CSE 587: Deep Learning for Natural Language Processing

Lecture 6. BERT and Pretraining

Rui Zhang

Spring 2023

PennState

Outline

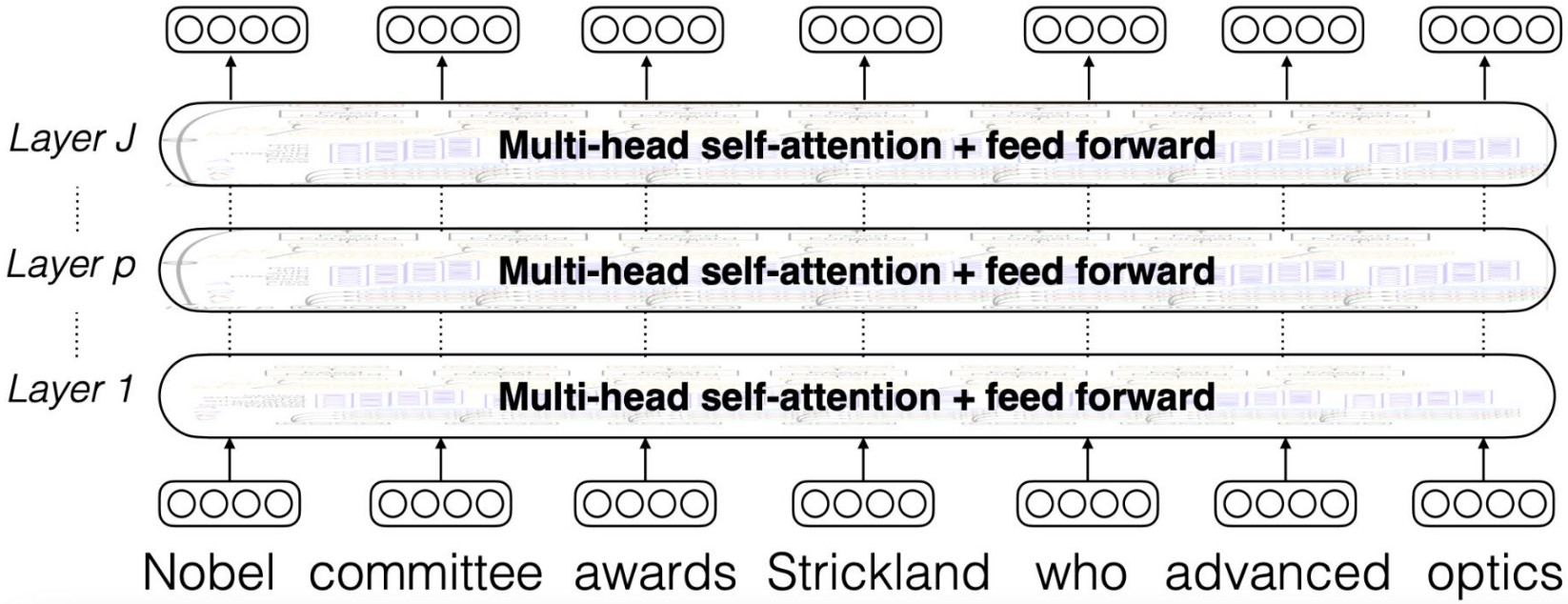
NLP

- Pretrained Language Models
- BERT and its variants
- Model Analysis

ML

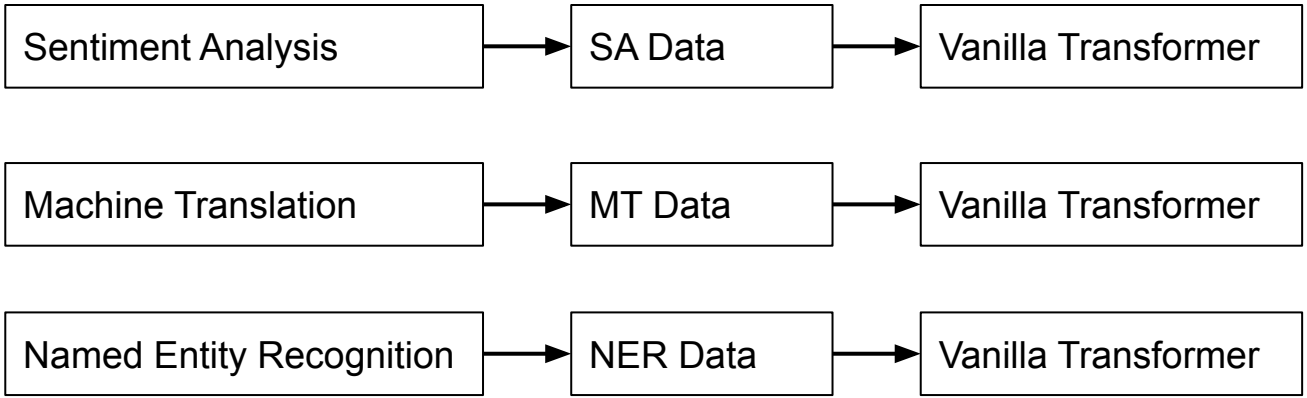
