NPTEL ONLINE CERTIFICATION COURSES

DEEP LEARNING FOR NATURAL LANGUAGE PROCESSING

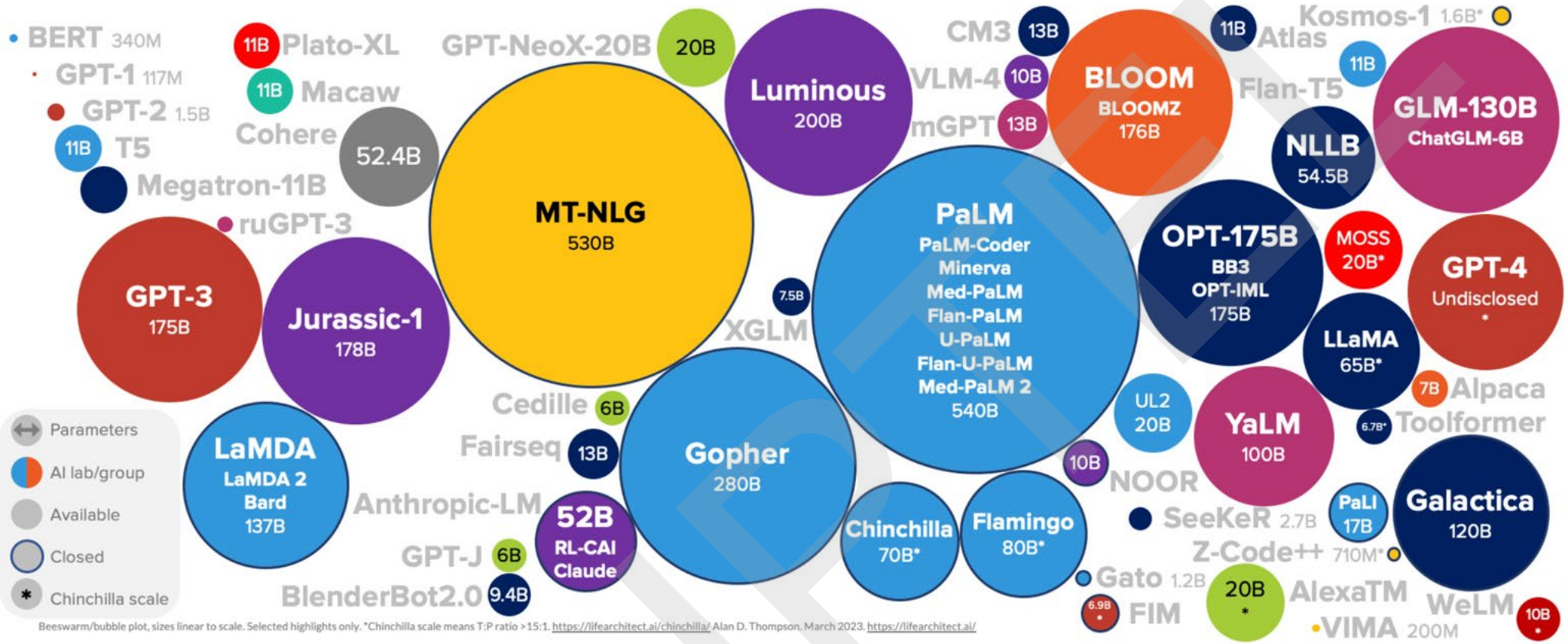
Lecture 26 : Pretraining

PROF. PAWAN GOYAL
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

CONCEPTS COVERED

- What is Pretraining?
- Pretrained LSTMs: ELMo
- Pretraining Transformers: Large Language Models

The idea of transfer learning



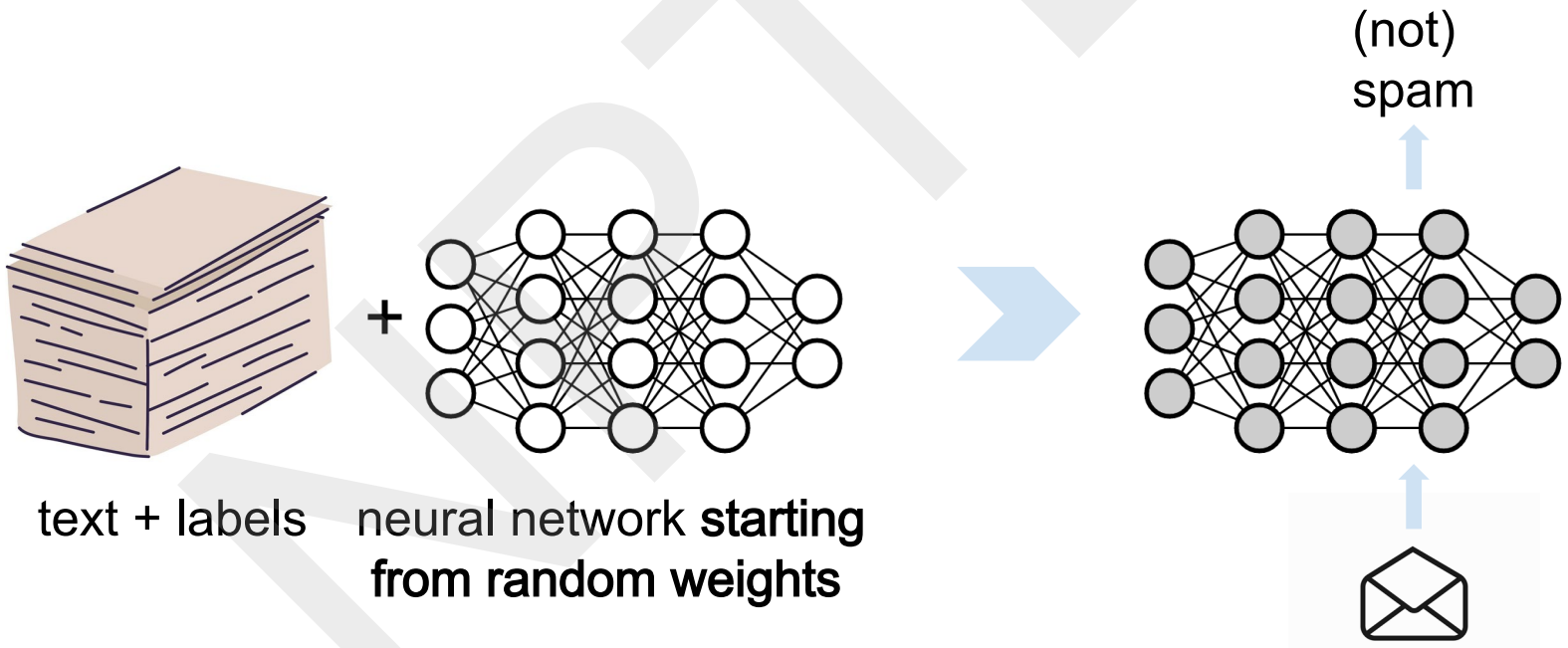
Transformers are not always better than RNNs by themselves.

However, what has made Transformers popular is that they can be combined with the idea of transfer learning.

Transformers have become the go-to model for building large pretrained language models which can be adapted for several tasks

https://rycolab.io/classes/llm-s24/

Standard Supervised Deep Learning



https://utah-cs6340-nlp.notion.site/Natural-Language-Processing-bd1a2ca290fc44f69556908ad8d25c70

Abundance in domain-specific NLP Applications

CUSTOMER SUPPORT DOMAIN

How can I encrypt my SD Card?

Section

Answer

Settings

Encrypt or decrypt SD card

You can encrypt your optional memory card (not included) to protect its data. This only allows the SD card information to be accessed from your device with a password.

1. From Settings, tap Biometrics and security > Encrypt or decrypt SD card.
2. Tap Encrypt SD card and follow the prompts to encrypt all data on your memory card.

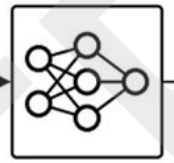
NOTE Performing a Factory data reset on your device prevents it from accessing an encrypted SD card. Before initiating a Factory data reset, make sure to decrypt the installed SD card first.

Decrypt SD card

You can decrypt an optional memory card (not included) if it was encrypted by this device. You may want to decrypt the memory card if you plan to use it with another device or before performing a Factory data

Question Answering

LEGAL DOMAIN



Output

Governing Law: "This Agreement shall be governed by the laws of the State of California without giving effect to conflict or choice of law principles." (Page 2)

Covenant Not to Sue: "In addition, Company shall not now or in the future contest the validity of Investor's ownership of its Intellectual Property." (Page 30)

Perpetual / Irrevocable License: "Company grants to Investor a worldwide, royalty-free, exclusive, irrevocable license (with the right to grant sublicenses)." (Page 151)

Clause Span Extraction

FINANCIAL DOMAIN

Cash and Cash Equivalents

Debt Instrument Convertible Conversion Price

As of December 31, 2020, we had cash equivalents of $*24.8* million and a closing stock price of $*18.20* per share.

We also acquired a business loan from the U.S. Bancorp of $*60.5* million.

Line of Credit Facility Maximum Borrowing Capacity

Impairment Loss

Finally, our firm reports *no* impairment loss for this year.

NER

Named Entity Recognition
NCBI disease, BC2GM, ...

Relation Extraction
EU-ADR, ChemProt, ...

Question Answering
BioASQ 5b, BioASQ 6b, ...

the adult renal failure cause ...
▶ O O B I O ...

Variants in the @GENE$ region contribute to @DISEASE$ susceptibility.
▶ True

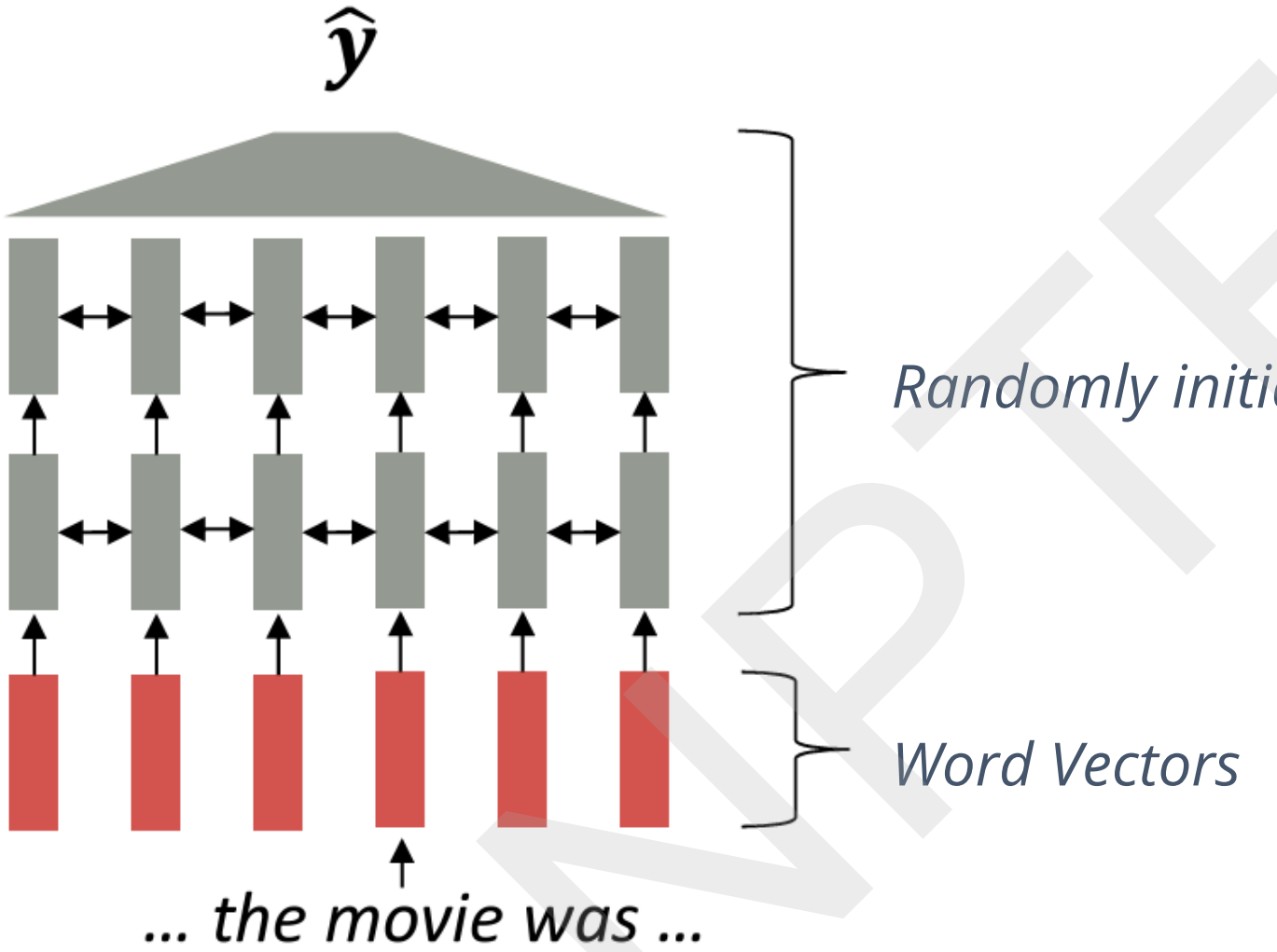
What does mTOR stands for?
▶ mammalian target of rapamycin

BIOMEDICAL DOMAIN

Several sentence and token level tasks

Very difficult to get sufficient labeled data for each task

Why is this an issue

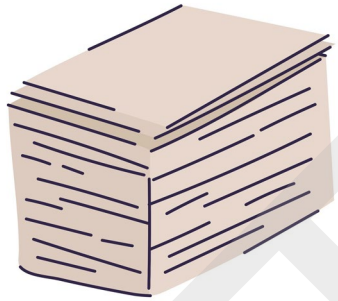


The training data we have for our downstream task (like question answering) must be sufficient to teach all contextual aspects of language.

Pretrain-then-Finetune (from 2018)

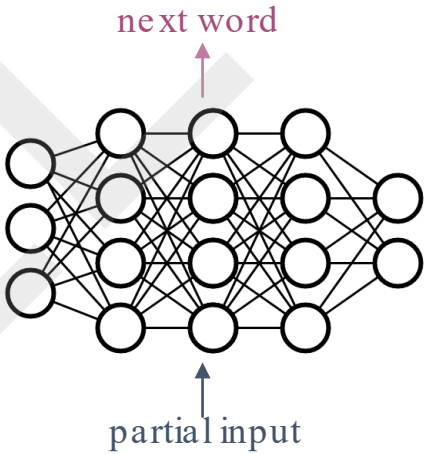
Stage 1:
Pretrain a model

neural network **starting from random weights**



text

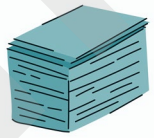
+



Objective: generate next or masked word
(does not require that people label the next word)

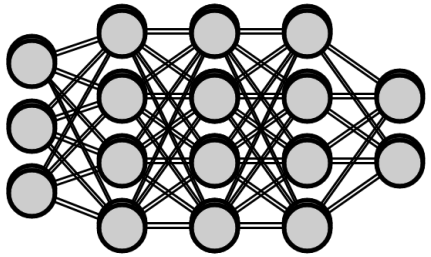
Stage 2:
Finetune the model

neural network **starting from pretrained weights**



text + **labels**

+



Objective: standard supervised training

https://utah-cs6340-nlp.notion.site/

Transfer Learning with Language Modeling

Language modelling is the popular choice of pretraining in NLP.

The key idea is to **train a neural network language model (on vast text corpora).** And **then, use it to transfer that knowledge to any target task in NLP** we care about.

This is a **very successful** idea in NLP:

- Supervised systems for various NLP tasks used to require **millions of examples** to learn tasks. NLP systems that use language models as a starting point, need much fewer - **thousands** of examples to do so!

https://rycolab.io/classes/llm-s24/

What does a model learn from pretraining?

- There are canines everywhere! One dog in the front room, and two dogs
- It wasn't just big it was enormous
- The author of "A Room of One's Own" is Virginia Woolf
- The doctor told me that he
- The square root of 4 is 2

https://web.stanford.edu/~jurafsky/slp3/

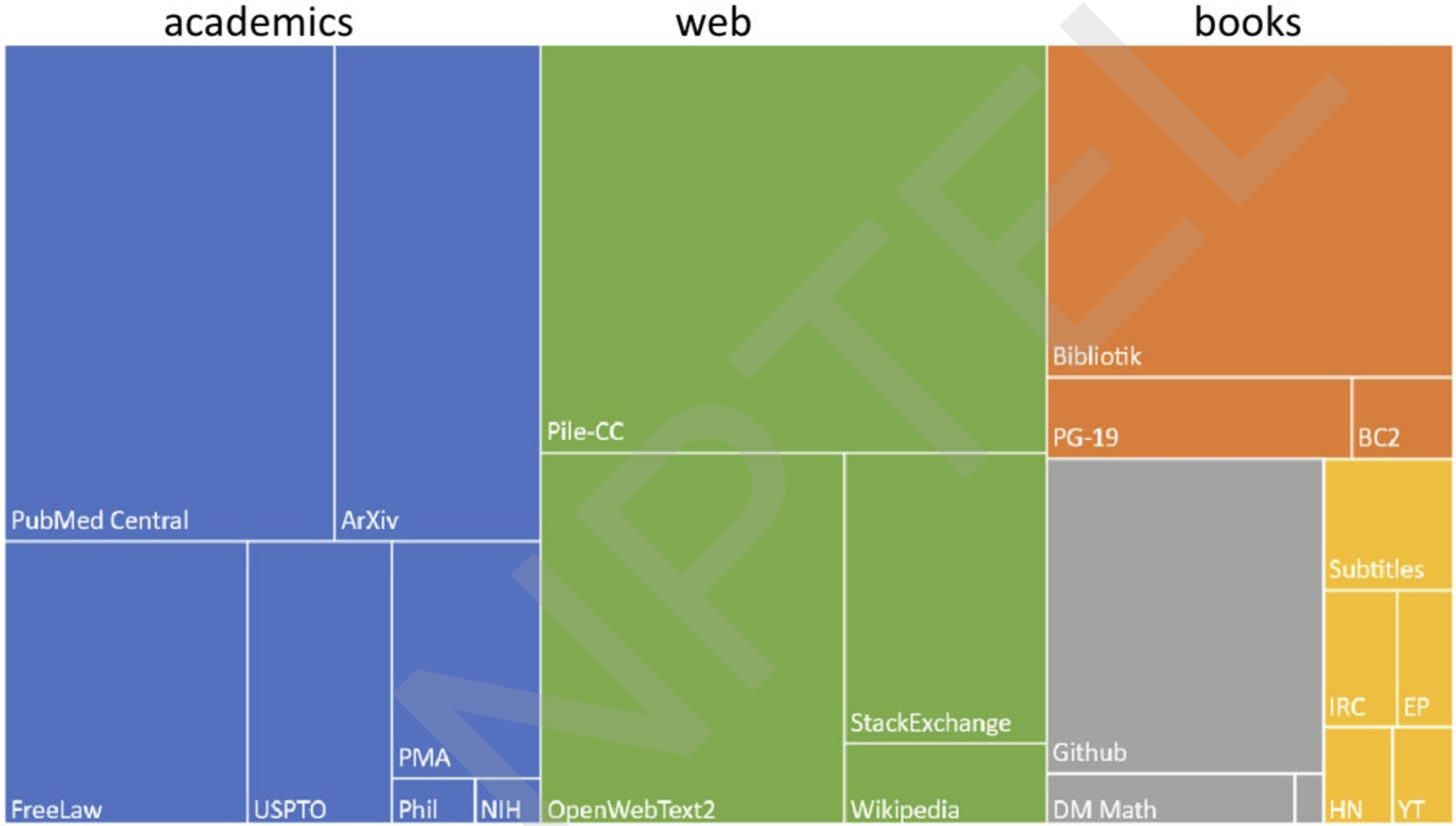
Big idea

- Text contains enormous amounts of knowledge
- Pretraining on lots of text with all that knowledge is what gives language models their ability to do so much

LLMs are mainly trained on the web

- Common crawl, snapshots of the entire web produced by the non- profit Common Crawl with billions of pages
- Colossal Clean Crawled Corpus (C4; Raffel et al. 2020), 156 billion tokens of English, filtered
  - What's in it? Mostly patent text documents, Wikipedia, and news sites

The Pile: a pretraining corpus



https://web.stanford.edu/~jurafsky/slp3/

dialog

Filtering for quality and safety

Quality is subjective

- Many LLMs attempt to match Wikipedia, books, particular websites
- Need to remove boilerplate, adult content
- Deduplication at many levels (URLs, documents, even lines)

Safety also subjective

- Toxicity detection is important, although that has mixed results
- Can mistakenly flag data written in dialects like African American English

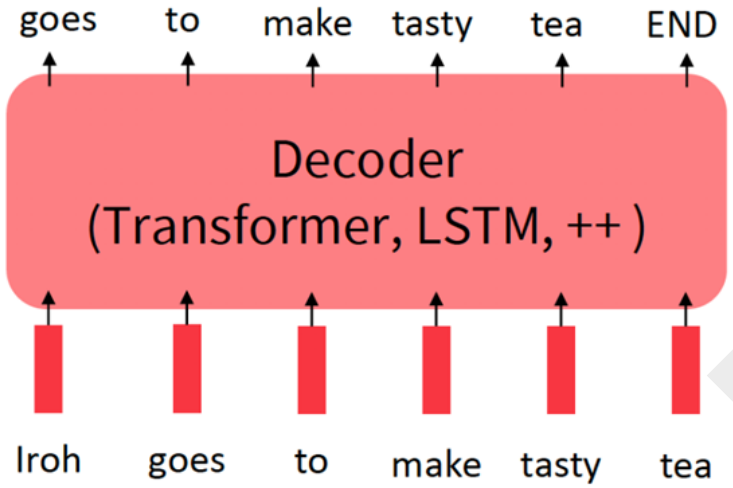
https://web.stanford.edu/~jurafsky/slp3/

The Pretraining / Fine Tuning paradigm

Pretraining can improve NLP applications by serving as parameter initialization.

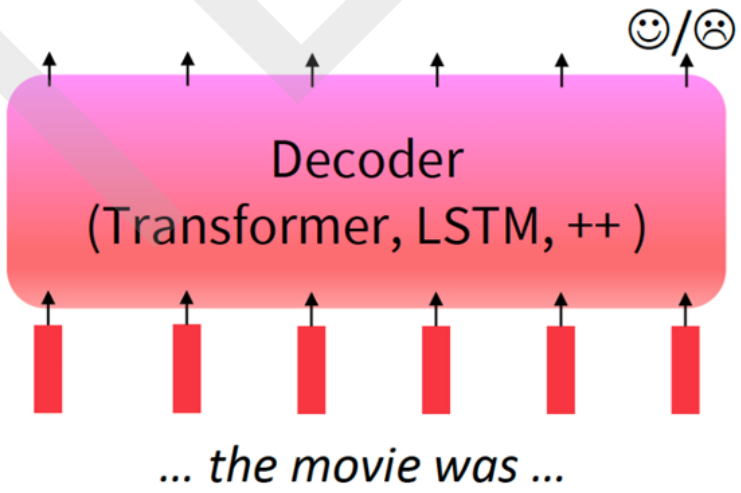
Step 1: Pretrain (on language modeling)

Lots of text; learn general things!



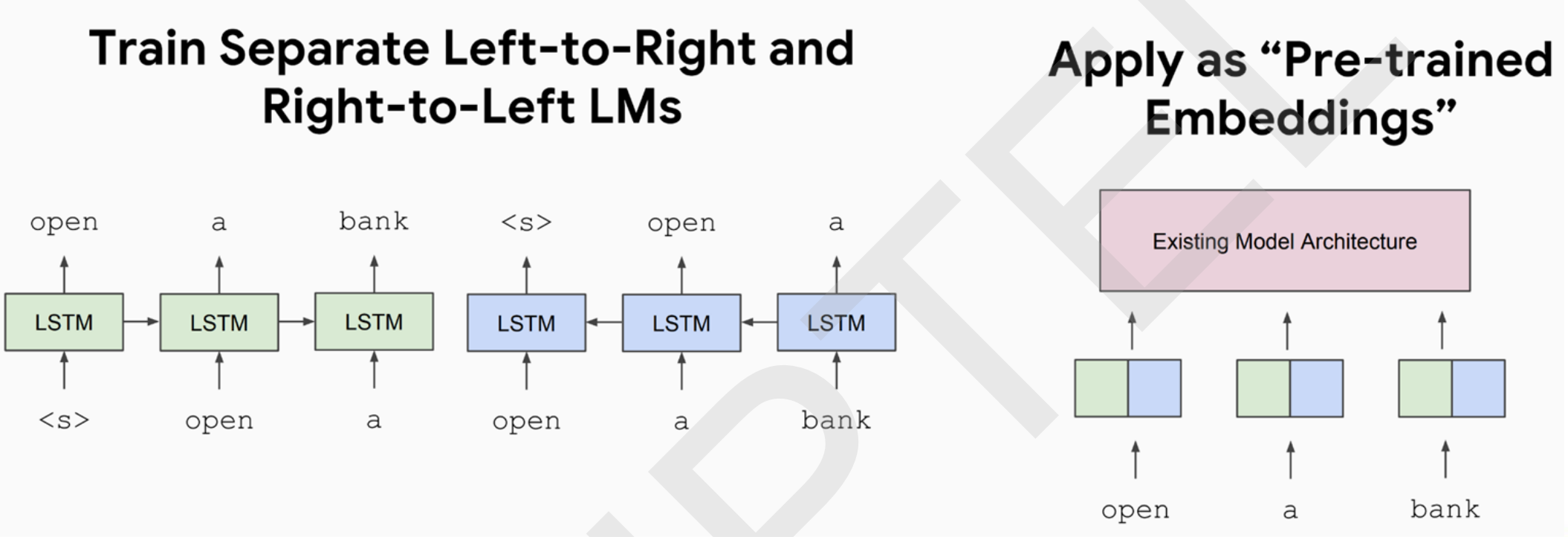
Step 2: Finetune (on your task)

Not many labels; adapt to the task!



