Natural Language Processing

Berkeley
N L P

LLMs: Training

BAIR NLP Workshop

- Tomorrow, Friday, October 18, in BWW and virtual
- Highlights:
  - **LLMs and Cognitive Systems**
  - LLM agents
  - Creativity in humans and LLMs
  - Interpretability
  - Language in interaction
- Agenda and link to RSVP: https://docs.google.com/document/d/1-WJfTMfYnCwlyIsJxXoRzjRJNlJ1MyiFkcFLQNYPviE/

Recap: What is a language model?

- Language models assign a probability to a sequence of words

$$p(\overline{y})$$

- We can decompose this probability using the chain rule

$$p(\overline{y}) = \Pi_{i=1}^{T} p(y_i | y_{0:i-1})$$

- We can autoregressively generate sequences from the language model by sampling from its token-level probability

$$p(y_i | y_{0:i-1})$$

- We can condition on our language distribution on something else

$$p(y_i | y_{0:i-1}; \overline{x})$$

How to Train a Base LM

1. Get some training data
2. Preprocess it (tokenize it)
3. Choose your architecture
4. Optimize a language modeling objective
5. Run inference!

Step 1: Get Training Data

Training Data

- Transformer models are very data-hungry
- Solution: just scrape the web
- **CommonCrawl:** publicly available web scrape collected since 2007 containing 250B webpages, comprising 82% of tokens used to train GPT-3

Data Sources

- Domain-specific webpages:
  - Code and mathematics: Github, StackOverflow
  - Academic and scientific work: arXiv, bioRxiv, PubMed
  - Books: Project Gutenberg
  - General knowlege: Wikipedia
- Domain-general sources:
  - Social media (reddit, Twitter)
  - News sites

Data Sources

The History of Large Language Models

1998	CPAT-Tree-Based Language Models with an Application for Text Verification in Chinese. ROCLing 1998. First use of LLM trigram I know of; 200M word corpus
2000	A Neural Probabilistic Language Model. Bengio, Ducharme & Vincent NIPS 2000 First neural language model built on 32 million token corpus, 31K vocab
2007	Large Language Models in Machine Translation. Brants, Popat, Xu, Och and **Dean**. EMNLP 2007. **2 trillion token corpus** n-gram model of up to 5-grams
2018	**GPT** (Radford, Narasimhan, Salimans & Sutskever) and **BERT** (Devlin, Chang, Lee & Toutanova). 3.3 billion token corpus
2020–	100+ billion parameter neural language models trained on > 1 trillion tokens: GPT-3, GPT-4, PaLM 2, Llama 3, Nemotron-4, ….

Chris Manning's COLM keynote, https://www.youtube.com/watch?v=c3N2H3Z5S3I

Web Scraping

1. Seed webcrawler with initial URLs
2. Identify new URLs via outlinks
3. Download HTML representation of webpage
4. Scrape HTML for raw text
5. Postprocess texts

$$\mathcal{D} = \left\{ \overline{d}^{(i)} \right\}_{i=1}^{N} \qquad \overline{d} = \langle b_0, \ldots, b_M \rangle$$

Web Data is Noisy

- Deduplication
- Remove junk / nonsense text that's very unlikely according to a simple n-gram language model
- Remove uninteresting pages with few inlinks
- Remove non-English data with external classifiers

Web Data is Unfiltered

- Personally identifiable information (PII) or other personal information
- Adult content
- Explicit hate speech, disinformation
- Copyrighted data
- Test data from NLP benchmarks…

Downstream Effects

Stable Diffusion produces copyright and trademarked images

Original:

Generated:

Codex generates code with non-permissive licenses

```
CBlockIndex * InsertBlockIndex(uint256 hash)
{
    if (hash.IsNull())
        return NULL;

    // Return existing
    BlockMap::iterator mi = mapBlockIndex.find(hash);
    if (mi != mapBlockIndex.end())
        return (*mi).second;

    CBlockIndex* pindexNew = new CBlockIndex();
    if (!pindexNew)
        throw runtime_error("LoadBlockIndex(): new CBlockIndex failed");
    mi = mapBlockIndex.insert(make_pair(hash, pindexNew)).first;
    pindexNew->phashBlock = &((*mi).first);

    return pindexNew;
}
```

Stable Diffusion generates real individuals

Social Impacts of Webscraping

- Trained language models encode:
  - Biases explicitly or implicitly encod
  - Personal information about individ
  - Copyrighted data

Karla Ortiz

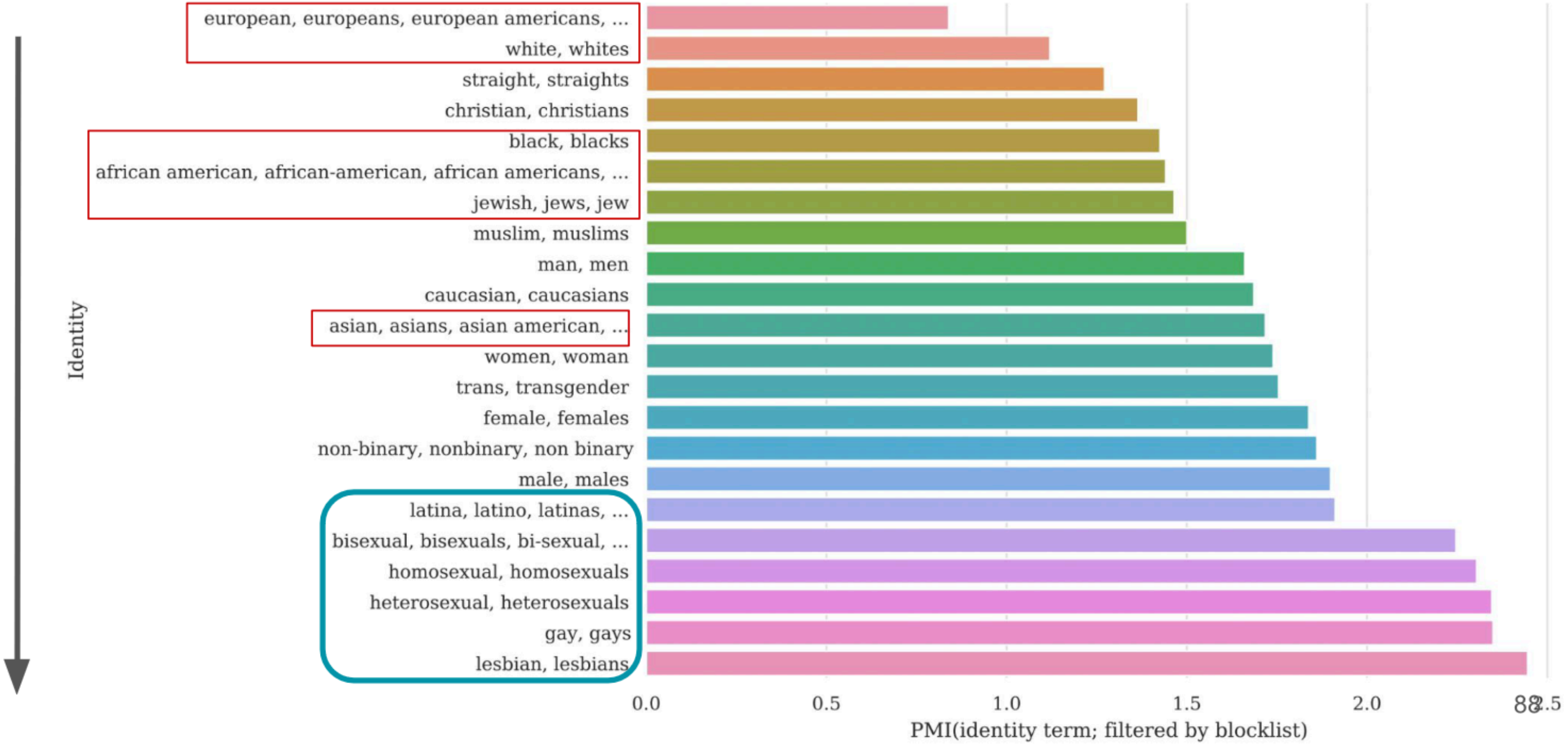
Sarah Andersen

Glaze, Shan et al. 2023, USENIX

Tradeoffs in Filtering

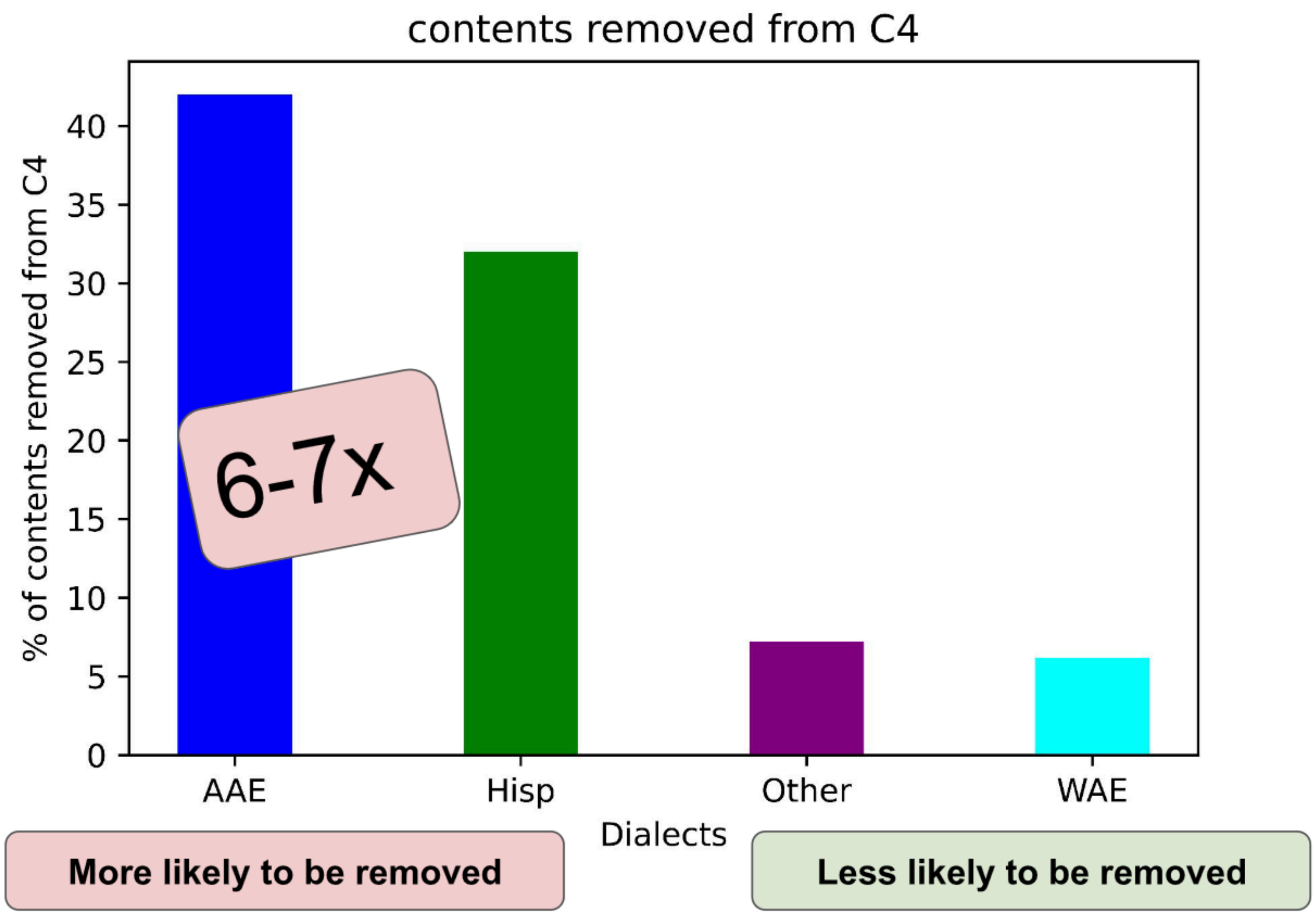
- Personally identifiable information (PII) or other personal information → Phone numbers of public companies' customer service lines?
- Adult content → Very culturally dependent
- Explicit hate speech, disinformation → What might appear to be hateful or toxic speech is context-dependent

Tradeoffs in Filtering



Banerjee and Rubungo, Princeton COS597G

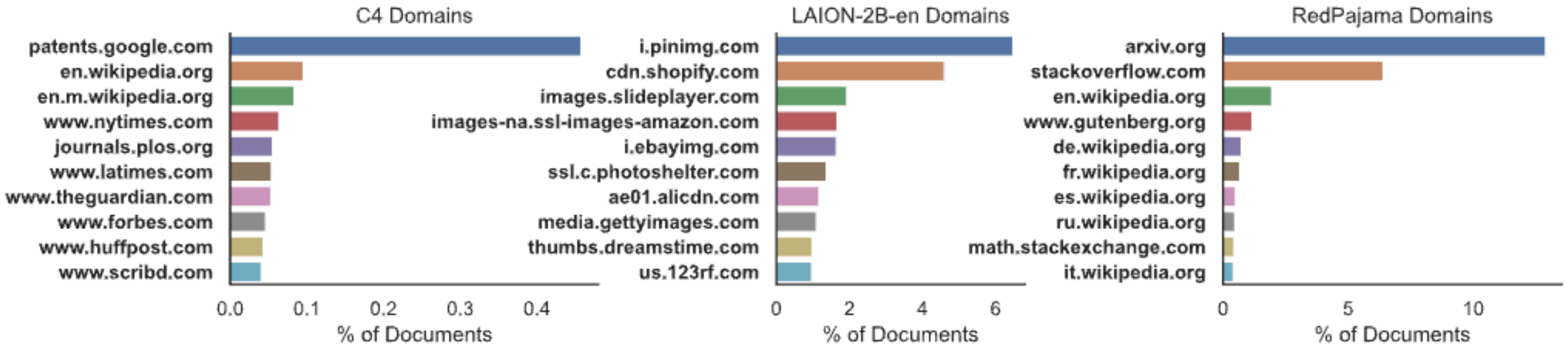
Tradeoffs in Filtering



Banerjee and Rubungo, Princeton COS597G

Pretraining Corpora

Dataset	Origin	Model	Size (GB)	# Documents	# Tokens	max(# Tokens)	min(# Tokens)
OpenWebText	Gokaslan & Cohen (2019)	GPT-2*	41.2	8,005,939	7,767,705,349	95,139	128
C4	Raffel et al. (2020)	T5	838.7	364,868,892	153,607,833,664	101,898	5
mC4-en	Chung et al. (2023)	umT5	14,694.0	3,928,733,374	2,703,077,876,916	181,949	1
OSCAR	Abadji et al. (2022)	BLOOM*	3,327.3	431,584,362	475,992,028,559	1,048,409	1
The Pile	Gao et al. (2020)	GPT-J/Neo & Pythia	1,369.0	210,607,728	285,794,281,816	28,121,329	0
RedPajama	Together Computer (2023)	LLaMA*	5,602.0	930,453,833	1,023,865,191,958	28,121,329	0
S2ORC	Lo et al. (2020)	SciBERT*	692.7	11,241,499	59,863,121,791	376,681	1
peS2o	Soldaini & Lo (2023)	-	504.3	8,242,162	44,024,690,229	97,043	154
LAION-2B-en	Schuhmann et al. (2022)	Stable Diffusion*	570.2	2,319,907,827	29,643,340,153	131,077	0
The Stack	Kocetkov et al. (2023)	StarCoder*	7,830.8	544,750,672	1,525,618,728,620	26,298,134	0

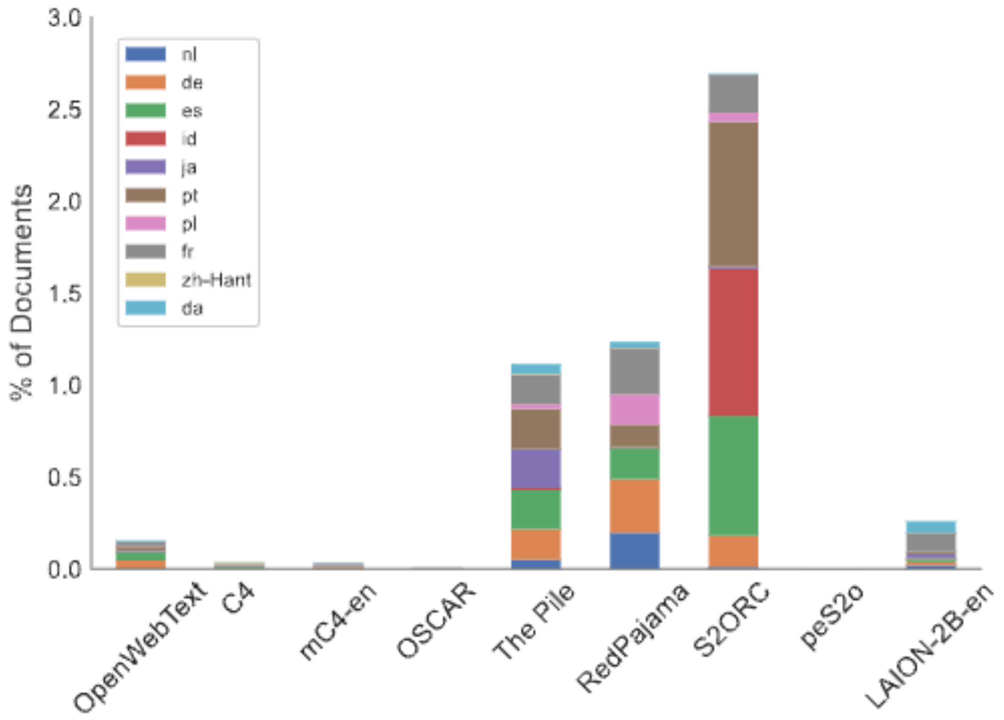


WIMBD: What's in my big data? Elazar et al. 2024

Pretraining Corpora

Table 5: Extrapolated PII frequencies. Count is the extrapolated frequency in a corpus and *Prec.* is our identification precision, estimated by manual analysis.

	Email Addresses		**Phone Numbers**		**IP Addresses**	
	Count	Prec.	Count	Prec.	Count	Prec.
OpenWebText	364K	*99*	533K	*87*	70K	*54*
OSCAR	62.8M	*100*	107M	*91*	3.2M	*43*
C4	7.6M	*99*	19.7M	*92*	796K	*56*
mC4-en	201M	*92*	4B	*66*	97.8M	*44*
The Pile	19.8M	*43*	38M	*65*	4M	*48*
RedPajama	35.2M	*100*	70.2M	*94*	1.1M	*30*
S2ORC	630K	*100*	1.4M	*100*	0K	*0*
peS2o	418K	*97*	227K	*31*	0K	*0*
LAION-2B-en	636K	*94*	1M	*7*	0K	*0*
The Stack	4.3M	*53*	45.4M	*9*	4.4M	*55*



WIMBD: What's in my big data? Elazar et al. 2024

Step 2: Tokenization

Tokenization

$$\mathcal{D} = \left\{ \overline{d}^{(i)} \right\}_{i=1}^{N}$$

"They currently play their home games at Acrisure Stadium."

$\downarrow$

"They" "currently" "play" "their" "home" "game" "#s" "at" "Acrisure" "Stadium" "."

$$\overline{d} = \langle b_0, \ldots, b_M \rangle$$

$\downarrow$

$$\overline{d} = \langle x_0, \ldots, x_{M'} \rangle$$

$$\forall x,\ x \in \mathcal{V}$$

Tokenization

- Maps from byte sequences to sequences of tokens, where each token is part of a set vocabulary

$$\bar{d} = \langle b_0, \ldots, b_M \rangle$$

$$\downarrow$$

$$\bar{d} = \langle x_0, \ldots, x_{M'} \rangle$$

$$\forall x,\ x \in \mathcal{V}$$

Tokenization

- Approach: simple heuristics (split on spaces, handle punctuation gracefully)

$$\bar{d} = \langle b_0, \ldots, b_M \rangle$$

$$\downarrow$$

$$\bar{d} = \langle x_0, \ldots, x_{M'} \rangle$$

$$\forall x,\ x \in \mathcal{V}$$

"They currently play their home games at Acrisure Stadium."

↓

"They" "currently" "play" "their" "home" "games" "at" "Acrisure" "Stadium" "."

Problem: requires defining heuristics, including for edge cases

Problem: heuristics are not generalizable to all languages

เราทุกคนเกิดมาอย่างอิสระ เราทุกคนมีความคิดและความเข้าใจเป็นของเราเอง เราทุกคนควรได้รับการปฏิบัติในทางเดียวกัน.

Example from CMU LLMs course

Tokenization

Turkish	English
ev	(the) house
evler	(the) houses
evin	your (sing.) house
eviniz	your (pl./formal) house
evim	my house
evimde	at my house
evlerinizin	of your houses
evlerinizden	from your houses
evlerinizdendi	(he/she/it) was from your houses
evlerinizdenmiş	(he/she/it) was (apparently/said to be) from your houses
Evinizdeyim.	I am at your house.
Evinizdeymişim.	I was (apparently) at your house.
Evinizde miyim?	Am I at your house?

it on $\overline{d} = \langle b_0, \ldots, b_M \rangle$

Case	Ending	Examples		Meaning
		***köy* "village"**	***ağaç* "tree"**	
Nominative	∅ (none)	*köy*	*ağaç*	(the) village/tree
Accusative	*-i* 4	*köyü*	*ağacı*	the village/tree
Genitive	*-in* 4	*köyün*	*ağacın*	the village's/tree's of the village/tree
Dative	*-e* 2	*köye*	*ağaca*	to the village/tree
Locative	*-de* 2	*köyde*	*ağaçta*	in/on/at the village/tree
Ablative	*-den* 2	*köyden*	*ağaçtan*	from the village/tree
Instrumental	*-le* 2	*köyle*	*ağaçla*	with the village/tree

Problem: many words never appear in the training data

Example from CMU LLMs course

Character- / Byte-Level Models

- Approach: vocabulary is simply all possible Unicode characters that might appear

$$\overline{d} = \langle b_0, \ldots, b_M \rangle$$

$$\downarrow$$

$$\overline{d} = \langle x_0, \ldots, x_{M'} \rangle$$

$$\forall x,\ x \in \mathcal{V}$$

Problem: representations of each character are not meaningful

Problem: model also needs to learn how to compose words from characters

Problem: input sequences become very long

Tokenization

- Approach: subword tokenization, where frequent words are kept whole and infrequent words are broken into parts

$$\bar{d} = \langle b_0, \ldots, b_M \rangle$$

$$\downarrow$$

$$\bar{d} = \langle x_0, \ldots, x_{M'} \rangle$$

$$\forall x,\ x \in \mathcal{V}$$

"They currently play their home games at Acrisure Stadium."

↓

'▁They', '▁currently', '▁play', '▁their', '▁home', '▁games', '▁at', '▁A', 'cris', 'ure', '▁Stadium', '.'

