Tokenization Contd. / Masked LMs

CSE 5525: Foundations of Speech and Natural Language Processing

https://shocheen.github.io/courses/cse-5525-spring-2025

THE OHIO STATE UNIVERSITY

Logistics

- Hw2 deadline: Monday.

- Final Project: sample project proposals are announced on teams. Will announce additional instructions for project proposals on Monday.

Last Class Recap: *Subword* Tokenization

- "Word"-level: issues with unknown words and information sharing, and gets complex fast
  - Also, fits poorly to some languages

- Character-level: long sequences, the model needs to do a lot of heavy lifting in representing that is encoded in plain-sight

- Subword tokenization – a middle ground
  - Byte Pair Encoding or BPE

Byte-Pair-Encoding (BPE) – **Token learner**

[coined by Gage et al., 1994; adapted to the task of word segmentation by **Sennrich et al., 2016**; see Gallé (2019) for more]

Raw train corpus ⇒ Vocabulary (a set of tokens)

- Pre-tokenize the corpus in words & append a special end-of-word symbol _ to each word
- Initialize vocabulary with the set of all individual **bytes**
- Choose 2 tokens that are most frequently adjacent ("A", "B")
    - Respect word boundaries: Run the algorithm inside words
- Add a new merged symbol ("AB") to the vocabulary
- Change the occurrence of the 2 selected tokens with the new merged token in the corpus
- Continues doing this until k merges are done

```
for i in range(num_merges):
  pairs = get_stats(vocab)
  best = max(pairs, key=pairs.get)
  vocab = merge_vocab(best, vocab)
```

[Jurafsky & Martin (2023)]

Byte-Pair-Encoding (BPE) – Token learner *Example*

end-of-word symbol

corpus	
5	l o w _
2	l o w e s t _
6	n e w e r _
3	w i d e r _
2	n e w _

word occurrence count in the corpus

each word is split into characters

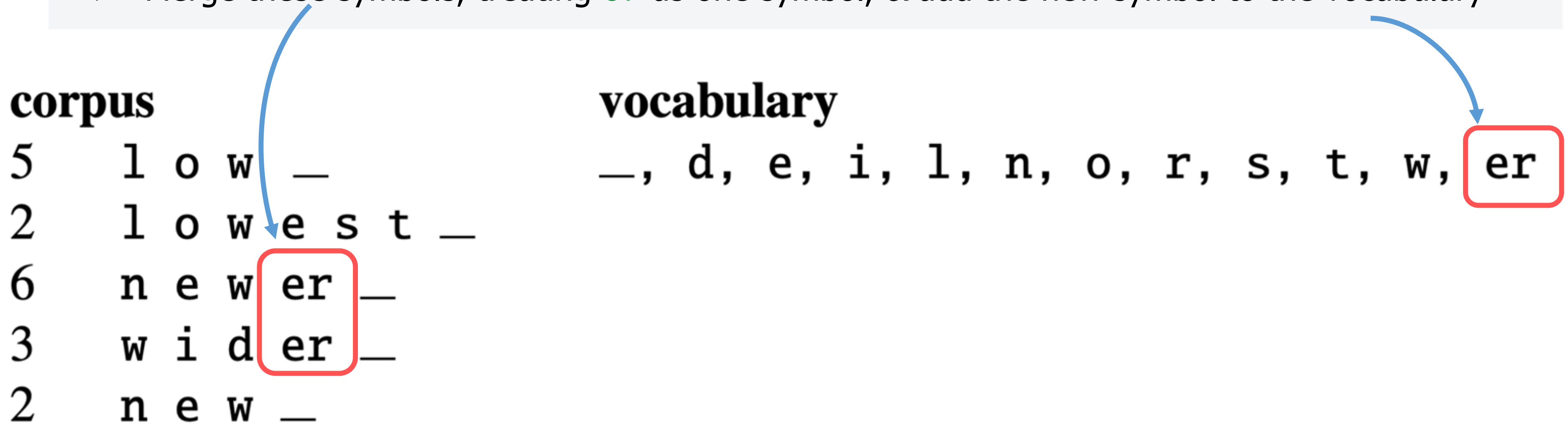
vocabulary

_, d, e, i, l, n, o, r, s, t, w

[Example from Jurafsky & Martin (2023), pages 18–19]

Byte-Pair-Encoding (BPE) – Token learner *Example*

- Counts all pairs of adjacent symbols
- The most frequent is the pair `e r` [a total of 9 occurrences]
- Merge these symbols, treating `er` as one symbol, & add the new symbol to the vocabulary



corpus

5	l o w _
2	l o w e s t _
6	n e w er _
3	w i d er _
2	n e w _

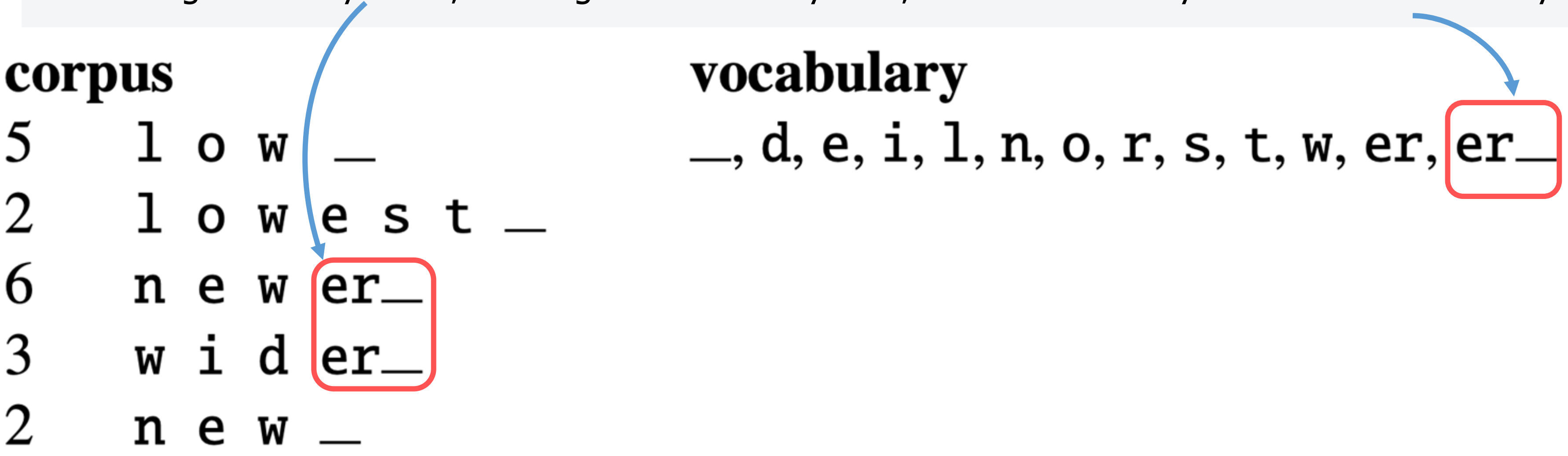
vocabulary

_, d, e, i, l, n, o, r, s, t, w, er

[Example from Jurafsky & Martin (2023), pages 18–19]

Byte-Pair-Encoding (BPE) – Token learner *Example*

- Counts all pairs of adjacent symbols
- The most frequent is the pair `e r _`
- Merge these symbols, treating `er_` as one symbol, & add the new symbol to the vocabulary



corpus

5	l o w _
2	l o w e s t _
6	n e w er_
3	w i d er_
2	n e w _

vocabulary

_, d, e, i, l, n, o, r, s, t, w, er, er_

[Example from Jurafsky & Martin (2023), pages 18–19]

Byte-Pair-Encoding (BPE) – Token learner *Example*

- Counts all pairs of adjacent symbols
- The most frequent is the pair n e
- Merge these symbols, treating ne as one symbol, & add the new symbol to the vocabulary

corpus

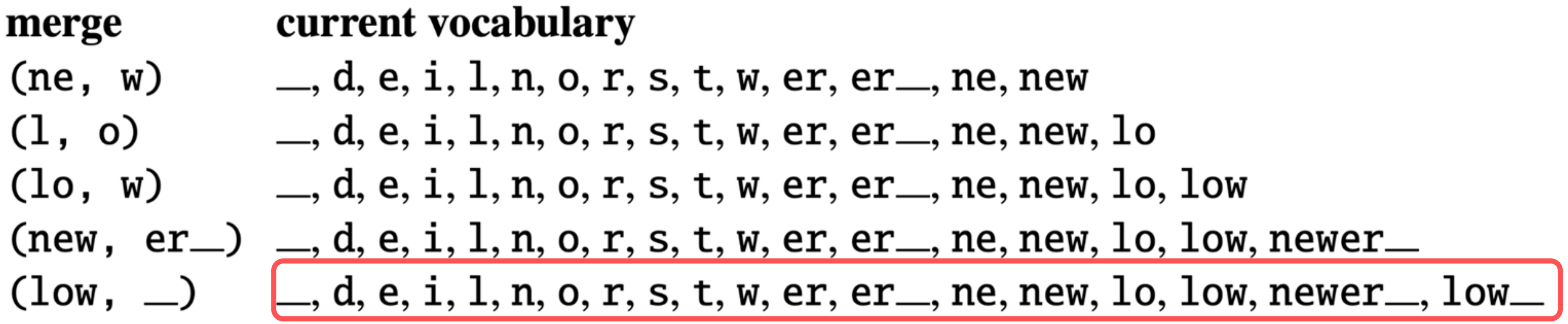
5	l o w _
2	l o w e s t _
6	ne w er_
3	w i d er_
2	ne w _

vocabulary

_, d, e, i, l, n, o, r, s, t, w, er, er_, ne

[Example from Jurafsky & Martin (2023), pages 18–19]

Byte-Pair-Encoding (BPE) – Token learner *Example*



merge	**current vocabulary**
(ne, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new
(l, o)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo
(lo, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low
(new, er_)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_
(low, _)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_, low_

Final vocabulary

[Example from Jurafsky & Martin (2023), pages 18–19]

After Training BPE

- You get a list of ordered merge rules

- A vocabulary: set of all unique tokens in the training corpus.

Byte-Pair-Encoding (BPE) – **Token segmenter**

[coined by Gage et al., 1994; adapted to the task of word segmentation by **Sennrich et al., 2016**; see Gallé (2019) for more]

- The token segmenter is used to tokenize a test sentence
  - Goal: Apply the merges we've learned from the training data, greedily, in the order we learned them

- First, we segment each test sentence word into characters

- Then, we apply the first merge rule
  - E.g., replace every instance of e r in the test corpus with er

- Then the second merge rule
  - E.g., replace every instance of er _ in the test corpus with er_

- And so on

[Example from Jurafsky & Martin (2023), pages 18–19]

Byte-Pair-Encoding (BPE) – **Token segmenter**

[coined by Gage et al., 1994; adapted to the task of word segmentation by **Sennrich et al., 2016**; see Gallé (2019) for more]

- Test example: slow_ → s l o w_ → s lo w_ -> s low_

- Test example: now → n o w

BPE can tokenize a word never seen at training time.

Leads to an open vocabulary model (well, almost)

Can often learn morphological segmentations

Deescalation → De escalat ion

[Example from Jurafsky & Martin (2023), pages 18–19]

BPE Summary

- A bottom-up tokenizer
  - Start with bytes / characters and keep merging until you reach a limit
  - A variant: WordPiece which uses longest prefix instead of merge rules to tokenize.

- Next up, Unigram-LM tokenizer
  - Takes a top-down approach

Unigram LM Tokenizer

1. Start with a **large base vocabulary**, **remove tokens** until a desired size is reached.

1. **How to construct a base vocabulary**: all substrings of pre-tokenized words

1. **How to remove tokens:**
   a. Compute **the unigram LM loss** over the corpus (more details later)
   b. Removing tokens increases this loss.
   c. Select and remove tokens that increase it the least.
   d. Repeat

Base Vocabulary

- The corpus:

  ("hug", 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)

- Initial Vocabulary (all strict substrings)

  ("h", 15) ("u", 36) ("g", 20) ("hu", 15) ("ug", 20) ("p", 17) ("pu", 17) ("n", 16)

  ("un", 16) ("b", 4) ("bu", 4) ("s", 5) ("hug", 15) ("gs", 5) ("ugs", 5)

Unigram LM loss

- Unigram LM loss = negative log **probability of the corpus**.

- **Probability of a corpus** = product of marginal **probability of individual words**

  p(pug hugs bugs) = p(pug) x p(hugs) x p (bugs)

Probability of a word

- product of marginal probability of its subwords (based on frequency)

  p(pug) = p("p") x p("u") x p("g")

  Or, p(pug) = p("pu") x p("g")

  Or, p(pug) = p("p") x p("ug")

  Choose highest of all possible splits

- How to this efficiently: Dynamic programming (Viterbi algorithm)

Unigram Tokenization Algorithm

1. Start with a base vocabulary

2. Compute the unigram loss, L, over the corpus

VERY SLOW!

3. For every token, w, in the vocabulary
    a. Remove w from the vocabulary and recompute the loss, L'(w)
    b. Define score(w) = L'(w) - L

4. Compute w* = min_w score(w).
    a. Remove w* from the vocabulary.
    b. Got to step 2. Repeat until a desired vocabulary size is reached.

Unigram Tokenization Algorithm (Slightly Faster)

1. Start with a base vocabulary

2. Compute the unigram loss, L, over the corpus

VERY SLOW!