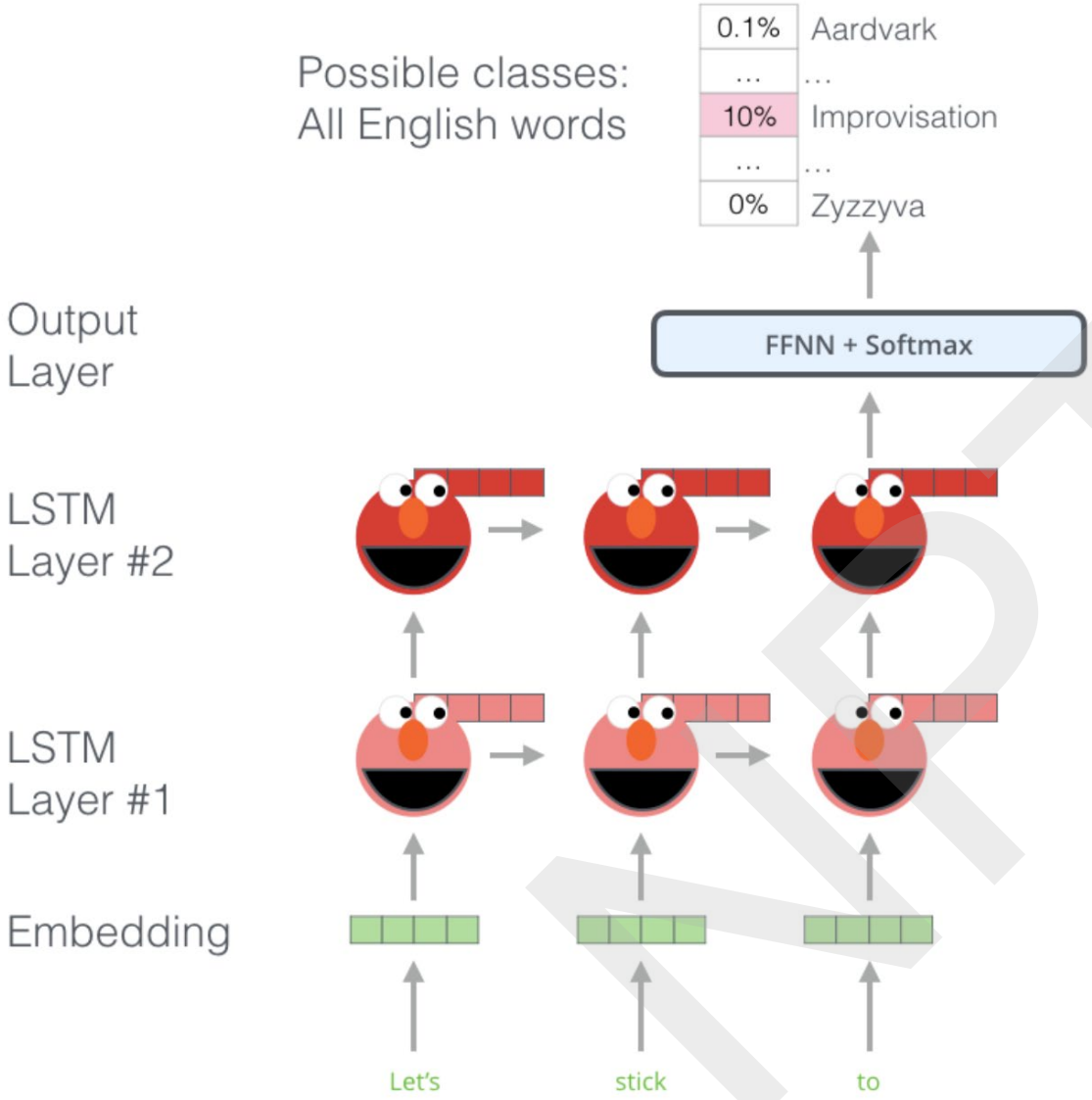
https://web.stanford.edu/class/cs224n/

Pretrained LSTMs: ELMo



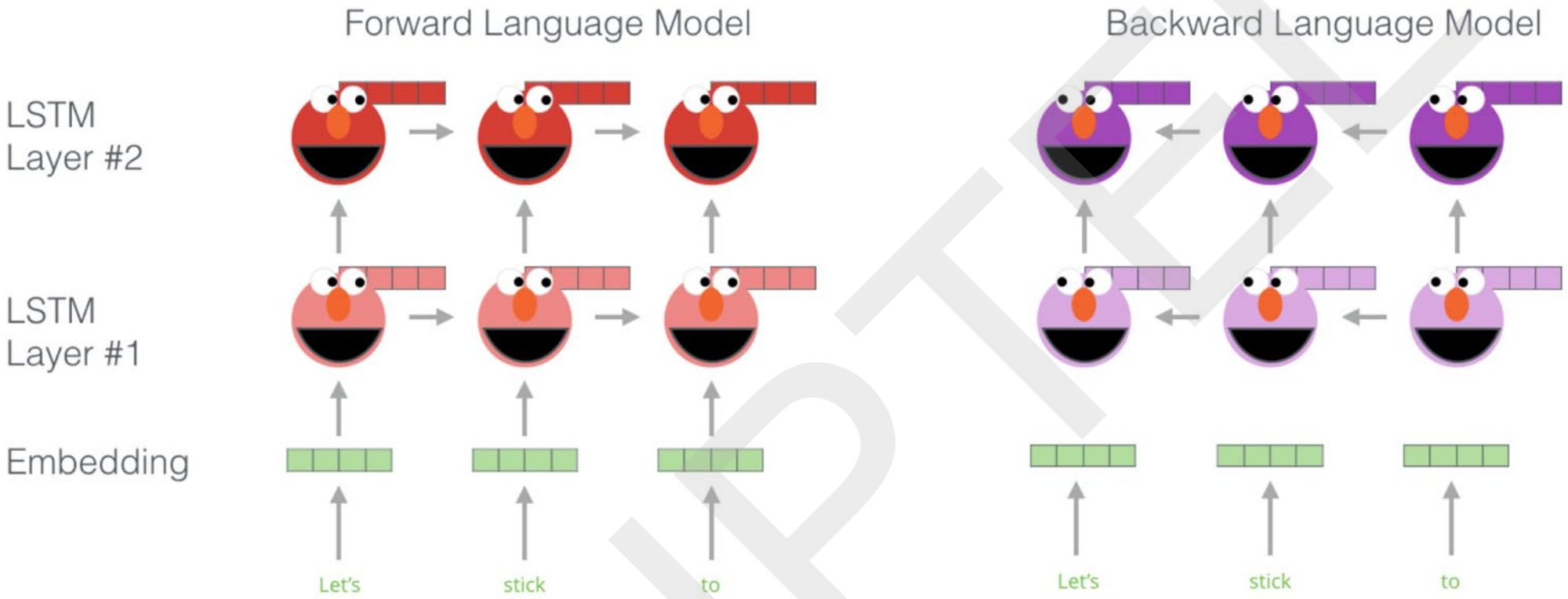
https://rycolab.io/classes/llm-s24/

ELMo: Secret



https://jalammar.github.io/illustrated-bert/

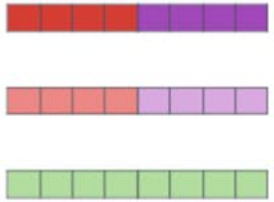
A bidirectional LM



https://jalammar.github.io/illustrated-bert/

ElMo: Obtaining the final embeddings

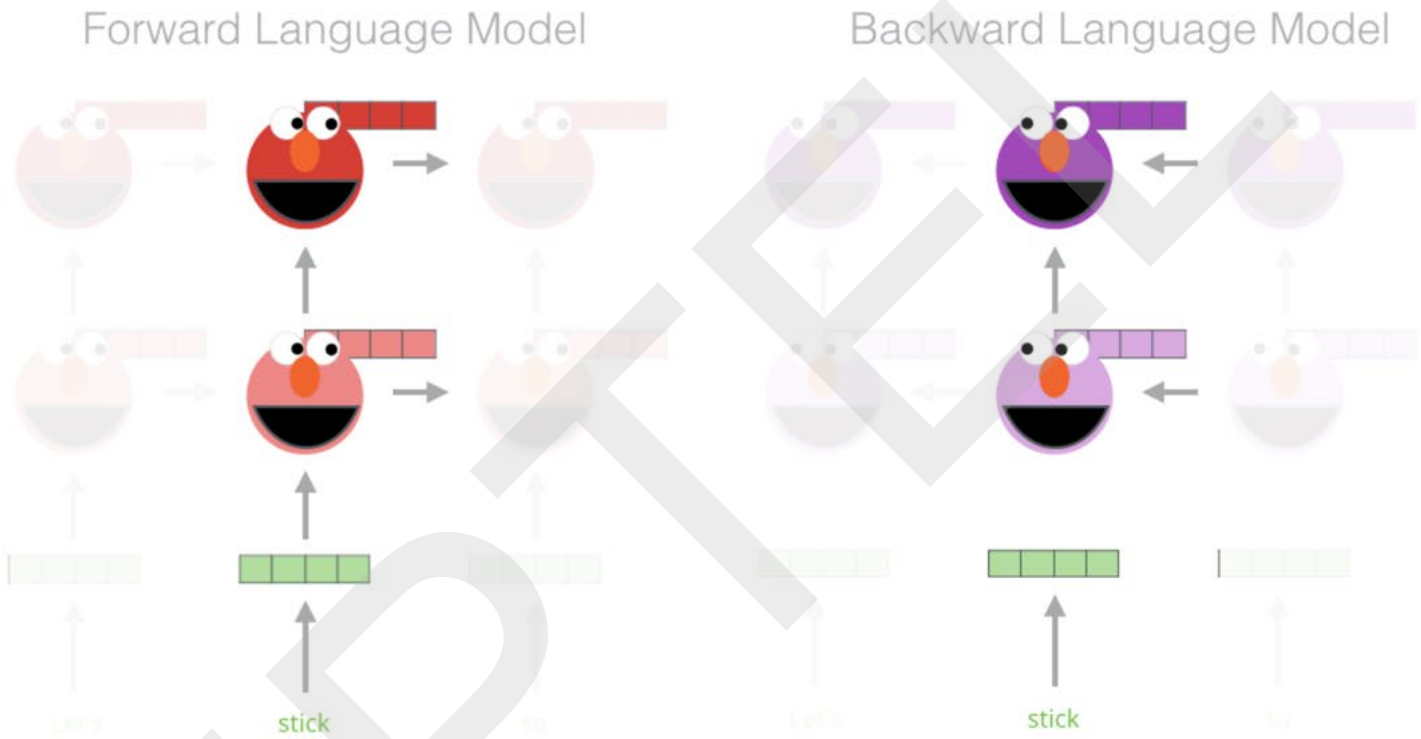
1- Concatenate hidden layers



2- Multiply each vector by a weight based on the task

3- Sum the (now weighted) vectors

ELMo embedding of "stick" for this task in this context



https://jalammar.github.io/illustrated-bert/

ELMo: More Details

Peters et al. (2018). Deep contextualized word representations. NAACL 2018.

- ELMo learns task-specific combination of biLM representations
- This is in contrast with just using the top layer of LSTM stack by TagLM

For an L-layer biLM, let R_k denote the set of representations for token t_k

$$
\begin{aligned}
R_k &= \{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} \mid j = 1, \ldots, L\} \\
&= \{\mathbf{h}_{k,j}^{LM} \mid j = 0, \ldots, L\},
\end{aligned}
$$

ELMo: More Details

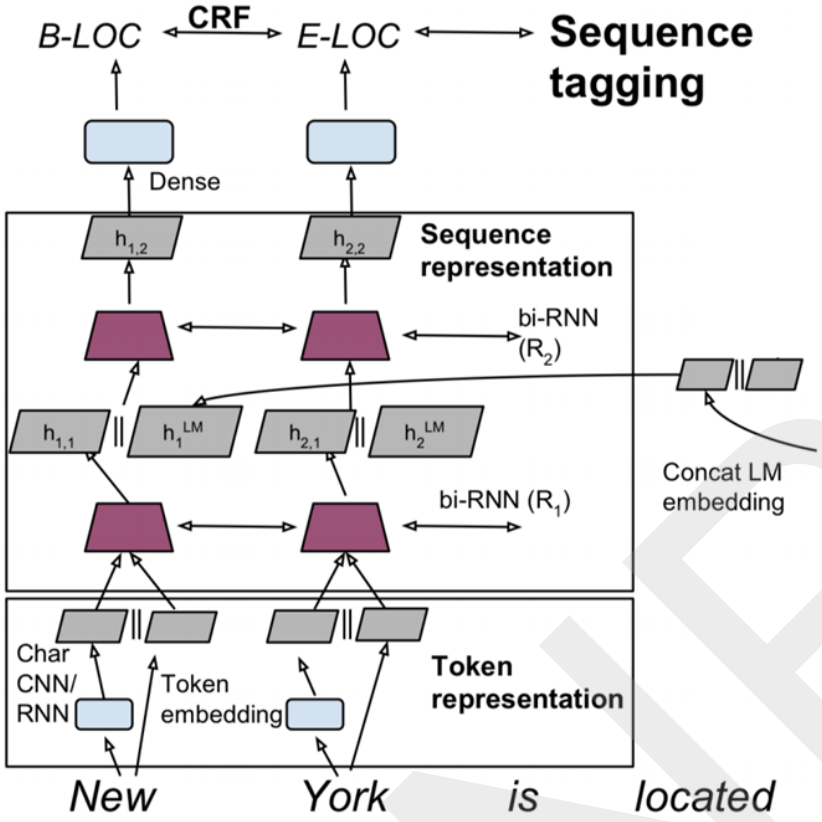
ELMo allows to obtain a task specific representation to each token t_k

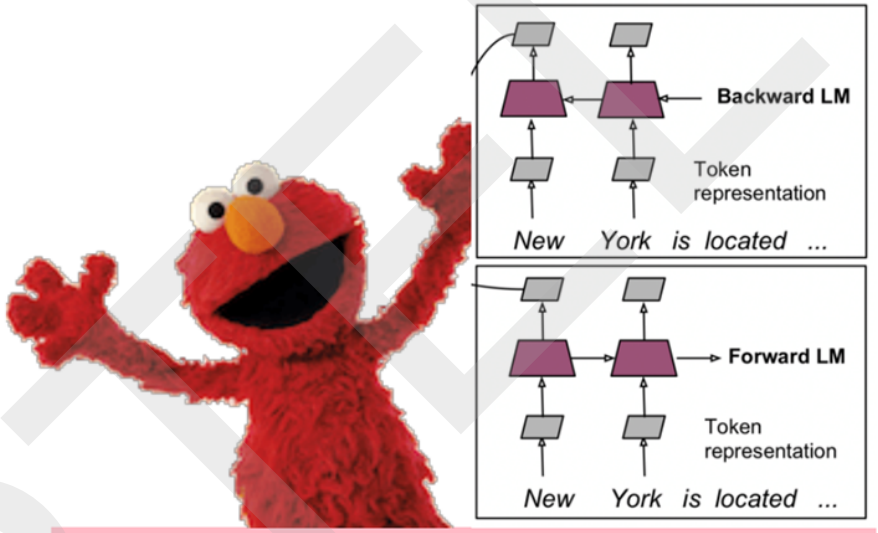
$$
\begin{aligned}
R_k &= \{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} \mid j = 1, \ldots, L\} \\
&= \{\mathbf{h}_{k,j}^{LM} \mid j = 0, \ldots, L\},
\end{aligned}
$$

$$
\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}
$$

- γ^{task} scales overall usefulness of ELMo to task
- s^{task} are softmax-normalized mixture model weights

Using ELMo for a task





ELMo representation:
A deep bidirectional neural LM

$$\mathbf{h}_{k,1} = [\overrightarrow{\mathbf{h}}_{k,1}; \overleftarrow{\mathbf{h}}_{k,1}; \mathbf{h}_k^{LM}].$$

Use learned, task-weighted average of (2) hidden layers

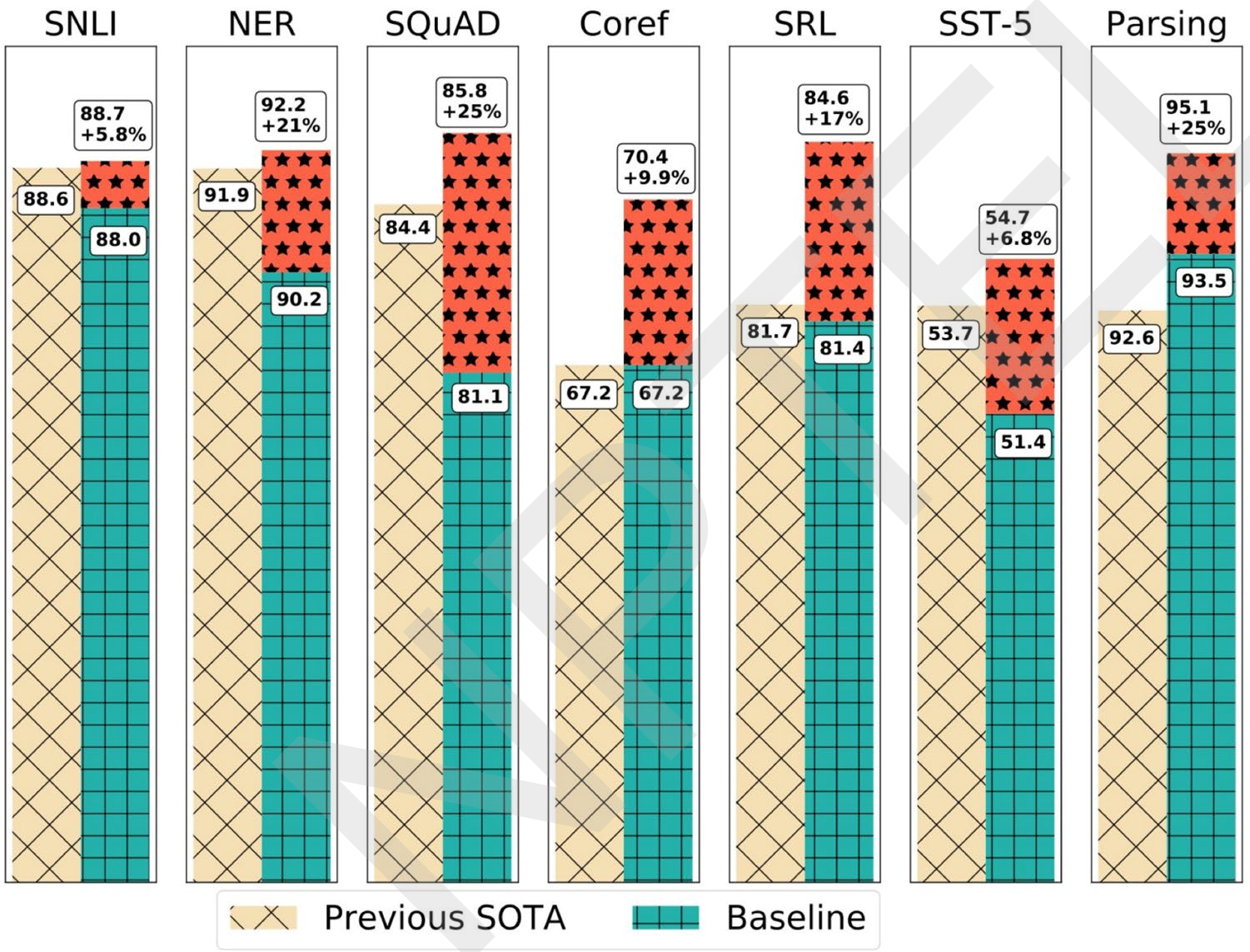
https://web.stanford.edu/class/cs224n/

ELMo: Weighting of layers

The two biLSTM NLM layers have differentiated uses/meanings

- Lower layer is better for lower-level syntax, etc. $\rightarrow$ Part-of-speech tagging, syntactic dependencies, NER
- Higher layer is better for higher-level semantics $\rightarrow$ Sentiment, Semantic role labeling, question answering, SNLI

ELMo: Results



https://rycolab.io/classes/llm-s24/

ElMo - How does the pretraining help with efficient fine-tuning?

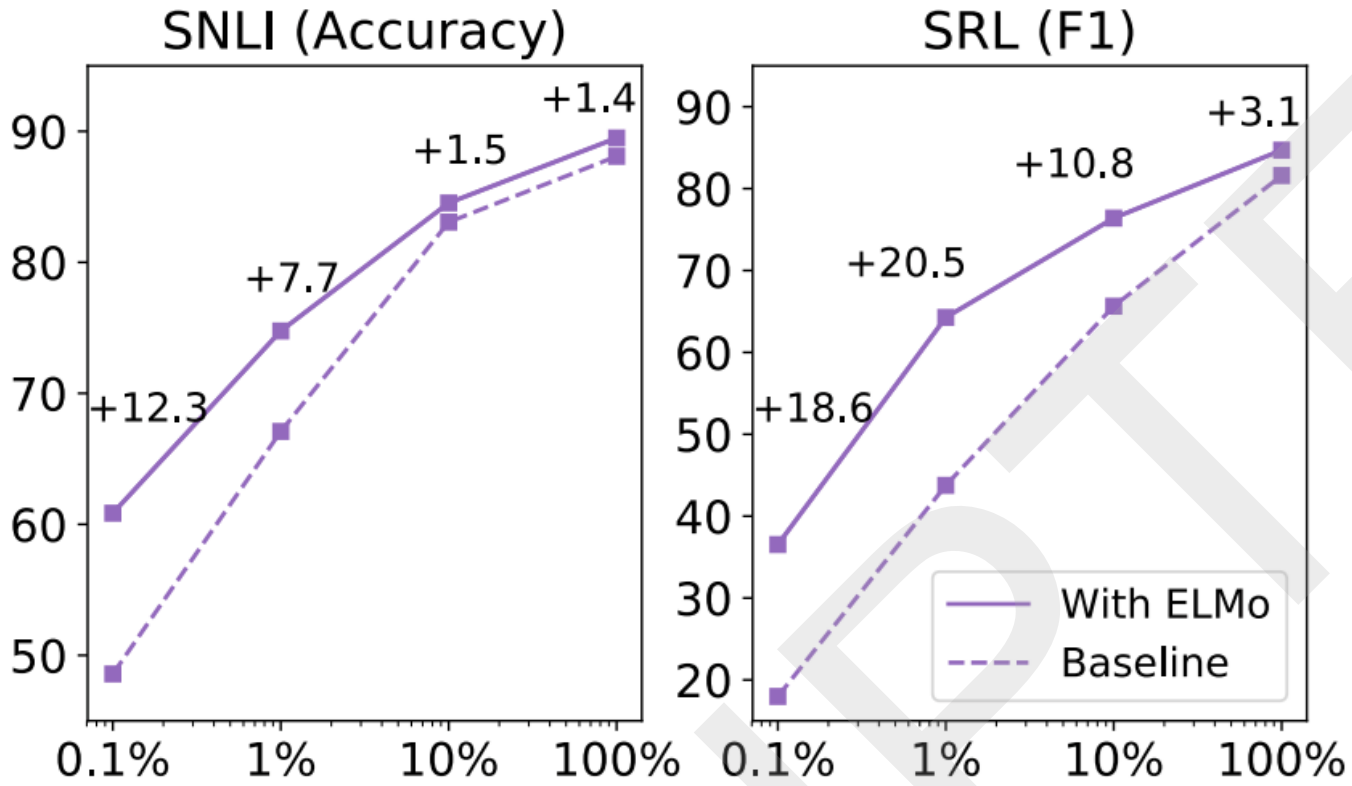
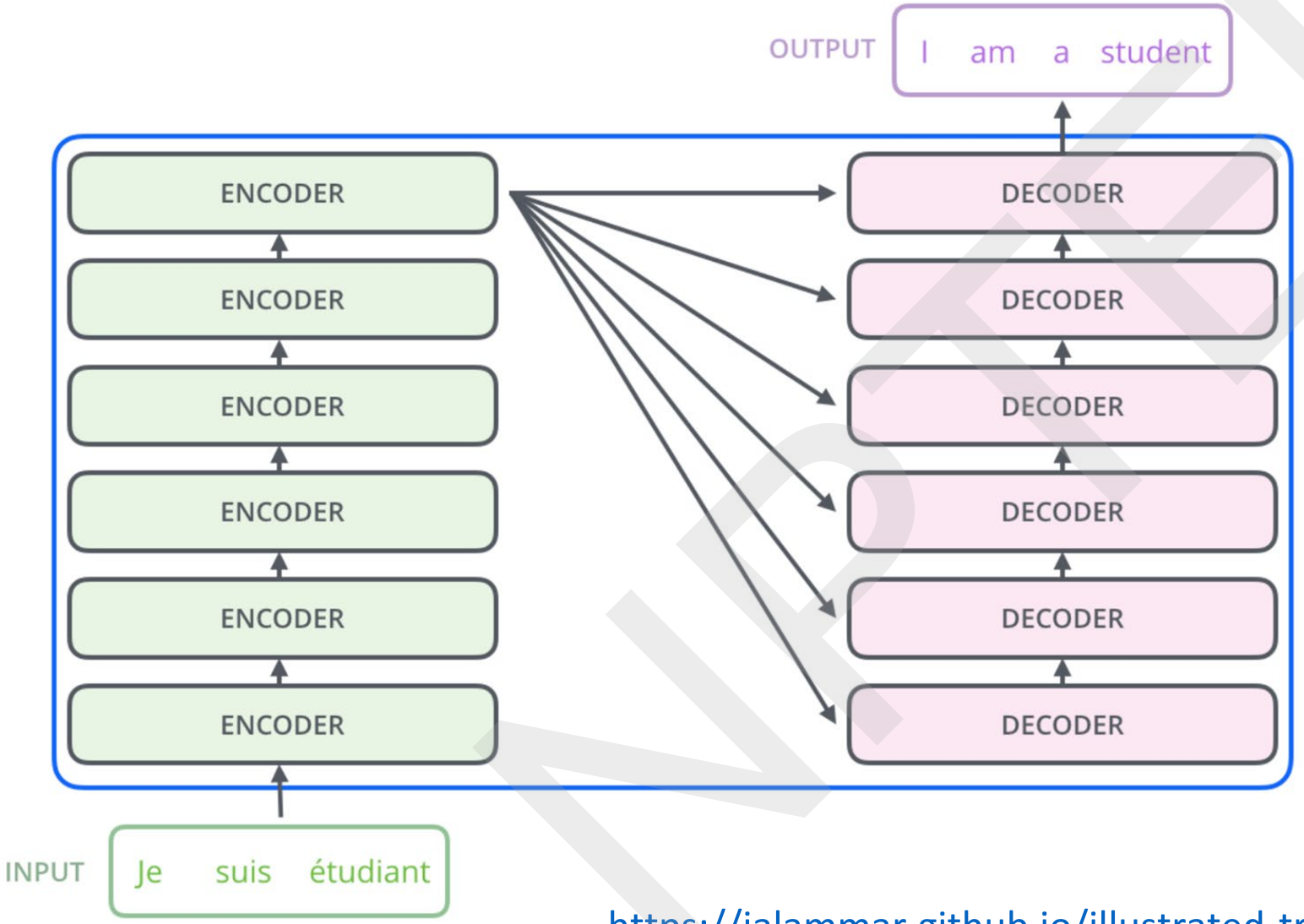


Figure 1: Comparison of baseline vs. ELMo performance for SNLI and SRL as the training set size is varied from 0.1% to 100%.

Try this problem

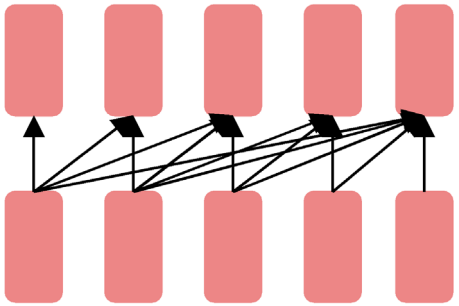
Suppose you trained a 2-layer (on top of word embeddings) forward LM as well as a 2-layer backward LM for ELMo. Assume that the 3-dimensional embedding for a word ***w*** be [0.2 0.4 -0.5], and the contextual representation for the first and second layer be [0.2 0.4 0.7] and [0.3 0.6 0.5] (forward LM), and [0.2 -0.4 0.7] and [0.3 0.6 -0.5] (backward LM), respectively. Assume that the softmax normalized mixture-model weights for the embedding, 1st and 2nd layers are 0.2, 0.3 and 0.5, respectively (for your particular task). What would be the ELMo representation that you can use for the word ***w***? What can you say about the task?

Using Transformers for Pretraining



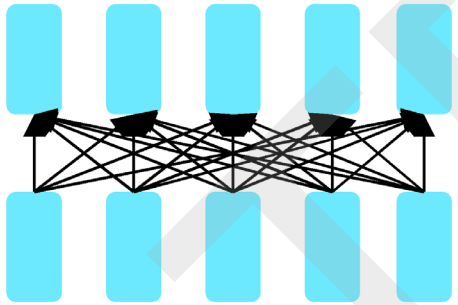
https://jalammar.github.io/illustrated-transformer/

Three architectures for large language models



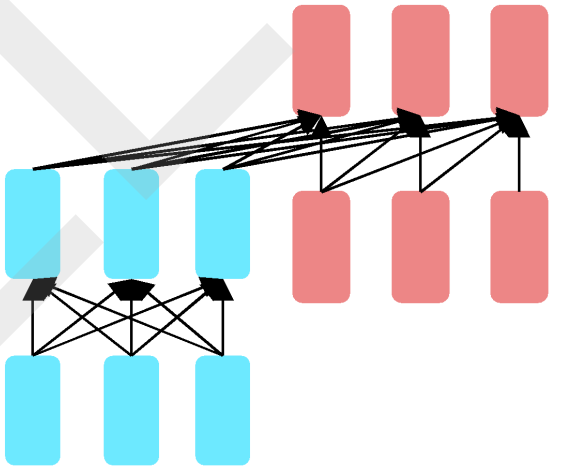
Decoders

GPT, Claude, Llama, Mixtral



Encoders

BERT family



Encoder-decoders

Flan-T5 BART

We will see how to pretrain for each of these architectures

https://web.stanford.edu/~jurafsky/slp3/

REFERENCES

Daniel Jurafsky and James H. Martin. 2024. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models, 3rd edition. Online manuscript released August 20, 2024. [Chapter 10]


THANK YOU

FREE ONLINE EDUCATION
swayam
शिक्षित भारत, उन्नत भारत

NPTEL

NPTEL ONLINE CERTIFICATION COURSES

DEEP LEARNING FOR NATURAL LANGUAGE PROCESSING

Lecture 27 : Pretraining Transformer Encoder

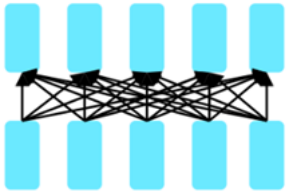
PROF. PAWAN GOYAL
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

CONCEPTS COVERED

- Pretraining BERT: Masked LM
- Contextual representation from BERT
- Fine-tuning BERT for various paradigms

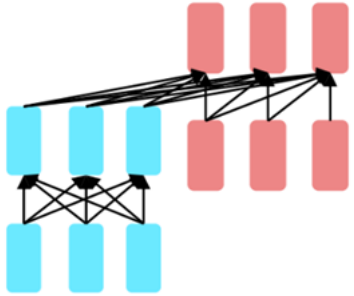
Pretraining for three types of architectures

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



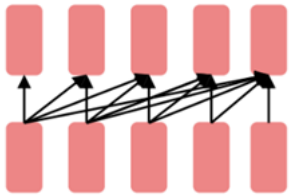
Encoders

- Gets bidirectional context – can condition on future!
- How do we train them to build strong representations?



Encoder-Decoders

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

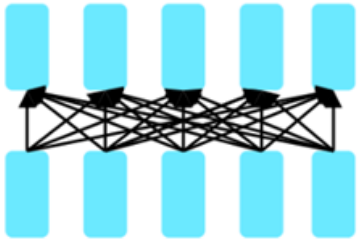


Decoders

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words

https://web.stanford.edu/class/cs224n/

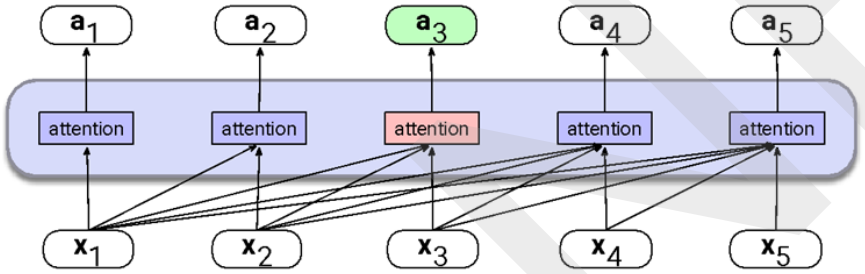
Pretraining Encoders



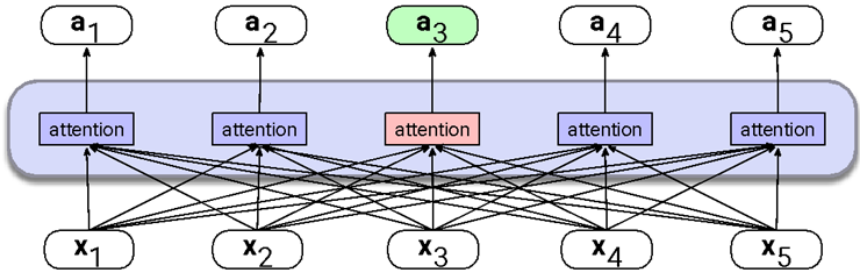
Encoders

- Gets bidirectional context – can condition on future!
- How do we train them to build strong representations?