Adam la tanış tı m

Adam: indirect object; la: instrumental case suffix; tanış: verb stem; tı: past tense suffix; m: indicator of subject

I met with the man

Adam ın kitab ı

Adam: possessor; ın: genitive suffix; kitab: possessed noun; ı: possessive ending

Man's book

Example from CMU LLMs course

Byte Pair Encoding

- Gradually constructs vocabulary given a target size
- Starts with a base vocabulary consisting of all characters in the training data
- Iteratively constructs vocabulary:
  - Tokenizes all training documents given the current vocabulary
  - Adds the most common bigram to the vocabulary
- Terminates when target vocabulary size is reached

$$\bar{d} = \langle b_0, \ldots, b_M \rangle$$

$$\downarrow$$

$$\bar{d} = \langle x_0, \ldots, x_{M'} \rangle$$

$$\forall x,\ x \in \mathcal{V}$$

Byte Pair Encoding

Documents + frequencies: (“hug”, 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)

(“h”, “u”, “g”, “p”, “n”, “b”, “s”)

Example from HuggingFace

Byte Pair Encoding

Documents + frequencies: (“hug”, 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)

(“h”, “u”, “g”, “p”, “n”, “b”, “s”) → ("h" "u" "g", 10), ("p" "u" "g", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "u" "g" "s", 5)

Example from HuggingFace

Byte Pair Encoding

Documents + frequencies: (“hug”, 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)

(“h”, “u”, “g”, “p”, “n”, “b”, “s”) $\longrightarrow$ ("h" "u" "g", 10), ("p" "u" "g", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "u" "g" "s", 5)

(“h”, “u”, “g”, “p”, “n”, “b”, “s”, “ug”) $\longrightarrow$ ("h" "ug", 10), ("p" "ug", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "ug" "s", 5)

Example from HuggingFace

Byte Pair Encoding

Documents + frequencies: (“hug”, 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)

(“h”, “u”, “g”, “p”, “n”, “b”, “s”) → ("h" "u" "g", 10), ("p" "u" "g", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "u" "g" "s", 5)

(“h”, “u”, “g”, “p”, “n”, “b”, “s”, “ug”) → ("h" "ug", 10), ("p" "ug", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "ug" "s", 5)

(“h”, “u”, “g”, “p”, “n”, “b”, “s”, “ug”, “un”) → ("h" "ug", 10), ("p" "ug", 5), ("p" "un", 12), ("b" "un", 4), ("h" "ug" "s", 5)

Example from HuggingFace

Byte Pair Encoding

Documents + frequencies: (“hug”, 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)

(“h”, “u”, “g”, “p”, “n”, “b”, “s”) → ("h" "u" "g", 10), ("p" "u" "g", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "u" "g" "s", 5)

(“h”, “u”, “g”, “p”, “n”, “b”, “s”, “ug”) → ("h" "ug", 10), ("p" "ug", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "ug" "s", 5)

(“h”, “u”, “g”, “p”, “n”, “b”, “s”, “ug”, “un”) → ("h" "ug", 10), ("p" "ug", 5), ("p" "un", 12), ("b" "un", 4), ("h" "ug" "s", 5)

(“h”, “u”, “g”, “p”, “n”, “b”, “s”, “ug”, “un”, “hug”) → (“hug", 10), ("p" "ug", 5), ("p" "un", 12), ("b" "un", 4), ("hug" "s", 5)

Example from HuggingFace

Byte Pair Encoding

Documents + frequencies: (“hug”, 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)

(“h”, “u”, “g”, “p”, “n”, “b”, “s”) → ("h" "u" "g", 10), ("p" "u" "g", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "u" "g" "s", 5)

(“h”, “u”, “g”, “p”, “n”, “b”, “s”, “ug”) → ("h" "ug", 10), ("p" "ug", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "ug" "s", 5)

(“h”, “u”, “g”, “p”, “n”, “b”, “s”, “ug”, “un”) → ("h" "ug", 10), ("p" "ug", 5), ("p" "un", 12), ("b" "un", 4), ("h" "ug" "s", 5)

(“h”, “u”, “g”, “p”, “n”, “b”, “s”, “ug”, “un”, “hug”) → (“hug", 10), ("p" "ug", 5), ("p" "un", 12), ("b" "un", 4), ("hug" "s", 5)

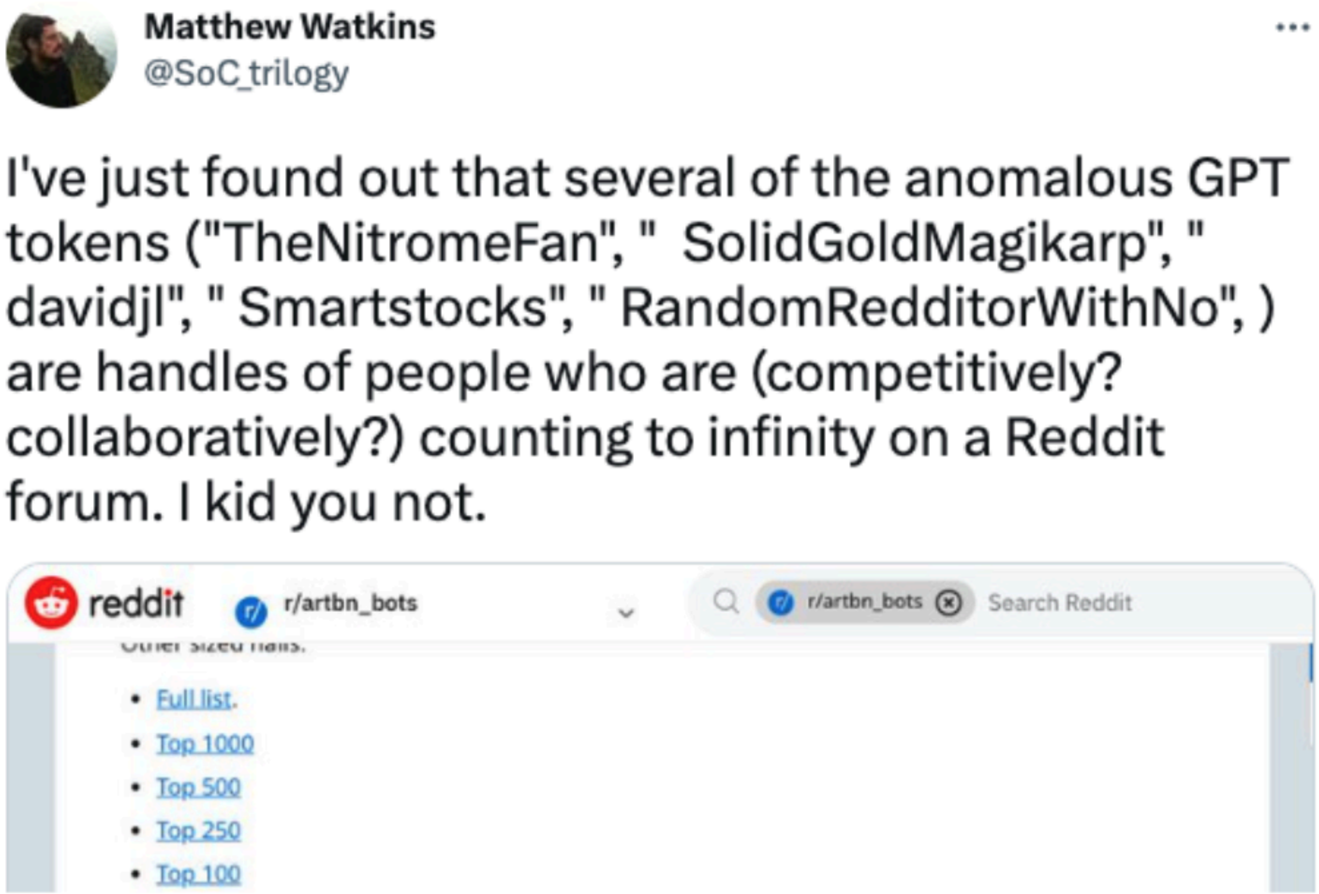
New word: “puns”

“p” “u” “n” “s” → “p” “un” “s” ✔

Modern Tokenization and Vocabularies

- Subword tokenization is used for all modern pretrained models (though people are still experimenting with character-based models)
- Vocabularies contain ~50-250k wordpieces
- Pretrained word embeddings (e.g. GloVe) aren't necessary

Modern Tokenization and Vocabularies



Rank	User	Counts
1	/u/davidjl123	163477
2	/u/Smartstocks	113829
3	/u/atomicimploder	103178
4	/u/TheNitromeFan	84581
5	/u/SolidGoldMagikarp	65753
6	/u/RandomRedditorWithNo	63434
7	/u/rideride	59024
8	/u/Mooraell	57785
9	/u/Removedpixel	55080
10	/u/Adinida	48415
11	/u/rschaosid	47132

Example from UT Austin CS 388

Step 3: Architecture

Recap: Feedfoward Networks

- Tokenize
- Embed
- Concatenate
- Linear layer
- Softmax
- Fixed window?
- Word averaging?

output distribution

$$\hat{\boldsymbol{y}} = \mathrm{softmax}(\boldsymbol{U}\boldsymbol{h} + \boldsymbol{b}_2) \in \mathbb{R}^{|V|}$$

hidden layer

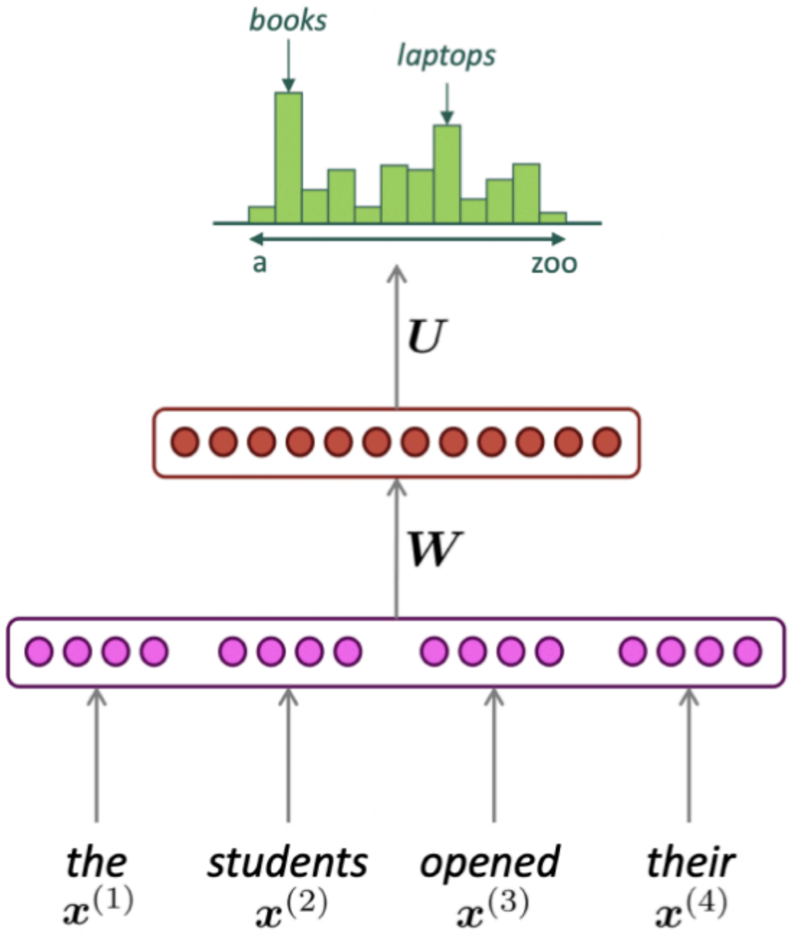
$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{e} + \boldsymbol{b}_1)$$

concatenated word embeddings

$$\boldsymbol{e} = [\boldsymbol{e}^{(1)}; \boldsymbol{e}^{(2)}; \boldsymbol{e}^{(3)}; \boldsymbol{e}^{(4)}]$$

words / one-hot vectors

$$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \boldsymbol{x}^{(4)}$$



Recap: Recurrence

output distribution

$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$

hidden states

$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$

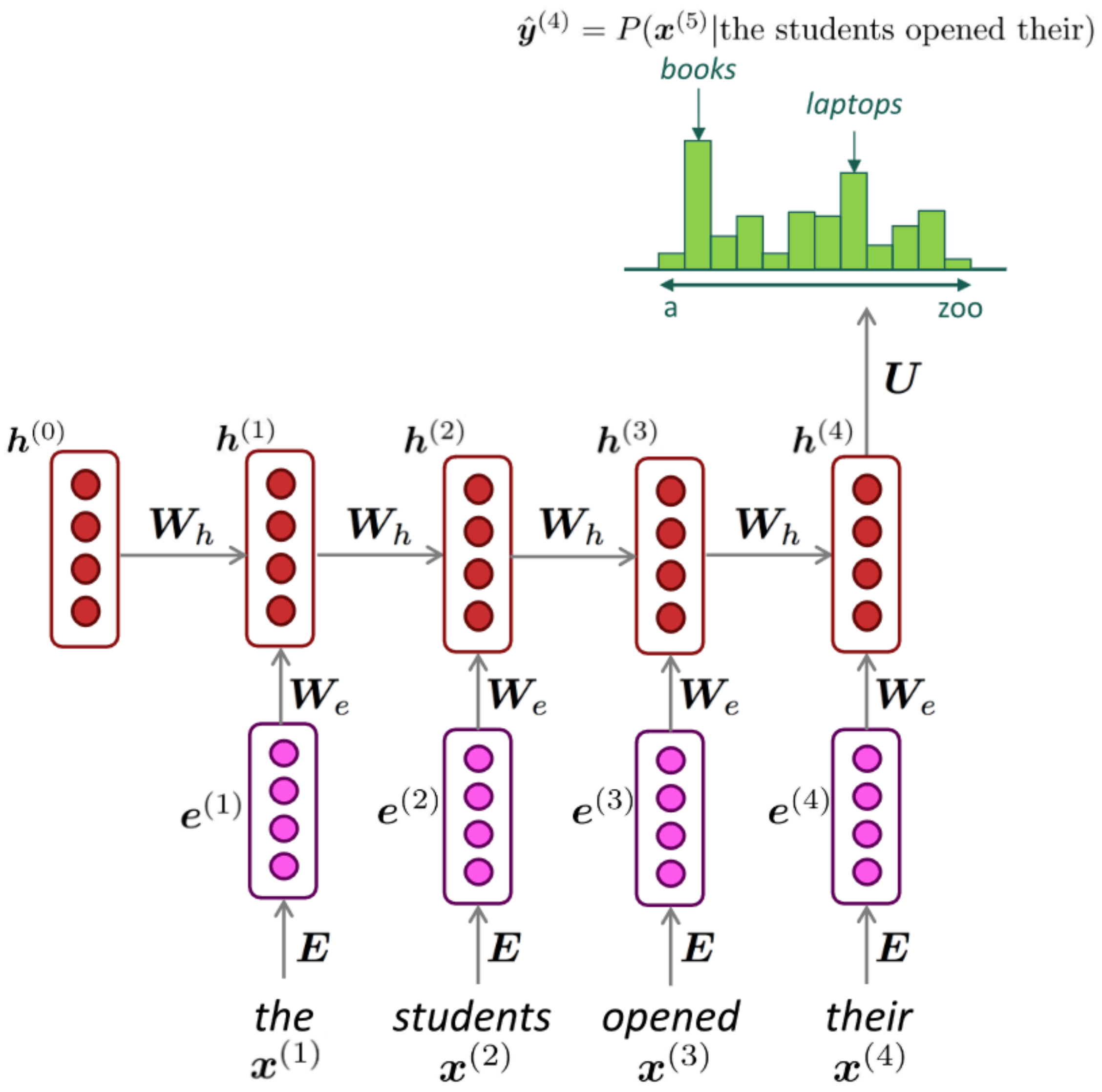
$\boldsymbol{h}^{(0)}$ is the initial hidden state

word embeddings

$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$

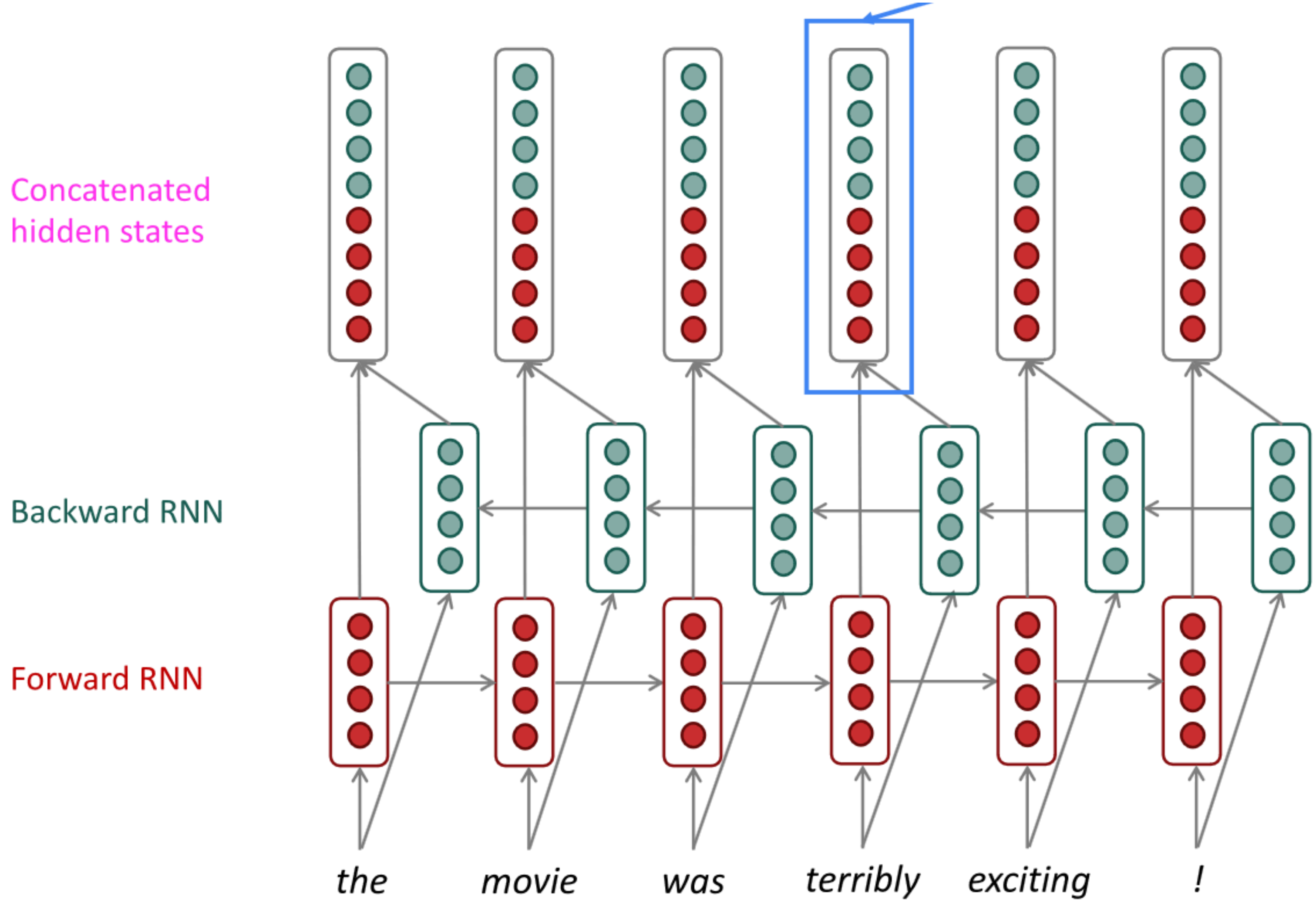
words / one-hot vectors

$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$



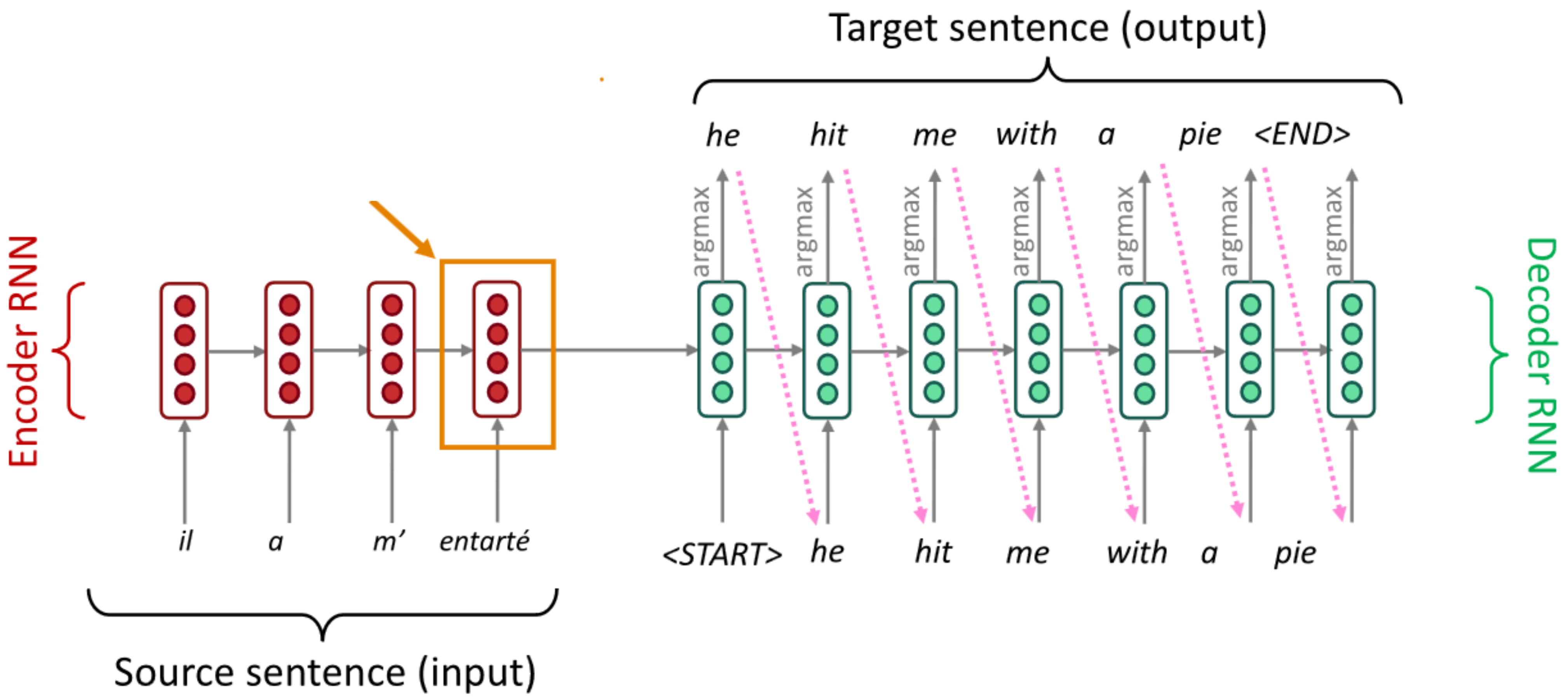
Slide from Stanford CS224

Recap: Recurrence



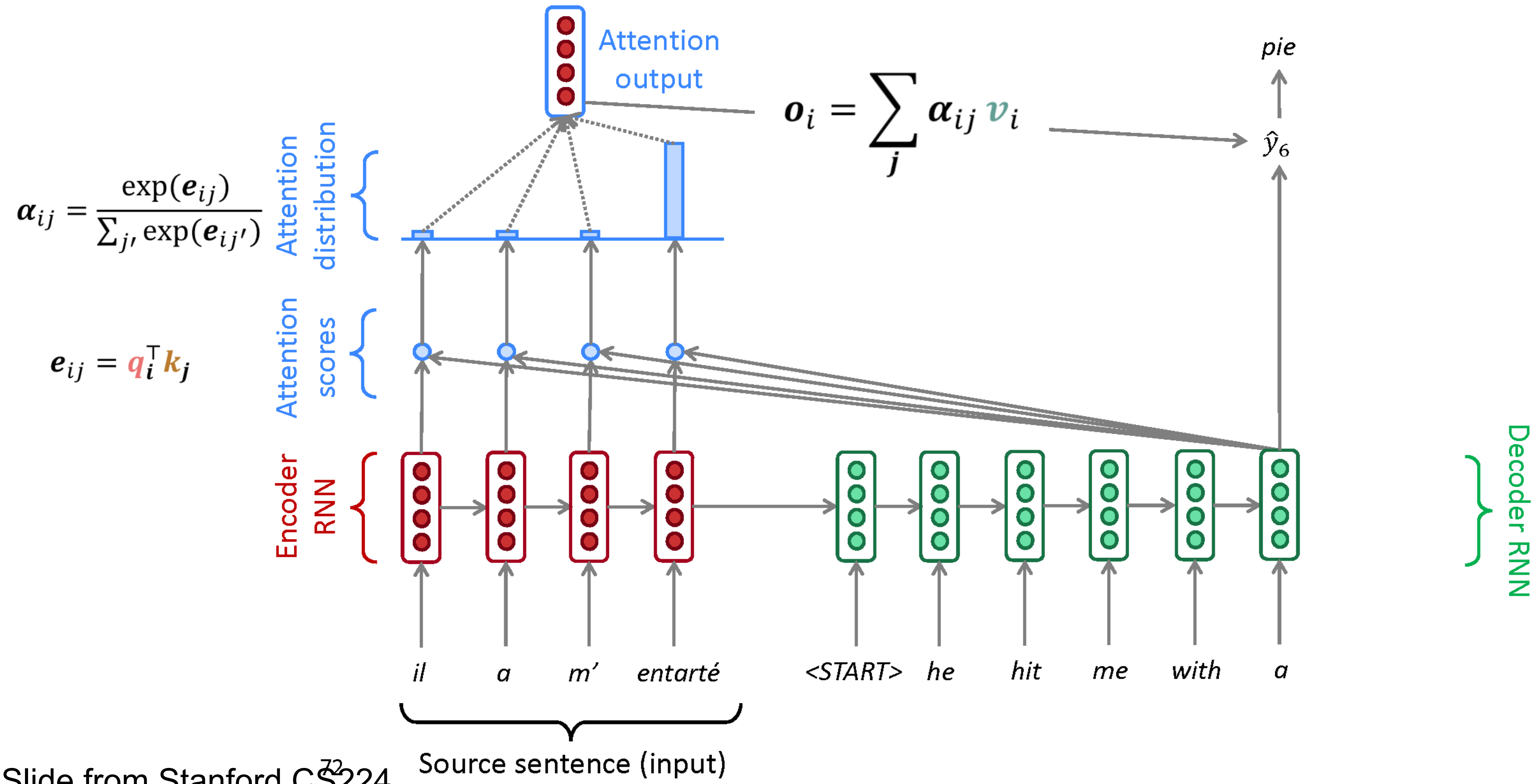
Slide from Stanford CS224

Recap: Recurrence



Slide from Stanford CS224

Recap: Attention



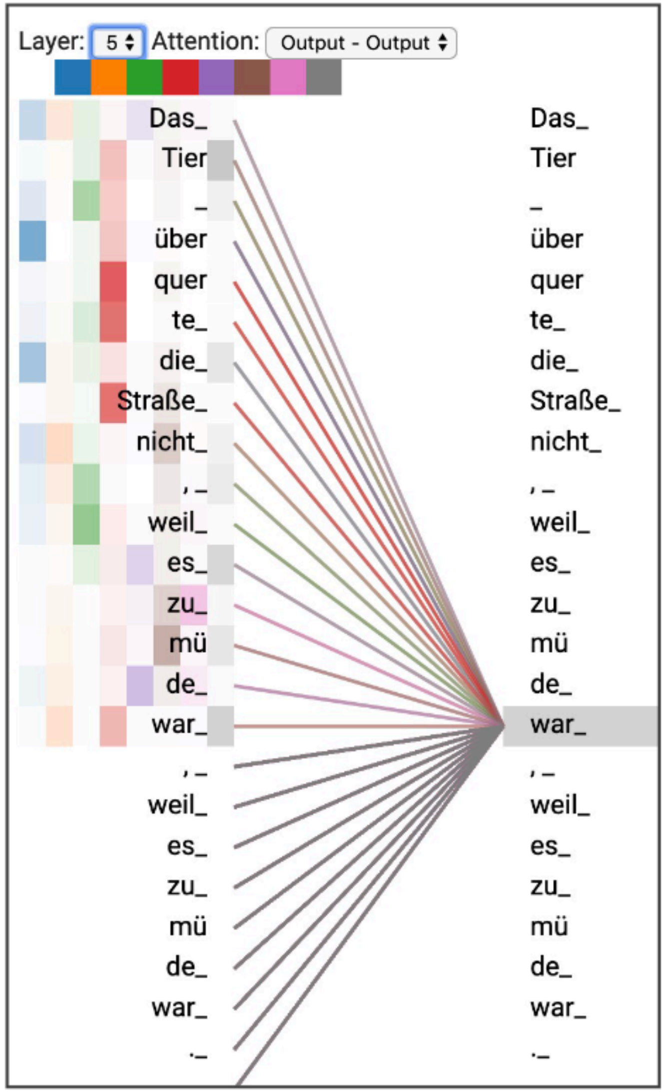
Recap: Attention

- Generic dot-product attention:

$$\boldsymbol{e}_{ij} = \boldsymbol{q}_i^{\top} \boldsymbol{k}_j \qquad \boldsymbol{\alpha}_{ij} = \frac{\exp(\boldsymbol{e}_{ij})}{\sum_{j'} \exp(\boldsymbol{e}_{ij'})} \qquad \boldsymbol{o}_i = \sum_j \boldsymbol{\alpha}_{ij} \boldsymbol{v}_i$$

- Self-attention: queries, keys, and values are all different transformations of the same item-level representation of some sequence:

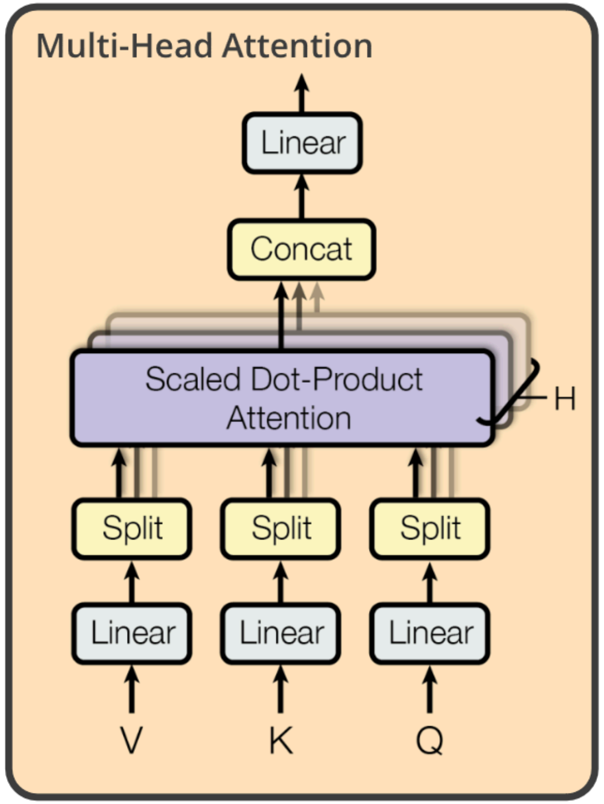
$$\boldsymbol{q}_i = Q\boldsymbol{x}_i \text{ (queries)} \qquad \boldsymbol{k}_i = K\boldsymbol{x}_i \text{ (keys)} \qquad \boldsymbol{v}_i = V\boldsymbol{x}_i \text{ (values)}$$



Slide from Stanford CS224 and CMU LLMs course

Multi-Head Attention

$\boldsymbol{q_i} = Q\boldsymbol{x_i}$ (queries) $\boldsymbol{k_i} = K\boldsymbol{x_i}$ (keys) $\boldsymbol{v_i} = V\boldsymbol{x_i}$ (values)



$$\mathrm{head}_1 = \mathrm{Attention}(\mathbf{Q}\mathbf{W}_1^Q, \mathbf{K}\mathbf{W}_1^K, \mathbf{V}\mathbf{W}_1^V)$$

$$\vdots$$

$$\mathrm{head}_H = \mathrm{Attention}(\mathbf{Q}\mathbf{W}_H^Q, \mathbf{K}\mathbf{W}_H^K, \mathbf{V}\mathbf{W}_H^V)$$

$$\mathrm{MultiHeadAtt}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_H)$$

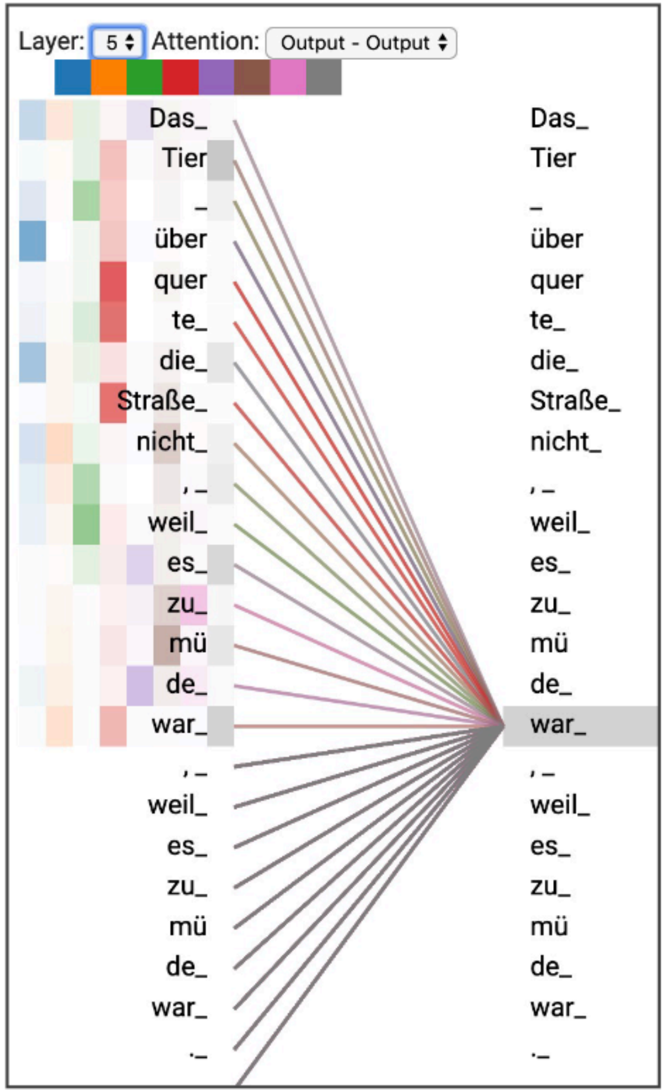
Inputs and outputs of each layer are the same dimensions:

$$\mathbf{Q} \in \mathbb{R}^{T \times d_{\mathrm{model}}}$$

$$\mathbf{K} \in \mathbb{R}^{T \times d_{\mathrm{model}}}$$

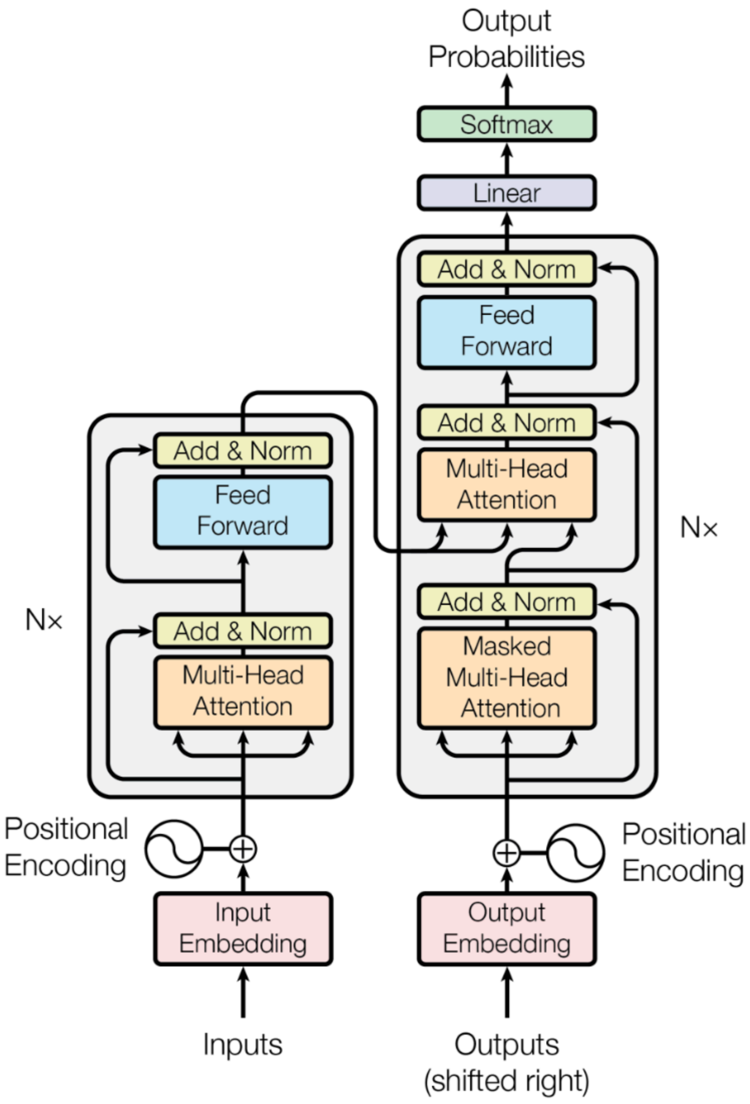
$$\mathbf{V} \in \mathbb{R}^{T \times d_{\mathrm{model}}}$$

$$\mathrm{MultiHeadAtt}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \in \mathbb{R}^{T \times d_{\mathrm{model}}}$$



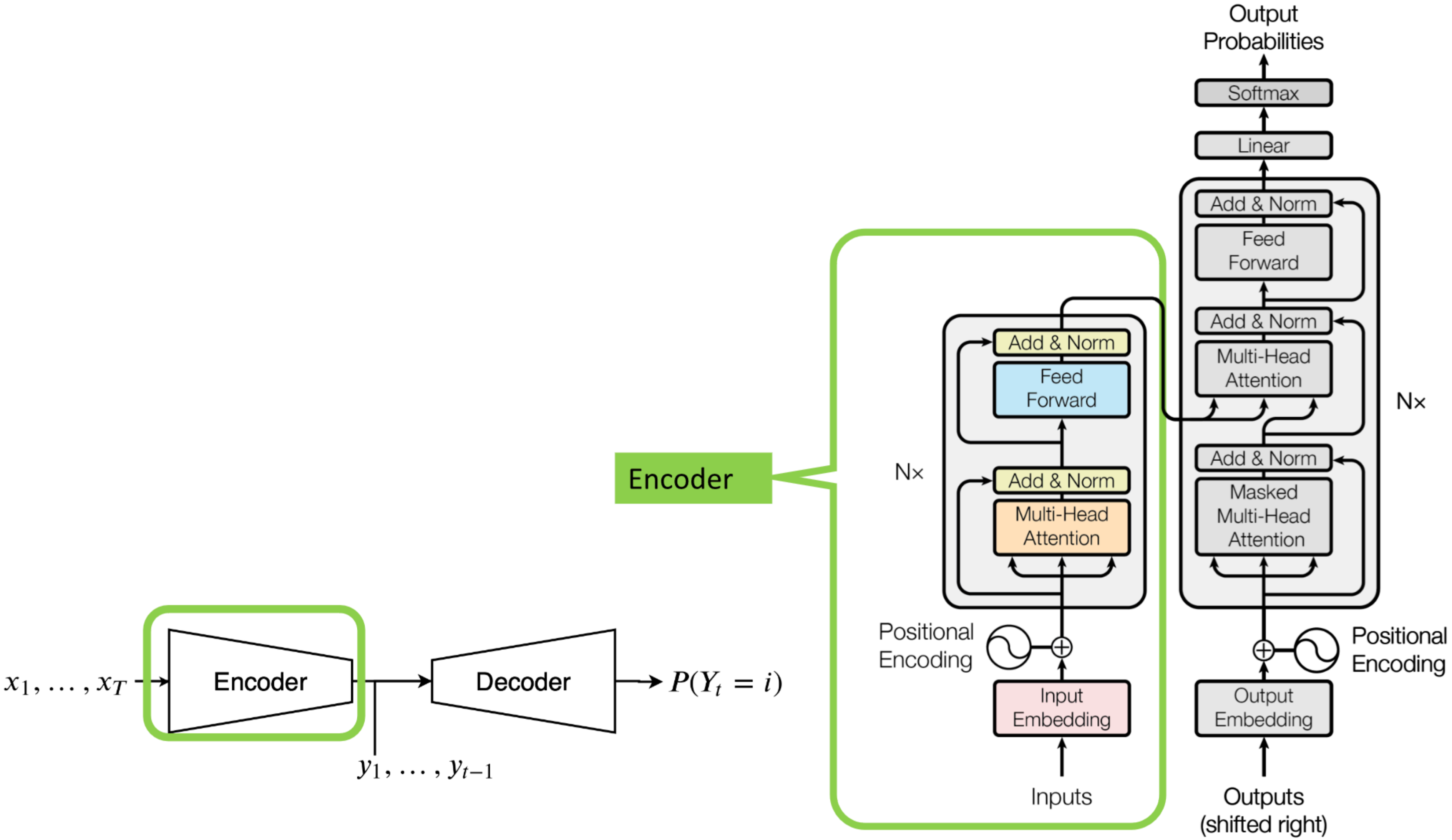
Slide from Stanford CS224 and CMU LLMs course

Transformer



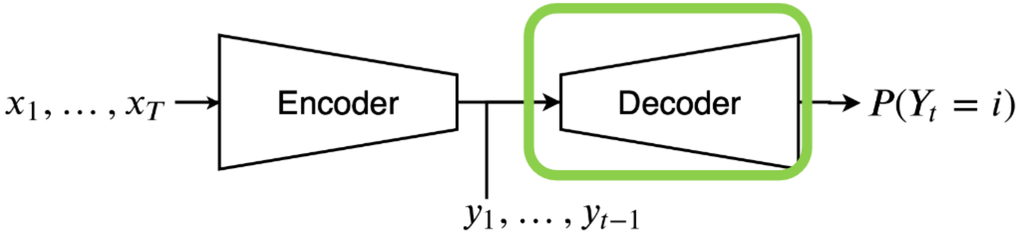
Vaswani et al. 2017, slide from CMU LLMs course

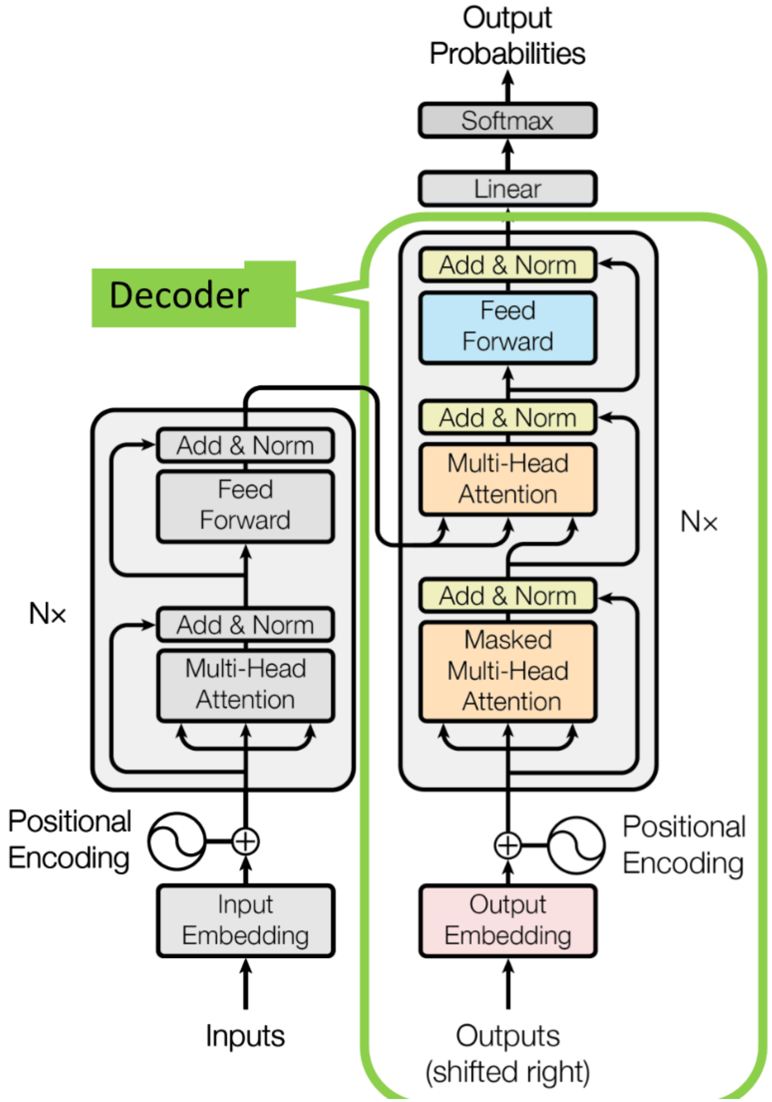
Encoder



Vaswani et al. 2017, slide from CMU LLMs course

Decoder



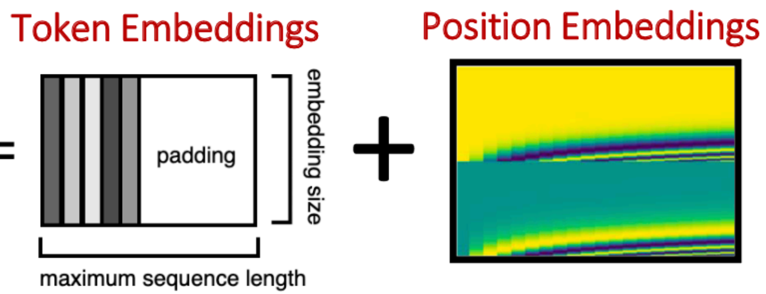


Vaswani et al. 2017, slide from CMU LLMs course

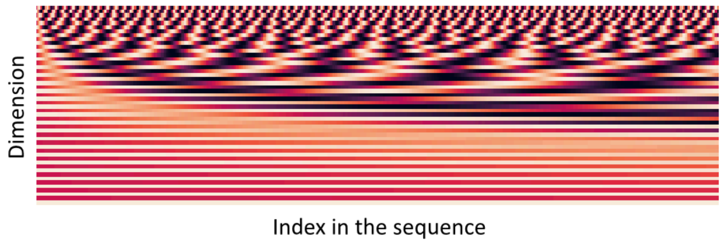
Encoder Input

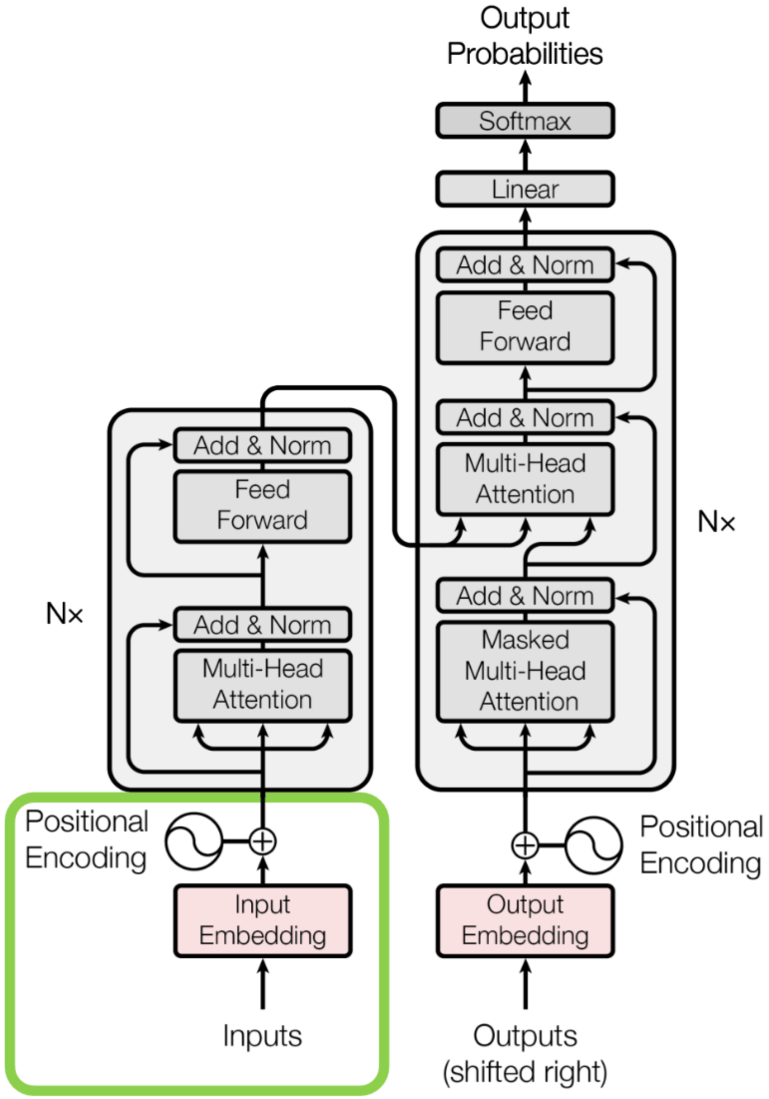
The input into the encoder looks like:

$$\mathbf{H}_0^{\text{enc}} =$$



$$\boldsymbol{p}_i = \begin{pmatrix} \sin(i/10000^{2*1/d}) \\ \cos(i/10000^{2*1/d}) \\ \vdots \\ \sin(i/10000^{2*\frac{d}{2}/d}) \\ \cos(i/10000^{2*\frac{d}{2}/d}) \end{pmatrix}$$

