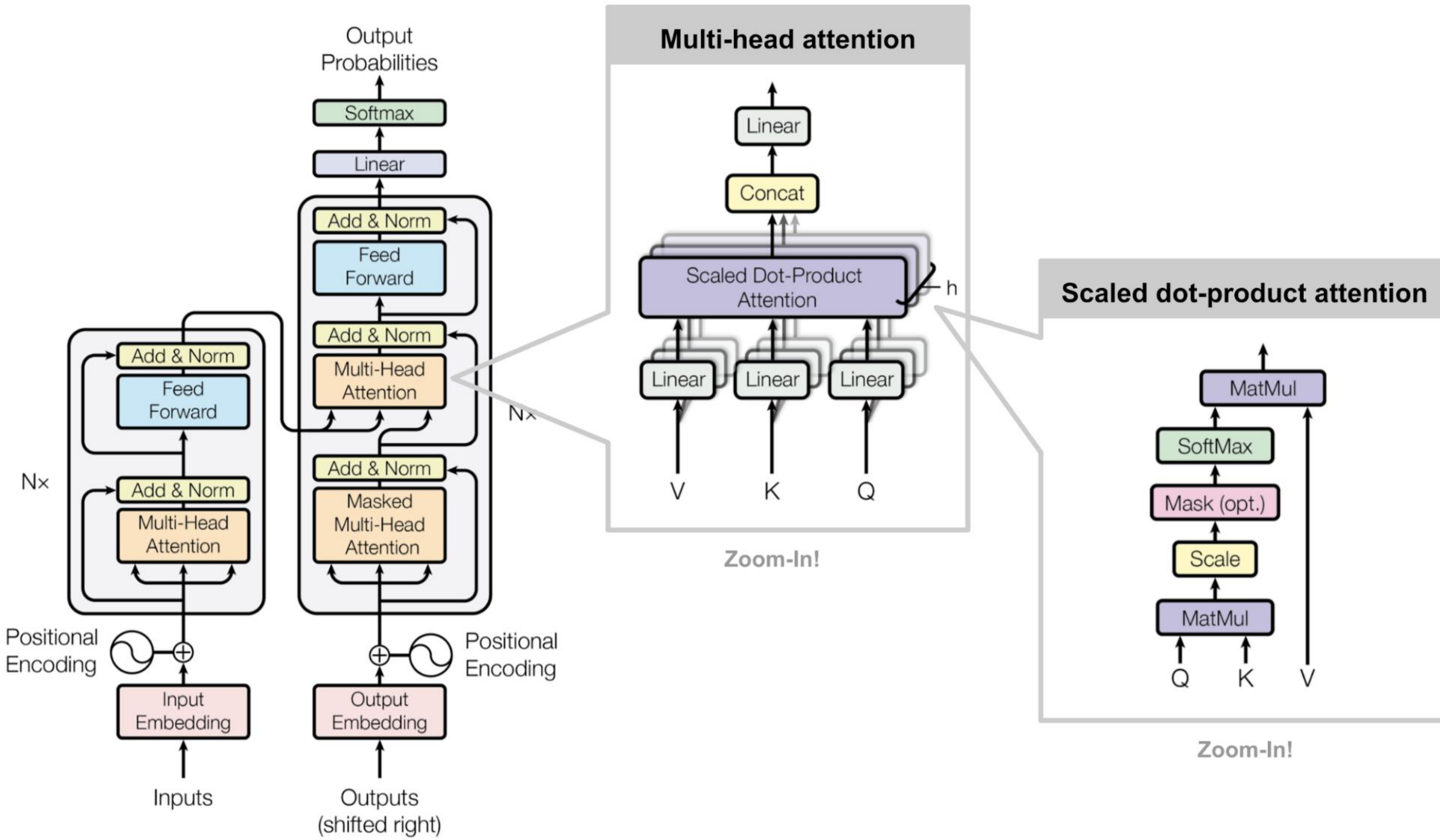
LayerNorm(x + SubLayer(x))

= 6

Masked: not to use the information in the future.

Vaswani et al. Attention is All You Need. 2017

Transformer Architecture Full



Transformer excels in translation

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Outline

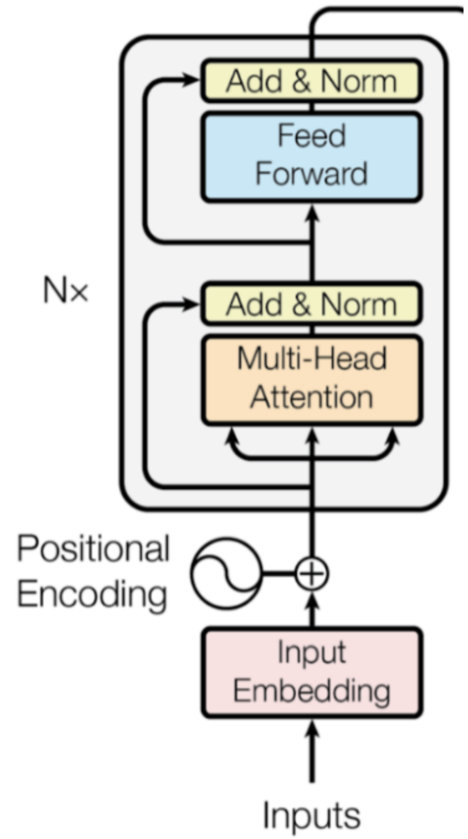
- Sentiment Analysis
- Topic Modeling
- Transformer → BERT → GPT → ChatGPT

- Revisit
  - Social Bot Detection
  - Food Rescue Difficulty Prediction

- Discussion

Bidirectional Encoder Representations from Transformers (BERT)