- map sequences of input embeddings (x1,...,x*n*)
- to sequences of output embeddings of the same length (h1,...,h*n*),
- where the output vectors have been contextualized using information from the entire input sequence.



a) A causal self-attention layer



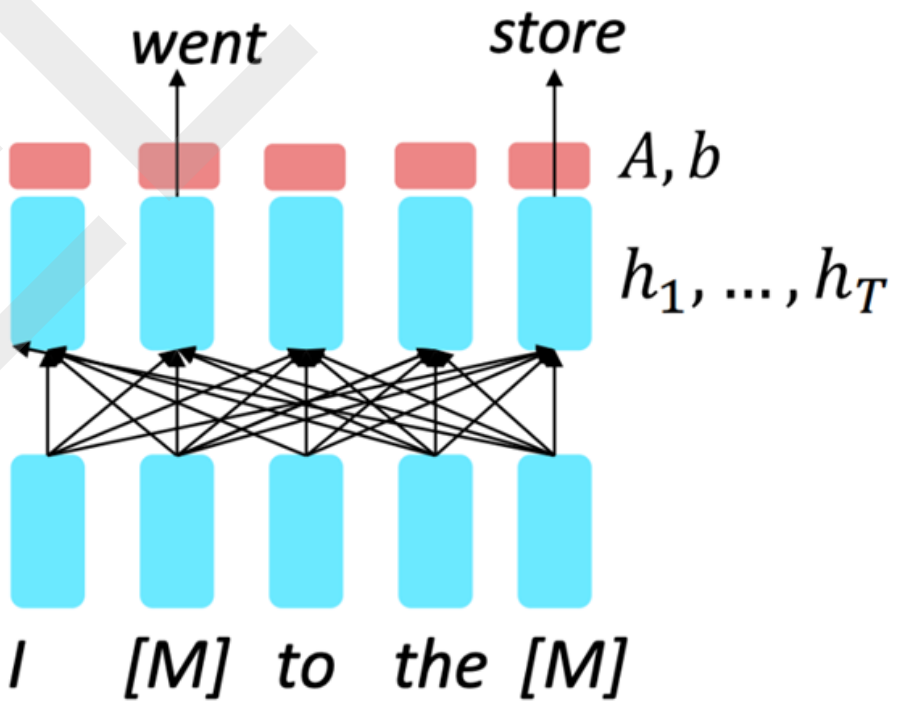
b) A bidirectional self-attention layer

https://web.stanford.edu/~jurafsky/slp3/

Solution: Use Masks

Idea: replace some fraction of words in the input with a special [MASK] token; predict these words.

$$h_1, \ldots, h_T = \text{Encoder}(w_1, \ldots, w_T)$$
$$y_i \sim Aw_i + b$$



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

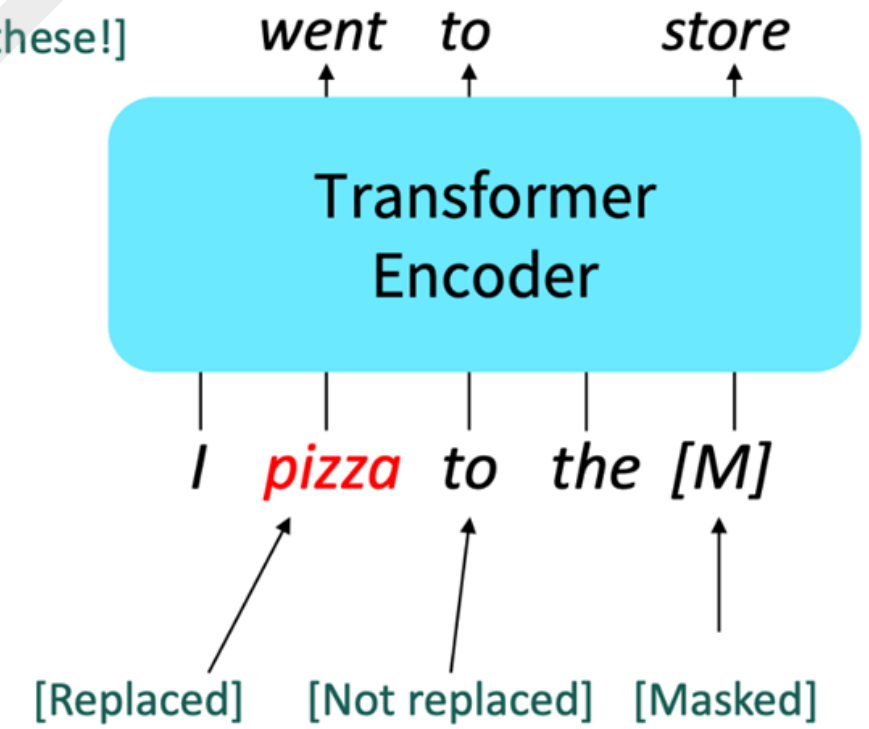
https://web.stanford.edu/class/cs224n/

BERT: Bidirectional Encoder Representations from Transformers

Devlin et al., 2018 proposed the "Masked LM" objective and **released the weights of a pretrained Transformer**, a model they labeled BERT.

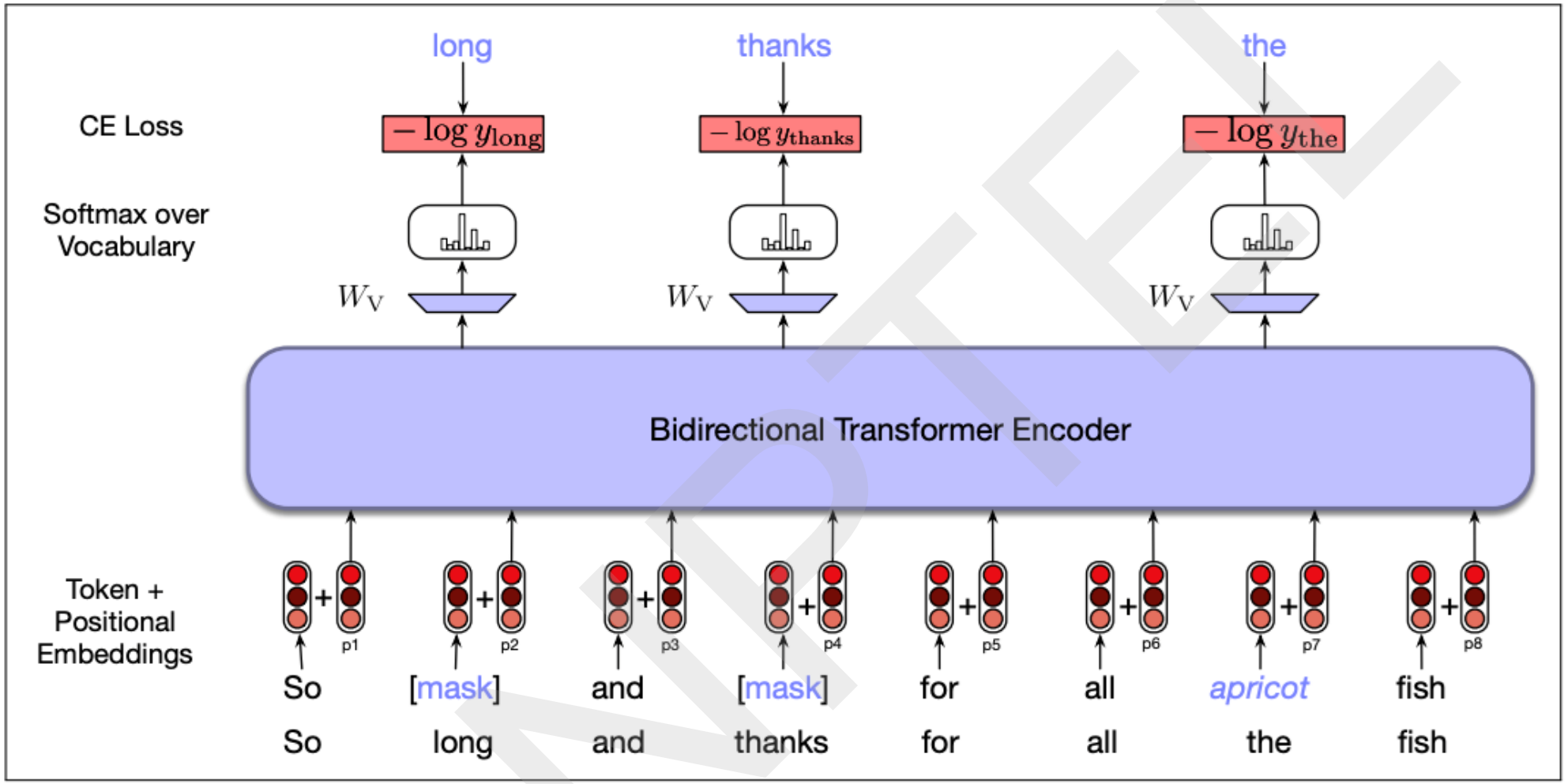
Some more details about Masked LM for BERT:

- Predict a random 15% of (sub)word tokens.
  - Replace input word with [MASK] 80% of the time
  - Replace input word with a random token 10% of the time
  - Leave input word unchanged 10% of the time (but still predict it!)
- 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!)



https://web.stanford.edu/class/cs224n/

BERT: another view



$$y_i = softmax(W_V h_i), W_V \in R^{|V| \times d_h}, h_i \in R^{d_h}$$

https://web.stanford.edu/~jurafsky/slp3/

Next Sentence Prediction

Given 2 sentences the model predicts if they are a real pair of adjacent sentences from the training corpus or a pair of unrelated sentences.

BERT introduces two special tokens

- [CLS] is prepended to the input sentence pair,
- [SEP] is placed between the sentences, and also after second sentence

And two more special embeddings

- [1st segment] and [2nd segment]
- These are added to the input embedding and positional embedding

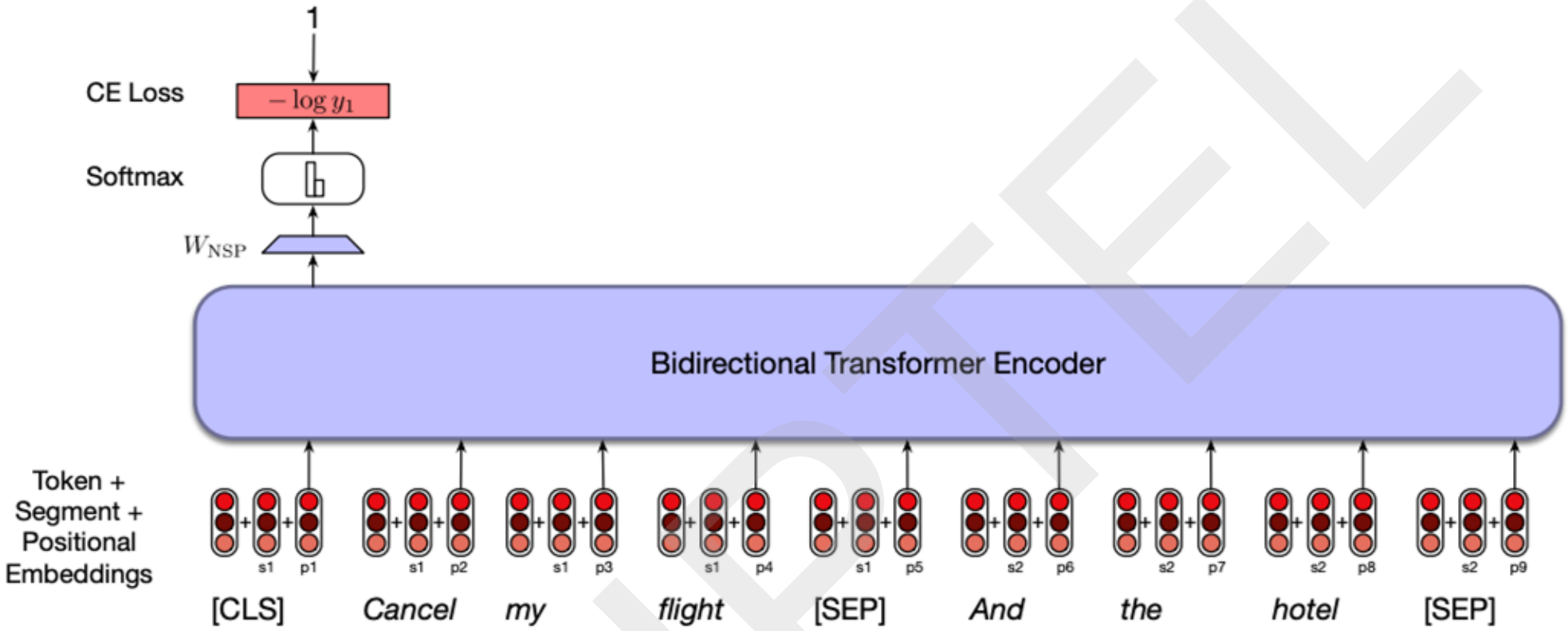
h^L_{CLS} from the final layer [CLS] token is input to classifier head (weights W_{NSP})

that predicts two classes:.

$$y_i = \text{softmax}(h^L_{CLS} W_{NSP})$$

https://web.stanford.edu/~jurafsky/slp3/

BERT: Next Sentence Prediction



$$y = softmax(W_{NSP}C), W_{NSP} \in R^{2 \times d_h}, C \in R^{d_h}$$

https://web.stanford.edu/~jurafsky/slp3/

BERT: Next Sentence Prediction

Why NSP?

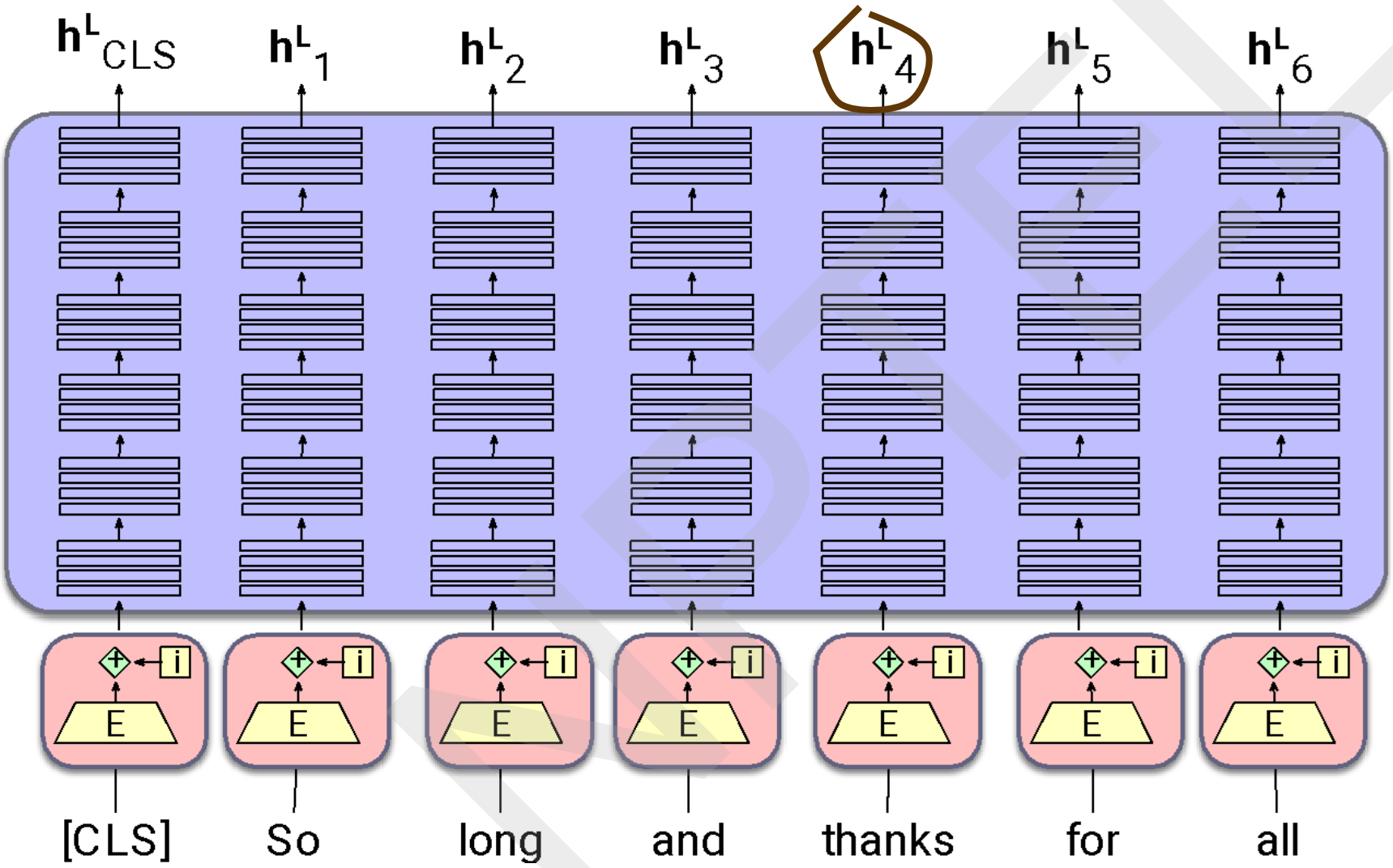
- Masking focuses on predicting words from surrounding contexts so as to produce effective word-level representations.
- Many applications require relationship between two sentences, e.g.,
  - paraphrase detection (detecting if two sentences have similar meanings),
  - entailment (detecting if the meanings of two sentences entail or contradict each other)
  - discourse coherence (deciding if two neighboring sentences form a coherent discourse)

More Details about BERT

- 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.”

https://web.stanford.edu/class/cs224n/

Contextual Embeddings to represent words



https://web.stanford.edu/~jurafsky/slp3/

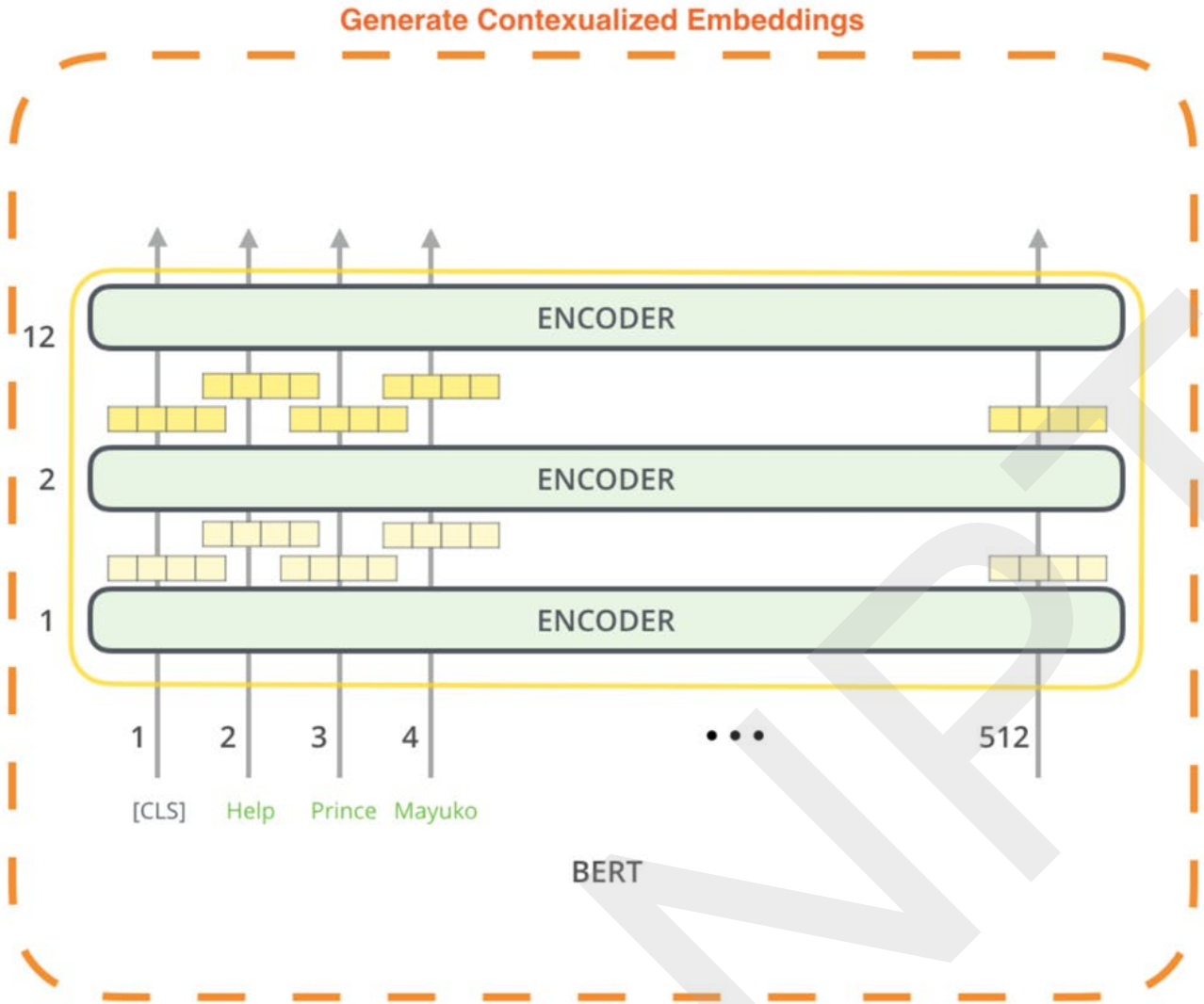
Static vs Contextual Embeddings

Static embeddings represent **word types** (dictionary entries)

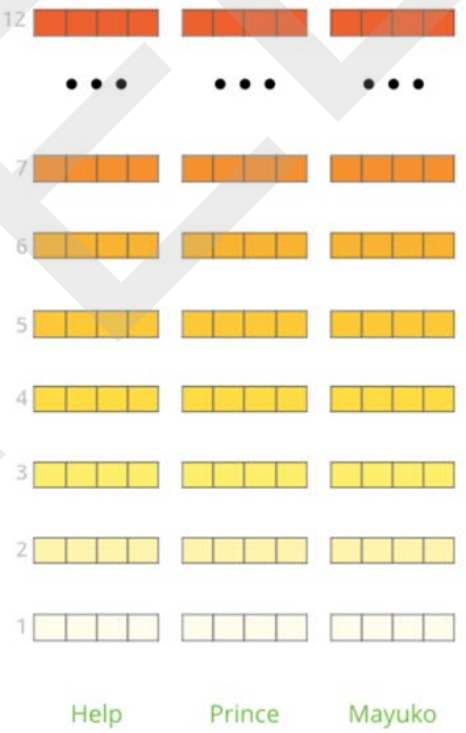
Contextual embeddings represent **word instances** (one for each time the word occurs in any context/sentence)

https://web.stanford.edu/~jurafsky/slp3/

Using BERT to create contextualized word embedding



The output of each encoder layer along each token's path can be used as a feature representing that token.

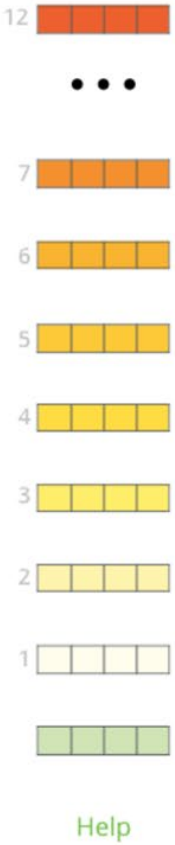


But which one should we use?

https://web.stanford.edu/class/cs224n/

Using BERT to create contextualized word embedding

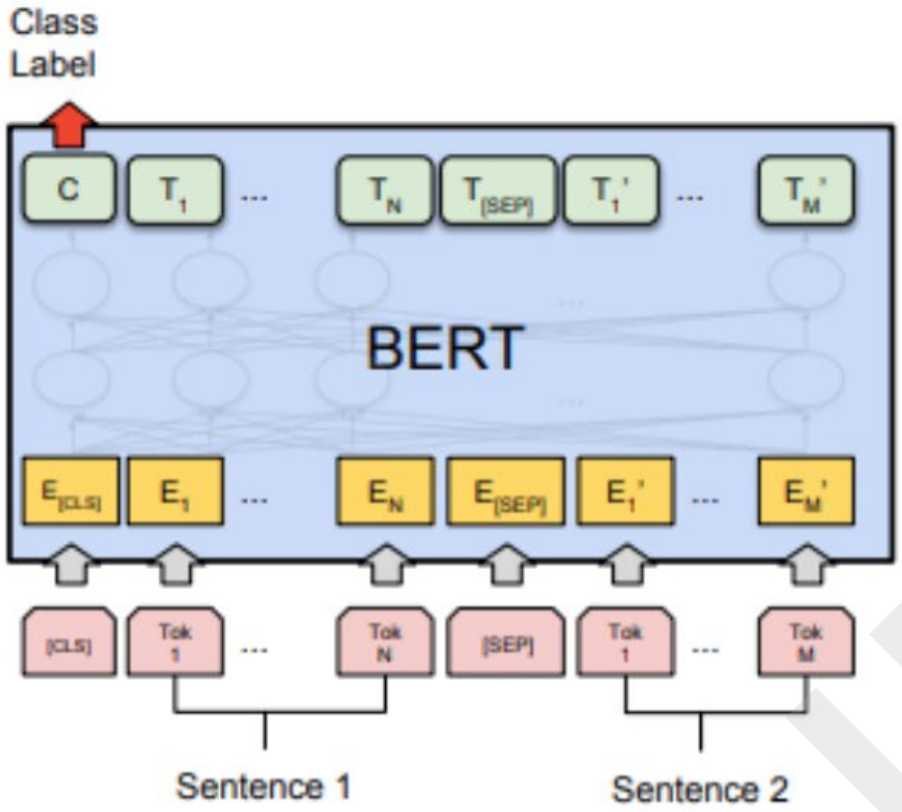
What is the best contextualized embedding for "Help" in that context?
For named-entity recognition task CoNLL-2003 NER