- Pretraining
- Finetuning

Transformer



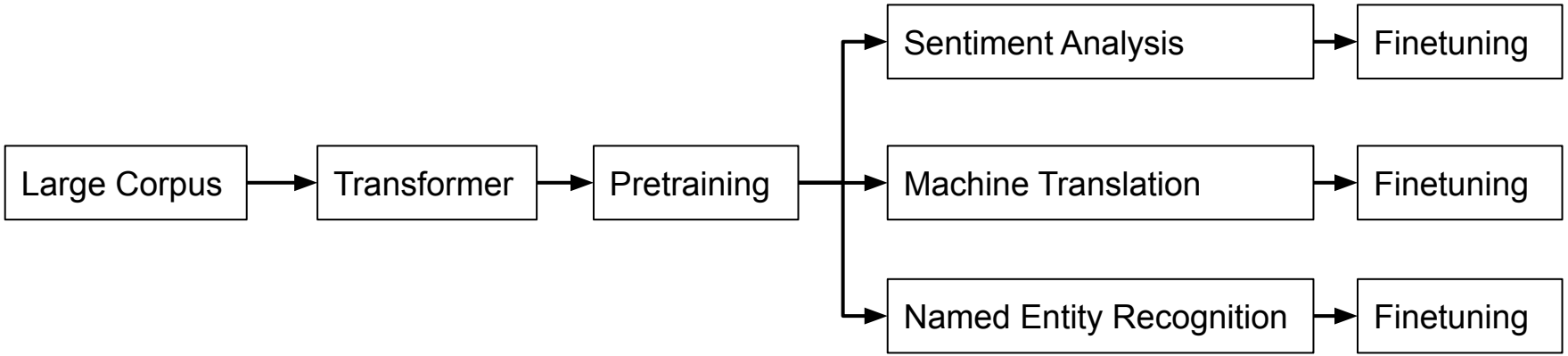
How is Transformer used

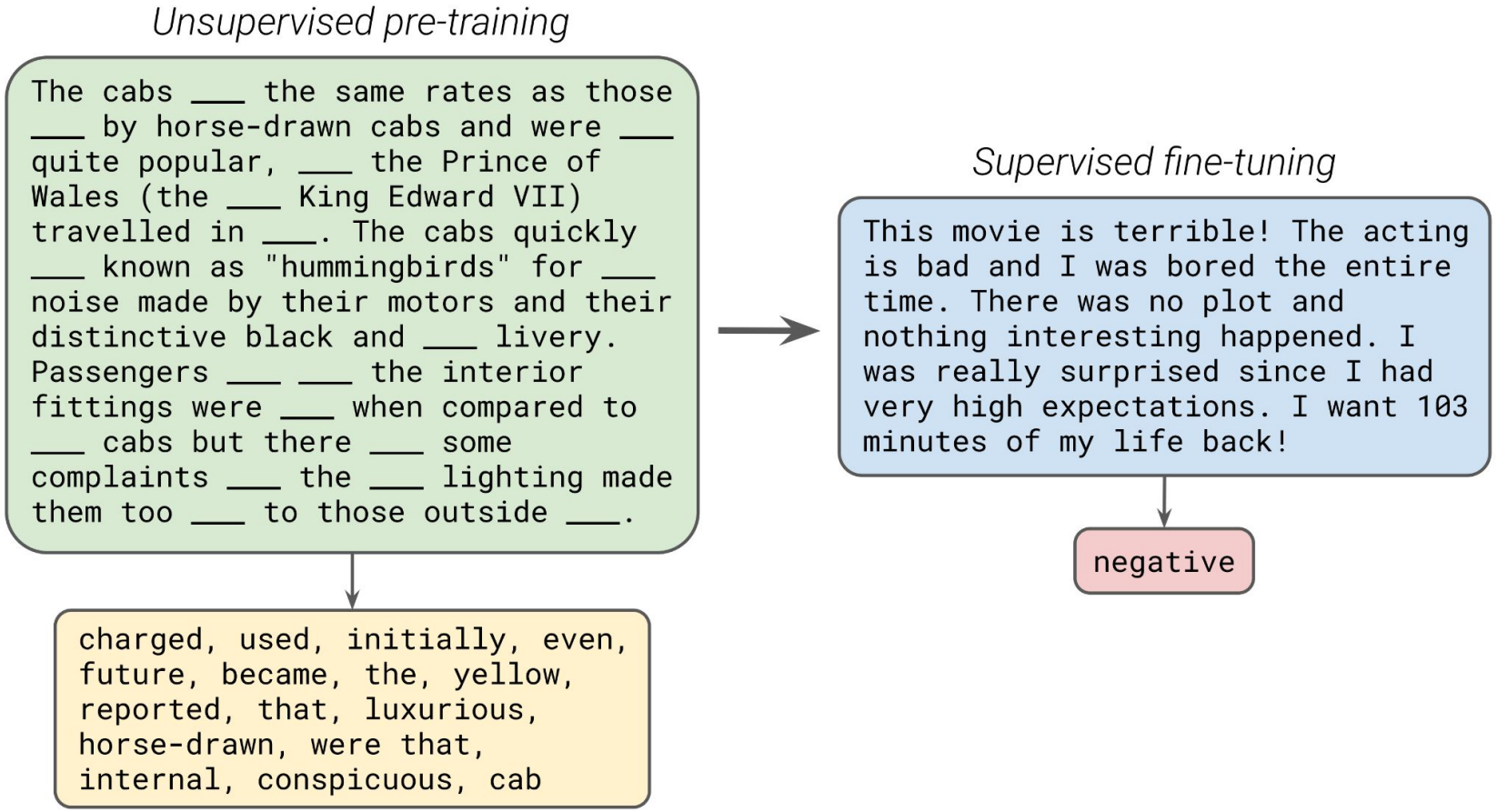
We can use Transformer separately for each task.
Transformer is initialized randomly, and trained for each dataset using supervised learning.