- BERT Architecture: Just use multi-layer bidirectional Transformer encoder



BERT Training

- Trained with two tasks

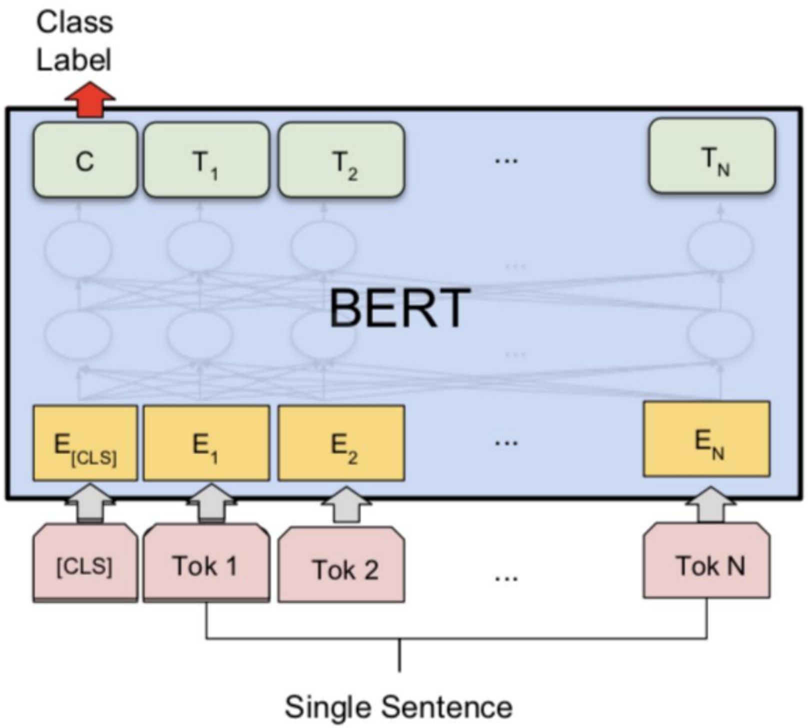
- Task 1: Mask language model (MLM)
  - Randomly mask 15% of tokens in each sequence
  - Train the model to predict the missing words

- Task 2: Next sentence prediction
  - Sample sentence pairs (A, B) so that: (a) 50% of the time, B follows A; (b) 50% of the time, B does not follow A
  - Train the model to process both sentences and output a binary label indicating whether B is the next sentence of A.

Use BERT in Downstream Tasks

- For classification tasks, we get the prediction by taking the final hidden state of the special first token [CLS], $h_L^{[CLS]}$ and multiplying it with a small weight matrix, i.e.,

$$output = softmax(h_L^{[CLS]} W_{CLS})$$



Get BERT Embeddings

```
from transformers import BertTokenizer, BertModel
import torch

# Initialize the tokenizer and model for BERT
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased')

# Sample text
text = "I like AI methods for social good!"
# Encode text
input_ids = tokenizer.encode(text, add_special_tokens=True)
# Add special tokens ([CLS] and [SEP])
input_ids = torch.tensor([input_ids]) # Convert to tensor

with torch.no_grad(): # Disable gradient calculation
    outputs = model(input_ids) # Get model outputs
    hidden_states = outputs.last_hidden_state
# The last hidden-state is the first element of the output tuple

# Get embeddings for each token
embeddings_per_token = hidden_states.squeeze()
# Remove batch dimension, resulting in a shape of [N, 768]

embeddings_per_token.shape # torch.Size([10, 768])
```

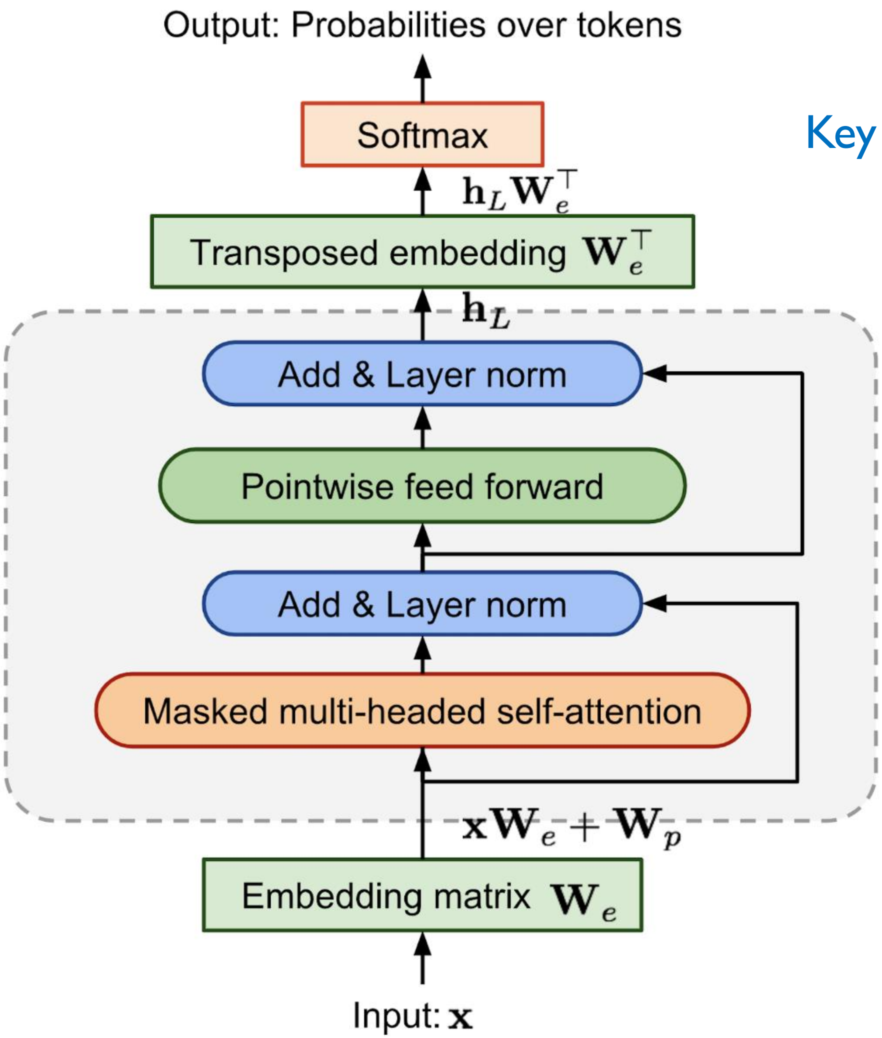
Pre-Training in Natural Language Processing

