

# Lecture 11 – Pretraining & Transformer Variants



Naveh Porat

July 2022

# Table of Contents

**01**

Recap

**02**

Pretraining

**03**

Encoders

**04**

Decoders

**05**

Encoder & Decoder

**06**

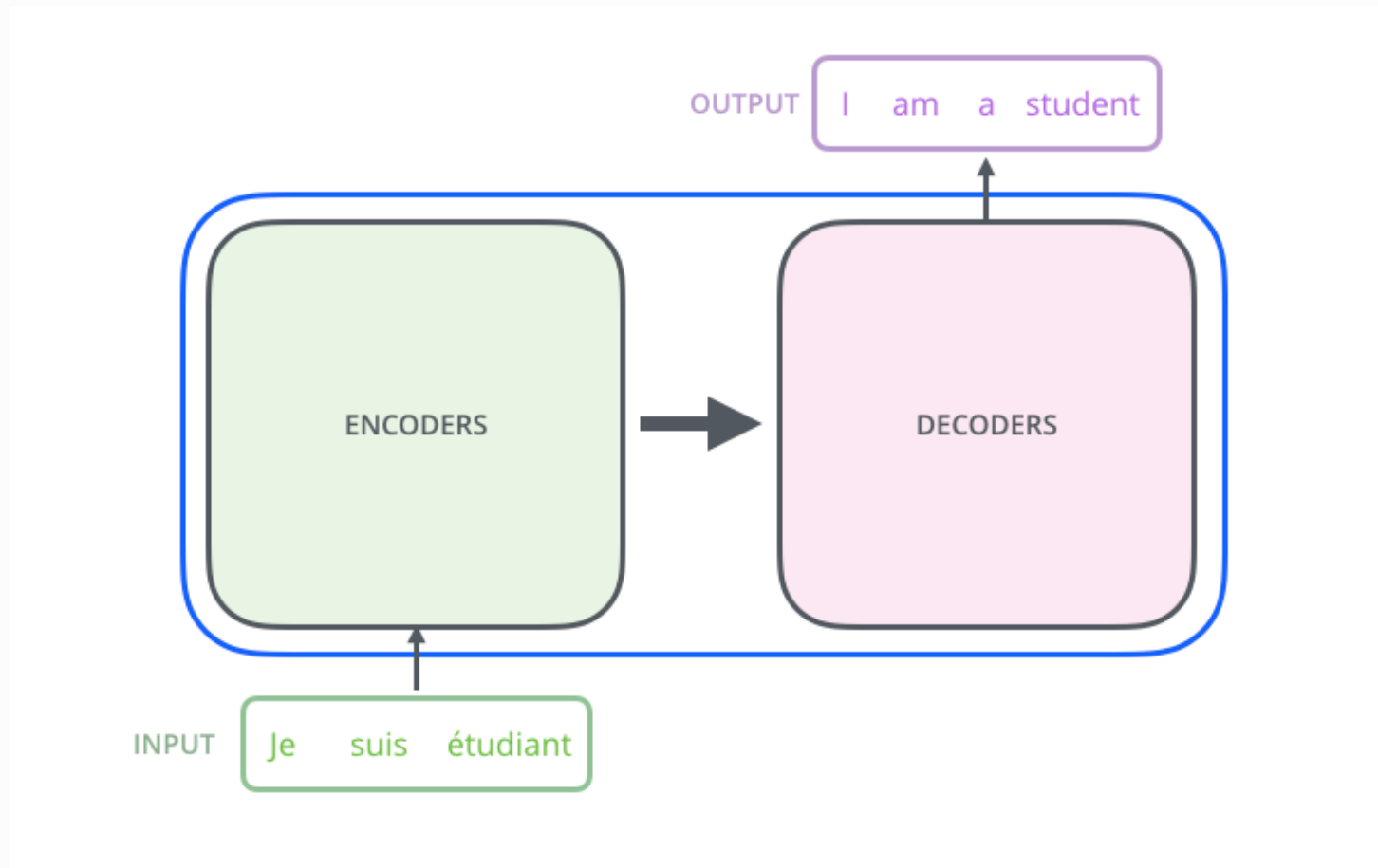
Other Variants

A dark blue background with the word "Recap" in white. Two vertical orange bars are positioned on the left and right sides of the slide.

# Recap

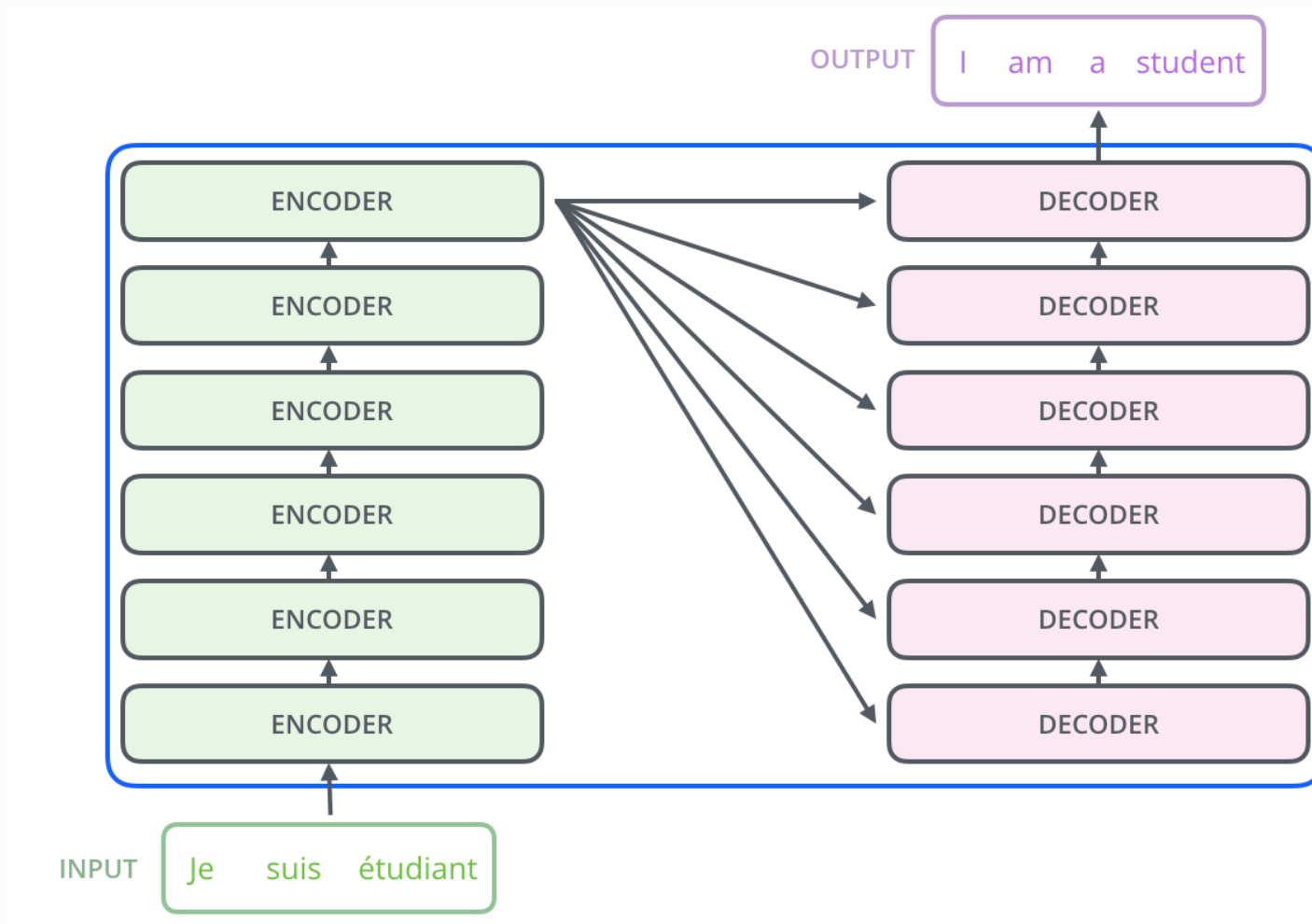
# Transformer Recap

The transformer is composed of two components: the encoder and the decoder.



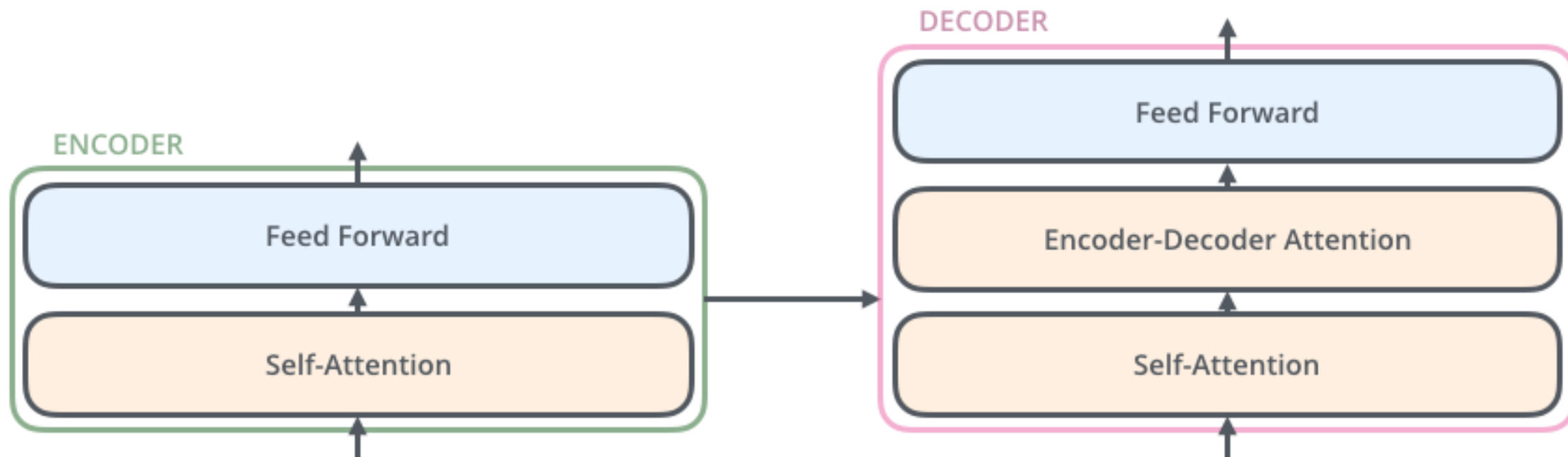
# A High-Level Look

Each of these components is split up into a series of layer with identical architectures.



# A High-Level Look

Each encoder\decoder layer has two components: attention + FF network

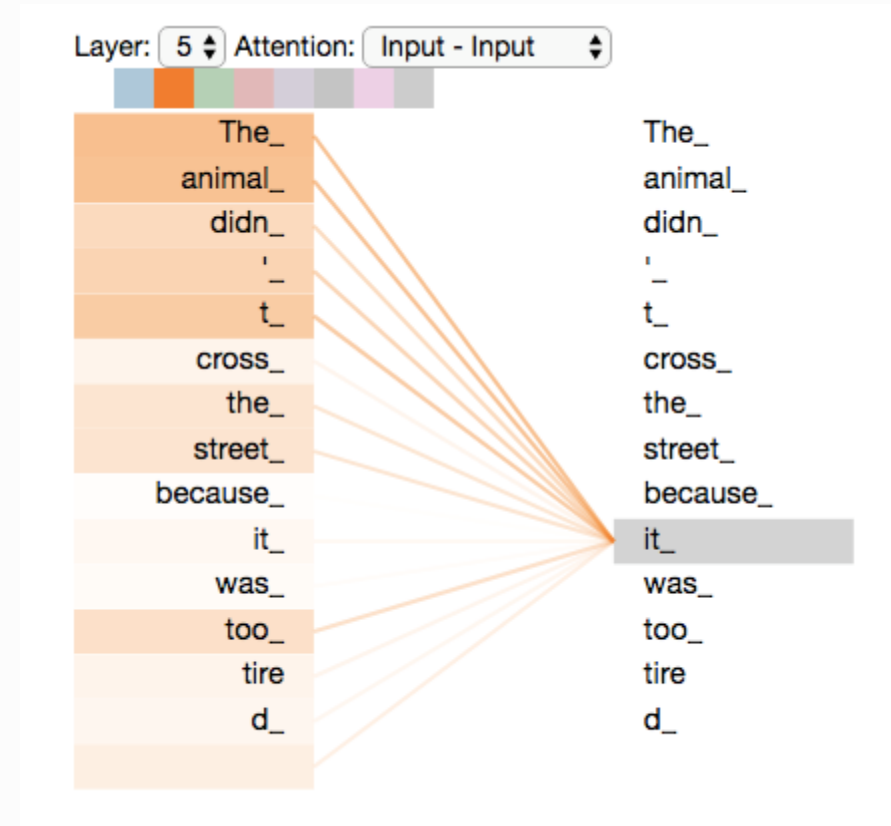


# Self-Attention at a High Level

Attention is a mechanism that allows each token in a sequence to attend to information from other tokens.

In the case of self attention, we saw that each token in a sequence can process information from any other token in the same sequence.

In cross-attention (as in the decoder) we saw that each token can process information from the context, e.g. the prompt embedded by the encoder.



# BERT

1 - **Semi-supervised** training on large amounts of text (books, wikipedia..etc).

The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.

## Semi-supervised Learning Step

**Model:**



**Dataset:**



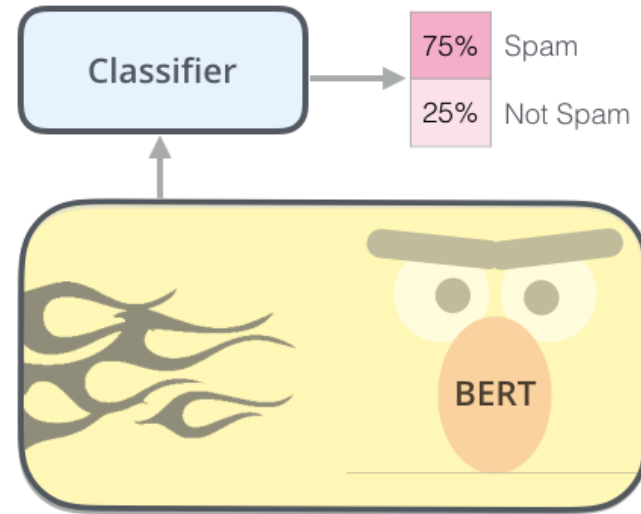
**Objective:**

Predict the masked word  
(language modeling)

2 - **Supervised** training on a specific task with a labeled dataset.