3. For every token, w, in the vocabulary
    a. Remove w from the vocabulary and recompute the loss, L'(w)
    b. Define score(w) = L'(w) - L

4. Compute **W** = x% of tokens with the lowest score.
    a. Remove w* from the vocabulary.
    b. Got to step 2. Repeat until a desired vocabulary size is reached.

Unigram Tokenization Algorithm (Slightly Faster)

1. Start with a base vocabulary

1. Compute the unigram loss, L

1. For every token, w, in the vocabulary
   a. Remove w from the vocabulary and recompute the loss, L'(w)
   b. Define score(w) = L'(w) - L

1. Compute **W** = x% of tokens with the lowest score.
   a. Remove **W** from the vocabulary.
   b. Go to step 2.

How to tokenize once the vocabulary is decided

- Tokenization which maximizes the unigram probability of the word
- (or find top *k* tokenizations)
- "Unhug" (For each position, the subwords with the best scores ending in that position:)

  Character 0 (u): "u" (score 0.171429)

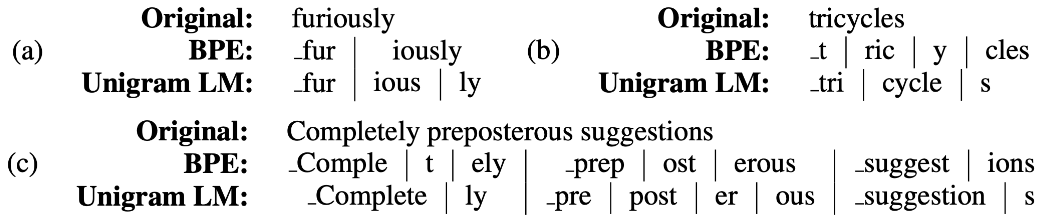
  Character 1 (n): "un" (score 0.076191)

  Character 2 (h): "un" "h" (score 0.005442)

  Character 3 (u): "un" "hu" (score 0.005442)

  Character 4 (g): "un" "hug" (score 0.005442) **[final tokenization]**

Unigram vs BPE



(a)	**Original:**	furiously
	BPE:	_fur | iously
	Unigram LM:	_fur | ious | ly
(b)	**Original:**	tricycles
	BPE:	_t | ric | y | cles
	Unigram LM:	_tri | cycle | s
(c)	**Original:**	Completely preposterous suggestions
	BPE:	_Comple | t | ely | _prep | ost | erous | _suggest | ions
	Unigram LM:	_Complete | ly | _pre | post | er | ous | _suggestion | s

1. Why unigram over BPE and WordPiece?
   a. Unigram finds **optimal coding length** of a sequence (according to Shannon's entropy).
   b. Unigram allows sampling multiple tokenizations for every word – subword regularization – robustness

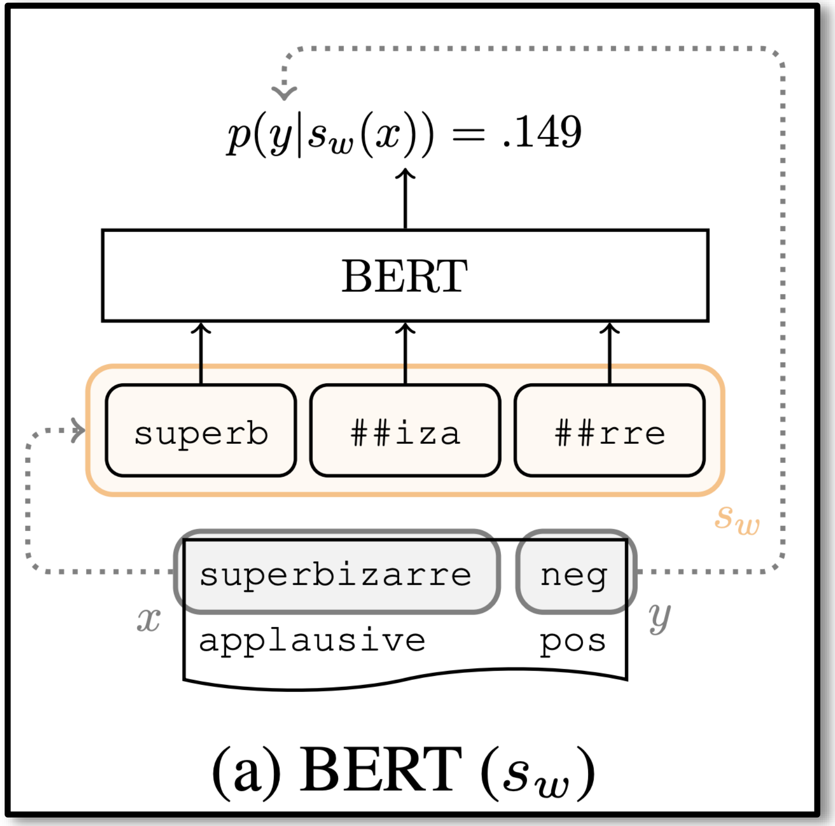
1. Why is Unigram tokenization not more popular?
   a. In many papers simply referred to as SentencePiece, which is actually not a tokenizer but a library wrapping several tokenizers
   b. Does the actual subword tokenizer matter as we scale up models?

Subword Models -- Summary

- Split text in tokens learned statistically from the training corpus
- Makes the model open vocabulary
- Subword methods prove to be an effective method of “compressing text”

Issues with subword models

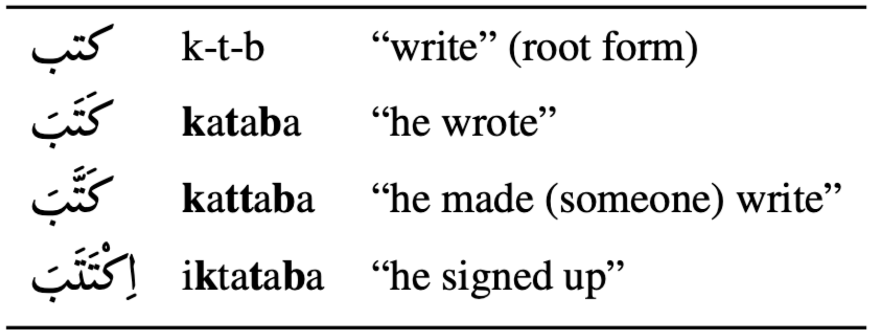
BERT thinks the sentiment of "superbizarre" is positive because its tokenization contains the token "superb"



(a) BERT (s_w)

[Hofmann et al., 2021]

Non-concatenative Languages



كتب	k-t-b	“write” (root form)
كَتَبَ	**k**a**t**a**b**a	“he wrote”
كَتَّبَ	**k**a**tt**a**b**a	“he made (someone) write”
اِكْتَتَبَ	i**k**ta**t**a**b**a	“he signed up”

Table 1: Non-concatenative morphology in Arabic.[4] The root contains only consonants; when conjugating, vowels, and sometimes consonants, are interleaved with the root. The root is not separable from its inflection via any contiguous split.

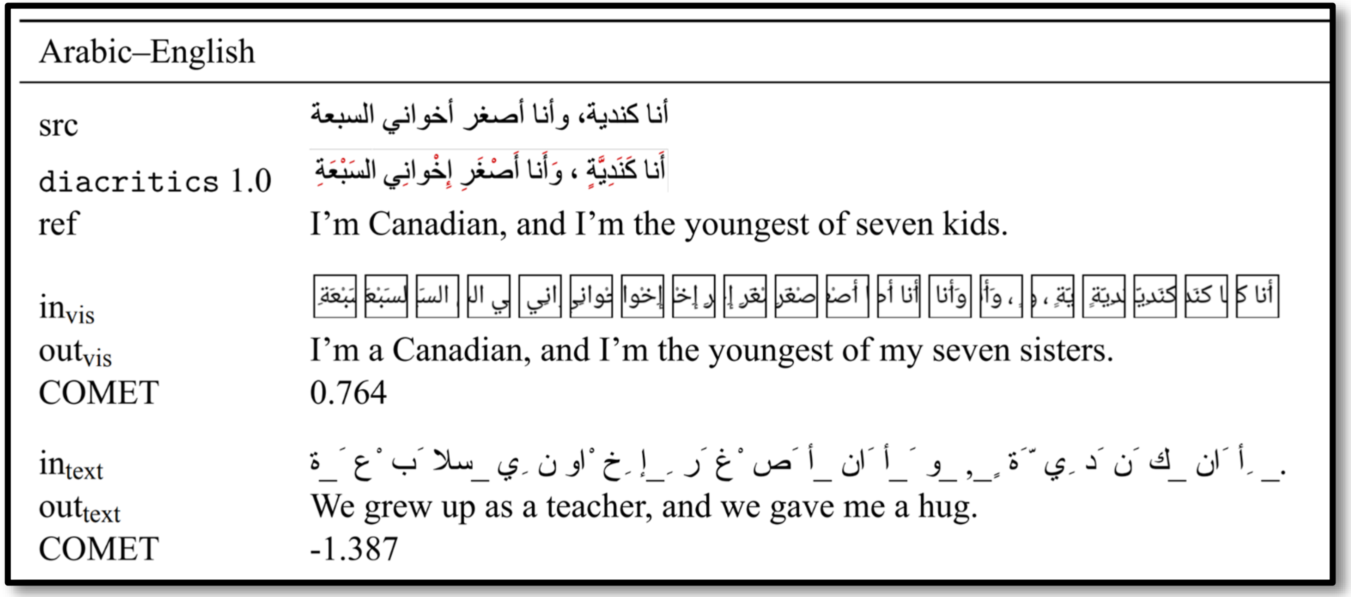
Subword Tokenization and “noise”

- He fell and broke his **coccix** (vs coccyx)
- Neighbor = 1 token, neighbour = two tokens
- John Smith = 2 tokens, Srinivas Ramanujam = ??

Subword Tokenization and “numbers”

- Why are LMs bad at basic arithmetic?
  - 3.14 is tokenized as 3.14
  - 3.15 is tokenized as 3 . 15

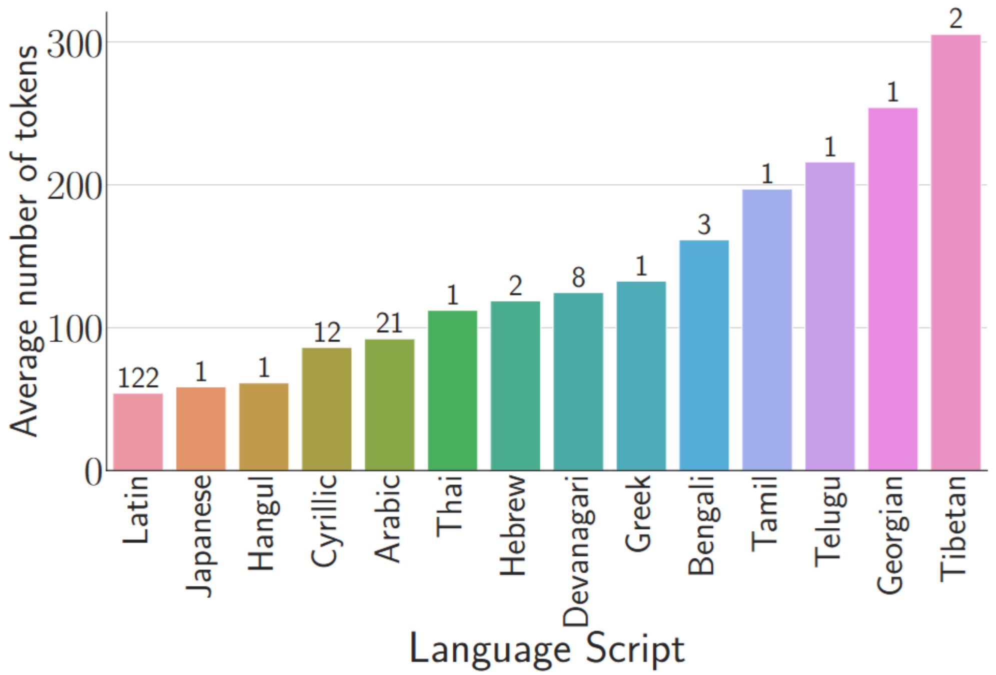
Natural phenomena like diacritics or a little easily human-readable noise lead to unexpected BPE sequences and catastrophic translation failure



Arabic–English	
src	أنا كندية، وأنا أصغر أخواني السبعة
diacritics 1.0	أَنا كَنَدِيَّةٍ ، وَأَنا أَصْغَرِ إِخْوانِي السَبْعَةِ
ref	I'm Canadian, and I'm the youngest of seven kids.
in_{vis}	أنا ك | ا كنَ | كَنَديَّ | دِيَّةٍ | يَّةٍ ، | ٍ ، وَ | وَأَ | وَأَنا | أنا أ | أَص | أَصْغَرِ | صْغَرِ | غَرِ إ | رِ إِخْ | إِخْوا | خْوانِ | انِي | ي ال | السَ | لسَبْعَ | سَبْعَةِ
out_{vis}	I'm a Canadian, and I'm the youngest of my seven sisters.
COMET	0.764
in_{text}	_ ِأ َان _ك َن َد ِي ّ َة ٍ_ ، _و َ_أ َان _أ َص ْغ َر ِ_إ ِخ ْاو ن ِي _سلا َب ْع َ_ة .
out_{text}	We grew up as a teacher, and we gave me a hug.
COMET	-1.387