- Training on a large-scale general domain data before training on a particular task
  - usually raw (unlabelled) and easily available corpus
  - self-supervised: using self-contracted signals
  - there are also cases with supervised pre-training
- Two stages:
  - Pre-train
  - Fine-tune

Outline

- Sentiment Analysis
- Topic Modeling
- Transformer → BERT → GPT → ChatGPT
- Revisit
  - Social Bot Detection
  - Food Rescue Difficulty Prediction
- Discussion

Generative Pre-training Transformer (GPT)



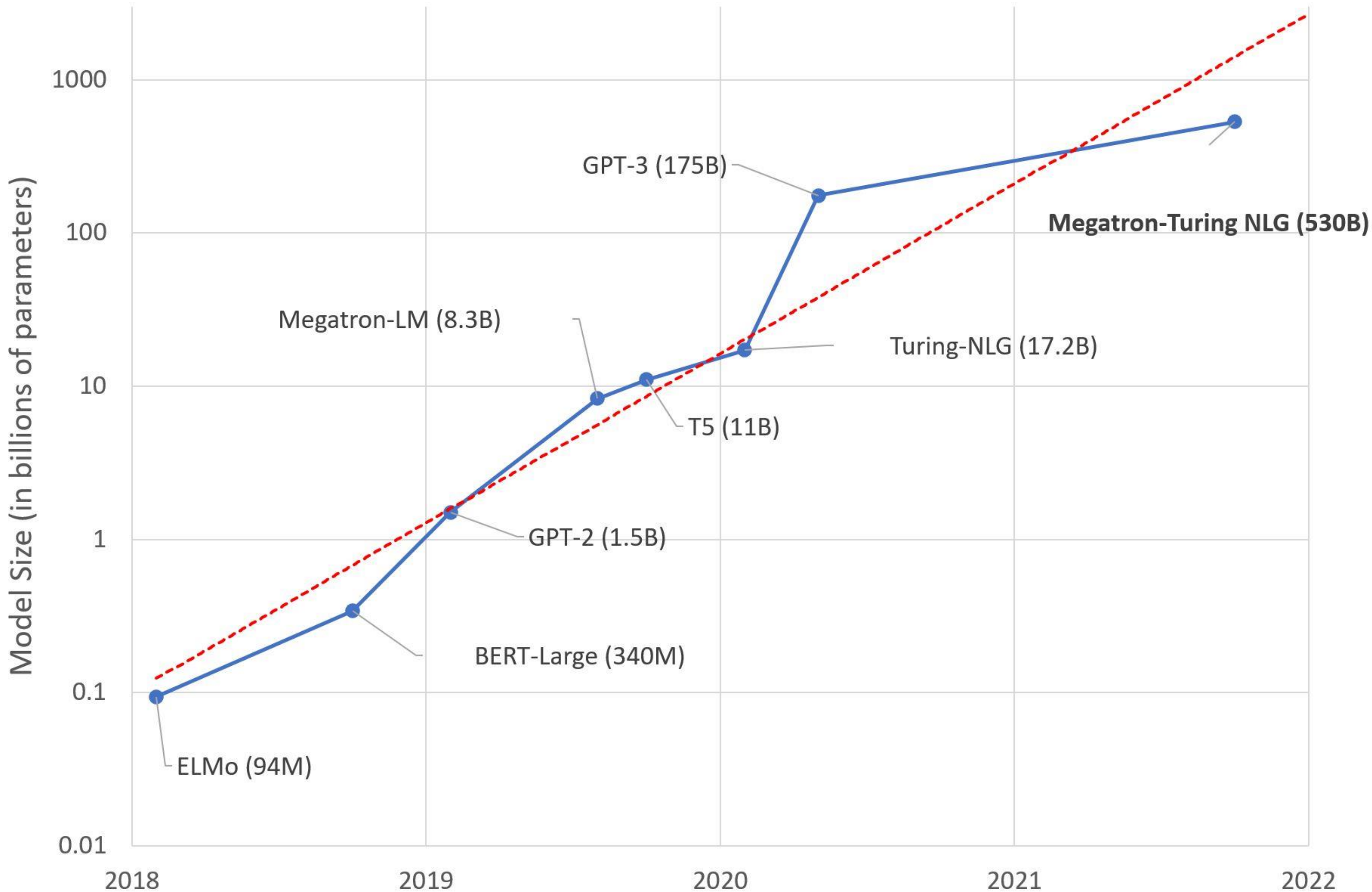
Key idea: Only use Decoder in Transformer

Transformer Block
Repeat x L=12

$$\mathbf{h}_\ell = \text{transformer\_block}(\mathbf{h}_{\ell-1})$$
$$\ell = 1, \ldots, L$$

https://lilianweng.github.io/posts/2019-01-31-lm/

Larger model + More data = Miracle!