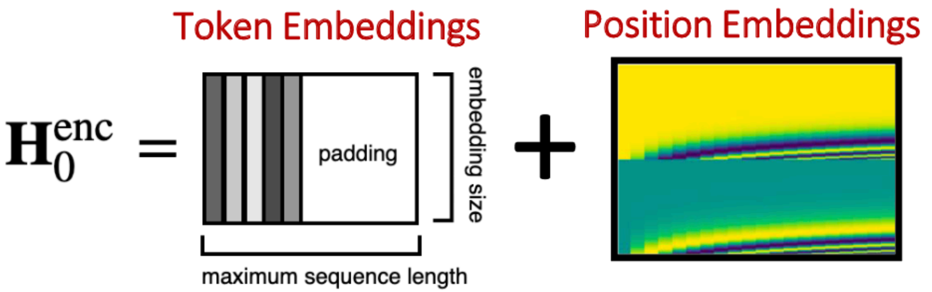




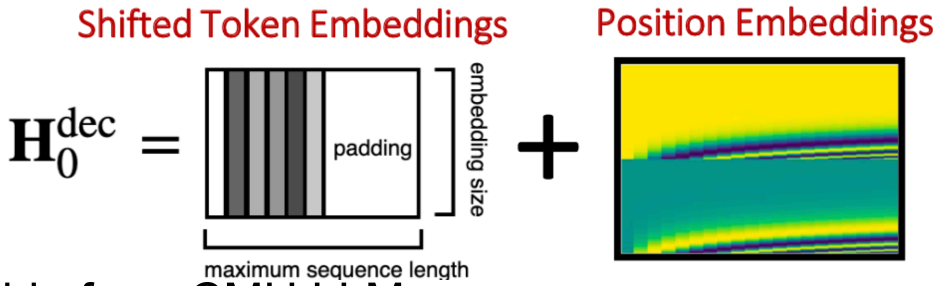
Vaswani et al. 2017, slide from CMU LLMs course and Stanford CS 224

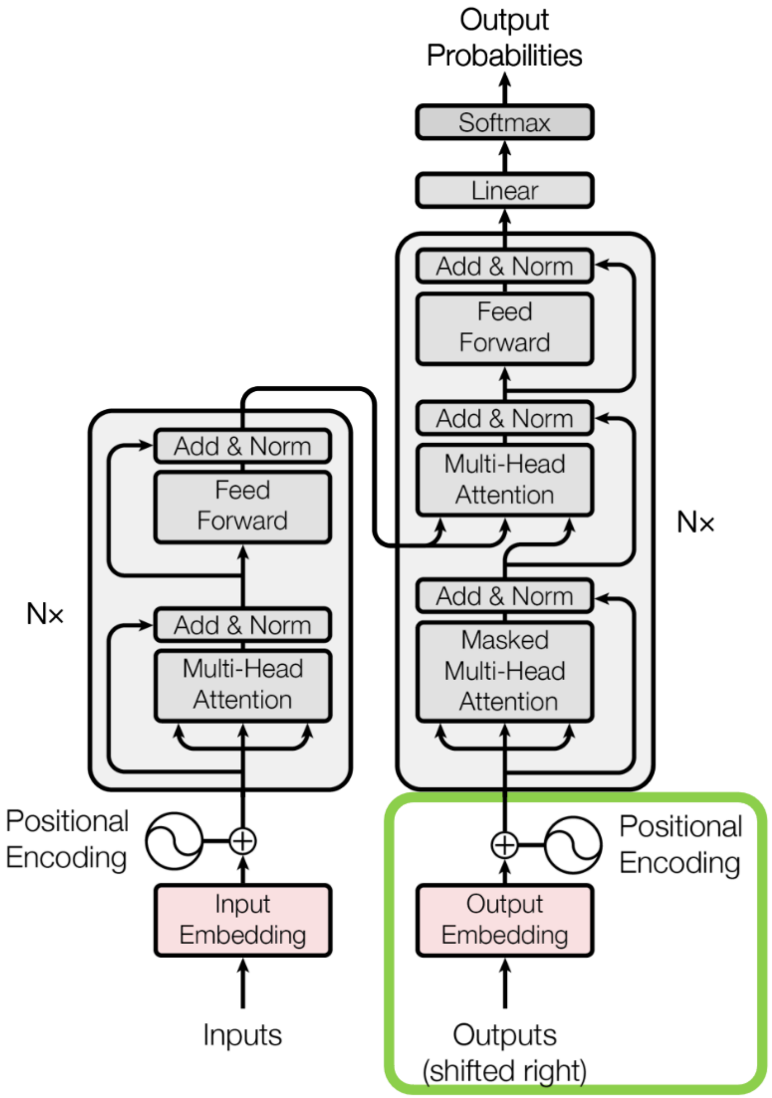
Decoder Input

The input into the encoder looks like:



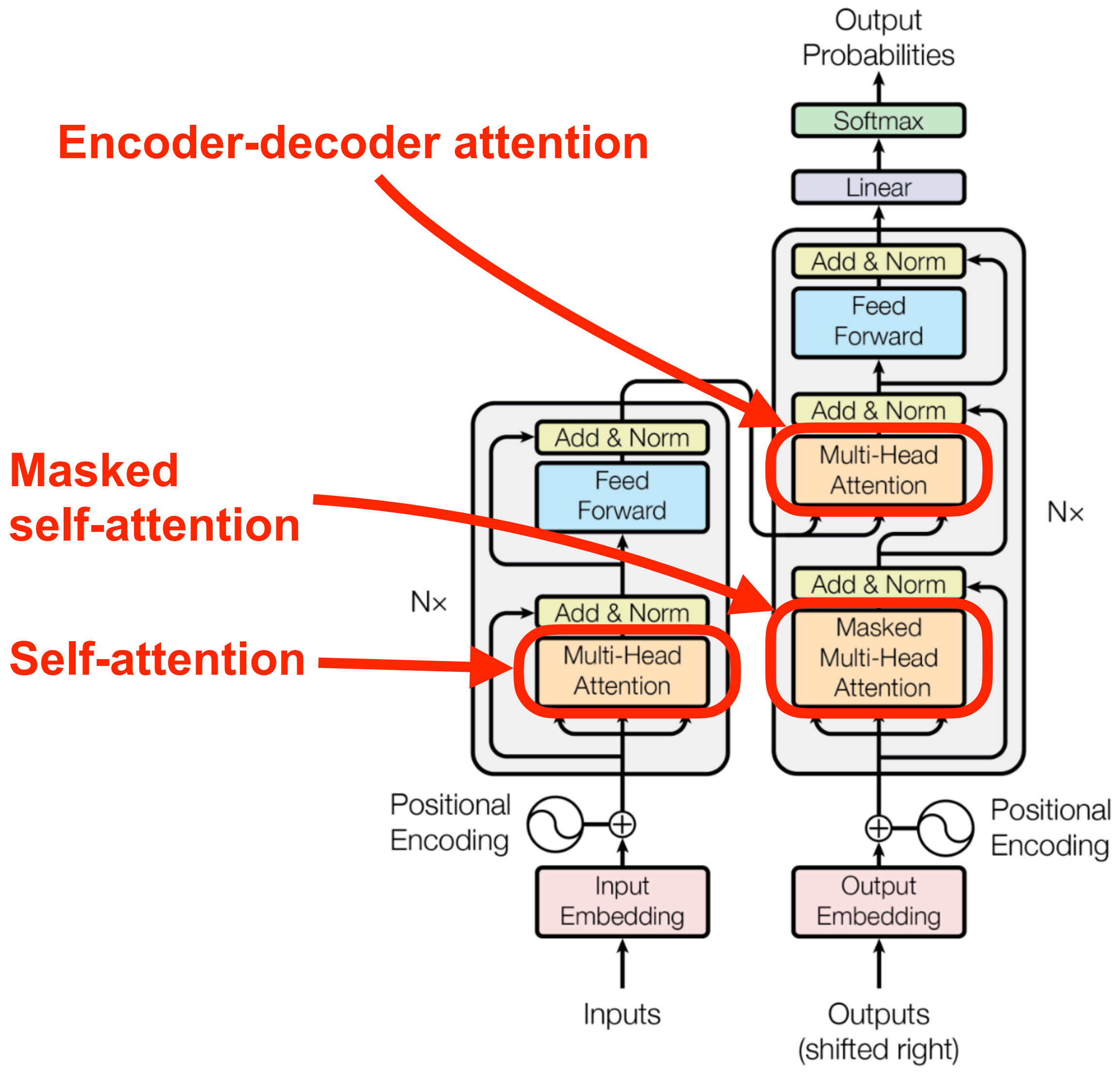
The input to the decoder looks like:





Vaswani et al. 2017, slide from CMU LLMs course

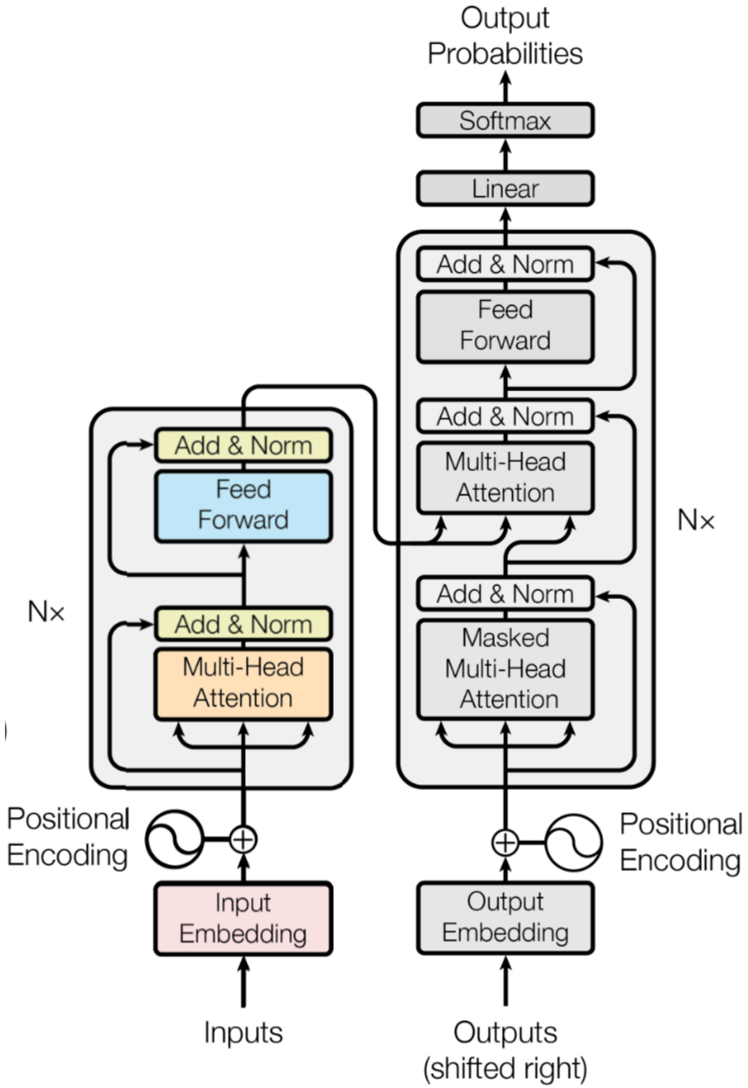
Attention



Vaswani et al. 2017, slide from CMU LLMs course

Encoder

$$\boxed{\text{Multi-Head Attention}} = \mathrm{MultiHeadAtt}(\mathbf{H}_i^{enc}, \mathbf{H}_i^{enc}, \mathbf{H}_i^{enc})$$



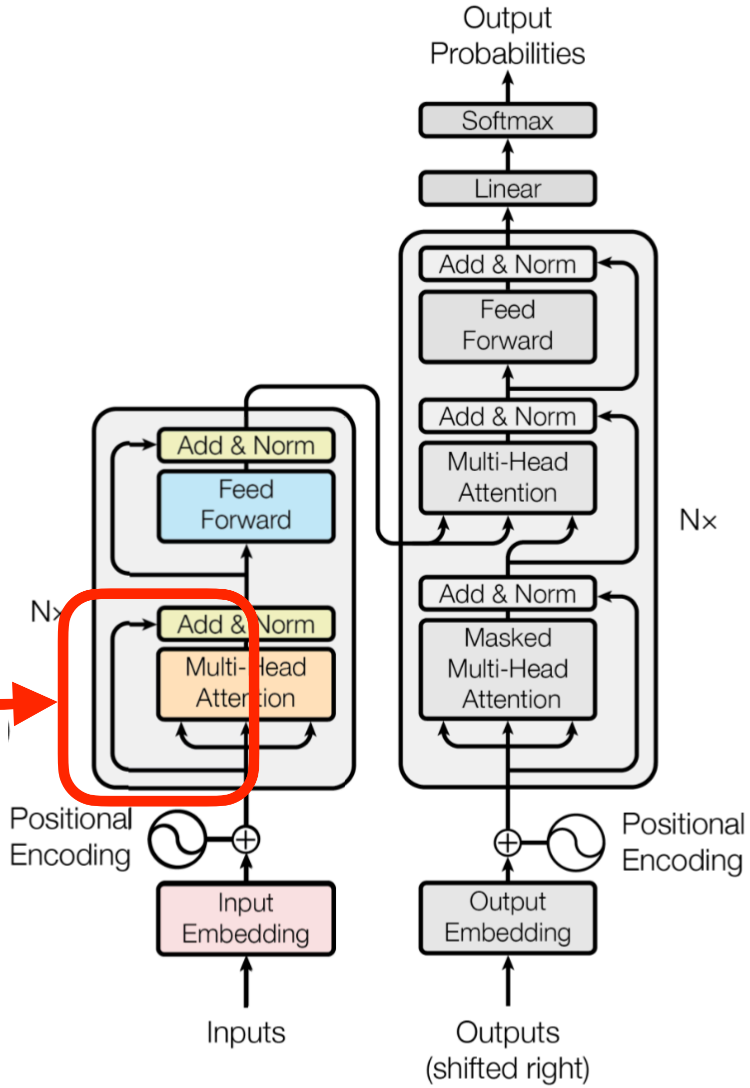
Vaswani et al. 2017, slide from CMU LLMs course

Encoder

$$\text{Multi-Head Attention} = \mathrm{MultiHeadAtt}(\mathbf{H}_i^{enc}, \mathbf{H}_i^{enc}, \mathbf{H}_i^{enc})$$

$$\text{Add \& Norm} = \mathrm{LayerNorm}(\text{Multi-Head Attention} \boxed{+\mathbf{H}_i^{enc})}$$

Residual connection



Vaswani et al. 2017, slide from CMU LLMs course

Encoder

Multi-Head Attention $= \mathrm{MultiHeadAtt}(\mathbf{H}_i^{enc}, \mathbf{H}_i^{enc}, \mathbf{H}_i^{enc})$

Add & Norm $= \mathrm{LayerNorm}($ Multi-Head Attention $+\mathbf{H}_i^{enc})$

Let $x \in \mathbb{R}^d$ be an individual (word) vector in the model.

Let $\mu = \sum_{j=1}^{d} x_j$; this is the mean; $\mu \in \mathbb{R}$.

Let $\sigma = \sqrt{\frac{1}{d}\sum_{j=1}^{d}(x_j - \mu)^2}$; this is the standard deviation; $\sigma \in \mathbb{R}$.

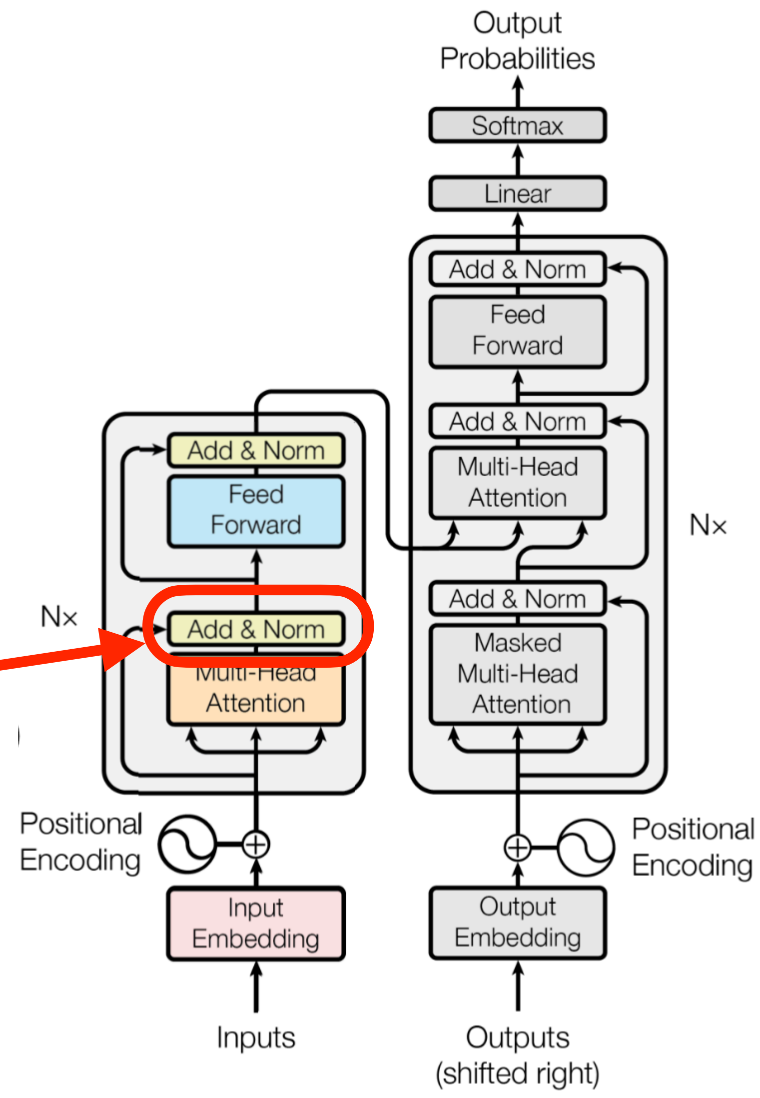
Let $\gamma \in \mathbb{R}^d$ and $\beta \in \mathbb{R}^d$ be learned "gain" and "bias" parameters. (Can omit!)

Then layer normalization computes:

$$\text{output} = \frac{x - \mu}{\sqrt{\sigma} + \epsilon} * \gamma + \beta$$

Normalize by scalar mean and variance

Modulate by learned elementwise gain and bias



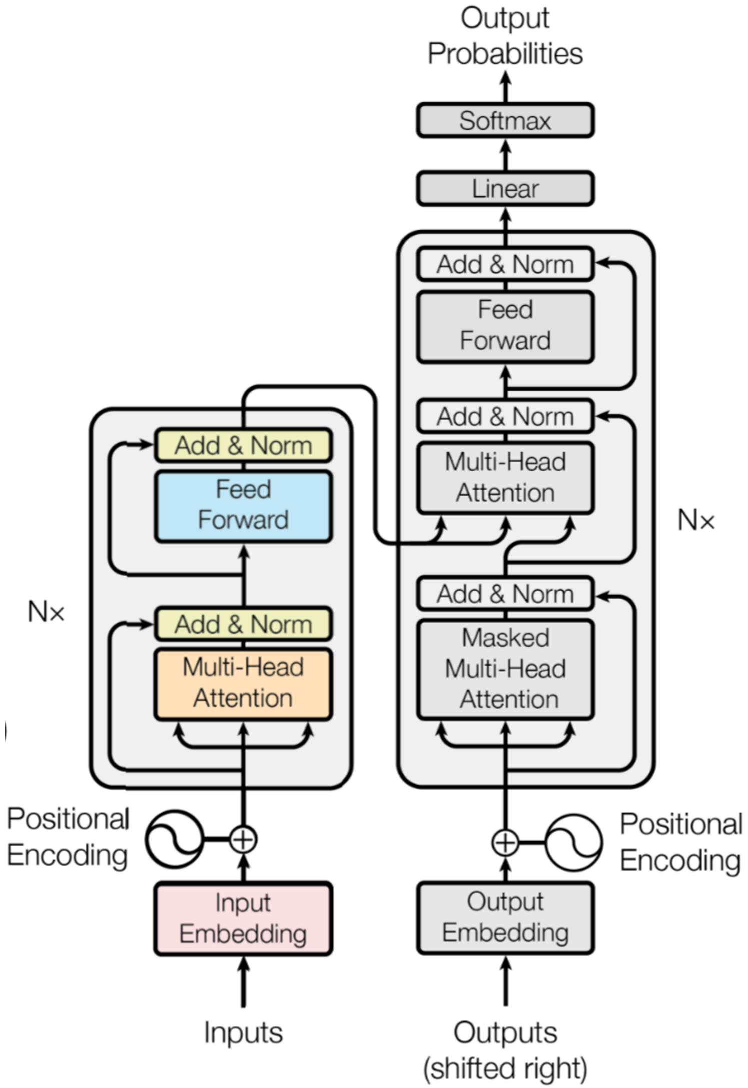
Vaswani et al. 2017, slide from CMU LLMs course

Encoder

$$\boxed{\text{Multi-Head Attention}} = \mathrm{MultiHeadAtt}(\mathbf{H}_i^{enc}, \mathbf{H}_i^{enc}, \mathbf{H}_i^{enc})$$

$$\boxed{\text{Add \& Norm}} = \mathrm{LayerNorm}(\boxed{\text{Multi-Head Attention}} + \mathbf{H}_i^{enc})$$

$$\boxed{\text{Feed Forward}} = \max(0, \boxed{\text{Add \& Norm}}\,\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2)$$



Vaswani et al. 2017, slide from CMU LLMs course

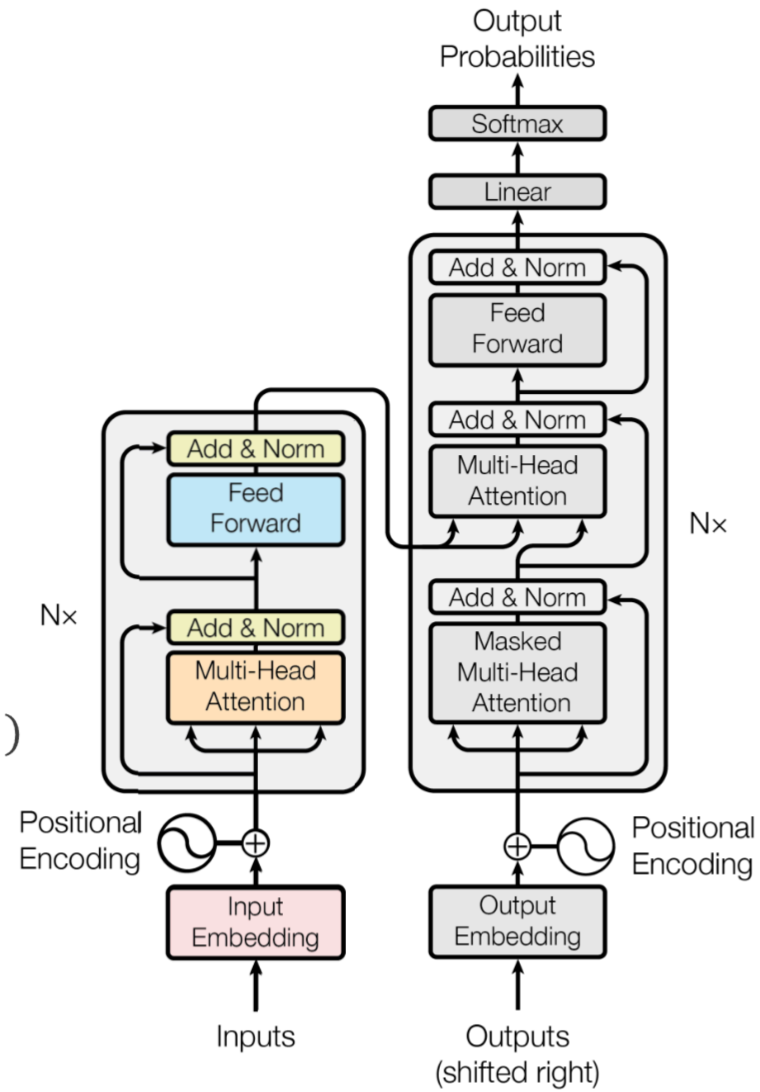
Encoder

$$\boxed{\text{Multi-Head Attention}} = \mathrm{MultiHeadAtt}(\mathbf{H}_i^{enc}, \mathbf{H}_i^{enc}, \mathbf{H}_i^{enc})$$

$$\boxed{\text{Add \& Norm}} = \mathrm{LayerNorm}(\boxed{\text{Multi-Head Attention}} + \mathbf{H}_i^{enc})$$

$$\boxed{\text{Feed Forward}} = \max(0, \boxed{\text{Add \& Norm}}\,\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2)$$

$$\boxed{\text{Add \& Norm (2)}} = \mathrm{LayerNorm}(\boxed{\text{Feed Forward}} + \boxed{\text{Add \& Norm}})$$