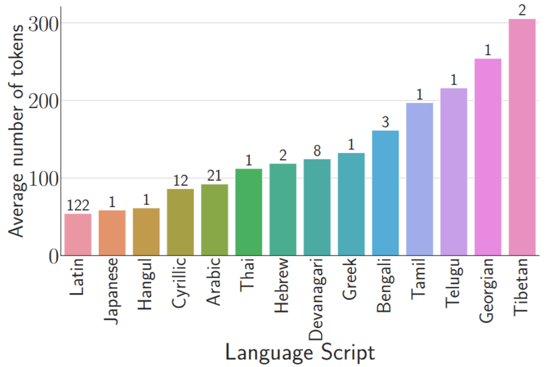
[Salesky et al., 2021]

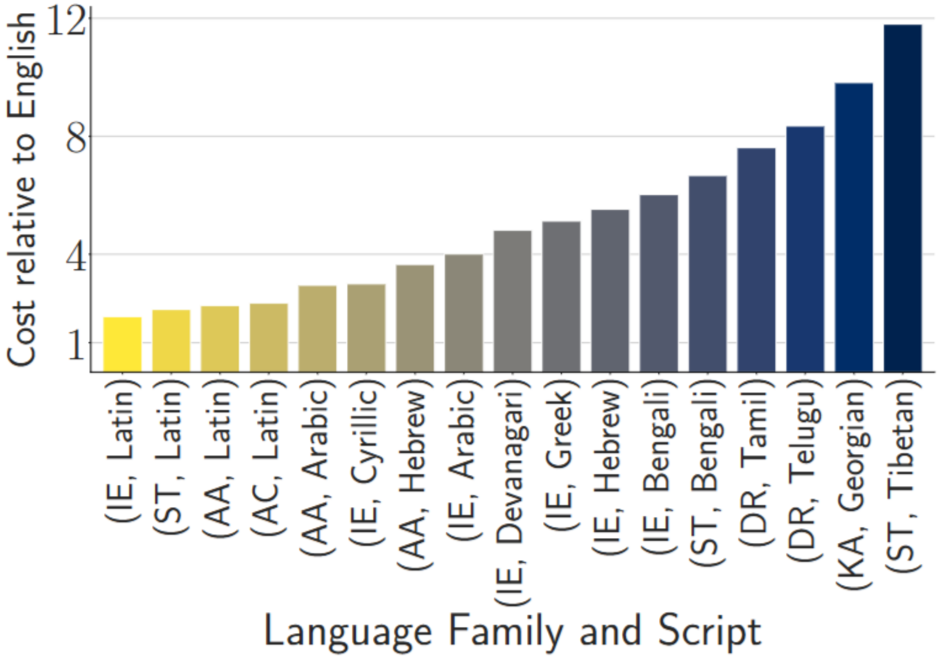
Sequence Lengths, Costs, and Performance



[2305.13707] Do All Languages Cost the Same? Tokenization in the Era of Commercial Language Models

Sequence Lengths, Costs, and Performance

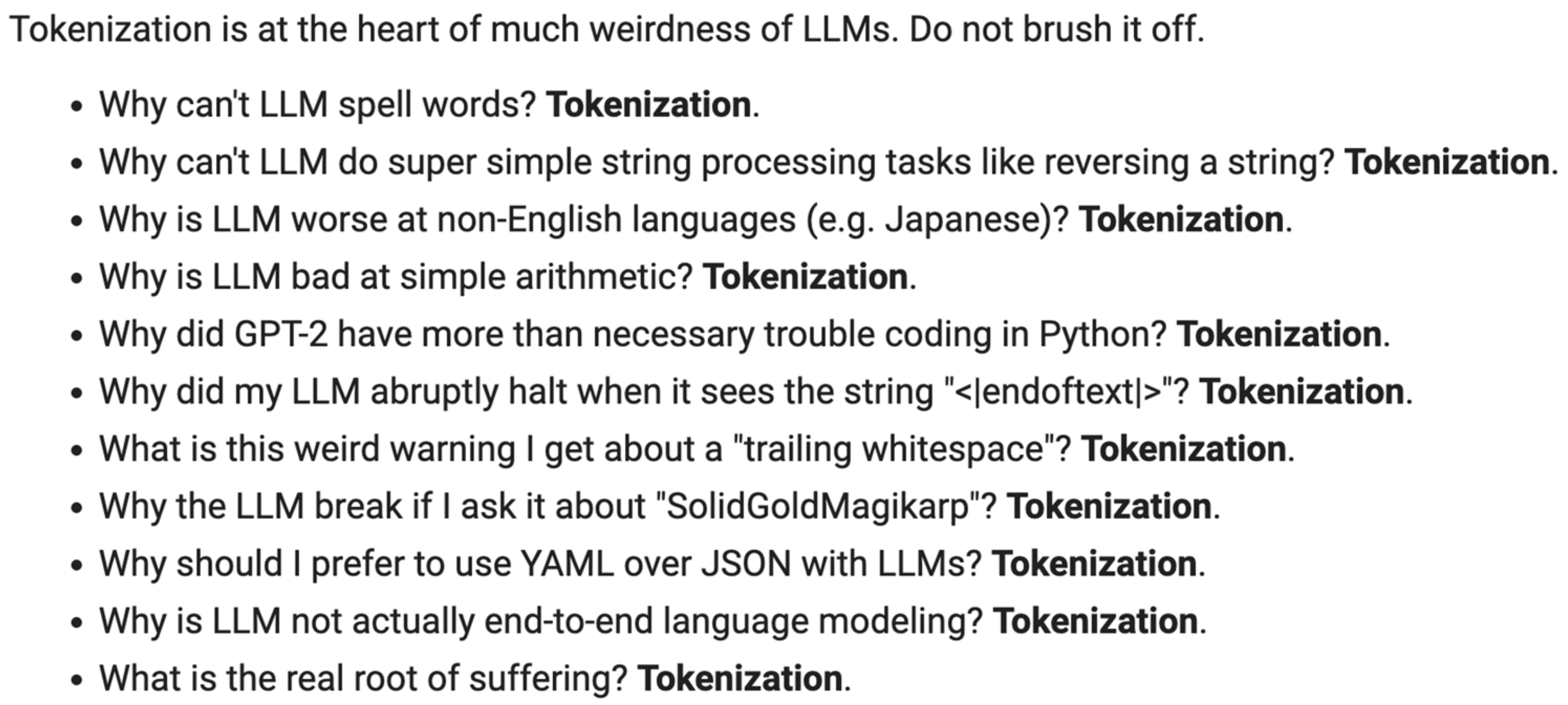




LLM APIs charge per token

[2305.13707] Do All Languages Cost the Same? Tokenization in the Era of Commercial Language Models

Tokenization



Tokenization is at the heart of much weirdness of LLMs. Do not brush it off.

- Why can't LLM spell words? **Tokenization**.
- Why can't LLM do super simple string processing tasks like reversing a string? **Tokenization**.
- Why is LLM worse at non-English languages (e.g. Japanese)? **Tokenization**.
- Why is LLM bad at simple arithmetic? **Tokenization**.
- Why did GPT-2 have more than necessary trouble coding in Python? **Tokenization**.
- Why did my LLM abruptly halt when it sees the string "<|endoftext|>"? **Tokenization**.
- What is this weird warning I get about a "trailing whitespace"? **Tokenization**.
- Why the LLM break if I ask it about "SolidGoldMagikarp"? **Tokenization**.
- Why should I prefer to use YAML over JSON with LLMs? **Tokenization**.
- Why is LLM not actually end-to-end language modeling? **Tokenization**.
- What is the real root of suffering? **Tokenization**.

[`Let's build the GPT Tokenizer' by Andrej Karpathy]

Cody Blakeney @code_star

Do you guys ever think about tokenizers?

Prithviraj (Raj) Ammanabrolu @rajammanabrolu · Sep 26, 2023

If there's one thing standing in the way of AGI, it's tokenizers

Luca Soldaini @soldni · Aug 21

Masked Language Models

Reminder: Different senses, but same embedding 👎

- *The Amazon **Basin** is home to the largest rainforest on Earth.*
- *She filled the **basin** with water to wash the dishes.*
- *The neurosurgeon examined the cranial **basin** for signs of trauma.*

word2vec, GloVe, or other **static embeddings** assign exactly the same embedding to each of these occurrences of the word "basin" despite their different senses

Our next goal:
Assign different, **contextualized embeddings** based on the surrounding context

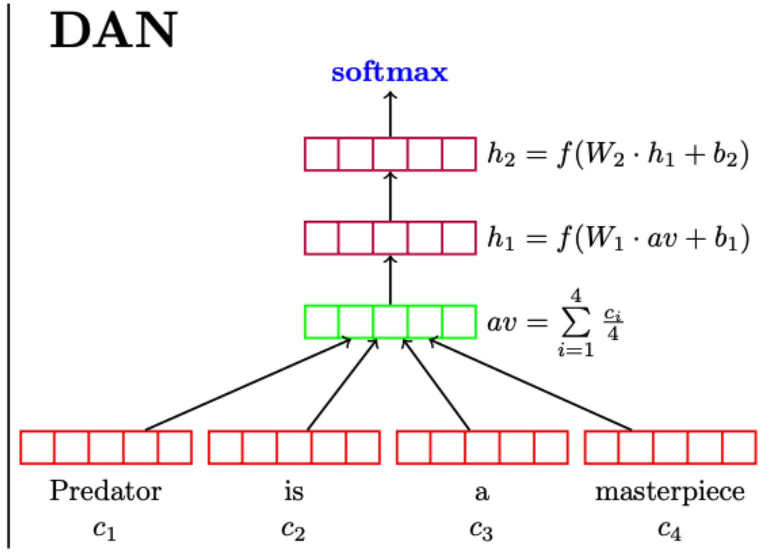
The final transformer encoder representation of each token is highly contextualized

Reminder: Token order & long-range dependencies with DAN 👎

Deep Averaging Network (DAN)

Transformer considers token order with positional embeddings

Due to multi-headed self-attention & cross-attention, we are able to model long-range dependencies


DAN
softmax
$h_2 = f(W_2 \cdot h_1 + b_2)$
$h_1 = f(W_1 \cdot av + b_1)$
$av = \sum_{i=1}^{4} \frac{c_i}{4}$
Predator c_1
is c_2
a c_3
masterpiece c_4

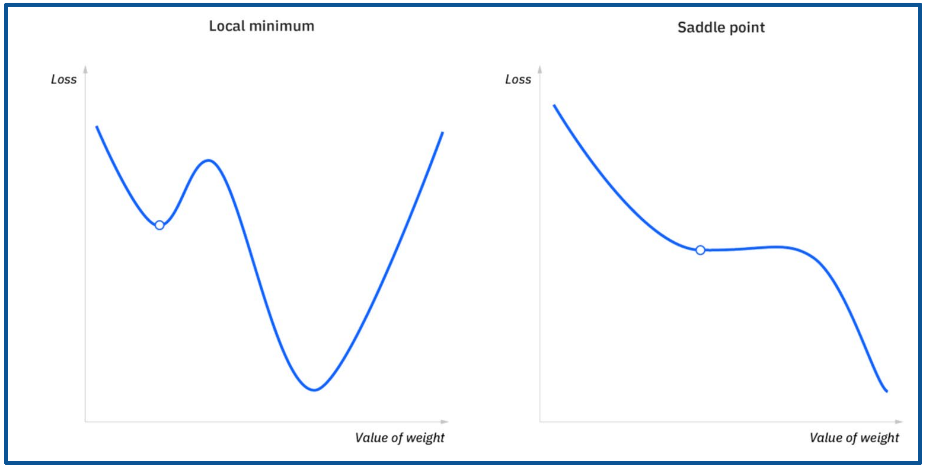
[Iyyer et al., 2015]

Reminder: Non-convex optimization, so initialization matters

Instead of starting from randomly initialized weights, we are going to make use of **large corpora** to **pretrain a model** and find weights that are a much better **starting point for all NLP tasks** than random weights

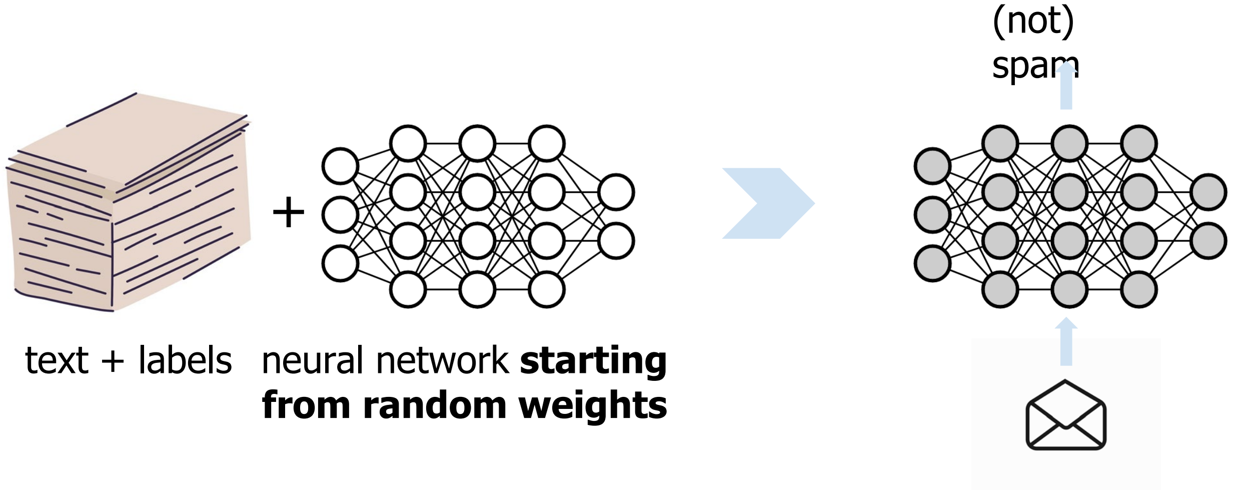
We are going to talk about this next!