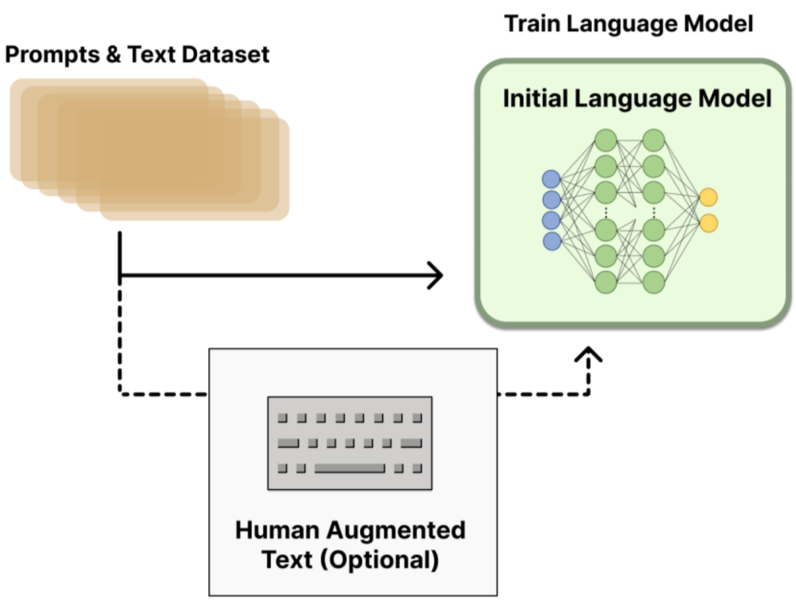


2023

https://huggingface.co/blog/large-language-models

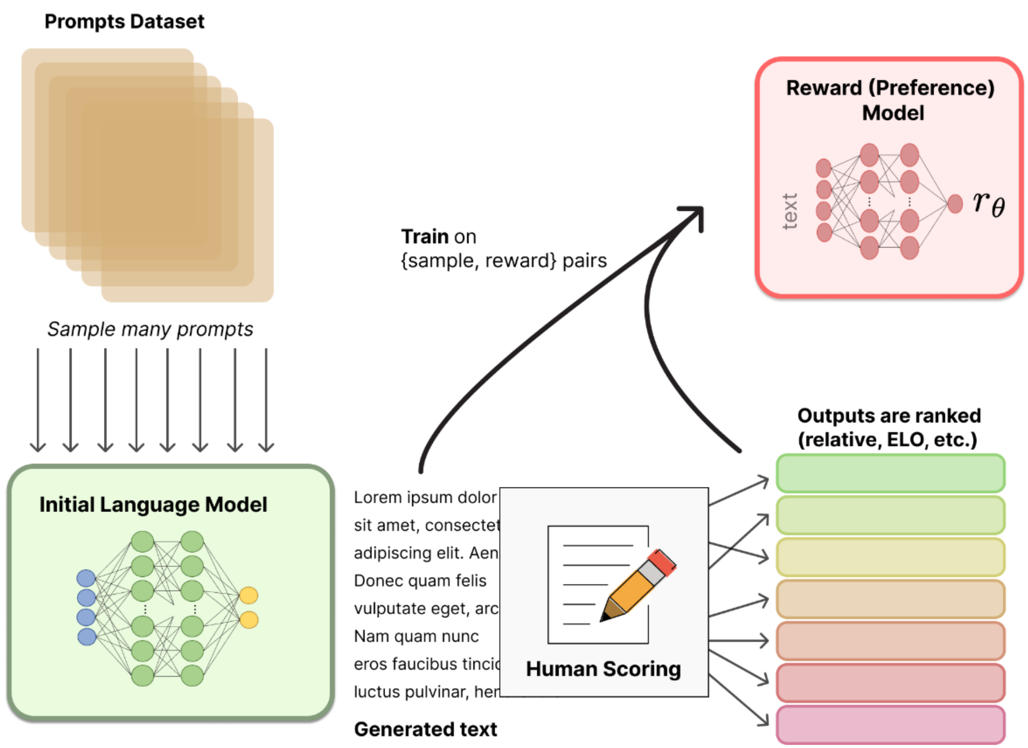
ChatGPT

- GPT-3.5 + Reinforcement Learning from Human Feedback (RLHF)
  - Step 1: Pretrain language model with human-provided answers to prompts



ChatGPT

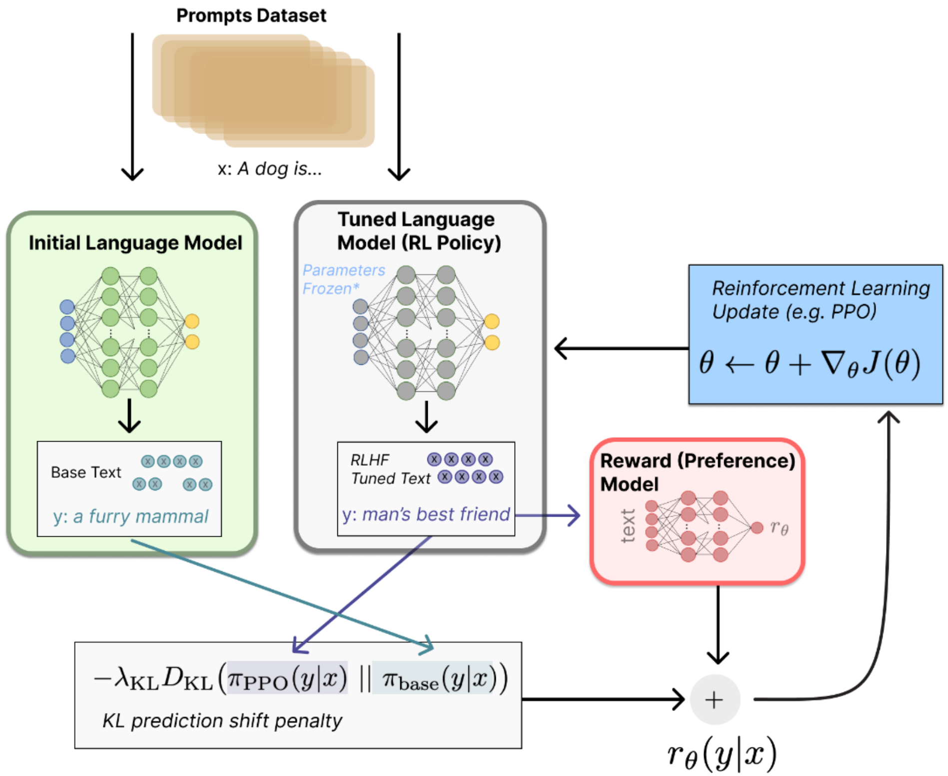
- GPT-3.5 + Reinforcement Learning from Human Feedback (RLHF)
  - Step 2: Train a reward model based on human ranking