## Supervised Learning Step

**Model:**  
(pre-trained  
in step #1)

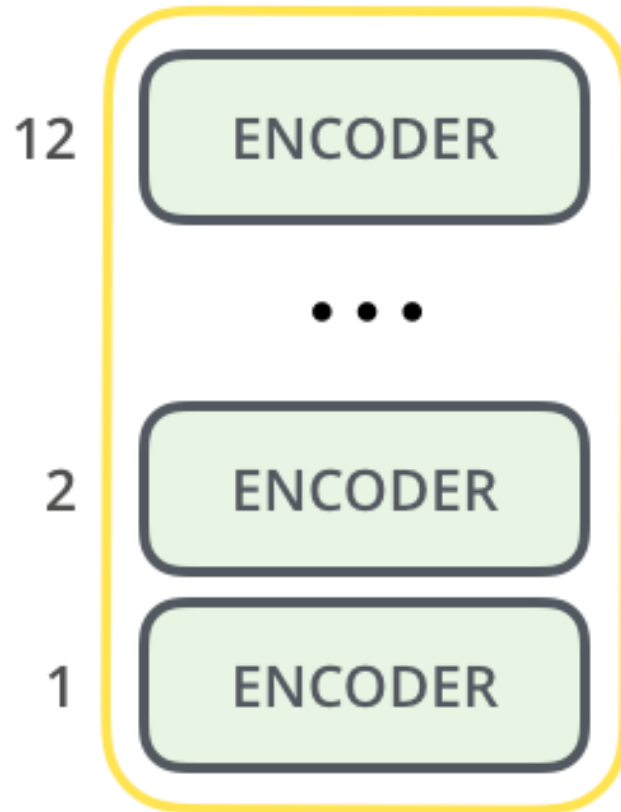


**Dataset:**

Email message	Class
Buy these pills	Spam
Win cash prizes	Spam
Dear Mr. Atreides, please find attached...	Not Spam

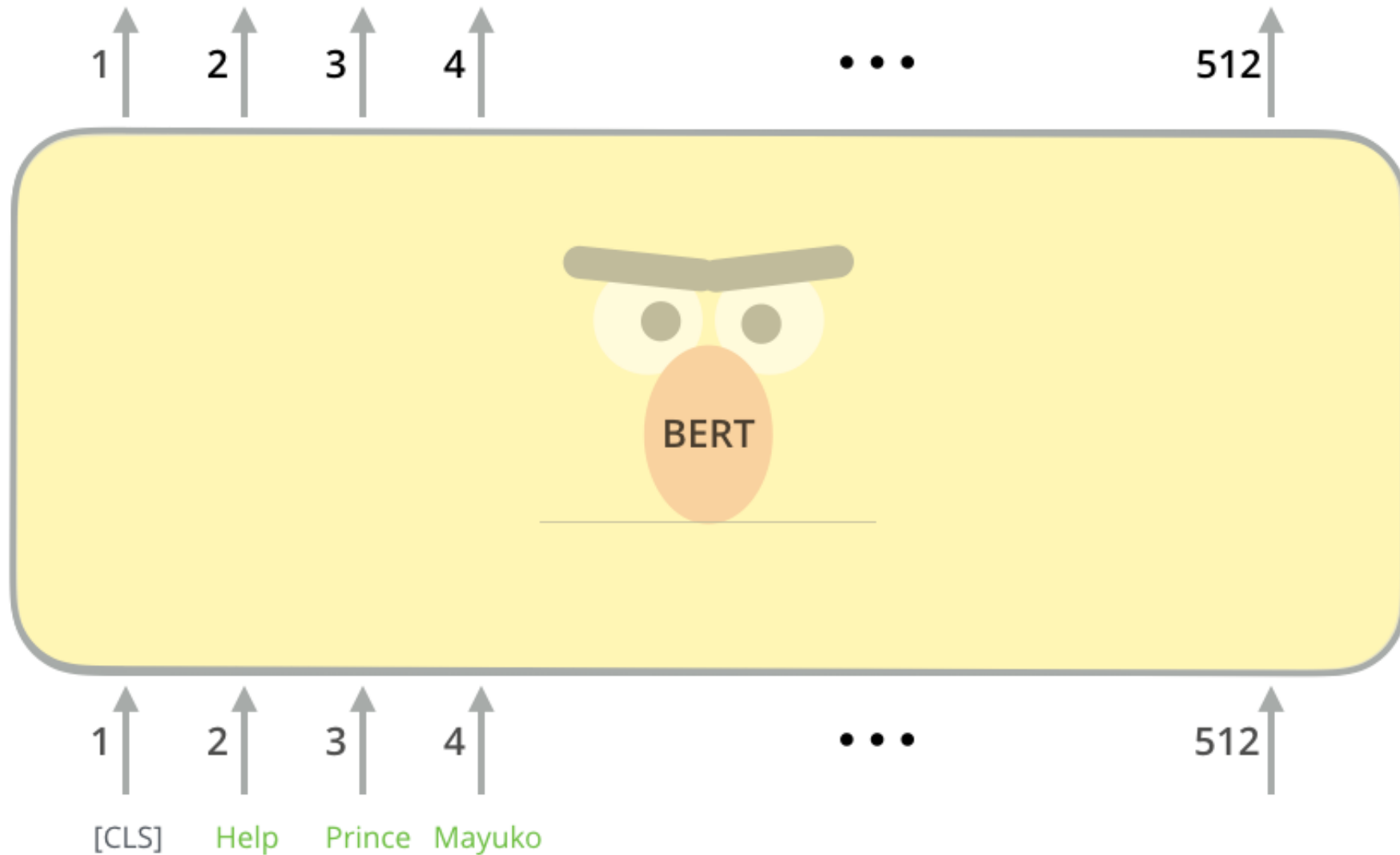


# BERT Architecture



BERT<sub>BASE</sub>

# Model Inputs



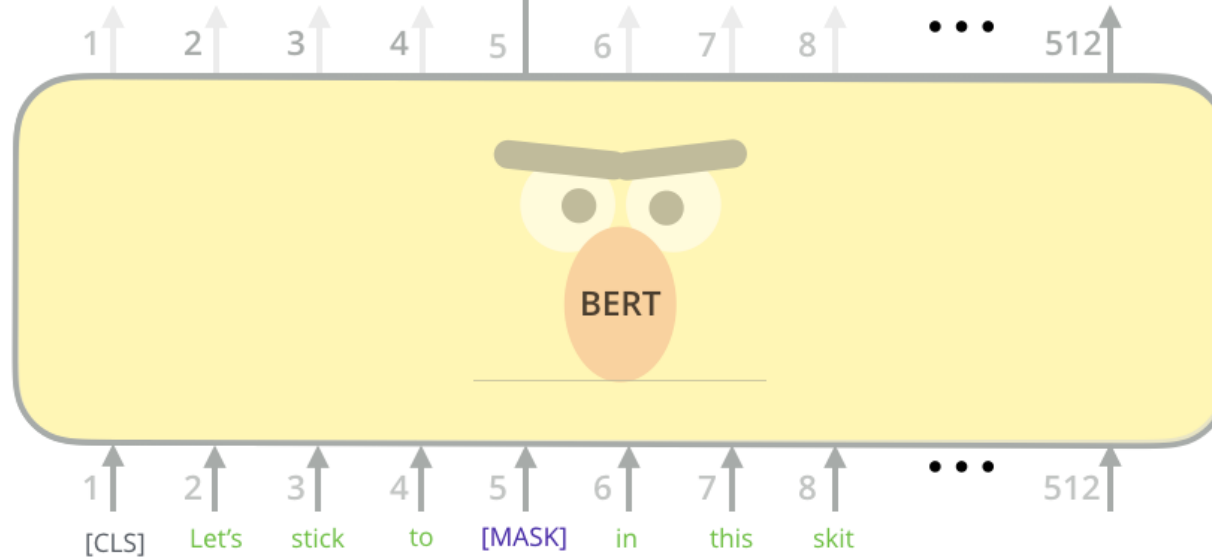
# BERT Training – Masked Language Modeling

Use the output of the masked word's position to predict the masked word

Possible classes:  
All English words

0.1%	Aardvark
...	...
10%	Improvisation
...	...
0%	Zyzyva

FFNN + Softmax



Randomly mask  
15% of tokens

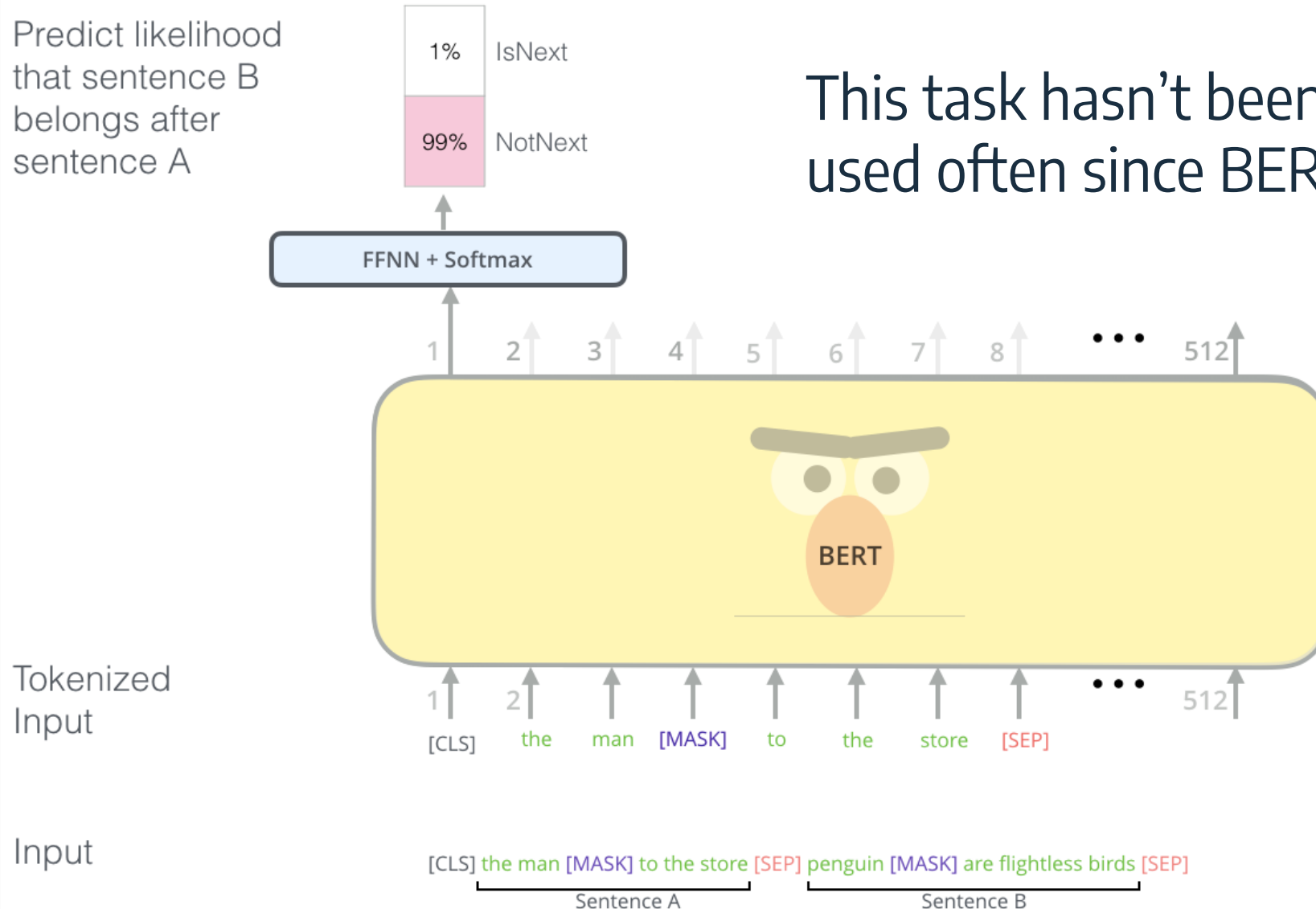
Input

[CLS] Let's stick to improvisation in this skit

# BERT Training – Next Sentence Prediction

Predict likelihood  
that sentence B  
belongs after  
sentence A

This task hasn't been  
used often since BERT



# Short Checklist

- Can model sequences of varying length ✓
- Provides contextual embedding ✓
- Trained on large amounts of text ✓
- Models long-term dependencies ✓
- Computes embeddings efficiently (concurrency) ✓
- Takes the sequence order into account ✓
- Processes text in a bidirectional manner ✓



# Short Checklist

Any Problems?  
More on that today



# Pretraining

What can we hope to  
learn from pretraining?

# Trivia

Stanford University is located in [MASK], California.



# Syntax

I put [MASK] fork down on the table.

# Coreference

The woman walked across the street, checking  
for traffic over [MASK] shoulder.

# Lexical Semantics/Topic

I went to the ocean to see the fish,  
turtles, seals, and [MASK].

# Sentiment

Overall, the value I got from the two hours watching it was the sum total of the popcorn and the drink. The movie was [MASK].

# Some Reasoning (this is harder)

Iroh went into the kitchen to make some tea.  
Standing next to Iroh, Zuko pondered his  
destiny. Zuko left the [MASK].

# Some Basic Arithmetic

I was thinking about the sequence that goes  
1, 1, 2, 3, 5, 8, 13, 21, [MASK].

# Biases

The doctor examined the patient's skin. Then  
[MASK] said: "everything looks just fine".








# Subword Modeling



# Word structure and subword models

Let's take a look at the assumptions we've made about a language's vocabulary.






We assume a fixed vocab of hundreds of thousands of words, built from the training set. All novel words seen at test time are mapped to a single UNK (OOV).

	<u>word</u>		<u>vocab mapping</u>	<u>embedding</u>
Common words	hat	→	pizza (index)	
	learn	→	tasty (index)	
Variations	taaaaasty	→	UNK (index)	
misspellings	laern	→	UNK (index)	
novel items	Transformerify	→	UNK (index)	

# Word structure and subword models

We use some greedy algorithm to find a good set of subwords in our corpus. Common words end up being a part of the subword vocabulary, while rarer words are split into (sometimes intuitive, sometimes not) components.

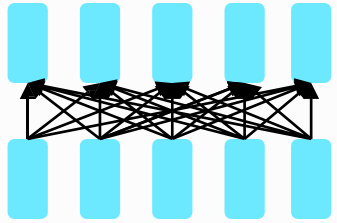
In the worst case, words are split into as many subwords as they have characters.

	word		vocab mapping	embedding
Common words	hat	→	hat	
	learn	→	learn	
Variations	taaaaasty	→	taa## aaa## sty	
misspellings	laern	→	la## ern##	
novel items	Transformerify	→	Transformer## ify	

# Models

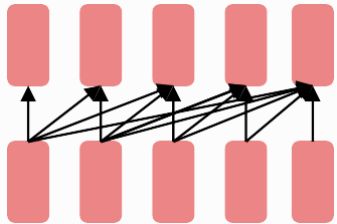
# Pre-training for three types of architectures

The neural architecture influences the type of pretraining, and natural use cases.



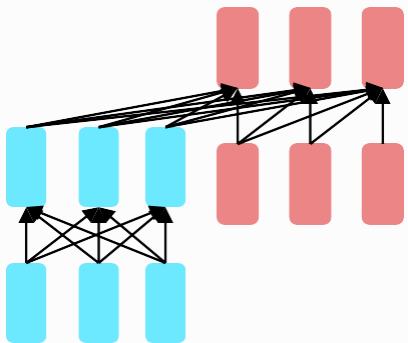
**Encoders**

- Gets bidirectional context – can condition on future!
- Question is, how do we pre-train them?



**Decoders**

- Language models! Impose a probability distribution on the language.
- Nice to generate from; can't condition on future words

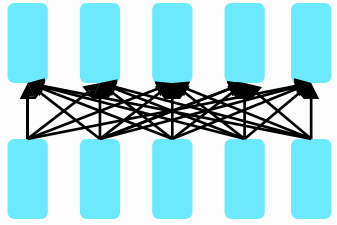


**Encoder-  
Decoders**

- Good parts of decoders and encoders?
- What's the best way to pretrain them?

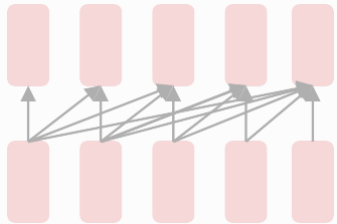
# Pre-training for three types of architectures

The neural architecture influences the type of pretraining, and natural use cases.



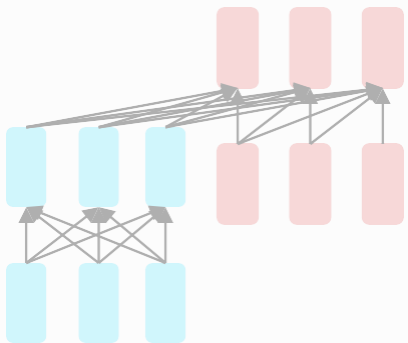
**Encoders**

- Gets bidirectional context – can condition on future!
- Question is, how do we pre-train them?



**Decoders**

- Language models! Impose a probability distribution on the language.
- Nice to generate from; can't condition on future words



**Encoder-  
Decoders**

- Good parts of decoders and encoders?
- What's the best way to pretrain them?