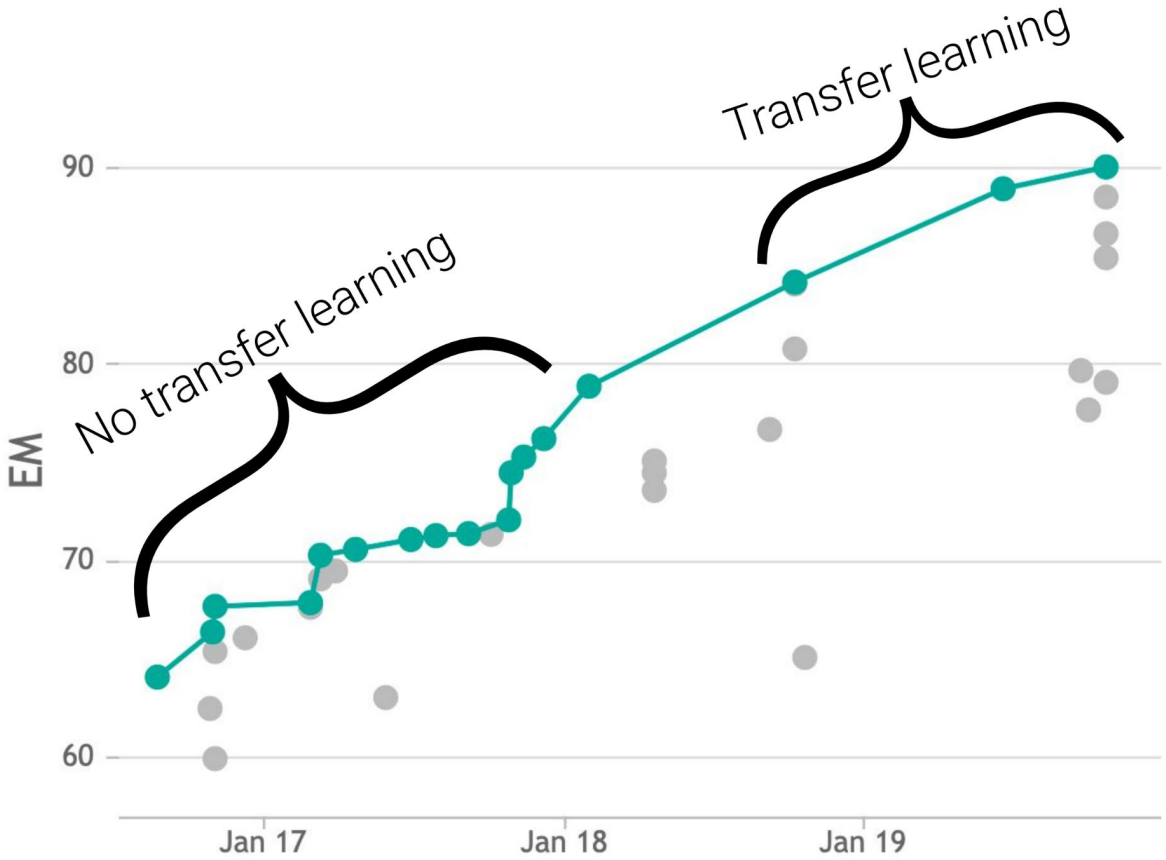
Pretraining

- First train Transformer using a lot of general text using *unsupervised* learning. This is called **pretraining**.
- Then train the pretrained Transformer for a specific task using *supervised* learning.This is called **finetuning.**
- The whole process can be called **transfer learning**.





Slides Credit: Colin Raffel



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

Slides Credit: Colin Raffel

BERT (Devlin et al. 2018)

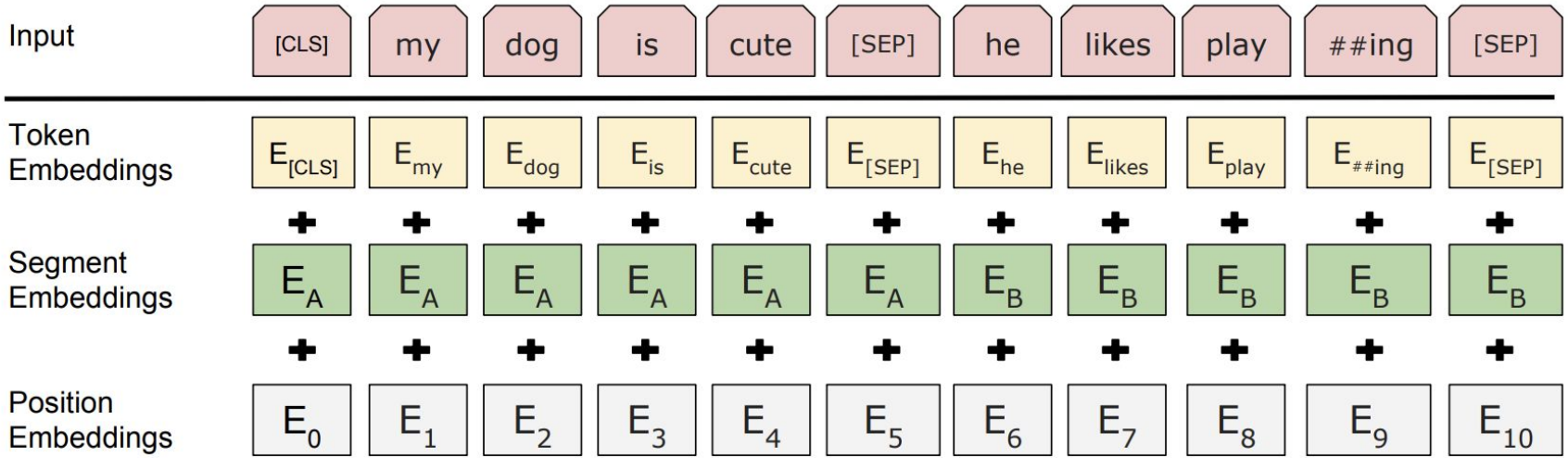
Model: Only use Transformer Encoder (no decoder part)