https://huggingface.co/blog/rlhf

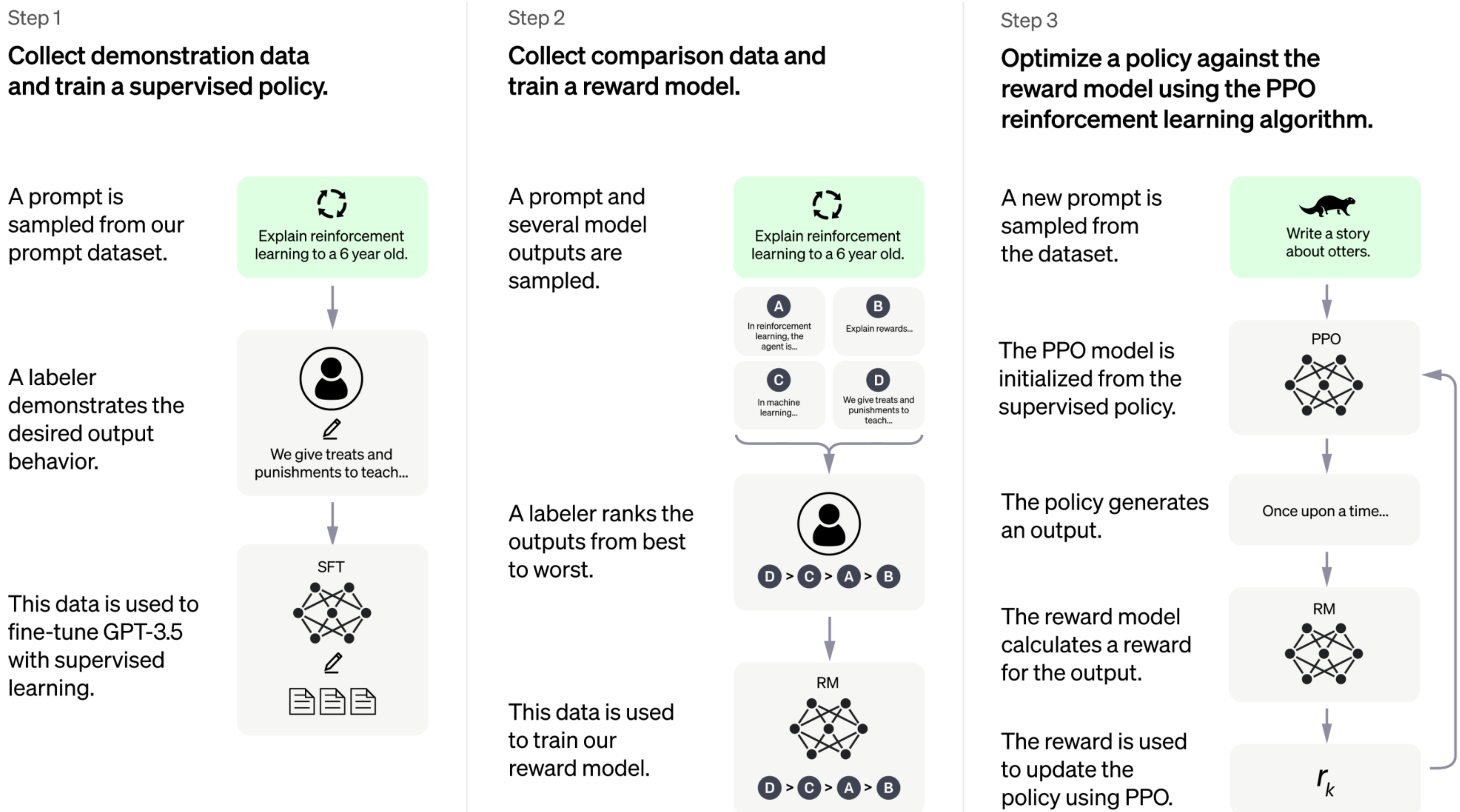
ChatGPT

- GPT-3.5 + Reinforcement Learning from Human Feedback (RLHF)
  - Step 3: Fine-tune language model with RL



https://huggingface.co/blog/rlhf

ChatGPT Summary



Use ChatGPT through API

```
from openai import OpenAI
client = OpenAI()

response = client.chat.completions.create(
  model="gpt-4-turbo-preview",
  messages=[
    {
      "role": "user",
      "content":
"Hello! What do you know about AI methods for social good?"
    }
  ],
  temperature=0.7,
  max_tokens=256,
  top_p=1,
  frequency_penalty=0,
  presence_penalty=0
)
```

- max_tokens (int): max #tokens to generate (1 to 4096)
- temperature (float): Controls randomness (0.0 to 2.0). Higher values → more randomness
- top_p (float): Alternative to sampling with temperature, called nucleus sampling. Values range from 0.0 to 1.0. Higher values means the model will take more risks.

Outline

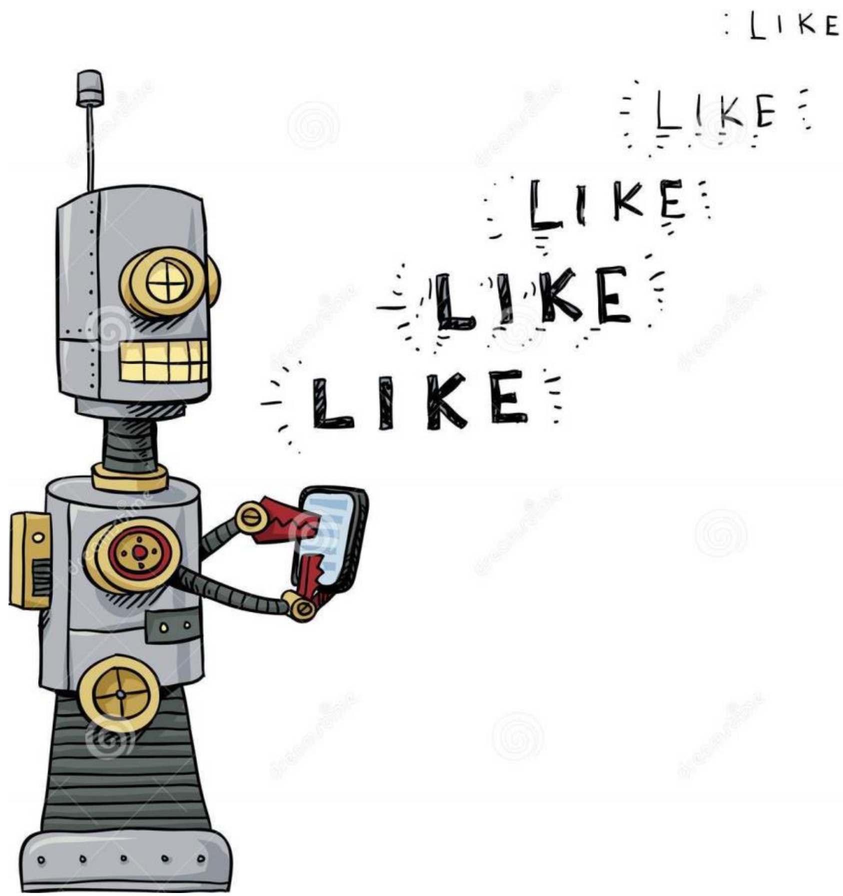
- Sentiment Analysis
- Topic Modeling
- Transformer → BERT → GPT → ChatGPT