Vaswani et al. 2017, slide from CMU LLMs course

Encoder

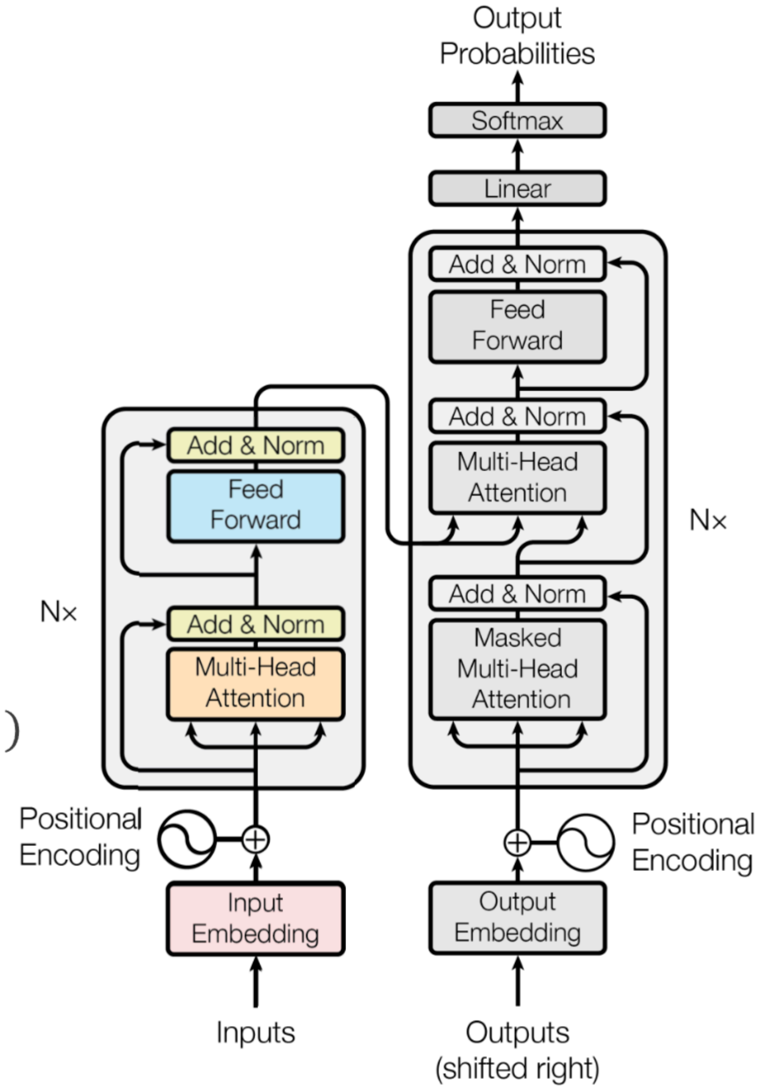
$$\boxed{\text{Multi-Head Attention}} = \mathrm{MultiHeadAtt}(\mathbf{H}_i^{enc}, \mathbf{H}_i^{enc}, \mathbf{H}_i^{enc})$$

$$\boxed{\text{Add \& Norm}} = \mathrm{LayerNorm}(\boxed{\text{Multi-Head Attention}} + \mathbf{H}_i^{enc})$$

$$\boxed{\text{Feed Forward}} = \max(0, \boxed{\text{Add \& Norm}}\,\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2)$$

$$\boxed{\text{Add \& Norm (2)}} = \mathrm{LayerNorm}(\boxed{\text{Feed Forward}} + \boxed{\text{Add \& Norm}})$$

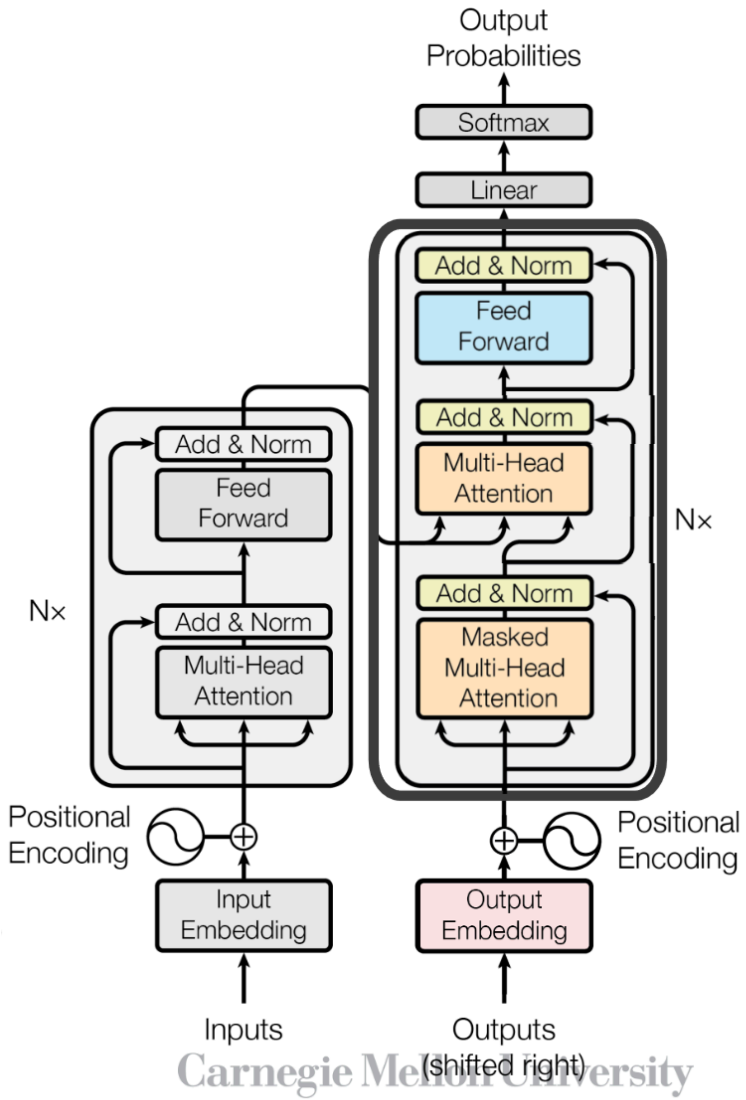
$$\mathbf{H}_{i+1}^{enc} = \boxed{\text{Add \& Norm(2)}}$$



Vaswani et al. 2017, slide from CMU LLMs course

Decoder

$$\boxed{\text{Masked Multi-Head Attention}} = \text{MultiHeadAtt}(\mathbf{H}_i^{dec}, \mathbf{H}_i^{dec}, \mathbf{H}_i^{dec})$$

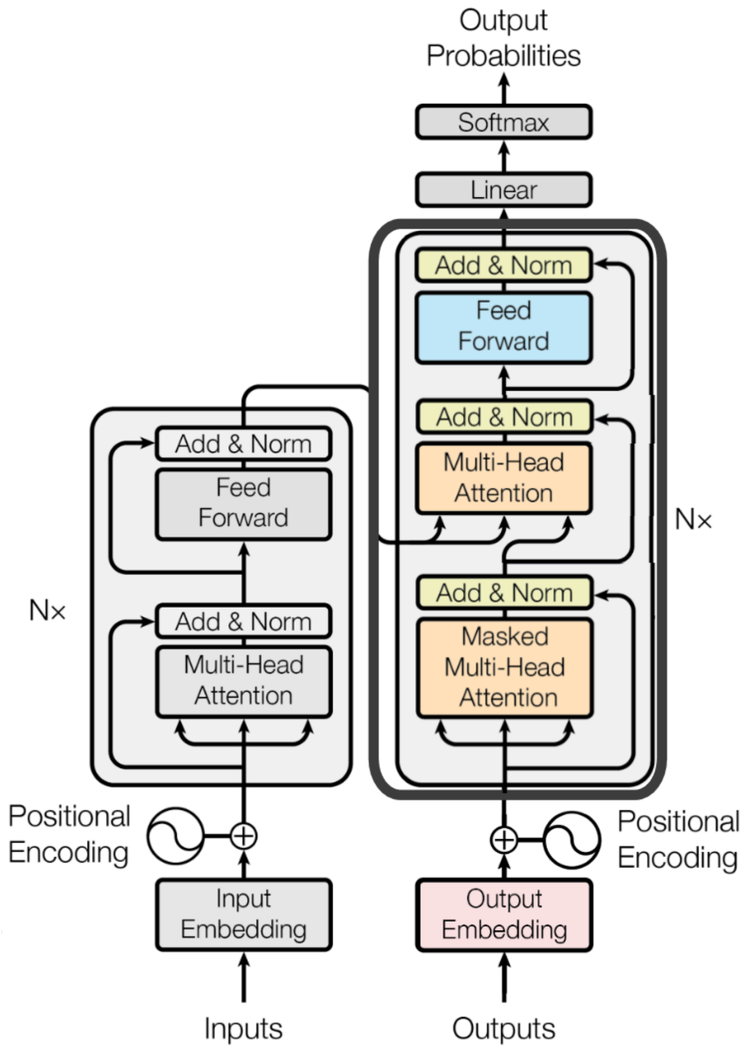


Vaswani et al. 2017, slide from CMU LLMs course

Decoder

$$\boxed{\text{Masked Multi-Head Attention}} = \mathrm{MultiHeadAtt}(\mathbf{H}_i^{dec}, \mathbf{H}_i^{dec}, \mathbf{H}_i^{dec})$$

$$\boxed{\text{Add \& Norm}} = \mathrm{LayerNorm}(\boxed{\text{Masked Multi-Head Attention}} + \mathbf{H}_i^{dec})$$



Carnegie Mellon University

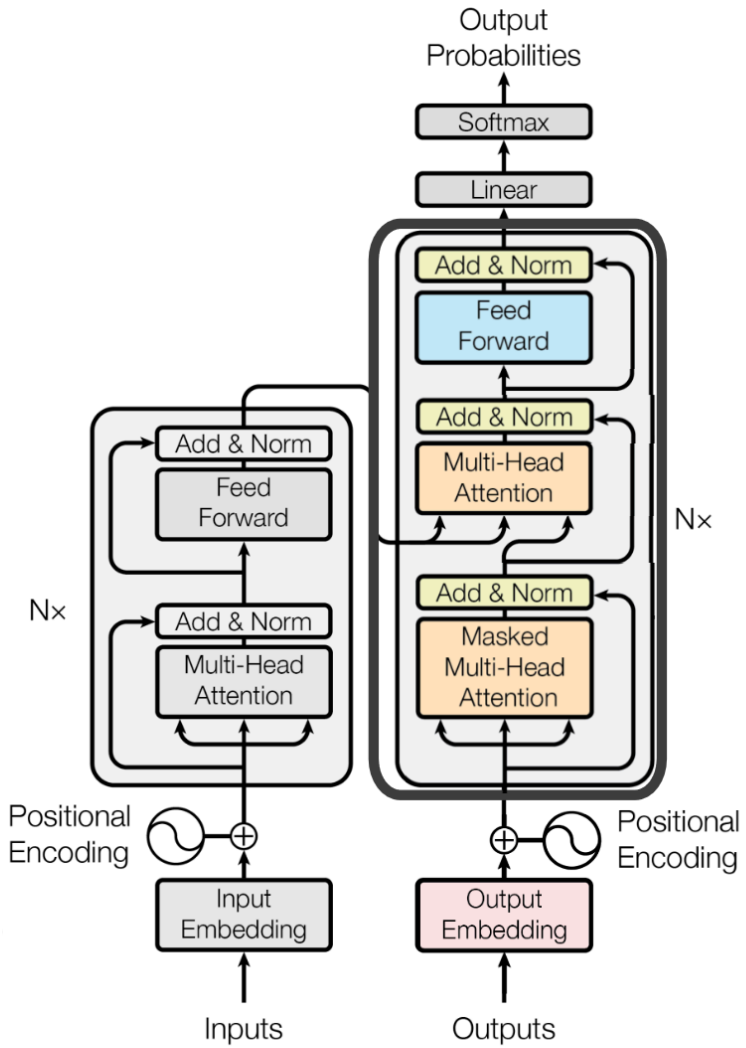
Vaswani et al. 2017, slide from CMU LLMs course

Decoder

$$\boxed{\text{Masked Multi-Head Attention}} = \mathrm{MultiHeadAtt}(\mathbf{H}_i^{dec}, \mathbf{H}_i^{dec}, \mathbf{H}_i^{dec})$$

$$\boxed{\text{Add \& Norm}} = \mathrm{LayerNorm}(\boxed{\text{Masked Multi-Head Attention}} + \mathbf{H}_i^{dec})$$

$$\boxed{\text{Enc-Dec Multi-Head Attention}} = \mathrm{MultiHeadAtt}(\mathbf{H}_i^{enc}, \mathbf{H}_i^{enc}, \boxed{\text{Add \& Norm}})$$



Vaswani et al. 2017, slide from CMU LLMs course

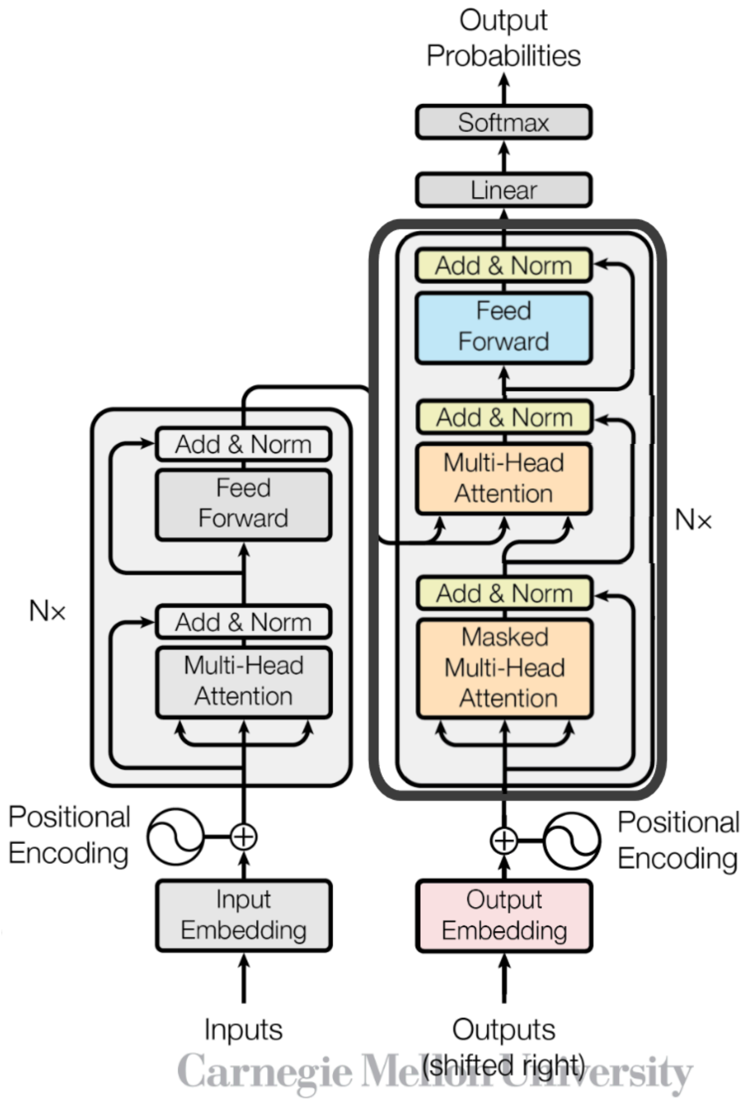
Decoder

$$\boxed{\text{Masked Multi-Head Attention}} = \mathrm{MultiHeadAtt}(\mathbf{H}_i^{dec}, \mathbf{H}_i^{dec}, \mathbf{H}_i^{dec})$$

$$\boxed{\text{Add \& Norm}} = \mathrm{LayerNorm}(\boxed{\text{Masked Multi-Head Attention}} + \mathbf{H}_i^{dec})$$

$$\boxed{\text{Enc-Dec Multi-Head Attention}} = \mathrm{MultiHeadAtt}(\mathbf{H}_i^{enc}, \mathbf{H}_i^{enc}, \boxed{\text{Add \& Norm}})$$

$$\boxed{\text{Add \& Norm (2)}} = \mathrm{LayerNorm}(\boxed{\text{Enc-Dec Multi-Head Attention}} + \boxed{\text{Add \& Norm}})$$



Vaswani et al. 2017, slide from CMU LLMs course

Decoder

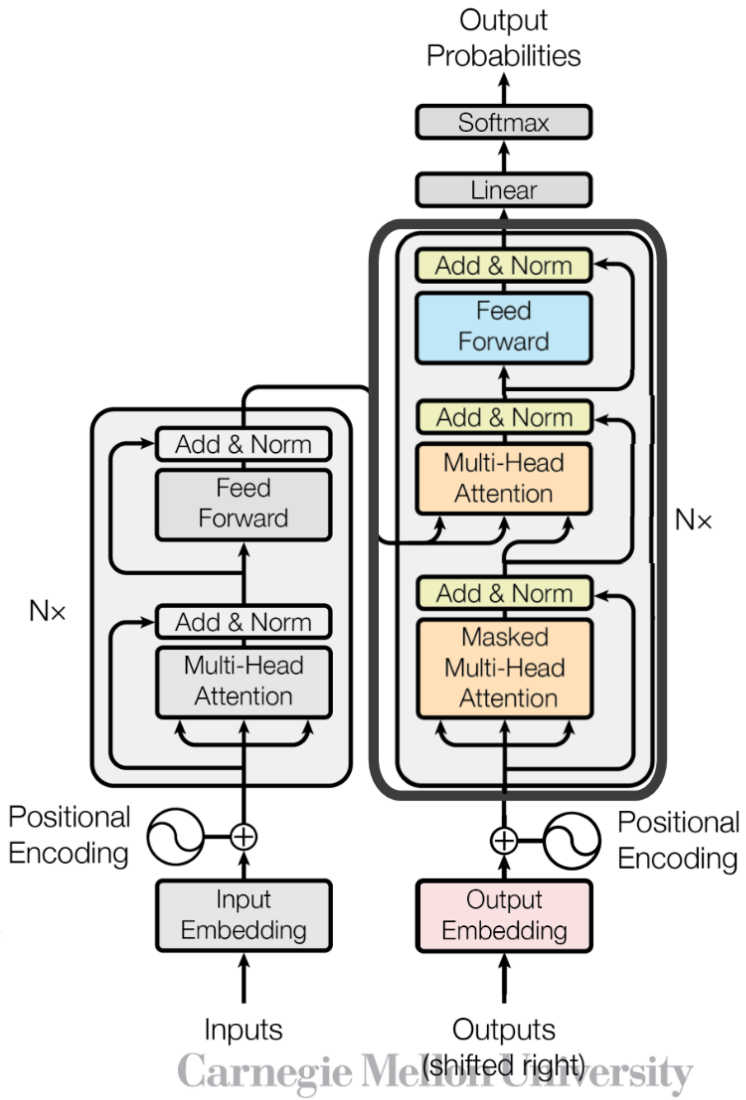
$$\boxed{\text{Masked Multi-Head Attention}} = \mathrm{MultiHeadAtt}(\mathbf{H}_i^{dec}, \mathbf{H}_i^{dec}, \mathbf{H}_i^{dec})$$

$$\boxed{\text{Add \& Norm}} = \mathrm{LayerNorm}(\boxed{\text{Masked Multi-Head Attention}} + \mathbf{H}_i^{dec})$$

$$\boxed{\text{Enc-Dec Multi-Head Attention}} = \mathrm{MultiHeadAtt}(\mathbf{H}_i^{enc}, \mathbf{H}_i^{enc}, \boxed{\text{Add \& Norm}})$$

$$\boxed{\text{Add \& Norm (2)}} = \mathrm{LayerNorm}(\boxed{\text{Enc-Dec Multi-Head Attention}} + \boxed{\text{Add \& Norm}})$$

$$\boxed{\text{Feed Forward}} = \max(0, \boxed{\text{Add \& Norm (2)}}\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2)$$



Decoder

$$\boxed{\text{Masked Multi-Head Attention}} = \text{MultiHeadAtt}(\mathbf{H}_i^{dec}, \mathbf{H}_i^{dec}, \mathbf{H}_i^{dec})$$

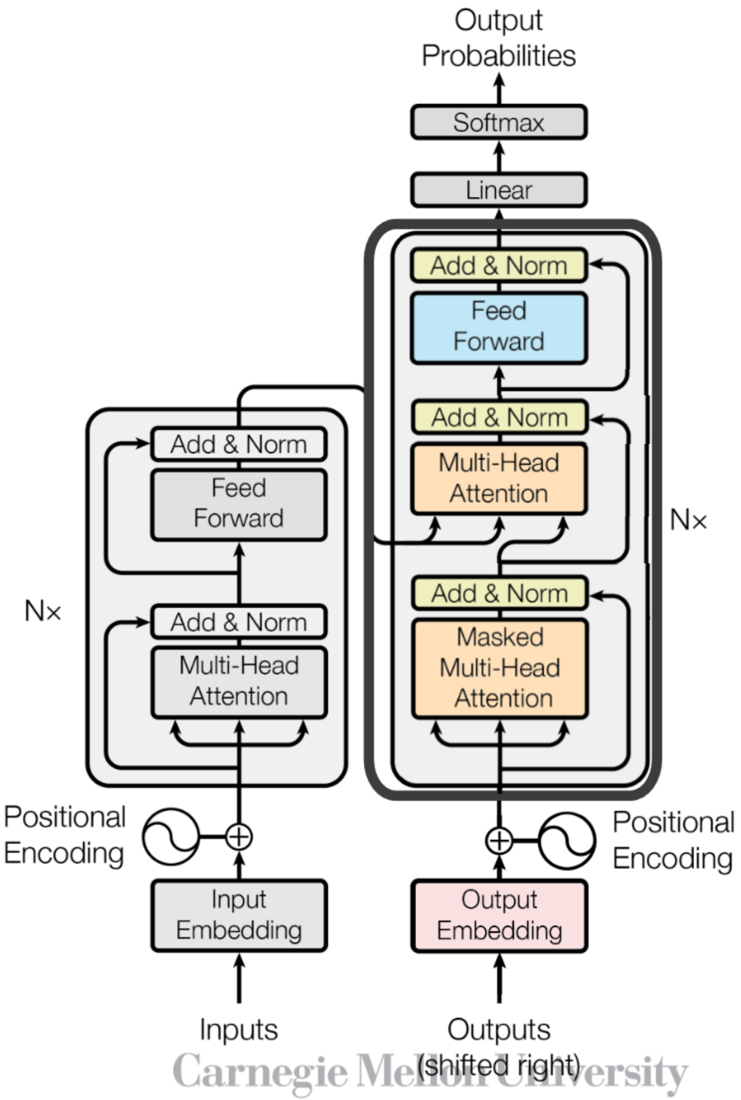
$$\boxed{\text{Add \& Norm}} = \text{LayerNorm}(\boxed{\text{Masked Multi-Head Attention}} + \mathbf{H}_i^{dec})$$

$$\boxed{\text{Enc-Dec Multi-Head Attention}} = \text{MultiHeadAtt}(\mathbf{H}_i^{enc}, \mathbf{H}_i^{enc}, \boxed{\text{Add \& Norm}})$$

$$\boxed{\text{Add \& Norm (2)}} = \text{LayerNorm}(\boxed{\text{Enc-Dec Multi-Head Attention}} + \boxed{\text{Add \& Norm}})$$

$$\boxed{\text{Feed Forward}} = \max(0, \boxed{\text{Add \& Norm (2)}}\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2)$$

$$\boxed{\text{Add \& Norm (3)}} = \text{LayerNorm}(\boxed{\text{Feed Forward}} + \boxed{\text{Add \& Norm (2)}})$$



Vaswani et al. 2017, slide from CMU LLMs course

Decoder

$$\boxed{\text{Masked Multi-Head Attention}} = \mathrm{MultiHeadAtt}(\mathbf{H}_i^{dec}, \mathbf{H}_i^{dec}, \mathbf{H}_i^{dec})$$

$$\boxed{\text{Add \& Norm}} = \mathrm{LayerNorm}(\boxed{\text{Masked Multi-Head Attention}} + \mathbf{H}_i^{dec})$$