Data: BooksCorpus (800 million words) + English Wikipedia (2,500 million words)

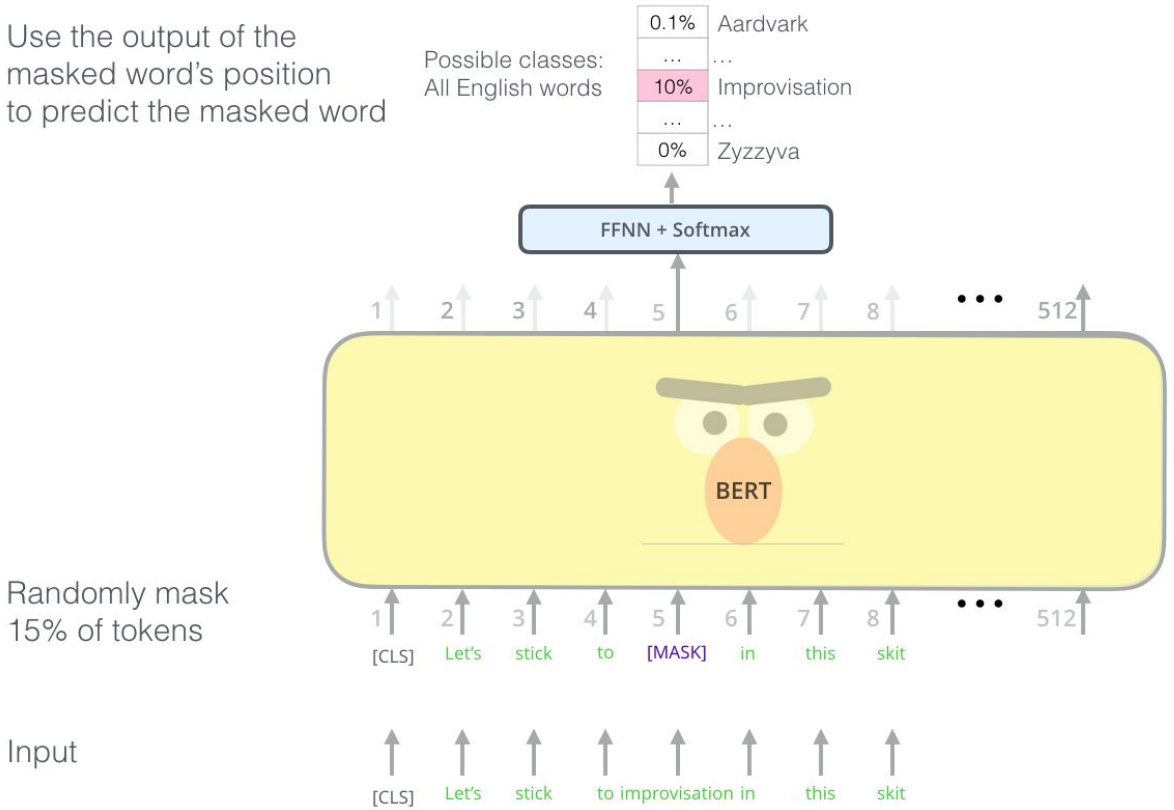
Training Objective

- Masked Language Modeling: predict word given bidirectional context.
- Next-sentence Prediction: predict the next sentence given the current sentence.

BERT Input


Input
[CLS] my dog is cute [SEP] he likes play ##ing [SEP]
Token Embeddings
E[CLS] Emy Edog Eis Ecute E[SEP] Ehe Elikes Eplay E##ing E[SEP]
Segment Embeddings
EA EA EA EA EA EA EB EB EB EB EB
Position Embeddings
E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 E10

