Lecture 05: Sequence-to-sequence Models & Transformers

Xiang Ren

USC CSCI 662 Advanced NLP

Spring 2026

Some slides adapted from Chris Manning and Dan Jurafsky

Announcements

Announcements + Logistics

- Project proposal feedback & grades WIP, expected early next week.
- Paper presentation schedule is out on website!
  - https://docs.google.com/spreadsheets/d/10OVp0QWnyy9chi12cTJUpzZQc6Ah34cWnFKAo0jxo-8/edit?gid=0#gid=0

Paper Presentation Schedule (starting Feb 18)

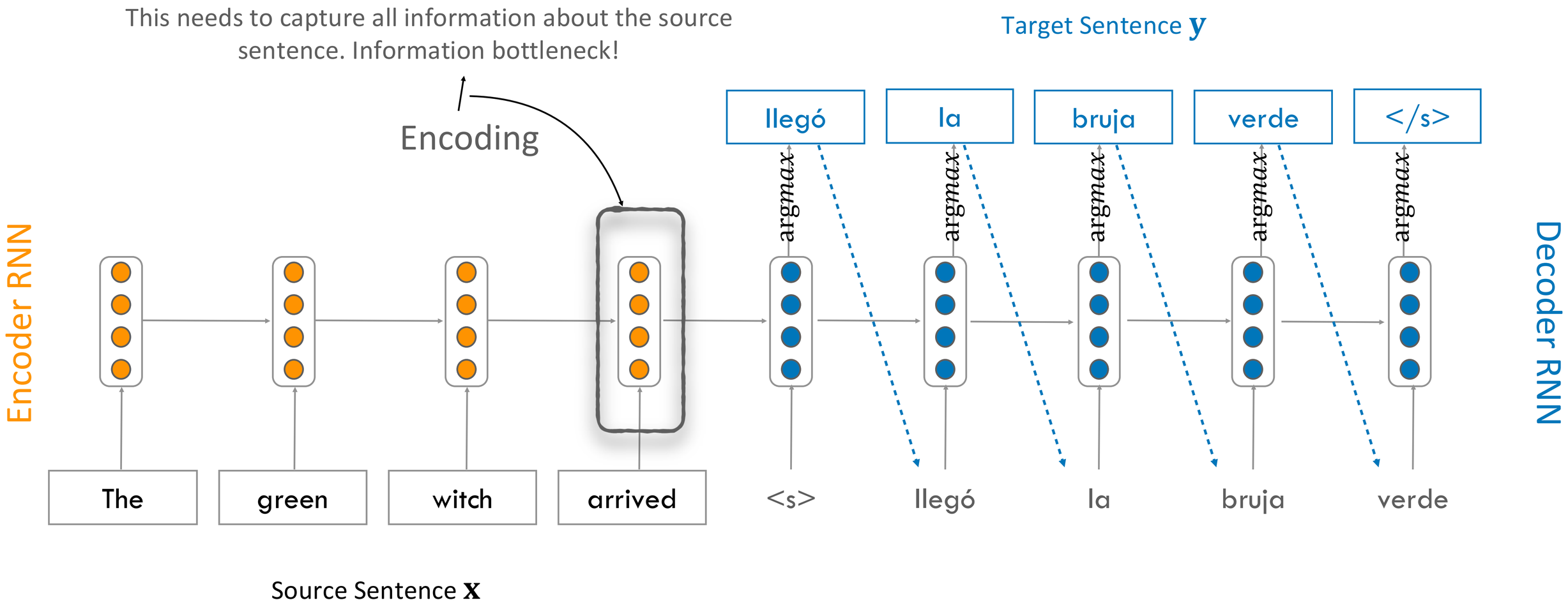
Name of Paper	Name of Pr	Discussion	Discussion	Day	Date
Collaborative gym: A framework for enabling and evaluating human-agent collaboration	Sheryl Mathev	Hong-En Chen	Anthony Liang	1	2/18/26
Sys2Bench: Inference-Time Computations for LLM Reasoning and Planning (arXiv:2502.12	I-Chun Liu	Zeyu Shanggua	Xinlei Yu	1	2/18/26
VisuLogic: A Benchmark for Evaluating Visual Reasoning in Multi-modal Large Language M	Zongjian Li	Ziyan Yang	Muzi Tao	2	2/25/26
Search-o1: Agentic Search-Enhanced Large Reasoning Models	Dimitrios Andr	Qihan Zhang	Efthymios Tsap	2	2/25/26
Humanity's Last Exam	Toan Nguyen	Didem Zeynep	Yifan Wu	2	2/25/26
rStar-Math: Small LLMs Can Master Math Reasoning with Self-Evolved Deep Thinking	Mohammad H	Zihan Wang	Didem Zeynep	3	3/4/26
MemoryAgentBench: Evaluating Memory in LLM Agents	Ziyan Yang	Zongjian Li	Jike Zhong	3	3/4/26
CoT-Self-Instruct: Building high-quality synthetic prompts for reasoning and non-reasoning t	Gengpei Qi	Yipeng Gao	Mohammad Ho	3	3/4/26
Constitutional Classifiers: Defending against Universal Jailbreaks across Thousands of Hou	Michael Duan	Sheryl Mathew	Anu Soneye	3	3/4/26
Can MLLMs Reason in Multimodality? EMMA: An Enhanced MultiModal ReAsoning Benchn	Efthymios Tsa	Muzi Tao	Chen Chu	3	3/4/26
Stop Overthinking: A Survey on Efficient Reasoning for Large Language Models	Stop Overthinl	Ania Serbina	Zhaoyuan Den	3	3/4/26
ReTool: Reinforcement Learning for Strategic Tool Use in LLMs	Qihan Zhang	I-Chun Liu	Sunwoo Lim	4	3/25/26
ToolRL: Reward is All Tool Learning Needs	Xinlei Yu	Haolin Xiong	Zhangyu Jin	4	3/25/26
TTRL: Test-Time Reinforcement Learning	Alexios Rusto	Sunwoo Lim	Letao Chen	4	3/25/26
It's Not That Simple: An Analysis of Simple Test-Time Scaling (arXiv:2507.14419)	Jike Zhong	Toan Nguyen	Zhaoyuan Den	4	3/25/26
The Illusion of Thinking: Understanding the Strengths and Limitations of Reasoning Models	Yifan Wu	DJ Bell	Chen Chu	4	3/25/26
Red-Teaming LLM Multi-Agent Systems via Communication Attacks	Qixin Hu	Michael Duan	Jike Zhong	4	3/25/26
Training Language Models to Reason Efficiently	Haolin Xiong	Anu Soneye	Zihan Wang	5	4/1/26
Layer by Layer: Uncovering Hidden Representations in Language Models	Ryan Swift	Hong-En Chen	Yanchen Liu	5	4/1/26
Reasoning Models Don't Always Say What They Think	Letao Chen	Efthymios Tsap	DJ Bell	5	4/1/26
Training Large Language Model to Reason in a Continuous Latent Space	Zhaoyuan Der	Bo-Ruei Huang	Ryan Swift	5	4/1/26
Understanding R1-Zero-Like Training: A Critical Perspective	Sunwoo Lim	Dimitrios Andre	Alexios Ruston	5	4/1/26
LIMO: Less is More for Reasoning	Yanchen Liu	Sheryl Mathew	Ania Serbina	5	4/1/26
Reinforcement Learning Teachers of Test Time Scaling (RLT) (arXiv:2506.08388)	Bo-Ruei Huan	Anthony Liang	Zeyu Shanggua	6	4/8/26
ROBOTOUILLE: Asynchronous Planning Benchmark for LLM Agents (ICLR 2025)	Didem Zeyne	I-Chun Liu	Ryan Swift	6	4/8/26
PaperArena: Benchmarking Tool-Using Agents	Yipeng Gao	Yifan Wu	Ziyan Yang	6	4/8/26

Paper Presentation

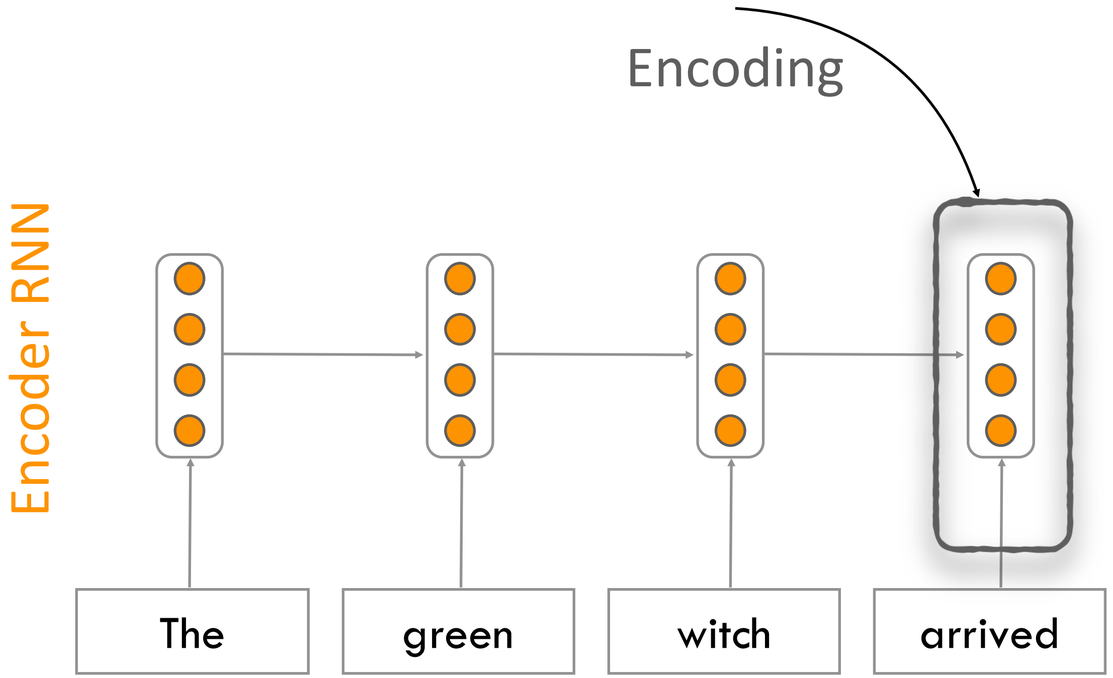
The slides of the presentation need to be shared a day before the presentation to the class.

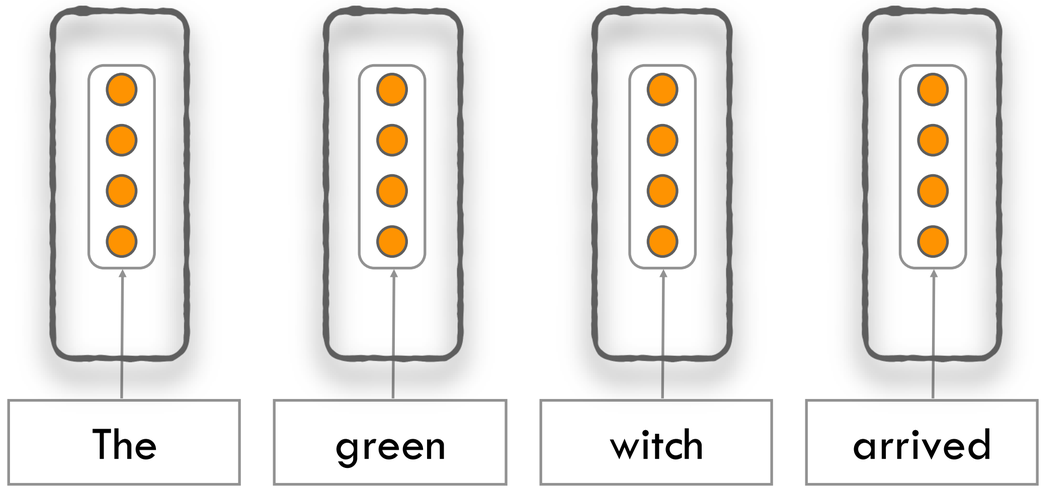
Please add to the slide deck TA sent the night before, and TA will distribute before the class in the morning.

Recap: Transformer Language Models & Pre-training


This needs to capture all information about the source sentence. Information bottleneck!
Encoding
Target Sentence y
Encoder RNN
Decoder RNN
The
green
witch
arrived
<s>
llegó
la
bruja
verde
</s>
argmax
Source Sentence x

Information Bottleneck: One Solution



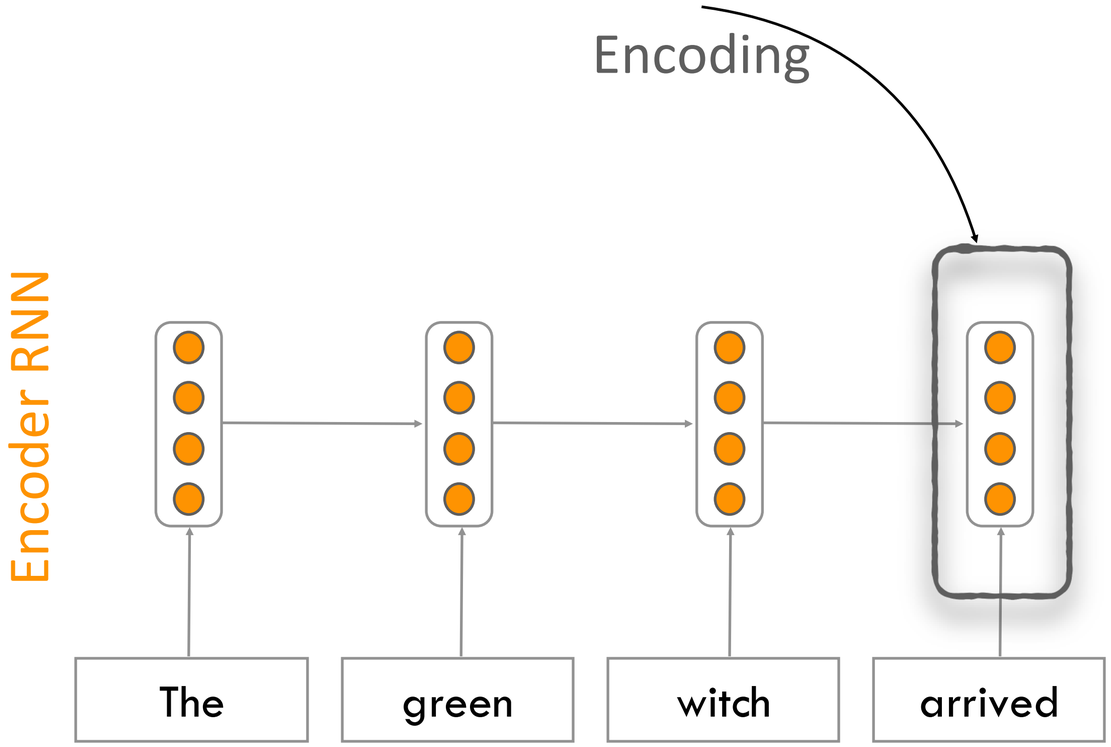


What if we had access to all hidden states?

How to create this?