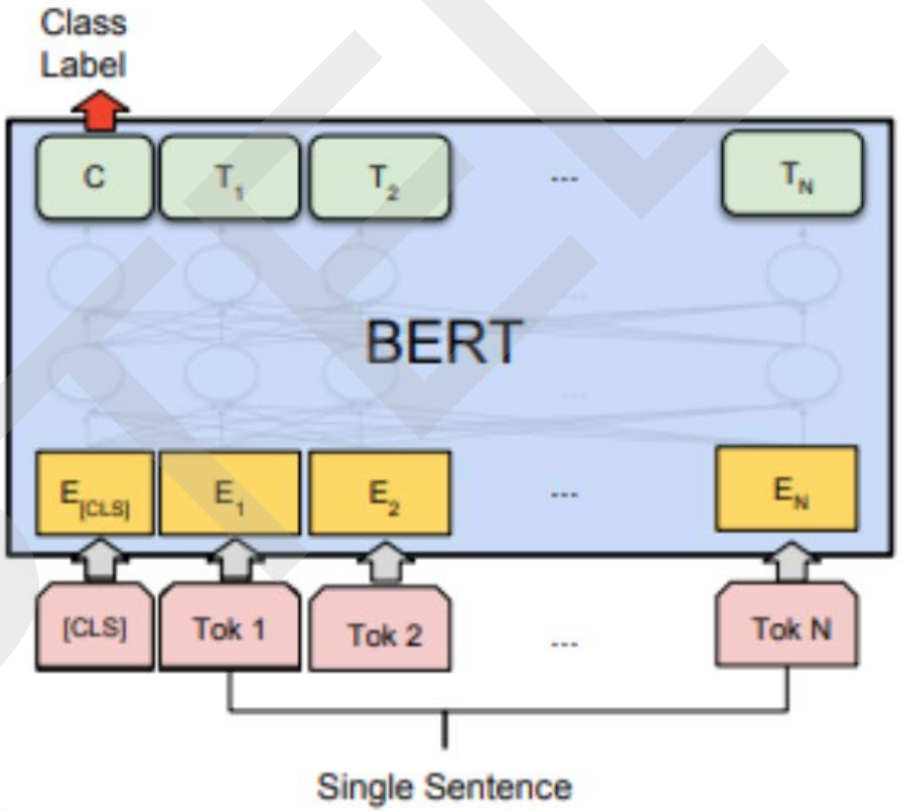
		Dev F1 Score
First Layer	Embedding	91.0
Last Hidden Layer	12	94.9
Sum All 12 Layers	12 + ... + 2 + 1 =	95.5
Second-to-Last Hidden Layer	11	95.6
Sum Last Four Hidden	12 + 11 + 10 + 9 =	95.9
Concat Last Four Hidden	9 10 11 12	96.1

https://web.stanford.edu/class/cs224n/

Using BERT for different tasks

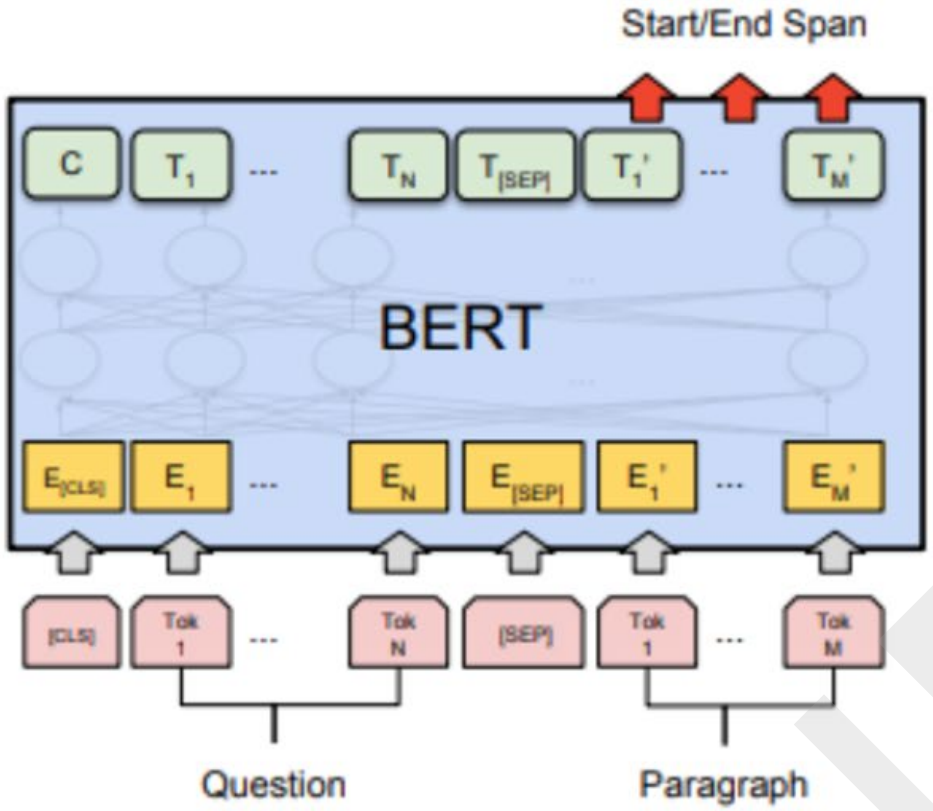


(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

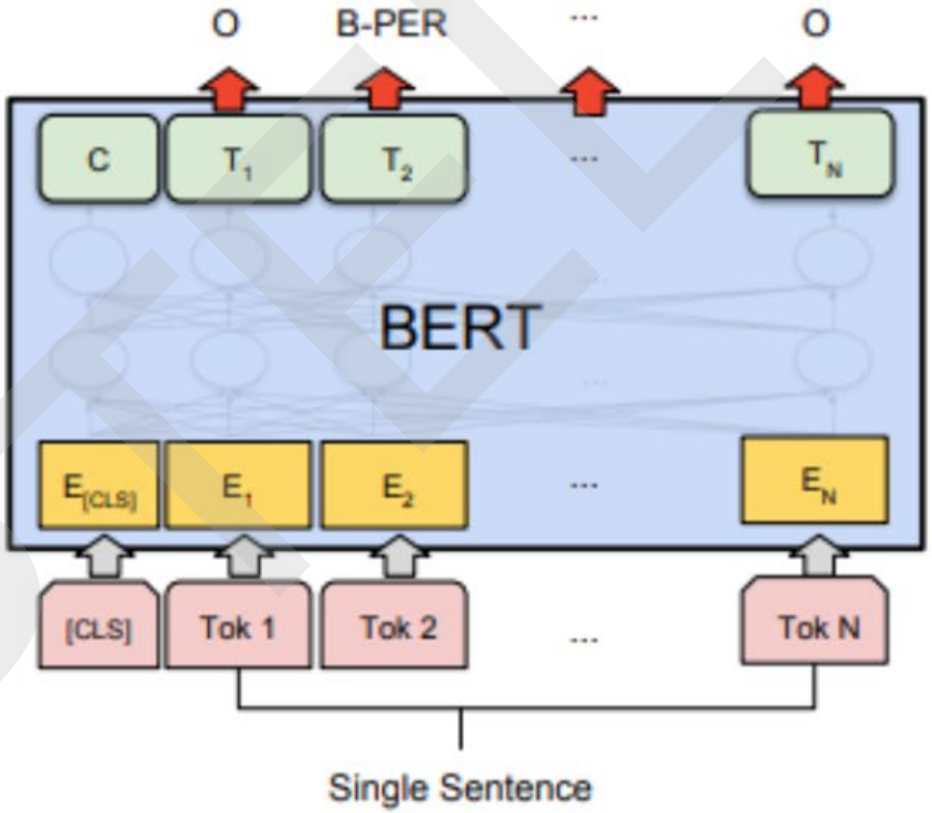


(b) Single Sentence Classification Tasks: SST-2, CoLA

Using BERT for different tasks



(c) Question Answering Tasks: SQuAD v1.1



(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Transfer Learning through Fine-tuning

- The power of pretrained language models lies in their ability to extract generalizations from large amounts of text
- To make practical use of these generalizations, we need to create interfaces from these models to downstream applications through a process called *fine-tuning*.
- Fine-tuning facilitates the creation of applications on top of pretrained models through the addition of a small set of application-specific parameters.
- The fine-tuning process consists of using labeled data from the application to train these additional application-specific parameters.
- Typically, this training will either freeze or make only minimal adjustments to the pretrained language model parameters.

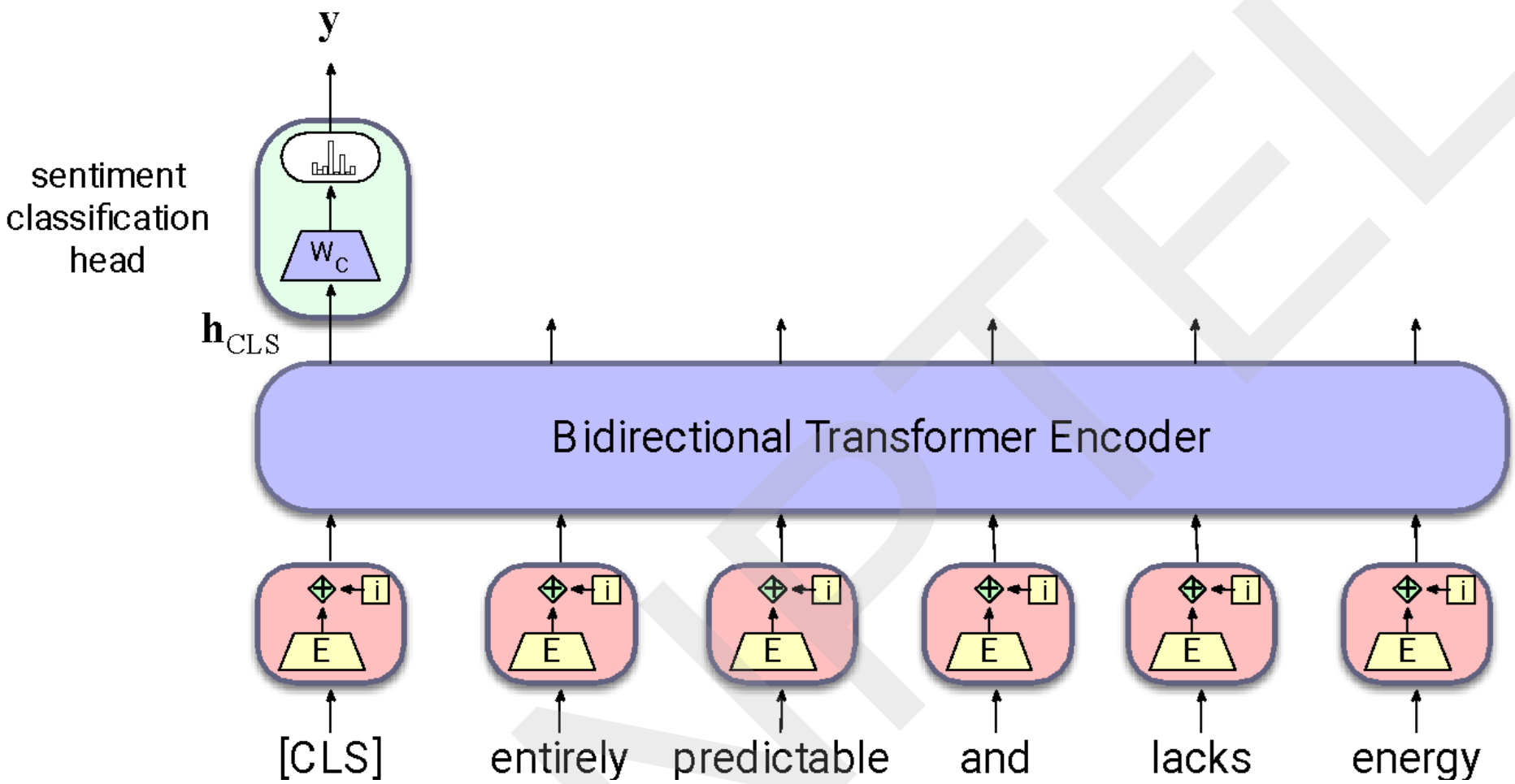
Fine Tuning for Sequence Classification

- With RNNs, we used the hidden layer associated with the final input element to stand for the entire sequence. In BERT, the [CLS] token plays the role of sentence embedding.
- This unique token is added to the vocabulary and is prepended to the start of all input sequences, both during pretraining and encoding.
- The output vector in the final layer of the model for the [CLS] input serves as the input to classifier head a classifier head.

Fine Tuning for Sequence Classification

- With RNNs, we used the hidden layer associated with the final input element to stand for the entire sequence. In BERT, the [CLS] token plays the role of sentence embedding.
- This unique token is added to the vocabulary and is prepended to the start of all input sequences, both during pretraining and encoding.
- The output vector $C \in R^{d_h}$ in the final layer of the model for the [CLS] input serves as the input to classifier head a classifier head.
- The only new parameters introduced during fine-tuning are classification layer weights $W_C \in R^{K \times d_h}$, where K is the number of labels.

Fine Tuning for Sequence Classification



https://web.stanford.edu/~jurafsky/slp3/

Pairwise Sequence Classification

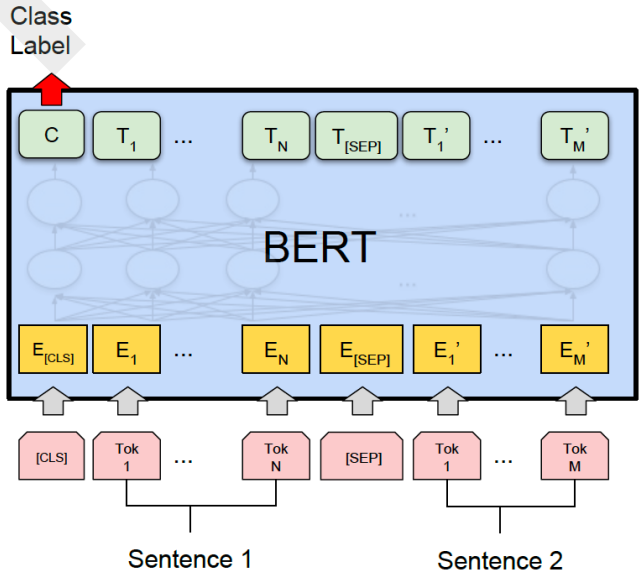
Example: multiNLI

- Pairs of sentences are given one of 3 labels: entails, contradicts and neutral.
- These labels describe a relationship between the meaning of the first sentence (the premise) and the second sentence (the hypothesis).

- `Neutral`
  - a: Jon walked back to the town to the smithy.
  - b: Jon traveled back to his hometown.
- `Contradicts`
  - a: Tourist Information offices can be very helpful.
  - b: Tourist Information offices are never of any help.
- `Entails`
  - a: I'm confused.
  - b: Not all of it is very clear to me.

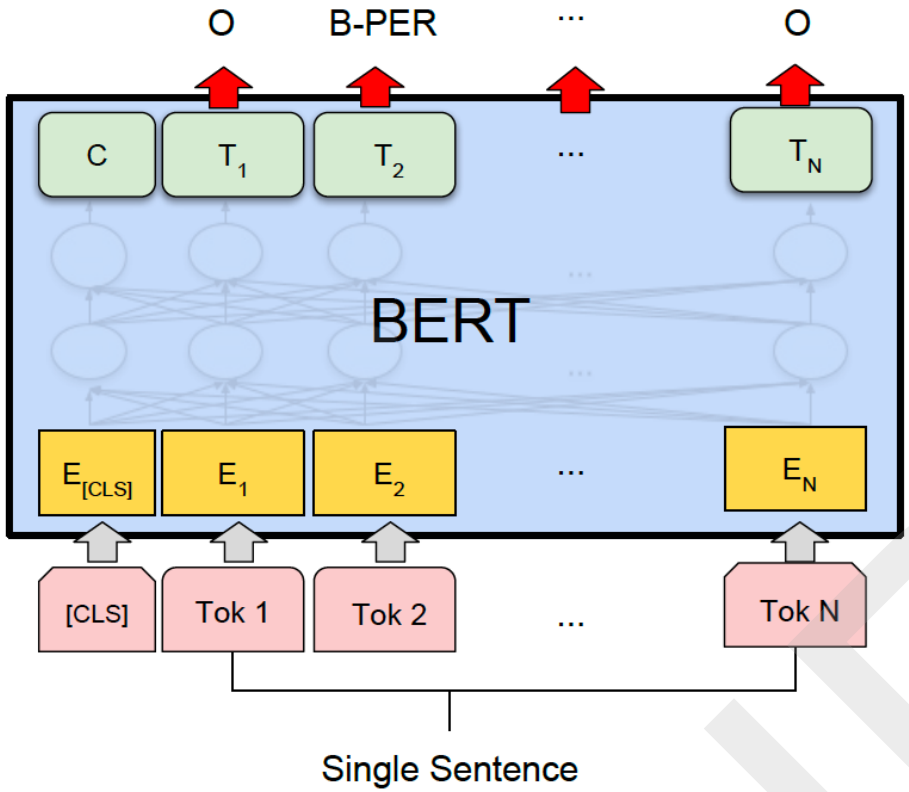
Pair-wise Sequence Classification

- As with NSP training, the two inputs are separated by a [SEP] token.
- As with sequence classification, the output vector associated with the prepended [CLS] token represents the model's view of the input pair.
- This vector C provides the input to a three-way classifier that can be trained on the MultiNLI training corpus.



(a) Sentence Pair Classification Tasks:

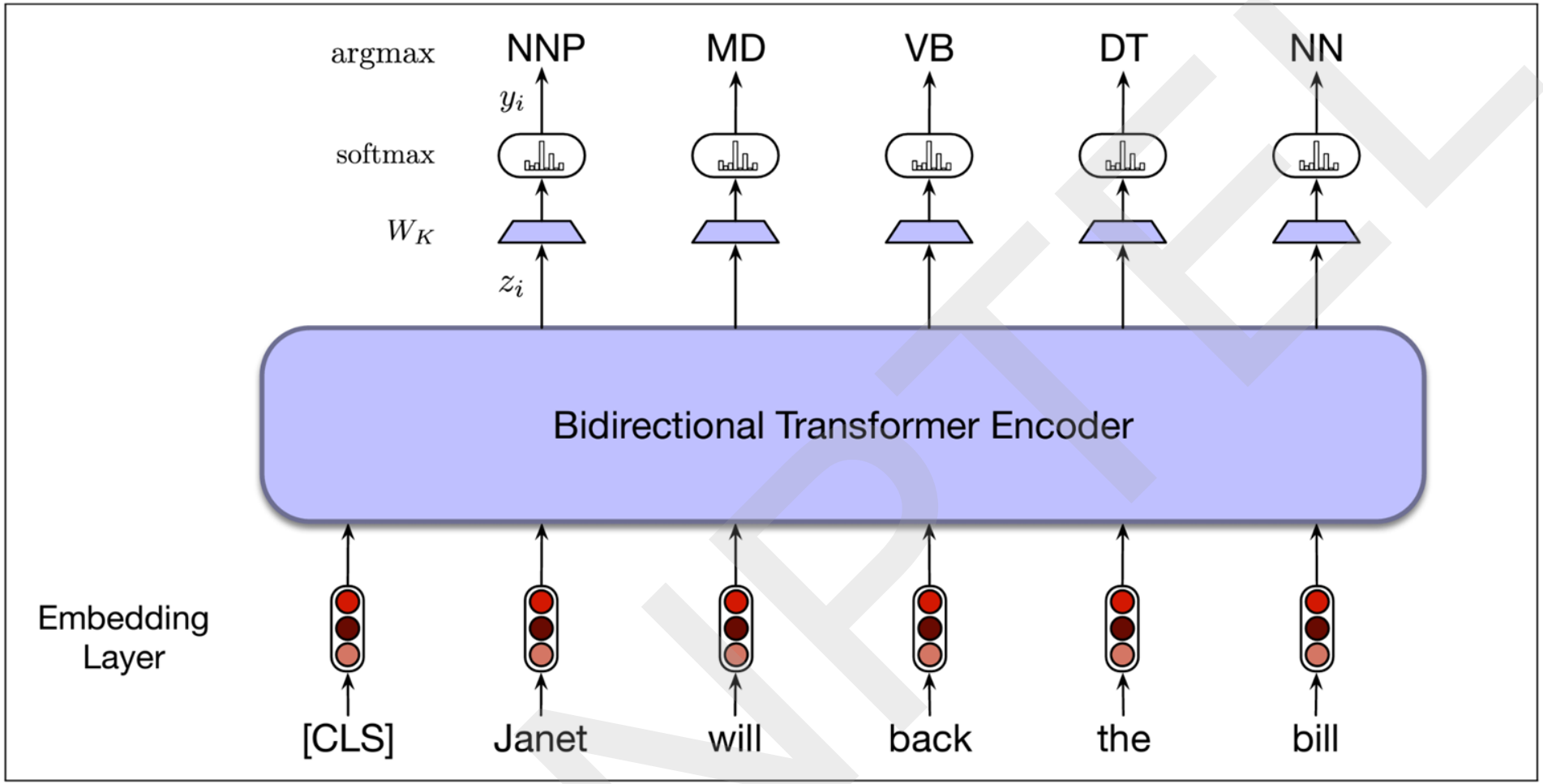
Sequence Labeling



(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

- Here, the final output vector corresponding to *each input token* is passed to a classifier that produces a softmax distribution over the possible set of tags.
- The set of weights to be learned for this additional layer is $W_K \in R^{k \times d_h}$, where k is the number of possible tags for the task.

POS Tagging



https://web.stanford.edu/~jurafsky/slp3/

Named Entity Recognition and BIO Scheme

Supervised training data for tasks like named entity recognition (NER) is typically in the form of BIO tags associated with text segmented at the word level. For example:

[$_{LOC}$ **Mt. Sanitas**] is in [$_{LOC}$ **Sunshine Canyon**]

would have the following set of per-word BIO tags.

(11.14) *Mt.* *Sanitas* *is* *in* *Sunshine* *Canyon* .
B-LOC I-LOC O O B-LOC I-LOC O

BIO Scheme with subwords

[$_{LOC}$ **Mt. Sanitas**] is in [$_{LOC}$ **Sunshine Canyon**]

would have the following set of per-word BIO tags.

(11.14) *Mt.* *Sanitas* *is* *in* *Sunshine* *Canyon* .
B-LOC I-LOC O O B-LOC I-LOC O

After wordPiece Tokenization:

```
'Mt', '.', 'San', '##itas', 'is', 'in', 'Sunshine', 'Canyon' '.
```

The sequence does not align with the original tags.

- **Training:** we can just assign the gold-standard tag associated with each word to all of the subword tokens derived from it.
- **Decoding:** the simplest approach is to use the argmax BIO tag associated with the first subword token of a word.

REFERENCES

Daniel Jurafsky and James H. Martin. 2024. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models, 3rd edition. Online manuscript released August 20, 2024. [Chapter 11]


THANK YOU

NPTEL ONLINE CERTIFICATION COURSES

DEEP LEARNING FOR NATURAL LANGUAGE PROCESSING

Lecture 28 : Pretraining Transformer Encoders, Encoder-Decoder

PROF. PAWAN GOYAL
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

CONCEPTS COVERED

- Fine-tuning BERT for Span-based application
- Pretraining Transformer Encoder-Decoder

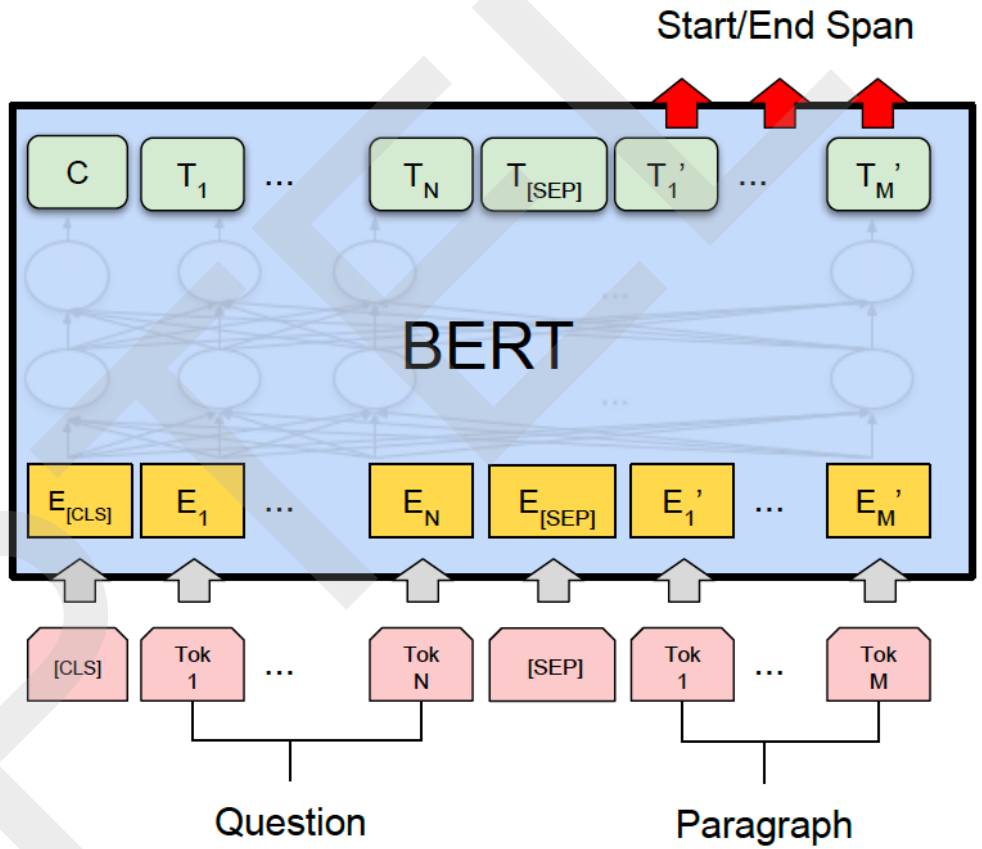
Fine-tuning for span-based applications

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under **gravity**. The main forms of precipitation include drizzle, rain, sleet, snow, **graupel** and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals **within a cloud**. Short, intense periods of rain in scattered locations are called "showers".

What causes precipitation to fall?
gravity

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

Where do water droplets collide with ice crystals to form precipitation?
within a cloud



(c) Question Answering Tasks: SQuAD v1.1

Fine-tuning for SQUAD

- We represent the input question and passage as a single packed sequence (with [SEP])
- We only introduce a start vector $S \in R^{d_h}$ and an end vector $E \in R^{d_h}$ during fine-tuning.
- The probability of word i being the start of the answer span is computed as a dot product between T_i and S followed by a softmax over all of the words in the paragraph.

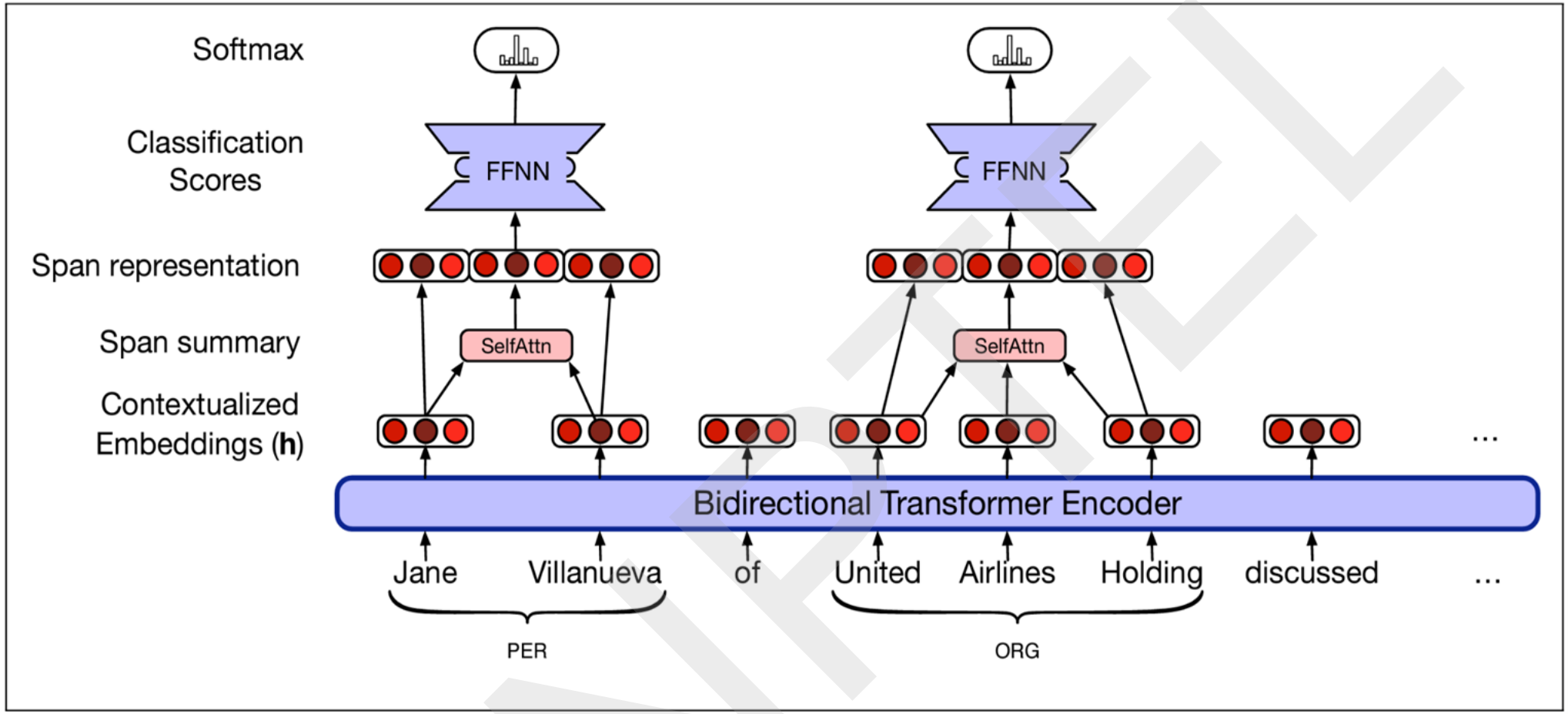
$$P_i = \frac{e^{S \cdot T_i}}{\sum_j e^{S \cdot T_j}}$$

- The analogous formula is used for the end of the answer span.
- The score of a candidate span from position i to position j is defined as $S \cdot T_i + E \cdot T_j$, and the maximum scoring span where $j \geq i$ is used as a prediction.

Fine-tuning for span-based applications

- Formally, given an input sequence x consisting of T tokens, $(x_1, x_2, \ldots, x_T)$, a span is a contiguous sequence of tokens with start i and end j such that $1 \leq i \leq j \leq T$.
- This formulation results in a total set of spans equal to $\frac{T(T+1)}{2}$.
- For practical purposes, span-based models often impose an application-specific length limit L, so the legal spans are limited to those where $j - i < L$.
- We'll refer to the enumerated set of legal spans in input x as $S(x)$

Span-oriented approach for named entity classification



https://web.stanford.edu/~jurafsky/slp3/

Span-oriented approach for named entity classification

Basic Idea

- A span-based approach to NER is a straightforward classification problem where each span in an input is assigned a class label.
- More formally, given an input sequence x, we want to assign a label y, from the set of valid NER labels, to each of the spans in $S(x)$.
- Since most of the spans in a given input will not be named entities we'll add the label NULL to the set of types in Y.

How to represent a span?

Basic Idea

Most schemes for representing spans make use of two primary components:

- representations of the span boundaries, and
- summary representations of the contents of each span

To compute a unified span representation, we concatenate the boundary representations with the summary representation.

How to represent a span?