Training Objective 1: Masked Language Modeling

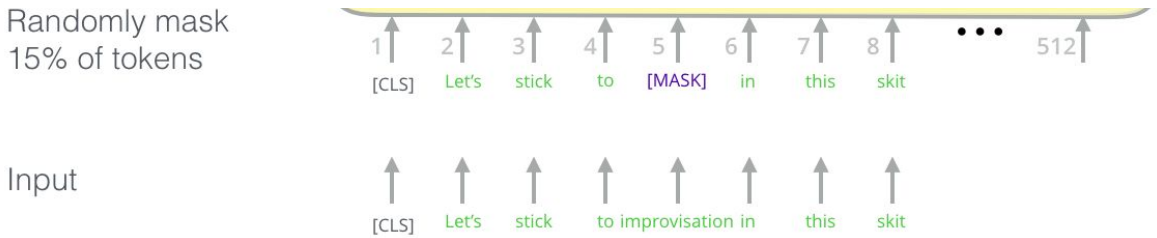


https://jalammar.github.io/illustrated-bert/

Training Objective 1: Masked Language Modeling

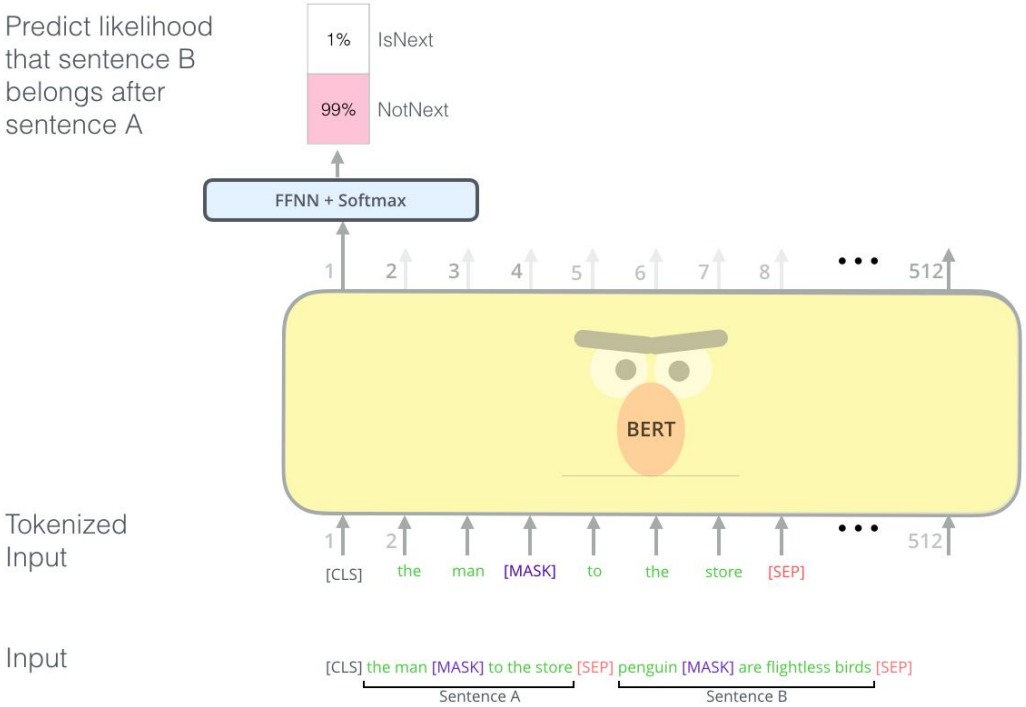
Predict a random 15% of (sub)word tokens, and of these 15%:

- 80%: Replace input word with [MASK]
- 10%: Replace input word with a random token
- 10%: Leave input word unchanged 10% (but still predict it!)



Training Objective 2: Next-sentence Prediction

Give two sentences as input, classify if the second sentence really follows the first one.



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

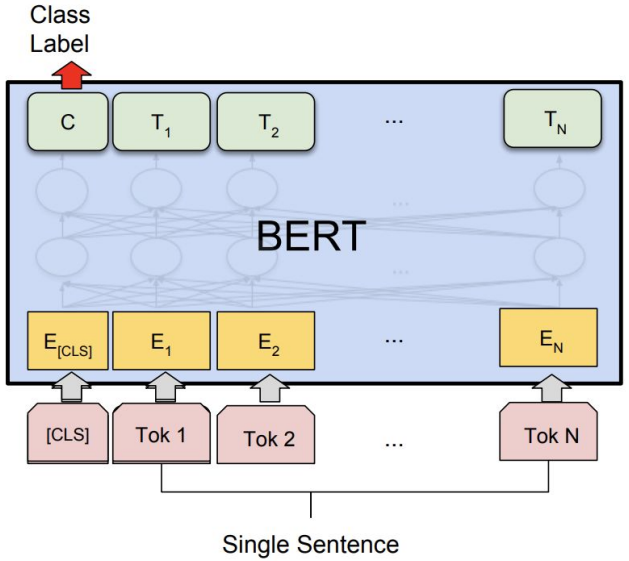
BERT on Different Tasks

We can use BERT for different tasks by changing the inputs and adding classification layers on top of output embeddings.

> *"We show that pre-trained representations reduce the need for many heavily-engineered task specific architectures. BERT is the first finetuning based representation model that achieves state-of-the-art performance on a large suite of sentence-level and token-level tasks, outperforming many task-specific architectures."*