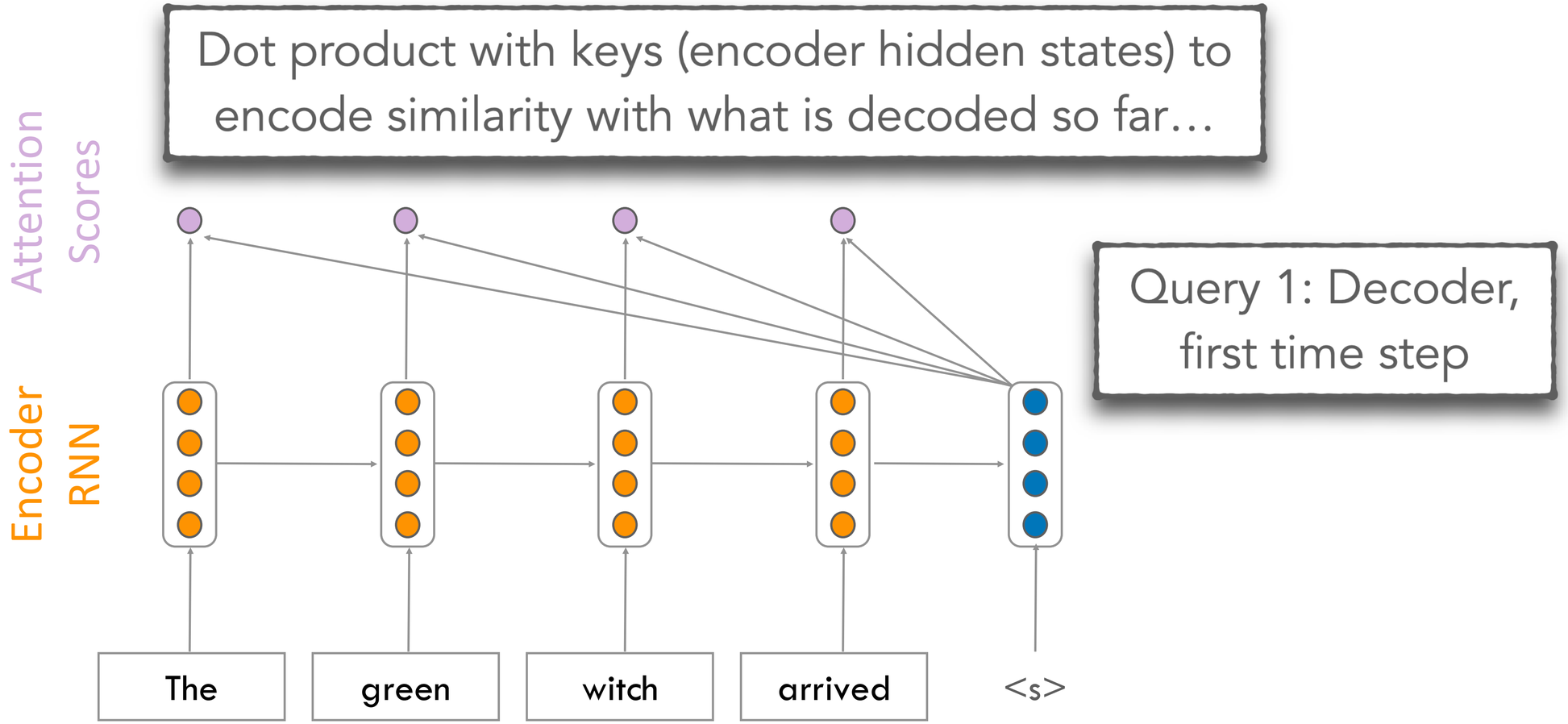
Attention Mechanism

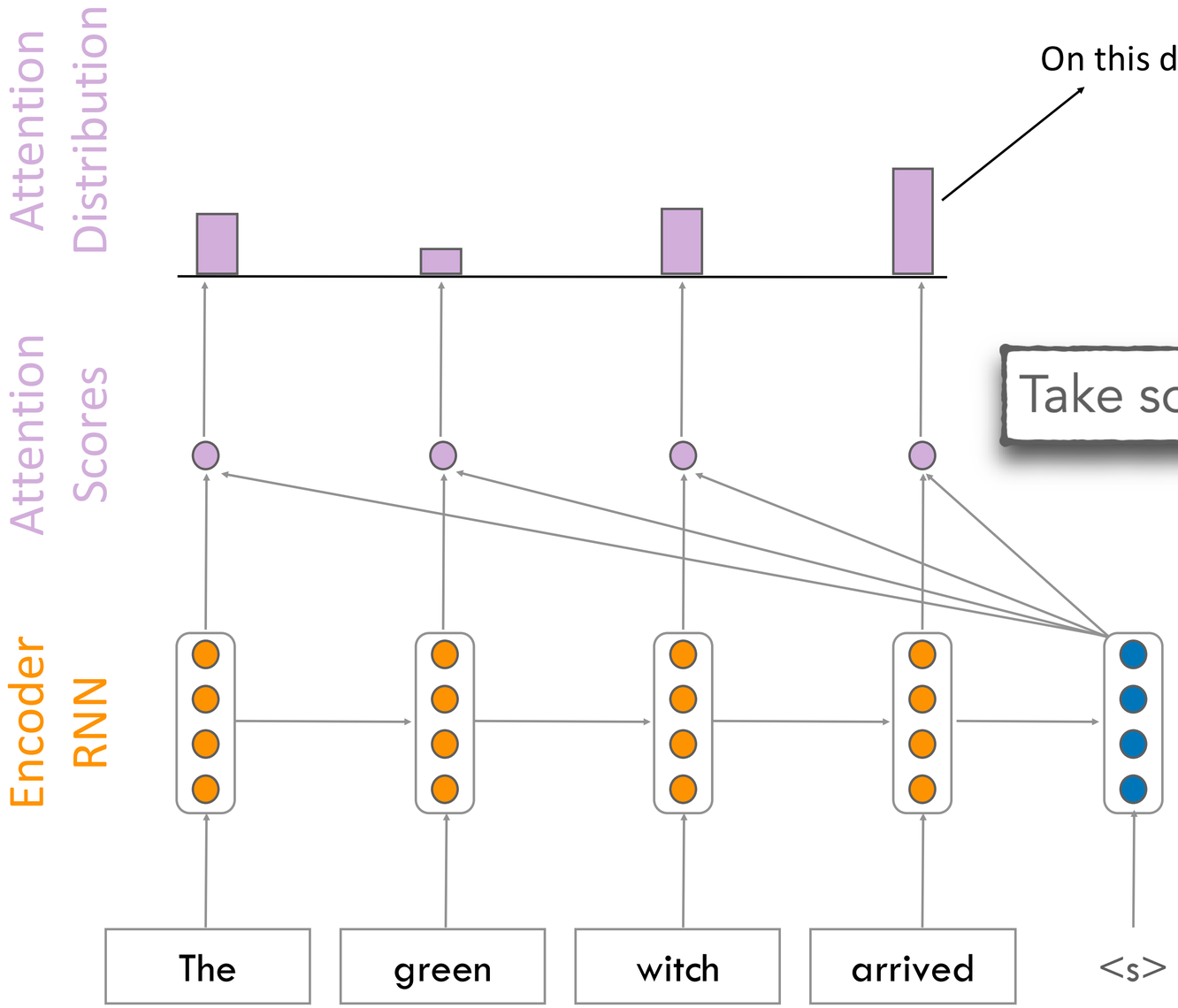
- Attention mechanisms allow the decoder to focus on a particular part of the source sequence at each time step
- Single fixed-length vector $\mathbf{c}_t$ by taking a weighted sum of all the encoder hidden states
  - One per time step of the decoder!
- In general, we have a single query vector and multiple key vectors.
- We want to score each query-key pair



Bahdanau et al., 2015

Seq2Seq with Attention

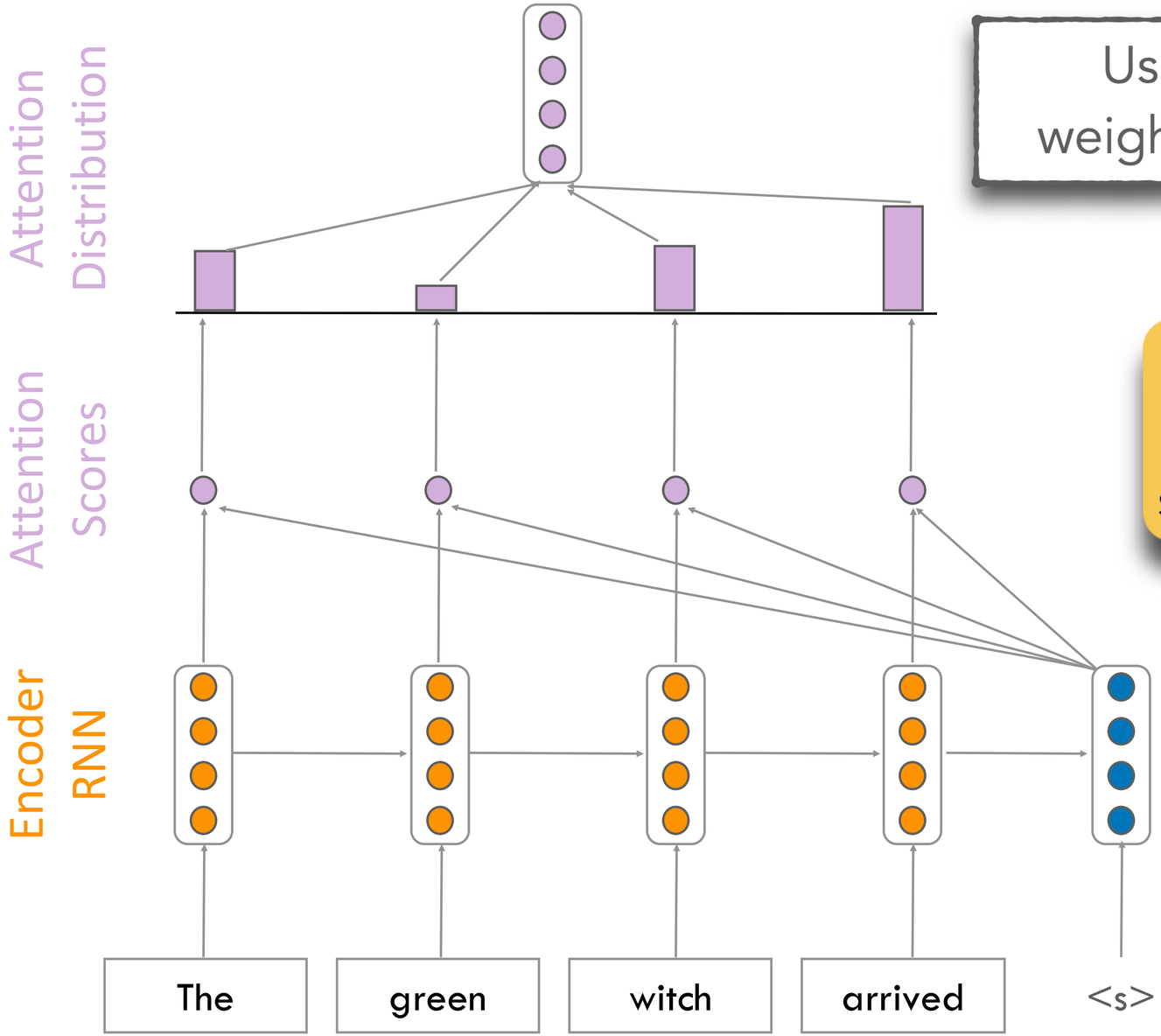




On this decoder tilmestep we are mostly focusing on the source token “arrived”

Take softmax to turn into probability distribution

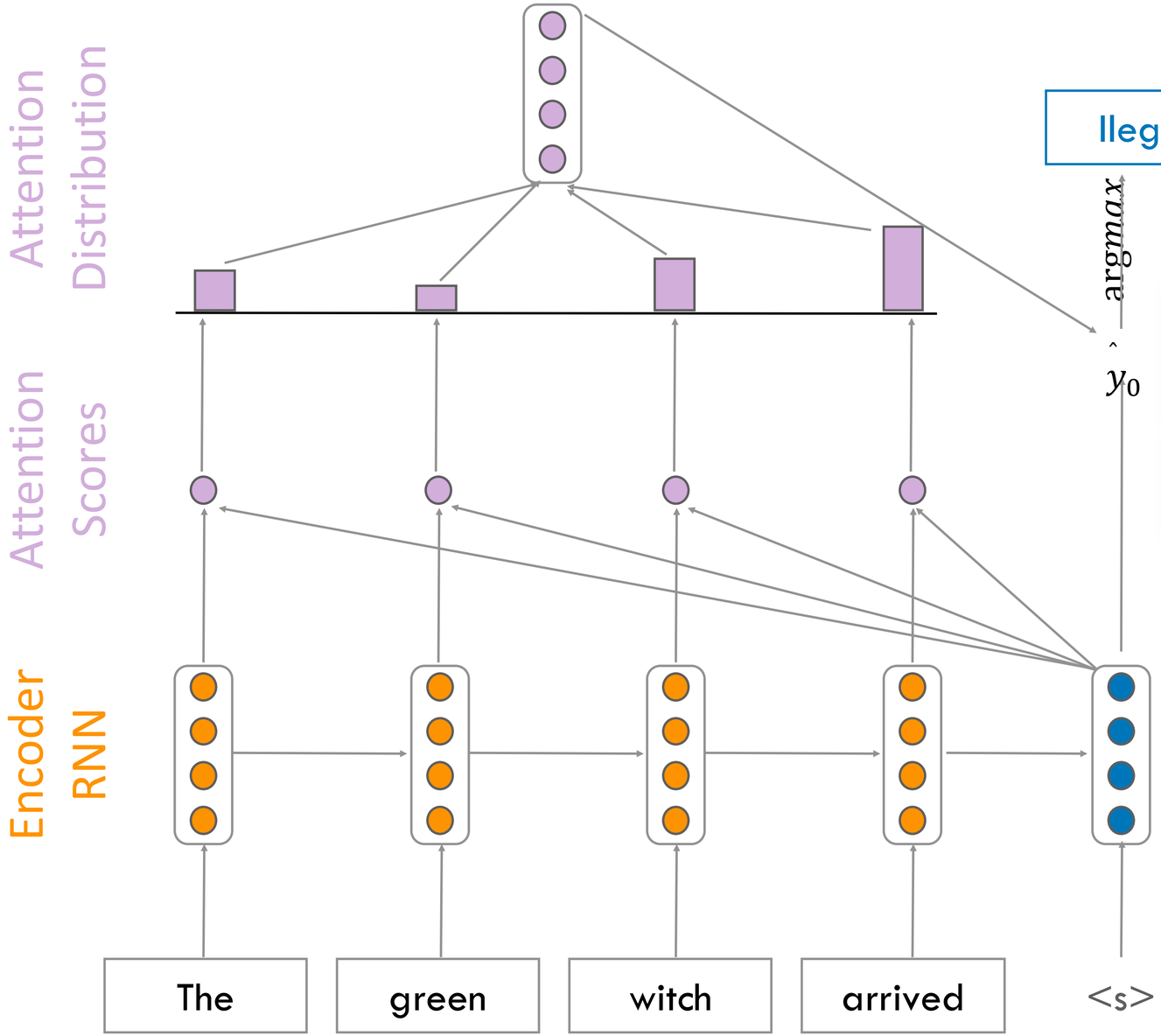
Source Sentence $\mathbf{x}$



Use the attention distribution to take a weighted sum of the encoder hidden states.

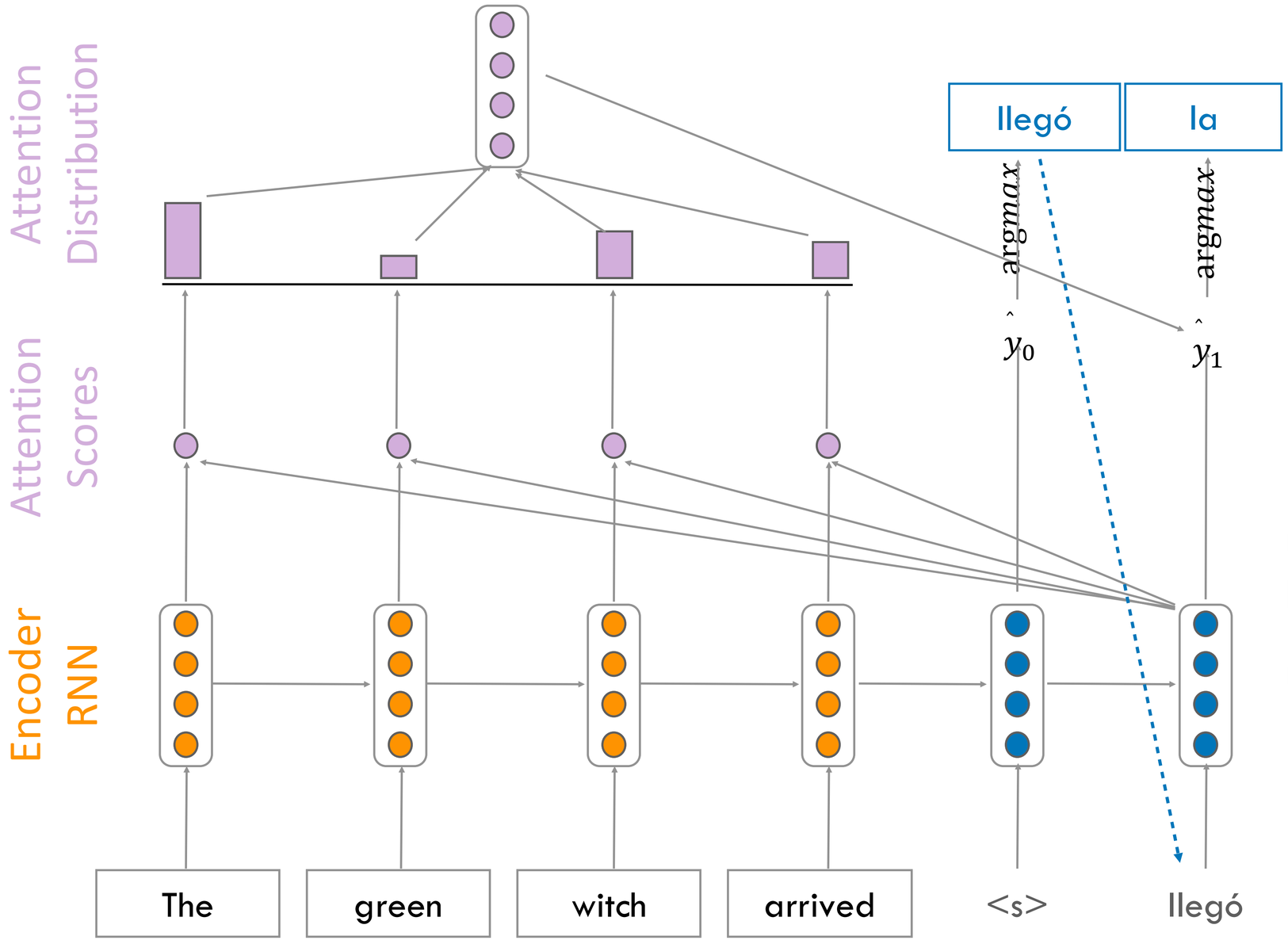
The attention output mostly contains information the hidden states that received high attention.