# Encoders

# BERT

**Architecture:** Encoder

---

**Task:** MLM + NSP

---

**Notes:** Eldest son in the  
Transformers family!

---

---



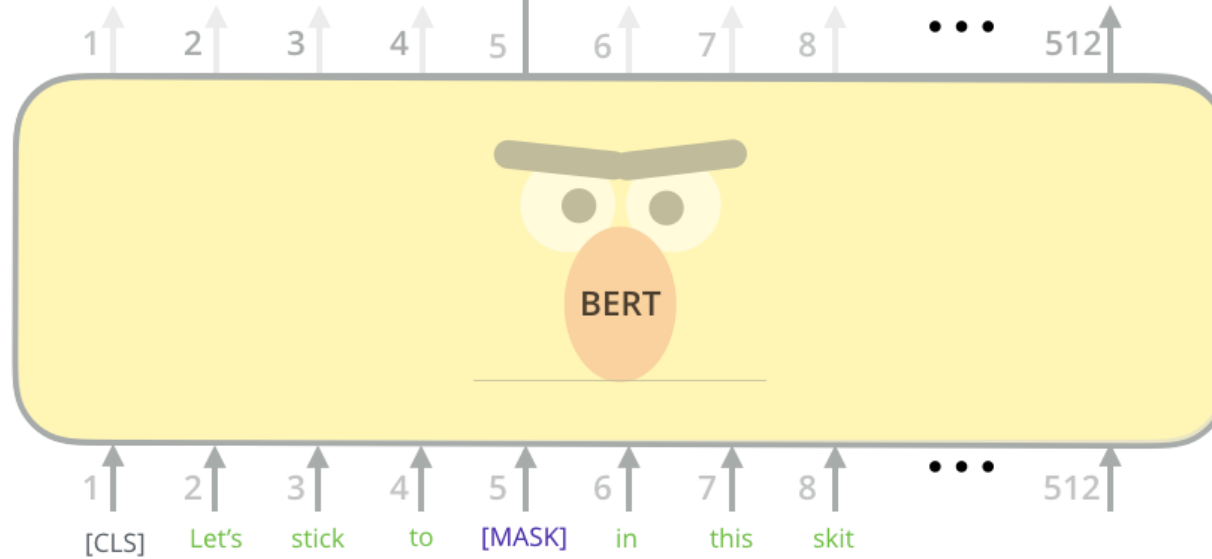
# BERT Training – Masked Language Modeling

Use the output of the masked word's position to predict the masked word

Possible classes:  
All English words

0.1%	Aardvark
...	...
10%	Improvisation
...	...
0%	Zyzyva

FFNN + Softmax



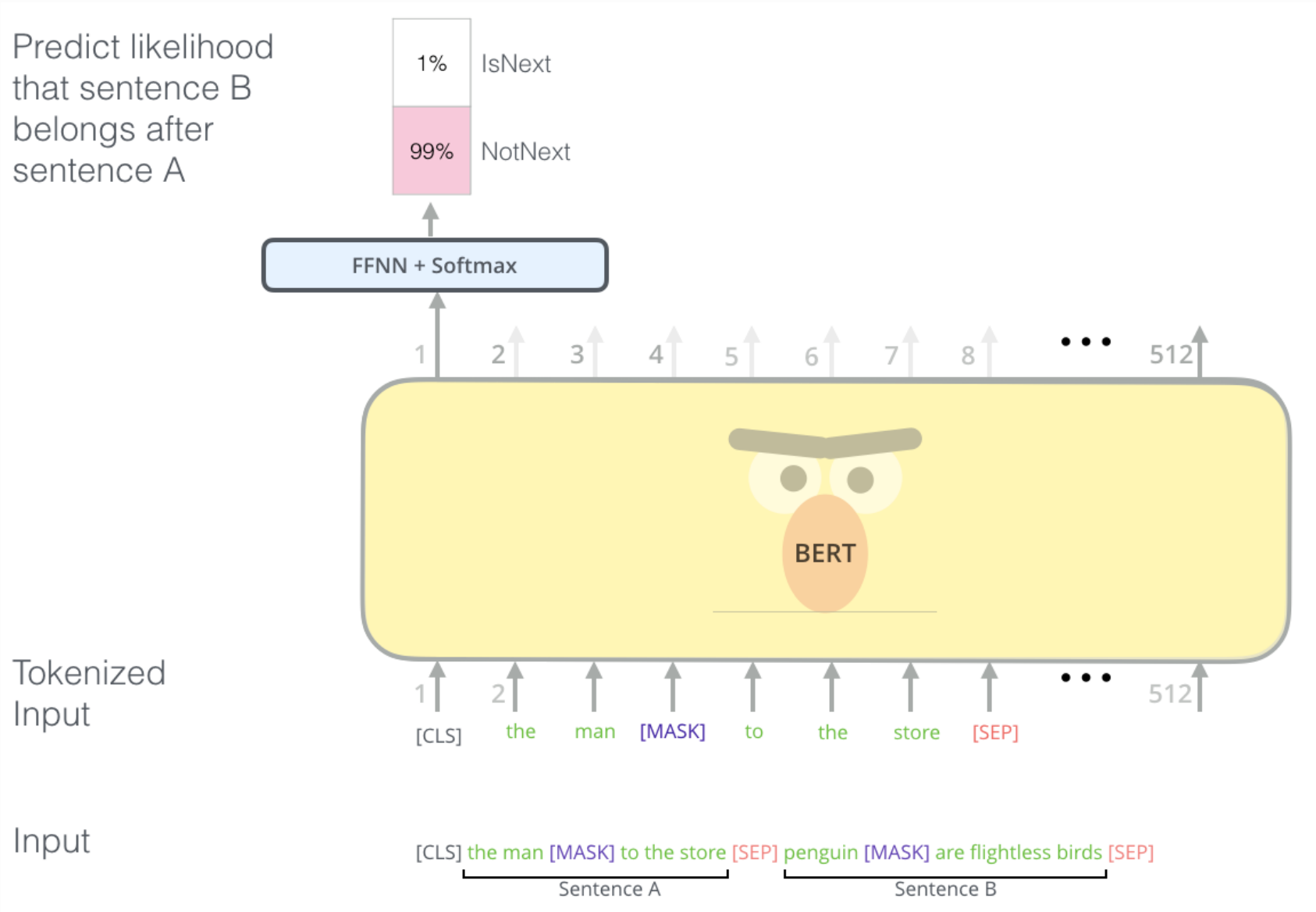
Randomly mask  
15% of tokens

Input

[CLS] Let's stick to improvisation in this skit



# BERT Training – Next Sentence Prediction



# RoBERTa

**Architecture:** Encoder

---

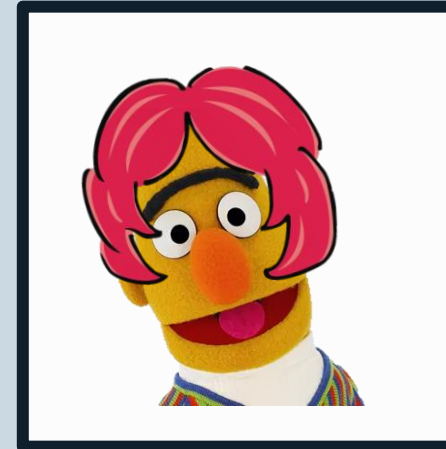
**Task:** MLM

---

**Notes:** Better optimized than  
BERT

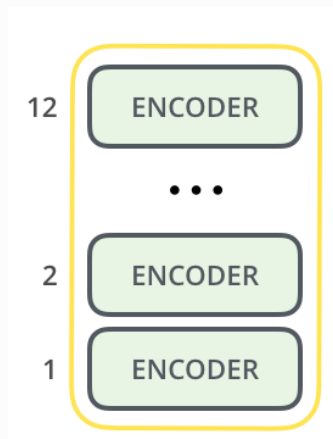
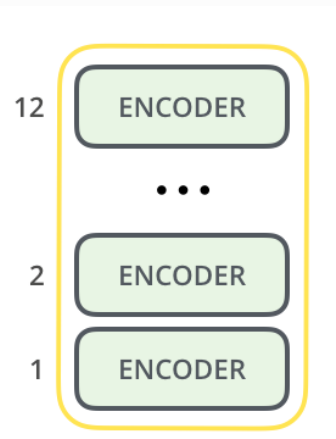
---

---



# Architecture

Same architecture!



# Static vs. Dynamic Masking



**Epoch 1:** Let's stick to [MASK] in the skit.

**Epoch 2:** Let's stick to [MASK] in the skit.

**Epoch 3:** Let's stick to [MASK] in the skit.

# Static vs. Dynamic Masking



**Epoch 1:** Let's stick to [MASK] in the skit.

**Epoch 2:** Let's stick [MASK] improvisation in the skit.

**Epoch 3:** Let's stick to improvisation in the [MASK].

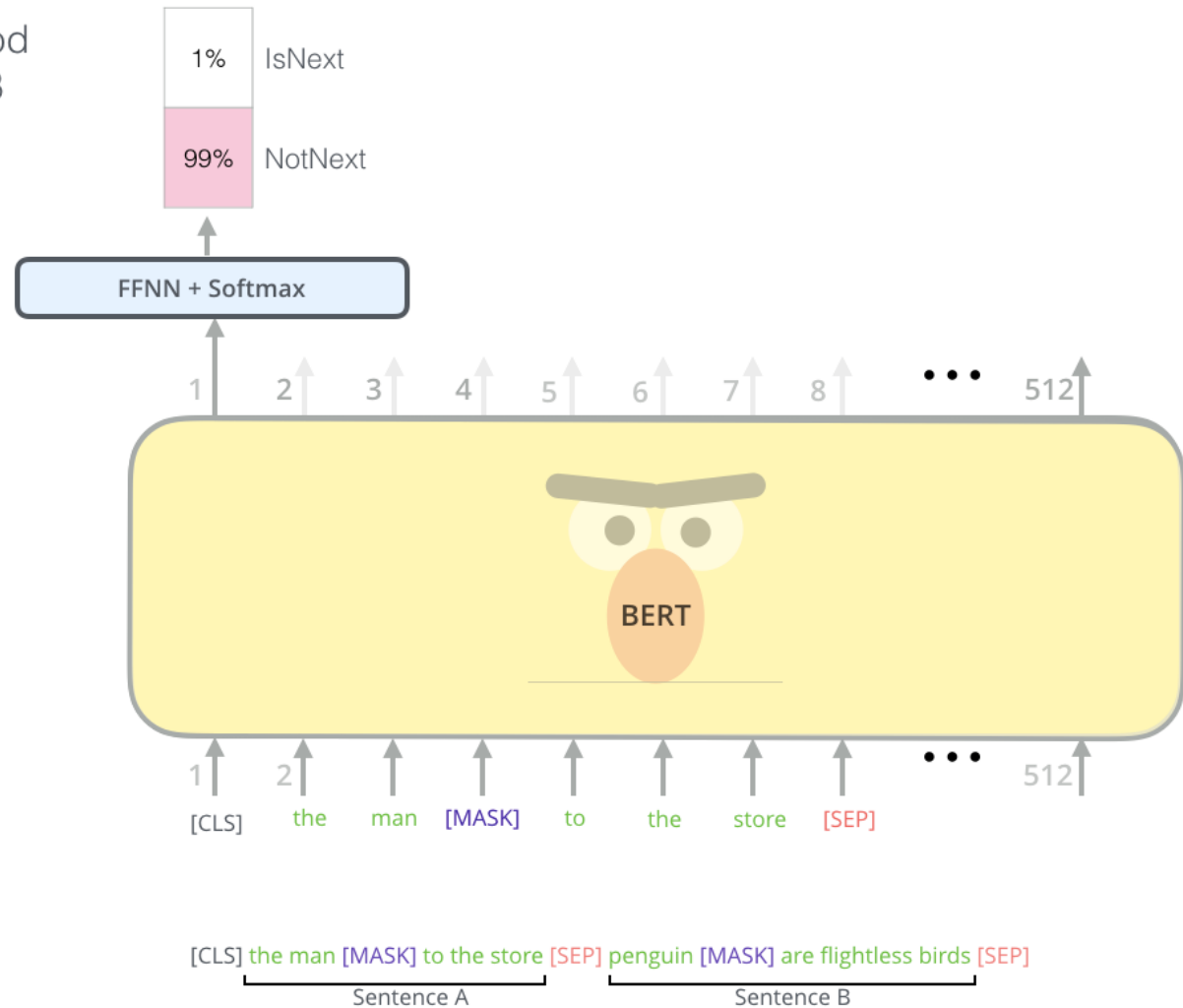
# No NSP



Predict likelihood  
that sentence B  
belongs after  
sentence A

Tokenized  
Input

Input



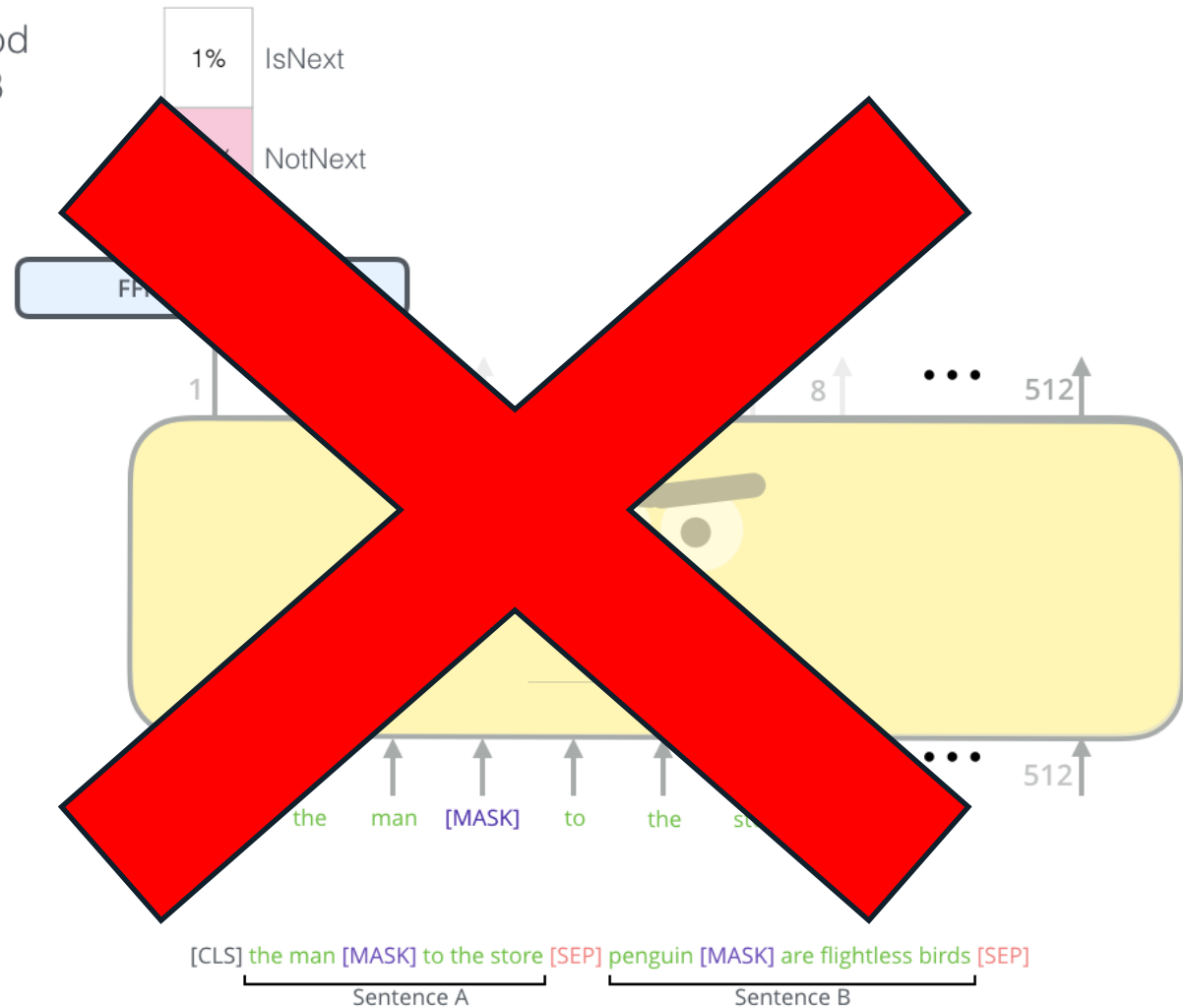
# No NSP



Predict likelihood  
that sentence B  
belongs after  
sentence A

Tokenized  
Input

Input



# Data & Optimization

Data

Batch Size

Pretraining Length



<



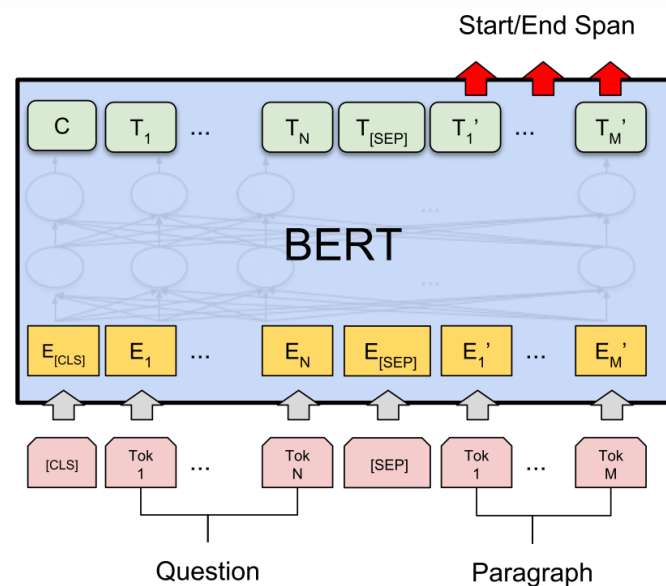


# Results

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	<b>94.6/89.4</b>	<b>90.2</b>	<b>96.4</b>
BERT <sub>LARGE</sub>						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
XLNet <sub>LARGE</sub>						
with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4
+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6

# Spans in NLP

- Many tasks in NLP require aggregating information of a **span** of text and not just individual tokens.
- For example, the most common form of the question answering task is to detect the span of the answer inside a text.