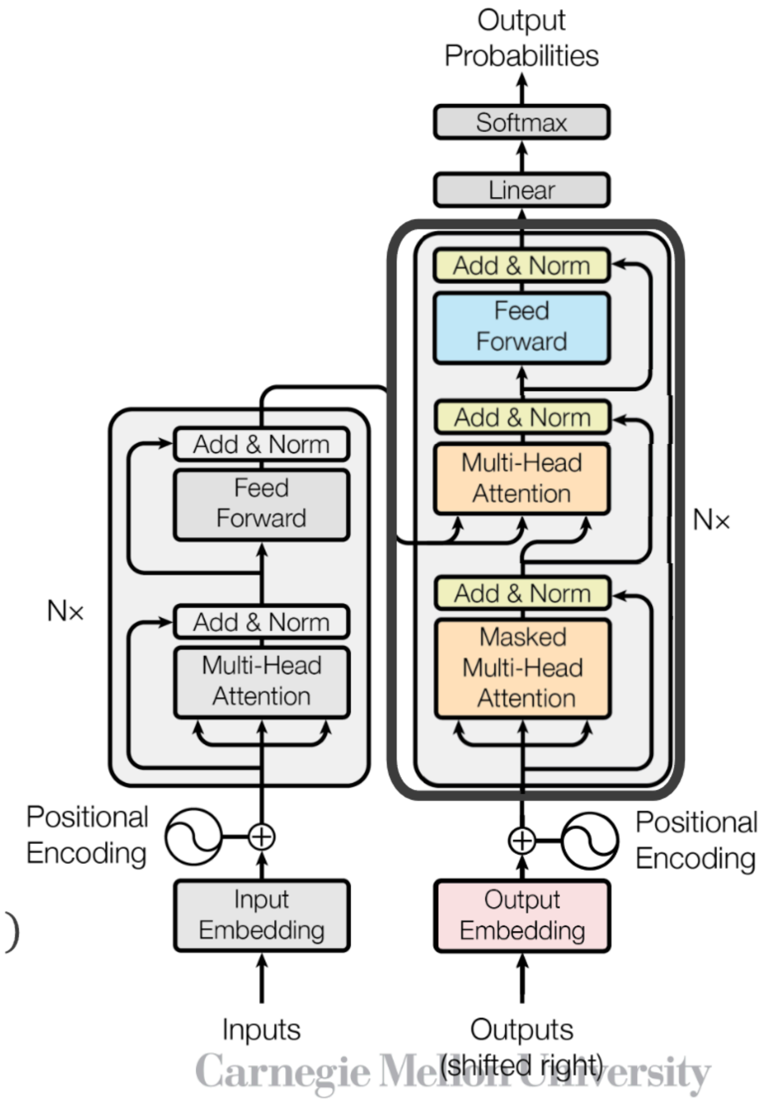
$$\boxed{\text{Enc-Dec Multi-Head Attention}} = \mathrm{MultiHeadAtt}(\mathbf{H}_i^{enc}, \mathbf{H}_i^{enc}, \boxed{\text{Add \& Norm}})$$

$$\boxed{\text{Add \& Norm (2)}} = \mathrm{LayerNorm}(\boxed{\text{Enc-Dec Multi-Head Attention}} + \boxed{\text{Add \& Norm}})$$

$$\boxed{\text{Feed Forward}} = \max(0, \boxed{\text{Add \& Norm (2)}}\,\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2$$

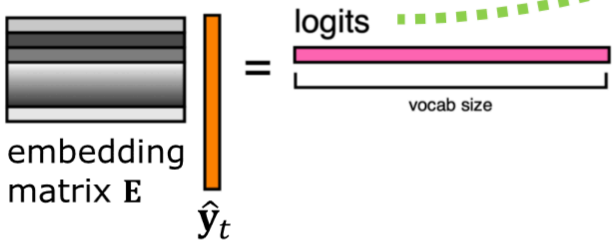
$$\boxed{\text{Add \& Norm (3)}} = \mathrm{LayerNorm}(\boxed{\text{Feed Forward}} + \boxed{\text{Add \& Norm (2)}})$$

$$\mathbf{H}_{i+1}^{\text{dec}} = \boxed{\text{Add \& Norm(3)}}$$

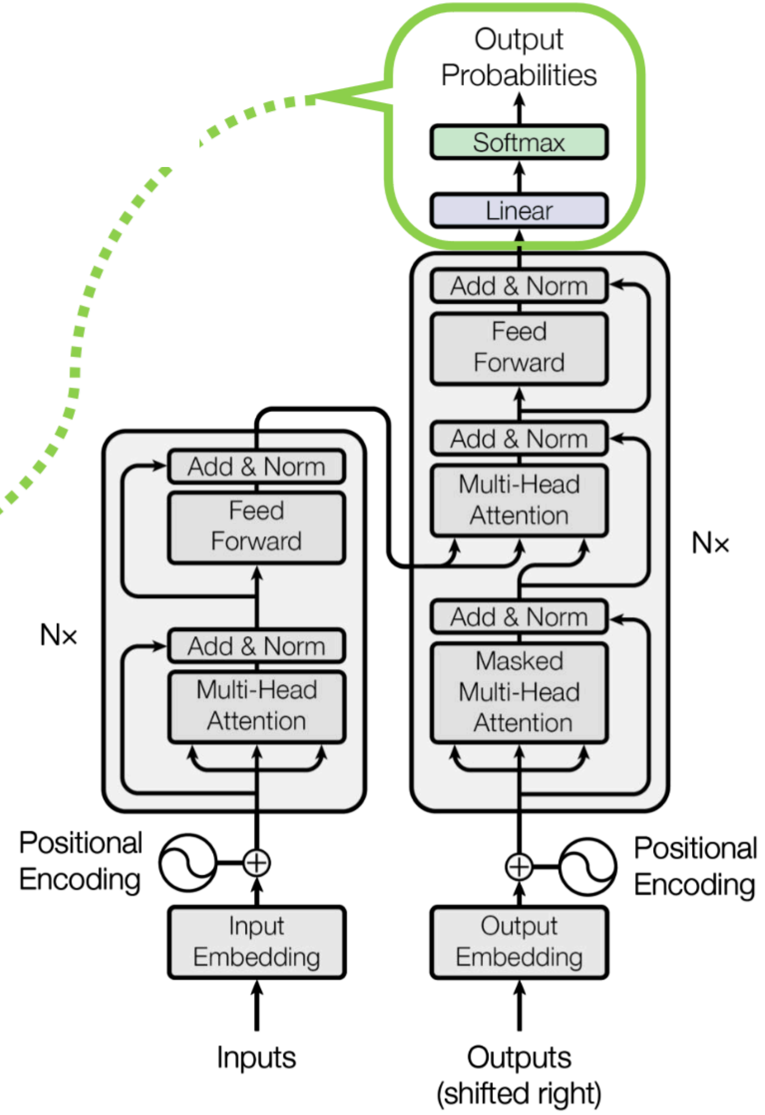


Vaswani et al. 2017, slide from CMU LLMs course

Output Probabilities



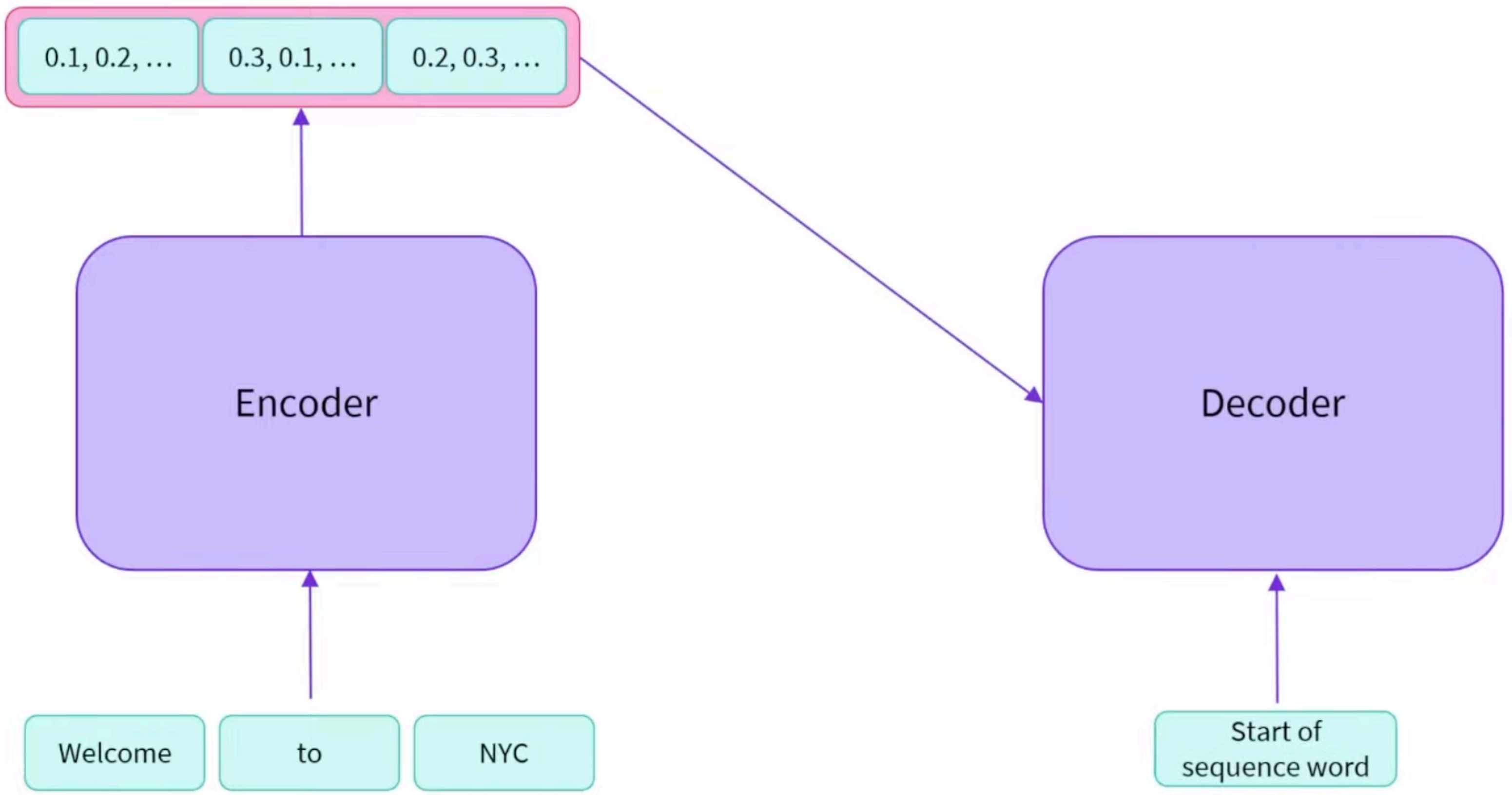
$$P(Y_t = i | \mathbf{x}_{1:T}, \mathbf{y}_{1:t-1}) = \frac{\exp(\mathbf{E}\hat{\mathbf{y}}_t[i])}{\sum_j \exp(\mathbf{E}\hat{\mathbf{y}}_t[j])}$$



Vaswani et al. 2017, slide from CMU LLMs course

Encoder-Decoder Inference

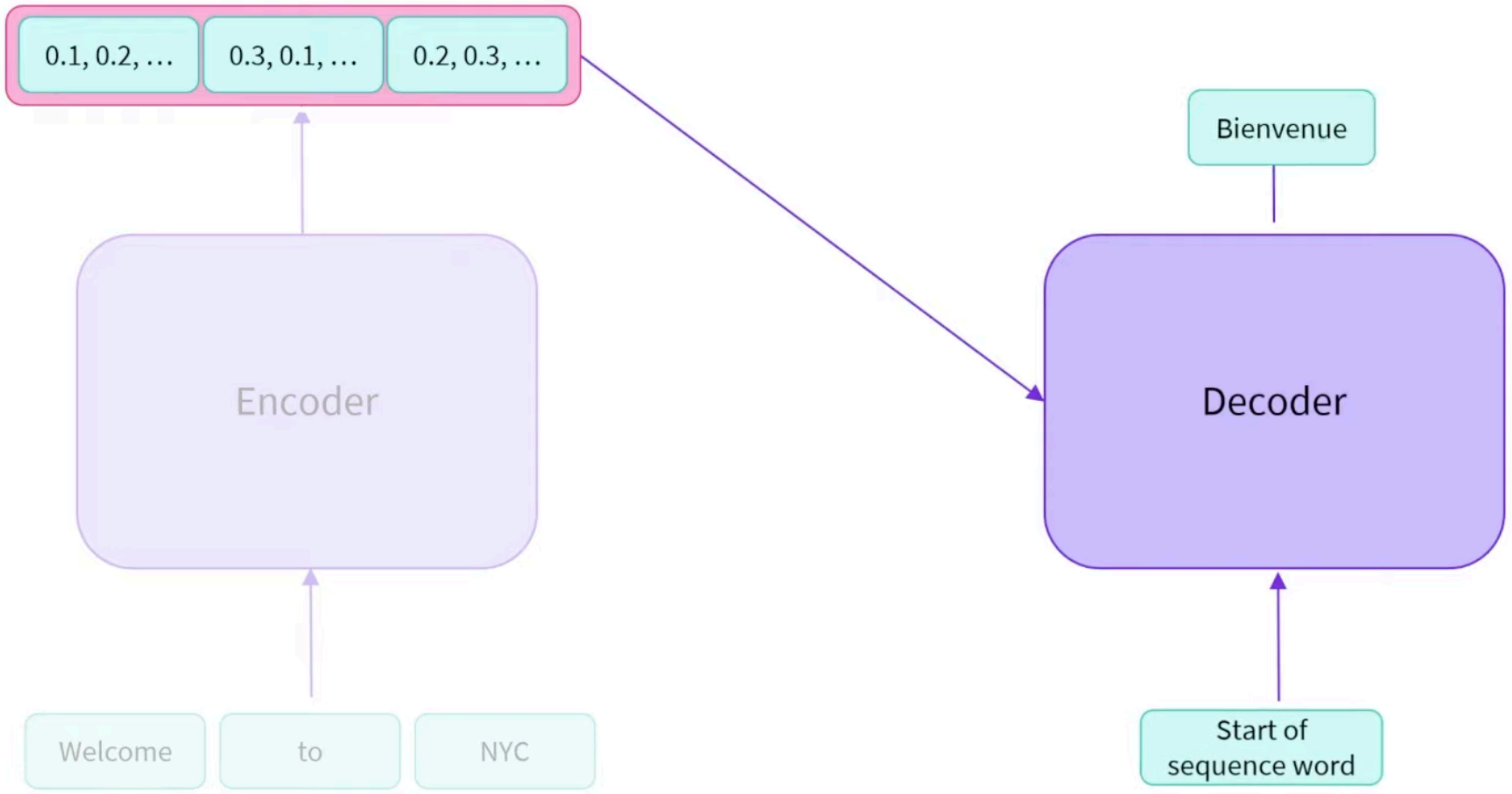
- Encode input sequence



slide from HuggingFace Transformers course: https://www.youtube.com/watch?v=0_4KEb08xrE&t=204s

Encoder-Decoder Inference

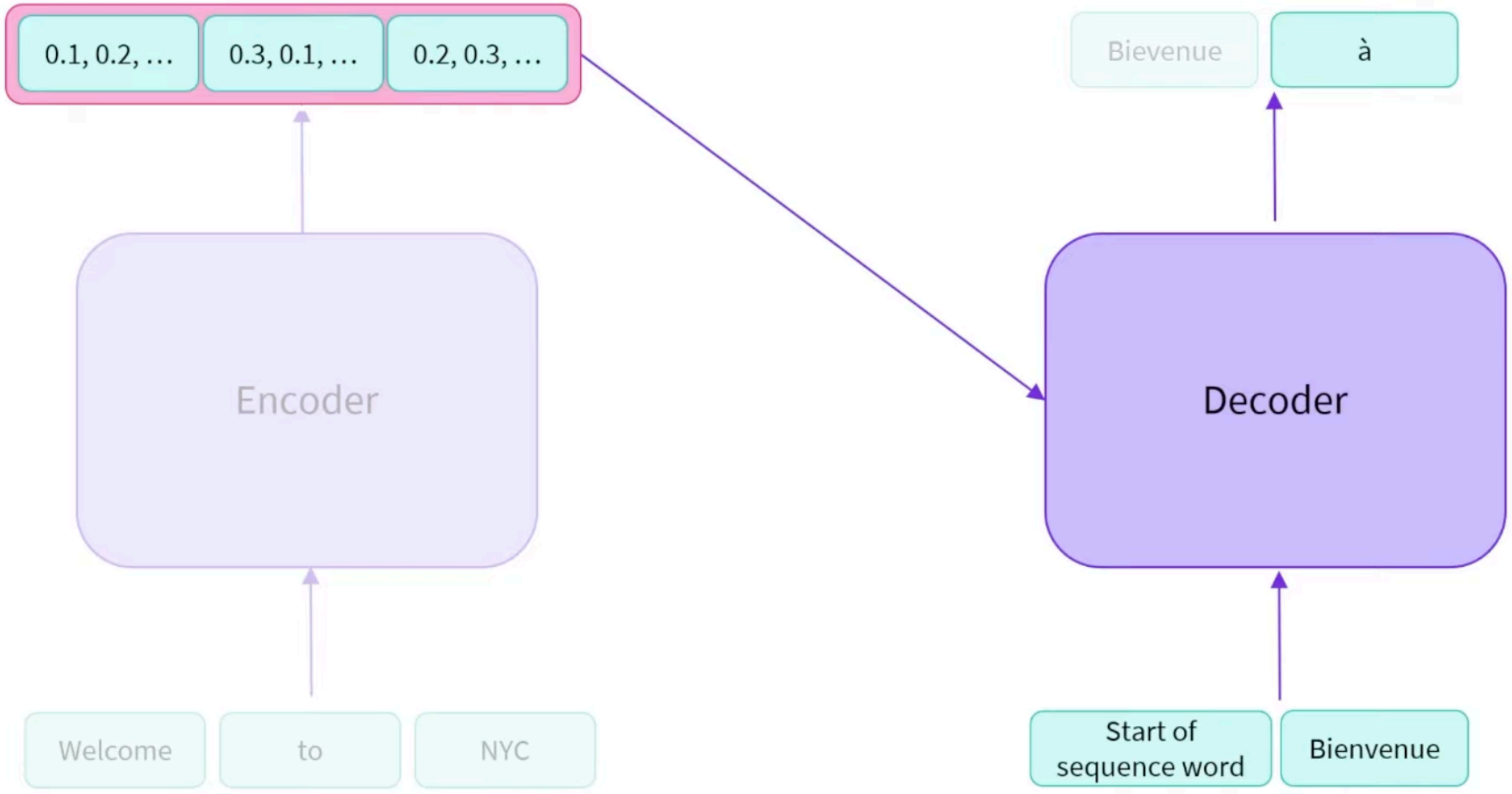
- Encode input sequence
- Attention over input token representations and <start>



slide from HuggingFace Transformers course: https://www.youtube.com/watch?v=0_4KEb08xrE&t=204s

Encoder-Decoder Inference

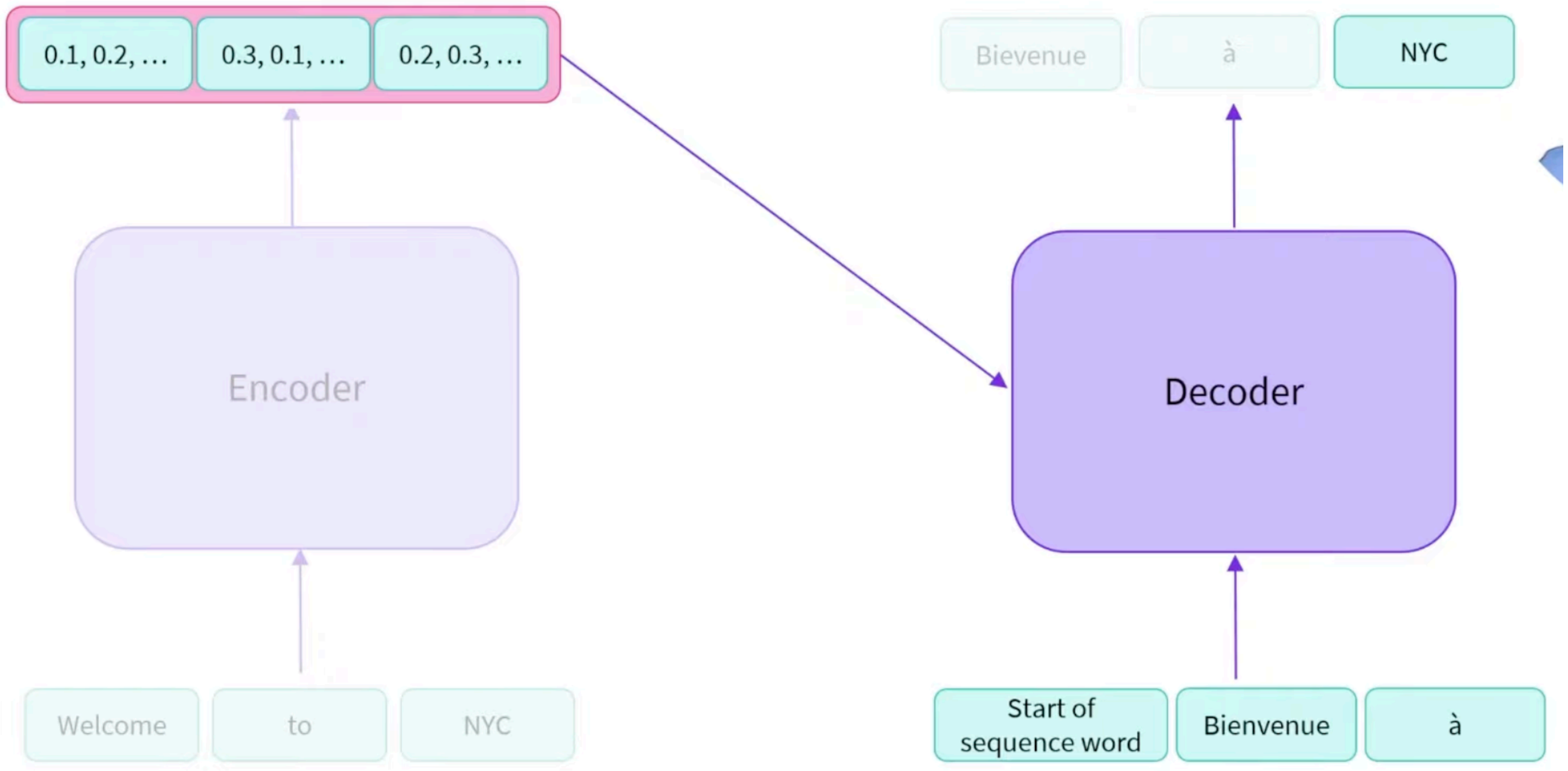
- Encode input sequence
- Attention over input token representations and <start>
- Self-attention



slide from HuggingFace Transformers course: https://www.youtube.com/watch?v=0_4KEb08xrE&t=204s

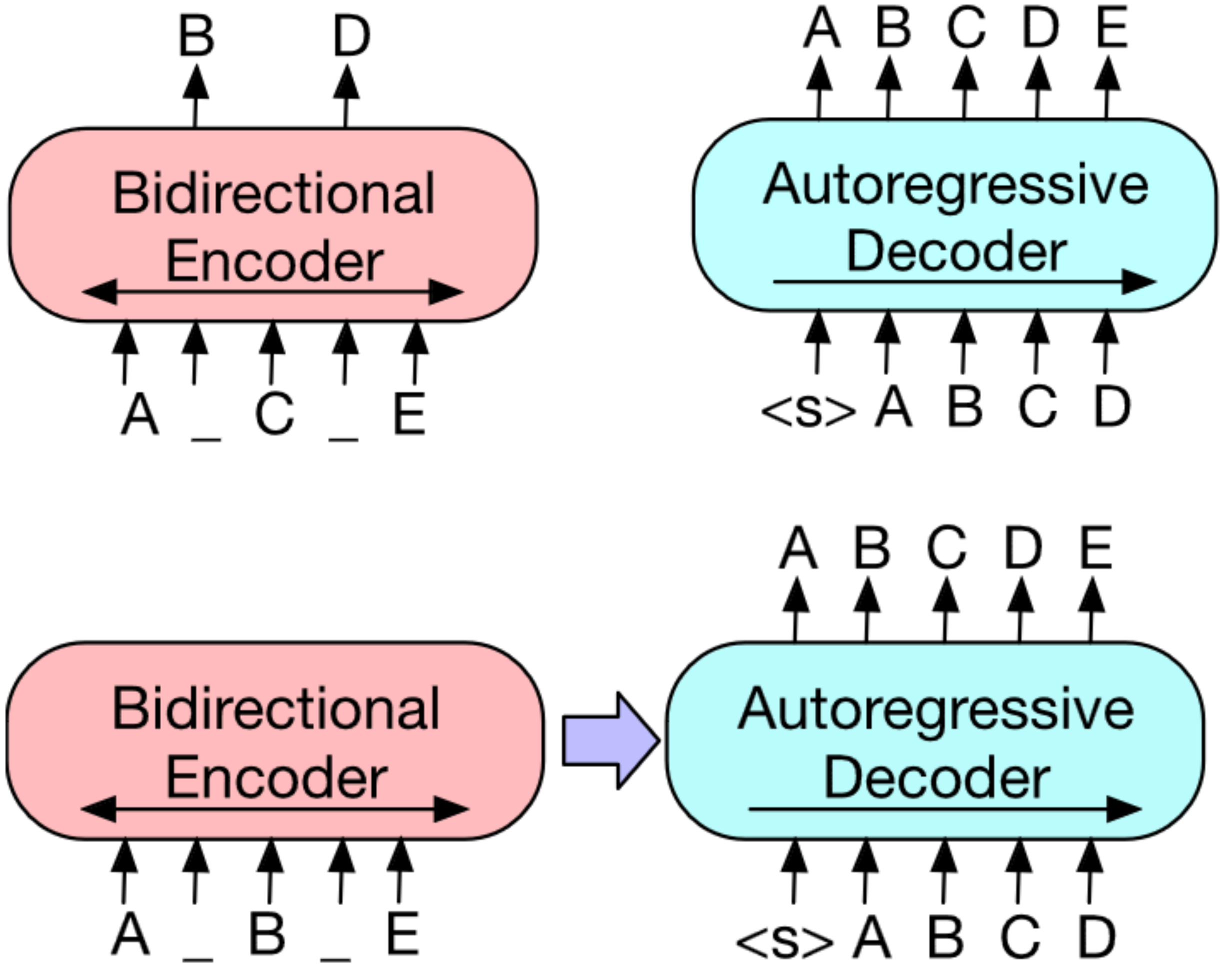
Encoder-Decoder Inference

- Encode input sequence
- Attention over input token representations and <start>
- Self-attention



slide from HuggingFace Transformers course: https://www.youtube.com/watch?v=0_4KEb08xrE&t=204s

Encoder, Decoder, Encoder-Decoder



BART, Lewis et al. 2019

Problems with the Transformer?