BERT on Different Tasks

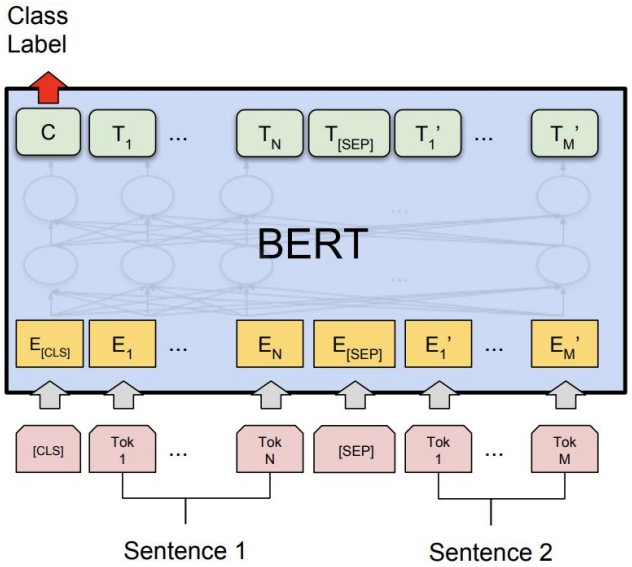
Sentence Classification



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

BERT on Different Tasks

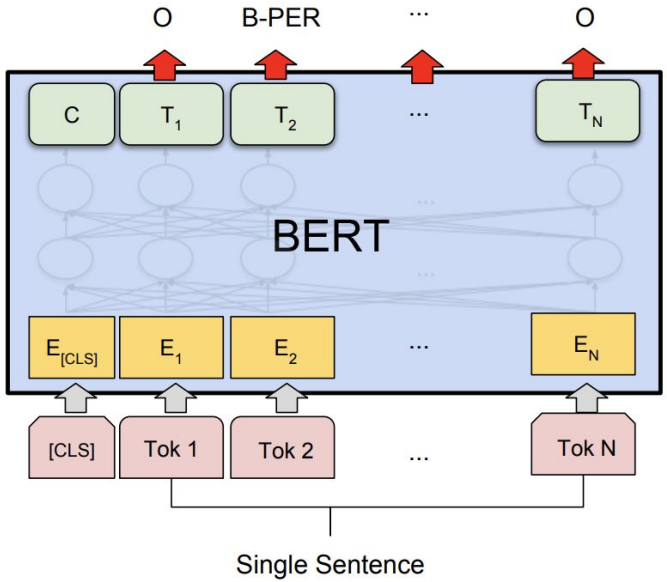
Sentence Pair Classification



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

BERT on Different Tasks

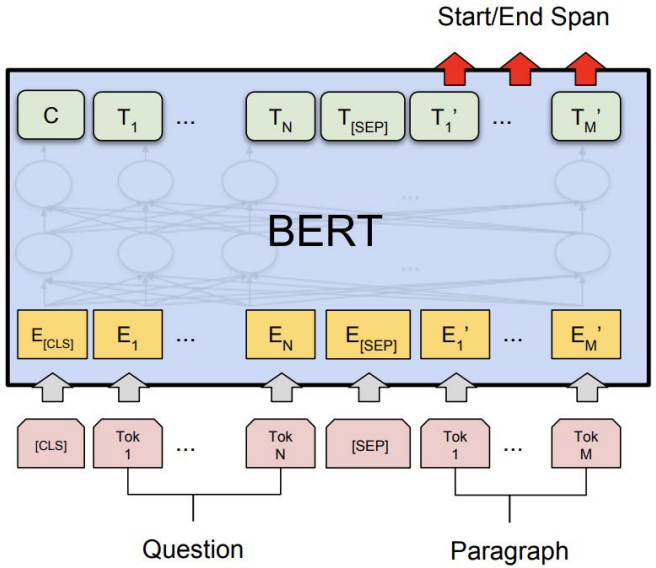
Sequence Labeling



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

BERT on Different Tasks

Question Answering



(c) Question Answering Tasks:
SQuAD v1.1

BERT

Initially two BERT models are trained and released with the paper

- **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.

Pretraining is expensive

- 64 TPU chips for a total of 4 days

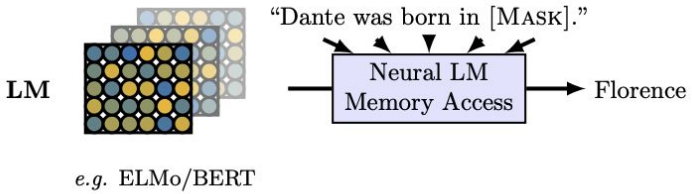
What Does BERT Learns?

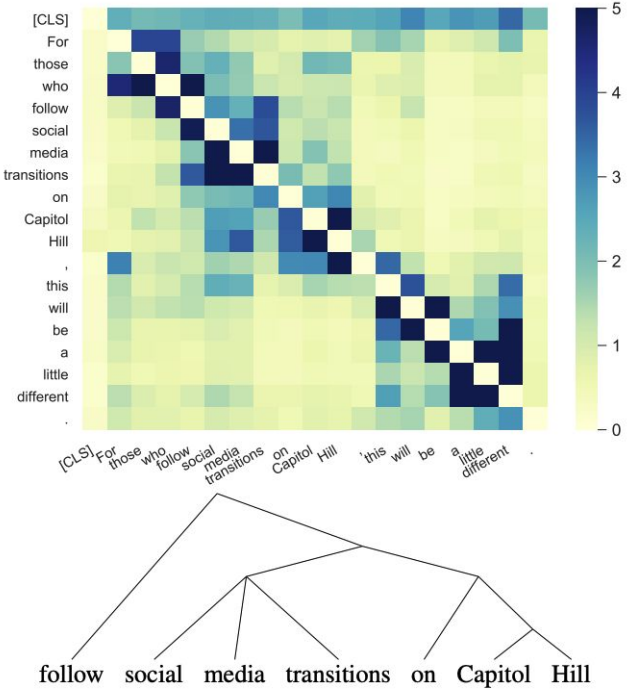
1. Semantic Knowledge

to tip a ___ : waiter > chef > robin

2. Syntactic Knowledge

3. World Knowledge





A Primer in BERTology: What we know about how BERT works (Rogers et al., 2020)

Why Pretraining Works? Is BERT just memorizing?