# SpanBERT

**Architecture:** Encoder

---

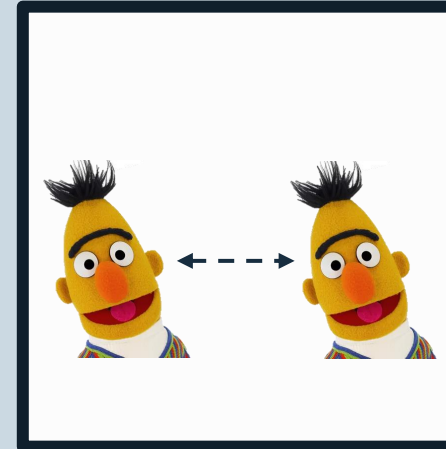
**Task:** MLM + SBO

---

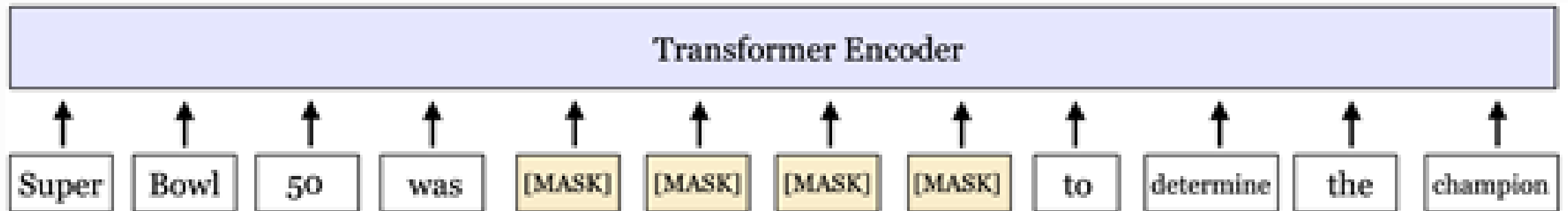
**Notes:** Trained on predicting  
masked **spans** of text.

---

---

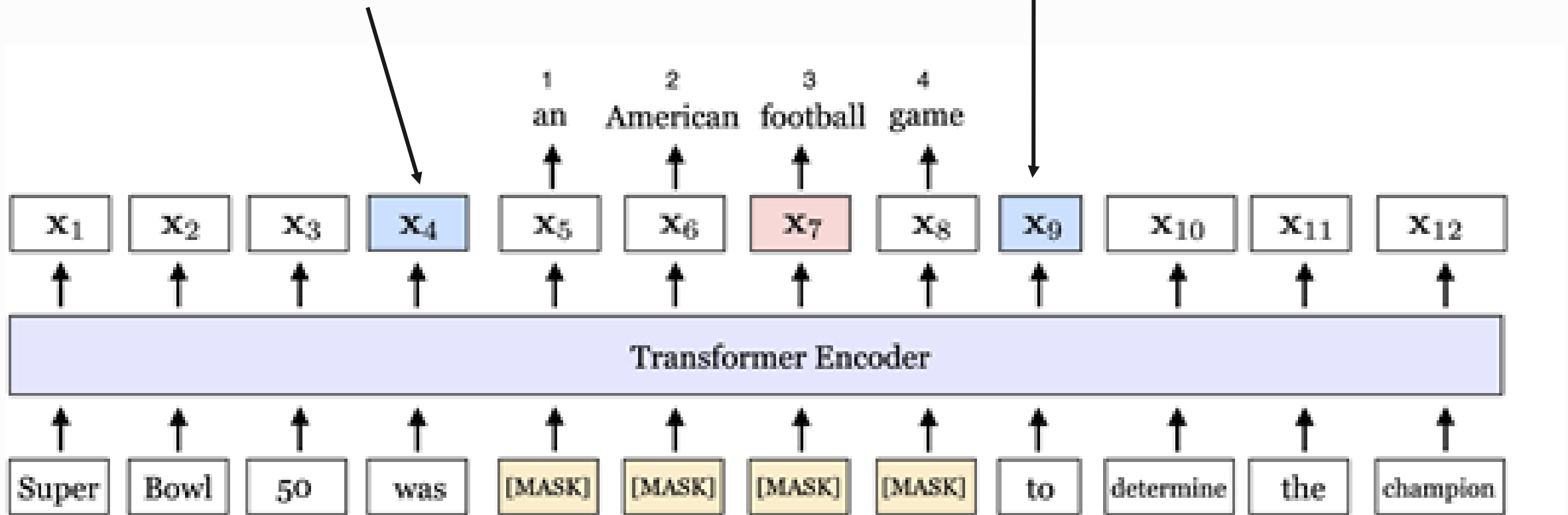


# Static vs. Spanned Masking



# Span Boundary Objective (SBO)

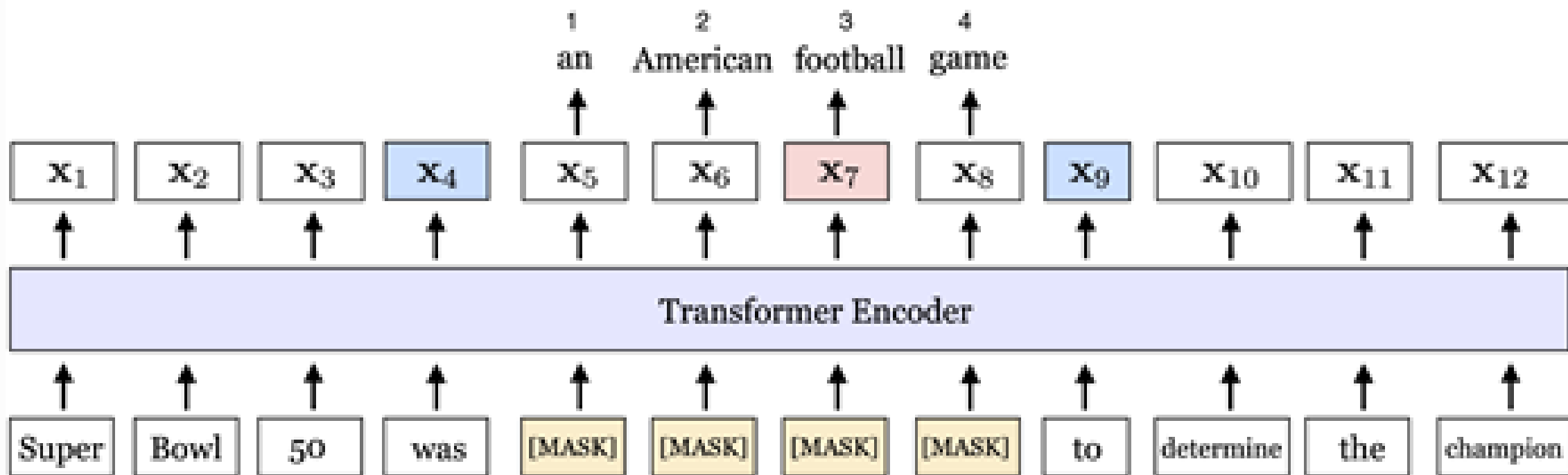
We would want the tokens at the boundary of the span to contain as much information about the span as possible



# Span Boundary Objective (SBO)

$$\mathcal{L}(\text{football}) = \mathcal{L}_{\text{MLM}}(\text{football}) + \mathcal{L}_{\text{SBO}}(\text{football})$$

$$= -\log P(\text{football} \mid \mathbf{x}_7) - \log P(\text{football} \mid \mathbf{x}_4, \mathbf{x}_9, \mathbf{p}_3)$$



# Results

	NewsQA	TriviaQA	SearchQA	HotpotQA	Natural Questions	Avg.
Google BERT	68.8	77.5	81.7	78.3	79.9	77.3
SpanBERT	<b>73.6</b>	<b>83.6</b>	<b>84.8</b>	<b>83.0</b>	<b>82.5</b>	<b>81.5</b>

# Structure in NLP

- Bert training does not explicitly incorporate the structure of text inside a model, aside from positional embeddings.
- Can we train a model in a way that makes him learn structure?



# StructBERT

**Architecture:** Encoder

---

**Task:** WSO + SSO

---

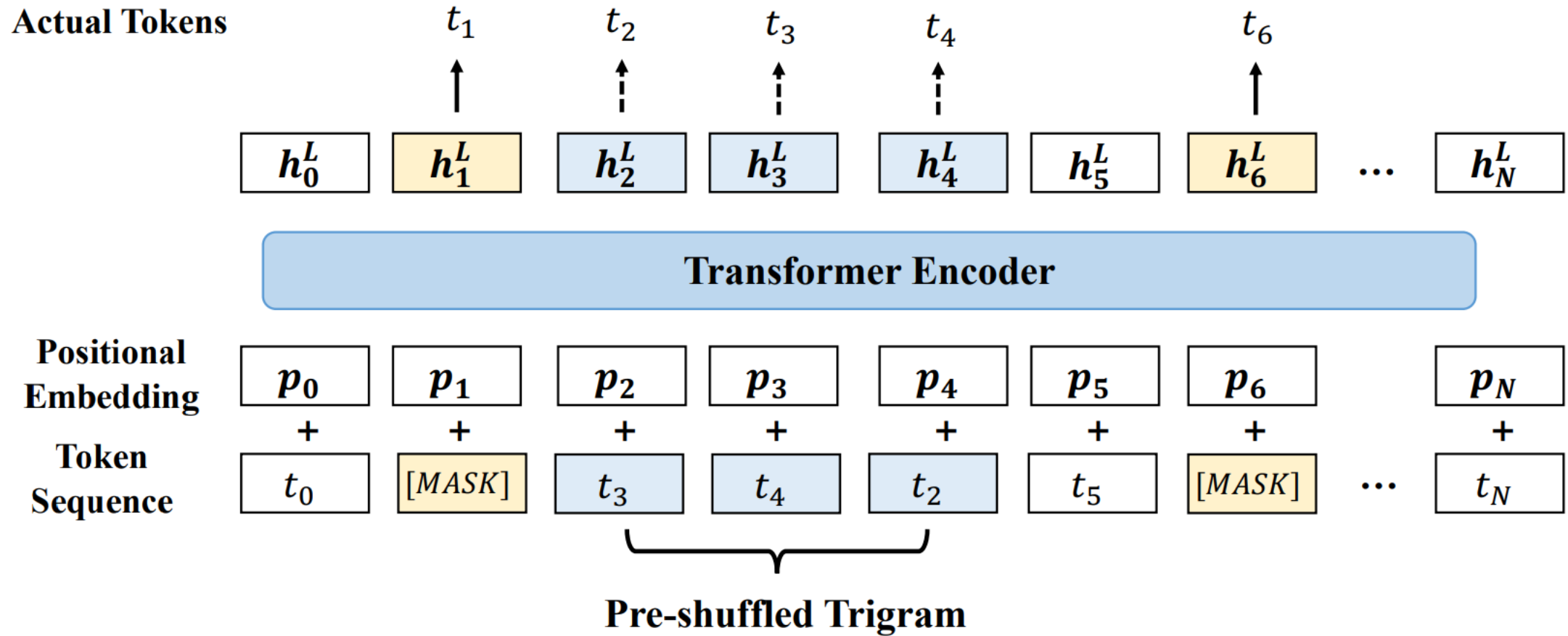
**Notes:** Trained to reorder  
shuffled text.