Simplest possible approach

- we can use the contextual embeddings of the start and end tokens of a span as the boundaries
- the average of the output embeddings within the span as the summary representation

$$g_{ij} = \frac{1}{(j-i)+1} \sum_{k=i}^{j} h_k$$

$$spanRep_{ij} = [h_i; h_j; g_{ij}]$$

Evaluation: General Language Understanding Evaluation (GLUE) Benchmark

Corpus	|Train|	|Test|	Task	Metrics	Domain
			Single-Sentence Tasks		
CoLA	8.5k	**1k**	acceptability	Matthews corr.	misc.
SST-2	67k	1.8k	sentiment	acc.	movie reviews
			Similarity and Paraphrase Tasks		
MRPC	3.7k	1.7k	paraphrase	acc./F1	news
STS-B	7k	1.4k	sentence similarity	Pearson/Spearman corr.	misc.
QQP	364k	**391k**	paraphrase	acc./F1	social QA questions
			Inference Tasks		
MNLI	393k	**20k**	NLI	matched acc./mismatched acc.	misc.
QNLI	105k	5.4k	QA/NLI	acc.	Wikipedia
RTE	2.5k	3k	NLI	acc.	news, Wikipedia
WNLI	634	**146**	coreference/NLI	acc.	fiction books

https://web.stanford.edu/class/cs224n/

Results on GLUE

- **QQP:** Quora Question Pairs (detect paraphrase questions)
- **QNLI**: natural language inference over question answering data
- **SST-2**: sentiment analysis
- **CoLA**: corpus of linguistic acceptability (detect whether sentences are grammatical.)
- **STS-B**: semantic textual similarity
- **MRPC**: microsoft paraphrase corpus
- **RTE**: a small natural language inference corpus

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
$BERT_{BASE}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$BERT_{LARGE}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

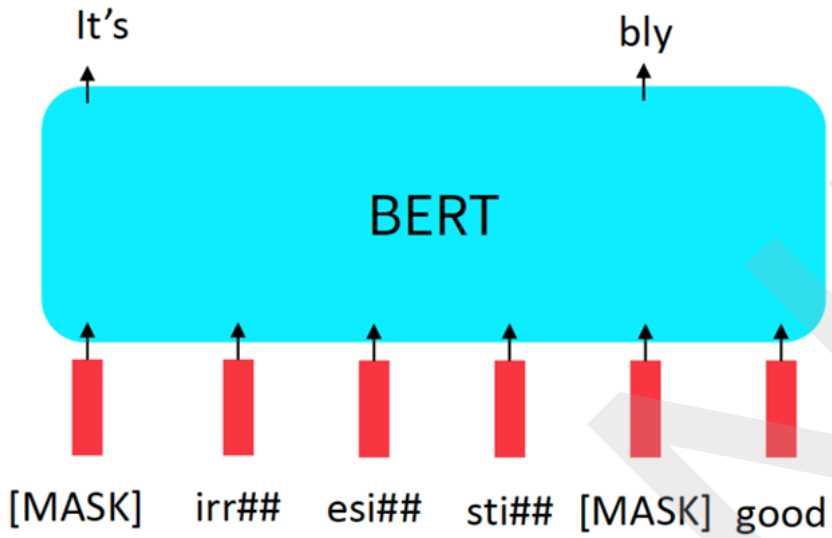
https://web.stanford.edu/class/cs224n/

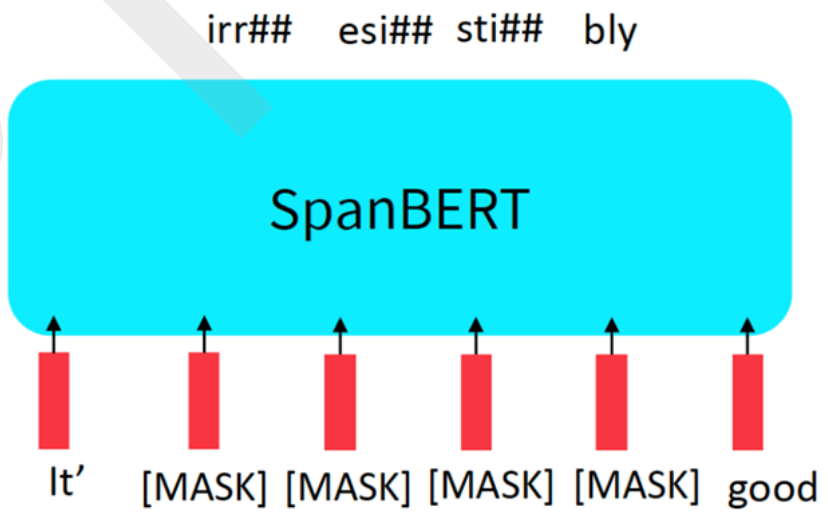
Extensions of BERT

You'll see a lot of BERT variants like RoBERTa, SpanBERT, +++

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





https://web.stanford.edu/class/cs224n/

Extensions of BERT

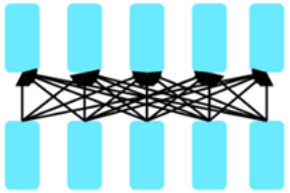
A takeaway from the RoBERTa paper

More compute, more data can improve pretraining even when not changing the underlying Transformer encoder.

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	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7

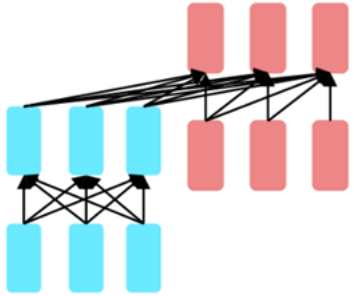
Pretraining for three types of architectures

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



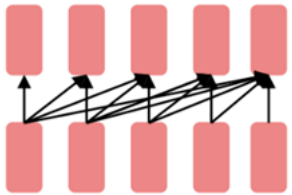
Encoders

- Gets bidirectional context – can condition on future!
- How do we train them to build strong representations?



Encoder-Decoders

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



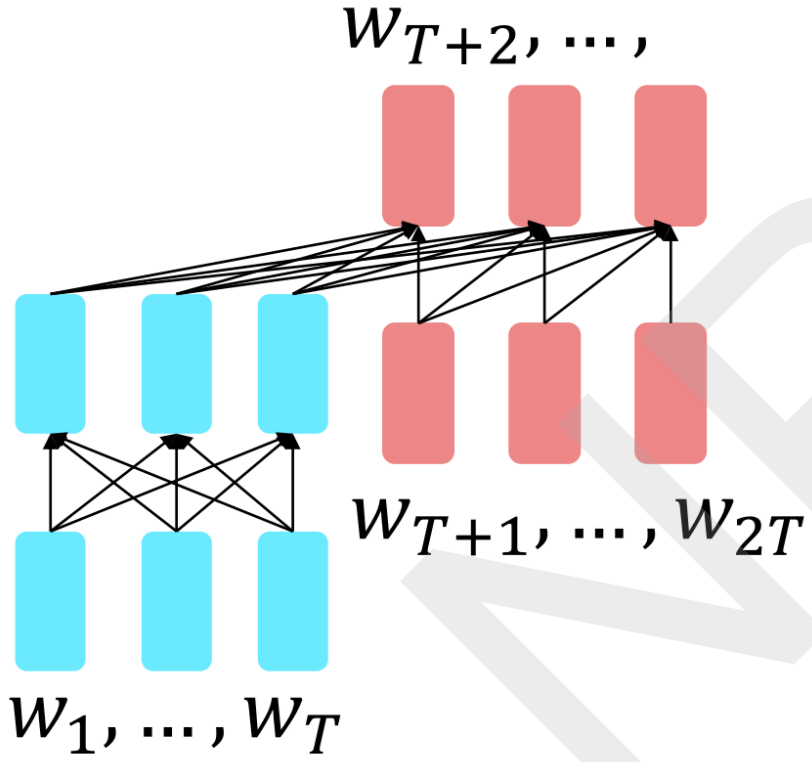
Decoders

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words

Pretraining encoder-decoders

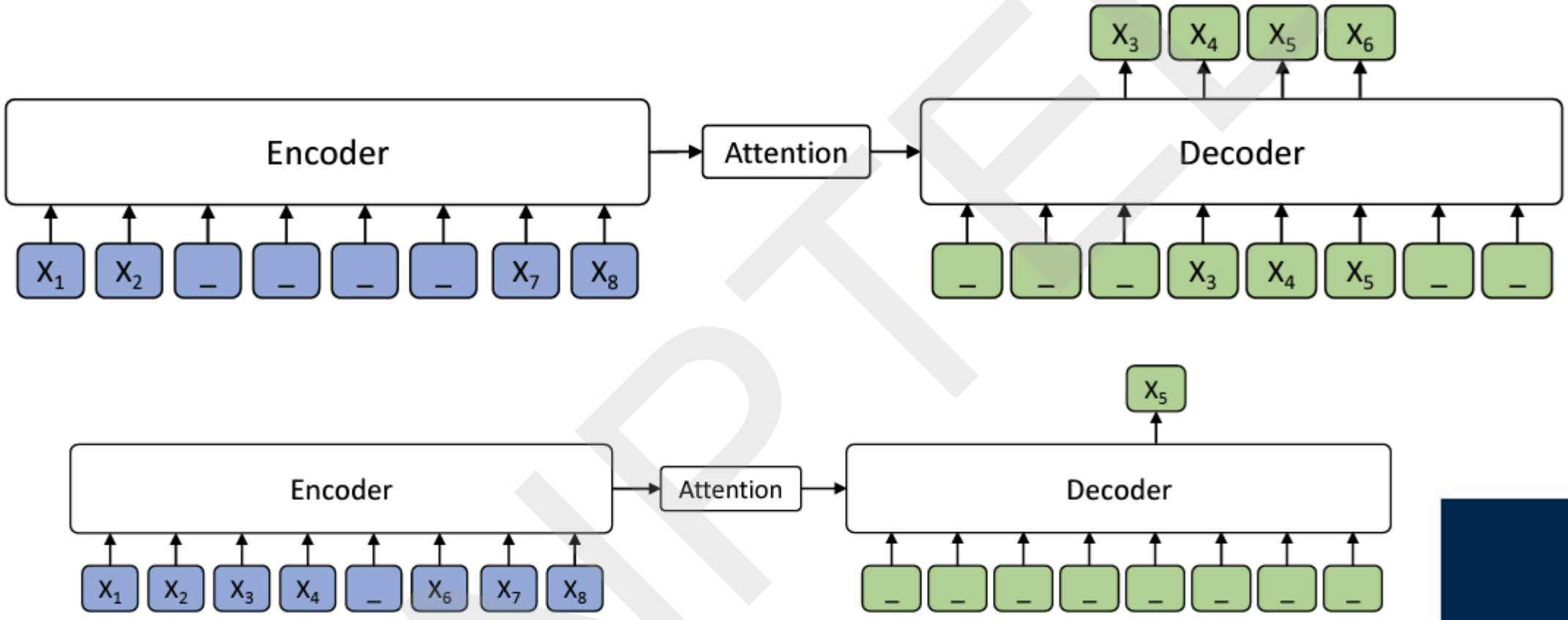
What pretraining objective to use?

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.



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

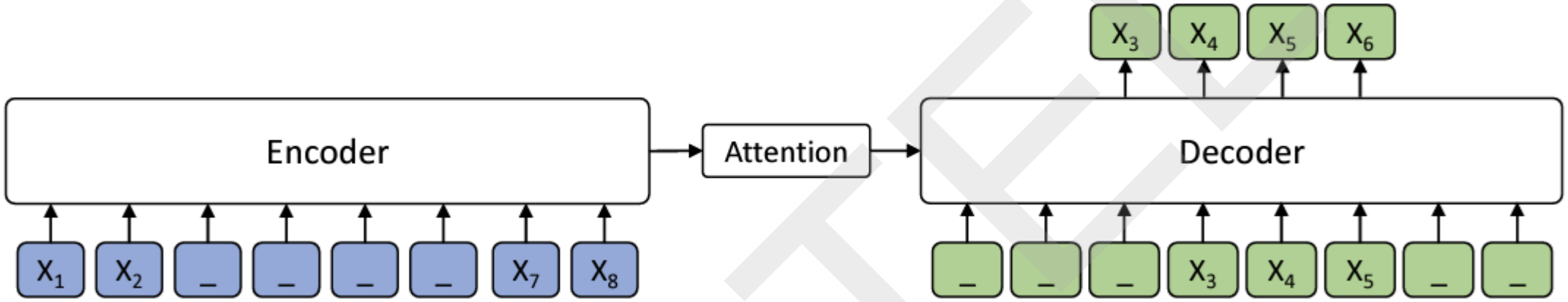
Masked Sequence to Sequence Pretraining (MASS)

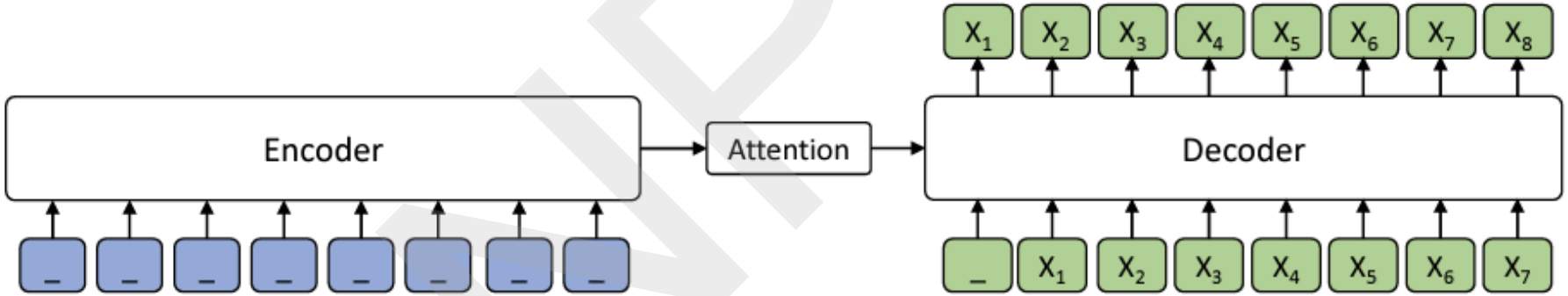


(a) Masked language modeling in BERT ($k = 1$)

Song, Kaitao, et al. "MASS: Masked Sequence to Sequence Pre-training for Language Generation." *International Conference on Machine Learning*. PMLR, 2019.

Masked Sequence to Sequence Pretraining (MASS)





(b) Standard language modeling ($k = m$)

T5: A New Training Objective

Original text

Thank you for inviting me to your party last week.

Raffel, Colin, et al. "Exploring the limits of transfer learning with a unified text-to-text transformer." *Journal of machine learning research* 21.140 (2020): 1-67.

T5: A New Training Objective

Span Corruption

Replace different-length spans from the input with unique placeholders;

Original text

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

Inputs

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

T5: A New Training Objective

Span Corruption

decode out the spans that were corrupted

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>

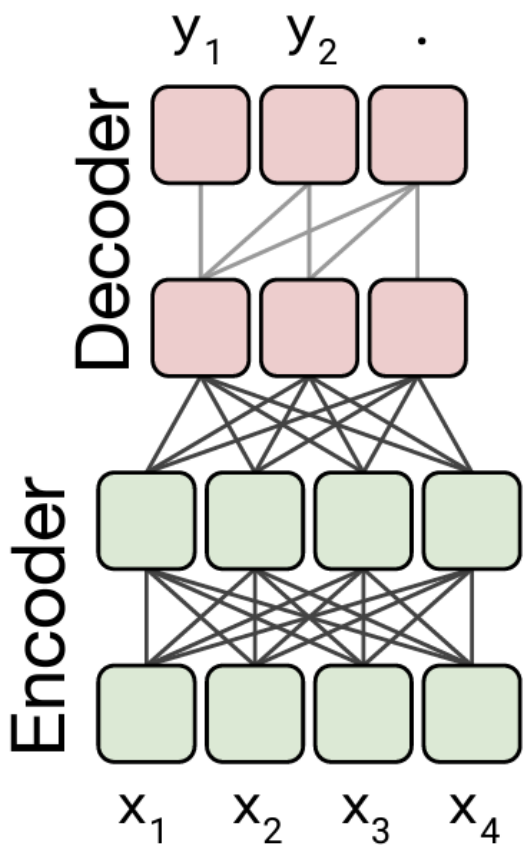
Various objectives considered by T5

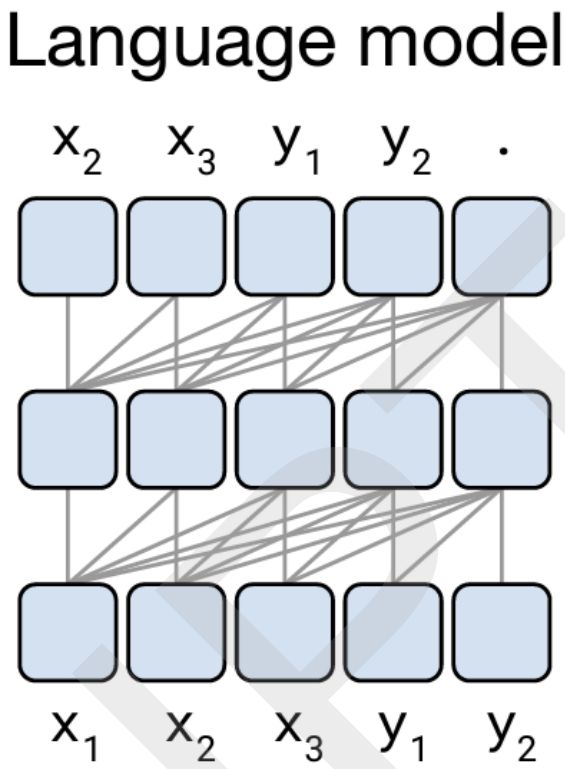
Objective	Inputs	Targets
Prefix language modeling	Thank you for inviting	me to your party last week .
BERT-style Devlin et al. (2018)	Thank you <M> <M> me to your party apple week .	*(original text)*
Deshuffling	party me for your to . last fun you inviting week Thank	*(original text)*
MASS-style Song et al. (2019)	Thank you <M> <M> me to your party <M> week .	*(original text)*
I.i.d. noise, replace spans	Thank you <X> me to your party <Y> week .	<X> for inviting <Y> last <Z>
I.i.d. noise, drop tokens	Thank you me to your party week .	for inviting last
Random spans	Thank you <X> to <Y> week .	<X> for inviting me <Y> your party last <Z>

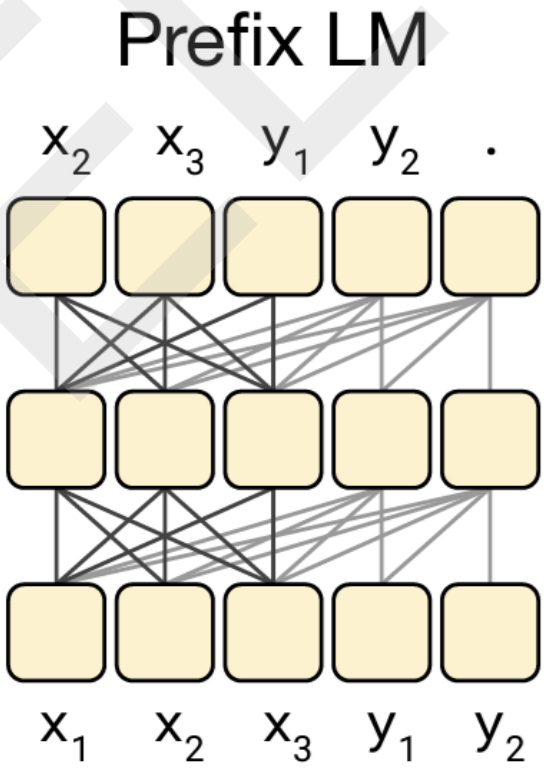
Objective	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
Prefix language modeling	80.69	18.94	77.99	65.27	**26.86**	39.73	**27.49**
BERT-style (Devlin et al., 2018)	**82.96**	**19.17**	**80.65**	**69.85**	**26.78**	**40.03**	**27.41**
Deshuffling	73.17	18.59	67.61	58.47	26.11	39.30	25.62

Objective	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
BERT-style (Devlin et al., 2018)	82.96	19.17	**80.65**	69.85	26.78	**40.03**	27.41
MASS-style (Song et al., 2019)	82.32	19.16	80.10	69.28	26.79	**39.89**	27.55
★ Replace corrupted spans	83.28	**19.24**	**80.88**	**71.36**	**26.98**	39.82	**27.65**
Drop corrupted tokens	**84.44**	**19.31**	**80.52**	68.67	**27.07**	39.76	**27.82**

Various architectures considered by T5



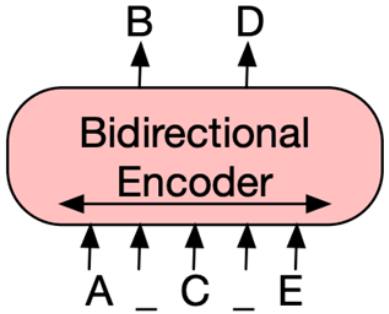




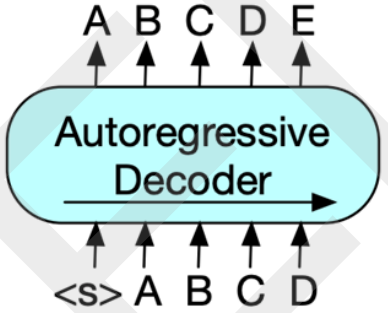
Various architectures considered by T5

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86
Prefix LM	Denoising	P	M	81.82	18.61	78.94	68.11	26.43	37.98	27.39
Encoder-decoder	LM	$2P$	M	79.56	18.59	76.02	64.29	26.27	39.17	26.86
Enc-dec, shared	LM	P	M	79.60	18.13	76.35	63.50	26.62	39.17	27.05
Enc-dec, 6 layers	LM	P	$M/2$	78.67	18.26	75.32	64.06	26.13	38.42	26.89
Language model	LM	P	M	73.78	17.54	53.81	56.51	25.23	34.31	25.38
Prefix LM	LM	P	M	79.68	17.84	76.87	64.86	26.28	37.51	26.76

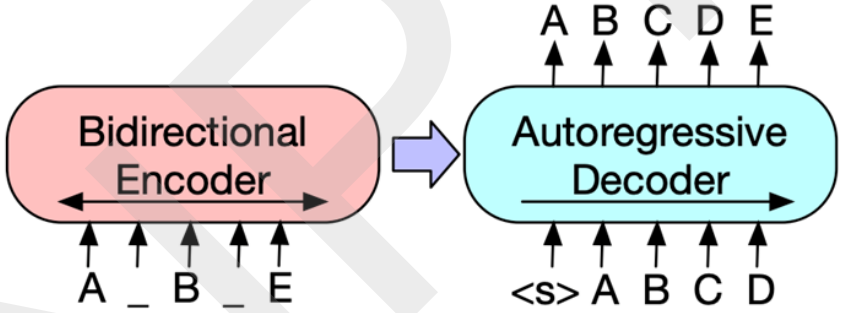
Bi-directional and Auto-Regressive Transformers (BART)



(a) BERT: Random tokens are replaced with masks, and the document is encoded bidirectionally. Missing tokens are predicted independently, so BERT cannot easily be used for generation.

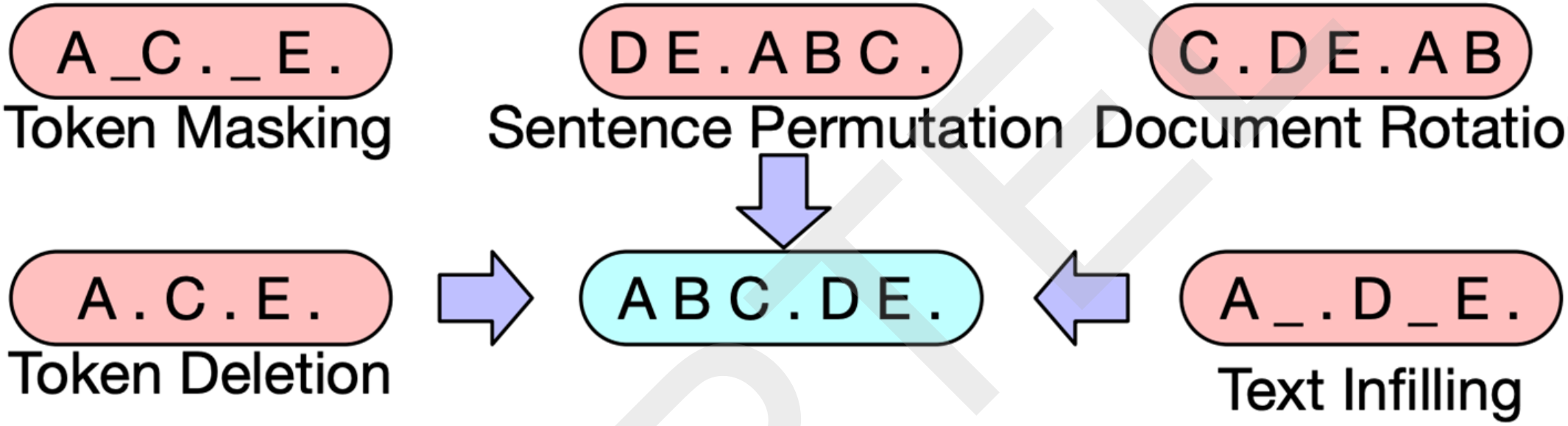


(b) GPT: Tokens are predicted auto-regressively, meaning GPT can be used for generation. However words can only condition on leftward context, so it cannot learn bidirectional interactions.



(c) BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitary noise transformations. Here, a document has been corrupted by replacing spans of text with mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and we use representations from the final hidden state of the decoder.

BART: Transformation for noising the input



How to fine-tune pretrained encoder-decoder?

No additional parameters required!

Just fine-tune the encoder decoder model through additional task-specific training data

Loss computed at decoder is used to update all the encoder and decoder parameters

How to fine-tune pretrained encoder-decoder?

T5 can be used for various tasks

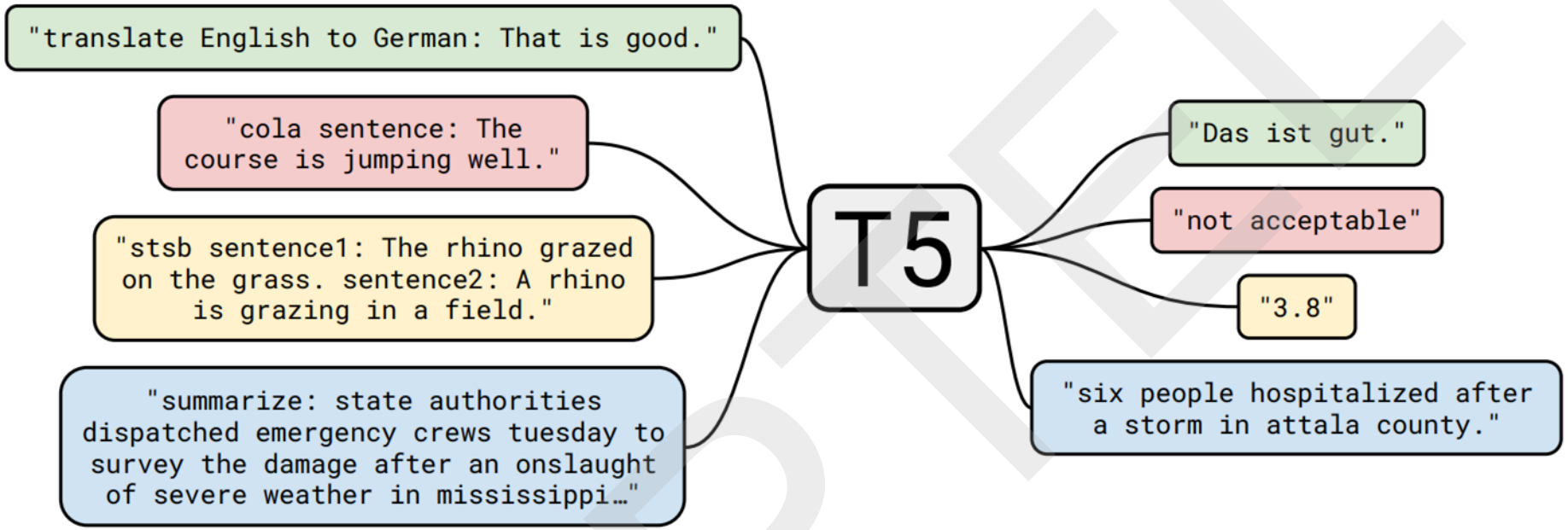


Figure 1: A diagram of our text-to-text framework. Every task we consider—including translation, question answering, and classification—is cast as feeding our model text as input and training it to generate some target text. This allows us to use the same model, loss function, hyperparameters, etc. across our diverse set of tasks. It also provides a standard testbed for the methods included in our empirical survey. "T5" refers to our model, which we dub the "**T**ext-**t**o-**T**ext **T**ransfer **T**ransformer".