Randomly initialized transformer still has this issue



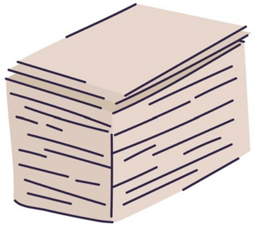
Source: https://www.ibm.com/topics/gradient-descent

Standard Supervised Deep Learning



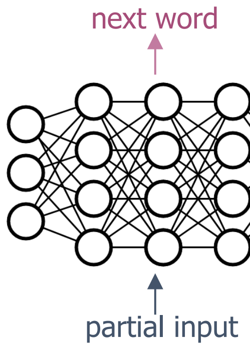
Pretrain-then-Finetune (also called post-train)

Stage 1:
Pretrain a model



text

+



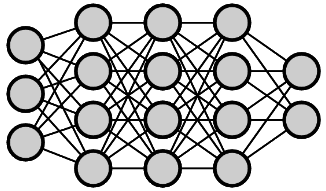
neural network
starting from
<u>random</u>
weights

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

Stage 2:
Finetune the model

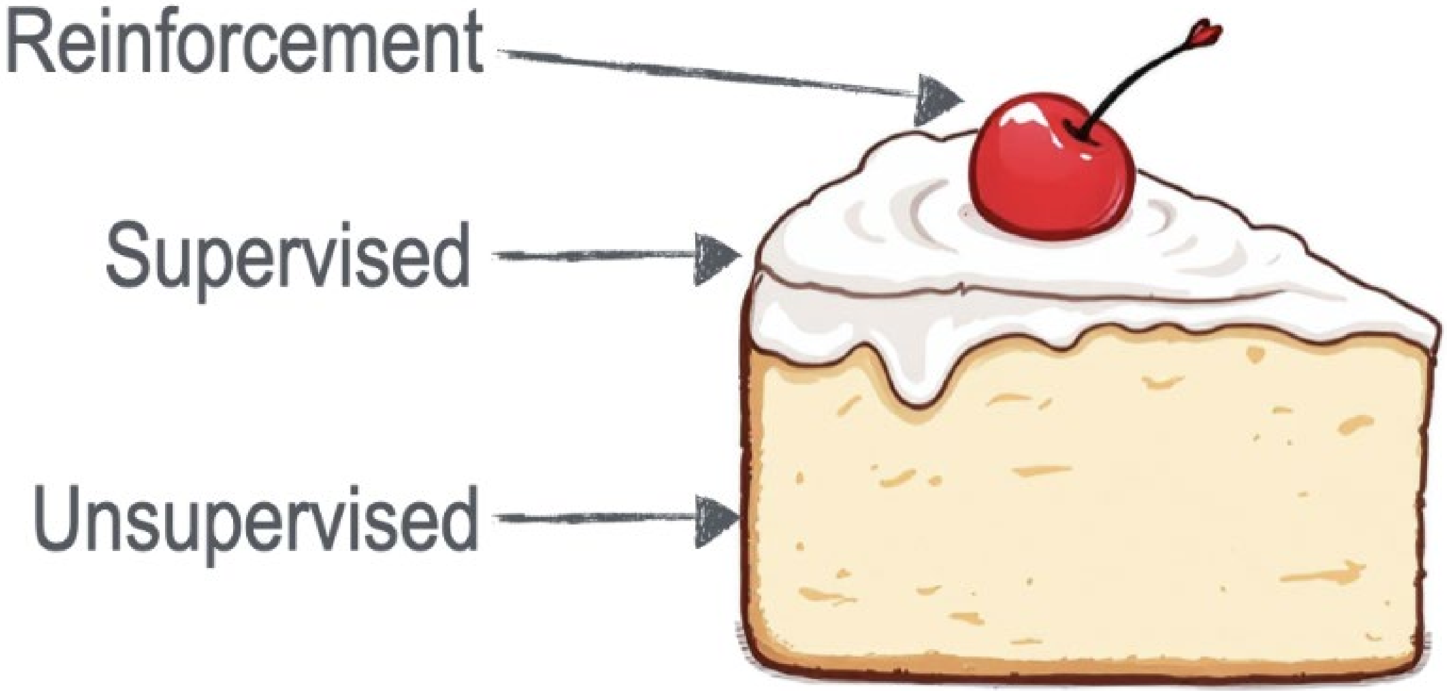
text + **labels**

+



neural network
starting from
<u>pretrained</u>
weights

Objective: standard supervised training



arxiv.org/pdf/2502.03038

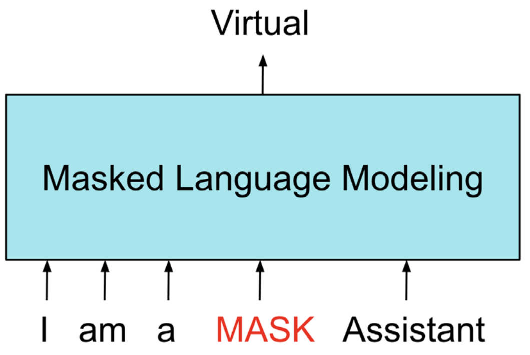
LM pretraining

- LMs so far: predict the next token given the previous tokens also sometimes called “Causal Language Modeling”.
- This enables a self-supervised task – train on only raw text (similar to word2vec).
- And get really interesting and useful representations.
  - With many usecases

Masked LMs

- LMs so far: predict the next token given the previous tokens
- Since we have a complete sentence?
  - So: can we incorporate future context to learn better representations
- How can we formulate a self-supervised prediction task?

Masked LMs



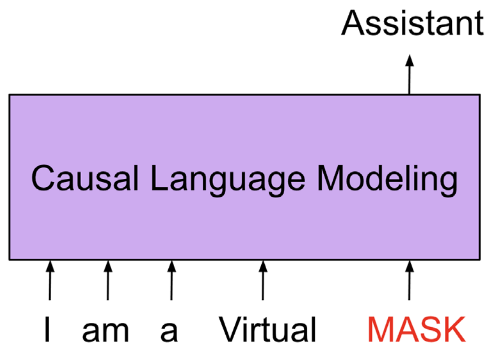
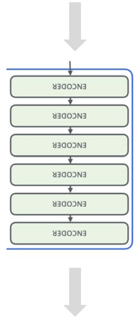


Image from https://www.holisticai.com/blog/from-transformer-architecture-to-prompt-engineering

Masked Language Modeling (MLM)

(text) Sylvester Stallone has made some **terrible** films in his lifetime, but this has got to be one of the **worst**. A totally **dull** story...

(masked text) Sylvester Stallone has made some **<mask>** films in his lifetime, but this has got to be one of the **<mask>**. A totally **<mask>** story...

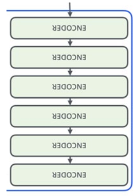


(output final layer) vector(Sylvester) ... vector(**<mask>**) vector(films) vector(in) vector(his) ... vector(the) vector(**<mask>**) ... vector(**<mask>**) vector(story)...

Masked Language Modeling (MLM)

(text) Sylvester Stallone has made some **terrible** films in his lifetime, but this has got to be one of the **worst**. A totally **dull** story...

(masked text) Sylvester Stallone has made some **<mask>** films in his lifetime, but this has got to be one of the **<mask>**. A totally **<mask>** story...



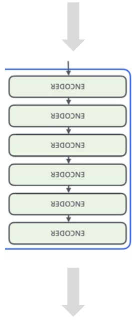
(output final layer) vector(Sylvester) ... vector(**<mask>**) vector(films) vector(in) vector(his) ... vector(the) vector(**<mask>**) ... vector(**<mask>**) vector(story)...

loss(predicted_word,“terrible”)

Masked Language Modeling (MLM)

(text) Sylvester Stallone has made some **terrible** films in his lifetime, but this has got to be one of the **worst**. A totally **dull** story...

(masked text) Sylvester Stallone has made some **<mask>** films in his lifetime, but this has got to be one of the **<mask>**. A totally **<mask>** story...



(output final layer) vector(Sylvester) ... vector(**<mask>**) vector(films) vector(in) vector(his) ... vector(the) vector(**<mask>**) ... vector(**<mask>**) vector(story)...

loss(predicted_word,"worst")

loss(predicted_word,"dull")

loss(predicted_word,"terrible")

} average these losses and do a gradient descent step

Masked LM: Only using Encoder transformer

- Encoders assume we have the complete sequence

- Self-attention computes weighted sum over entire context (i.e., entire sequence)

- There is no generation problem, we just want representations
  - We will learn how to use them later on

BERT

Bidirectional Encoder Representations from Transformers

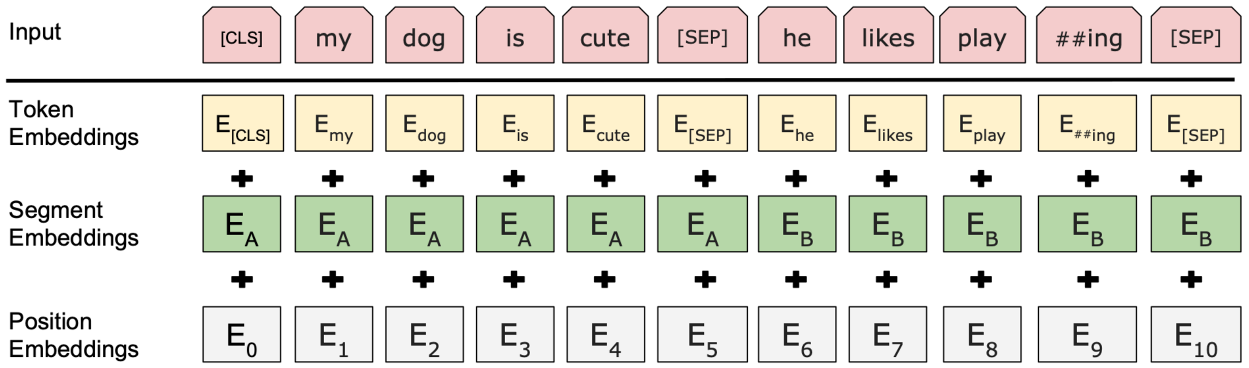
- Encoder transformer
- BERT Base: 12 transformer blocks, 768-dim word-piece tokens, 12 self-attention heads $\rightarrow$ 110M parameters
- BERT Large: 24 transformer blocks, 1024-dim word-piece tokens, 16 self-attention heads $\rightarrow$ 340M parameters
- 100's of variants since BERT came out.

[Devlin et al. 2018]

BERT

Inputs

- One or two sentences
  - Word-piece token embeddings
  - Position and segment embeddings



[figure from Devlin et al. 2018]

BERT

Training

- Data: raw text
- Two objectives:
  - Masked LM
  - Next-sentence prediction
- Later development in “RoBERTa”:
  - More data, no next-sentence prediction, dynamic masking

BERT
Masking Recipe for Training

- Randomly mask and predict 15% of the tokens
  - For 80% (of these 15%) replace the input token with a special token [MASK]
  - For 10% replace with a random token from the vocabulary
  - For 10% keep the same (input and output are the same, trivial task)

BERT

Next-sentence Prediction

- Input: [CLS] Text chunk 1 [SEP] Text chunk 2
- Training data: 50% of the time, take the true next chunk of text, 50% of the time take a random other chunk
- Predict whether the next chunk is the true next chunk
- Prediction is done on the [CLS] output representation

BERT

Related Techniques

- Central Word Prediction Objective (context2vec) [Melamud et al. 2016]
- Machine Translation Objective (CoVe) [McMann et al. 2017]
- Bi-directional Language Modeling Objective (ELMo) [Peters et al. 2018]
- Then BERT came …
- … and many more followed (the whole sesame street arrived)

BERT

What Do We Get?

- We can feed complete sentences to BERT
- For each token, we get a contextualized representation
  - Meaning: computed taking the other tokens in the sentence into acocunt
- In contrast to word2vec representations that fixed and do not depend on context
- While word2vec vectors are forced to mix multiple senses, BERT can provide more instance-specific vectors

BERT

How Do We Use It?

- Widely supported by existing frameworks
  - E.g., Transformers library by Hugging Face
- We will soon see how to use it when working with annotated data
- Large BERT models quickly outperformed human performance on several NLP tasks
  - But what it meant beyond benchmarking was less clear
- Started an arms race towards bigger and bigger models, which quickly led to the LLMs of today

BERT

What It Is Not Great For?