---

---

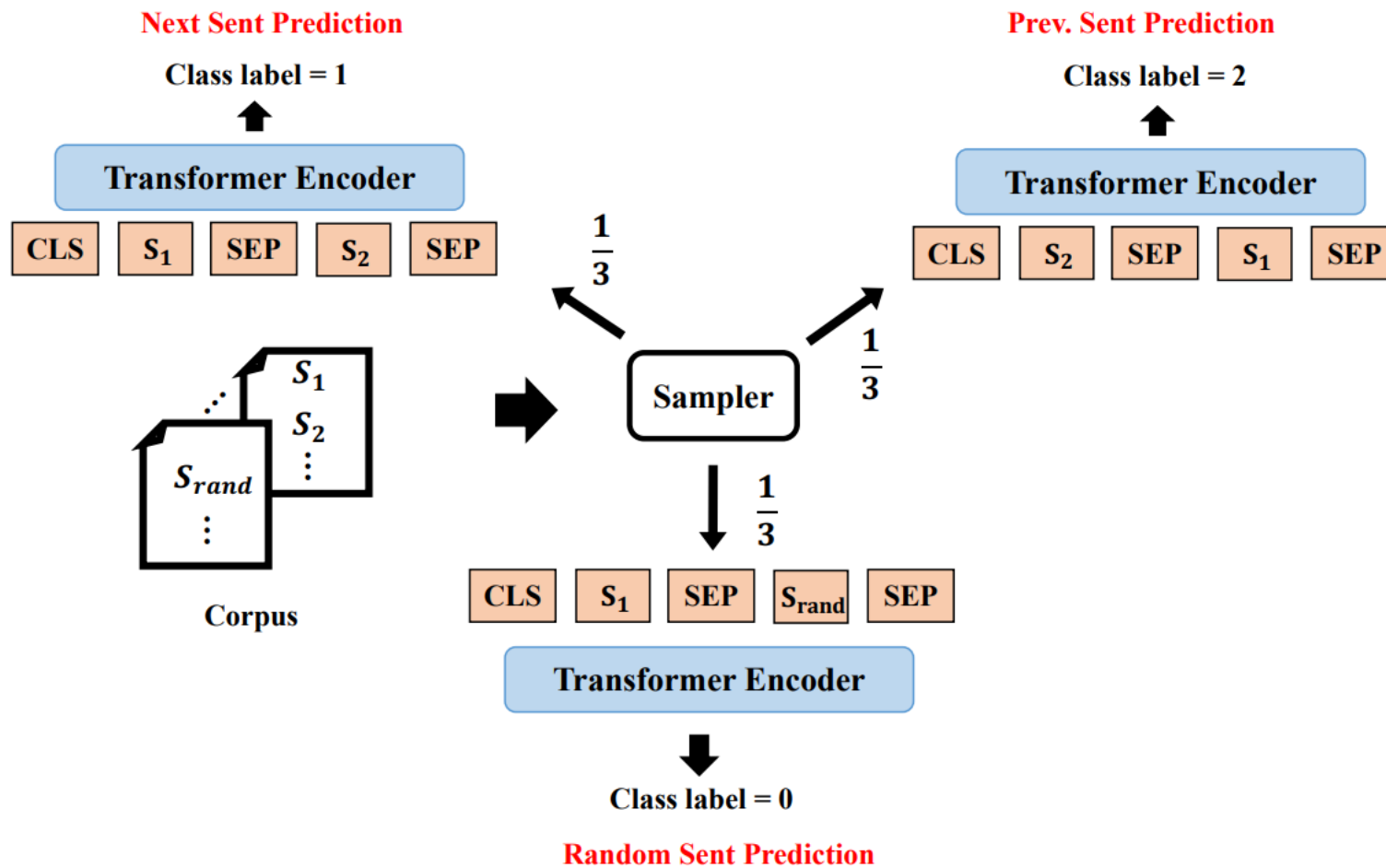


# Word Structural Objective



(a) Word Structural Objective

# Sentence Structural Objective



(b) Sentence Structural Objective

# Results

System	CoLA 8.5k	SST-2 67k	MRPC 3.5k	STS-B 5.7k	QQP 363k	MNLI 392k	QNLI 108k	RTE 2.5k	WNLI 634	AX	Average
Human Baseline	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4	92.0/92.8	91.2	93.6	95.9	-	
BERTLarge [6]	60.5	94.9	89.3/85.4	87.6/86.5	72.1/89.3	86.7/85.9	92.7	70.1	65.1	39.6	80.5
BERT on STILTs [19]	62.1	94.3	90.2/86.6	88.7/88.3	71.9/89.4	86.4/85.6	92.7	80.1	65.1	28.3	82.0
SpanBERT [12]	64.3	94.8	90.9/87.9	89.9/89.1	71.9/89.5	88.1/87.7	94.3	79.0	65.1	45.1	82.8
Snorkel MeTaL [22]	63.8	96.2	91.5/88.5	90.1/89.7	73.1/89.9	87.6/87.2	93.9	80.9	65.1	39.9	83.2
MT-DNN++ [14]	65.4	95.6	91.1/88.2	89.6/89.0	72.7/89.6	87.9/87.4	95.8	85.1	65.1	41.9	83.8
MT-DNN ensemble [14]	65.4	96.5	92.2/89.5	89.6/89.0	73.7/89.9	87.9/87.4	96.0	85.7	65.1	42.8	84.2
StructBERTBase	57.2	94.7	89.9/86.1	88.5/87.6	72.0/89.6	85.5/84.6	92.6	76.9	65.1	39.0	80.9
StructBERTLarge	65.3	95.2	92.0/89.3	90.3/89.4	74.1/90.5	88.0/87.7	95.7	83.1	65.1	43.6	83.9
StructBERTLarge ensemble	68.6	95.2	92.5/90.1	91.1/90.6	74.4/90.7	88.2/87.9	95.7	83.1	65.1	43.9	84.5
XLNet ensemble [32]	67.8	96.8	93.0/90.7	91.6/91.1	74.2/90.3	90.2/89.8	98.6	86.3	90.4	47.5	88.4
RoBERTa ensemble [15]	67.8	96.7	92.3/89.8	92.2/91.9	74.3/90.2	90.8/90.2	98.9	88.2	89.0	48.7	88.5
Adv-RoBERTa ensemble	68.0	96.8	93.1/90.8	92.4/92.2	<b>74.8/90.3</b>	<b>91.1/90.7</b>	98.8	<b>88.7</b>	89.0	<b>50.1</b>	88.8
StructBERTRoBERTa ensemble	<b>69.2</b>	<b>97.1</b>	<b>93.6/91.5</b>	<b>92.8/92.4</b>	<b>74.4/90.7</b>	90.7/90.3	<b>99.2</b>	87.3	<b>89.7</b>	47.8	<b>89.0</b>

# Electra

**Architecture:** Encoder

---

**Task:** Replaced Token Detection

---

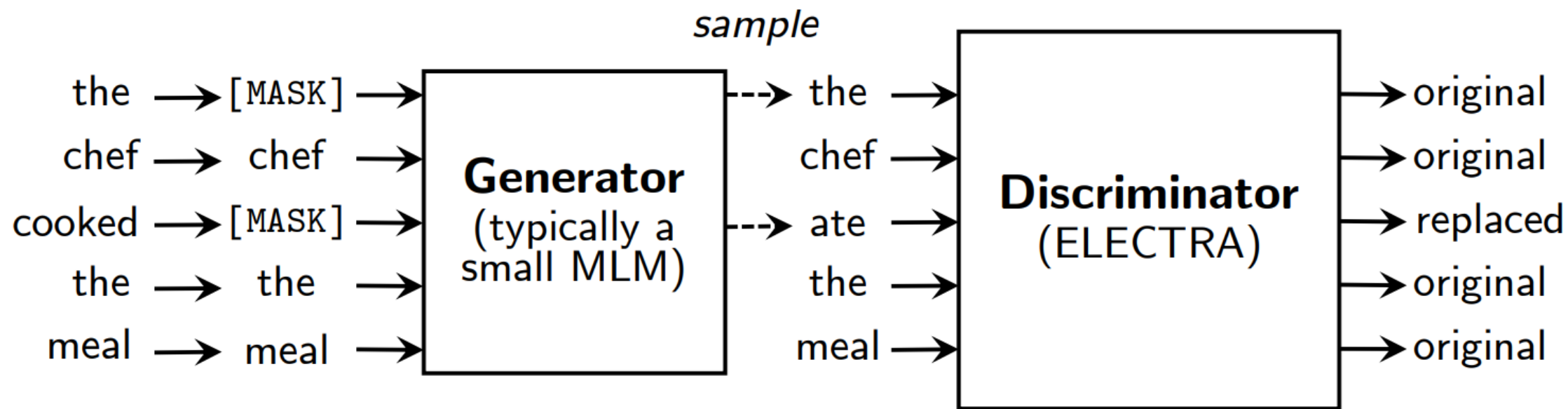
**Notes:** Uses a signal from all  
the text and not just 15%.

---

---



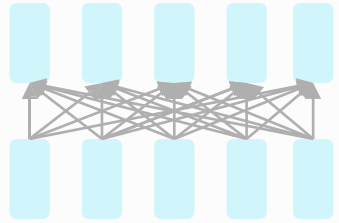
# Guest Presentation!



# Decoders

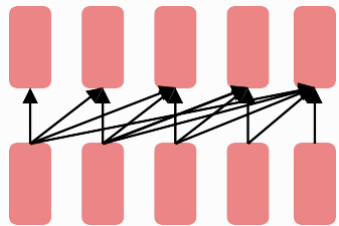
# Pre-training for three types of architectures

The neural architecture influences the type of pretraining, and natural use cases.



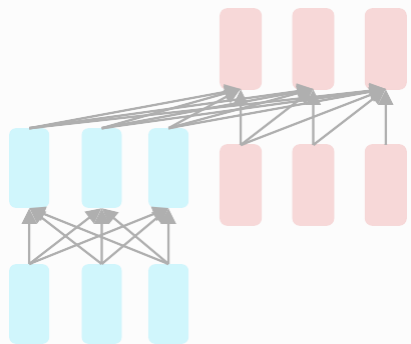
**Encoders**

- Gets bidirectional context – can condition on future!
- Question is, how do we pre-train them?



**Decoders**

- Language models! Impose a probability distribution on the language.
- Nice to generate from; can't condition on future words



**Encoder-  
Decoders**

- Good parts of decoders and encoders?
- What's the best way to pretrain them?



# GPT-2

**Architecture:** Decoder

---

**Task:** Language Modeling

---

**Notes:** Great at writing

---

text!

---

---



# What is a Language Model?

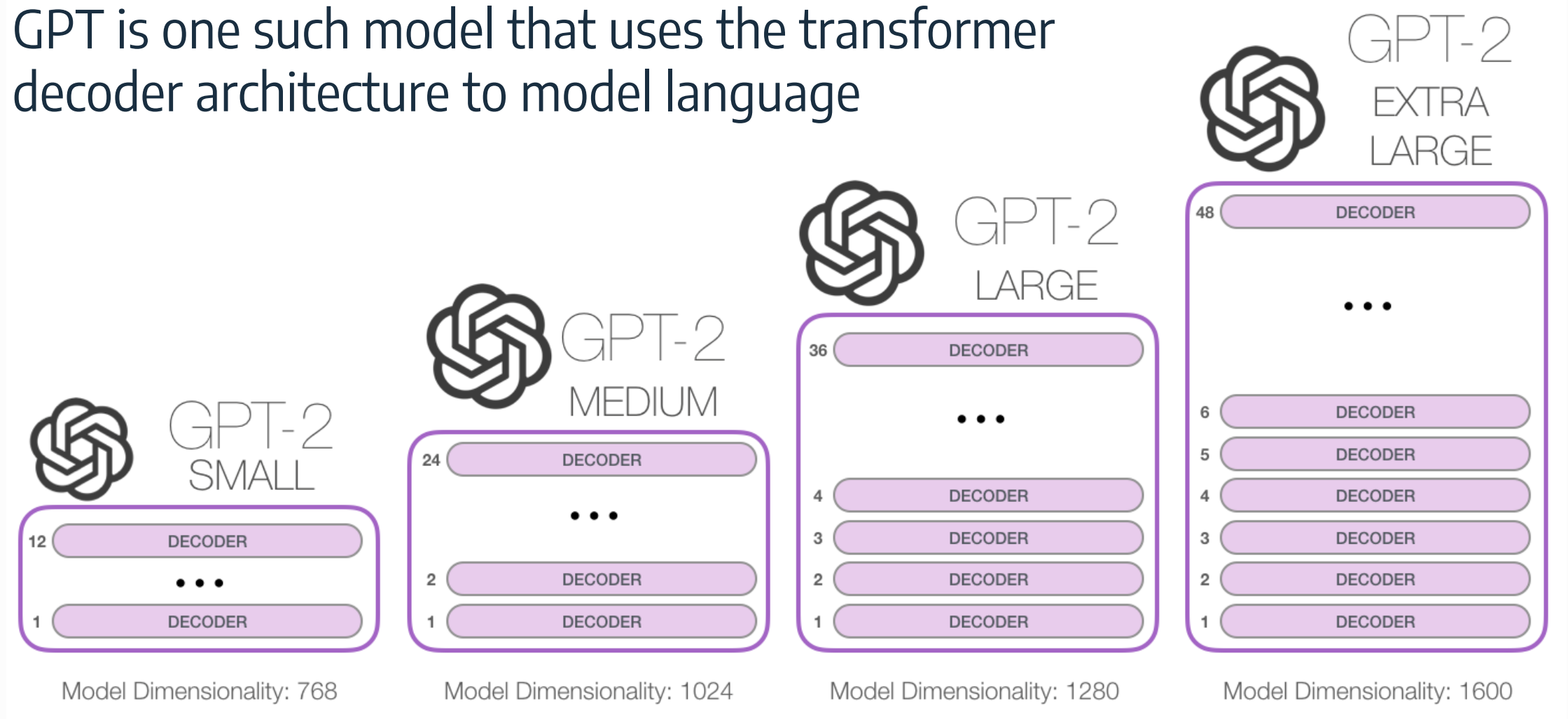
A **Language Model** (LM) is a model that outputs the probability of the next token given a context:

$$p(t_n | t_1, \dots, t_{n-1})$$



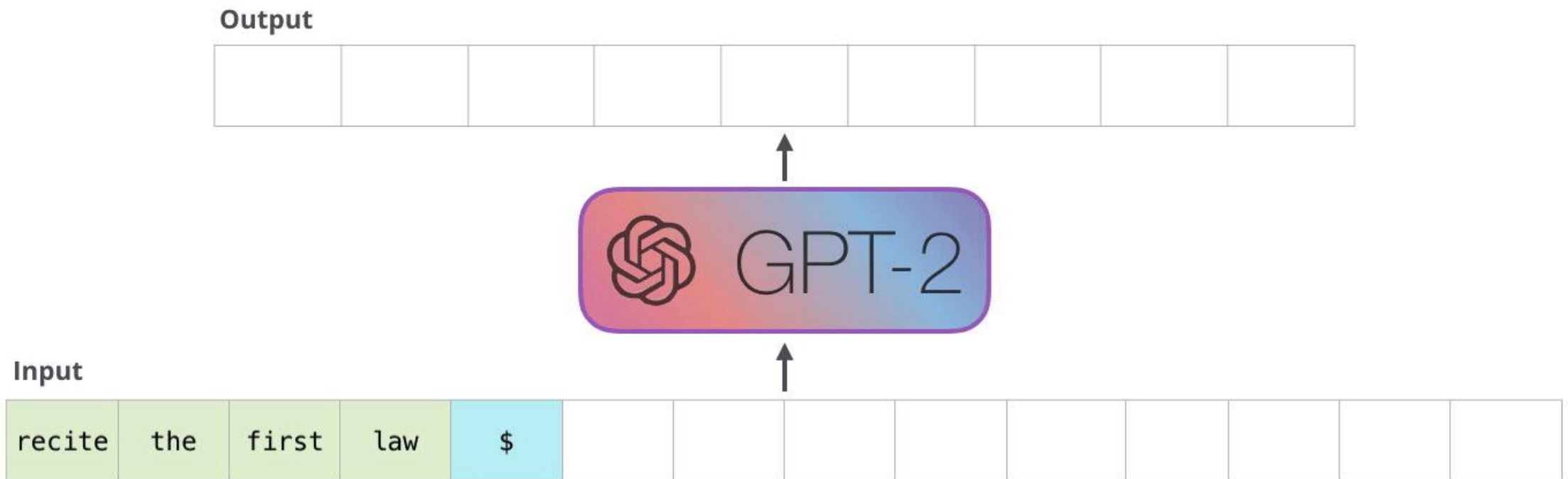
# Architecture

GPT is one such model that uses the transformer decoder architecture to model language



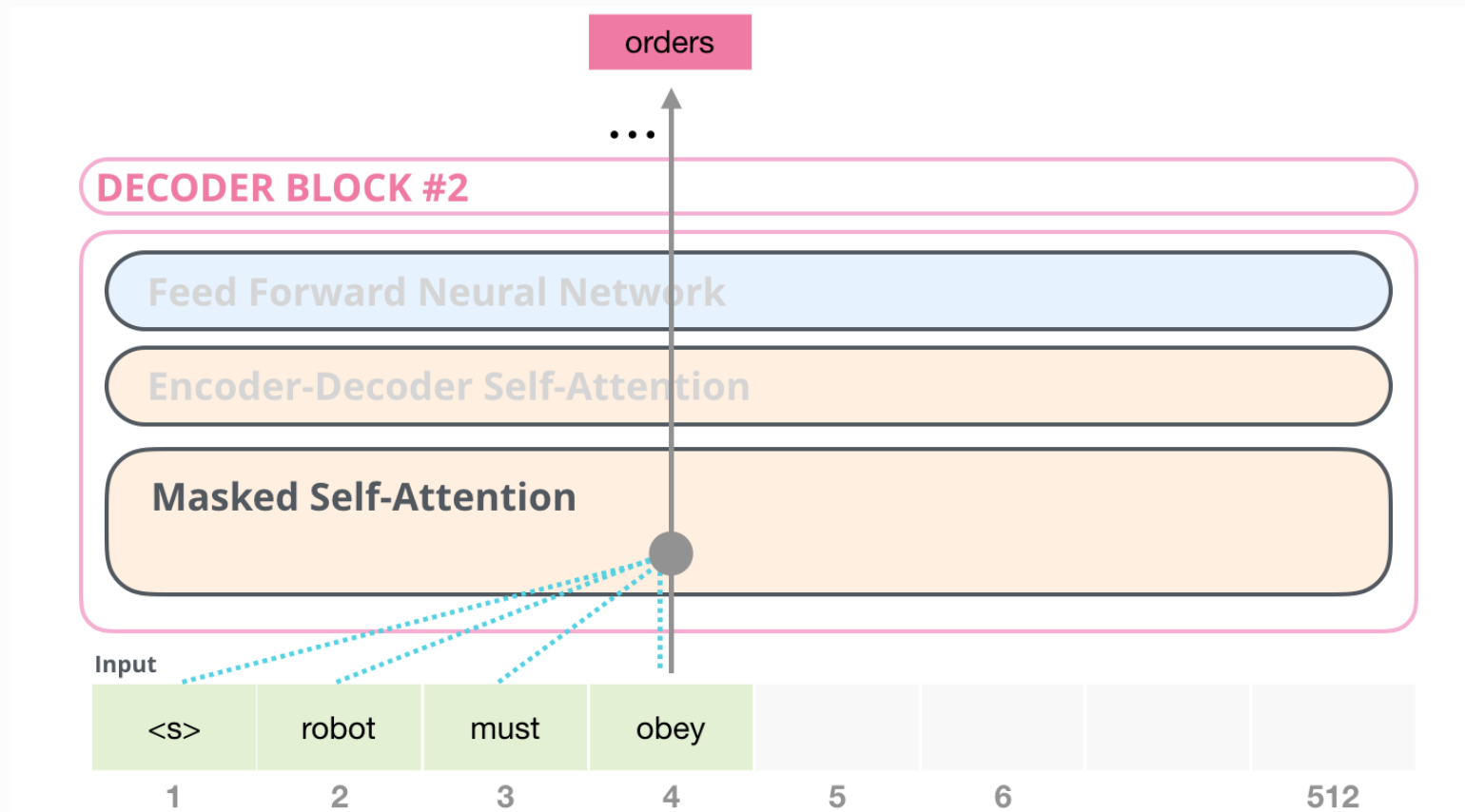
# One Token at a Time

GPT is an **autoregressive** model, that is, it generates one token at a time and adds it to the context for the next iteration.