REFERENCES

Daniel Jurafsky and James H. Martin. 2024. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models, 3rd edition. Online manuscript released August 20, 2024. [Chapter 11]


THANK YOU

NPTEL ONLINE CERTIFICATION COURSES

DEEP LEARNING FOR NATURAL LANGUAGE PROCESSING

Lecture 29 : Pretraining Transformer Decoder

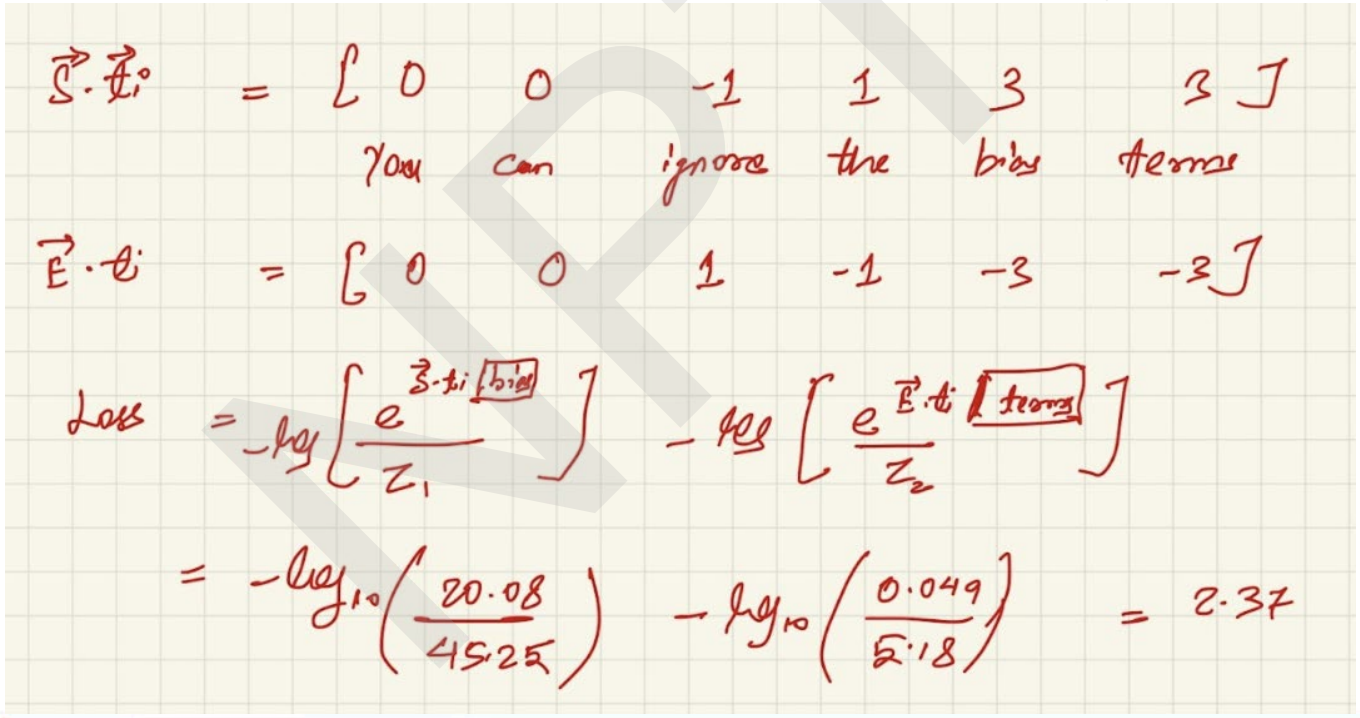
PROF. PAWAN GOYAL
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

CONCEPTS COVERED

- Pretraining Transformer Decoder
- Fine-tuning GPT for various NLP tasks
- GPT series: Zero-shot and In-context Learning

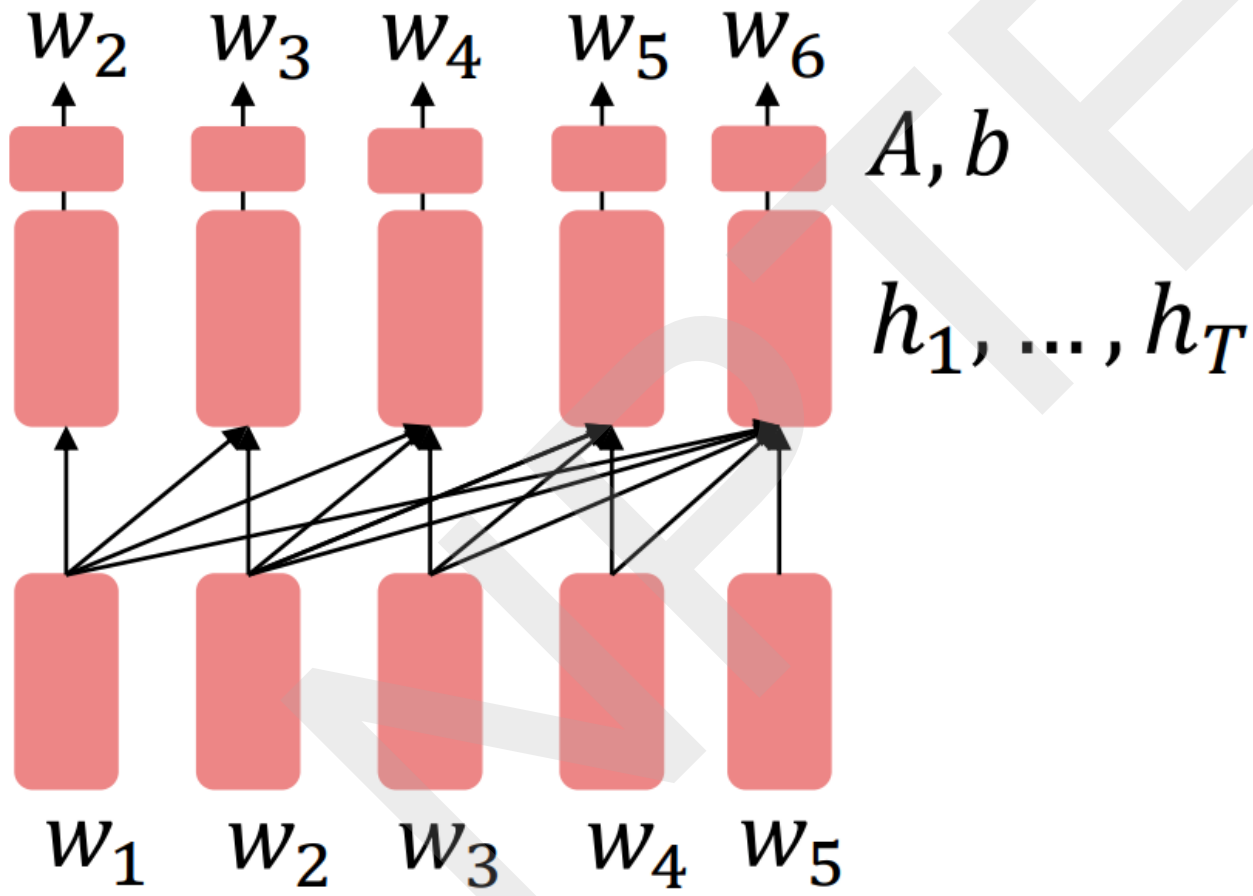
Try This Problem

Suppose you are using BERT for reading comprehension based question answering. Suppose your input paragraph is, “You can ignore the bias terms”, and assume that each individual word is part of the vocabulary and the underlined span is the ground truth answer. For the sake of simplicity, assume that you are working with 2-dimensional hidden states. Let your start and end vectors be [1,-1] and [-1,1], respectively, and the final embedding for the words in the input sentence be [-1,-1], [1,1], [1,2], [2,1], [1,-2] and [2,-1], respectively. (a) If this was a sentence in the training, what would be the corresponding loss for this sentence? (b) If this was a sentence during inference, how many spans would you have to consider? (c) Assume that it is given that the span can have at most 2 words. What will be the output of the model at inference time?



Pretraining Decoders

It's natural to pretrain decoders as language models



https://web.stanford.edu/class/cs224n/

Generative Pretrained Transformer (GPT)

- 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.

Vocabulary larger than BERT

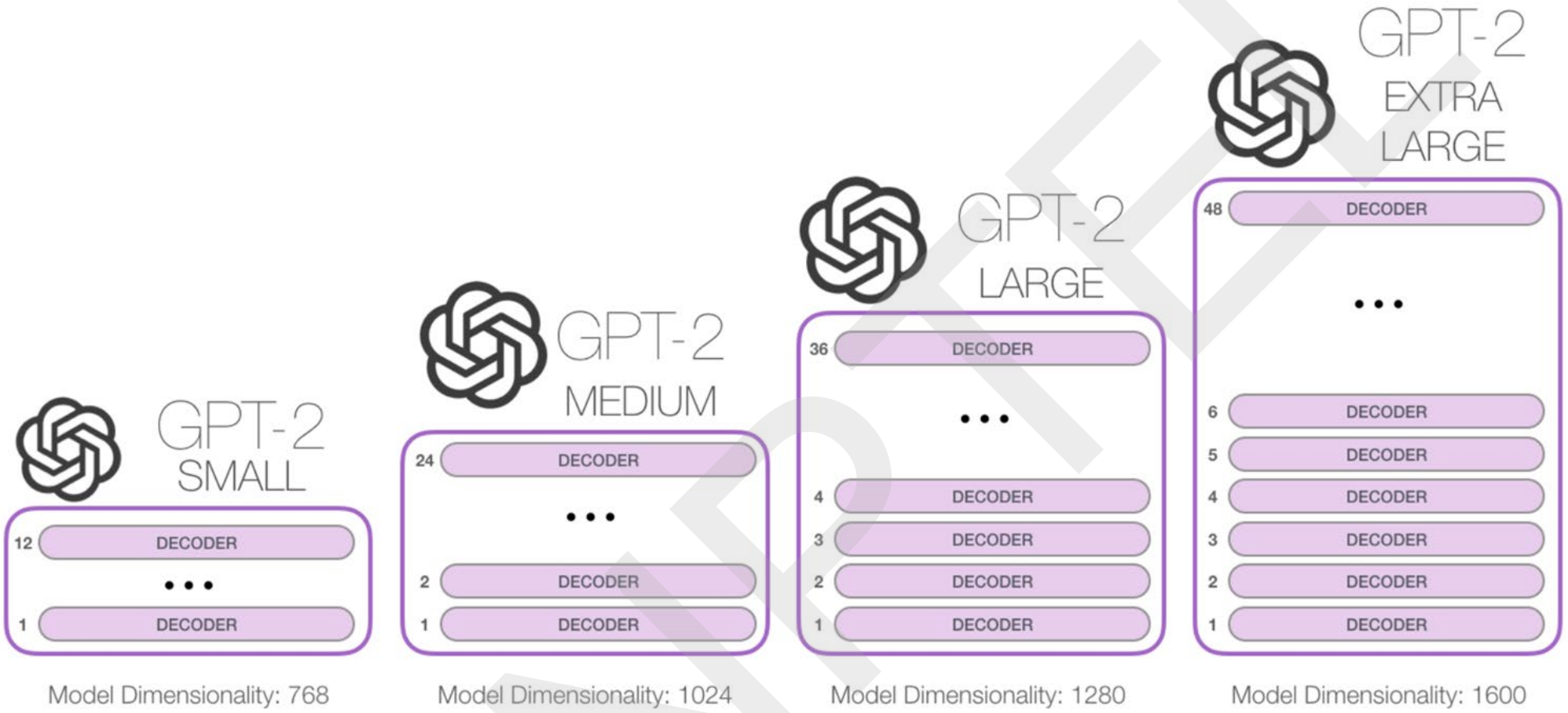
https://web.stanford.edu/class/cs224n/

Comparison of GPT series

Characteristic	GPT-1	GPT-2	GPT-3
Parameters	117 Million	1.5 Billion	175 Billion
Decoder Layers	12	48	96
Context Token Size	512	1024	2048
Hidden Layer Size	768	1600	12288
Batch Size	64	512	3.2M

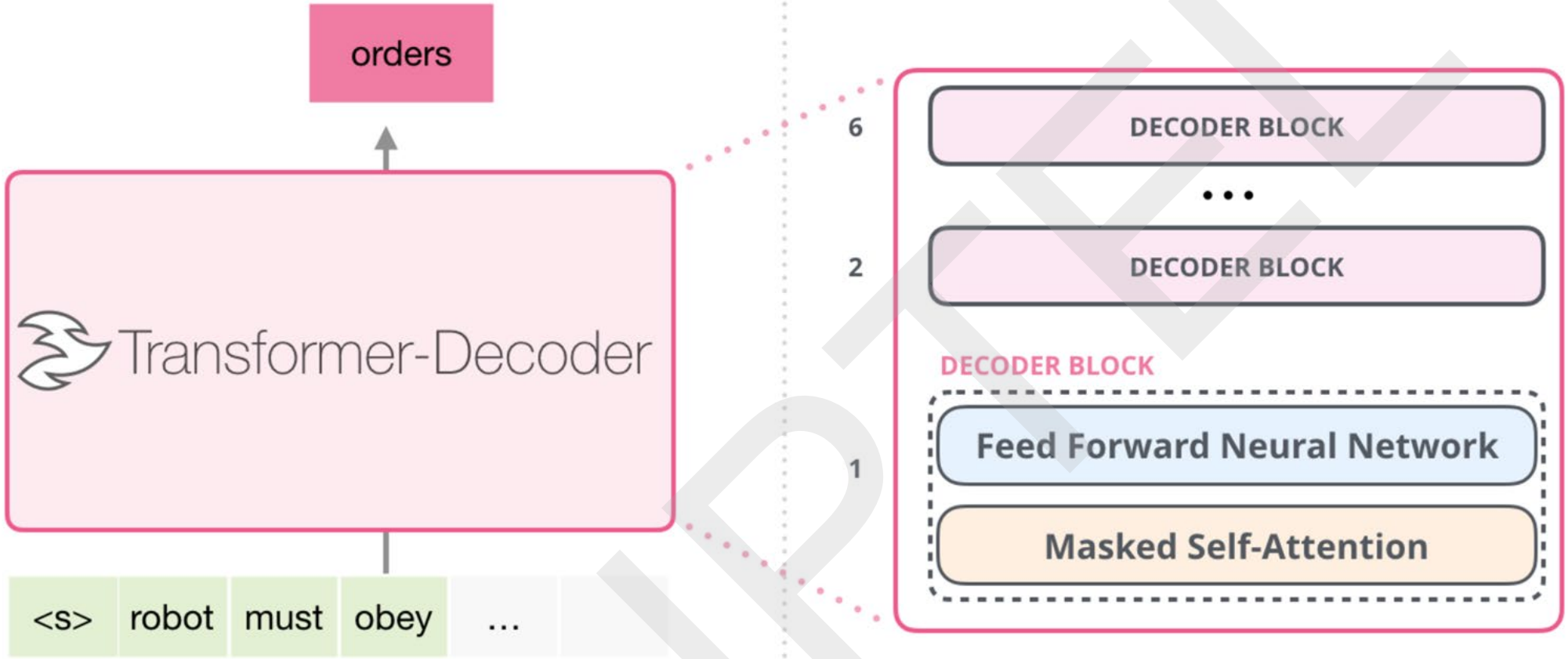
Image Source: “Large Language Models: A Deep Dive”

Pretrained Decoders



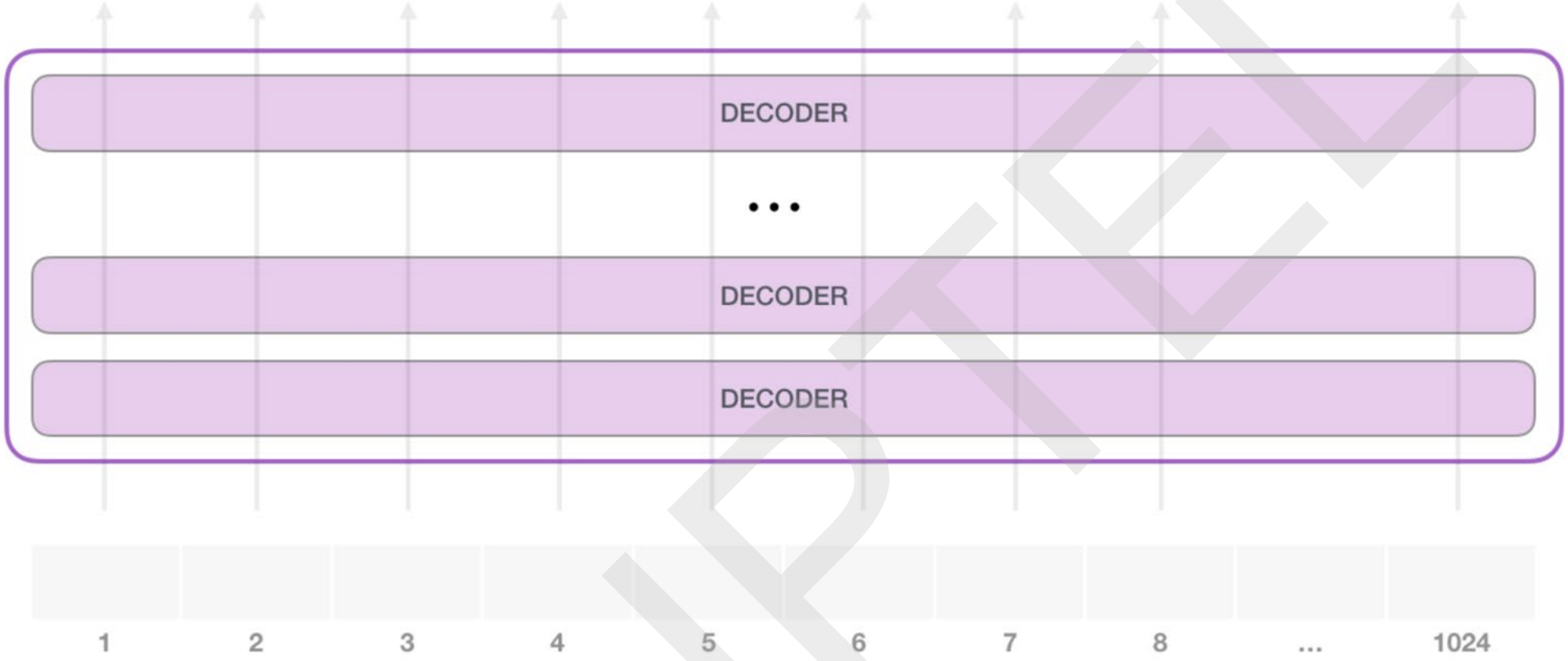
https://jalammar.github.io/illustrated-gpt2/

GPT: Transformer-Decoder



https://jalammar.github.io/illustrated-gpt2/

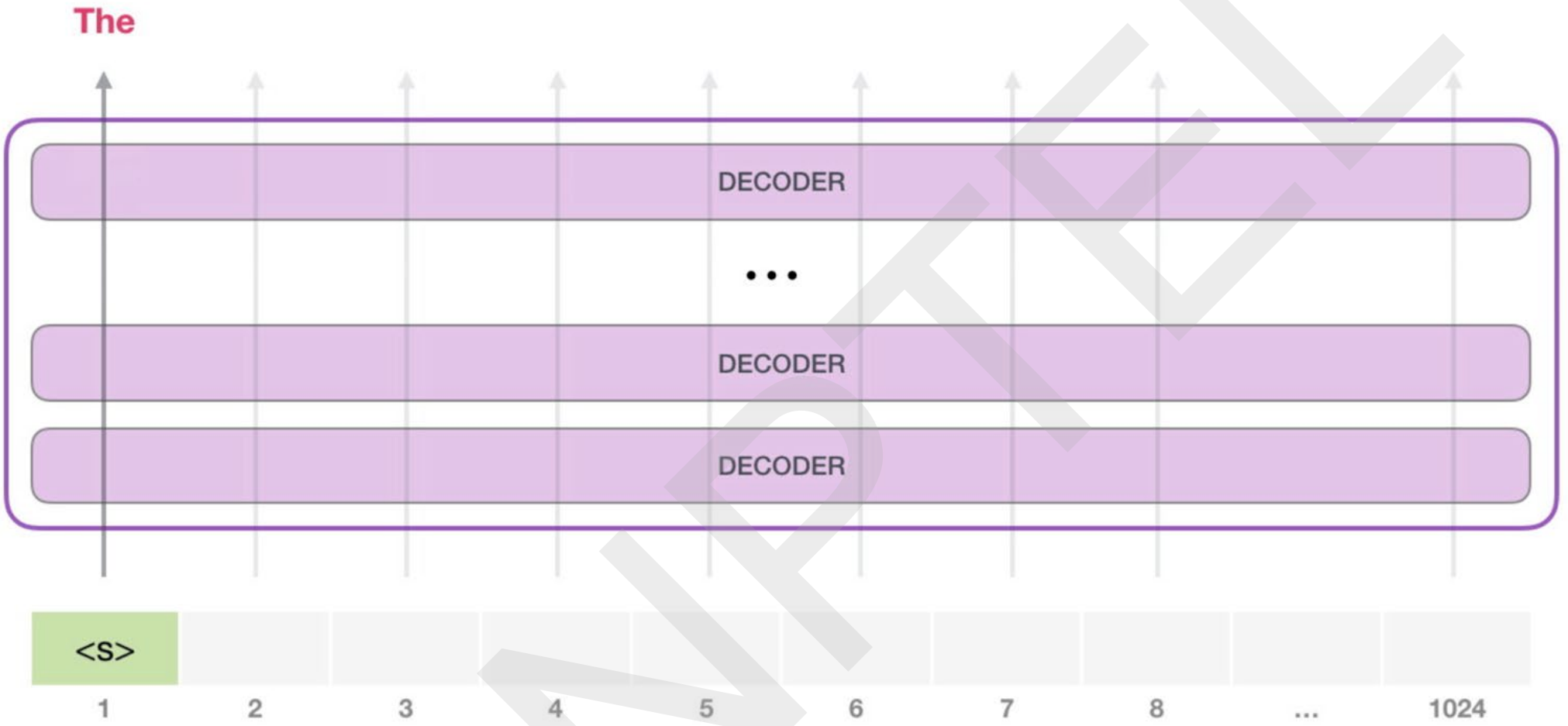
GPT-2: Quick Summary



The GPT-2 can process 1024 tokens. Each token flows through all the decoder blocks along its own path.

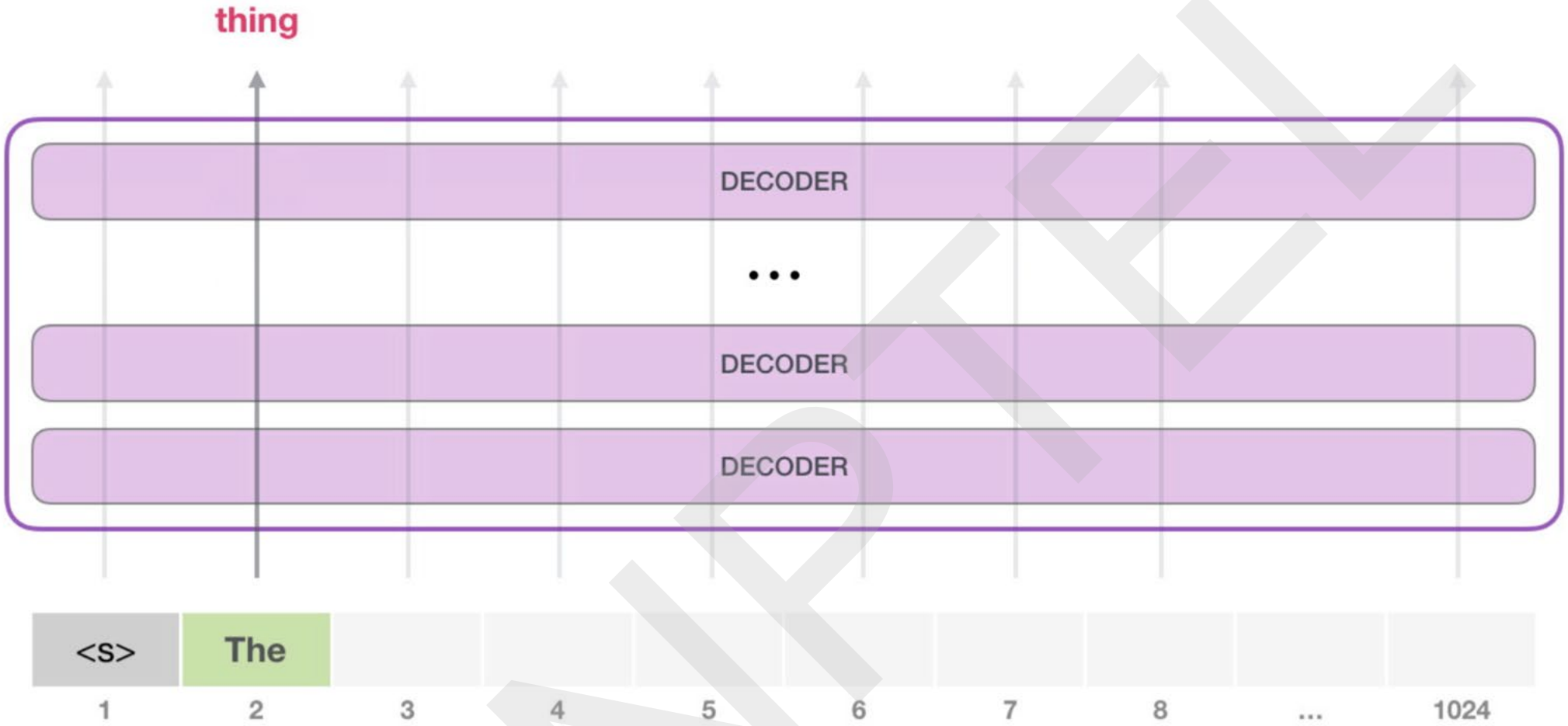
https://jalammar.github.io/illustrated-gpt2/

GPT-2: Quick Summary



https://jalammar.github.io/illustrated-gpt2/

GPT-2: Quick Summary

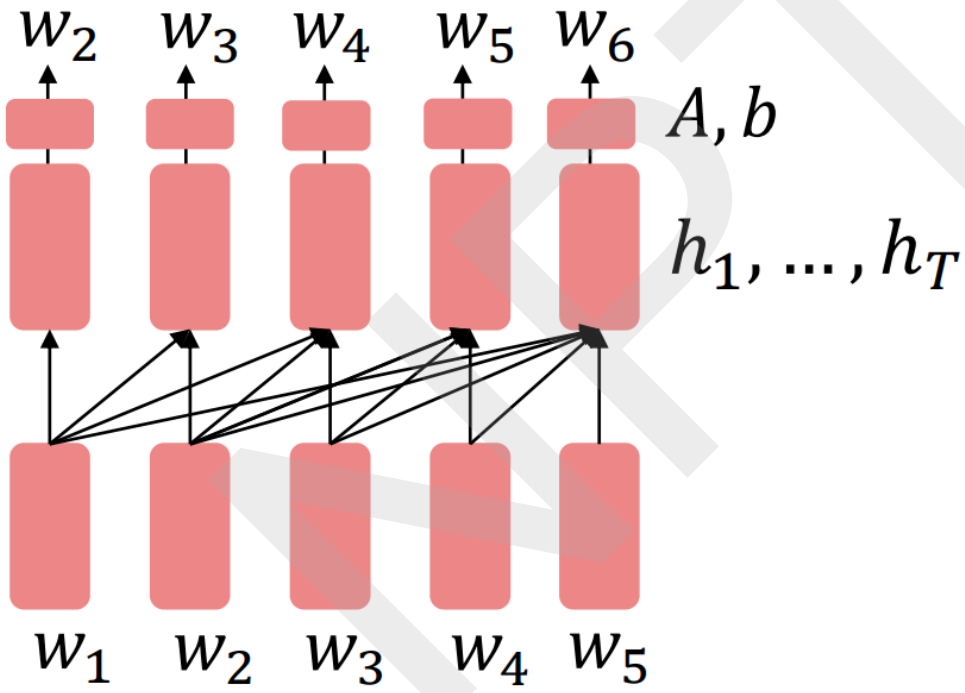


https://jalammar.github.io/illustrated-gpt2/

How to use Pretrained Decoders?