- primarily used for discriminative tasks, Cannot generate text
  - Can manipulate in weird ways to generate text but not the original purpose of this model

Finetuning a MLM-pretrained model

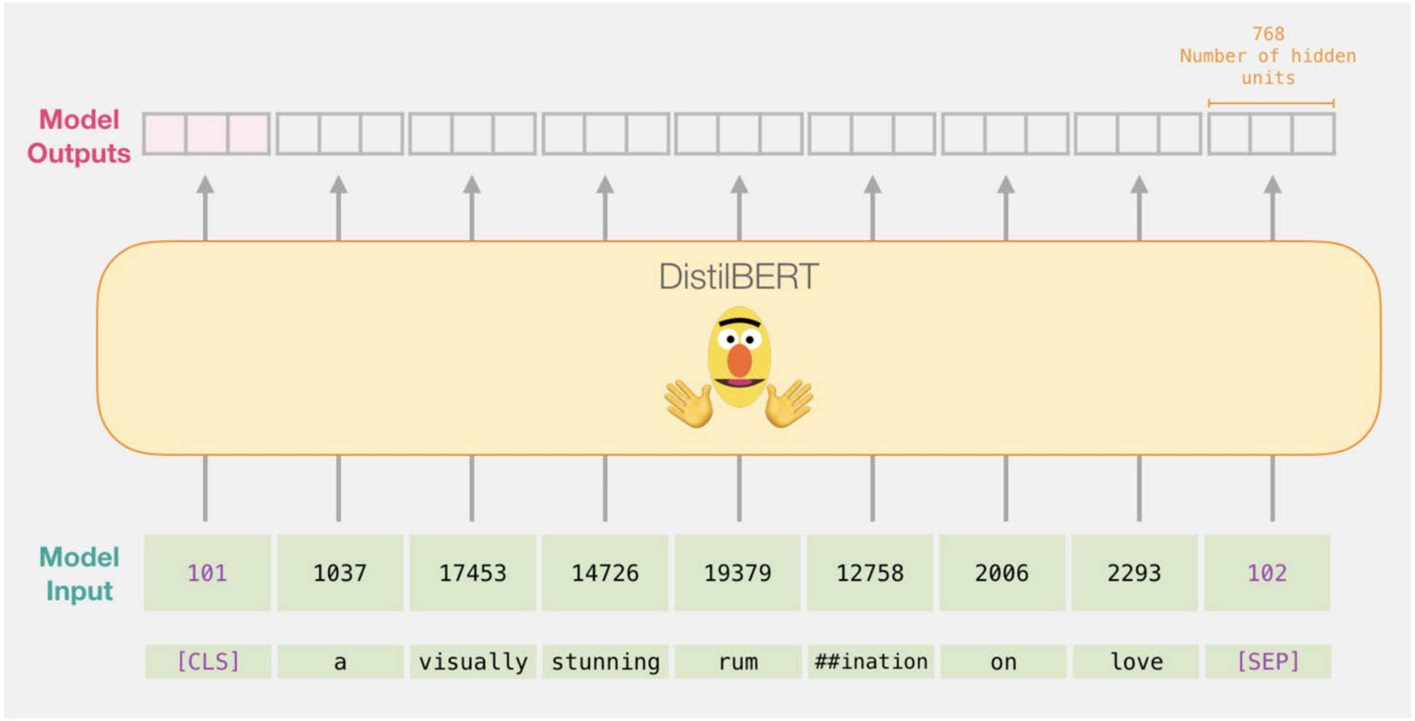


Figure: Jay Alammar

Finetuning a MLM-pretrained model

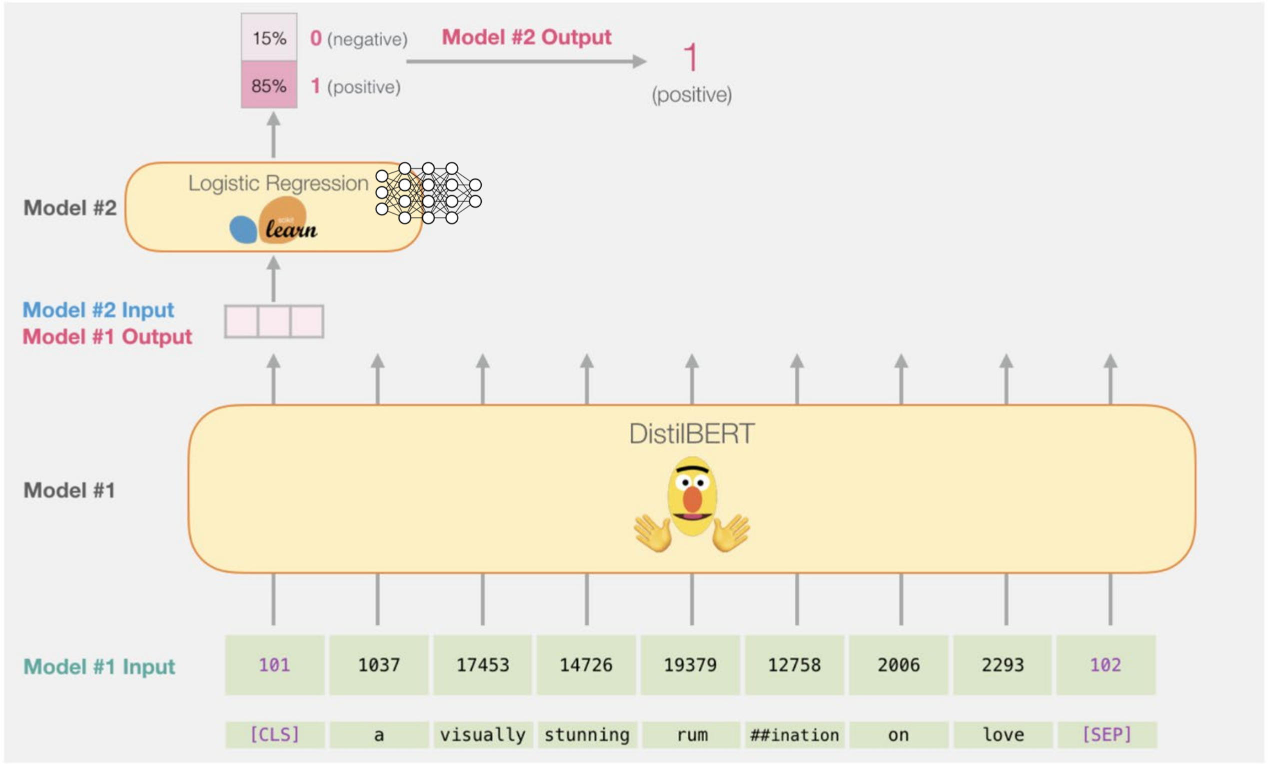
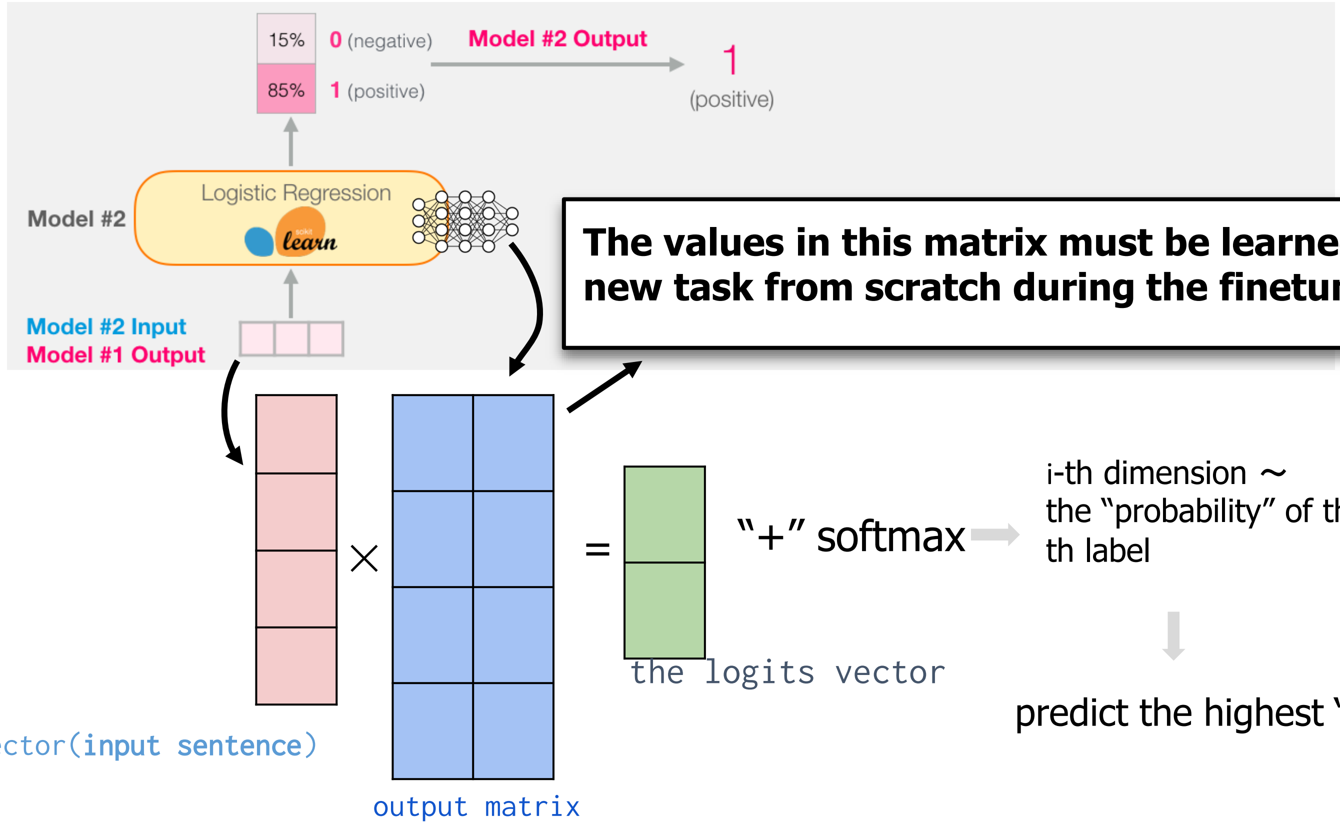


Figure: Jay Alammar

Output layer when finetuning a MLM-pretrained LM



The values in this matrix must be learned for each new task from scratch during the finetuning stage

"+" softmax → i-th dimension ~ the "probability" of the i-th label

predict the highest "probability" label

Figure: A Visual Guide to Using BERT for the First Time by Jay Alammar

Finetuning a MLM-pretrained model

- Add a special [CLS] token to the beginning of the sequence & [SEP] at the end of it
- Use the d-dimensional vector representation of the [CLS] token from the final layer to represent the entire sequence
  - Reminder: This makes sense because of the self-attention
- **Replace the pretrained output layer with a new output layer that has a new randomly initialized output weight matrix of the size `d x n_labels`**
- Multiply it with the `d x n_labels` output matrix ⇒ softmax ⇒ argmax = predicted label
- Cross entropy / NLL loss using the human-annotated labels

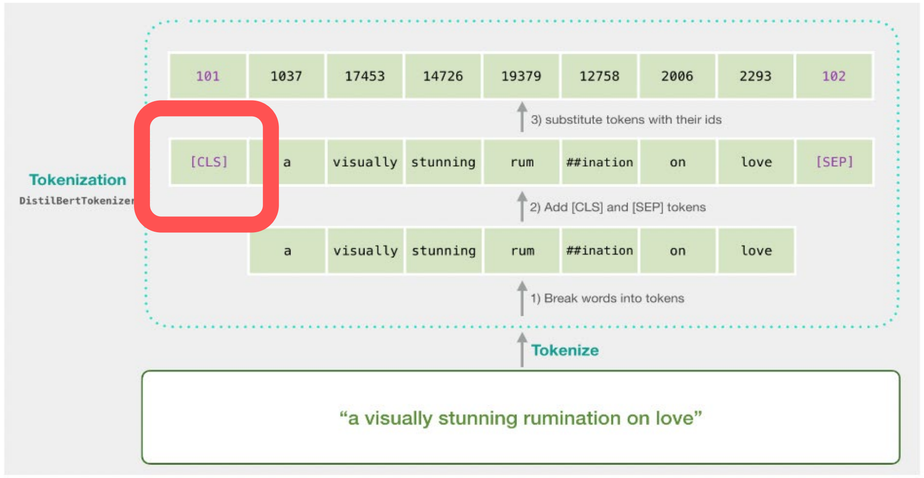
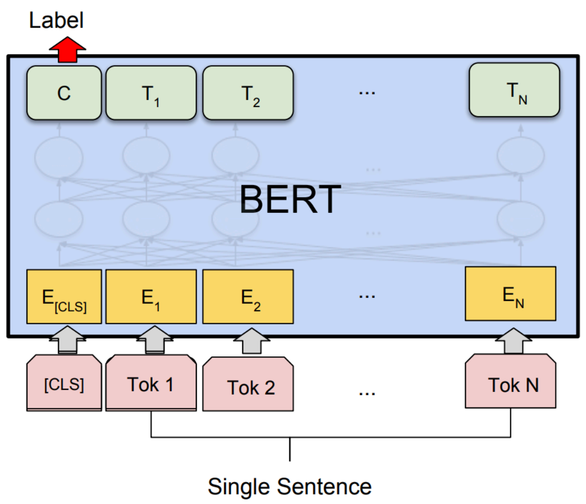
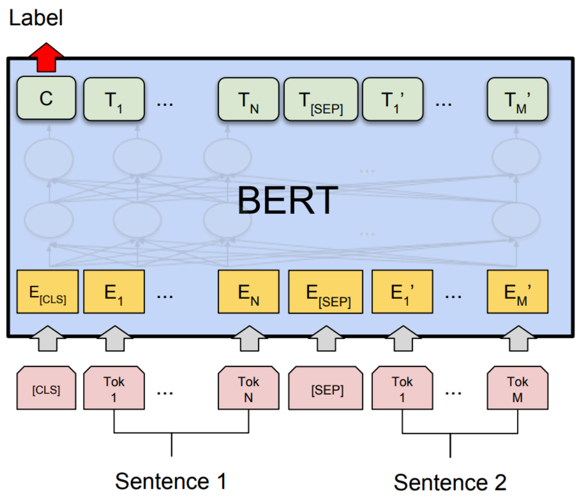


Figure: Jay Alammar

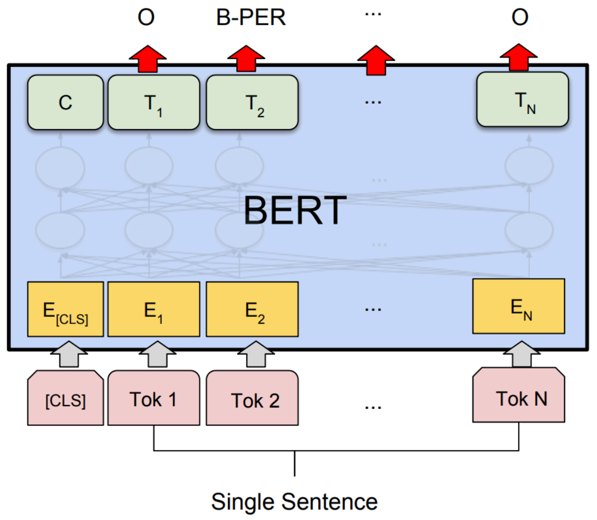
What can BERT do?