Source Sentence $\mathbf{x}$



Concatenate attention output with decoder hidden state, then use to compute $\hat{y}_0$ as before

Source Sentence $\mathbf{x}$

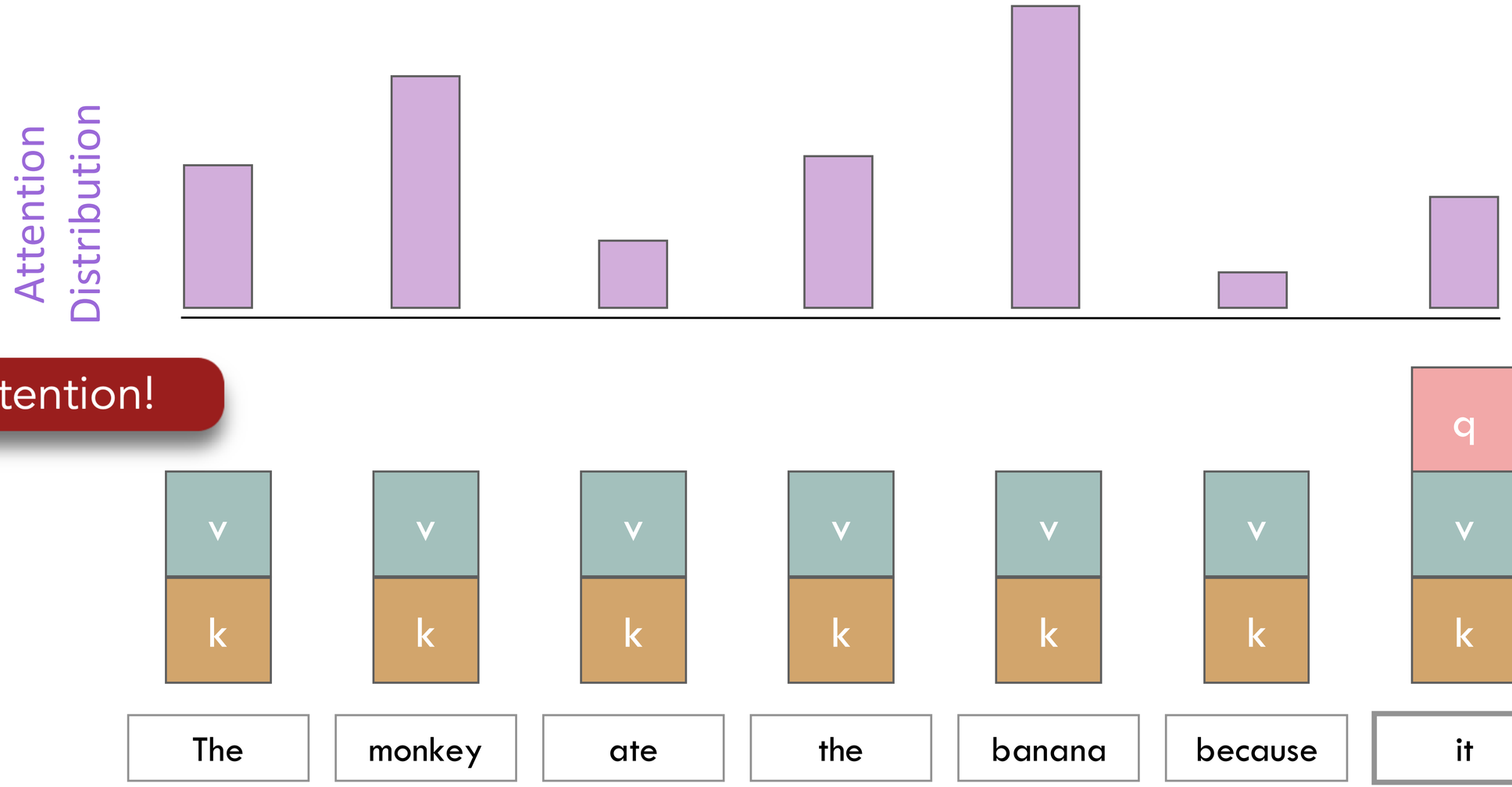


Query 2: Decoder, second time step

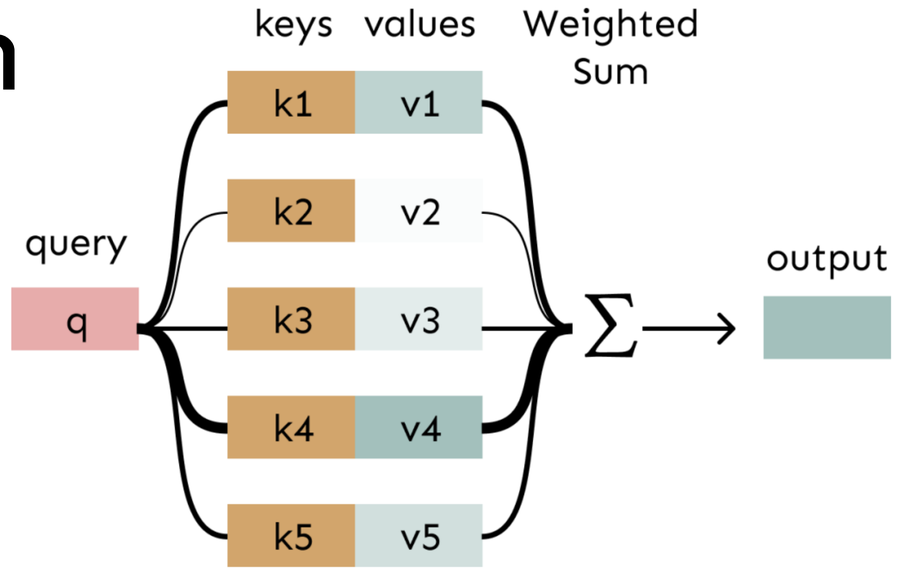
Source Sentence **x**

Transformers: Self-Attention

Attention in the decoder



Self-Attention



Keys, Queries, Values from the same sequence

Let $\mathbf{w}_{1:N}$ be a sequence of words in vocabulary V

For each $\mathbf{w}_i$, let $\mathbf{x}_i = \mathbf{E}_{w_i}$, where $\mathbf{E} \in \mathbb{R}^{d \times V}$ is an embedding matrix.

1. Transform each word embedding with weight matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V}$, each in $\mathbb{R}^{d \times d}$

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

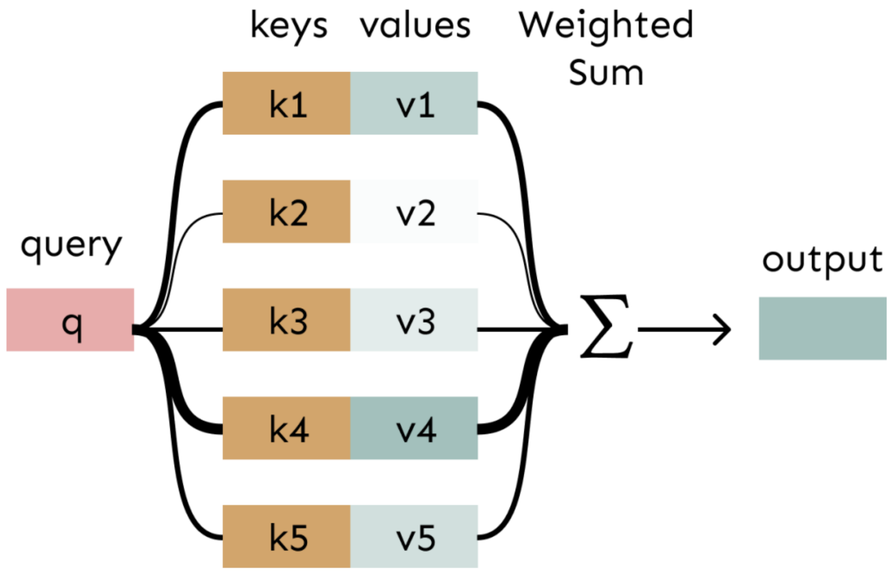
2. Compute pairwise similarities between keys and queries; normalize with softmax

$$\boldsymbol{e_{ij}} = q_i^\top \boldsymbol{k_j} \qquad \boldsymbol{\alpha_{ij}} = \frac{\exp(\boldsymbol{e_{ij}})}{\sum_{j'} \exp(\boldsymbol{e_{ij'}})}$$

3. Compute output for each word as weighted sum of values

$$\boldsymbol{o_i} = \sum_j \boldsymbol{\alpha_{ij}} \, v_i$$

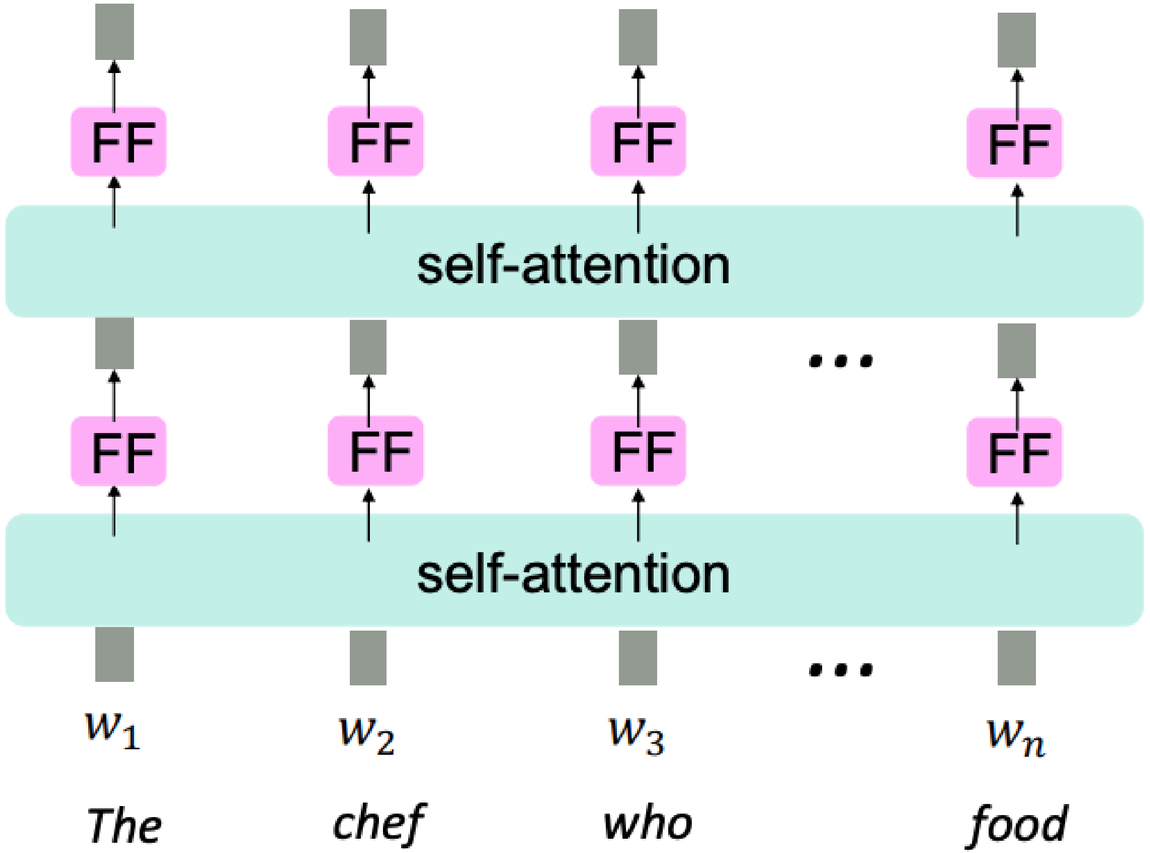
Why Self-Attention?



- Self-attention allows a network to directly extract and use information from arbitrarily large contexts without the need to pass it through intermediate recurrent connections as in RNNs

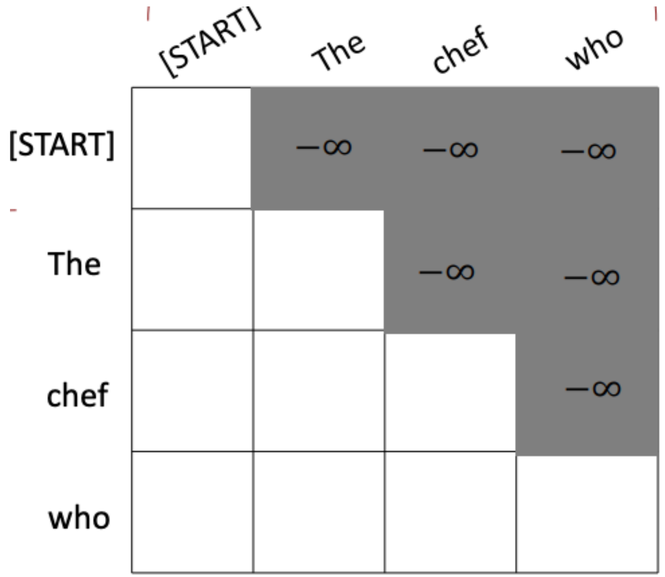
Self-Attention and Weighted Averages

- Problem: there are no element-wise **nonlinearities** in self-attention; stacking more self-attention layers just re-averages value vectors
- Solution: add a feed-forward network to post-process each output vector.



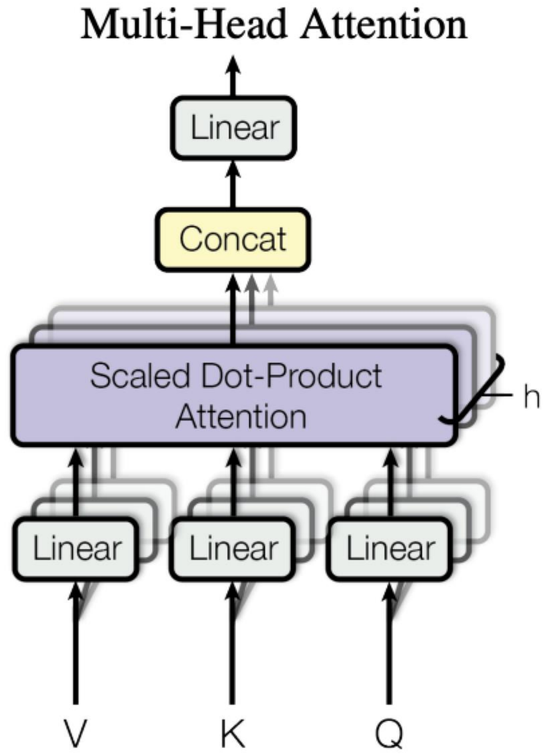
Self Attention and Future Information

- Problem: Need to ensure we don't "look at the future" when predicting a sequence
  - e.g. Target sentence in machine translation or generated sentence in language modeling
  - To use self-attention in decoders, we need to ensure we can't peek at the future.
- Solution (Naïve): At every time step, we could change the set of keys and queries to include only past words.
  - (Inefficient!)
- Solution: To enable parallelization, we mask out attention to future words by setting attention scores to $-\infty$



Multi-headed attention

- What if we want to look in multiple places in the sentence at once?
  - For word i, self-attention "looks" where $\mathbf{x}_i^T \mathbf{Q}^T (\mathbf{K}\mathbf{x}_j)$ is high, but maybe we want to focus on different j for different reasons?
- We'll define multiple attention "heads" through multiple $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ matrices
- Let $\mathbf{Q}_l, \mathbf{K}_l, \mathbf{V}_l$, each in $\mathbb{R}^{d \times \frac{d}{h}}$, where h is the number of attention heads, and $1 \leq l \leq h$.
- Each attention head performs attention independently:
- Then the outputs of all the heads are combined!

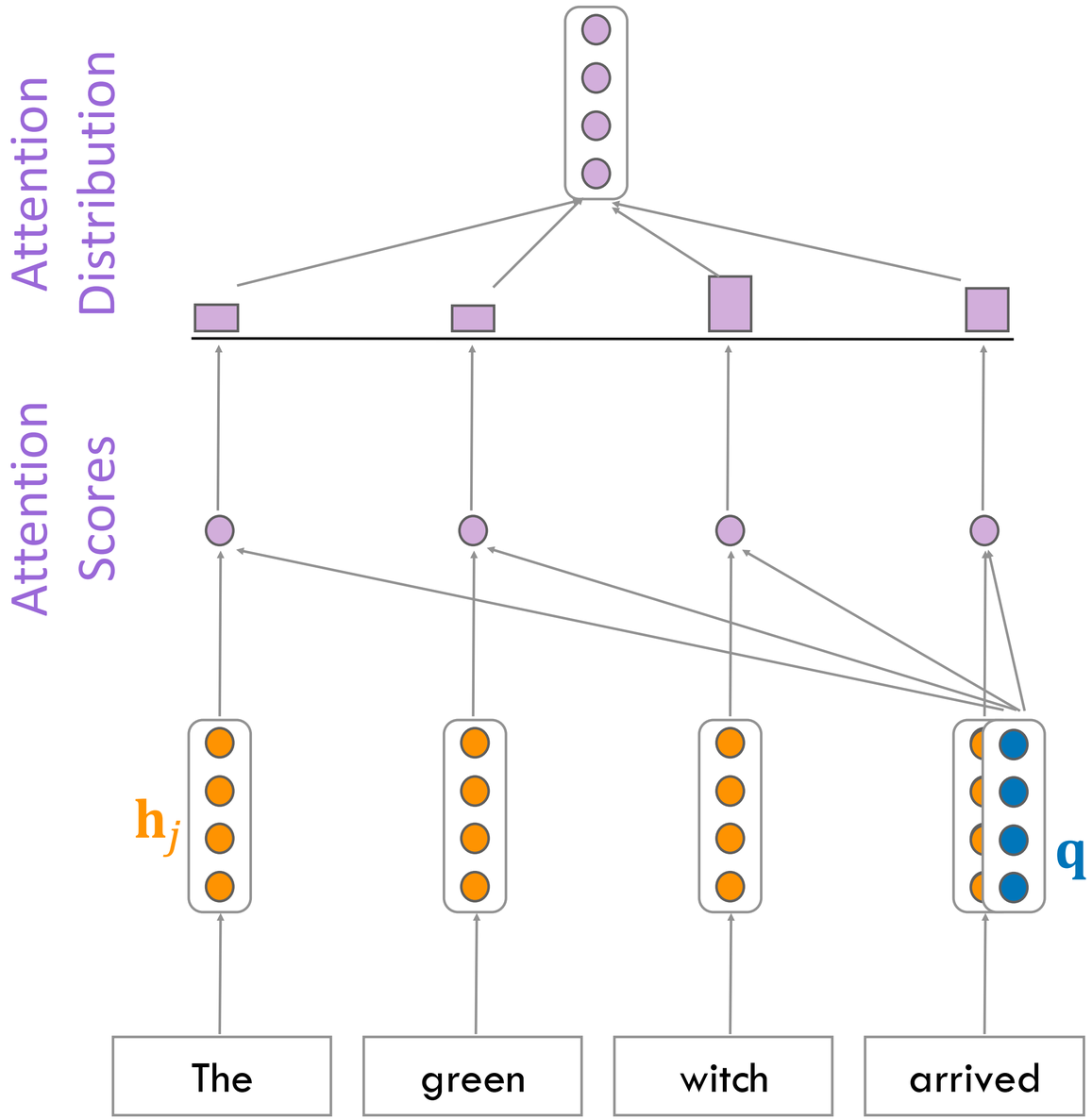


Each head gets to "look" at different things, and construct value vectors differently

Self-Attention: Order Information?

- Not necessarily (and not typically) based on Recurrent Neural Nets
- No more order information!
- Since self-attention doesn't build in order information, we need to encode the order of the sentence in our keys, queries, and values.

Do feedforward nets contain order information?