This is helpful in tasks where the output is a sequence

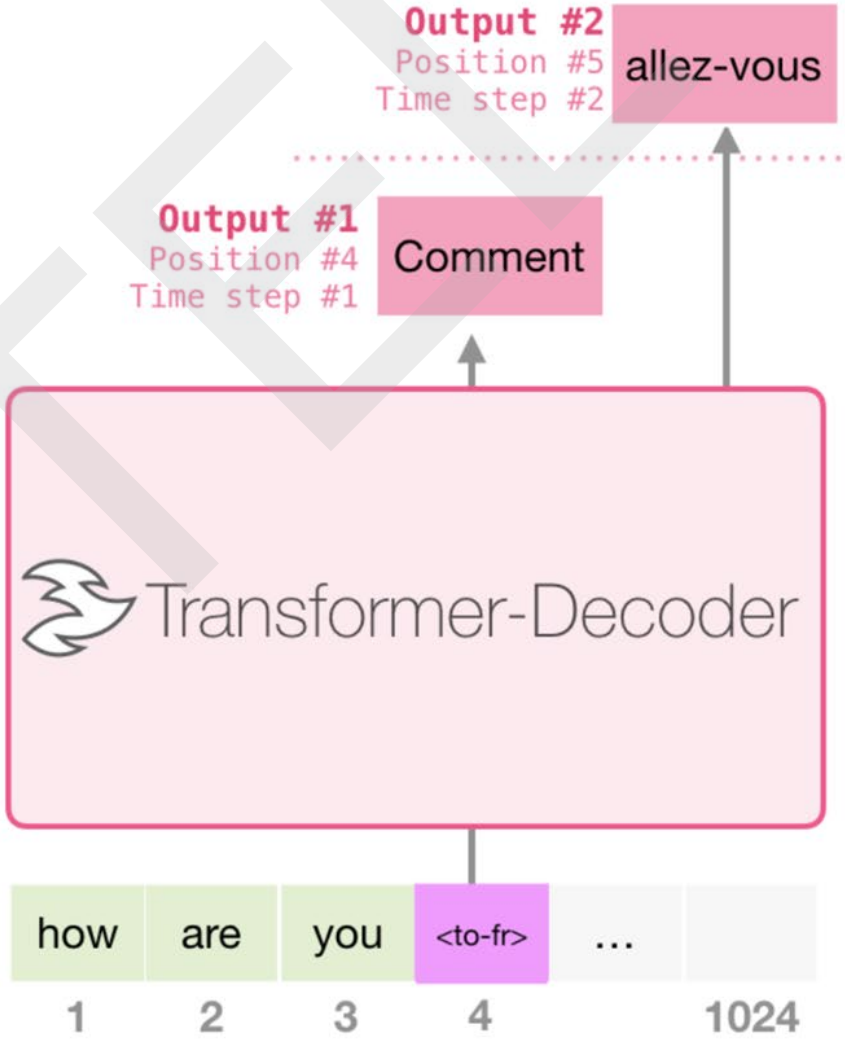
- Dialogue (context=dialogue history)
- Summarization (context=document)



GPT-2: Machine Translation

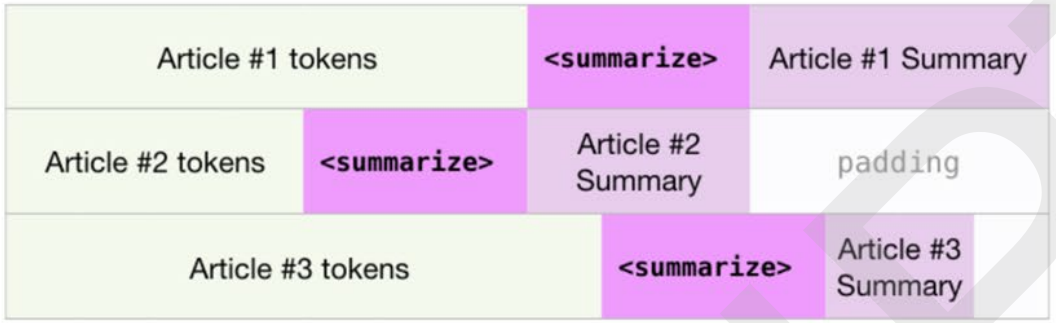
Training Dataset

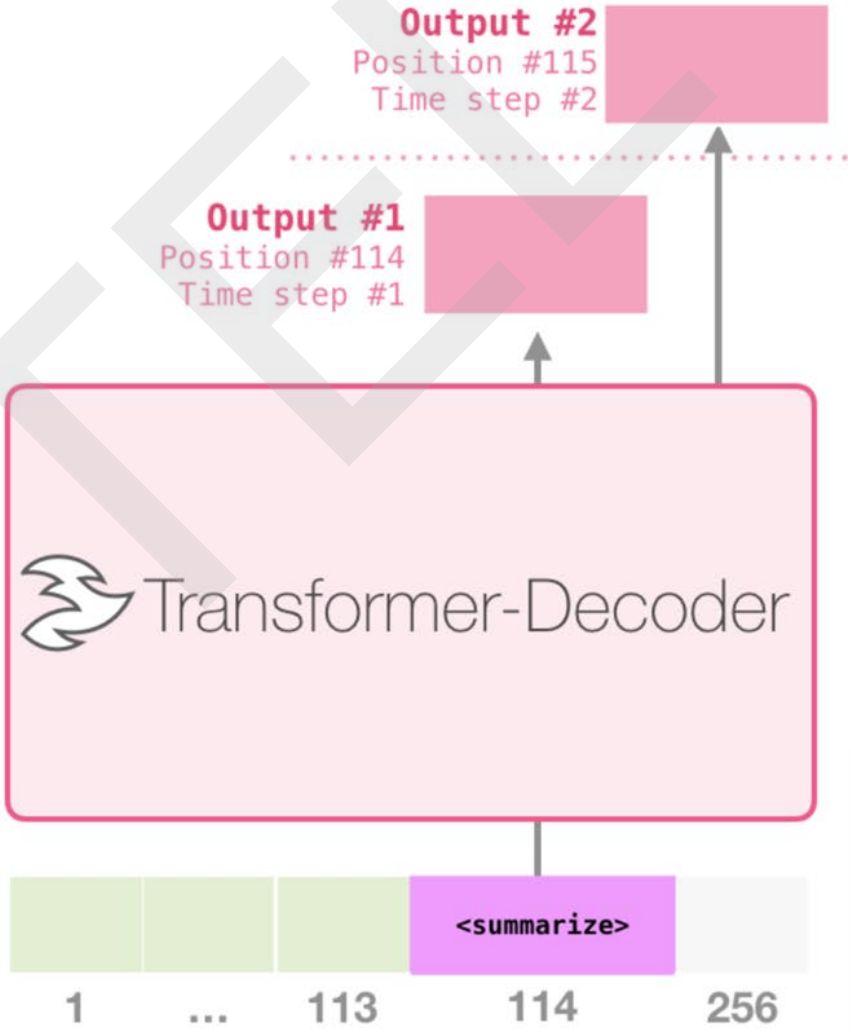
I	am	a	student	<to-fr>	je	suis	étudiant
let	them	eat	cake	<to-fr>	Qu'ils	mangent	de
good	morning	<to-fr>	Bonjour				



https://jalammar.github.io/illustrated-gpt2/

GPT-2: Summarization





https://jalammar.github.io/illustrated-gpt2/

GPT: Beyond Language Modeling

Basic Idea

The decoder can work with a prompt!

Any NLP task can be expressed in a probabilistic framework as estimating a conditional distribution $p(output|input)$.

For instance, reading comprehension training example

(answer the question, document, question, answer)

How to format inputs?

Natural Language Inference

Premise: *The man is in the doorway*
Hypothesis: *The person is near the door*
} **entailment**

Radford et al., 2018 evaluate on natural language inference.

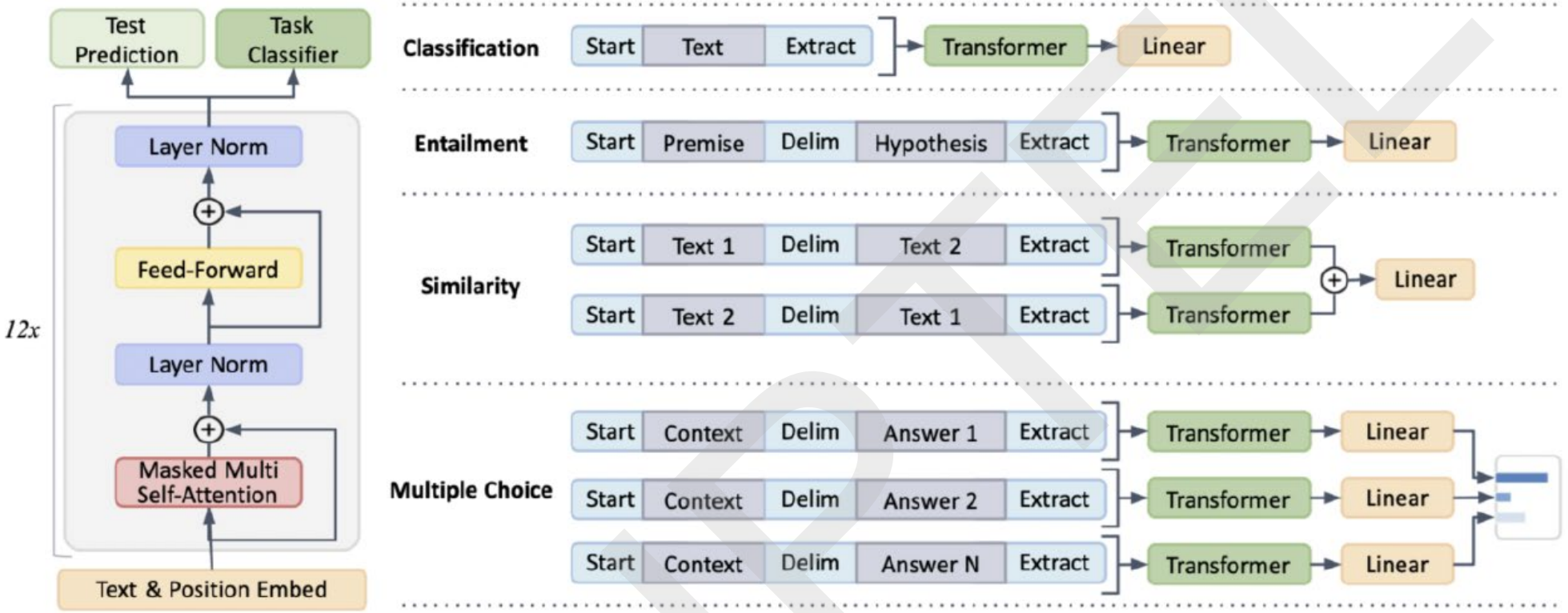
Here's roughly how the input was formatted, as a sequence of tokens for the decoder.

[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.

https://web.stanford.edu/class/cs224n/

GPT



The GPT-1 architecture and designated training objectives employed for training. Structured inputs are converted into sequences of tokens for fine-tuning different tasks, which the pre-trained model processes, followed by implementing a linear layer with a softmax layer.

Image Source: “Large Language Models: A Deep Dive”

GPT-2: Zero shot and in-context: Beginning

- “We demonstrate language models can perform down-stream tasks in a zero-shot setting – without any parameter or architecture modification.”
- “To induce summarization behavior we add the text `TL;DR:` after the article and generate 100 tokens”
- “We test whether GPT-2 has begun to learn how to translate from one language to another. In order to help it infer that this is the desired task, we condition the language model on a context of example pairs of the format `english sentence = french sentence` and then after a final prompt of `english sentence =` we sample from the model with greedy decoding and use the first generated sentence as the translation.”
- “Similar to translation, the context of the language model is seeded with example question answer pairs which helps the model infer the short answer style of the dataset.”’

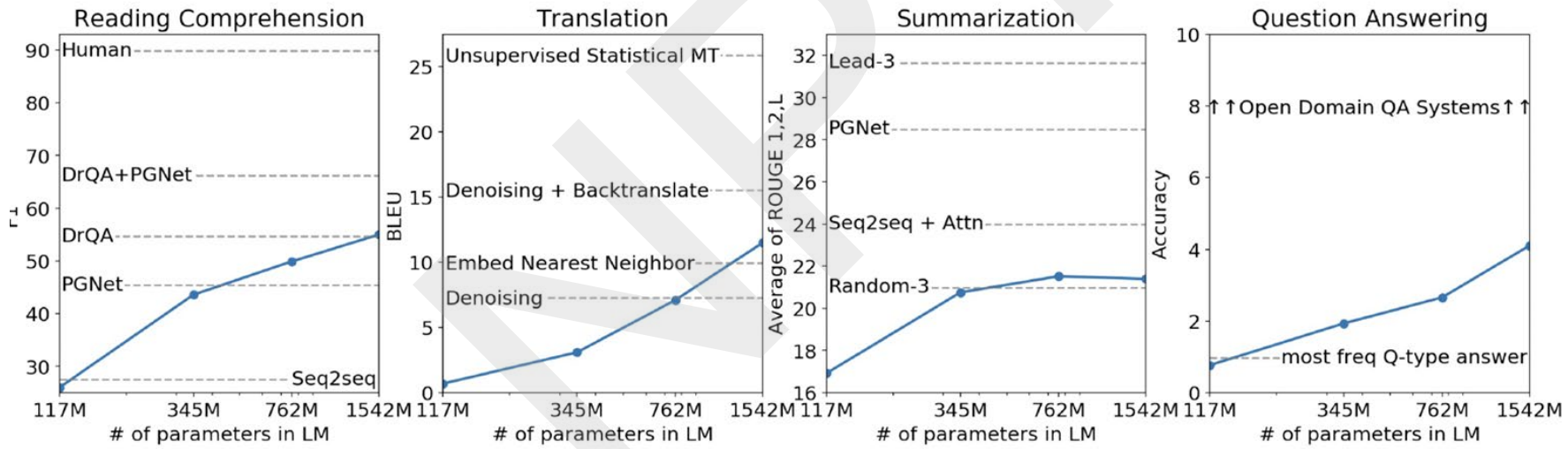
https://web.stanford.edu/class/cs224u/

GPT-2

GPT-2 (Radford et al. 2019) scaled the GPT model to a larger model.

Demonstrated that language models are able to do NLP tasks **without any explicit supervision (zero-shot)**.

- e.g., when conditioned on a document plus questions, the answers generated by GPT-2 reach 55 F1 on the CoQA dataset.



Zero-shot task performance of GPT-2 as a function of model size on many NLP tasks

https://rycolab.io/classes/llm-s24/

Cultural Moment: GPT-3

“Here we show that scaling up language models greatly improves task-agnostic, few-shot performance, sometimes even reaching competitiveness with prior state-of-the-art fine-tuning approaches. Specifically, we train GPT-3, an autoregressive language model with 175 billion parameters, 10x more than any previous non-sparse language model, and test its performance in the few-shot setting. For all tasks, GPT-3 is applied without any gradient updates or fine-tuning, with tasks and few-shot demonstrations specified purely via text interaction with the model. GPT-3 achieves strong performance on many NLP datasets, including translation, question-answering, and cloze tasks, as well as several tasks that require on-the-fly reasoning or domain adaptation, such as unscrambling words, using a novel word in a sentence, or performing 3-digit arithmetic. At the same time, we also identify

https://web.stanford.edu/class/cs224u/

In-context learning: Terminology

- In-context learning: A frozen LM performs a task only by conditioning on the prompt text.
- Few-shot in-context learning: (1) The prompt includes examples of the intended behavior, and (2) no examples of the intended behavior were seen in training.
  - We are unlikely to be able to verify (2).
  - "Few-shot" is also used in supervised learning with the sense of "training on few examples". The above is different.
- Zero-shot in-context learning: (1) The prompt includes no examples of the intended behavior (but it can contain other instructions), and (2) no examples of the intended behavior were seen in training.
  - We are unlikely to be able to verify (2).
  - Formatting and other instructions seem like a gray area, but we will allow them in this category.

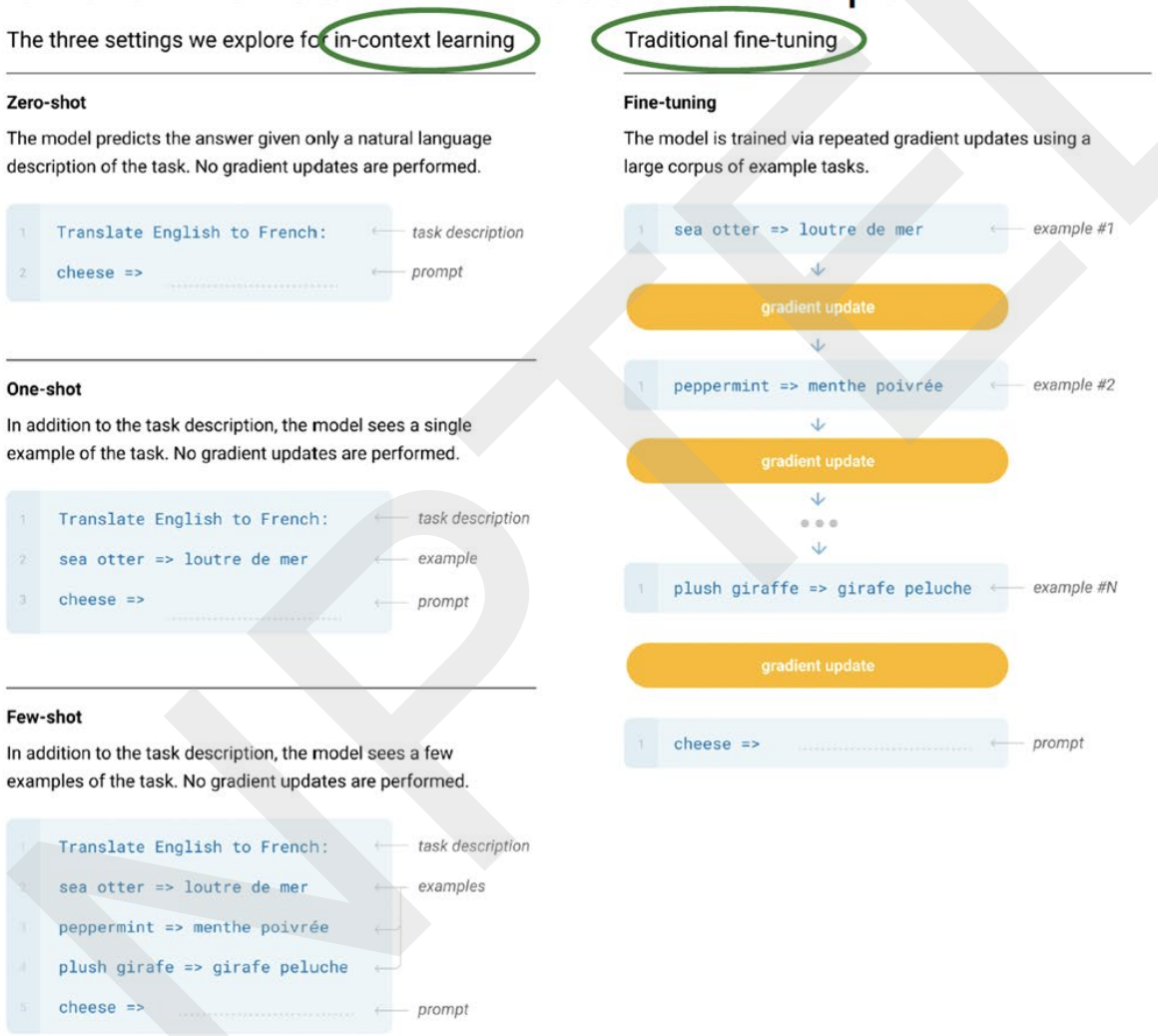
https://web.stanford.edu/class/cs224u/

GPT-3

Generative Pretrained Transformer series of models from OpenAI

GPT-3 (Brown et al. 2020) further scaled the GPT-2 model to 175 Billion (100 times larger compared to the largest GPT-2 model).

In addition to improved zero-shot performance, GPT-3 also exhibits strong few-shot **"in-context learning" ability**



In in-context learning, we append the task description and a few demonstrations of the task to the original input as the input for the language model, which produces the prediction by auto-regressively predicting next tokens

https://rycolab.io/classes/llm-s24/

Prompting as the standard interface for GPT-3

Input: x = "I love this movie"

⇩

Template: [x] Overall, it was [z]

⇩

Prompting: x' = "I love this movie. Overall it was [z]"

Prompting: x' = "I love this movie. Overall it was [z]"

⇩

Predicting: x' = "I love this movie. Overall it was fantastic"

https://phontron.com/class/anlp2024/assets/slides/anlp-07-prompting.pdf

REFERENCES

Kamath, Uday, et al. Large Language Models: A Deep Dive: Bridging Theory and Practice. Springer Nature, 2024. [Chapter 2]


THANK YOU

NPTEL ONLINE CERTIFICATION COURSES

DEEP LEARNING FOR NATURAL LANGUAGE PROCESSING

Lecture 30 : More on Pretraining

PROF. PAWAN GOYAL
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

CONCEPTS COVERED

- Domain-specific Pretraining
- Multilingual Pretraining
- Uptraining

Domain-specific Pretraining

Domain-specific Pretraining

Pretrained models make use of general domain corpora such as news and Wikipedia.

How to extend these to various domains such as Scientific, Finance, Biomedical, etc.?

Can we pretrain these on data from these domains?

SCIBERT

SCIBERT is a pretrained language model based on BERT but trained on a large corpus of scientific text.

So, what is the recipe?

1. Do we pretrain transformers from scratch on scientific domain data?
2. Do we start from BERT check-points and continue pretraining on scientific domain?

While Option 2 looks better, answer is not always that simple.
Why?

Domain Vocabulary

BERT uses WordPiece for unsupervised tokenization of the input text.
The vocabulary is built such that it contains the most frequently used words or subword units.
We refer to the original vocabulary released with BERT as BASEVOCAB

SCIBERT authors used their pretrained data and created the same sized vocabulary: *SCIVOCAB*

The overlap between SCIVOCAB and BASEVOCAB: 42%!!
A substantial difference in frequently used words between scientific and general domain texts

Domain Vocabulary: So how does it matter?

We can continue pretraining BERT for scientific domain only with the original vocabulary

To replace the new vocabulary, the model has to be pretrained from Scratch.

As per the experiments, using pretraining (from scratch) with SCIVOCAB gave slight edge over (continual) pretraining with BASEVOCAB.

> ***From the paper:*** Given the disjoint vocabularies and the magnitude of improvement over BERT-Base, we suspect that while an in-domain vocabulary is helpful, SCIBERT benefits *most from the scientific corpus pretraining*.

Side-effect of continual pretraining

Catastrophic Forgetting: Continual pretraining on new domain might lead to forgetting of the previous knowledge

TASK	CoLA	SST2	MRPC		STS		QQP		MNLI	QNLI	RTE
METRIC	Matthews CC	Acc.	F1-score	Acc.	Pearson CC	Spearman CC	F1-score	Acc.	Acc.	Acc.	Acc.
$RoBERTa_{BASE}$	**63.71**	94.15	92.71	89.71	90.91	**90.66**	**89.1**	**91.84**	**87.24**	92.26	**80.14**
$EManuals_{RoBERTa}$	51.82 (-11.89)	91.97 (-2.18)	91.42 (-1.29)	87.99 (-1.72)	88.4 (-2.51)	88.36 (-2.3)	88.65 (-0.45)	91.55 (-0.29)	85.15 (-2.09)	91.34 (-0.92)	70.4 (-9.74)

RoBERTa model pretrained on E-manuals domain shows catastrophic forgetting on GLUE benchmark

Remedies: *Mix general pretrain data, mix parameters (mix-out)*

https://arxiv.org/pdf/2306.06190

Multilingual Pretraining

Multilingual pre-training

- So far, we've mainly talked about pretraining and fine-tuning models on English text.

- One approach: pretrain BERT-like models on *monolingual* data from a different language
  - "BERTje > Dutch, "FlauBERT" > French, "PhoBERT" > Vietnamese, etc.

- Another approach: pretrain models on a large mixture of many languages
  - mBERT, mBART, XLM-R, mT5, byT5, etc.
  - Allows for transfer learning *across* languages

https://people.cs.umass.edu/~miyyer/cs685/

mBERT: Encoder Transformer

Architecture: Almost same as the BERT model.

Dataset: For the pre-training, it is trained on the Wikipedia pages of 104 languages with a shared word piece vocabulary.

Vocabulary: For tokenization, a 110k shared WordPiece vocabulary is used.

Training Objective: Same as BERT.

mBERT: Temperature based data sampling

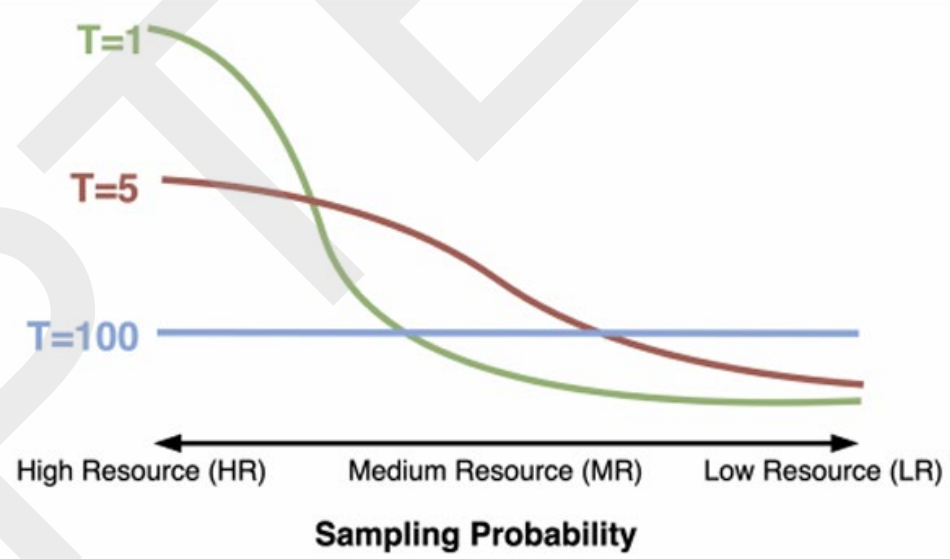
Temperature based data sampling *to address data skew*
(during vocab construction and pre-training)

Data Distribution
Language *l*

$$p_l = \frac{D_l}{\Sigma_k D_k}$$

Sampling Probability
Language *l*

$$p_l^{\frac{1}{T}}$$



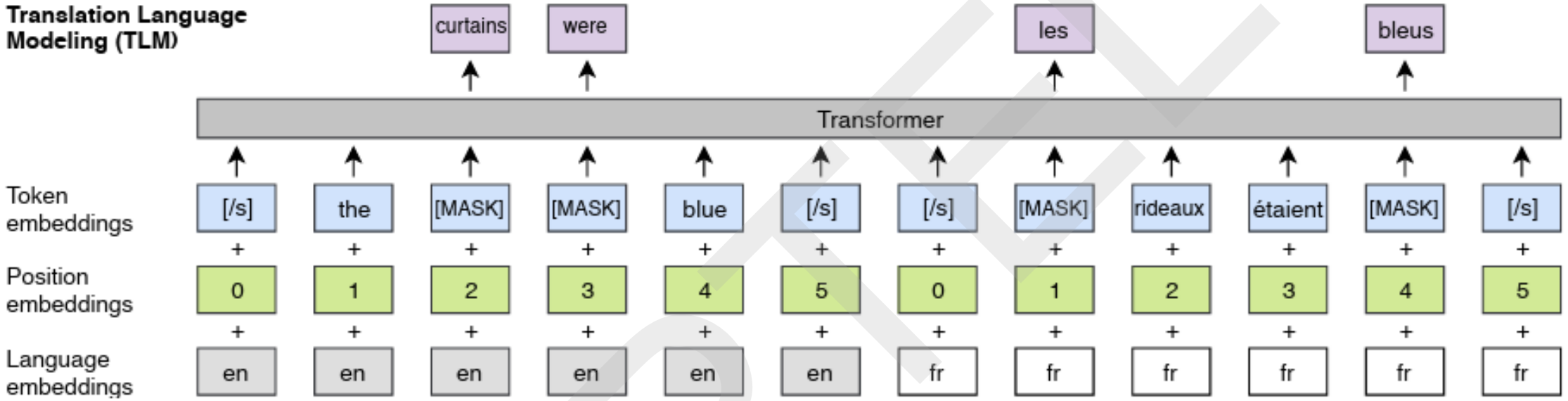
https://anoopkunchukuttan.gitlab.io/publications/presentations/
wintersc_iitguwahati_multilingual_model_jan25.pdf

Multilingual Models: mBERT

- `BERT-Large, Uncased (Whole Word Masking)` : 24-layer, 1024-hidden, 16-heads, 340M parameters
- `BERT-Large, Cased (Whole Word Masking)` : 24-layer, 1024-hidden, 16-heads, 340M parameters
- `BERT-Base, Uncased` : 12-layer, 768-hidden, 12-heads, 110M parameters
- `BERT-Large, Uncased` : 24-layer, 1024-hidden, 16-heads, 340M parameters
- `BERT-Base, Cased` : 12-layer, 768-hidden, 12-heads , 110M parameters
- `BERT-Large, Cased` : 24-layer, 1024-hidden, 16-heads, 340M parameters
- `BERT-Base, Multilingual Cased (New, recommended)` : 104 languages, 12-layer, 768-hidden, 12-heads, 110M parameters
- `BERT-Base, Multilingual Uncased (Orig, not recommended)` **(Not recommended, use** `Multilingual Cased` **instead)**: 102 languages, 12-layer, 768-hidden, 12-heads, 110M parameters
- `BERT-Base, Chinese` : Chinese Simplified and Traditional, 12-layer, 768-hidden, 12-heads, 110M parameters

https://github.com/google-research/bert?tab=readme-ov-file

XLM: Cross-Lingual Language Model Pretraining

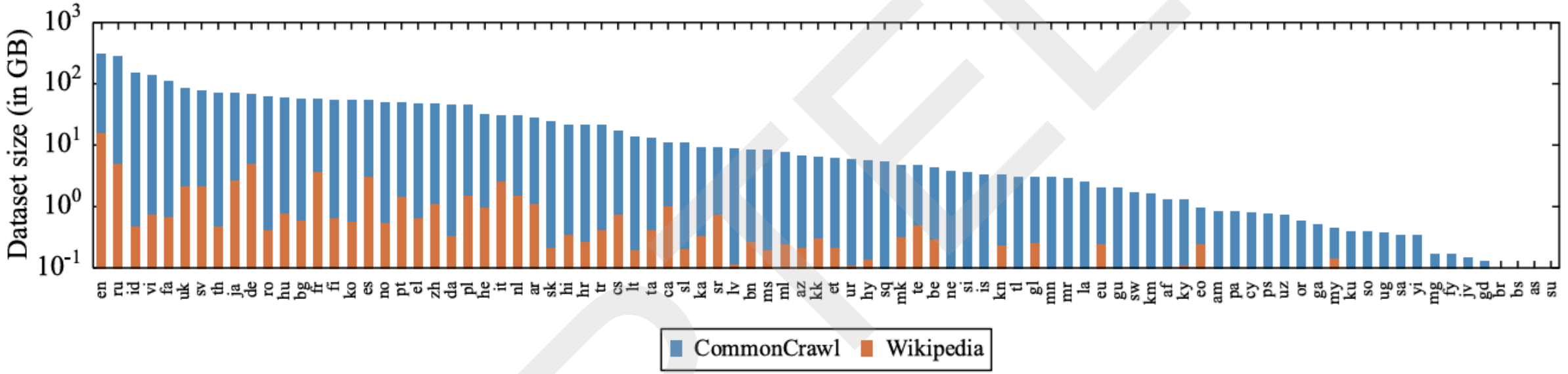


Input two sentence from different languages
Model alternates with MLM and TLM objectives
TLM ➔ the model can look at both sentences to predict masked token

https://arxiv.org/abs/1901.07291

https://anoopkunchukuttan.gitlab.io

XLM-R: Scaling the amount of training data



Amount of data in GiB (log-scale) for the 88 languages that appear in both the Wiki-100 corpus used for mBERT and XLM-100, and the CC-100 used for XLM-R. CC-100 increases the amount of data by several orders of magnitude, in particular for low-resource languages.

https://arxiv.org/pdf/1911.02116

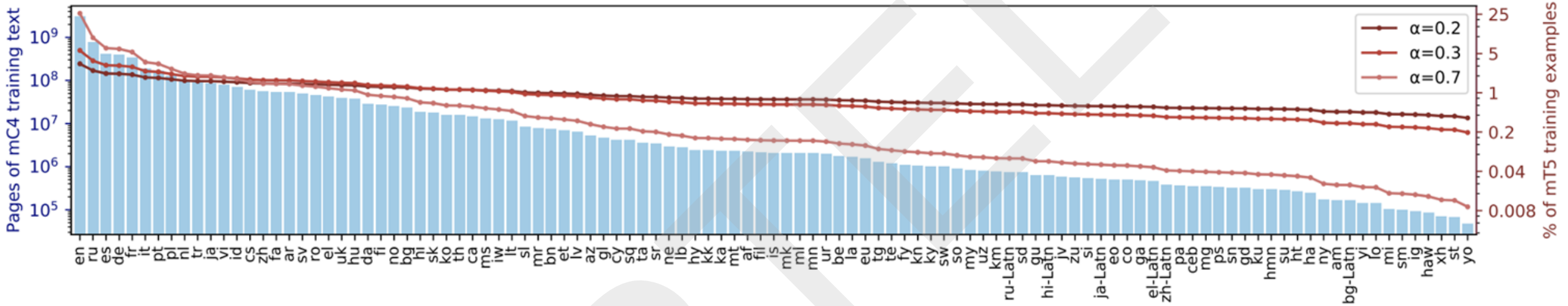
Results on MLQA

Model	train	#lgs	en	es	de	ar	hi	vi	zh	Avg
BERT-Large†	en	1	80.2 / 67.4	-	-	-	-	-	-	-
mBERT†	en	102	77.7 / 65.2	64.3 / 46.6	57.9 / 44.3	45.7 / 29.8	43.8 / 29.7	57.1 / 38.6	57.5 / 37.3	57.7 / 41.6
XLM-15†	en	15	74.9 / 62.4	68.0 / 49.8	62.2 / 47.6	54.8 / 36.3	48.8 / 27.3	61.4 / 41.8	61.1 / 39.6	61.6 / 43.5
XLM-R_{Base}	en	100	77.1 / 64.6	67.4 / 49.6	60.9 / 46.7	54.9 / 36.6	59.4 / 42.9	64.5 / 44.7	61.8 / 39.3	63.7 / 46.3
XLM-R	en	100	**80.6 / 67.8**	**74.1 / 56.0**	**68.5 / 53.6**	**63.1 / 43.5**	**69.2 / 51.6**	**71.3 / 50.9**	**68.0 / 45.4**	**70.7 / 52.7**

F1 and EM (exact match) scores for zero-shot classification where models are fine-tuned on the English SQuAD dataset and evaluated on the 7 languages of MLQA.

https://arxiv.org/pdf/1911.02116

mC4 Dataset (used by mT5)



107 languages, lower-resource languages upsampled based on their frequency in the dataset

https://people.cs.umass.edu/~miyyer/cs685/

mC4 Dataset (used by mT5)

Model	Architecture	Parameters	# languages	Data source
mBERT (Devlin, 2018)	Encoder-only	180M	104	Wikipedia
XLM (Conneau and Lample, 2019)	Encoder-only	570M	100	Wikipedia
XLM-R (Conneau et al., 2020)	Encoder-only	270M – 550M	100	Common Crawl (CCNet)
mBART (Lewis et al., 2020b)	Encoder-decoder	680M	25	Common Crawl (CC25)
MARGE (Lewis et al., 2020a)	Encoder-decoder	960M	26	Wikipedia or CC-News
mT5 (ours)	Encoder-decoder	300M – 13B	101	Common Crawl (mC4)

mT5 paper, Sue et al., ACL 2021

XLNI: Cross-Lingual Zero-shot learning

- We are given labeled training data for task X only in language A. Can we build a model that can make predictions for task X in a different language B?