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



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

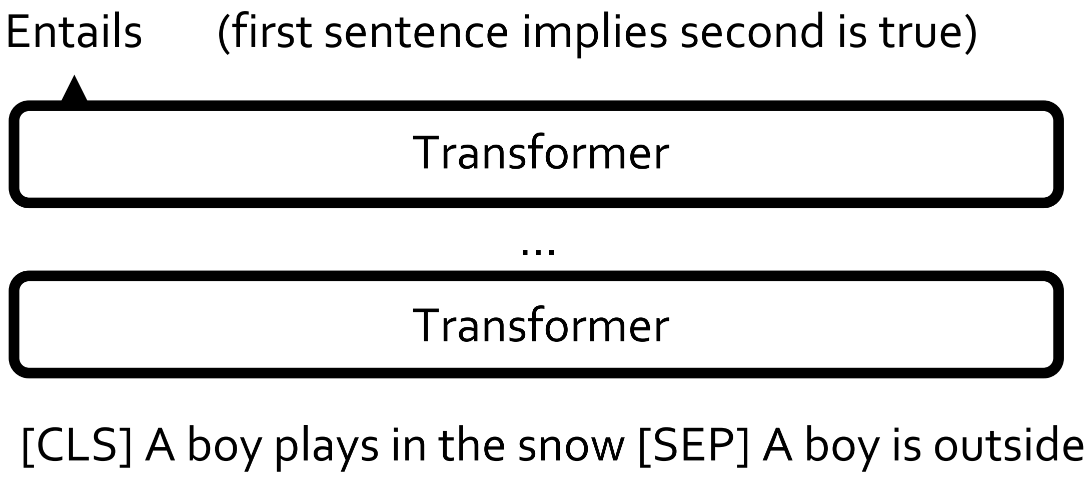


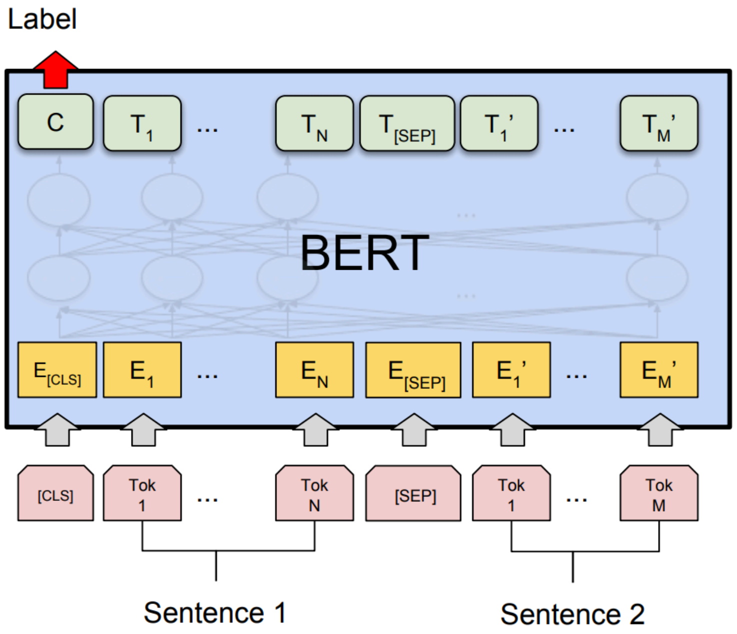
(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

- Artificial [CLS] token is used as the vector to do classification from
- Sentence pair tasks (entailment): feed both sentences into BERT
- BERT can also do tagging by predicting tags at each word piece

Devlin et al. (2019)

What can BERT do?


Entails
(first sentence implies second is true)
Transformer
...
Transformer
[CLS] A boy plays in the snow [SEP] A boy is outside



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

- How does BERT model sentence pairs?
- Transformers can capture interactions between the two sentences, even though the NSP objective doesn't really cause this to happen

SQuAD

Q: What was Marie Curie the first female recipient of?

Passage: One of the most famous people born in Warsaw was Marie Skłodowska-Curie, who achieved international recognition for her research on radioactivity and was the first female recipient of the **Nobel Prize**. Famous musicians include Władysław Szpilman and Frédéric Chopin. Though Chopin was born in the village of Żelazowa Wola, about 60 km (37 mi) from Warsaw, he moved to the city with his family when he was seven months old. Casimir Pulaski, a Polish general and hero of the American Revolutionary War, was born here in 1745.

Answer = Nobel Prize

- Assume we know a passage that contains the answer. More recent work has shown how to retrieve these effectively (will discuss when we get to QA)

SQuAD

Q: What was Marie Curie the first female recipient of?

Passage: One of the most famous people born in Warsaw was Marie Skłodowska-Curie, who achieved international recognition for her research on radioactivity and was the first female recipient of the **Nobel Prize**. ...

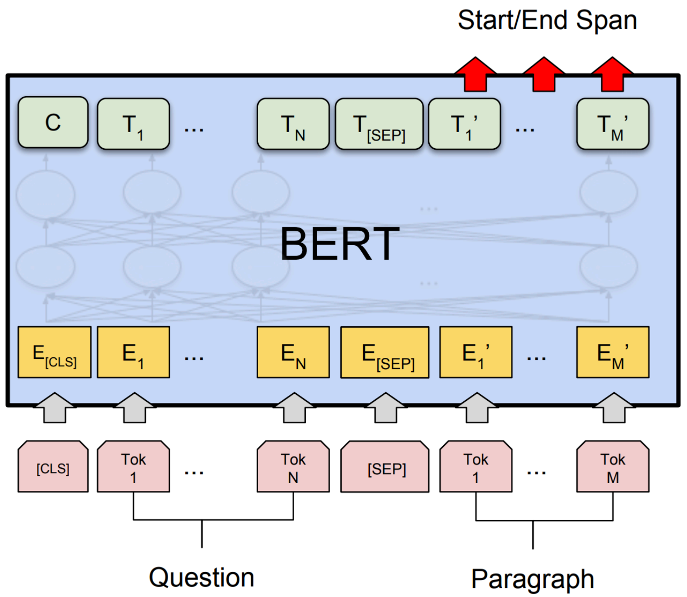
- Predict answer as a pair of (start, end) indices given question q and passage p; compute a score for each word and softmax those

P(start | q, p) =

0.01	0.01	0.01	0.85	0.01	
↑	↑	↑	↑	↑	
recipient	of	the	**Nobel**	**Prize**	.

P(end | q, p) = same computation but different params

QA with BERT

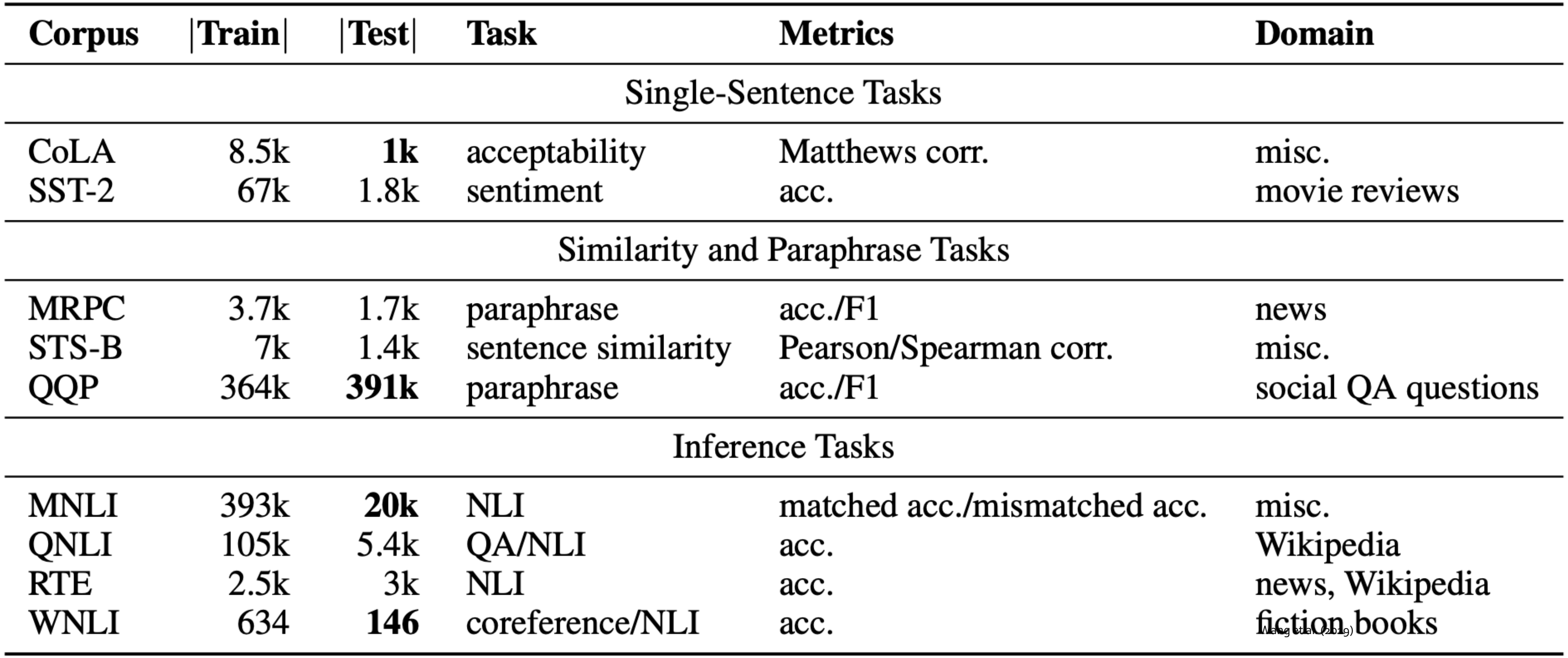


What was Marie Curie the first female recipient of ? [SEP] One of the most famous people born in Warsaw was Marie ...

Devlin et al. (2019)

BERT results, BERT variants

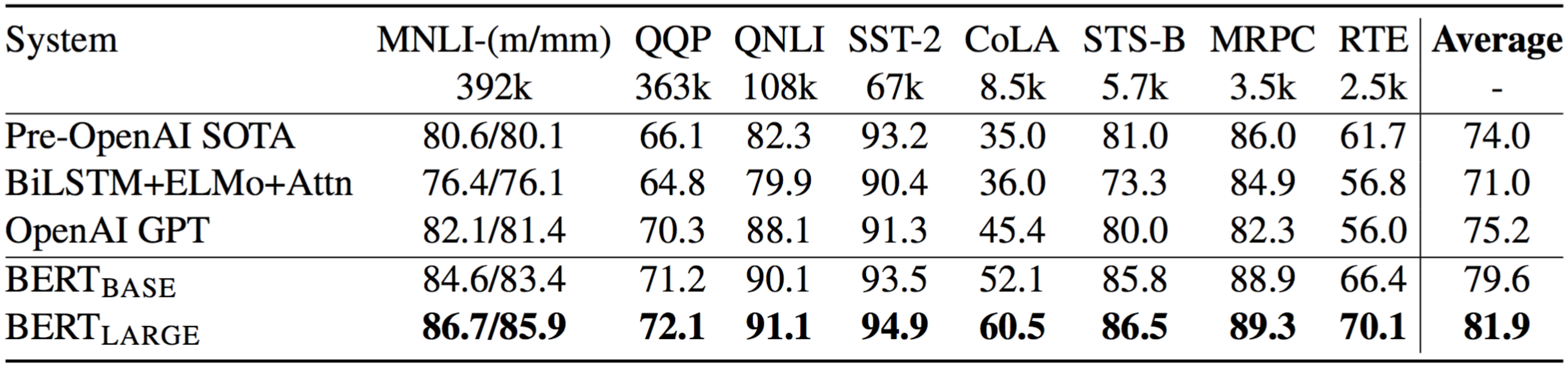
Evaluation: GLUE



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

Wang et al. (2019)