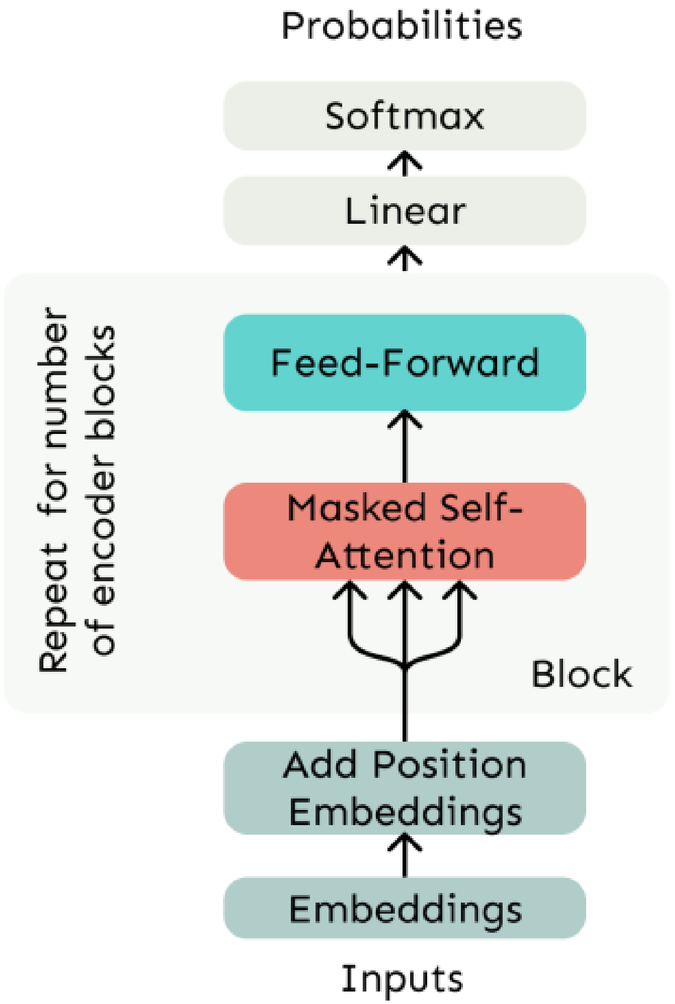
Positional Embeddings

- Maps integer inputs (for positions) to real-valued vectors
    - one per position in the entire context
- Can be randomly initialized and can let all $\mathbf{p}_i$ be learnable parameters (most common)
- Pros:
    - Flexibility: each position gets to be learned to fit the data
- Cons:
    - Definitely can't extrapolate to indices outside $1, \ldots, n$.
    - There will be plenty of training examples for the initial positions in our inputs and correspondingly fewer at the outer length limits

Putting it all together: Transformer Blocks

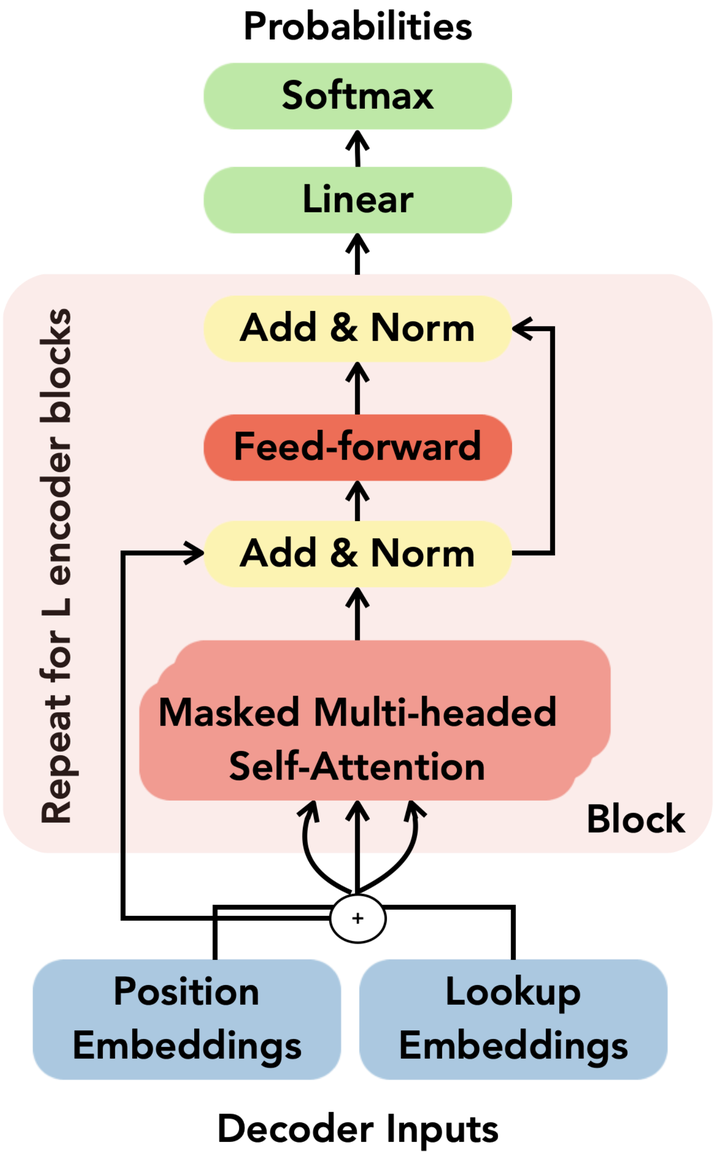
Self-Attention Transformer Building Block

- Self-attention:
  - the basis of the method; with multiple heads
- Position representations:
  - Specify the sequence order, since self-attention is an unordered function of its inputs.
- Nonlinearities:
  - At the output of the self-attention block
  - Frequently implemented as a simple feedforward network.
- Masking:
  - In order to parallelize operations while not looking at the future.
  - Keeps information about the future from "leaking" to the past.



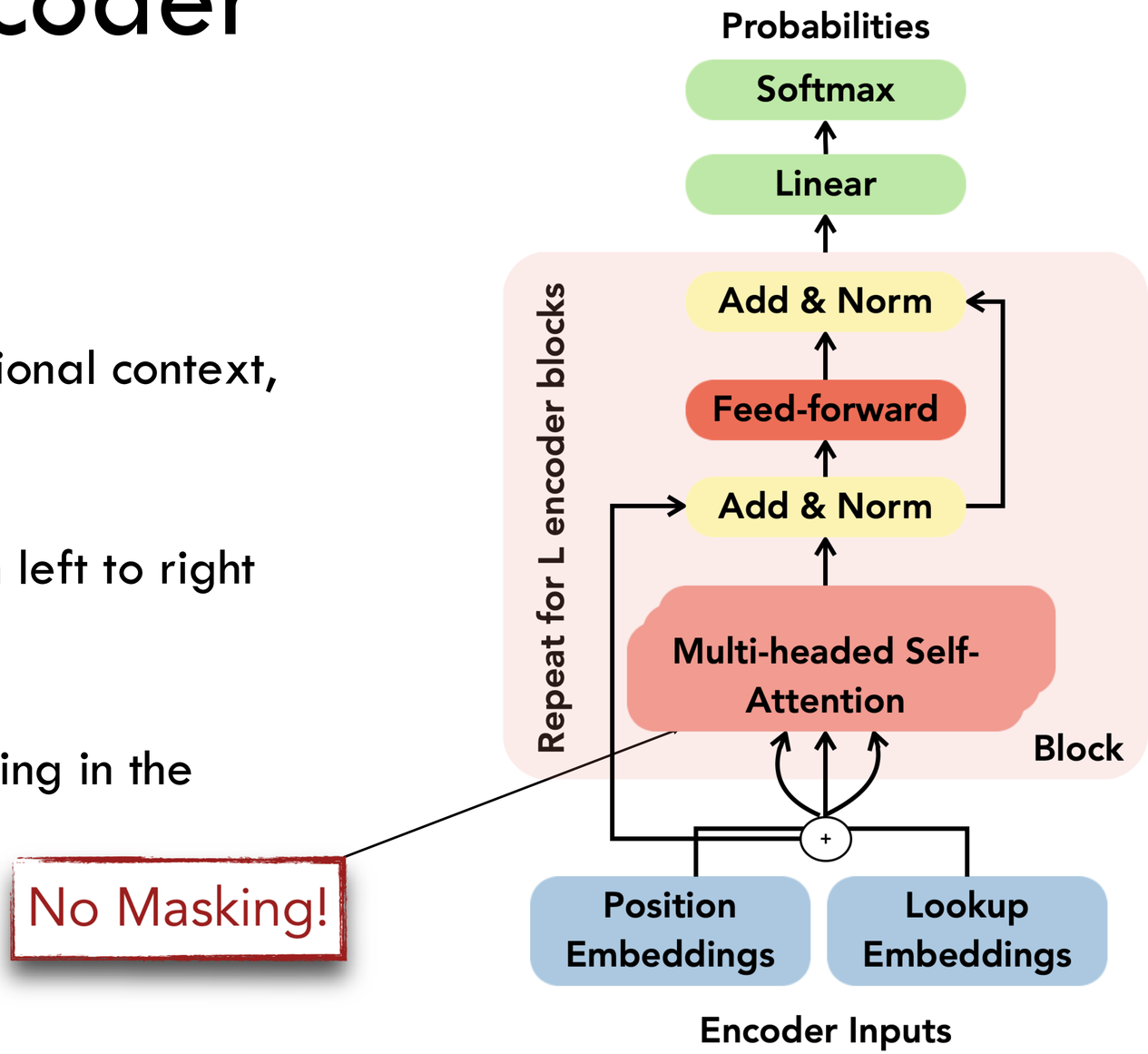
The Transformer Decoder

- The Transformer Decoder is a stack of Transformer Decoder Blocks.
- Each Block consists of:
  - Self-attention
  - Add & Norm
  - Feed-Forward
  - Add & Norm
- Output layer is as always a softmax layer



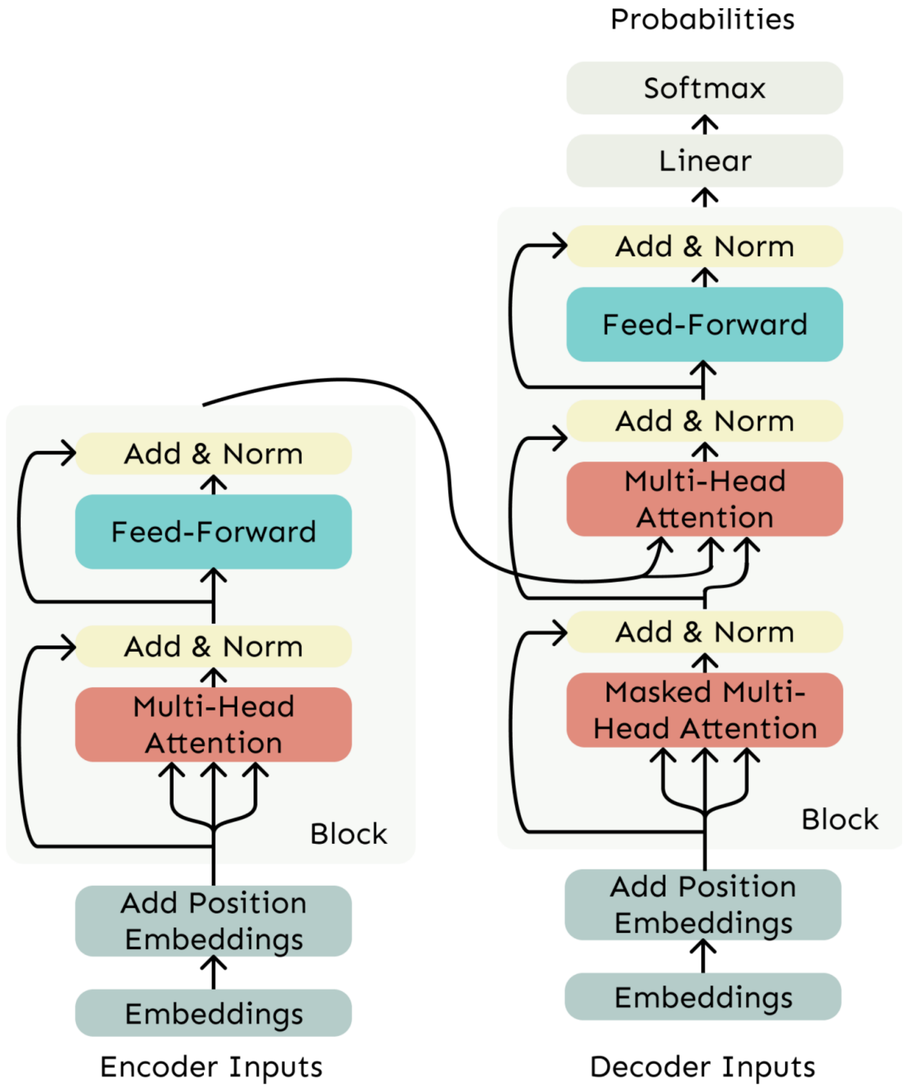
The Transformer Encoder

- The Transformer **Decoder** constrains to unidirectional context, as for language models.
- What if we want **bidirectional** context, i.e. both left to right as well as right to left?
- The only difference is that we **remove** the masking in the self-attention.



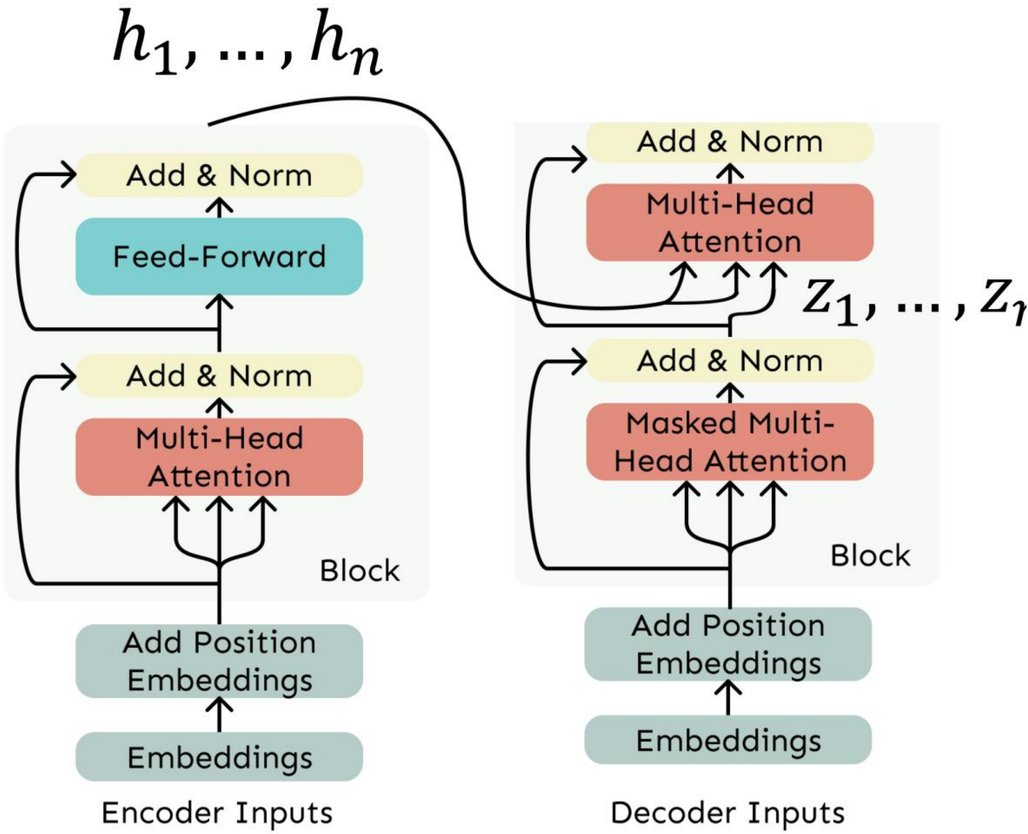
The Transformer Encoder-Decoder

- For this kind of seq2seq format, we often use a Transformer Encoder-Decoder.
- We use a normal Transformer Encoder.
- Our Transformer Decoder is modified to perform cross-attention to the output of the Encoder.



Cross Attention

- We saw that self -attention is when keys, queries, and values come from the same source.
- In the decoder, we have attention that looks more like what we saw last week.
- Let $\mathbf{h}_1, \ldots, \mathbf{h}_n$ be output vectors from the Transformer encoder; $\mathbf{h}_i \in \mathbb{R}^d$
- Let $\mathbf{z}_1, \ldots, \mathbf{z}_n$ be input vectors from the Transformer decoder, $\mathbf{h}_i \in \mathbb{R}^d$
- Then keys and values are drawn from the encoder (like a memory):
  - $\mathbf{k}_i = \mathbf{K}\mathbf{h}_i, \mathbf{v}_i = \mathbf{V}\mathbf{h}_i$
- And the queries are drawn from the decoder, $\mathbf{q}_i = \mathbf{Q}\mathbf{z}_i$



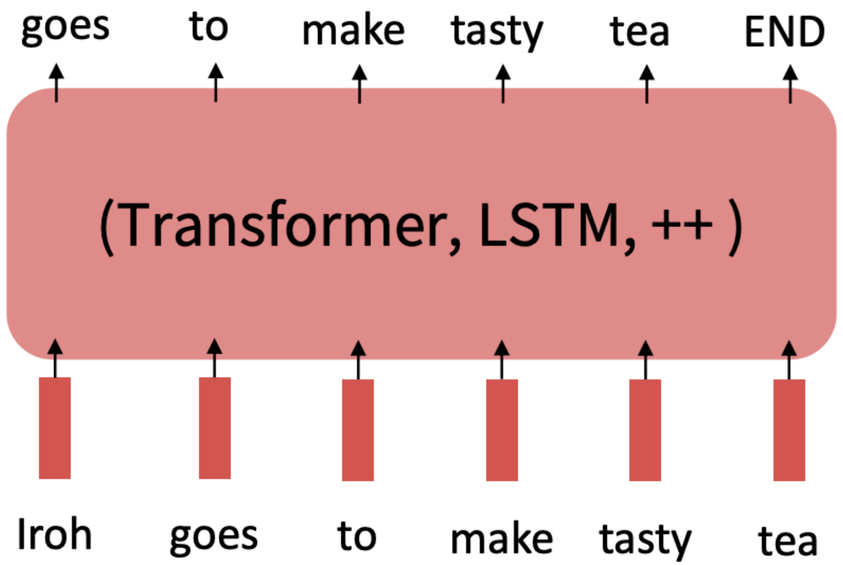
The Pre-training and Fine-tuning Paradigm

The Pretraining / Finetuning Paradigm

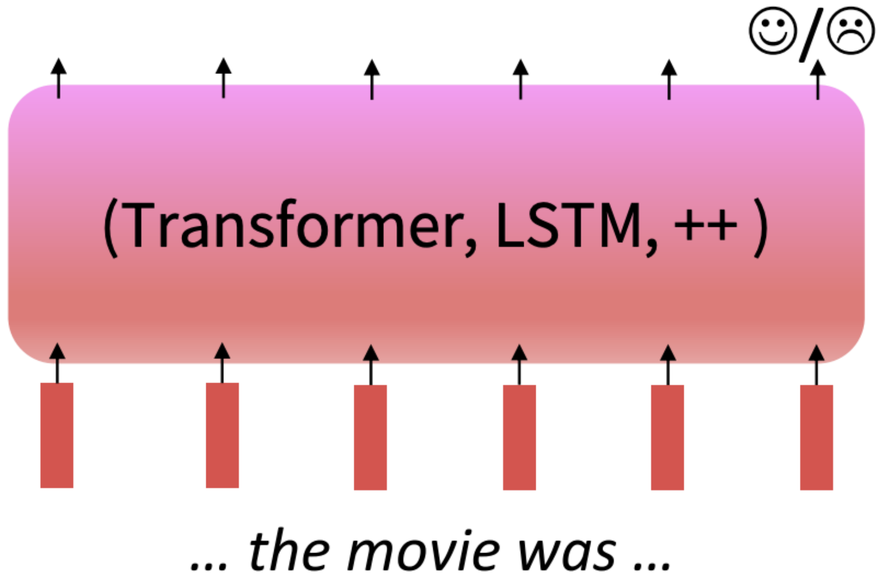
- Pretraining can improve NLP applications by serving as parameter initialization.