# Decoder Block

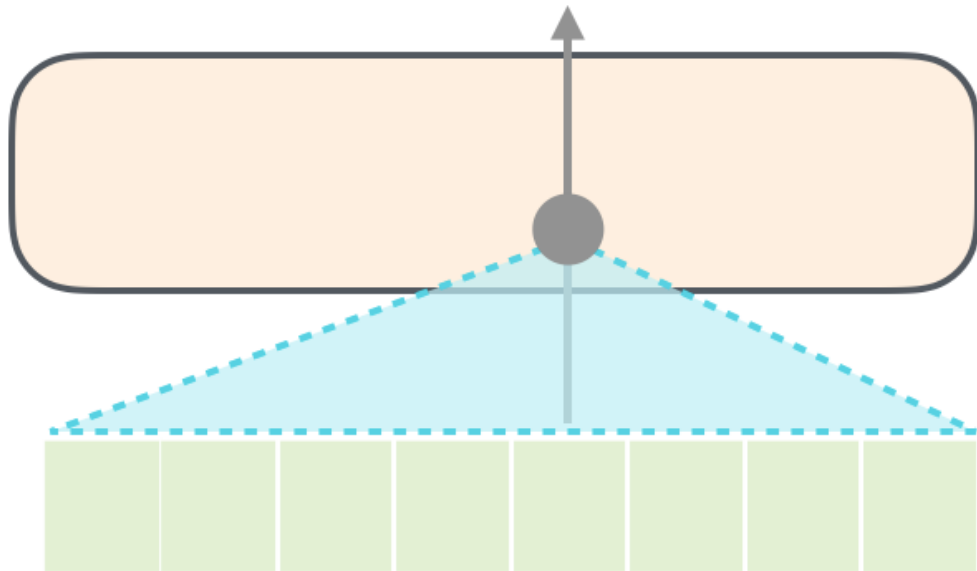
Recall we saw that the transformer decoder block uses **masked self-attention** in order to mask the future of the sentence.



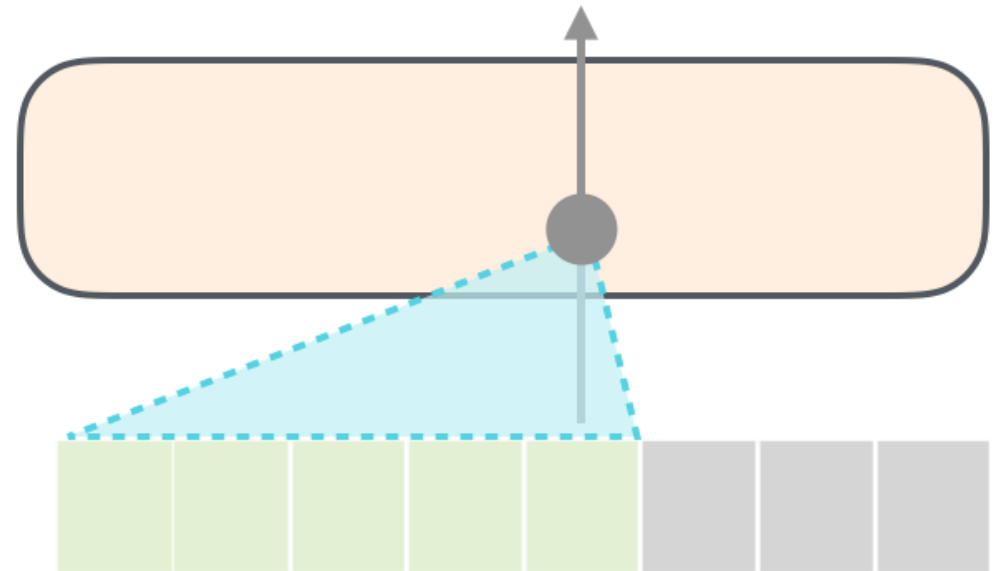
# Masked Self-Attention

The masking is done by replacing the attention scores in the appropriate place with  $-\infty$ , so after the softmax their score would be 0. For further details, see this [blog post](#).

Self-Attention

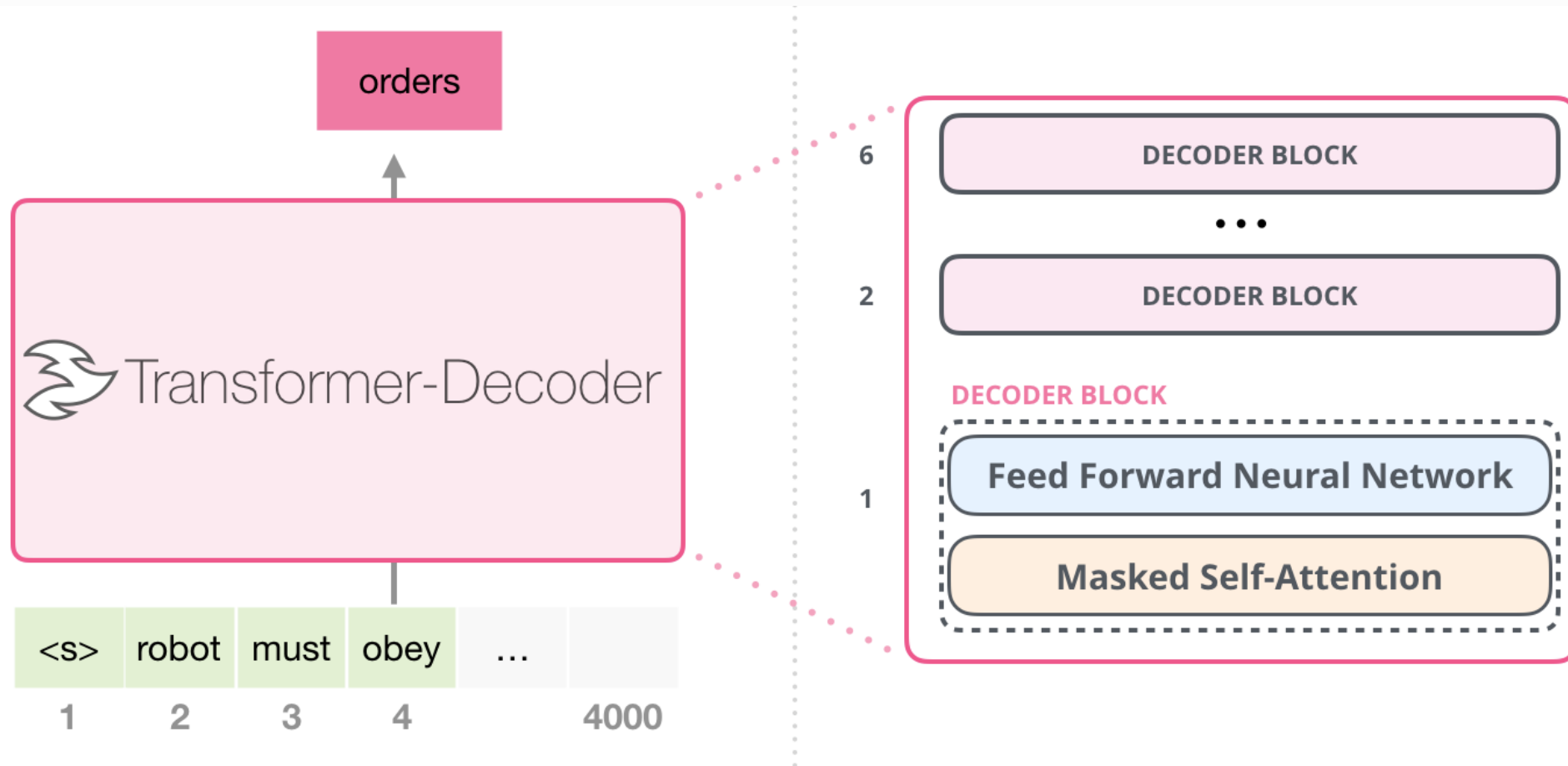


Masked Self-Attention



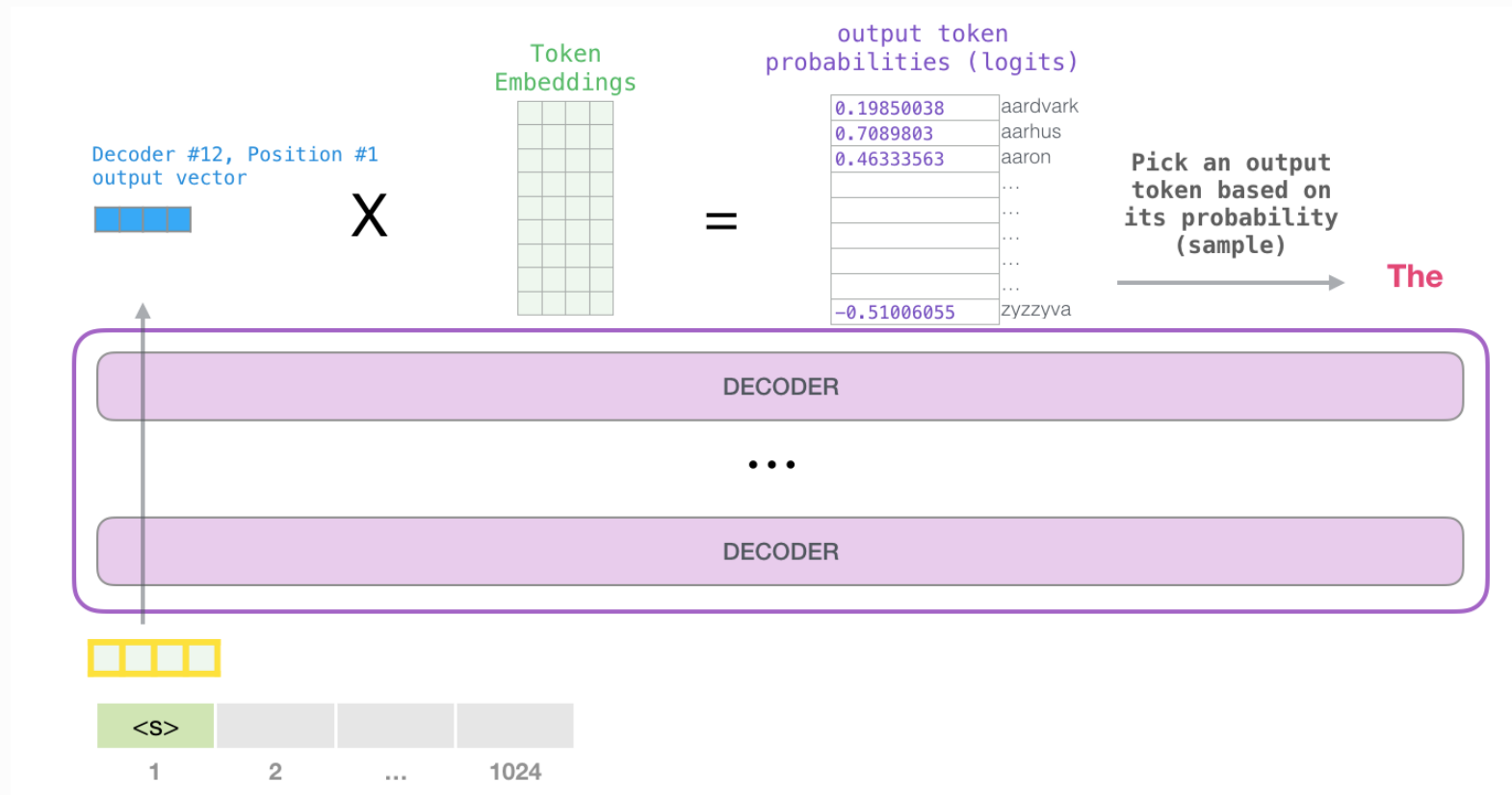
# Decoder-Only Block

Since GPT uses only decoder, there is no need in encoder-decoder attention.



# The Decoding

In the decoding phase, instead of passing the output vector of the last token through a linear classifier, GPT multiplies it by the input embedding matrix and applies a softmax to get a probability distribution over the vocabulary





# The Decoding

Then, there are 3 strategies to get the actual token:

1. Taking the token with the largest score.
2. Sample a token using the distribution.
3. Sample a token from the top k scores of the distribution

output token probabilities (logits)

model vocabulary size  
50,257

0.19850038	aardvark
0.7089803	aarhus
0.46333563	aaron
	...
	...
	...
	...
	...
-0.51006055	zyzzyva



# Encoder & Decoder



# Text to Text Transfer Transformer (T5)

**Architecture:** Encoder + Decoder

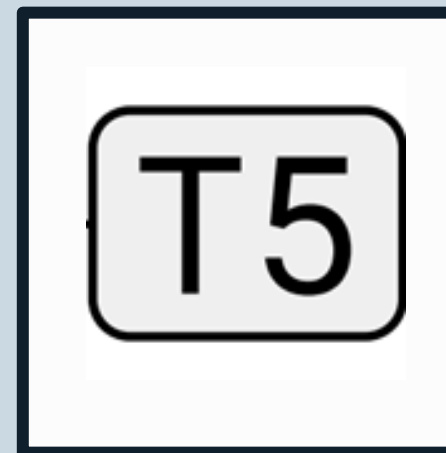
---

**Task:** Span Completion

---

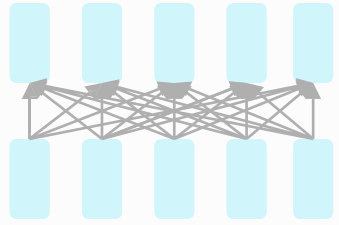
**Notes:** Is a result of extensive  
experimentation of pretraining techniques.

---



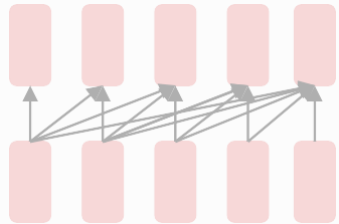
# Pre-training for three types of architectures

The neural architecture influences the type of pretraining, and natural use cases.



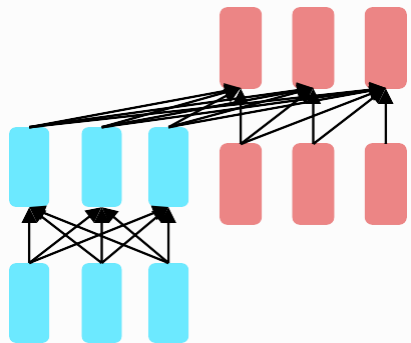
**Encoders**

- Gets bidirectional context – can condition on future!
- Question is, how do we pre-train them?