- Revisit
  - Social Bot Detection
  - Food Rescue Difficulty Prediction

- Discussion

Exercise revisited: Social Bot Detection

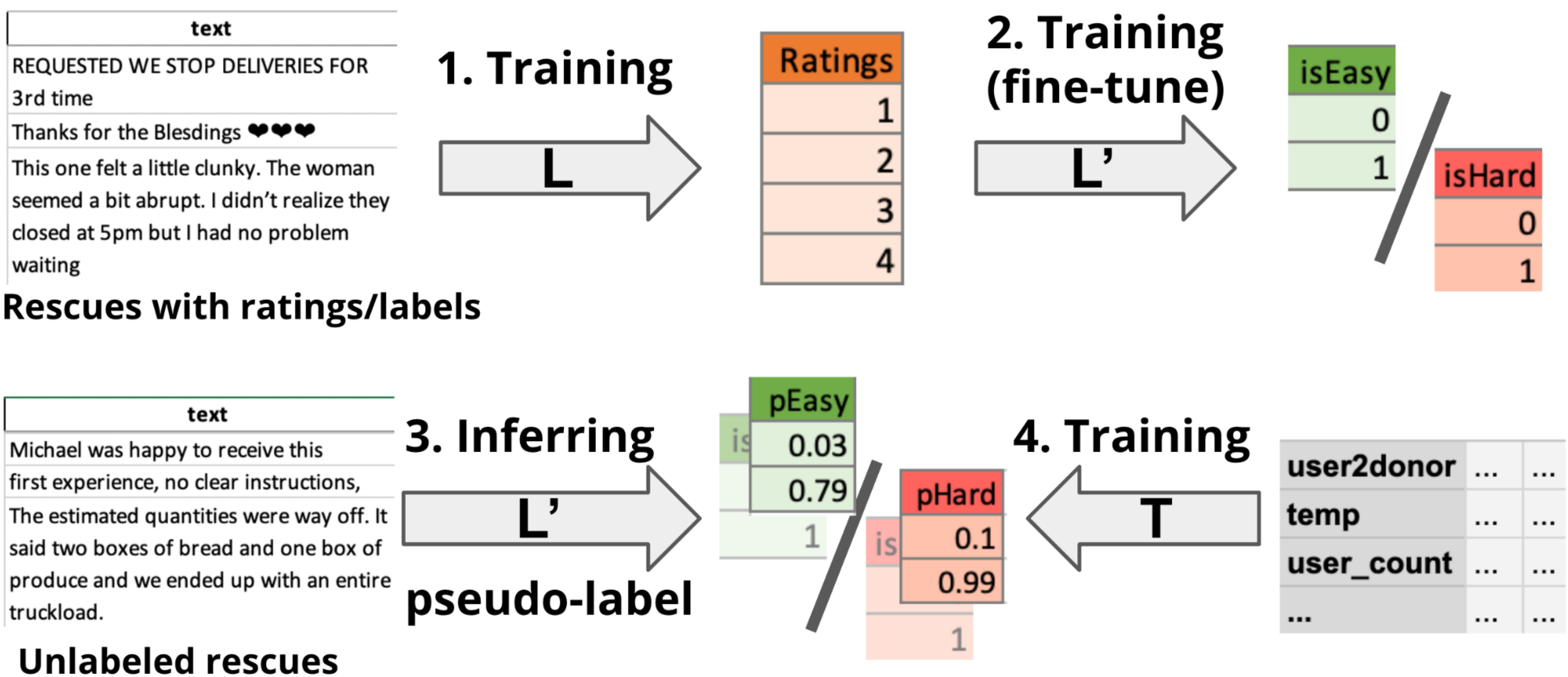
- How to use NLP techniques?



https://www.adweek.com/digital/social-bots-twitter-minor-nuisance/

Case Study Revisited: Food Rescue Difficulty Prediction

- How to complete step 1&2?