Key idea: "Pretrain once, finetune many times."

Step 1: Pretrain (on language corpora)
Lots of text; learn general things!

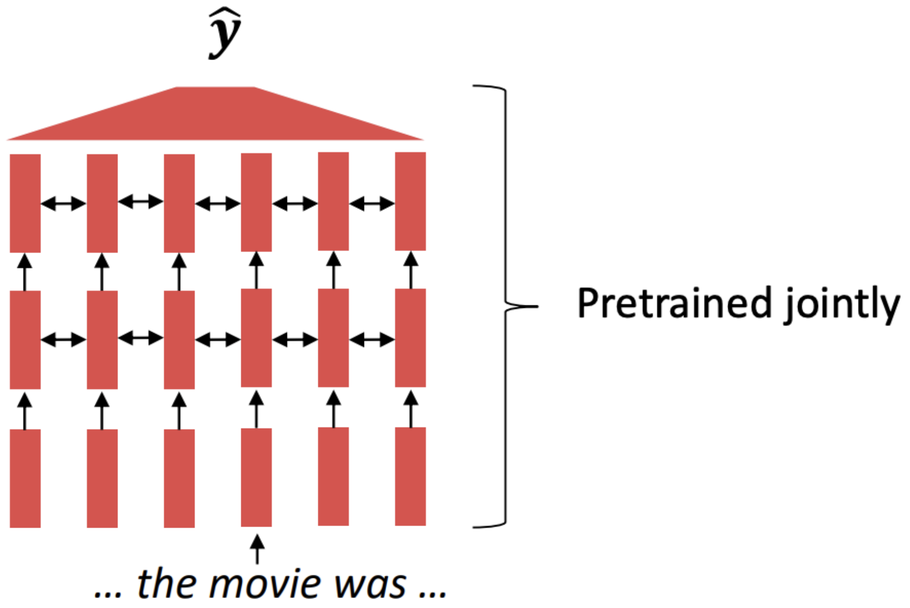


Step 2: Finetune (on your task data)
Not many labels; adapt to the task!



Pretraining Entire Models

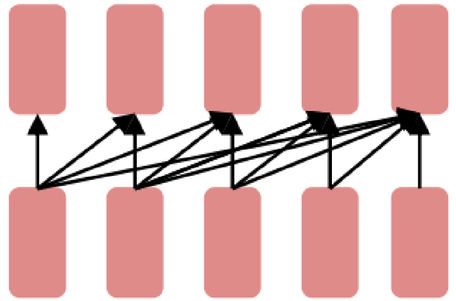
- In modern NLP:
  - All (or almost all) parameters in NLP networks are initialized via pretraining.
  - This has been exceptionally effective at building strong:
    - representations of language
    - parameter initializations for strong NLP models.



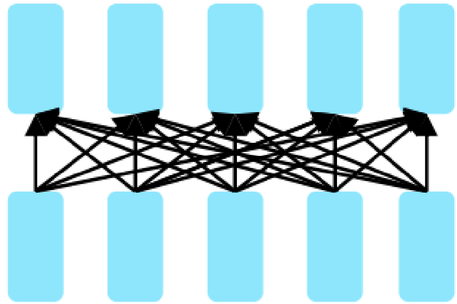
[This model has learned how to represent entire sentences through pretraining]

Pretraining

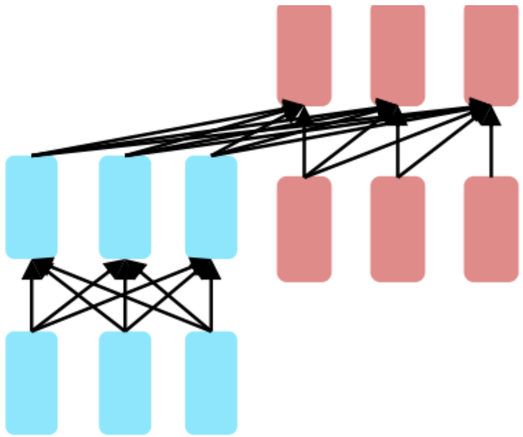
- Not restricted to language modeling! Can be any task
- But most successful if the task definition is very general. Hence, language modeling is a great pretraining option
- Three options!



Decoders

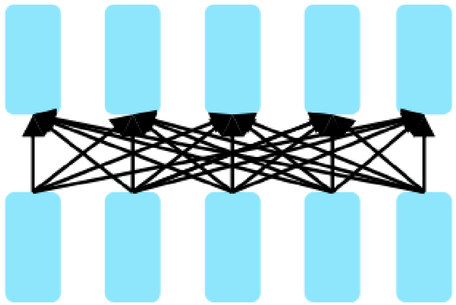


Encoders



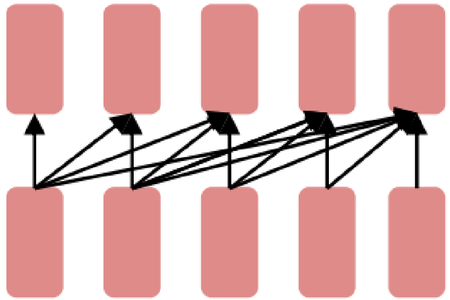
Encoder-Decoders

Pretraining for three types of architectures



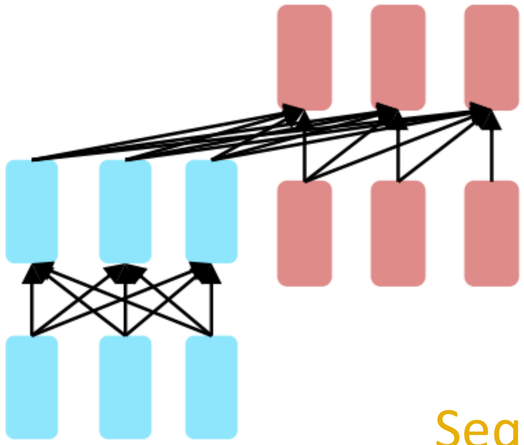
Encoders

Bidirectional Context



Decoders

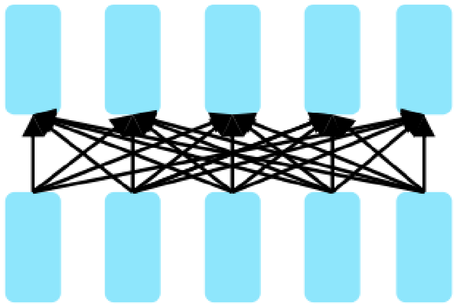
Language Models



Encoder-Decoders

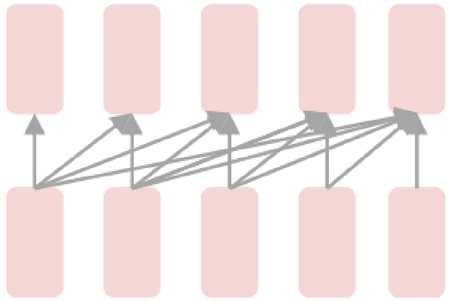
Sequence-to-sequence

Pretraining for three types of architectures



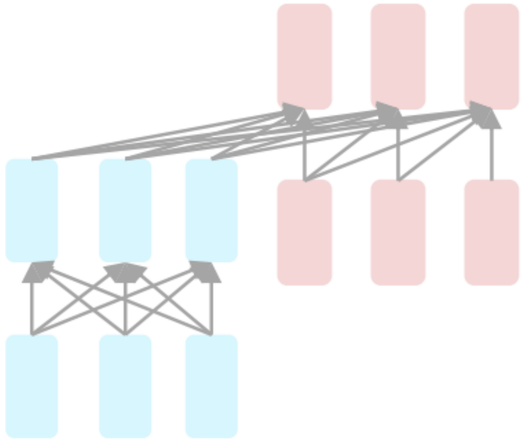
Encoders

Bidirectional Context



Decoders

Language Models



Encoder-Decoders

Sequence-to-sequence

Pretraining Encoders: Bidirectional Context

I was thinking about the sequence that goes 1, 1, 2, 3, 5, 8, 13, 21, ____

Universal Studios Theme Park is located in _____, California

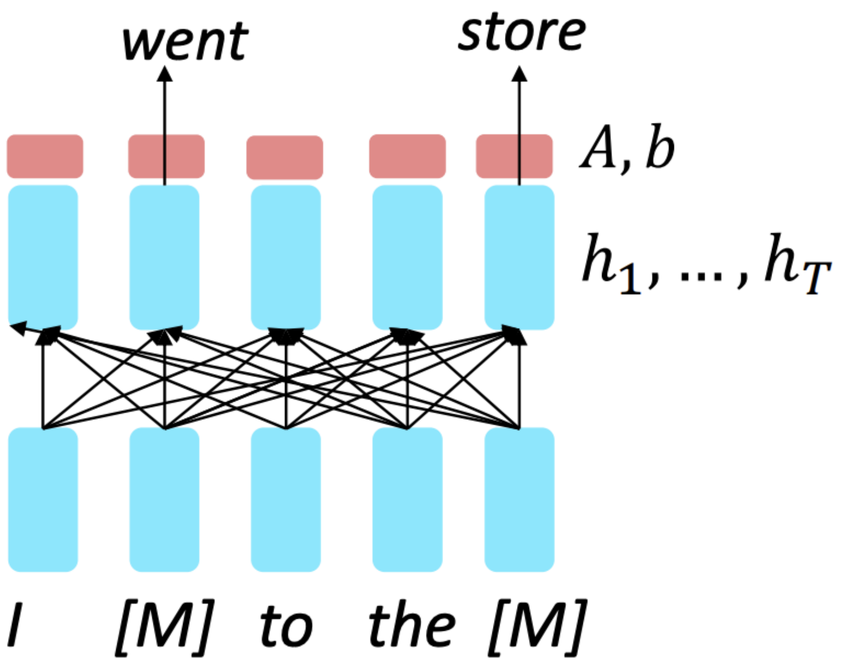
Problem: Input Reconstruction

'Cause darling i'm a _____ dressed like a daydream

Bidirectional context is important to reconstruct the input!

Pretraining Encoders: Objective

- Encoders get bidirectional context, so we can't do autoregressive language modeling!
- Idea: replace some fraction of words in the input with a special [MASK] token; predict these words.
  - $h_1, \ldots, h_T = Encoder(w_1, \ldots, w_T)$
  - $y_i \approx Ah_i + b$



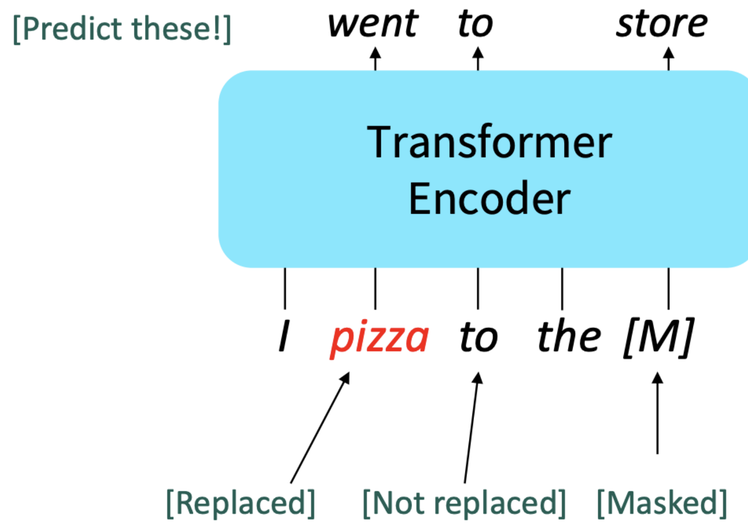
- Only add loss terms from words that are "masked out."
- If $\tilde{x}$ is the masked version of x, we're learning $p_\theta(\tilde{x}|x)$.
- Called Masked LM
- Special type of language modeling

Masked Language Modeling

BERT: Bidirectional Encoder Representations from Transformers

Devlin et al., 2018 proposed the "Masked LM" objective and released BERT, a Transformer, pretrained to:

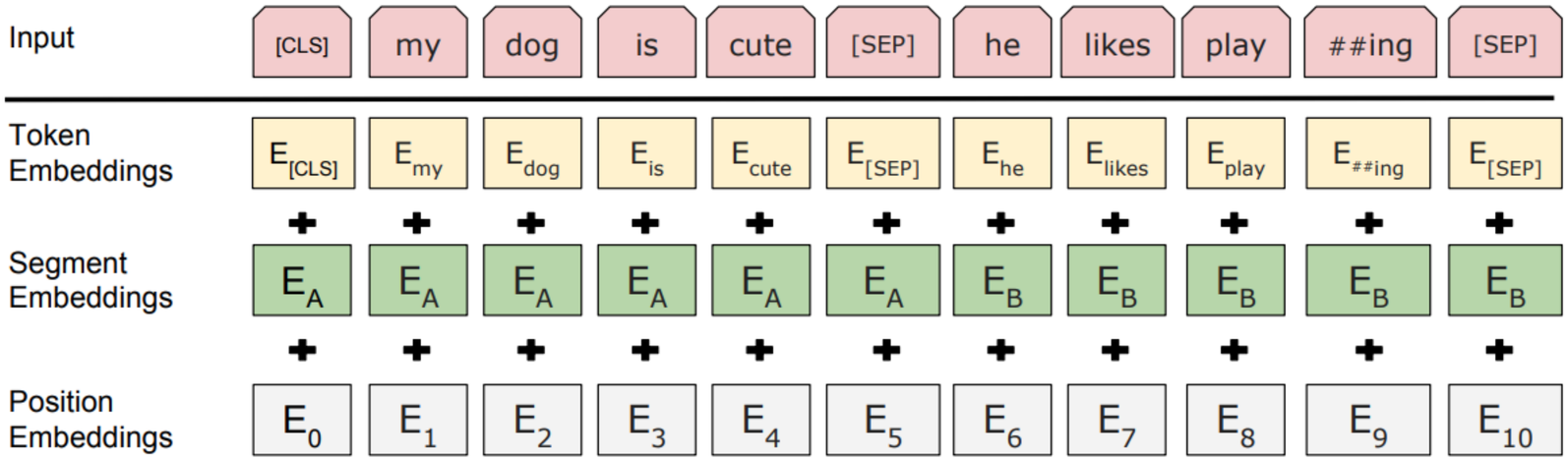
- 15% of the input tokens in a training sequence are sampled for learning, these are to be predicted by the model
- Of these
  - 80% are replaced with [MASK]
  - 10% are replaced with randomly selected tokens,
  - Remaining 10% are left unchanged



Why? Doesn't let the model get complacent and not build strong representations of non-masked words. (No masks are seen at fine-tuning time!)

BERT: Bidirectional Encoder Representations from Transformers

- The pretraining input to BERT was two separate contiguous chunks of text:

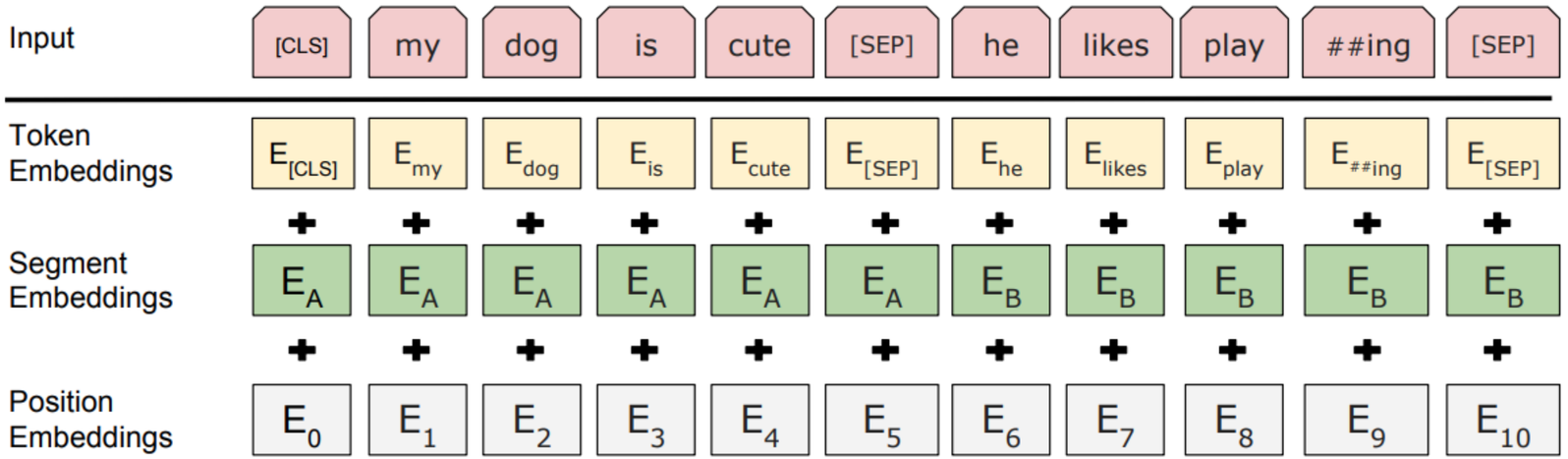


- BERT was trained to predict whether one chunk follows the other or is randomly sampled.
  - [CLS] and [SEP] tokens
  - [SEP] is used for next sentence prediction - do these sentences follow each other?
  - [CLS] for text classification / connection to fine-tuning

BERT: Training Details

- Two models were released:
  - BERT-base: 12 layers, 768-dim hidden states, 12 attention heads, 110 million params.
  - BERT-large: 24 layers, 1024-dim hidden states, 16 attention heads, 340 million params.
- Trained on:
  - BooksCorpus (800 million words)
  - English Wikipedia (2,500 million words)
- Pretraining is expensive and impractical on a single GPU.
  - BERT was pretrained with 64 TPU chips for a total of 4 days.
    - (TPUs are special tensor operation acceleration hardware)
- Finetuning is practical and common on a single GPU
  - "Pretrain once, finetune many times."