- Fixed context lengths "solved" with position embeddings
- Self-attention has quadratic cost $O(n^2 d)$

- Plug: Annotated Transformer (Sasha Rush): http://nlp.seas.harvard.edu/annotated-transformer/

Step 4: Optimization

Recap: Language Modeling Objective

- Assume we have training dataset including documents comprising sequences of bytes

$$\mathcal{D} = \left\{ \overline{d}^{(i)} \right\}_{i=1}^{N} \qquad \overline{d} = \langle b_0, \ldots, b_M \rangle$$

- Our objective is to find the LM parameters that maximize the probability of this dataset

$$\theta^* = \arg\max_{\theta} \Pi_{\overline{d} \in \mathcal{D}} p\left(\overline{d}; \theta\right)$$

- We assume documents are *tokenized* into sequences that the LM models autoregressively:

$$\overline{d} = \langle x_0, \ldots, x_{M'} \rangle \qquad p(\overline{d}; \theta) = \Pi_{j=1}^{M'} p(x_j \mid \langle x_0, \ldots, x_{j-1}; \theta)$$

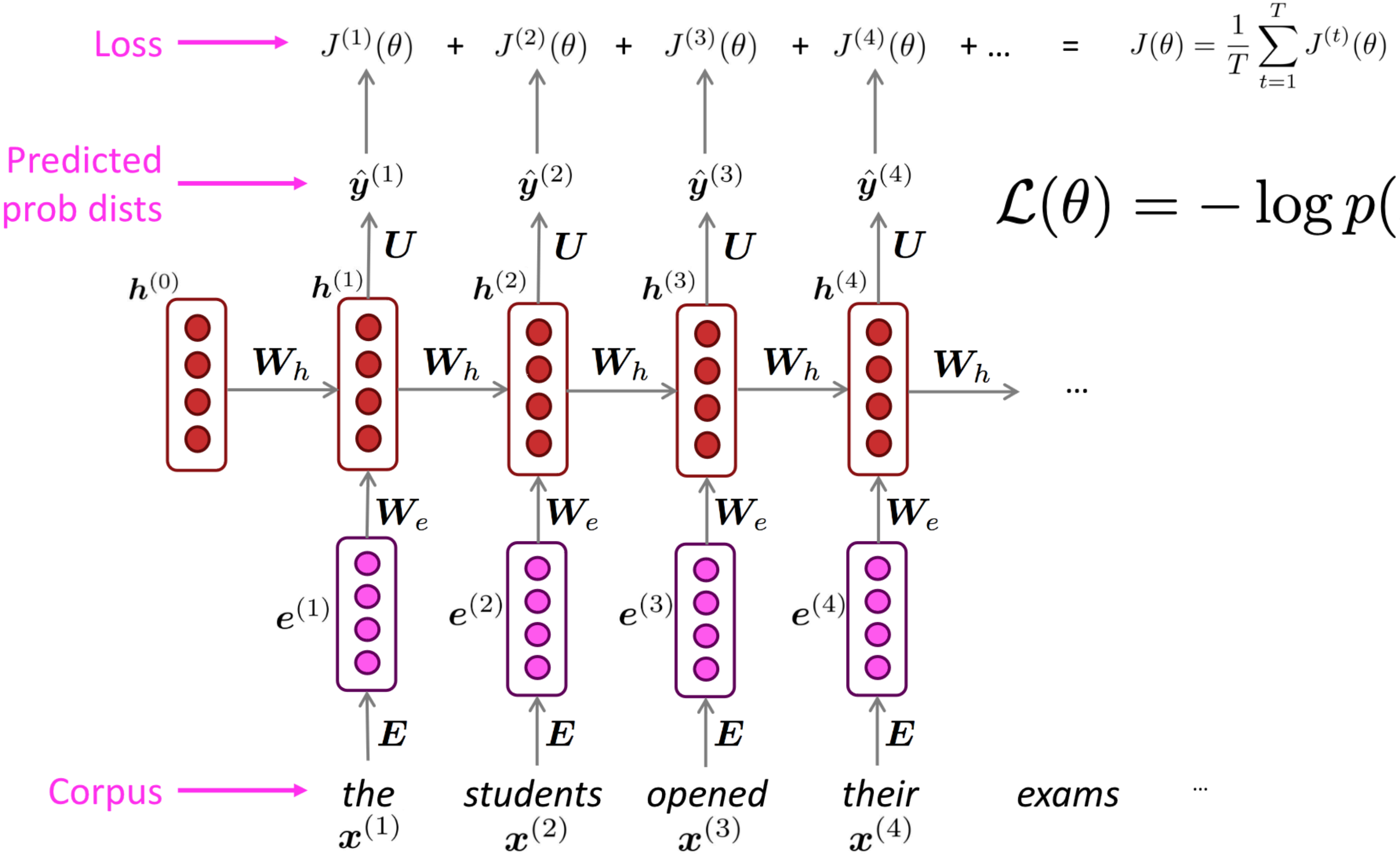
Recap: Language Modeling Objective

- Loss for step *i* is cross-entropy between true distribution p^* (i.e., one-hot) and predicted distribution:

$$\mathcal{L}(\theta) = -\sum_{x \in \mathcal{V}} p^*(x_i = x \mid \langle x_0 \ldots x_{i-1} \rangle) \log p(x_i = x \mid \langle x_0 \ldots x_{i-1} \rangle; \theta)$$

$$\mathcal{L}(\theta) = -\log p(x_i \mid \langle x_0 \ldots x_{i-1} \rangle; \theta)$$

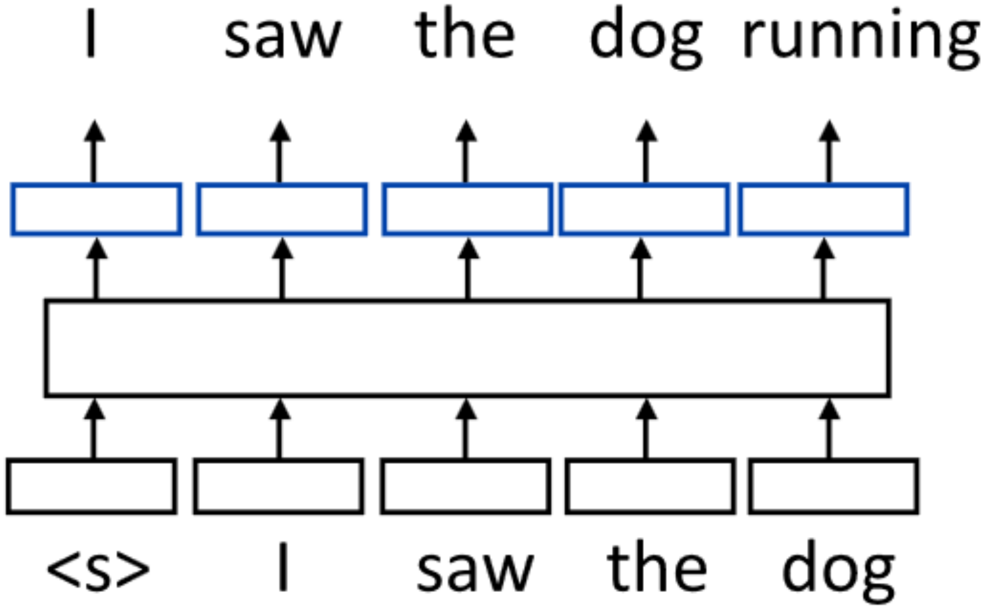
Next token prediction



$$\mathcal{L}(\theta) = -\log p(x_i \mid \langle x_0 \ldots x_{i-1} \rangle; \theta)$$

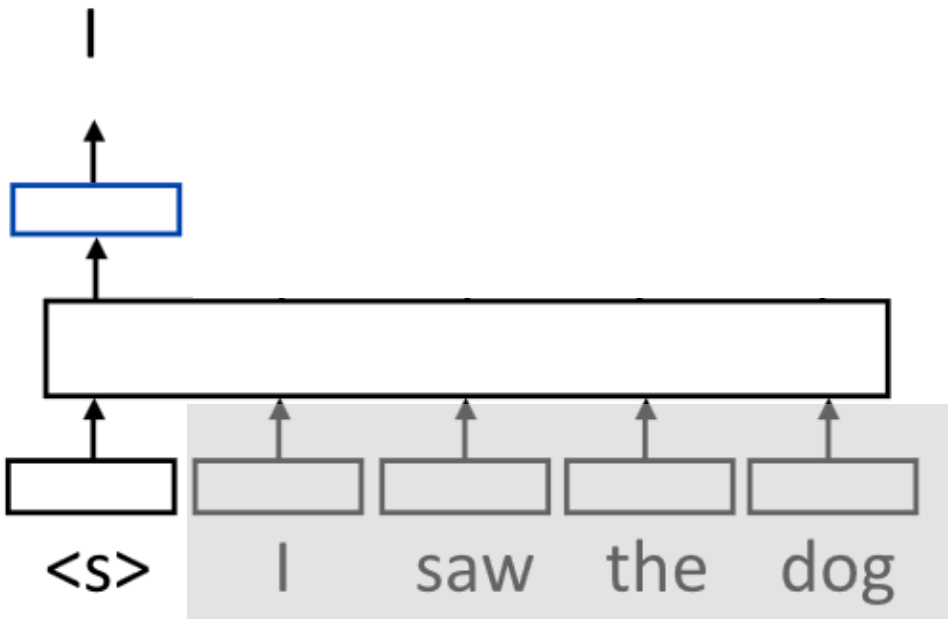
Slide from Stanford CS224

Next token prediction in Transformers



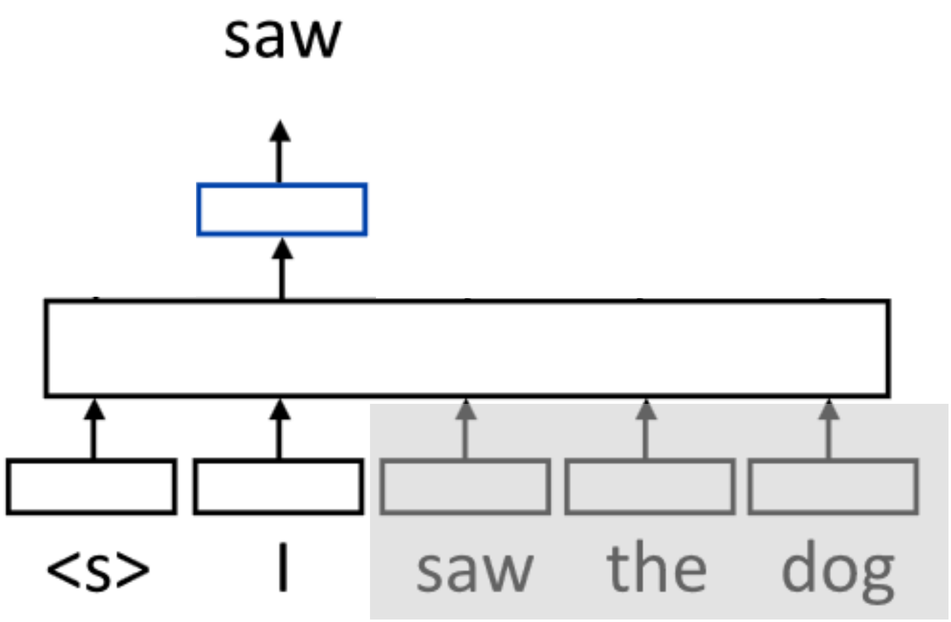
Slide from Greg Durrett, UT Austin CS 388

Next token prediction in Transformers



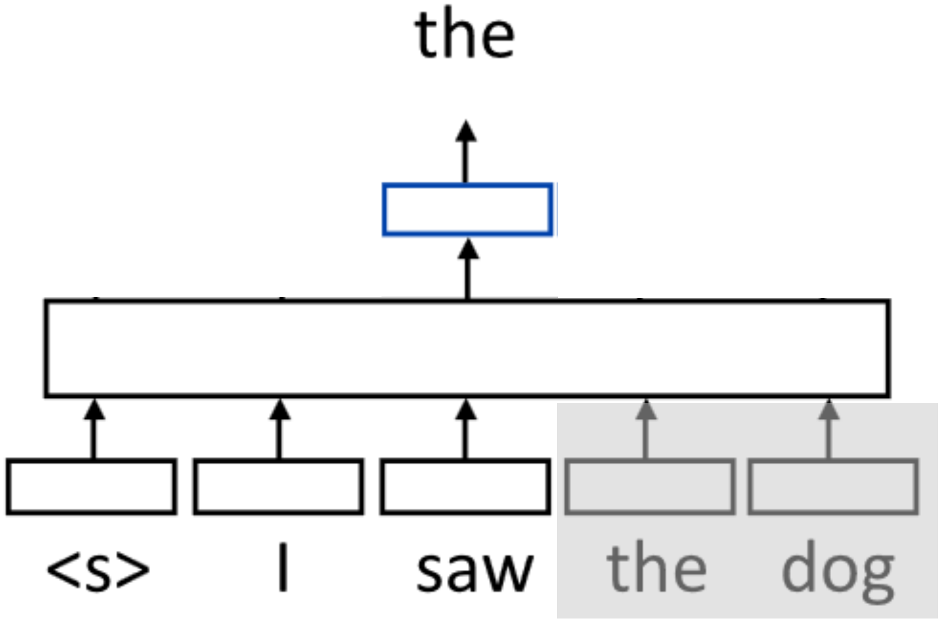
Slide from Greg Durrett, UT Austin CS 388

Next token prediction in Transformers



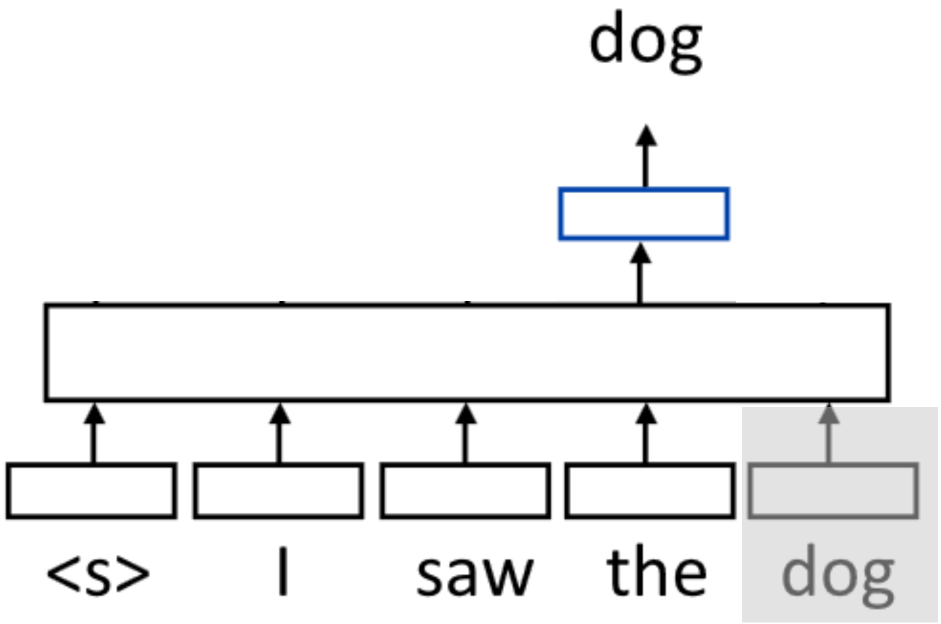
Slide from Greg Durrett, UT Austin CS 388

Next token prediction in Transformers



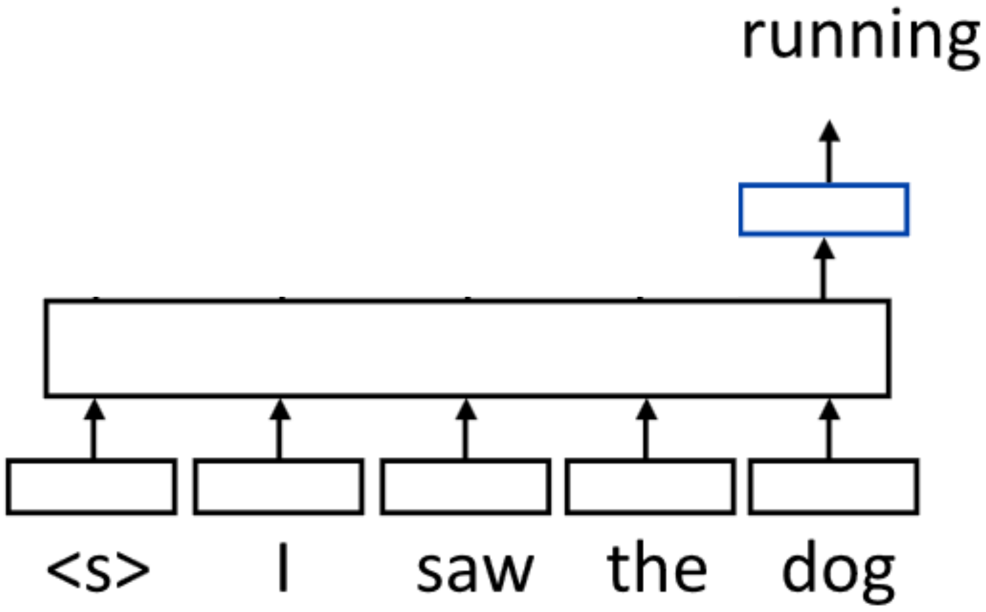
Slide from Greg Durrett, UT Austin CS 388

Next token prediction in Transformers



Slide from Greg Durrett, UT Austin CS 388

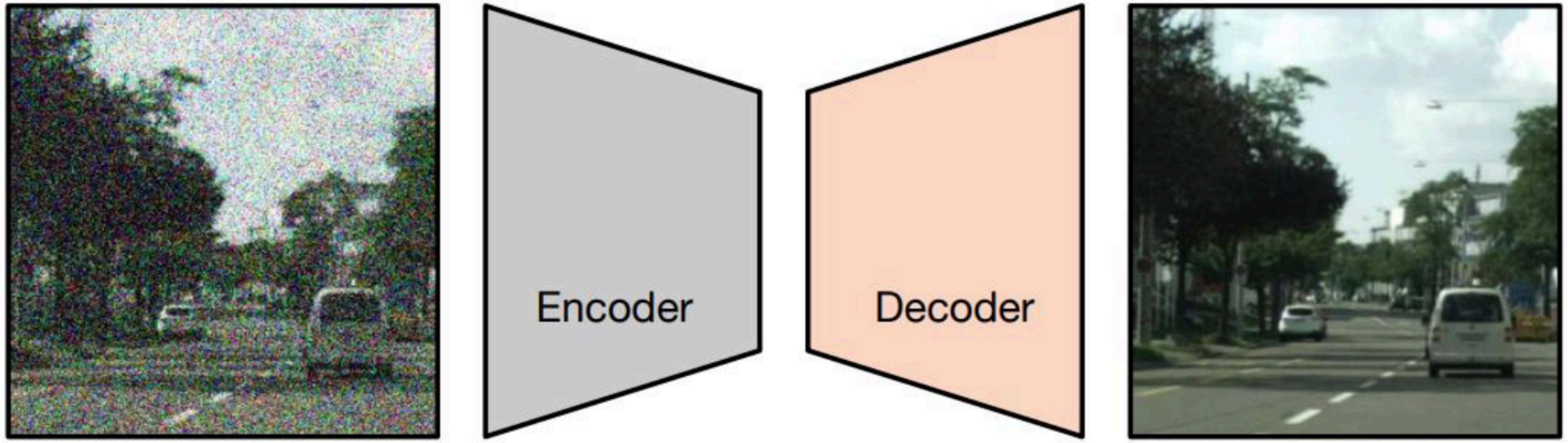
Next token prediction in Transformers



Slide from Greg Durrett, UT Austin CS 388

Denoising Objectives

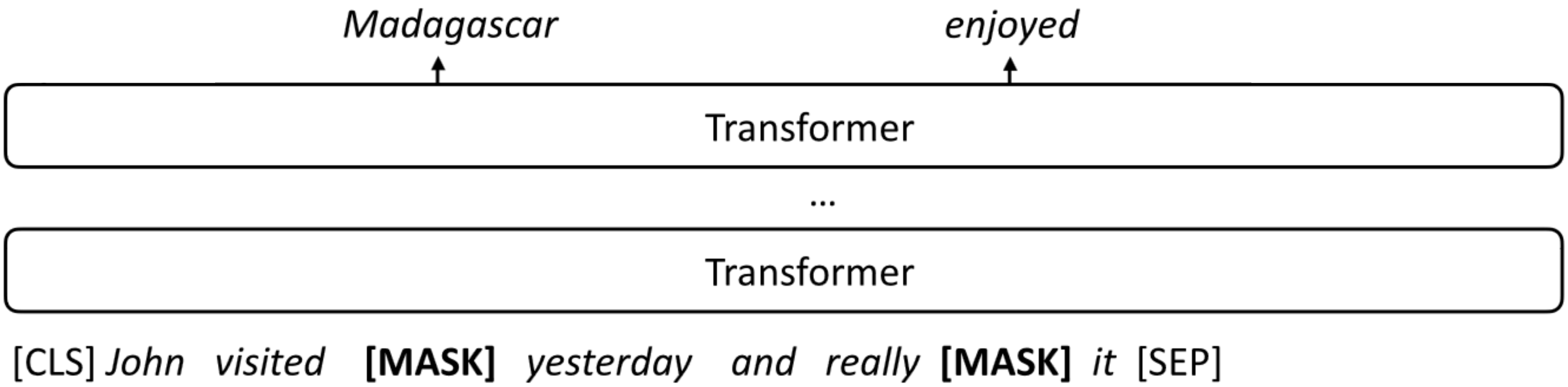
- Our goal: learn a distribution over text sequences
- Our assumption so far: this distribution is only backwards-looking (conditioned on prefix of the sequence)
- What if we remove this assumption?