Size of Data (~15GB)

- BookCorpus (4.6 GB)
- English Wikipeida (10 GB)

plain_text

- **Size of downloaded dataset files:** 1124.87 MB
- **Size of the generated dataset:** 4629.00 MB
- **Total amount of disk used:** 5753.87 MB

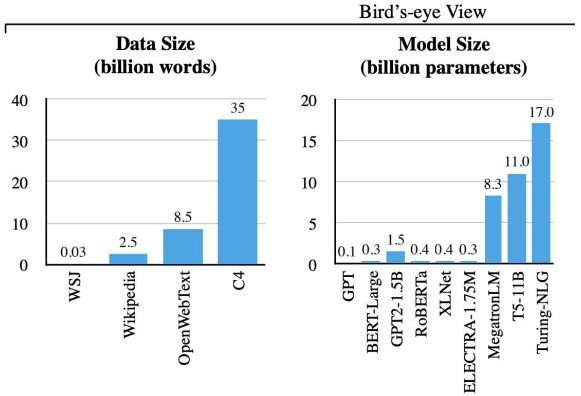
:0220301.en

- **Size of downloaded dataset files:** 20598.31 MB
- **Size of the generated dataset:** 20275.52 MB
- **Total amount of disk used:** 40873.83 MB

Size of Model

- bert-base (110M x 4 bytes = 0.44 GB)
- bert-large (340M x 4 bytes = 1.36 GB)

Not just memorize, but also compress



How much does it cost?

The Cost of Training NLP Models: A Concise Overview (Sharir et al., 2020)

1. The cost of one training run
2. A typical fully-loaded cost with hyper-parameter tuning and multiple runs per setting

- $2.5k - $50k (110 million parameter model)
- $10k - $200k (340 million parameter model)
- $80k - $1.6m (1.5 billion parameter model)

How to Train BERT with an Academic Budget (Izsak et al., 2021)

24-hours, $50~$100

Our recipe consists of many different components that together amount to very large speedups:

- Short sequences (Devlin et al., 2019)
- Single-sequence training (Joshi et al., 2020)
- Training larger models (Li et al., 2020b)
- DeepSpeed (Rasley et al., 2020)
- Sparse token prediction (Liu et al., 2019)
- Fused implementations
- Avoiding disk I/O
- Large batch sizes (Liu et al., 2019)
- Large learning rates (Liu et al., 2019)
- Short warmup
- Synchronizing schedule with time budget (Li et al., 2020a)

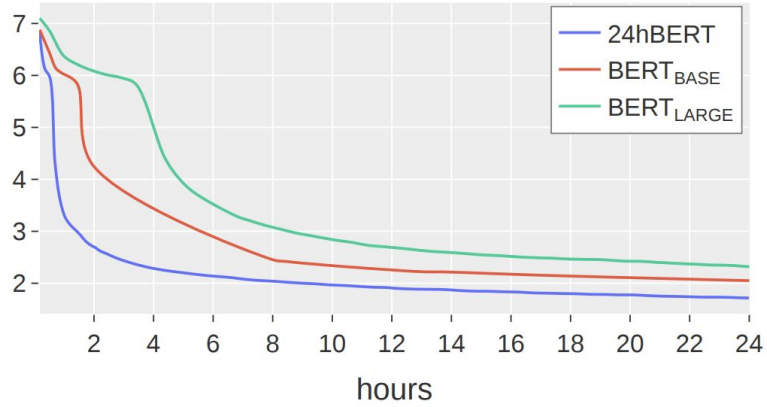
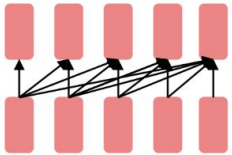


Figure 2: The validation-set loss of 24hBERT compared to the original BERT model configurations.

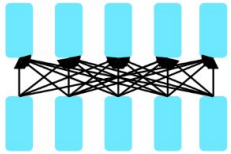
Pretraining for three types of architectures

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



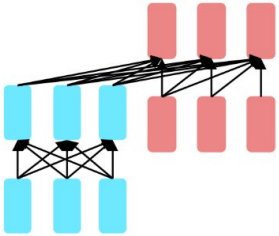
Decoders

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



Encoders

- Gets bidirectional context – can condition on future!
- Wait, how do we pretrain them?



Encoder-Decoders

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

Slides from John Hewitt

RoBERTa (Liu et al. 2019)

RoBERTa: A Robustly Optimized BERT Pretraining Approach

Model: same as BERT
Data: same as bert
Training Objective

- MLM same as BERT, but train longer
- Remove next-sentence prediction.

Takeaway: more compute and more data can help; next-sentence prediction not necessary.