**Decoders**

- Language models! Impose a probability distribution on the language.
- Nice to generate from; can't condition on future words



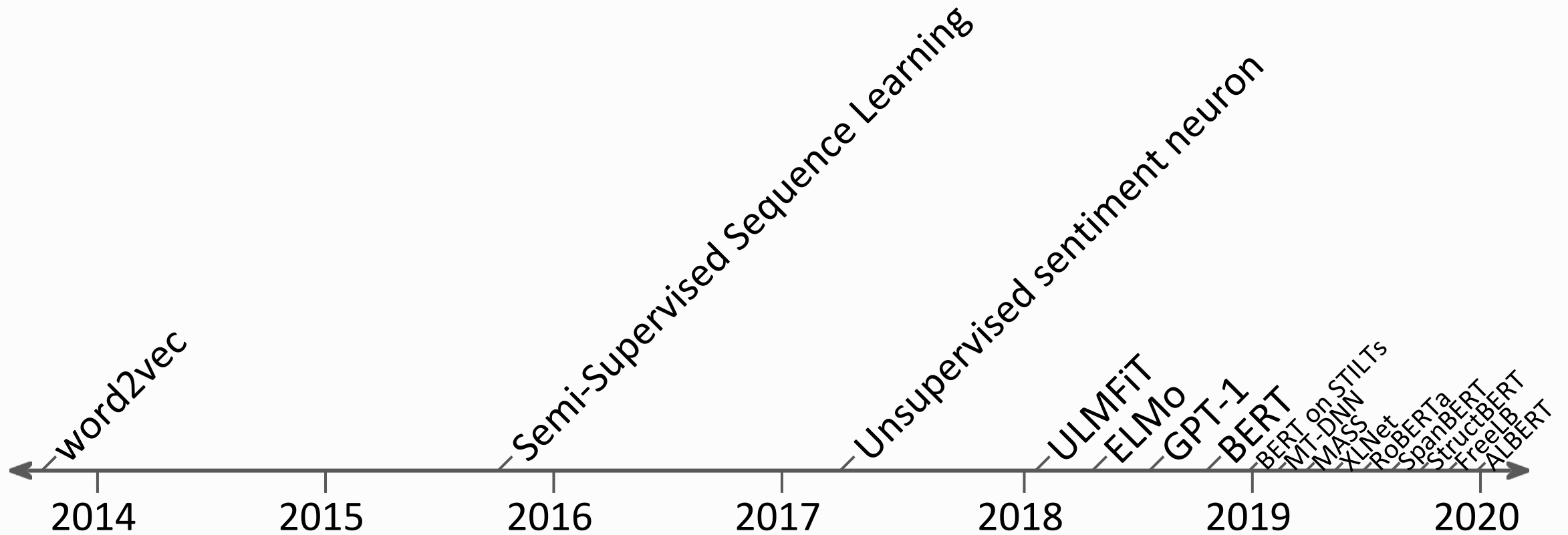
**Encoder-  
Decoders**

- Good parts of decoders and encoders?
- What's the best way to pretrain them?

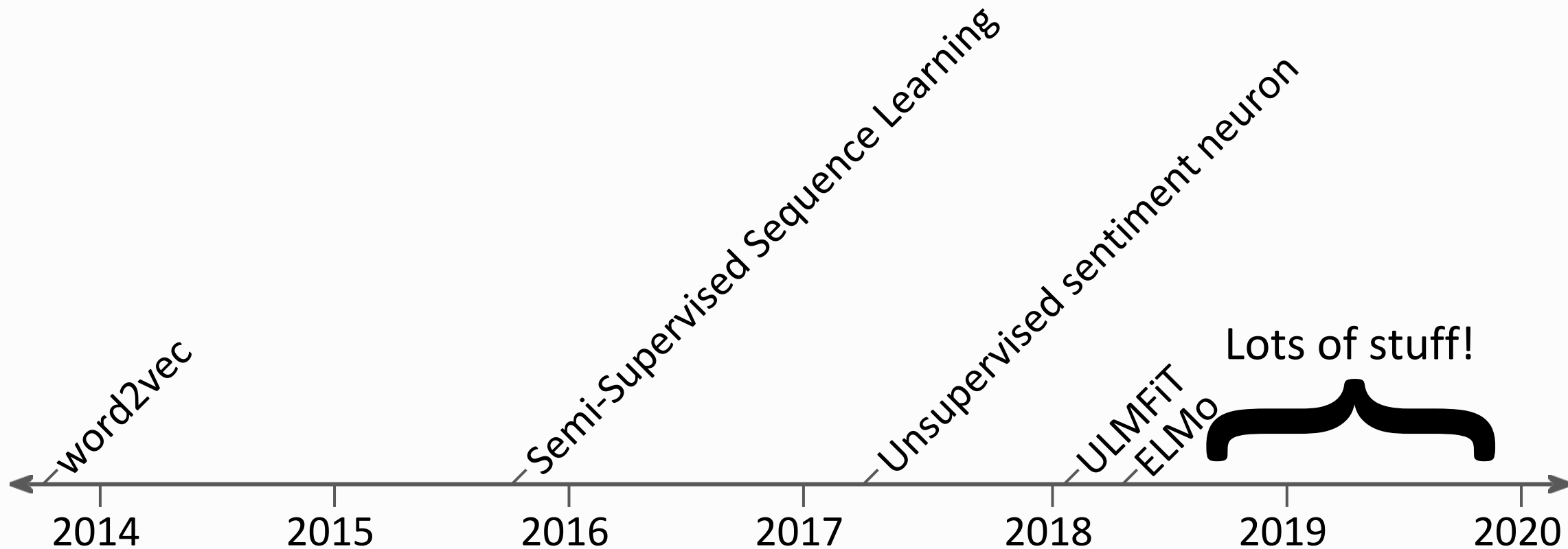
# T5: Research Questions

Which transfer learning methods work best, and what happens when we scale them up?

# NLP Timeline



# NLP Timeline



# Models in Modern NLP

- Paper A proposes an unsupervised pre-training technique called "FancyLearn".
- Paper B proposes another pre-training technique called "FancierLearn" and achieves better results.
- Paper A uses Wikipedia for unlabeled data.
- Paper B uses Wikipedia and the Toronto Books Corpus.
- *Is FancierLearn better than FancyLearn?*



# Models in Modern NLP

- Paper A proposes an unsupervised pre-training technique called "FancyLearn".
- Paper B proposes another pre-training technique called "FancierLearn" and achieves better results.
- Paper A uses a model with 100 million parameters.
- Paper B uses a model with 200 million parameters.
- *Is FancierLearn better than FancyLearn?*


# Models in Modern NLP

- Paper A proposes an unsupervised pre-training technique called "FancyLearn".
- Paper B proposes another pre-training technique called "FancierLearn" and achieves better results.
- Paper A pre-trains on 100 billion tokens of unlabeled data.
- Paper B pre-trains on 200 billion tokens of unlabeled data.
- Is FancierLearn better than FancyLearn?

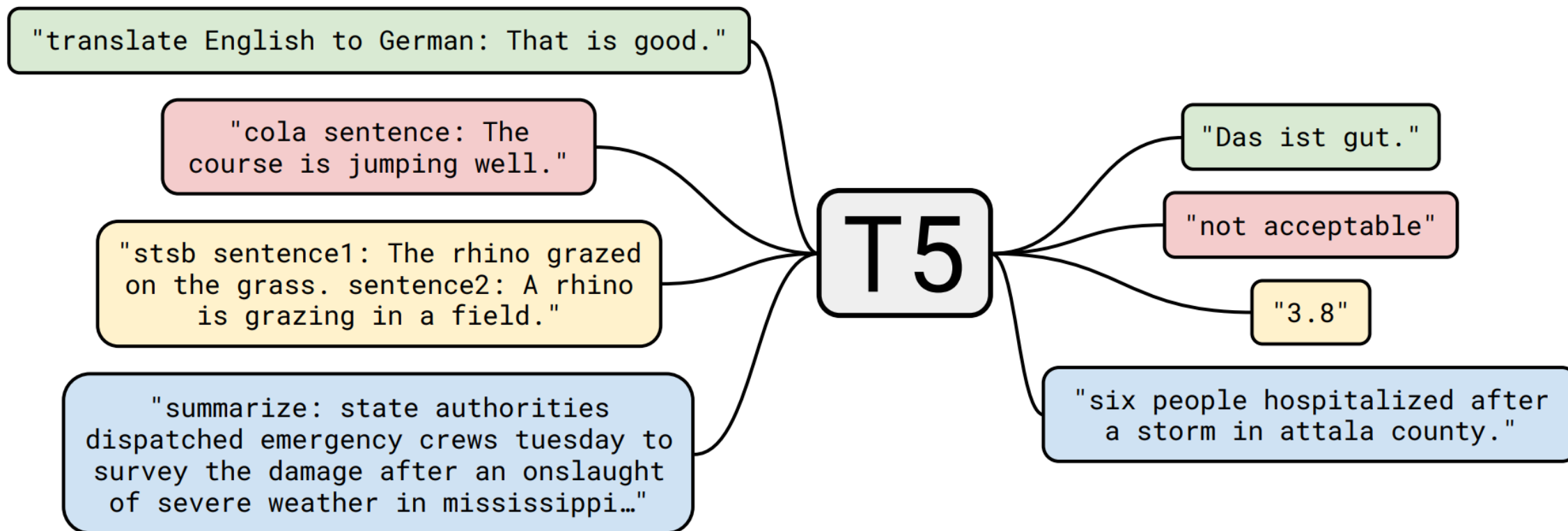
# Models in Modern NLP

- Paper A proposes an unsupervised pre-training technique called "FancyLearn".
- Paper B proposes another pre-training technique called "FancierLearn" and achieves better results.
- Paper A uses the Adam optimizer.
- Paper B uses SGD with momentum.
- Is FancierLearn better than FancyLearn?

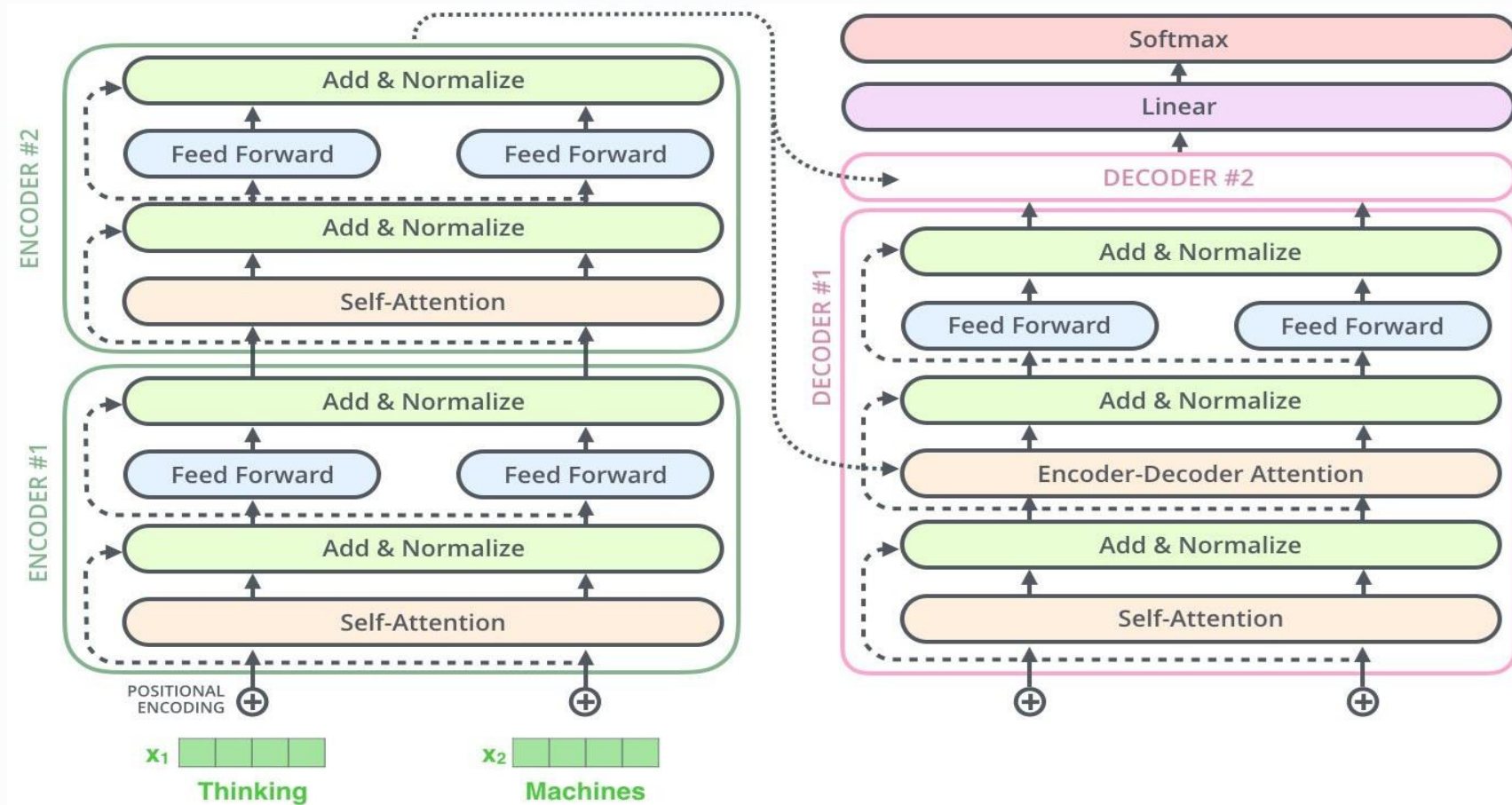
**Given the current  
landscape of transfer  
learning for NLP, what  
works best? And how  
far can we push the  
tools we already have?**



# Text to Text Transfer Transformer (T5)



# Architecture: Encoder & Decoder



Source: <http://jalammar.github.io/illustrated-transformer/>

# Common Crawl Web Extracted Text

## Menu

Lemon

## Introduction

The lemon, Citrus Limon (L.) Osbeck, is a species of small evergreen tree in the flowering plant family rutaceae.

The tree's ellipsoidal yellow fruit is used for culinary and non-culinary purposes throughout the world, primarily for its juice, which has both culinary and cleaning uses. The juice of the lemon is about 5% to 6% citric acid, with a ph of around 2.2, giving it a sour taste.

## Article

The origin of the lemon is unknown, though lemons are thought to have first grown in Assam (a region in northeast India), northern Burma or China.

A genomic study of the lemon indicated it was a hybrid between bitter orange (sour orange) and citron.

Please enable JavaScript to use our site.

Home  
Products  
Shipping  
Contact  
FAQ

Dried Lemons, \$3.59/pound

Organic dried lemons from our farm in California.

Lemons are harvested and sun-dried for maximum flavor.

Good in soups and on popcorn.

The lemon, Citrus Limon (L.) Osbeck, is a species of small evergreen tree in the flowering plant family rutaceae.

The tree's ellipsoidal yellow fruit is used for culinary and non-culinary purposes throughout the world, primarily for its juice, which has both culinary and cleaning uses.

The juice of the lemon is about 5% to 6% citric acid, with a ph of around 2.2, giving it a sour taste.

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Curabitur in tempus quam. In mollis et ante at consectetur.

Aliquam erat volutpat.

Donec at lacinia est.

Duis semper, magna tempor interdum suscipit, ante elit molestie urna, eget efficitur risus nunc ac elit.

Fusce quis blandit lectus.

Mauris at mauris a turpis tristique lacinia at nec ante.

Aenean in scelerisque tellus, a efficitur ipsum.

Integer justo enim, ornare vitae sem non, mollis fermentum lectus.

Mauris ultrices nisl at libero porta sodales in ac orci.

```
function Ball(r) {  
    this.radius = r; this.area  
    = pi * r ** 2; this.show =  
    function(){  
        drawCircle(r);  
    }  
}
```

# Common Crawl Web Extracted Text

Menu

Lemon

Introduction

The lemon, Citrus Limon (L.) Osbeck, is a species of small evergreen tree in the flowering plant family rutaceae.

The tree's ellipsoidal yellow fruit is used for culinary and non-culinary purposes throughout the world, primarily for its juice, which has both culinary and cleaning uses. The juice of the lemon is about 5% to 6% citric acid, with a ph of around 2.2, giving it a sour taste.

Article

The origin of the lemon is unknown, though lemons are thought to have first grown in Assam (a region in northeast India), northern Burma or China.

A genomic study of the lemon indicated it was a hybrid between bitter orange (sour orange) and citron.

Please enable JavaScript to use our site.

Home  
Products  
Shipping  
Contact  
FAQ

Dried Lemons, \$3.59/pound

Organic dried lemons from our farm in California.

Lemons are harvested and sun-dried for maximum flavor.

Good in soups and on popcorn.

The lemon, Citrus Limon (L.) Osbeck, is a species of small evergreen tree in the flowering plant family rutaceae.

The tree's ellipsoidal yellow fruit is used for culinary and non-culinary purposes throughout the world, primarily for its juice, which has both culinary and cleaning uses.

The juice of the lemon is about 5% to 6% citric acid, with a ph of around 2.2, giving it a sour taste.

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Curabitur in tempus quam. In mollis et ante at consectetur.

Aliquam erat volutpat.

Donec at lacinia est.

Duis semper, magna tempor interdum suscipit, ante elit molestie urna, eget efficitur risus nunc ac elit.

Fusce quis blandit lectus.

Mauris at mauris a turpis tristique lacinia at nec ante.

Aenean in scelerisque tellus, a efficitur ipsum.

Integer justo enim, ornare vitae sem non, mollis fermentum lectus.