BERT: Contextual Embeddings



- BERT results in contextual embeddings
  - Embeddings for tokens in context, not just word type embeddings like word2vec, GloVe
  - Can be used for measuring the semantic similarity of two words in context
  - Useful in linguistic tasks that require precise models of word meaning

BERT: Results

- BERT was massively popular and hugely versatile; finetuning BERT led to new state-of-the-art results on a broad range of tasks.

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Various Text Classification tasks like sentiment classification

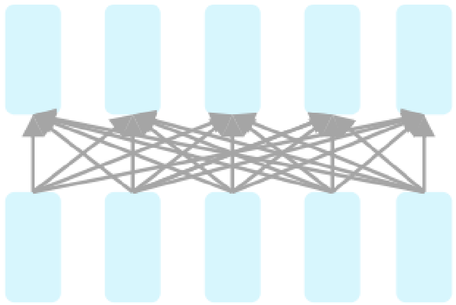
BERT: Extensions

- Some generally accepted improvements to the BERT pretraining formula:
  - RoBERTa: mainly just train BERT for longer and remove next sentence prediction!
  - SpanBERT: masking contiguous spans of words makes a harder, more useful pretraining task
- A lot of BERT variants that used the BERT formula
  - ALBERT: BERT with parameter-reduction techniques
  - DistilBERT:
  - DeBERTa: Decoding-enhanced BERT with disentangled attention
  - FlauBERT: BERT for French
  - XLNet: Multilingual BERT
  - Etc.
- BERTology: How and why BERT worked so well

BERT: Overview

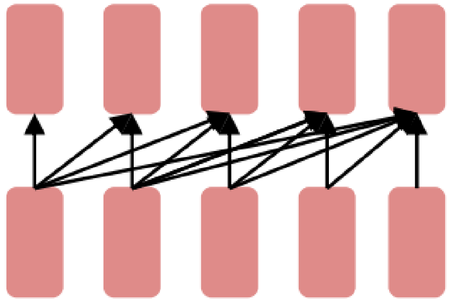
- [SEP]: Later work has argued this "next sentence prediction" is not necessary
- In general, more compute, more data can improve pretraining even when not changing the underlying Transformer encoder
- Results in contextual embeddings
- Key Limitation:
  - Cannot be used for generation
  - No pretraining encoders can be used for autoregressive generation very naturally
    - For this, we need to have a decoder!

Pretraining for three types of architectures



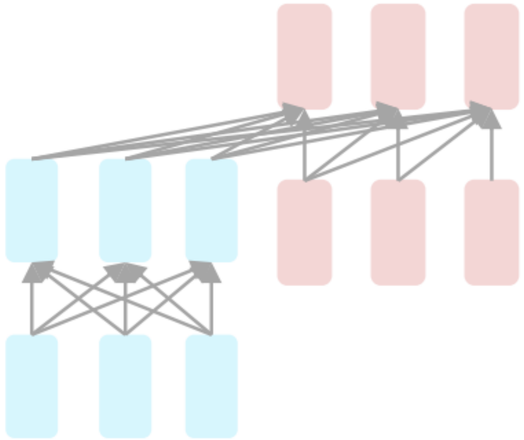
Encoders

Bidirectional Context



Decoders

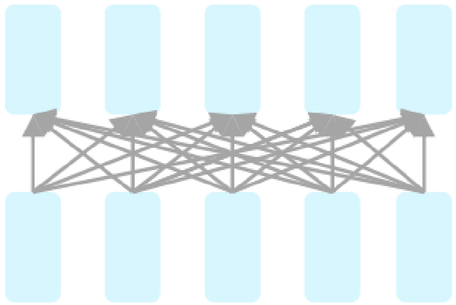
Language Models



Encoder-Decoders

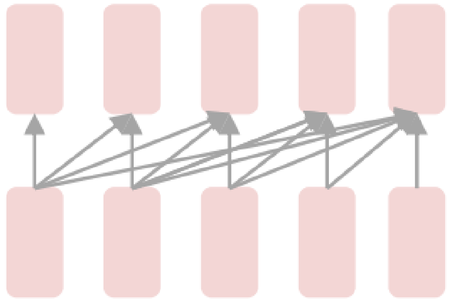
Sequence-to-sequence

Pretraining for three types of architectures



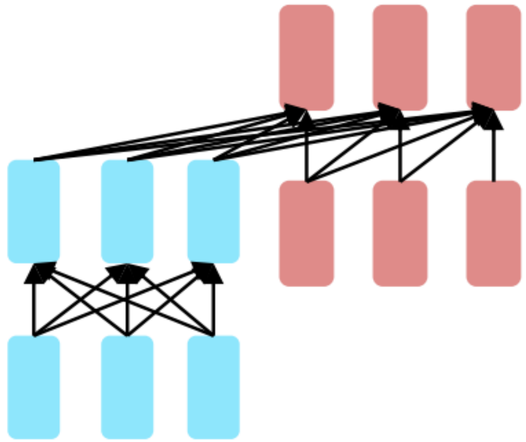
Encoders

Bidirectional Context



Decoders

Language Models



Encoder-Decoders

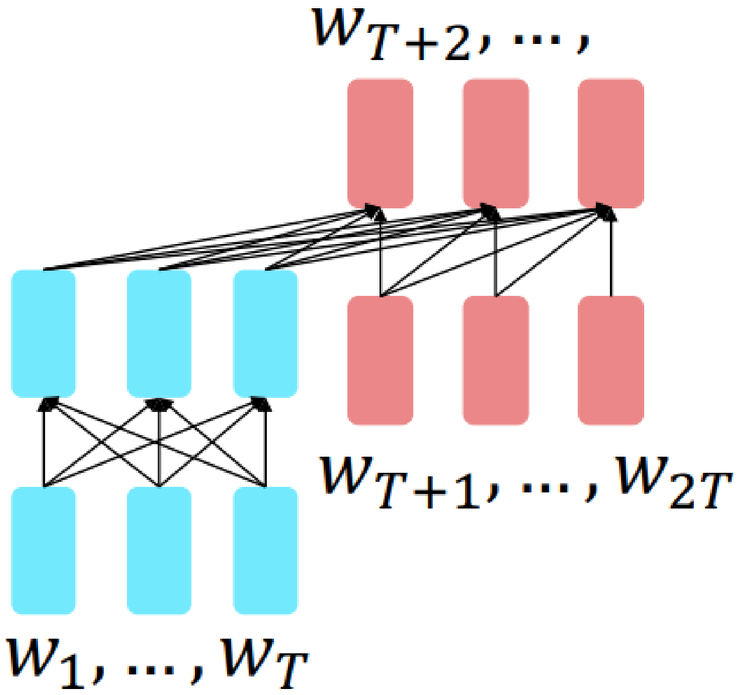
Sequence-to-sequence

Pretraining Encoder-Decoder Models

- For encoder-decoders, we could do something like language modeling, but where a prefix of every input is provided to the encoder and is not predicted.

$$h_1, \ldots, h_T = \text{Encoder}(w_1, \ldots, w_T)$$
$$h_{T+1}, \ldots, h_2 = Decoder(w_1, \ldots, w_T, h_1, \ldots, h_T)$$
$$y_i \sim Ah_i + b, i > T$$

The encoder portion benefits from bidirectional context; the decoder portion is used to train the whole model through language modeling.



T5: A Pretrained Encoder-Decoder Model

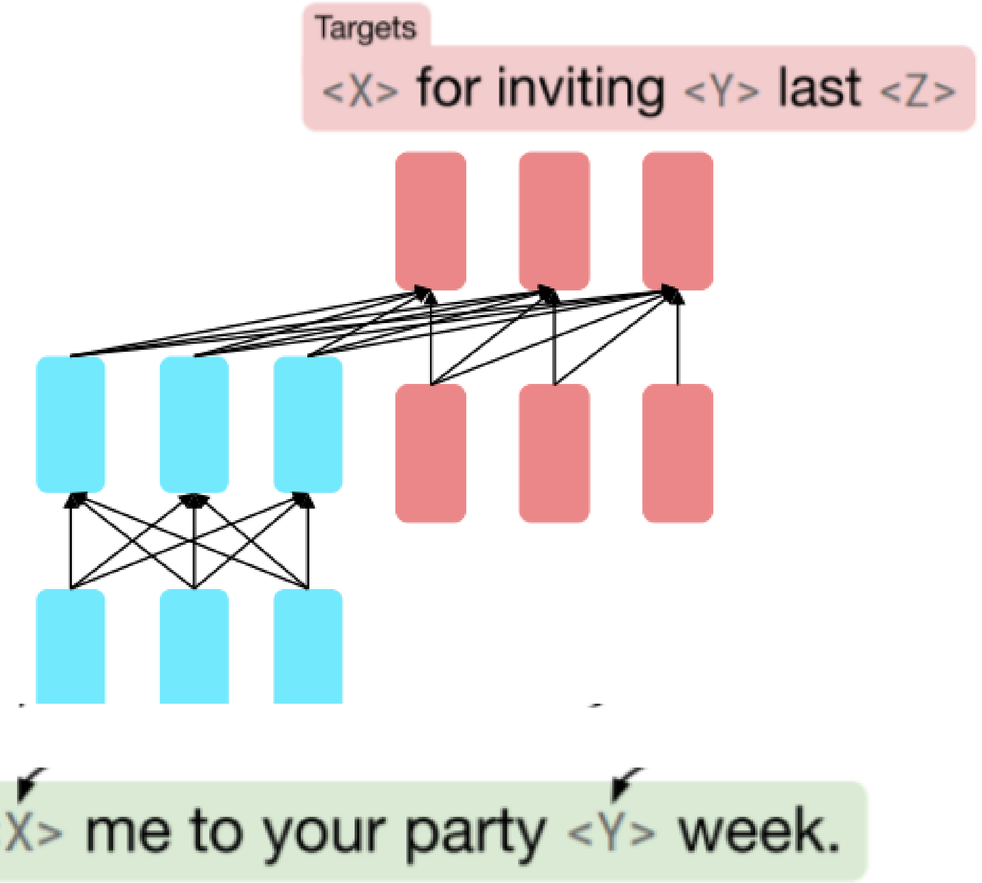
- Raffel et al., 2018 built T5, which uses as a span corruption pretraining objective

Replace different-length spans from the input with unique placeholders; decode out the spans that were removed!

Original text

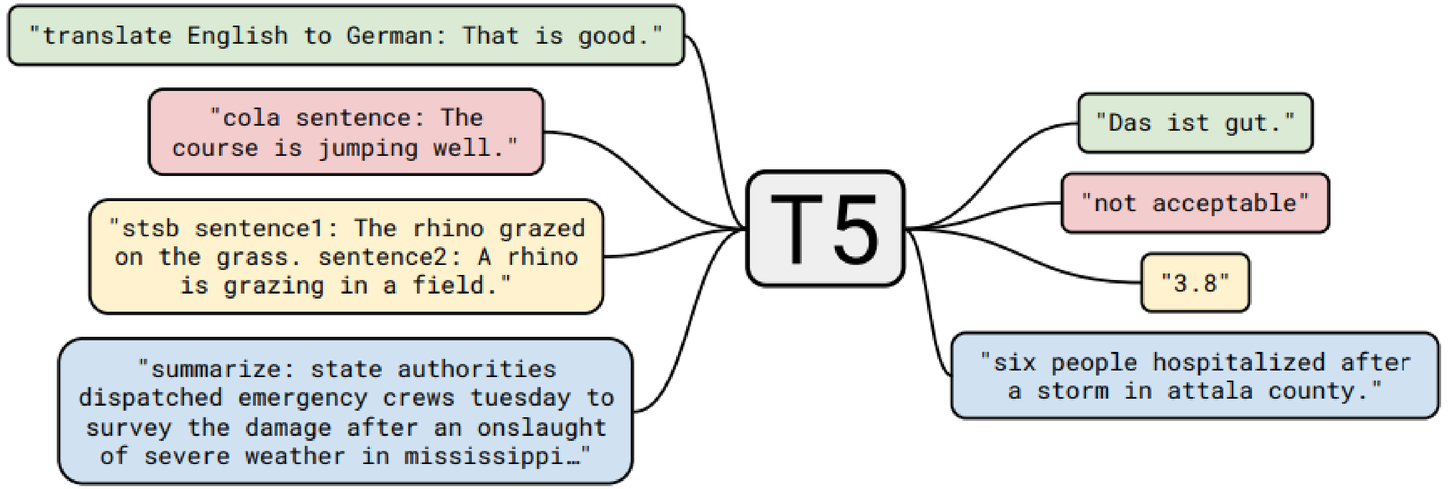
Thank you ~~for inviting~~ me to your party ~~last~~ week.

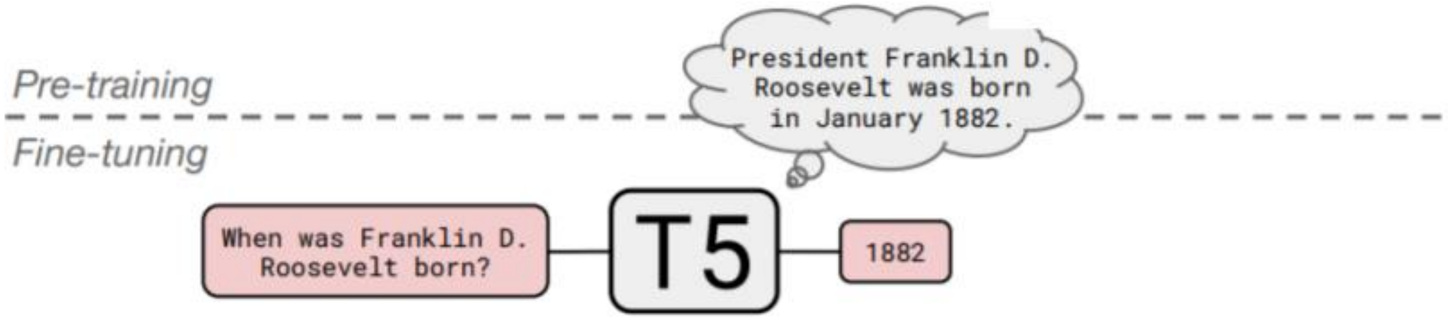
Still uses an objective that looks like language modeling at the decoder side.



T5: Task Preparation

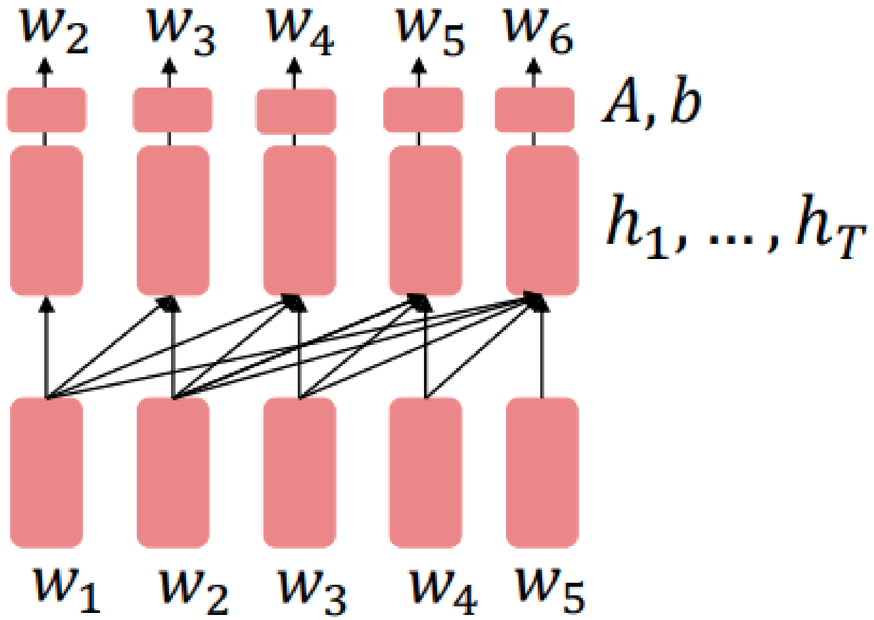
T5 can be finetuned to answer a wide range of tasks, where the input and output are expressed as a sequence of tokens





Pretraining Decoders: Generators

- More natural: pretrain decoders as language models and then use them as generators, finetuning their $p_\theta(w_t|w_{1:t-1})$
  - $h_1, \ldots, h_T = Decoder(w_1, \ldots, w_T)$
- $w_t \approx Ah_{t-1} + b$
- Where A, b were pretrained in the language model!



The linear layer has been pretrained

- This is helpful in tasks where the output is a sequence at pretraining time!
  - Dialogue (context=dialogue history)
  - Summarization (context=document)

Generative Pretrained Transformer (GPT)

- 2018's GPT was a big success in pretraining a decoder!
  - Transformer decoder with 12 layers, 117M parameters.
  - 768-dimensional hidden states, 3072-dimensional feed-forward hidden layers.
  - Byte-pair encoding with 40,000 merges
  - Trained on BooksCorpus: over 7000 unique books.
    - Contains long spans of contiguous text, for learning long-distance dependencies.
  - The acronym "GPT" never showed up in the original paper; it could stand for "Generative PreTraining" or "Generative Pretrained Transformer"

OpenAI

[Radford et al., 2018]

Adapting GPT

- Natural Language Inference: Label pairs of sentences as entailing/contradictory/neutral
  - Premise: The man is in the doorway
  - Hypothesis: The person is near the door

- Radford et al., 2018 evaluate on natural language inference by formatting the input as a sequence of tokens for the decoder
  - [START] The man is in the doorway [DELIM] The person is near the door [EXTRACT]
  - The linear classifier is applied to the representation of the **[EXTRACT] token.**

[Radford et al., 2018]

GPT: Results on Classification

- Outperforms Recurrent Neural Nets

Method	MNLI-m	MNLI-mm	SNLI	SciTail	QNLI	RTE
ESIM + ELMo [44] (5x)	-	-	89.3	-	-	-
CAFE [58] (5x)	80.2	79.0	89.3	-	-	-
Stochastic Answer Network [35] (3x)	80.6	80.1	-	-	-	-
CAFE [58]	78.7	77.9	88.5	83.3		
GenSen [64]	71.4	71.3	-	-	82.3	59.2
Multi-task BiLSTM + Attn [64]	72.2	72.1	-	-	82.1	**61.7**
Finetuned Transformer LM (ours)	**82.1**	**81.4**	**89.9**	**88.3**	**88.1**	56.0

[Radford et al., 2018]

GPT-2

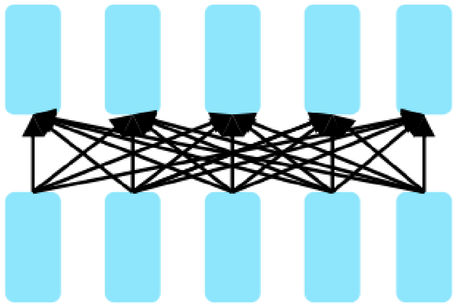
- GPT-2, a larger version (1.5B) of GPT trained on more data, was shown to produce relatively convincing samples of natural language.
- Moved away from classification, only generation

Context (human-written): In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.
GPT-2: The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science. Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved. Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

GPT-3 and beyond

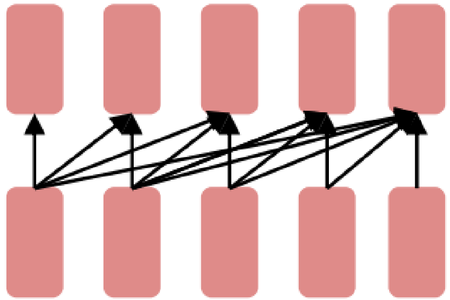
- Markedly improved generation capabilities by greatly increasing data and model scale
- Solving all tasks through generation
- Even obviating the need to fine-tune!