Brempong et al. 2022, CVPR

Masking / Infilling Objectives

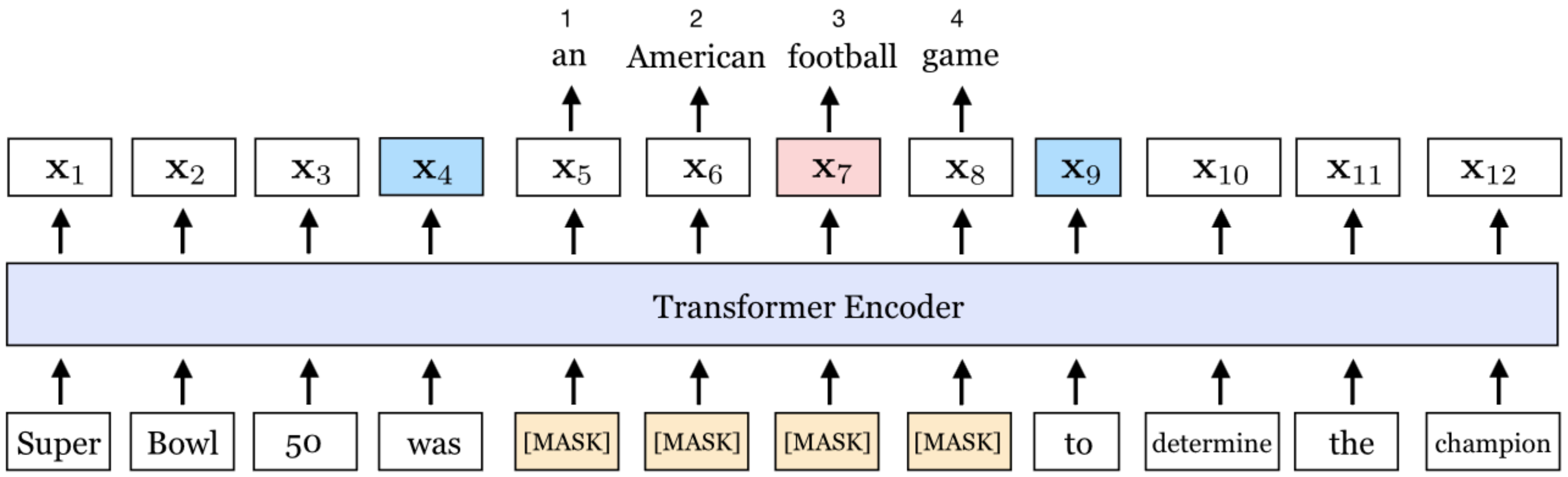
- Randomly mask out ~15% of tokens in the input, and try to predict them from past *and future* context



BERT, Devlin et al. 2019 (slide from UT Austin CS 388)

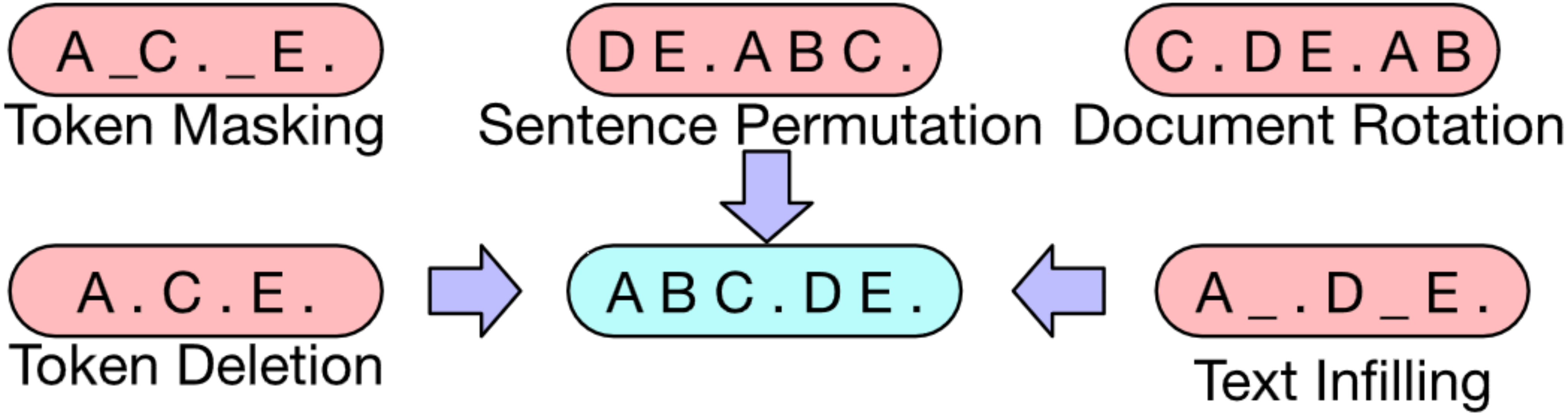
Masking / Infilling Objectives

- Randomly mask out ~15% of tokens in the input, and try to predict them from past *and future* context
- Or mask out spans of text



SpanBERT, Joshi et al. 2020 (TACL)

Auxiliary Objectives



BART, Lewis et al. 2019

Step 5: Inference

Recap: What is a language model?

- Language models assign a probability to a sequence of words $p(\overline{y})$
- We can decompose this probability using the chain rule $p(\overline{y}) = \Pi_{i=1}^{T} p(y_i | y_{0:i-1})$
- We can autoregressively generate sequences from the language model by sampling from its token-level probability $p(y_i | y_{0:i-1})$
- We can condition on our language distribution on something else $p(y_i | y_{0:i-1}; \overline{x})$

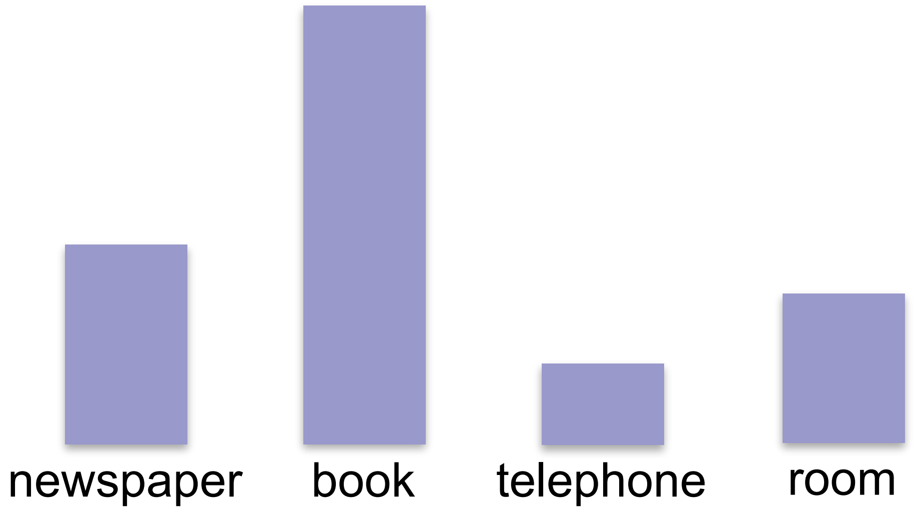
What can we do with language models?

- Computing probabilities of a sequence
- Autoregressive sequence generation

Decoding strategies

- Argmax (greedy decoding)

$$y_T = \arg\max_{y \in \mathcal{V}} p(y \mid y_{0:t-1})$$

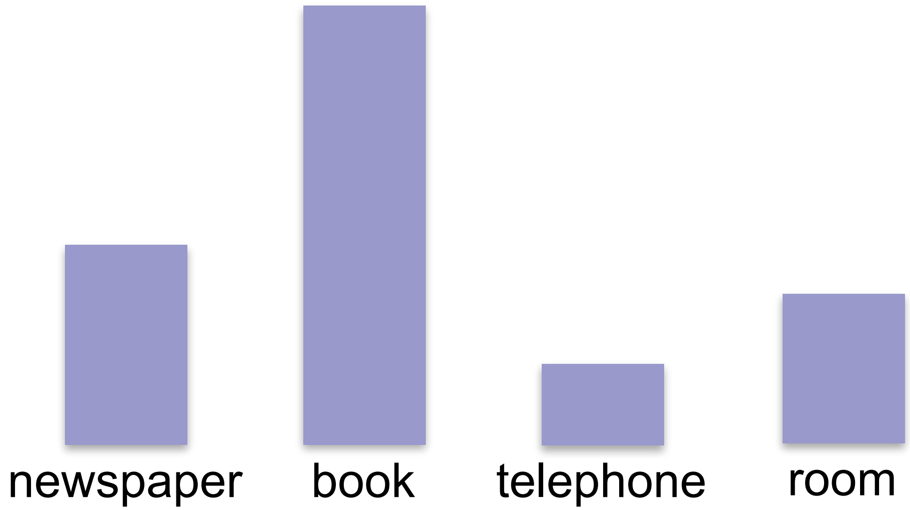


Decoding strategies

- Argmax (greedy decoding)
- Sampling from language model directly

$$y_T = \arg\max_{y \in \mathcal{V}} p(y \mid y_{0:t-1})$$

$$y_T \sim p(\cdot \mid y_{0:t-1})$$



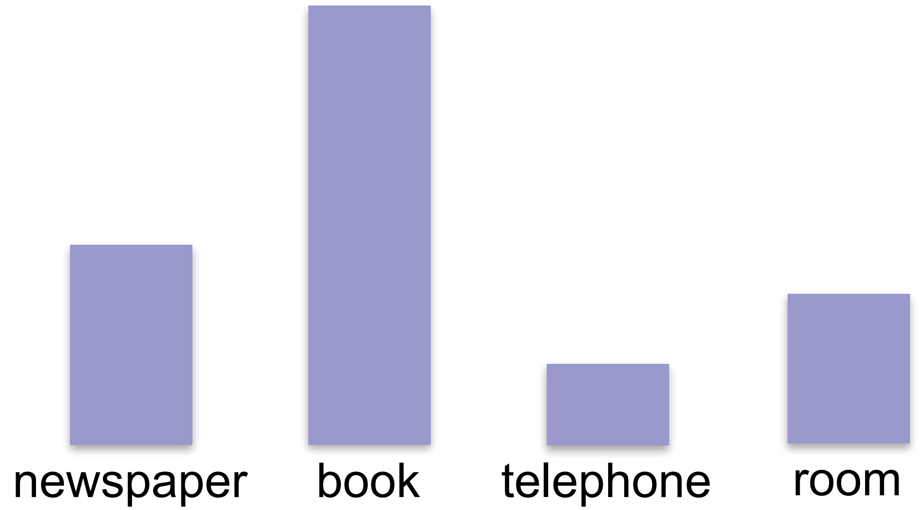
Decoding strategies

- Argmax (greedy decoding)
- Sampling from language model directly
- Adjusting temperature of distribution

$$y_T = \arg\max_{y \in \mathcal{V}} p(y \mid y_{0:t-1})$$

$$y_T \sim p(\cdot \mid y_{0:t-1})$$

$$p'(y_T = y) = \frac{\exp(z_y/T)}{\sum_{y' \in \mathcal{V}}(z_{y'}/T)}$$



Decoding strategies

- Argmax (greedy decoding)

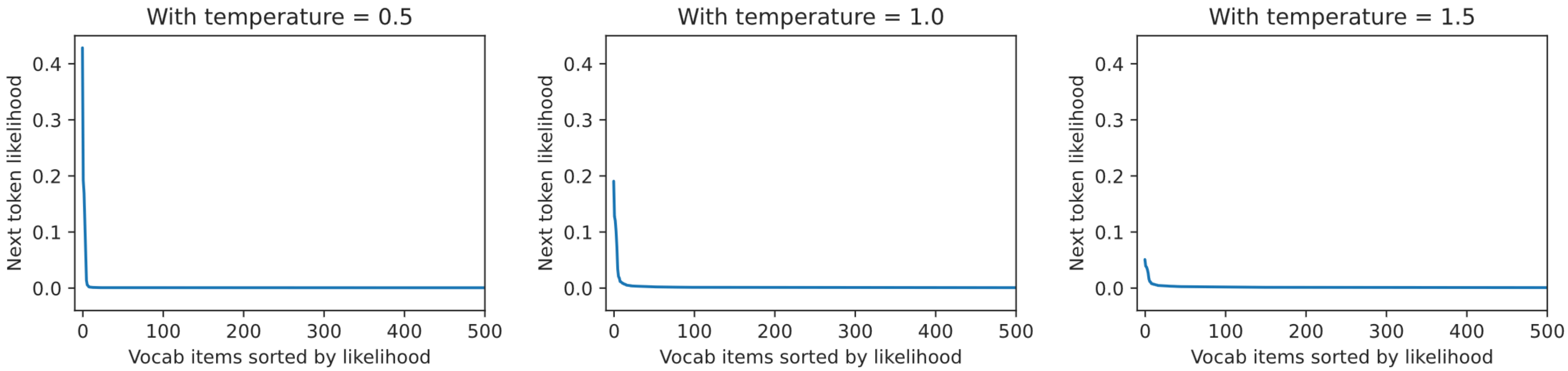
$$y_T = \arg\max_{y \in \mathcal{V}} p(y \mid y_{0:t-1})$$

- Sampling from language model directly

$$y_T \sim p(\cdot \mid y_{0:t-1})$$

- Adjusting temperature of distribution

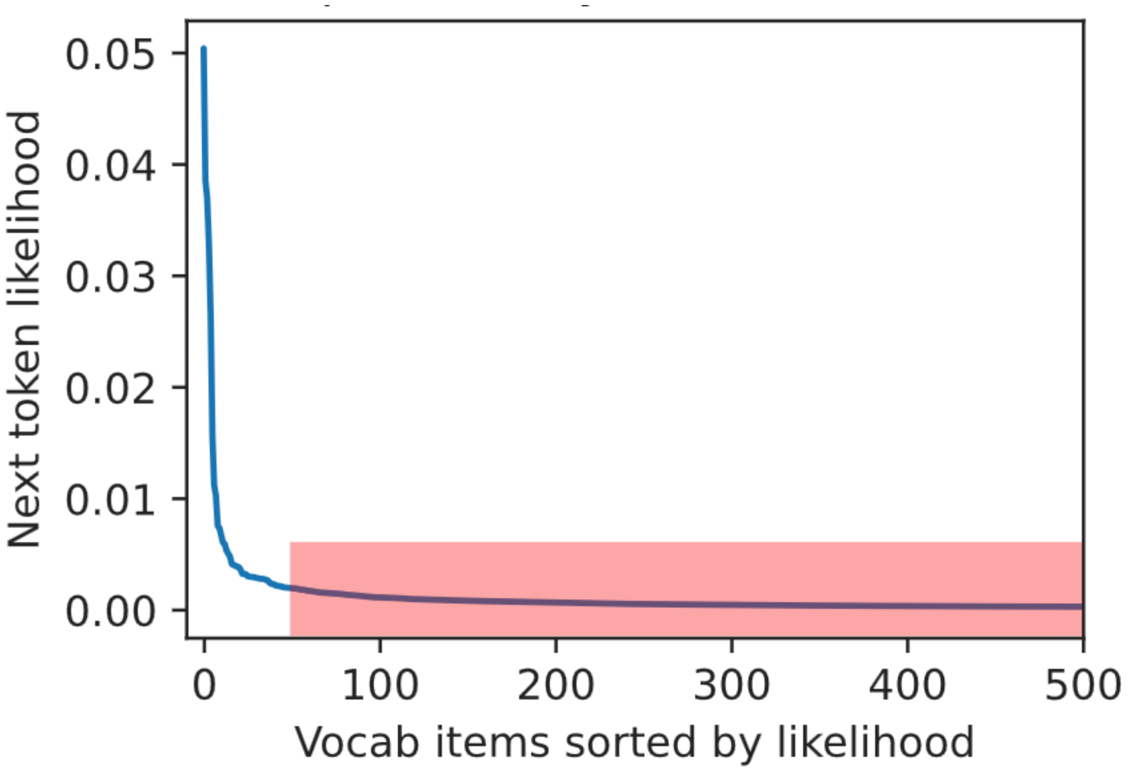
$$p'(y_T = y) = \frac{\exp(z_y/T)}{\sum_{y' \in \mathcal{V}}(z_{y'}/T)}$$



Slide from Daphne Ippolito / Chenyan Xiong, CMU LLMs course http://cmu-llms.org/

Decoding strategies

- Top-k sampling: reassign probability mass from all but the top *k* tokens to the top *k* tokens



Slide from Daphne Ippolito / Chenyan Xiong, CMU LLMs course http://cmu-llms.org/

Decoding strategies

- Nucleus sampling: reassign probability mass to the most probable tokens whose cumulative probability is at least *p*

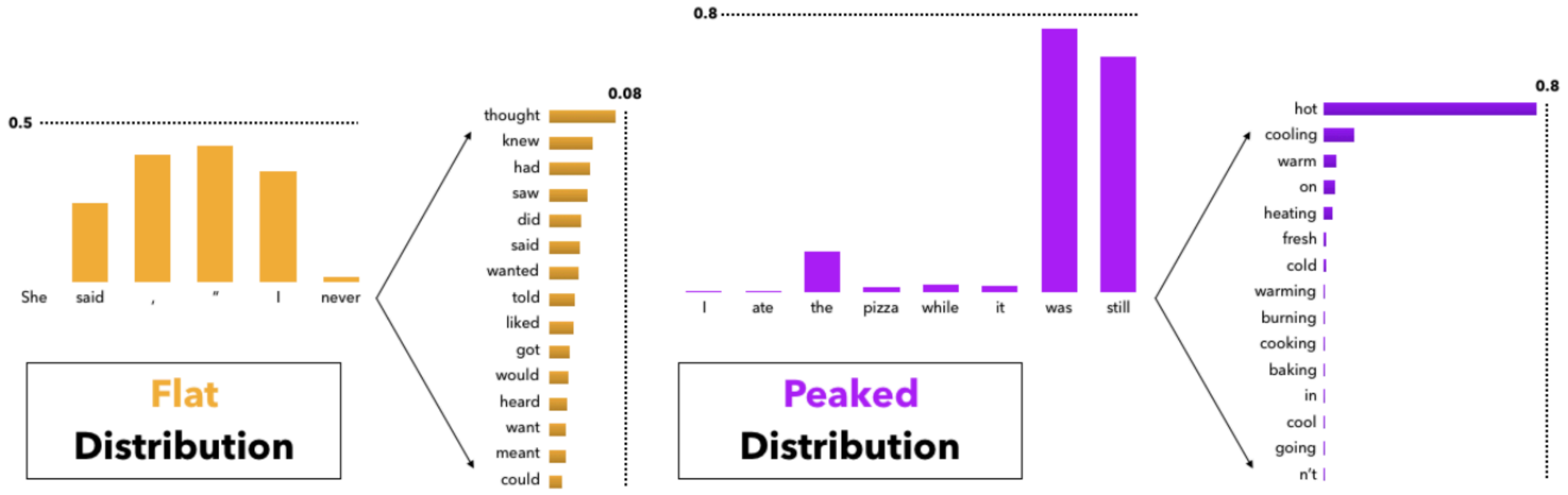


Figure 5: The probability mass assigned to partial human sentences. Flat distributions lead to many moderately probable tokens, while peaked distributions concentrate most probability mass into just a few tokens. The presence of flat distributions makes the use of a small k in top-k sampling problematic, while the presence of peaked distributions makes large k's problematic.

Holtzman et al. 2020, ICLR

Beam search

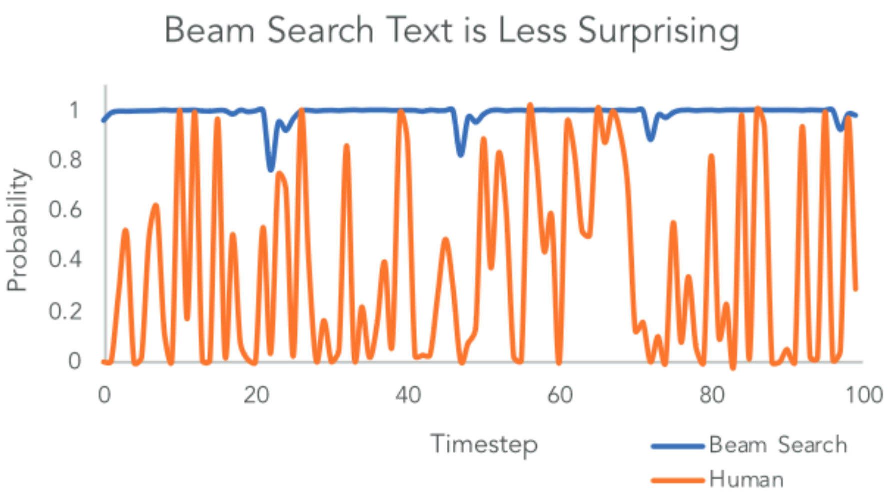
- It's intractable to find the *most probable sequence* according to a language model
- Greedy search doesn't yield the most probably sequence
- Instead: beam search
  - Approximate the search by keeping around candidate continuations
  - At the end, choose the highest probability sequence in the beam

$$\overline{y}^* = \arg\max_{\overline{y} \in \mathcal{V}^*} p(\overline{y})$$

$$y_t = \arg\max_{y \in \mathcal{V}} p(y \mid y_{0:t-1})$$

Beam search

- But do we even want to find the highest-probability sequence according to a LM?
- Human language is noisy and surprising
- Optimizing for LM probability leads to repetitive and uninteresting text



Beam Search	**Human**
...to provide an overview of the current state-of-the-art in the field of computer vision and machine learning, and to provide an overview of the current state-of-the-art in the field of computer vision and machine learning, and to provide an overview of the current state-of-the-art in the field of computer vision and machine learning, and to provide an overview of the current state-of-the-art in the field of computer vision and machine learning, and...	...which grant increased life span and three years warranty. The Antec HCG series consists of five models with capacities spanning from 400W to 900W. Here we should note that we have already tested the HCG-620 in a previous review and were quite satisfied With its performance. In today's review we will rigorously test the Antec HCG-520, which as its model number implies, has 520W capacity and contrary to Antec's strong beliefs in multi-rail PSUs is equipped...

Holtzman et al. 2020, ICLR

Beam search

- But do we even want to find the highest-probability sequence according to a LM?
- Human language is noisy and surprising
- Optimizing for LM probability leads to repetitive and uninteresting text

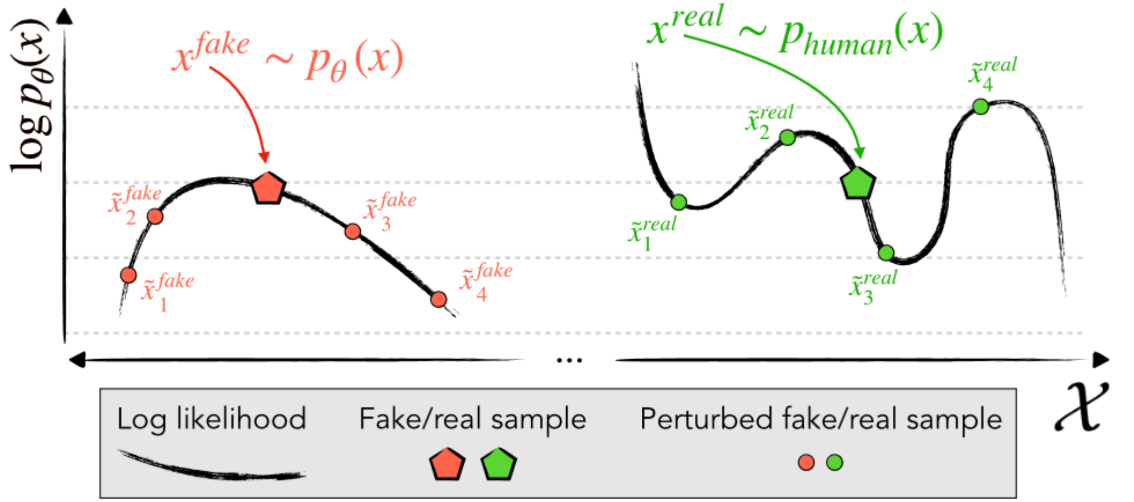


Figure 2. We identify and exploit the tendency of machine-generated passages $x \sim p_\theta(\cdot)$ **(left)** to lie in negative curvature regions of $\log p(x)$, where nearby samples have lower model log probability on average. In contrast, human-written text $x \sim p_{real}(\cdot)$ **(right)** tends not to occupy regions with clear negative log probability curvature; nearby samples may have higher or lower log probability.

DetectGPT, Mitchell et al. 2023, ICML