Mauris ultrices nisl at libero porta sodales in ac orci.

```
function Ball(r) {  
    this.radius = r; this.area  
    = pi * r ** 2; this.show =  
    function(){  
        drawCircle(r);  
    }  
}
```



# Pretraining Objective

Original text

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



Inputs

Thank you <X> me to your party <Y> week.

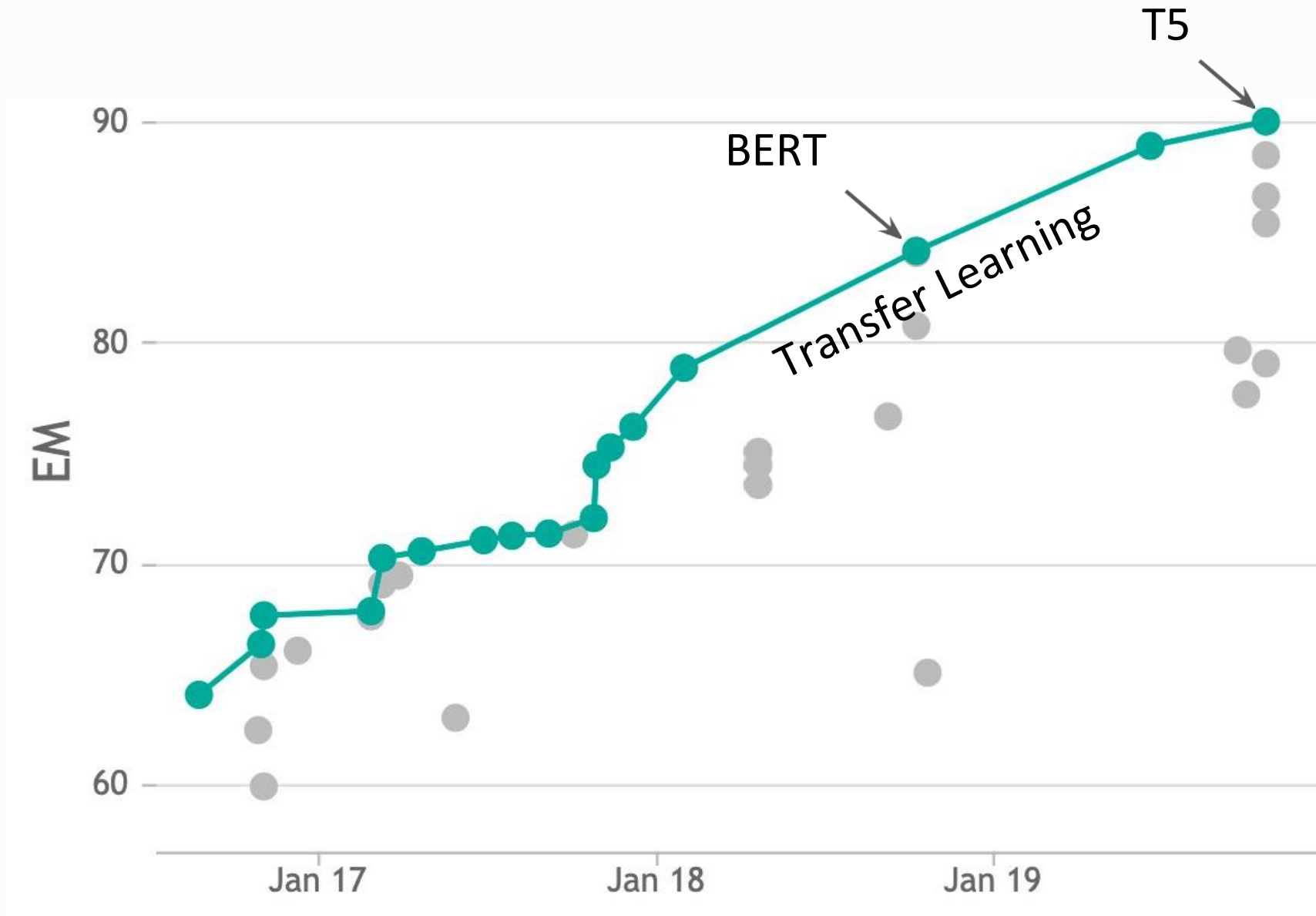
Targets

<X> for inviting <Y> last <Z>

# Experiments

- The authors of the paper tried an **enormous** amount of different:
  - Architectures
  - Objectives
  - Optimizers
  - Hyperparameters
- gave the best results. Refer to the paper for more.

# Results on SQuAD



Source:

<https://paperswithcode.com/sota/question-answering-on-squad11-dev>



# Other Variants

# The Transformer Isn't Perfect

Although the transformer based-models we saw brought huge advancements in the field of NLP, they still have some weak-points that make them impractical. In this section we will see two different models, each tries to remedy a different disadvantage in the Transformers:

- BigBird
- DistillBERT



# BigBird

**Architecture:** Encoder + Sparse Attention

---

**Task:** MLM

---

**Notes:** Can process longer  
texts.

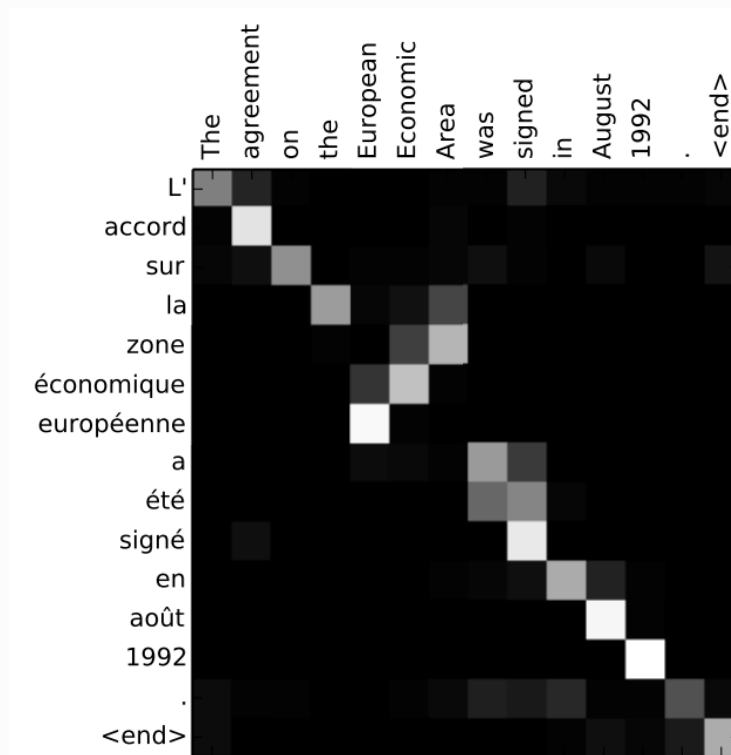
---

---



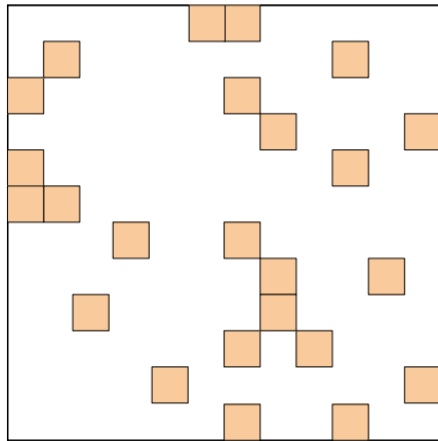
# BigBird

- **Problem:** Transformers can't process long texts, because the complexity of the attention mechanism is quadratic in the input length. Thus the regular transformer can process up to 512 tokens.

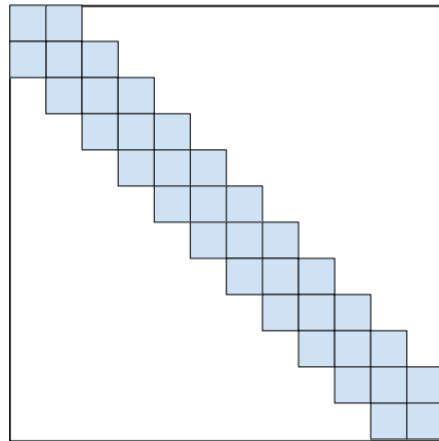


# BigBird

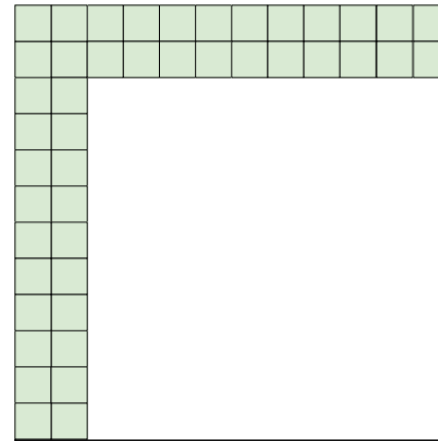
**Solution:** Sparse attention mechanisms, when carefully designed, can achieve a  $O(n)$  complexity but still allow information to flow in the sequence (see paper for more information).



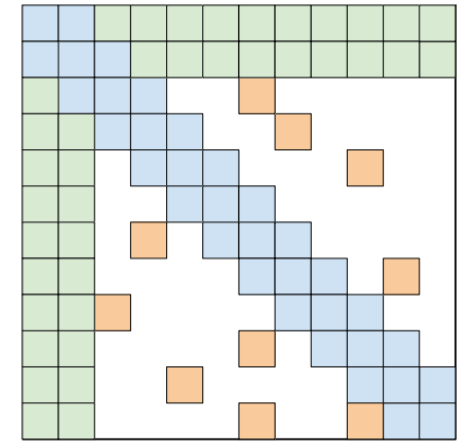
(a) Random attention



(b) Window attention



(c) Global Attention



(d) BIGBIRD



# BigBird

**Results:** BigBird can process up to 4096 (!) tokens, and shows SOTA results on Question Answering.

Model	HotpotQA			NaturalQ		TriviaQA	WikiHop
	Ans	Sup	Joint	LA	SA	Full	MCQ
RoBERTa	73.5	83.4	63.5	-	-	74.3	72.4
Longformer	74.3	84.4	64.4	-	-	75.2	75.0
BIGBIRD-ITC	<b>75.7</b>	86.8	67.7	70.8	53.3	<b>79.5</b>	<b>75.9</b>
BIGBIRD-ETC	75.5	<b>87.1</b>	<b>67.8</b>	<b>73.9</b>	<b>54.9</b>	78.7	<b>75.9</b>

Table 2: QA Dev results using Base size models. We report accuracy for WikiHop and F1 for HotpotQA, Natural Questions, and TriviaQA.

# DistillBERT

**Architecture:** Small Encoder

---

**Task:** MLM + Distillation

---

**Notes:** An efficient, distilled  
BERT.

---

---



# DistillBERT

**Problem:** The pretrained language models have many parameters, resulting in long inference time. In addition, even loading a large LM requires expensive hardware.

**Naïve Solution:** Train smaller LMs. Problem: worse results.

So the question is, is there a way to get large-LM like results while using less parameters?

**Better Solution:** Compress a large LM!



# Background – Model Compression

There are three main ways to compress a large model:

1. **Pruning:** Start with a large model, then iteratively remove layers/attention heads/weights and continue training of smaller model.
2. **Quantization:** Represent model's weights using lower precision. Instead of 32floats use 16float for example.
3. **Knowledge Distillation:** Train a small model to mimic the large model's outputs.



# Background - Knowledge Distillation

- We have a teacher model T and a student model S, both give an input return logits for the possible classes.
- The student is trained using an extrapolation of two losses:
  1. The loss on the task.

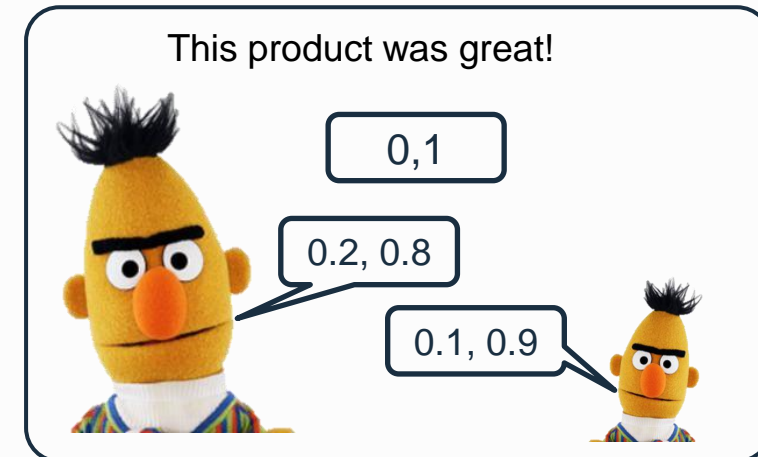
$$L_{task} = CE(\hat{y}_{student}, y)$$

2. The KL divergence from the teacher model's distribution:

$$L_{KL} = KL(\hat{y}_{student}, \hat{y}_{teacher})$$

The final loss will be:

$$L = \alpha \cdot L_{task} + (1 - \alpha) \cdot L_{KL}$$



# DistillBERT

- DistillBERT is just a distilled version of bert.
- The distillation occurred during the training of a bert model and used a loss very similar to what we saw earlier (with a small addition)
- The resulting model has 40% less params and has faster inference by 60%.

Table 3: **DistilBERT is significantly smaller while being constantly faster.** Inference time of a full pass of GLUE task STS-B (sentiment analysis) on CPU with a batch size of 1.

Model	# param. (Millions)	Inf. time (seconds)
ELMo	180	895
BERT-base	110	668
DistilBERT	66	410

# DistillBERT

**Results:** DistillBERT retains 97% percent of BERT's performance (!) across a variety of tasks.

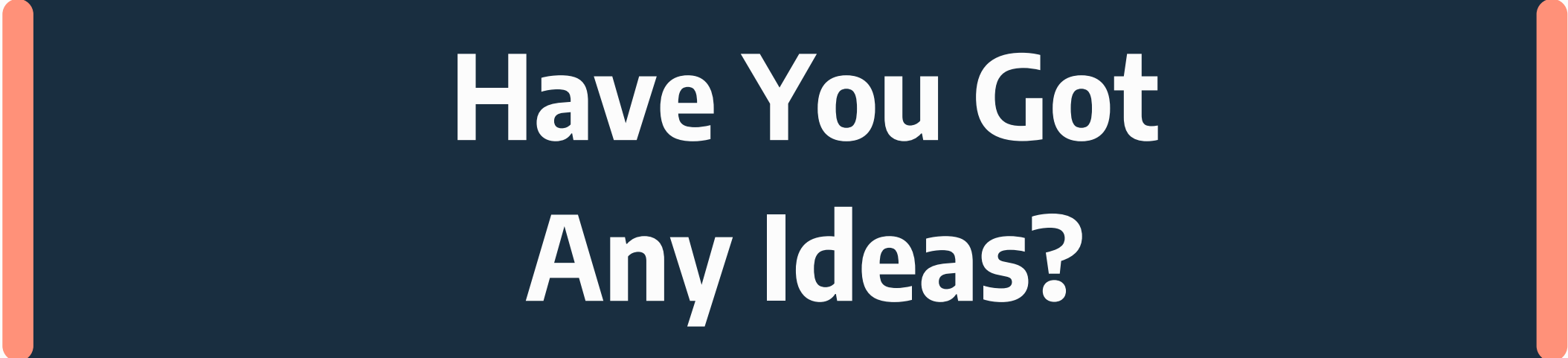
Table 1: **DistilBERT retains 97% of BERT performance.** Comparison on the dev sets of the GLUE benchmark. ELMo results as reported by the authors. BERT and DistilBERT results are the medians of 5 runs with different seeds.

Model	Score	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	WNLI
ELMo	68.7	44.1	68.6	76.6	71.1	86.2	53.4	91.5	70.4	56.3
BERT-base	79.5	56.3	86.7	88.6	91.8	89.6	69.3	92.7	89.0	53.5
DistilBERT	77.0	51.3	82.2	87.5	89.2	88.5	59.9	91.3	86.9	56.3

# Other Models

- ALBERT – A lite BERT with parameter sharing
- BART – Language Autoencoder
- Ernie – LM with knowledge graphs
- Retrieval-Based models
- Multilingual Models (more on that later!)
- Distill\_\_\_\_\_
- Multi-Modal Models





**Have You Got  
Any Ideas?**