Pretraining for three types of architectures



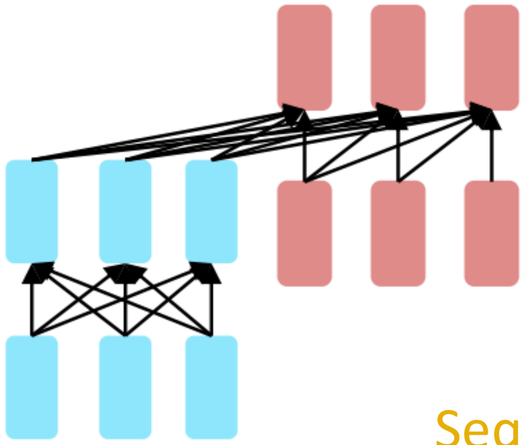
Encoders

Bidirectional Context



Decoders

Language Models



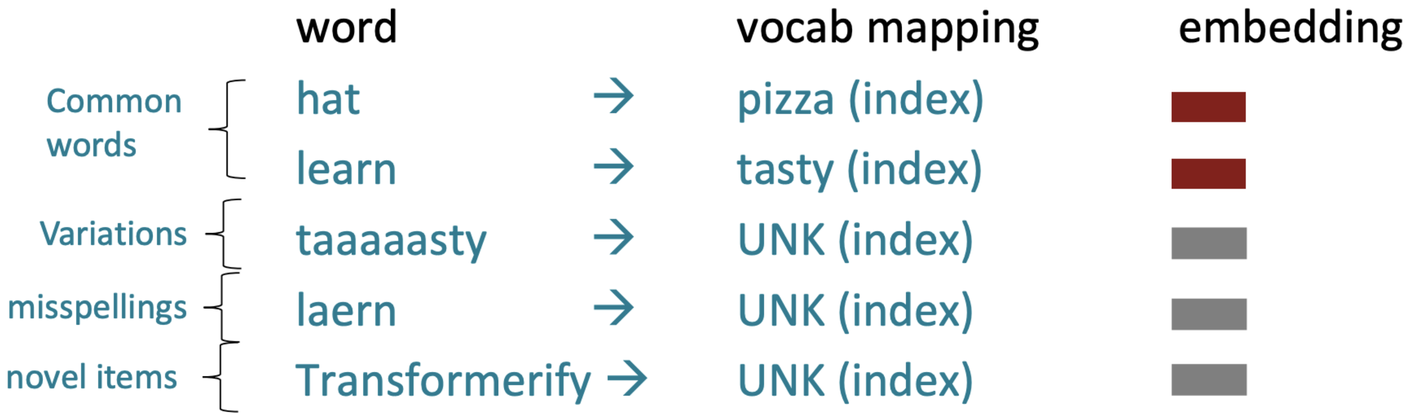
Encoder-Decoders

Sequence-to-sequence

Tokenization in Transformers

Word Structure and Subword Models

- So far, we have made some assumptions about a language's vocabulary
- We assume a fixed vocabulary of tens of thousands of words, built from the training set
- All novel words seen at test time are mapped to a single UNK



Word Structure in Language

- Finite vocabulary assumptions make even less sense in many languages.
  - Many languages exhibit complex morphology, or word structure.
  - The effect is: more word types, each occurring fewer times.

-ambia = to tell

Example: Swahili verbs can have hundreds of conjugations, each encoding a wide variety of information. (Tense, mood, definiteness, negation, information about the object, ++)

Conjugation of *-ambia*

Non-finite forms		
Form	Positive	Negative
Infinitive	kuambia	kutoambia
Simple finite forms		
Positive form	Singular	Plural
Imperative	ambia	ambieni
Habitual	huambia	

Complex finite forms

Polarity	Persons				Persons / Classes		Classes											
	1st		2nd		3rd / M-wa		M-mi		Ma		Ki-vi		N		U	Ku	Pa	Mu
	Sg.	Pl.	Sg.	Pl.	Sg. / 1	Pl. / 2	3	4	5	6	7	8	9	10	11 / 14	15 / 17	16	18
Past [less ▲]																		
Positive	niliambia naliambia	tuliambia twaliambia	uliambia waliambia	mliambia mwaliambia	aliambia	waliambia	uliambia	iliambia	liliambia	yaliambia	kiliambia	viliambia	iliambia	ziliambia	uliambia	kuliambia	paliambia	muliambia
Negative	sikuambia	hatukuambia	hukuambia	hamkuambia	hakuambia	hawakuambia	haukuambia	haikuambia	halikuambia	hayakuambia	hakikuambia	havikuambia	haikuambia	hazikuambia	haukuambia	hakukuambia	hapakuambia	hamukuambia
Present [less ▲]																		
Positive	ninaambia naambia	tunaambia	unaambia	mnaambia	anaambia	wanaambia	unaambia	inaambia	linaambia	yanaambia	kinaambia	vinaambia	inaambia	zinaambia	unaambia	kunaambia	panaambia	munaambia
Negative	siambii	hatuambii	huambii	hamambii	haambii	hawaambii	hauambii	haiambii	haliambii	hayaambii	hakiambii	haviambii	haiambii	haziambii	hauambii	hakuambii	hapaambii	hamuambii
Future [less ▲]																		
Positive	nitaambia	tutaambia	utaambia	mtaambia	ataambia	wataambia	utaambia	itaambia	litaambia	yataambia	kitaambia	vitaambia	itaambia	zitaambia	utaambia	kutaambia	pataambia	mutaambia
Negative	sitaambia	hatutaambia	hutaambia	hamtaambia	hataambia	hawataambia	hautaambia	haitaambia	halitaambia	hayataambia	hakitaambia	havitaambia	haitaambia	hazitaambia	hautaambia	hakutaambia	hapataambia	hamutaambia
Subjunctive [less ▲]																		
Positive	niambie	tuambie	uambie	mambie	aambie	waambie	uambie	iambie	liambie	yaambie	kiambie	viambie	iambie	ziambie	uambie	kuambie	paambie	muambie
Negative	nisiambie	tusiambie	usiambie	msiambie	asiambie	wasiambie	usiambie	isiambie	lisiambie	yasiambie	kisiambie	visiambie	isiambie	zisiambie	usiambie	kusiambie	pasiambie	musiambie
Present Conditional [less ▲]																		
Positive	ningeambia	tungeambia	ungeambia	mngeambia	angeambia	wangeambia	ungeambia	ingeambia	lingeambia	yangeambia	kingeambia	vingeambia	ingeambia	zingeambia	ungeambia	kungeambia	pangeambia	mungeambia
Negative	nisingeambia singeambia	tusingeambia hatungeambia	usingeambia hungeambia	msingeambia hamngeambia	asingeambia hangeambia	wasingeambia hawangeambia	usingeambia haungeambia	isingeambia haingeambia	lisingeambia halingeambia	yasingeambia hayangeambia	kisingeambia hakingeambia	visingeambia havingeambia	isingeambia haingeambia	zisingeambia hazingeambia	usingeambia haungeambia	kusingeambia hakungeambia	pasingeambia hapangeambia	musingeambia hamungeambia
Past Conditional [less ▲]																		
Positive	ningaliambia	tungaliambia	ungaliambia	mngaliambia	angaliambia	wangaliambia	ungaliambia	ingaliambia	lingaliambia	yangaliambia	kingaliambia	vingaliambia	ingaliambia	zingaliambia	ungaliambia	kungaliambia	pangaliambia	mungaliambia
Negative	nisingaliambia singaliambia	tusingaliambia hatungaliambia	usingaliambia hungaliambia	msingaliambia hamngaliambia	asingaliambia hangaliambia	wasingaliambia hawangaliambia	usingaliambia haungaliambia	isingaliambia haingaliambia	lisingaliambia halingaliambia	yasingaliambia hayangaliambia	kisingaliambia hakingaliambia	visingaliambia havingaliambia	isingaliambia haingaliambia	zisingaliambia hazingaliambia	usingaliambia haungaliambia	kusingaliambia hakungaliambia	pasingaliambia hapangaliambia	musingaliambia hamungaliambia
Conditional Contrary to Fact [less ▲]																		
Positive	ningeliambia	tungeliambia	ungeliambia	mngeliambia	angeliambia	wangeliambia	ungeliambia	ingeliambia	lingeliambia	yangeliambia	kingeliambia	vingeliambia	ingeliambia	zingeliambia	ungeliambia	kungeliambia	pangeliambia	mungeliambia
Gnomic [less ▲]																		
Positive	naambia	twaambia	waambia	mwaambia	aambia	waambia	waambia	yaambia	laambia	yaambia	chaambia	vyaambia	yaambia	zaambia	waambia	kwaambia	paambia	mwaambia
Perfect [less ▲]																		

Source: Wiktionary

Subword Modeling

- Solution: look at subwords
- Subword modeling in NLP encompasses a wide range of methods for reasoning about structure below the word level
  - Subwords may be parts of words, characters, bytes
- The dominant modern paradigm is to learn a vocabulary of parts of words (subword tokens).
- At training and testing time, each word is split into a sequence of known subwords.

Byte-pair encoding

- Byte-pair encoding is a simple, effective strategy for defining a subword vocabulary
- Adapted for word segmentation from data compression technique (Gage, 1994)
  - Instead of merging frequent pairs of bytes, we merge characters or character sequences

Byte-pair encoding

- Algorithm:
    1. Start with a vocabulary containing only characters and an “end-of-word” symbol.
    2. Using a corpus of text, find the most common adjacent characters “a,b”; add “ab” as a subword.
    3. Replace instances of the character pair with the new subword; repeat until desired vocabulary size.

BPE in action

Corpus

low	lower	newest
low	lower	newest
low	widest	newest
low	widest	newest
low	widest	newest

Corpus

low</w>	lower</w>	newest</w>
low</w>	lower</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>

Corpus

l o w </w>	l o w e r </w>	n e w e s t </w>
l o w </w>	l o w e r </w>	n e w e s t </w>
l o w </w>	w i d e s t </w>	n e w e s t </w>
l o w </w>	w i d e s t</w>	n e w e s t </w>
l o w </w>	w i d e s t </w>	n e w e s t </w>

Vocabulary

d	e	i	l	n	o	s	t	w
es								

Frequency

d-e (3)	l-o (7)	t-</w> (8)
e-r (2)	n-e (5)	w-</w> (5)
e-s (8)	o-w (7)	w-e (7)
e-w (5)	r-</w> (2)	w-i (3)
i-d (3)	s-t (8)	

Source: https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/

BPE in action

Corpus

low	lower	newest
low	lower	newest
low	widest	newest
low	widest	newest
low	widest	newest

Corpus

low</w>	lower</w>	newest</w>
low</w>	lower</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>

Corpus

l o w </w>	l o w e r </w>	n e w es t </w>
l o w </w>	l o w e r </w>	n e w es t </w>
l o w </w>	w i d es t </w>	n e w es t </w>
l o w </w>	w i d es t</w>	n e w es t </w>
l o w </w>	w i d es t </w>	n e w es t </w>

Vocabulary

d	e	i	l	n	o	s	t	w
es	est							

Frequency

d-es (3)	l-o (7)	w-</w> (5)
e-r (2)	n-e (5)	w-es (5)
e-w (5)	o-w (7)	w-e (2)
es-t (8)	r-</w> (2)	w-i (3)
i-d (3)	t-</w> (8)	

Source: https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/

BPE in action

Corpus

low	lower	newest
low	lower	newest
low	widest	newest
low	widest	newest
low	widest	newest

Corpus

low</w>	lower</w>	newest</w>
low</w>	lower</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>

Corpus

l o w </w>	l o w e r </w>	n e w est </w>
l o w </w>	l o w e r </w>	n e w est </w>
l o w </w>	w i d est </w>	n e w est </w>
l o w </w>	w i d est </w>	n e w est </w>
l o w </w>	w i d est </w>	n e w est </w>

Vocabulary

d	e	i	l	n	o	s	t	w
es	est	est</w>	lo	low	low</w>	ne	new	newest</w>

After 10 merges

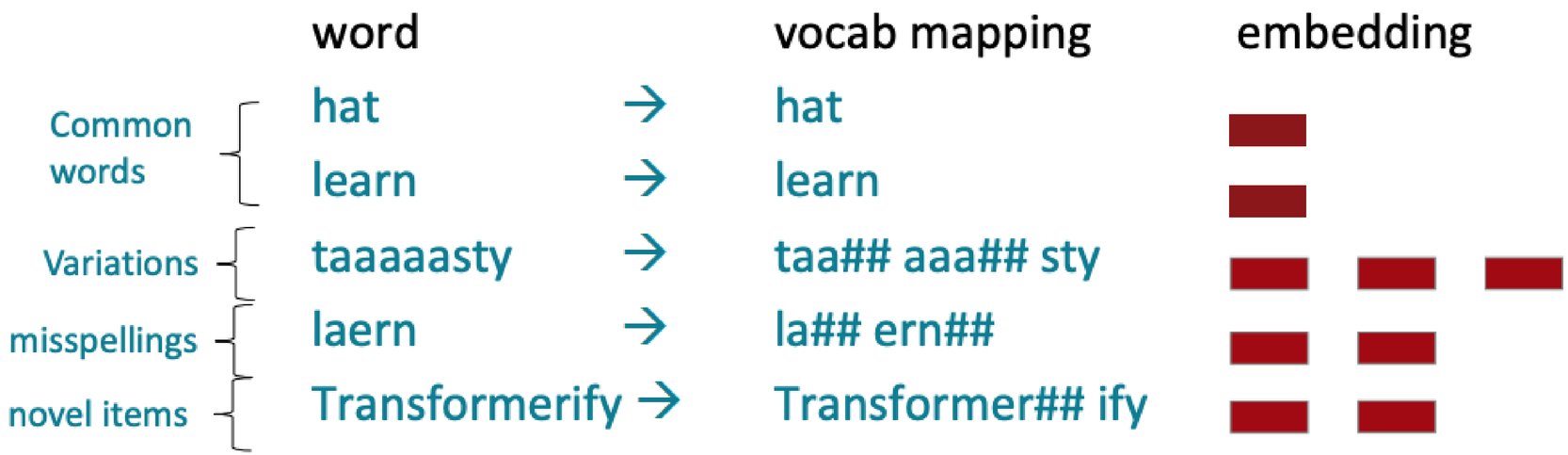
Source: https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/

Byte-pair encoding

- At test time, first split words into sequences of characters, then apply the learned operations to merge the characters into larger, known symbols
- Originally used in NLP for machine translation; now a similar method (WordPiece) is used in pretrained models.

Word structure and subword models

- Common words end up being a part of the subword vocabulary, while rarer words are split into (sometimes intuitive, sometimes not) components.
- In the worst case, words are split into as many subwords as they have characters.