Outline

- Sentiment Analysis
- Topic Modeling
- Transformer → BERT → GPT → ChatGPT

- Revisit
  - Social Bot Detection
  - Food Rescue Difficulty Prediction

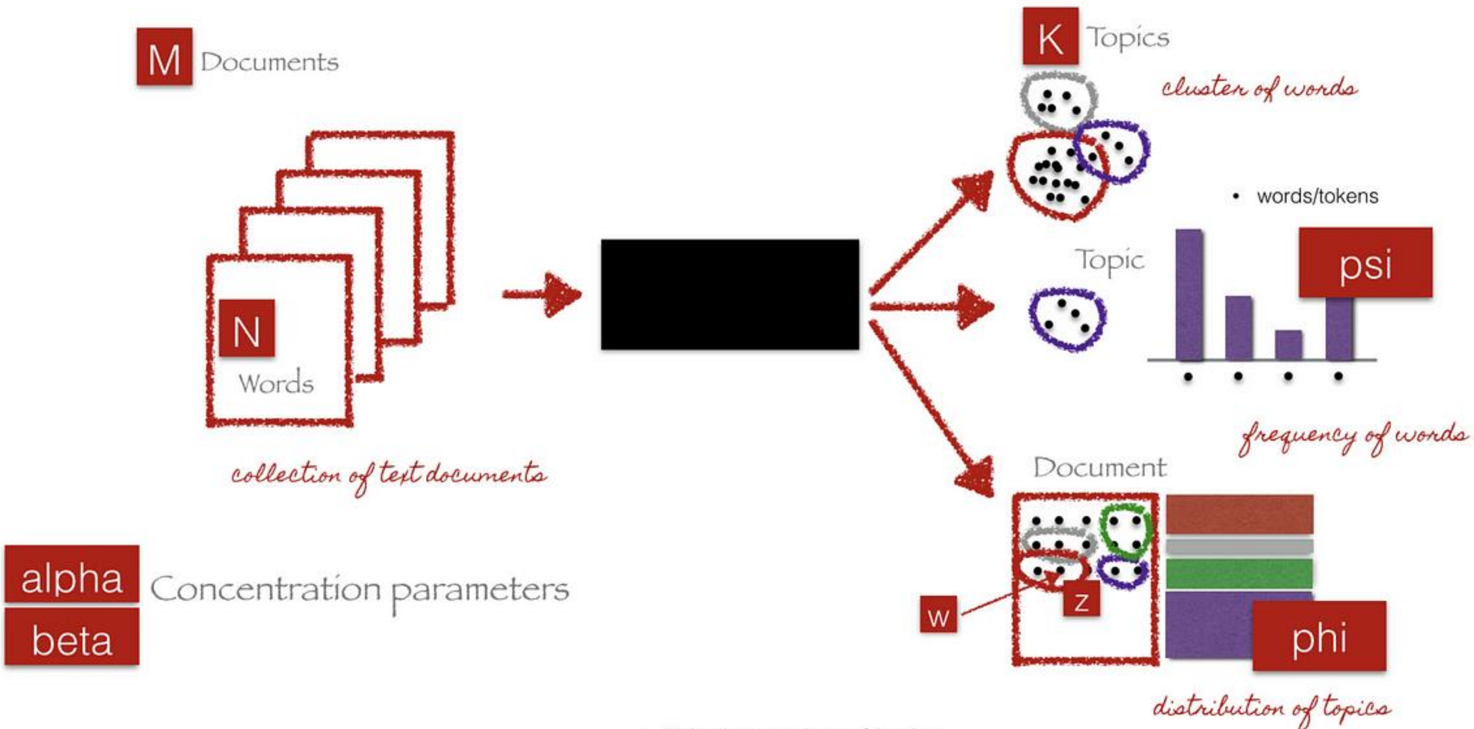
- Discussion

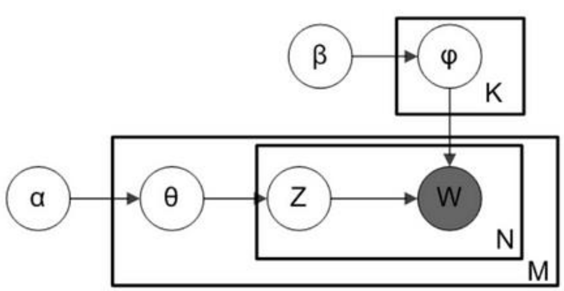
Discussion

- Pick a UN sustainable development goal, and discuss how NLP techniques can help achieve the goal

Backup Slides

Latent Dirichlet Allocation (LDA)

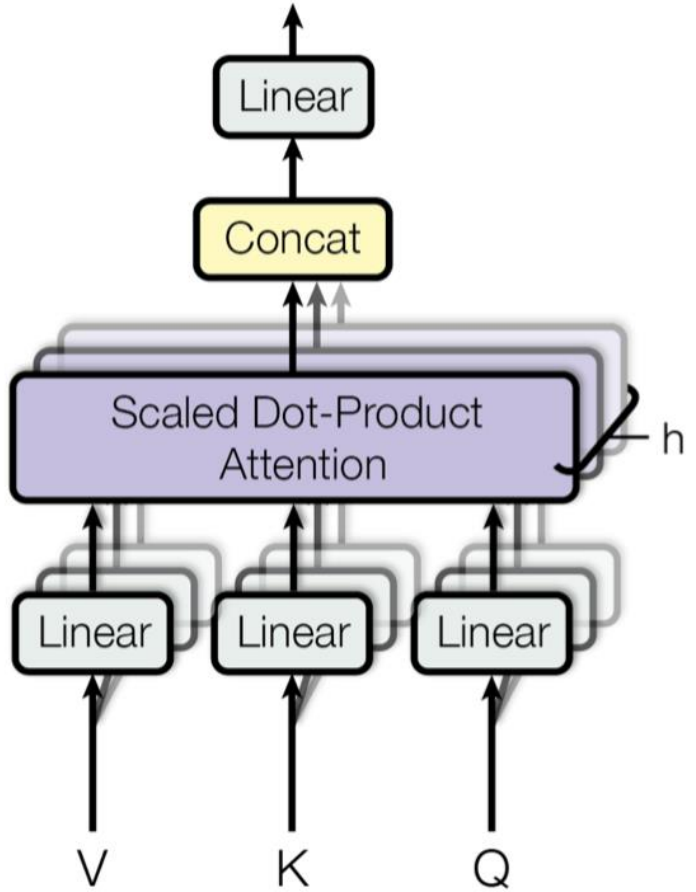




- K is the number of topics
- N is the number of words in the document
- M is the number of documents to analyse
- α is the Dirichlet-prior concentration parameter of the per-document topic distribution
- β is the same parameter of the per-topic word distribution
- φ(k) is the word distribution for topic k
- θ(i) is the topic distribution for document i
- z(i,j) is the topic assignment for w(i,j)
- w(i,j) is the j-th word in the i-th document
- φ and θ are Dirichlet distributions, z and w are multinomials.