- **Idea**: leverage information from high-resource languages to help improve performance on low-resource languages.

- **Zero-shot** learning: no labeled data is available for the target task X in language B, although unlabeled data in language B might be available for pretraining

https://people.cs.umass.edu/~miyyer/cs685/

XLNI Benchmark

Language	Premise / Hypothesis	Genre	Label
English	You don't have to stay there. You can leave.	Face-To-Face	Entailment
French	La figure 4 montre la courbe d'offre des services de partage de travaux. Les services de partage de travaux ont une offre variable.	Government	Entailment
Spanish	Y se estremeció con el recuerdo. El pensamiento sobre el acontecimiento hizo su estremecimiento.	Fiction	Entailment
German	Während der Depression war es die ärmste Gegend, kurz vor dem Hungertod. Die Weltwirtschaftskrise dauerte mehr als zehn Jahre an.	Travel	Neutral
Swahili	Ni silaha ya plastiki ya moja kwa moja inayopiga risasi. Inadumu zaidi kuliko silaha ya chuma.	Telephone	Neutral
Russian	И мы занимаемся этим уже на протяжении 85 лет. Мы только начали этим заниматься.	Letters	Contradiction
Chinese	让我告诉你，美国人最终如何看待你作为独立顾问的表现。 美国人完全不知道您是独立律师。	Slate	Contradiction

https://people.cs.umass.edu/~miyyer/cs685/

mT5: Benchmark Comparison

Model	Sentence pair		Structured	Question answering		
	XNLI	PAWS-X	WikiAnn NER	XQuAD	MLQA	TyDiQA-GoldP
Metrics	Acc.	Acc.	F1	F1 / EM	F1 / EM	F1 / EM
Cross-lingual zero-shot transfer (models fine-tuned on English data only)						
mBERT	65.4	81.9	62.2	64.5 / 49.4	61.4 / 44.2	59.7 / 43.9
XLM	69.1	80.9	61.2	59.8 / 44.3	48.5 / 32.6	43.6 / 29.1
InfoXLM	81.4	-	-	- / -	73.6 / 55.2	- / -
X-STILTs	80.4	87.7	64.7	77.2 / 61.3	72.3 / 53.5	76.0 / 59.5
XLM-R	79.2	86.4	65.4	76.6 / 60.8	71.6 / 53.2	65.1 / 45.0
VECO	79.9	88.7	65.7	77.3 / 61.8	71.7 / 53.2	67.6 / 49.1
RemBERT	80.8	87.5	**70.1**	79.6 / 64.0	73.1 / 55.0	77.0 / 63.0
mT5-Small	67.5	82.4	50.5	58.1 / 42.5	54.6 / 37.1	35.2 / 23.2
mT5-Base	75.4	86.4	55.7	67.0 / 49.0	64.6 / 45.0	57.2 / 41.2
mT5-Large	81.1	88.9	58.5	77.8 / 61.5	71.2 / 51.7	69.9 / 52.2
mT5-XL	82.9	89.6	65.5	79.5 / 63.6	73.5 / 54.5	75.9 / 59.4
mT5-XXL	**85.0**	**90.0**	69.2	**82.5 / 66.8**	**76.0 / 57.4**	**80.8 / 65.9**

https://arxiv.org/pdf/2010.11934v3

Adding translation data

Model	Sentence pair	
	XNLI	PAWS-X
Metrics	Acc.	Acc.
Cross-lingual zero-shot transfer (models fine-tu		
mBERT	65.4	81.9
XLM	69.1	80.9
InfoXLM	81.4	-
X-STILTs	80.4	87.7
XLM-R	79.2	86.4
VECO	79.9	88.7
RemBERT	80.8	87.5
mT5-Small	67.5	82.4
mT5-Base	75.4	86.4
mT5-Large	81.1	88.9
mT5-XL	82.9	89.6
mT5-XXL	**85.0**	**90.0**

What if we use a machine translation system to get more labeled data (e.g., translate all the labeled English text to other languages)?

Translate-train (models fine-tuned on English		
XLM-R	82.6	90.4
FILTER + Self-Teaching	83.9	91.4
VECO	83.0	91.1
mT5-Small	64.7	79.9
mT5-Base	75.9	89.3
mT5-Large	81.8	91.2
mT5-XL	84.8	91.0
mT5-XXL	**87.8**	**91.5**

https://people.cs.umass.edu/~miyyer/cs685/

Language Family specific: indicBERT

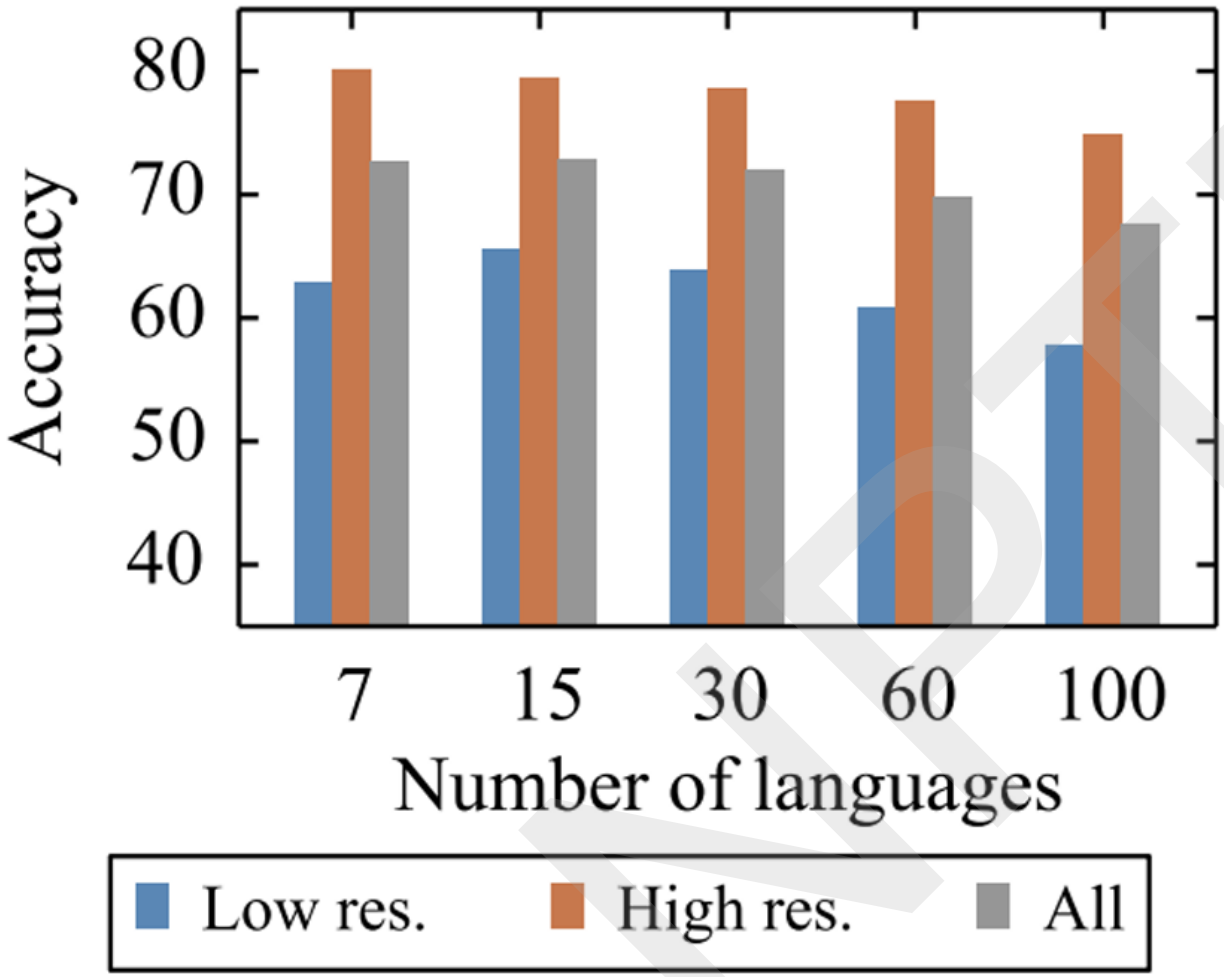
IndicBERT➔ model for 22 Indian languages + English)

Models	Classification					Structure Prediction		QA	Retreival
	Indic Sentiment	Indic XNLI	Indic COPA	Indic XPara.	MASSIVE (Intent)	Naama-Padam	MASSIVE (Slotfill)	Indic QA	FLORES
IndicBERT v1	61.8	42.8	51.0	47.5	-	25.3	-	10.1	1.1
mBERT	69.5	54.7	51.7	55.2	13.2	63.0	6.2	32.9	32.3
XLMR	84.0	69.7	60.1	56.7	66.6	71.7	50.0	44.8	3.1
MuRIL	85.1	72.4	58.9	**60.8**	77.2	**74.3**	57.0	48.3	52.3
v1-data	85.7	66.4	52.4	49.6	25.8	58.3	34.4	37.6	54.9
IndicBERT v2	**88.3**	73.0	62.7	56.9	78.8	73.2	56.7	47.7	69.4
+Samanantar	**88.3**	74.3	**63.0**	57.0	78.8	72.4	**57.3**	49.2	64.7
+Back-Trans.	87.5	69.7	53.8	50.7	77.4	71.9	54.6	42.2	68.6
IndicBERT-SS	88.1	**73.9**	64.2	56.4	**80.7**	66.6	**57.3**	**49.7**	**71.2**

Table 4: Results averaged across **languages** from the IndicXTREME benchmark. We report F1 scores for Structure Prediction & QA, and accuracy for the other tasks.

https://anoopkunchukuttan.gitlab.io

Curse of multilinguality



The "*curse of multilinguality*" (Conneau et al., 2020): For a fixed-size model, the per-language capacity decreases as we increase the number of languages…

https://people.cs.umass.edu/~miyyer/cs685/

Extending Vocabulary

<s> Gaganyaan is an Indian crewed orbital spacecraft intended to be the formative spacecraft of the Indian Human Spaceflight Programme.

<s> गगनयान <0xE0><0xA4><0x8F>क भारतीय चालक दल कक्षीय अंतरिक्ष यान है जिसका <0xE0><0xA4><0x89>द्देश्य भारतीय मानव अंतरिक्ष <0xE0><0xA4><0x89>ड़ान कार्यक्रम का प्रारंभिक अंतरिक्ष यान होना है।

Fertility = Average number of tokens per word

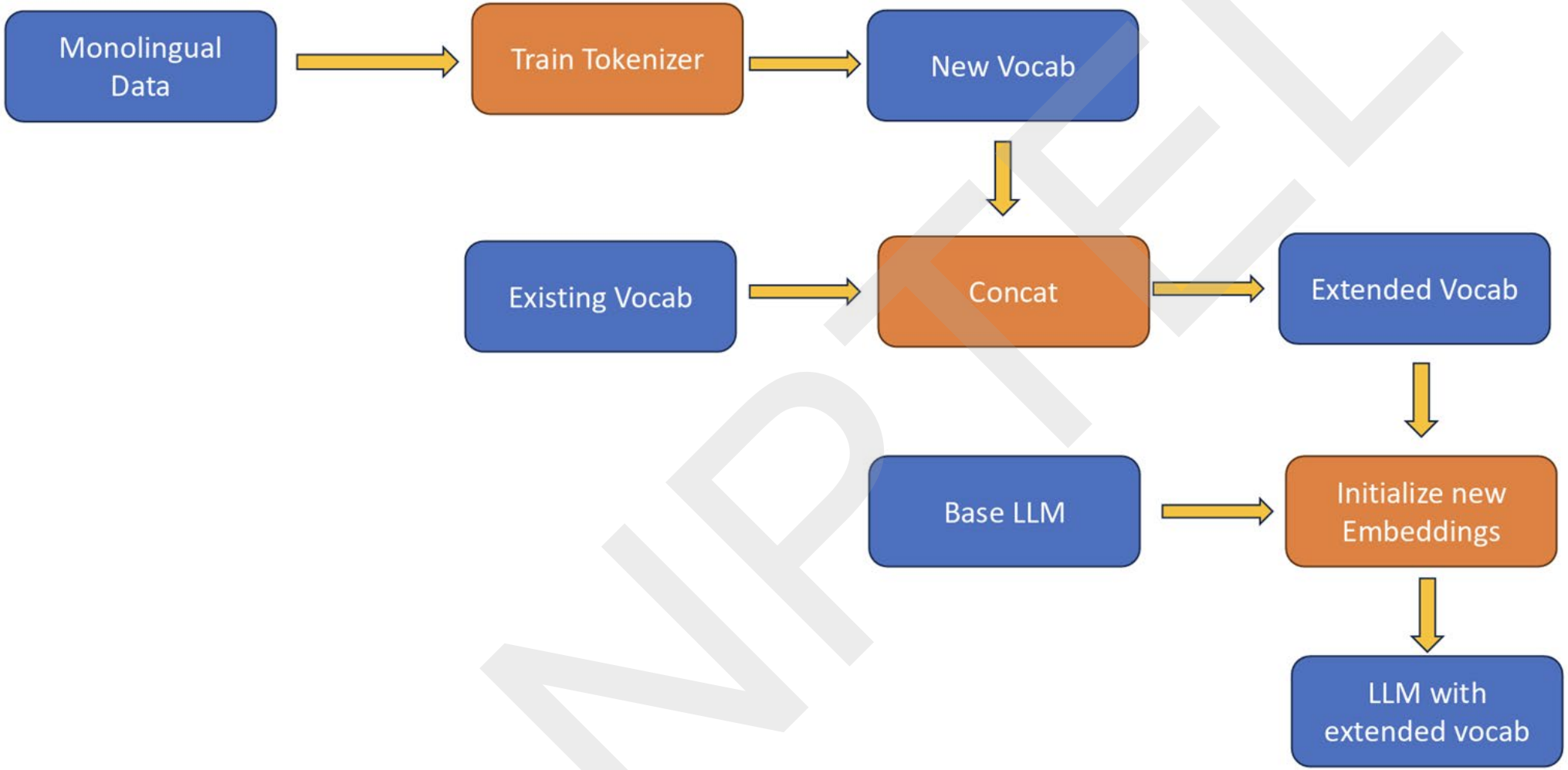
High Fertility → More memory consumption, more decoding time, limit on longest processable sequence

Model	Hindi Fertility
GPT4	5.32
Llama2	5.83
Mistral	5.60
BLOOM	1.38

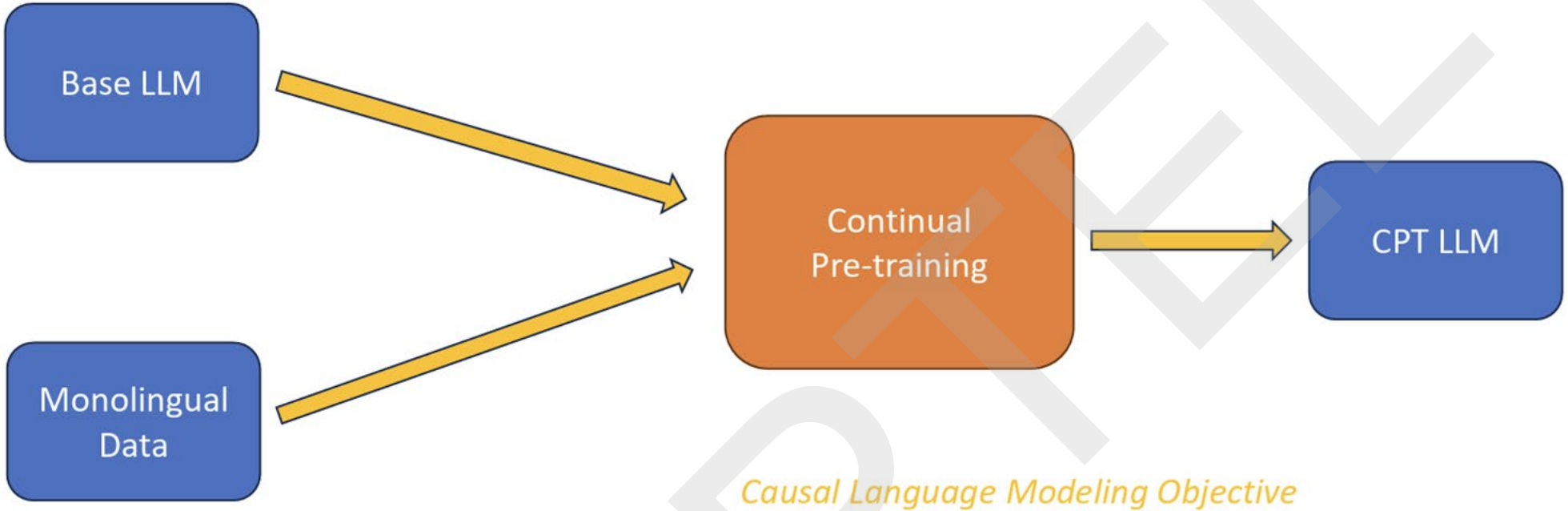
What if the vocabulary is under-represented?

https://anoopkunchukuttan.gitlab.io

How to extend tokenizer vocabulary?



Continually Pretrain



Causal Language Modeling Objective

$$p(\mathbf{x}) = p(x_1, x_2, ..., x_T) = \prod_{t=1}^{T} p(x_t|\mathbf{x}_{<t})$$

To avoid forgetting English competence and knowledge

- Include English in the pre-training data

To align English and new language

- Pre-train on parallel data
- Pre-train using romanized data

https://anoopkunchukuttan.gitlab.io

Related Concept: Uptraining

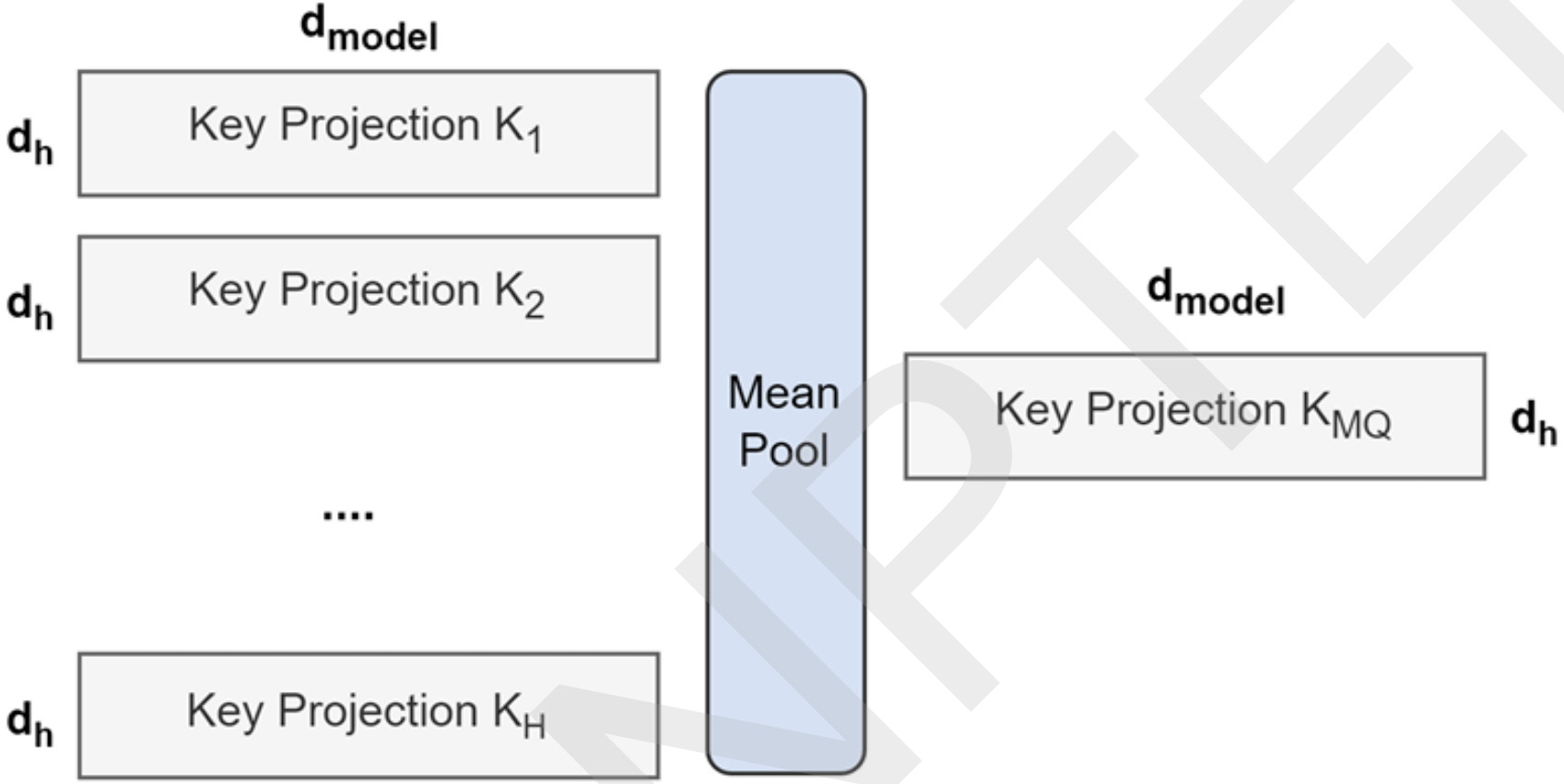
Uptraining: How to make pretrained models work with a slightly different architecture?

E.g., Group-query attention or Multi-query attention

Uptraining: Uptrained models are initialized from public T5.1.1 checkpoints. The key and value heads are mean-pooled to the appropriate MQA or GQA structure, and then pre-trained for a further (α=0.05) proportion of original pre-training steps with the original pre-training setup and dataset from (Raffel et al., 2020).

https://arxiv.org/pdf/2305.13245

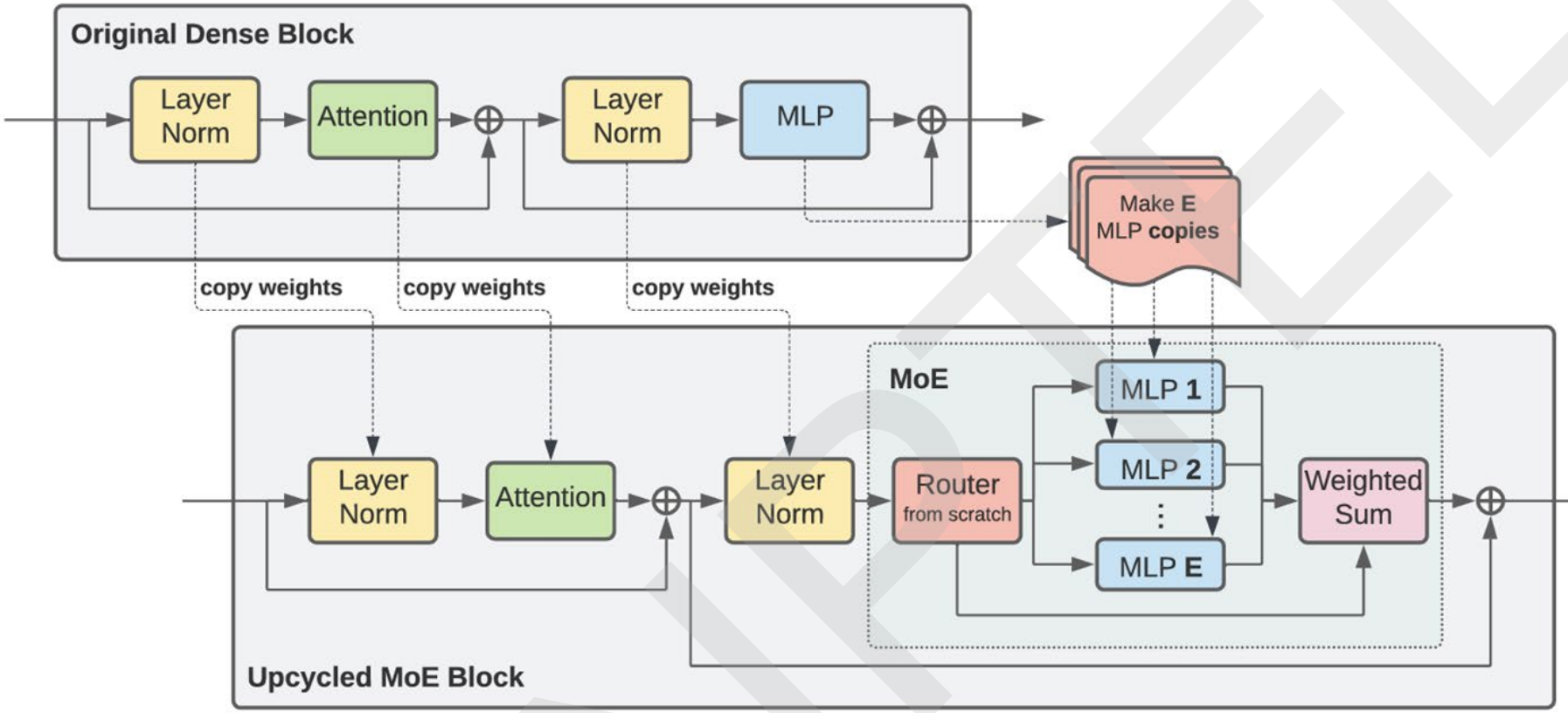
Uptraining: How to make pretrained models work with a slightly different architecture?



Overview of conversion from multi-head to multi-query attention. Key and value projection matrices from all heads are mean pooled into a single head.

https://arxiv.org/pdf/2305.13245

Uptraining: Training MoEs from dense checkpoints



All parameters, and optionally their optimizer state, are copied from the original checkpoint, except those corresponding to the MoE router, which does not exist in the original architecture. In particular, the experts in the new MoE layer are identical copies of the original MLP layer that is replaced.

https://arxiv.org/pdf/2212.05055

REFERENCES

Various reference papers with links in the slides


THANK YOU