Results

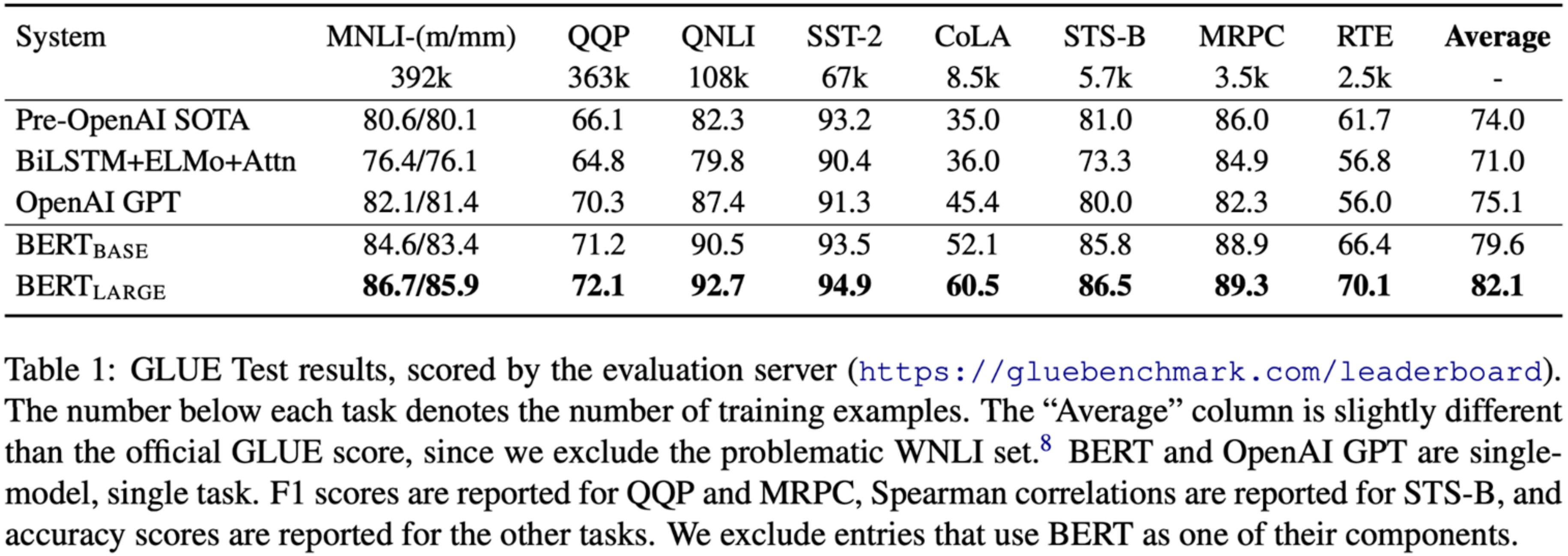


System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
BERT$_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
BERT$_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

- Huge improvements over prior work
- Effective at “sentence pair” tasks: textual entailment (does sentence A imply sentence B), paraphrase detection

Devlin et al. (2018)

Significant improvements from pretraining

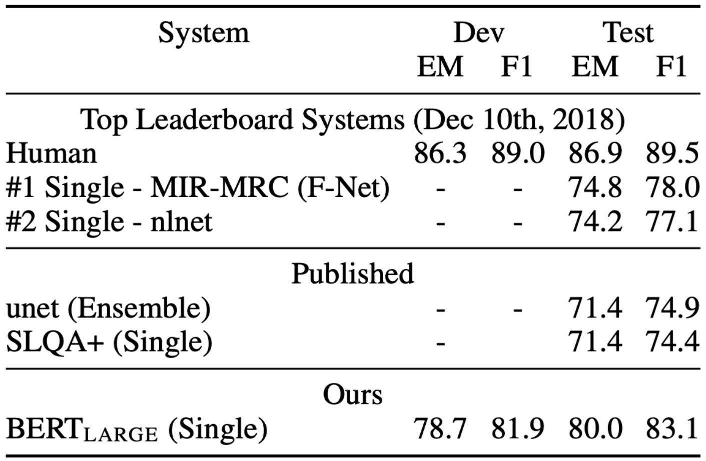


System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1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

Table 1: GLUE Test results, scored by the evaluation server (`https://gluebenchmark.com/leaderboard`). The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set.[8] BERT and OpenAI GPT are single-model, single task. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.

Source: Devlin et al., 2018 (BERT)

Significant improvements from pretraining

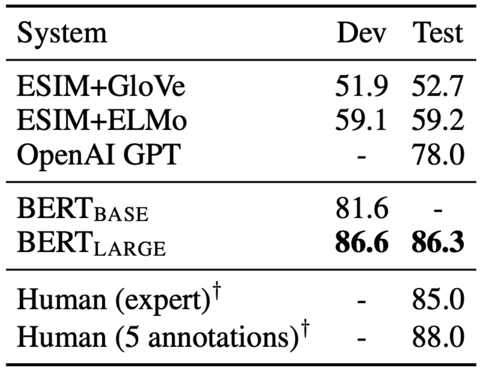


System	Dev EM	Dev F1	Test EM	Test F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	86.3	89.0	86.9	89.5
#1 Single - MIR-MRC (F-Net)	-	-	74.8	78.0
#2 Single - nlnet	-	-	74.2	77.1
Published				
unet (Ensemble)	-	-	71.4	74.9
SLQA+ (Single)	-		71.4	74.4
Ours				
BERT$_{\text{LARGE}}$ (Single)	78.7	81.9	80.0	83.1

Table 3: SQuAD 2.0 results. We exclude entries that use BERT as one of their components.

Source: Devlin et al., 2018 (BERT)

Significant improvements from pretraining



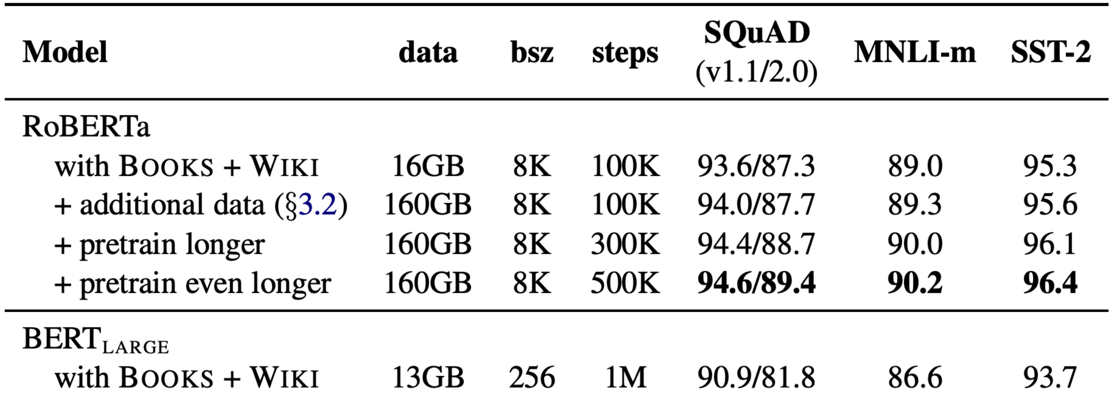
System	Dev	Test
ESIM+GloVe	51.9	52.7
ESIM+ELMo	59.1	59.2
OpenAI GPT	-	78.0
BERT$_{\text{BASE}}$	81.6	-
BERT$_{\text{LARGE}}$	**86.6**	**86.3**
Human (expert)†	-	85.0
Human (5 annotations)†	-	88.0

Table 4: SWAG Dev and Test accuracies. †Human performance is measured with 100 samples, as reported in the SWAG paper.

Source: Devlin et al., 2018 (BERT)

RoBERTa

- “Robustly optimized BERT”
- 160GB of data instead of 16 GB
- Dynamic masking: standard BERT uses the same MASK scheme for every epoch, RoBERTa recomputes them
- New training + more data = better performance

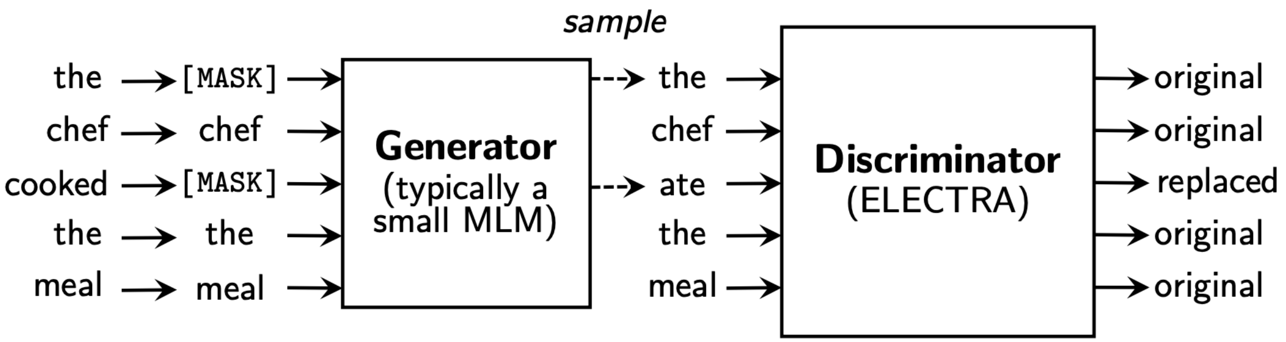


Model	**data**	**bsz**	**steps**	**SQuAD** (v1.1/2.0)	**MNLI-m**	**SST-2**
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
BERT$_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7

Liu et al. (2019)

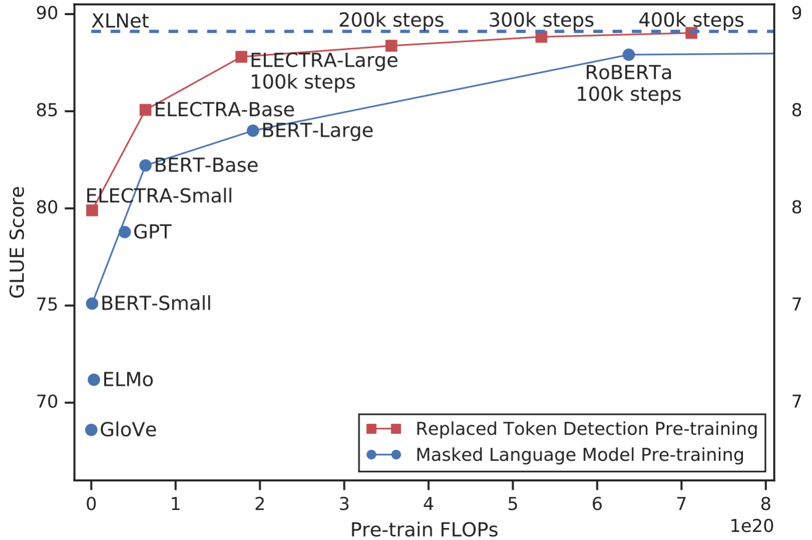
ELECTRA

Clark et al. (2020)



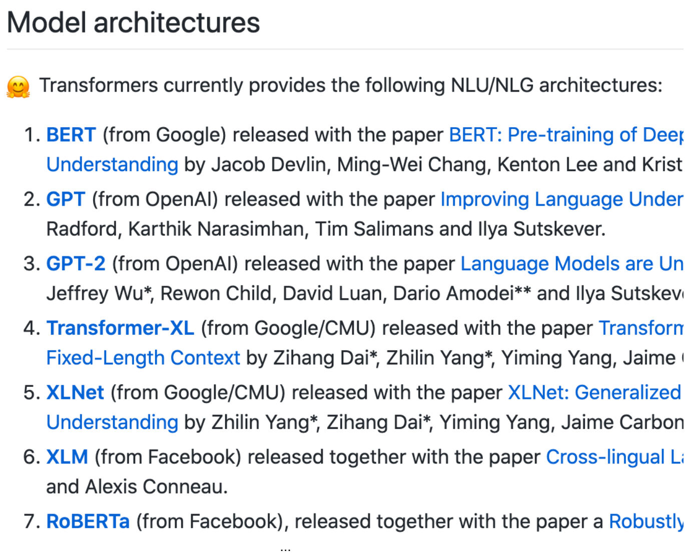
- Discriminator to *detect* replaced tokens rather than a generator to actually *predict* what those tokens are

- More efficient, strong performance



Using BERT

- HuggingFace Transformers: big open-source library with most pre-trained architectures implemented, weights available
- Lots of standard models… and "community models"

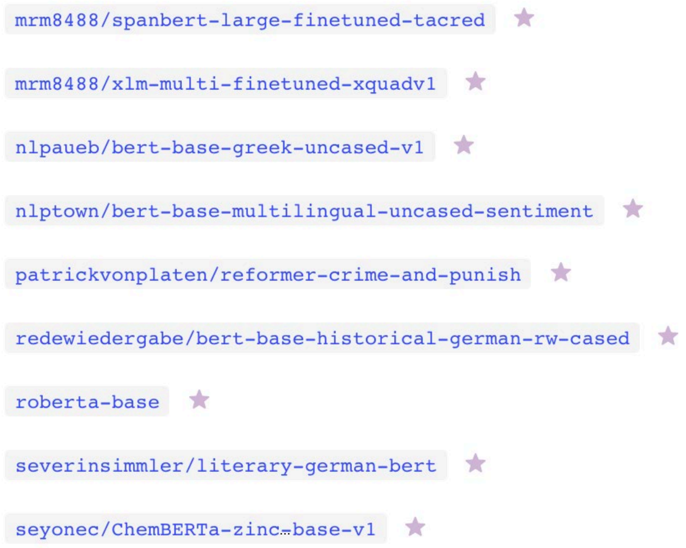


Model architectures

🤗 Transformers currently provides the following NLU/NLG architectures:

1. **BERT** (from Google) released with the paper BERT: Pre-training of Deep Understanding by Jacob Devlin, Ming-Wei Chang, Kenton Lee and Krist
2. **GPT** (from OpenAI) released with the paper Improving Language Under Radford, Karthik Narasimhan, Tim Salimans and Ilya Sutskever.
3. **GPT-2** (from OpenAI) released with the paper Language Models are Un Jeffrey Wu*, Rewon Child, David Luan, Dario Amodei** and Ilya Sutskev
4. **Transformer-XL** (from Google/CMU) released with the paper Transform Fixed-Length Context by Zihang Dai*, Zhilin Yang*, Yiming Yang, Jaime
5. **XLNet** (from Google/CMU) released with the paper XLNet: Generalized Understanding by Zhilin Yang*, Zihang Dai*, Yiming Yang, Jaime Carbon
6. **XLM** (from Facebook) released together with the paper Cross-lingual L and Alexis Conneau.
7. **RoBERTa** (from Facebook), released together with the paper a Robustly

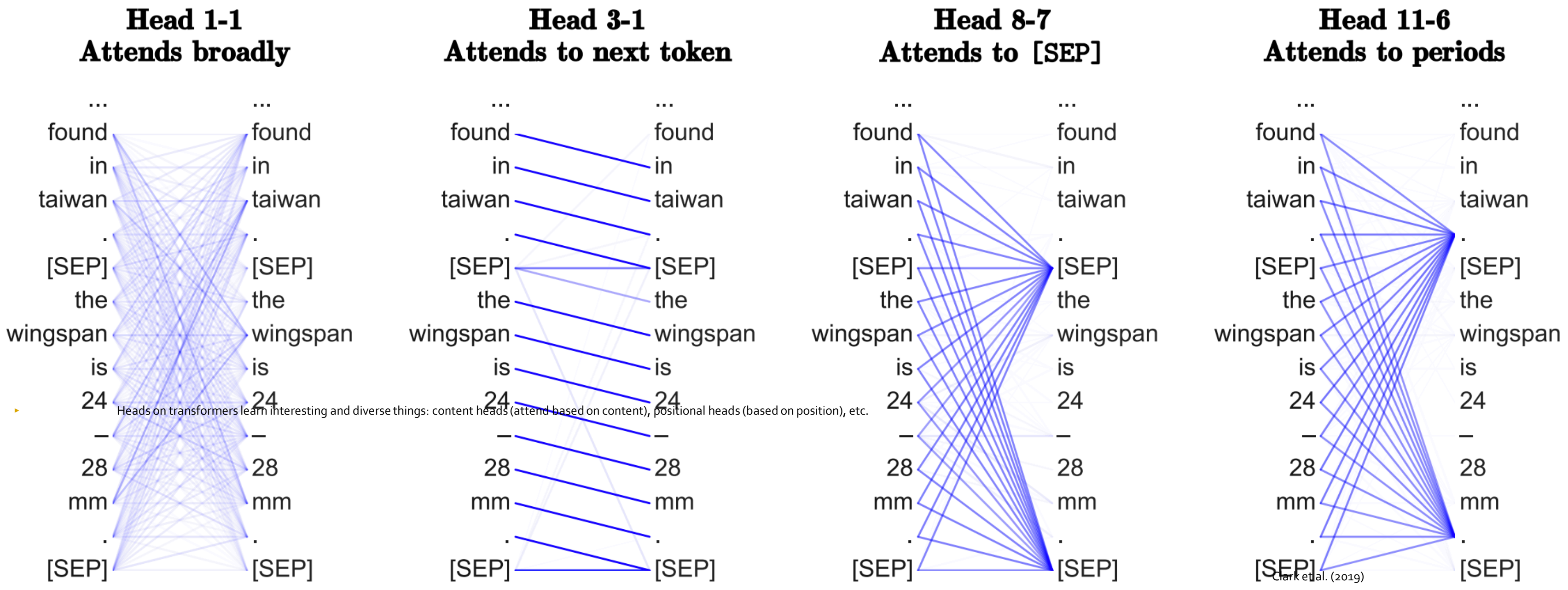
…



```
mrm8488/spanbert-large-finetuned-tacred ★
mrm8488/xlm-multi-finetuned-xquadv1 ★
nlpaueb/bert-base-greek-uncased-v1 ★
nlptown/bert-base-multilingual-uncased-sentiment ★
patrickvonplaten/reformer-crime-and-punish ★
redewiedergabe/bert-base-historical-german-rw-cased ★
roberta-base ★
severinsimmler/literary-german-bert ★
seyonec/ChemBERTa-zinc-base-v1 ★
```

…

What does BERT learn?



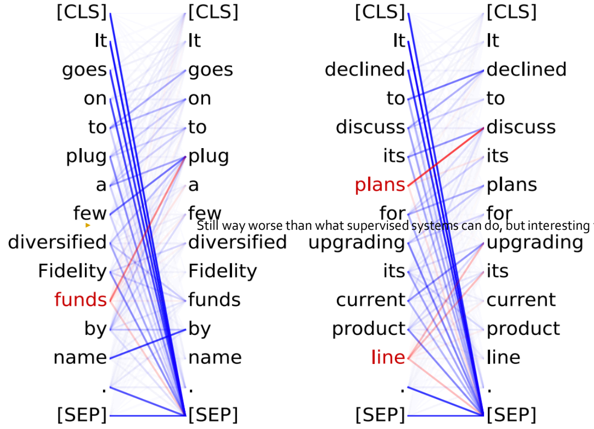
- Heads on transformers learn interesting and diverse things: content heads (attend based on content), positional heads (based on position), etc.

Clark et al. (2019)

What does BERT learn?

Head 8-10

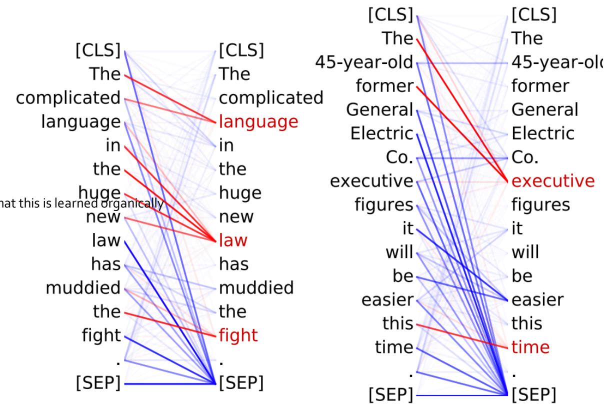
- **Direct objects** attend to their verbs
- 86.8% accuracy at the `dobj` relation


[CLS] It goes on to plug a few diversified Fidelity funds by name . [SEP]
[CLS] It declined to discuss its plans for upgrading its current product line . [SEP]

Still way worse than what supervised systems can do, but interesting that this is learned organically

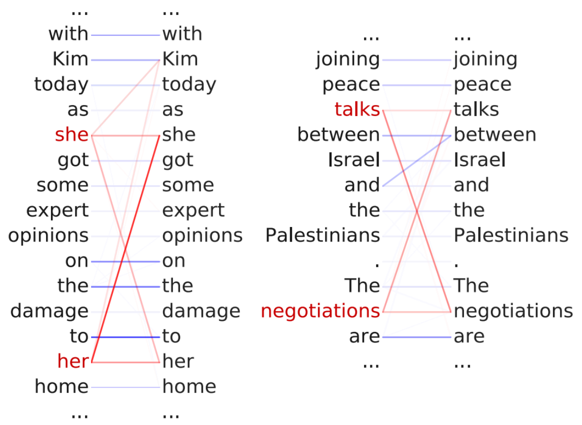
Head 8-11

- **Noun modifiers** (e.g., determiners) attend to their noun
- 94.3% accuracy at the `det` relation


[CLS] The complicated language in the huge new law has muddied the fight . [SEP]
[CLS] The 45-year-old former General Electric Co. executive figures it will be easier this time . [SEP]

Head 5-4

- **Coreferent** mentions attend to their antecedents
- 65.1% accuracy at linking the head of a coreferent mention to the head of an antecedent


... with Kim today as she got some expert opinions on the damage to her home ...
... joining peace talks between Israel and the Palestinians . The negotiations are ...

Clark et al. (2019)

BERT / RoBERTa / DeBERTa

[Devlin et al., 2018 / Liu et al., 2019 / He et al., 2023]

- Encoder-only transformer
- Masked language modeling (MLM), ~~next sentence prediction~~
- 110M, 340M parameters

⚓ These models are a good option if you want to solve a text classification problem for which you have thousands of labeled datapoints & you know how to train a model (which you all will know after this course)

⚓ Don't work for generation as good as decoder-only or encoder-decoder models

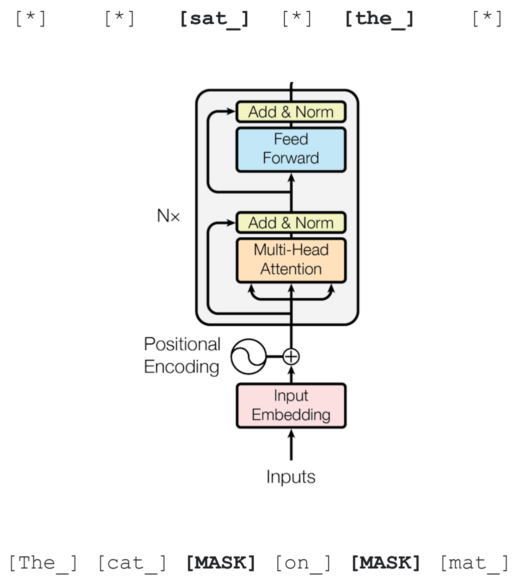
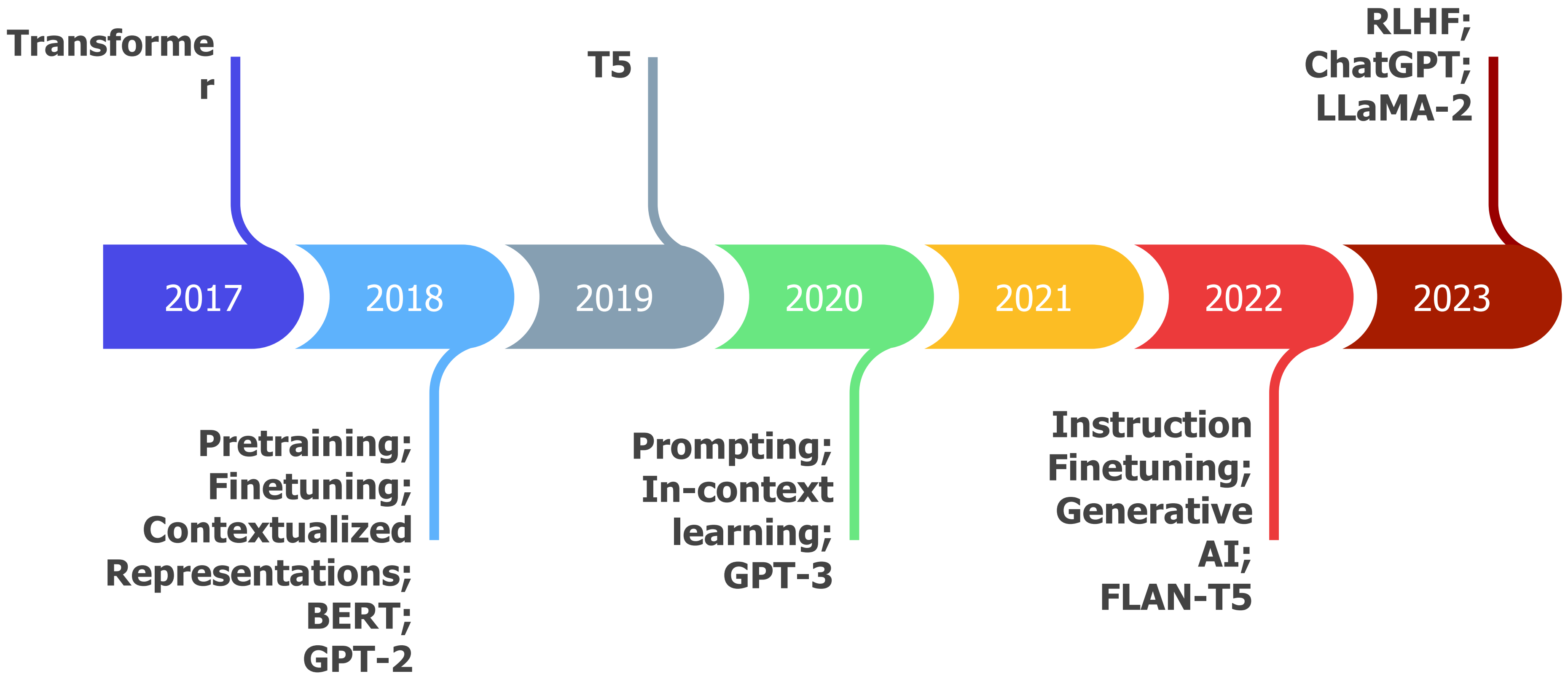


Figure by: Lucas Beyer


Transforme
r
2017
2018
Pretraining;
Finetuning;
Contextualized
Representations;
BERT;
GPT-2
T5
2019
2020
Prompting;
In-context
learning;
GPT-3
2021
2022
Instruction
Finetuning;
Generative
AI;
FLAN-T5
RLHF;
ChatGPT;
LLaMA-2
2023