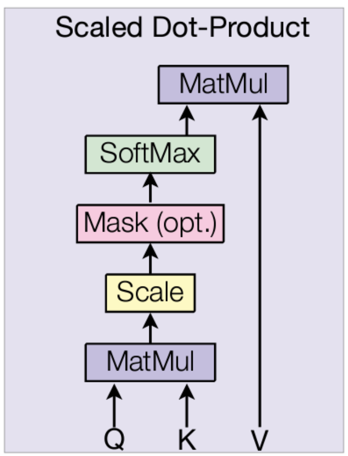
Transformer

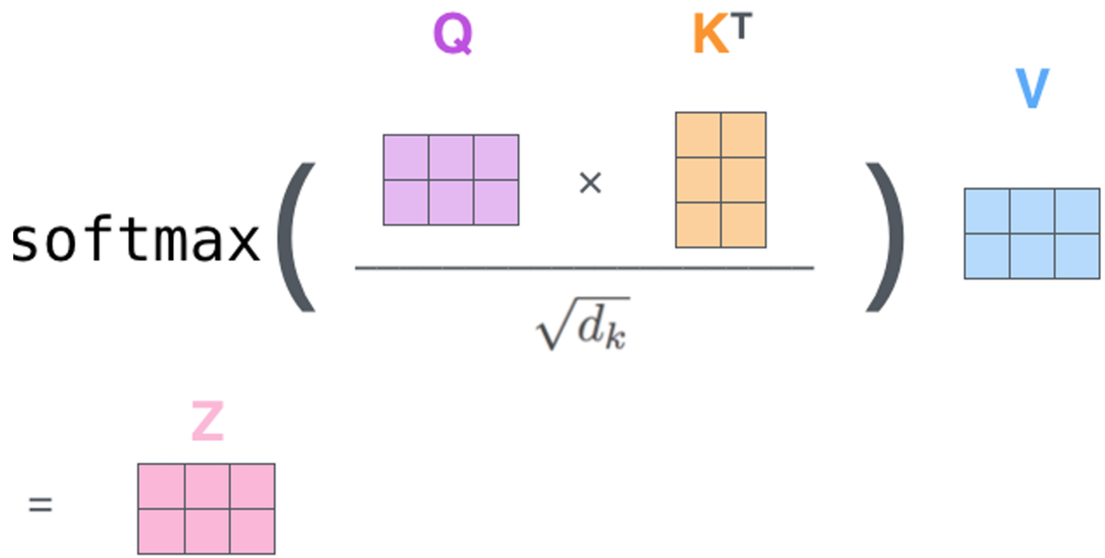
- Multi-head self-attention



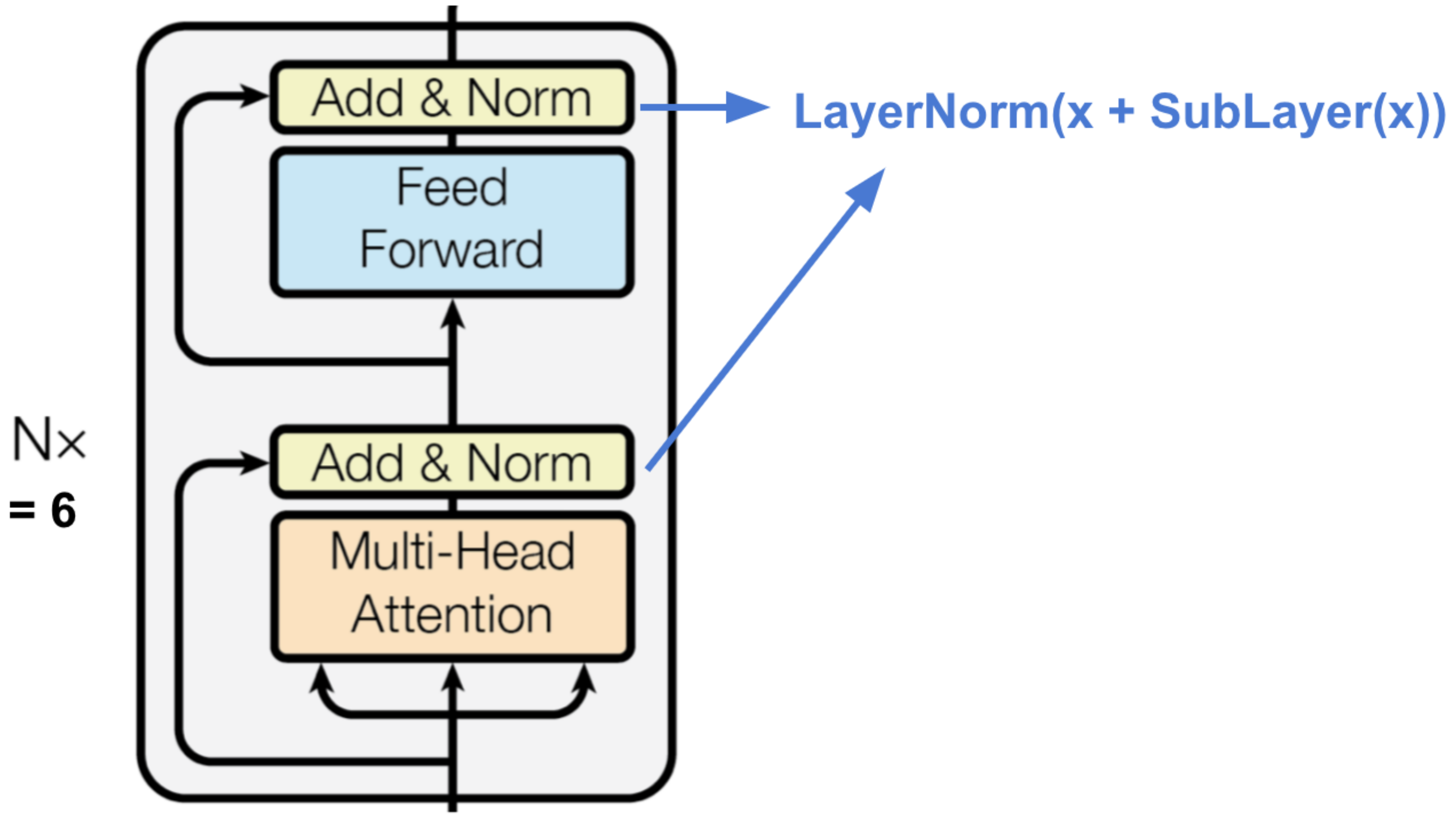
Scaled Dot-Product Attention in Transformer

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





Transformer Encoder



Transformer Decoder