Natural Language Generation

Natural Language Generation

- Natural language understanding and natural language generation are two sides of the same coin
  - In order to generate good language, you need to understand language
  - If you understand language, you should be able to generate it (with some effort)
- NLG is the workhorse of many classic and novel applications
  - AI Assistants
  - Translators
  - Search summarizers

NLG Use Cases

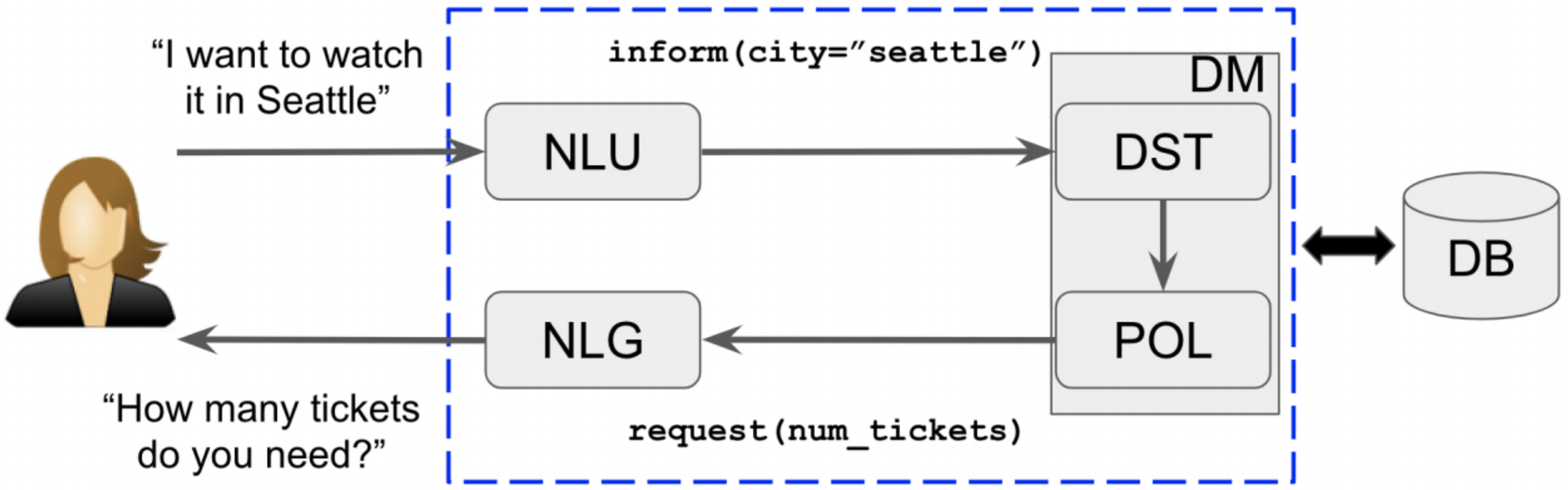
Simple and Effective Multi-Paragraph Reading Comprehension

Christopher Clark, Matt Gardner · Computer Science · ACL · 29 October 2017

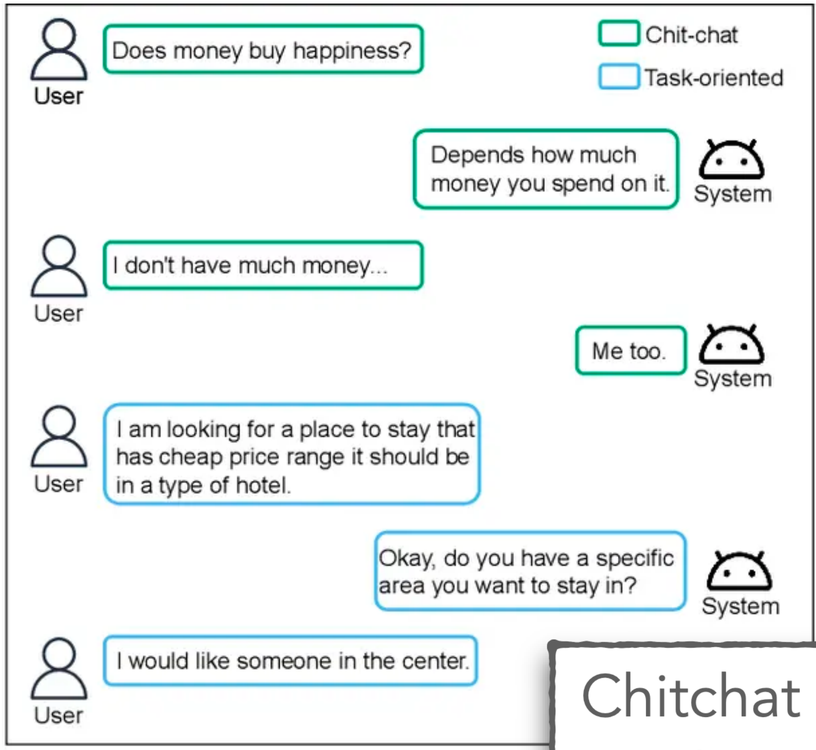
TLDR We propose a state-of-the-art pipelined method for training neural paragraph-level question answering models on document QA data. Expand

236 PDF · View PDF on arXiv Save Alert Cite Research Feed

Summarization



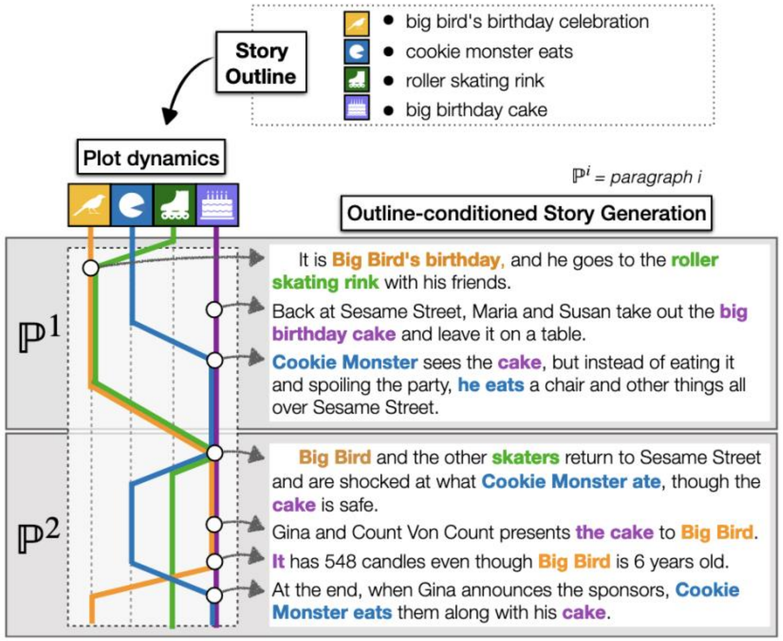
Task-driven Dialog



Chitchat Dialog

More Interesting NLG Uses

Creative stories



Rashkin et al., 2020

Data-to-text

Table Title: Robert Craig (American football)
Section Title: National Football League statistics
Table Description:None

		RUSHING					RECEIVING				
YEAR	TEAM	ATT	YDS	AVG	LNG	TD	NO.	YDS	AVG	LNG	TD
1983	SF	176	725	4.1	71	8	48	427	8.9	23	4
1984	SF	155	649	4.2	28	4	71	675	9.5	64	3
1985	SF	214	1050	4.9	62	9	92	1016	11	73	6
1986	SF	204	830	4.1	25	7	81	624	7.7	48	0
1987	SF	215	815	3.8	25	3	66	492	7.5	35	1
1988	SF	310	1502	4.8	46	9	76	534	7.0	22	1
1989	SF	271	1054	3.9	27	6	49	473	9.7	44	1
1990	SF	141	439	3.1	26	1	25	201	8.0	31	0
1991	RAI	162	590	3.6	15	1	17	136	8.0	20	0
1992	MIN	105	416	4.0	21	4	22	164	7.5	22	0
1993	MIN	38	119	3.1	11	1	19	169	8.9	31	1
Totals	-	1991	8189	4.1	71	56	566	4911	8.7	73	17

Craig finished his eleven NFL seasons with 8,189 rushing yards and 566 receptions for 4,911 receiving yards.

Parikh et al., 2020

Visual description

Two children are sitting at a table in a restaurant. The children are one little girl and one little boy. The little girl is eating a pink frosted donut with white icing lines on top of it. The girl has blonde hair and is wearing a green jacket with a black long sleeve shirt underneath. The little boy is wearing a black zip up jacket and is holding his finger to his lip but is not eating. A metal napkin dispenser is in between them at the table. The wall next to them is white brick. Two adults are on the other side of the short white brick wall. The room has white circular lights on the ceiling and a large window in the front of the restaurant. It is daylight outside.

Krause et al., 2017

Broad Spectrum of NLG Tasks

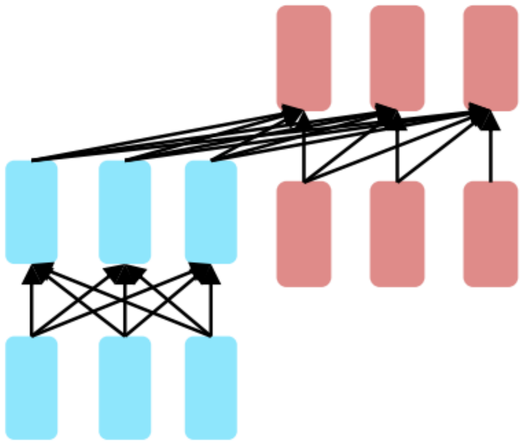
Open-ended generation: the output distribution still has high freedom.

Non-open-ended generation: the input mostly determines the output generation.

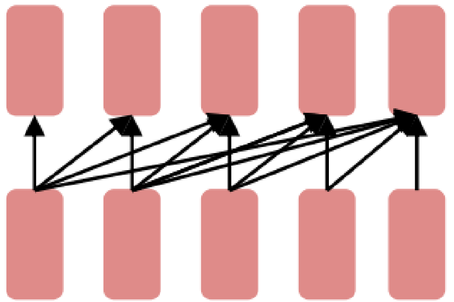
Broad Spectrum of NLG Tasks

Less Open-Ended — More Open-Ended

- Machine Translation
- Summarization
- Task-driven Dialog
- Chitchat Dialog
- Story Generation



Encoder-Decoders

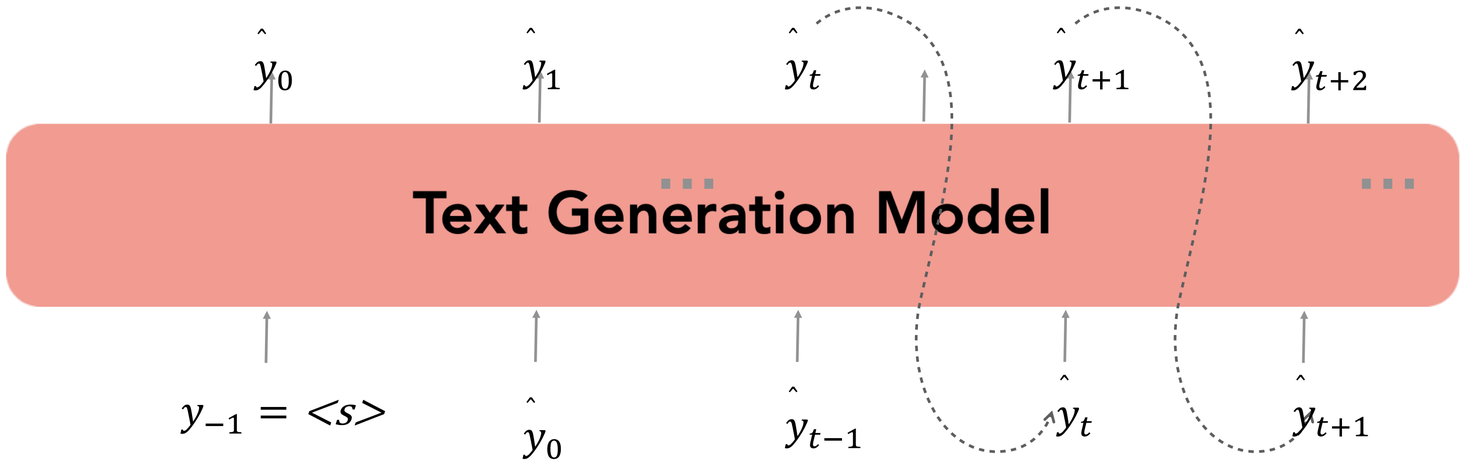


Decoders

Language Generation: Fundamentals

- In autoregressive text generation models, at each time step t, our model takes in a sequence of tokens as input $S = f_\theta(y_{<t}) \in \mathbb{R}^V$ and outputs a new token, $\hat{y}_t$
- For model $f_\theta(\cdot)$ and vocabulary V, we get scores $S = f_\theta(y_{<t}) \in \mathbb{R}^V$

$$P(w|y_{<t}) = \frac{\exp(S_w)}{\sum_{v \in V} \exp(S_v)}$$

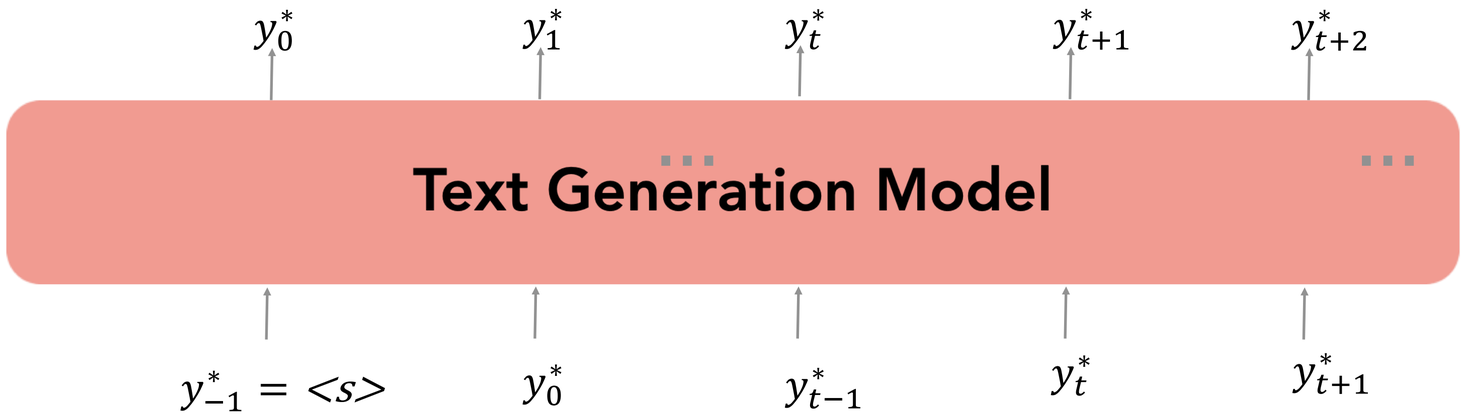


Language Generation: Training

- Trained one token at a time to maximize the probability of the next token y_t^* given preceding words $y_{<t}^*$

$$\mathcal{L} = -\sum_{t=1}^{T} \log P(y_t|y_{<t}) = -\sum_{t=1}^{T} \log \frac{\exp(S_{y_t|y_{<t}})}{\sum_{v \in V} \exp(S_{v|y_{<t}})}$$

- Classification task at each time step trying to predict the actual word y_t^* in the training data
- "Teacher forcing" (reset at each time step to the ground truth)



Teacher Forcing

- Strategy for training decoders / language models
- At each time step t in decoding we force the system to use the gold target token from training as the next input x_{t+1}, rather than allowing it to rely on the (possibly erroneous) decoder output $\hat{y}_t$
- Runs the risk of exposure bias!
  - During training, our model's inputs are gold context tokens from real, human-generated texts
  - At generation time, our model's inputs are previously–decoded tokens

Language Generation: Inference

- At inference time, our decoding algorithm defines a function to select a token from this distribution:

Inference / Decoding Algorithm

$$\hat{y}_t = g(P(y_t|y_{<t}))$$

- The "obvious" decoding algorithm is to greedily choose the highest probability next token according to the model at each time step

$$g = \text{arg}max$$

$$\hat{y}_t = \text{arg}\max_{w \in V}(P(y_t = w|y_{<t}))$$

Paper Presentation Schedule (starting Feb 18)

Name of Paper	Name of Pr	Discussion	Discussion	Day	Date
Collaborative gym: A framework for enabling and evaluating human-agent collaboration	Sheryl Mathev	Hong-En Chen	Anthony Liang	1	2/18/26
Sys2Bench: Inference-Time Computations for LLM Reasoning and Planning (arXiv:2502.12	I-Chun Liu	Zeyu Shanggua	Xinlei Yu	1	2/18/26
VisuLogic: A Benchmark for Evaluating Visual Reasoning in Multi-modal Large Language M	Zongjian Li	Ziyan Yang	Muzi Tao	2	2/25/26
Search-o1: Agentic Search-Enhanced Large Reasoning Models	Dimitrios Andr	Qihan Zhang	Efthymios Tsap	2	2/25/26
Humanity's Last Exam	Toan Nguyen	Didem Zeynep	Yifan Wu	2	2/25/26
rStar-Math: Small LLMs Can Master Math Reasoning with Self-Evolved Deep Thinking	Mohammad H	Zihan Wang	Didem Zeynep	3	3/4/26
MemoryAgentBench: Evaluating Memory in LLM Agents	Ziyan Yang	Zongjian Li	Jike Zhong	3	3/4/26
CoT-Self-Instruct: Building high-quality synthetic prompts for reasoning and non-reasoning t	Gengpei Qi	Yipeng Gao	Mohammad Ho	3	3/4/26
Constitutional Classifiers: Defending against Universal Jailbreaks across Thousands of Hou	Michael Duan	Sheryl Mathew	Anu Soneye	3	3/4/26
Can MLLMs Reason in Multimodality? EMMA: An Enhanced MultiModal ReAsoning Benchn	Efthymios Tsa	Muzi Tao	Chen Chu	3	3/4/26
Stop Overthinking: A Survey on Efficient Reasoning for Large Language Models	Stop Overthinl	Ania Serbina	Zhaoyuan Den	3	3/4/26
ReTool: Reinforcement Learning for Strategic Tool Use in LLMs	Qihan Zhang	I-Chun Liu	Sunwoo Lim	4	3/25/26
ToolRL: Reward is All Tool Learning Needs	Xinlei Yu	Haolin Xiong	Zhangyu Jin	4	3/25/26
TTRL: Test-Time Reinforcement Learning	Alexios Rusto	Sunwoo Lim	Letao Chen	4	3/25/26
It's Not That Simple: An Analysis of Simple Test-Time Scaling (arXiv:2507.14419)	Jike Zhong	Toan Nguyen	Zhaoyuan Den	4	3/25/26
The Illusion of Thinking: Understanding the Strengths and Limitations of Reasoning Models	Yifan Wu	DJ Bell	Chen Chu	4	3/25/26
Red-Teaming LLM Multi-Agent Systems via Communication Attacks	Qixin Hu	Michael Duan	Jike Zhong	4	3/25/26
Training Language Models to Reason Efficiently	Haolin Xiong	Anu Soneye	Zihan Wang	5	4/1/26
Layer by Layer: Uncovering Hidden Representations in Language Models	Ryan Swift	Hong-En Chen	Yanchen Liu	5	4/1/26
Reasoning Models Don't Always Say What They Think	Letao Chen	Efthymios Tsap	DJ Bell	5	4/1/26
Training Large Language Model to Reason in a Continuous Latent Space	Zhaoyuan Der	Bo-Ruei Huang	Ryan Swift	5	4/1/26
Understanding R1-Zero-Like Training: A Critical Perspective	Sunwoo Lim	Dimitrios Andre	Alexios Ruston	5	4/1/26
LIMO: Less is More for Reasoning	Yanchen Liu	Sheryl Mathew	Ania Serbina	5	4/1/26
Reinforcement Learning Teachers of Test Time Scaling (RLT) (arXiv:2506.08388)	Bo-Ruei Huan	Anthony Liang	Zeyu Shanggua	6	4/8/26
ROBOTOUILLE: Asynchronous Planning Benchmark for LLM Agents (ICLR 2025)	Didem Zeyne	I-Chun Liu	Ryan Swift	6	4/8/26
PaperArena: Benchmarking Tool-Using Agents	Yipeng Gao	Yifan Wu	Ziyan Yang	6	4/8/26

Paper Presentation Requirement

20% of the grade

Each student will complete **one in-depth research paper presentation (15-17 mins presentation + 5-7 mins discussion)** during the semester using slides,

Each student will also serve as a **designated discussion lead for *two* additional paper presentations.**

Paper Presentation: Evaluation

The presentation will be assessed on the student's ability to clearly explain (50%):

- the paper's motivation
- technical approach
- key results

To critically evaluate its assumptions, limitations, and contributions (30%)

To situate the work within the broader LLM and NLP research landscape (20%)