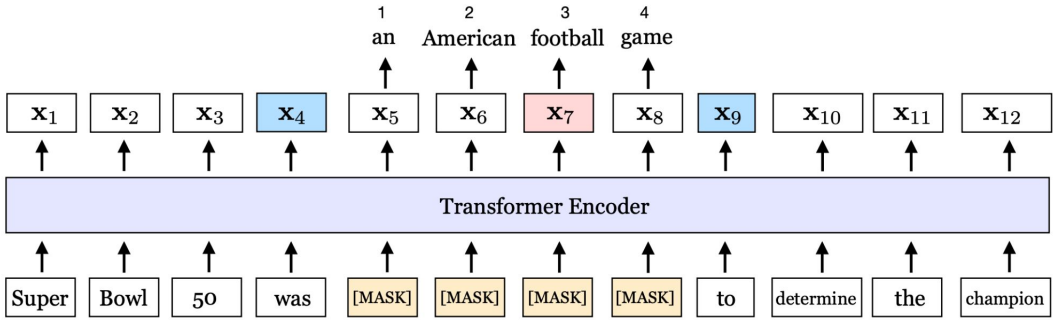
SpanBERT (Joshi et al., 2019)

SpanBERT: Improving Pre-training by Representing and Predicting Spans

Model: same as BERT
Data: same as bert
Training Objective

- Masking contiguous random spans, rather than random tokens
- Training the span boundary representations to predict the entire content of the masked span, without relying on the individual token representations within it.

Takeaway: predicting entire spans is better than random tokens



GPT (Radford et al., 2018)

GPT

- Generative Pretrained Transformer
- Generative PreTraining

Model: only Transformer decoder.
Data: BooksCorpus: over 7000 unique books.
Training Objective: Language Modeling

Followed by GPT-2 and GPT-3

- GPT (Jun 2018): 117 million parameters
- GPT-2 (Feb 2019): 1.5 billion parameters
- GPT-3 (July 2020): 175 billion parameters

ELECTRA (Clark et al. 2020)

ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators

Model: Same as BERT

Data: Same as BERT

Training Objective:

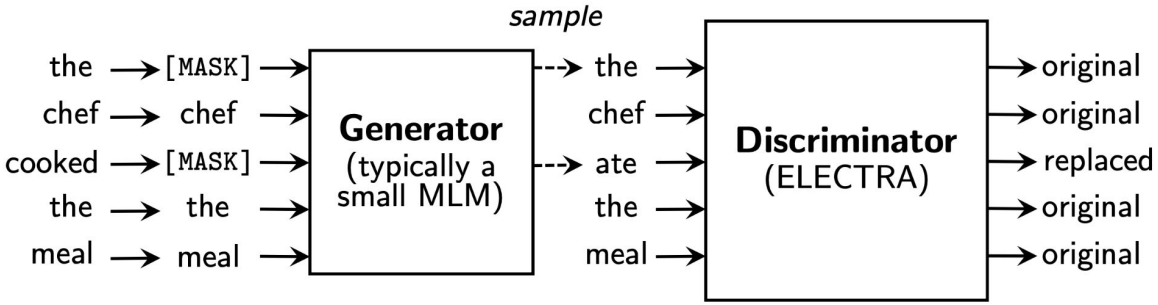


Figure 2: An overview of replaced token detection. The generator can be any model that produces an output distribution over tokens, but we usually use a small masked language model that is trained jointly with the discriminator. Although the models are structured like in a GAN, we train the generator with maximum likelihood rather than adversarially due to the difficulty of applying GANs to text. After pre-training, we throw out the generator and only fine-tune the discriminator (the ELECTRA model) on downstream tasks.

ELECTRA (Clark et al. 2020)

the task is defined over all input tokens rather than just the small subset that was masked out. As a result, the contextual representations learned by our approach substantially outperform the ones learned by BERT given the same model size, data, and compute. This makes training more efficient.

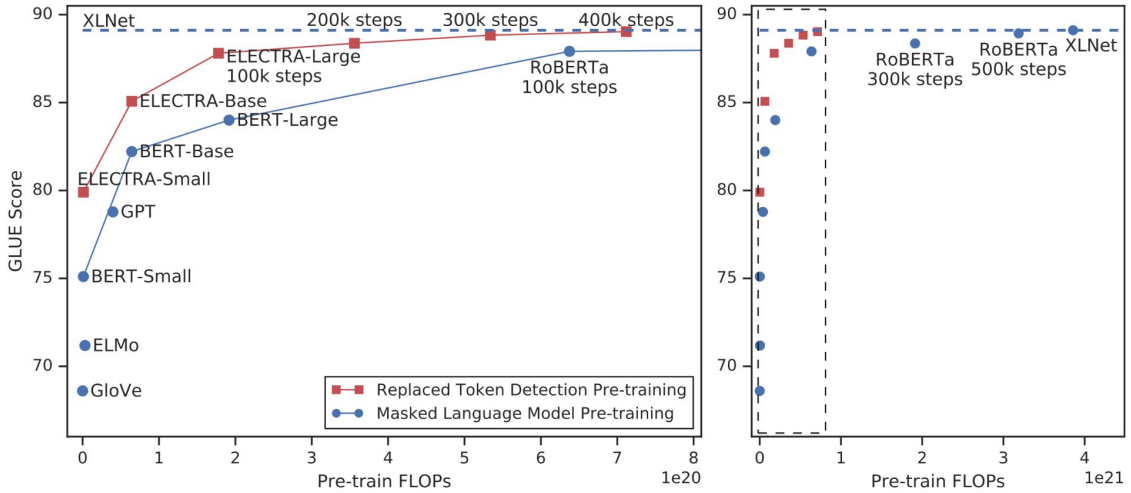


Figure 1: Replaced token detection pre-training consistently outperforms masked language model pre-training given the same compute budget. The left figure is a zoomed-in view of the dashed box.

T5 (Raffel et al., 2019)

T5: “Text-to-Text Transfer Transformer”

Treat every text processing problem as a “text-to-text” problem, i.e. taking text as input and producing new text as output.

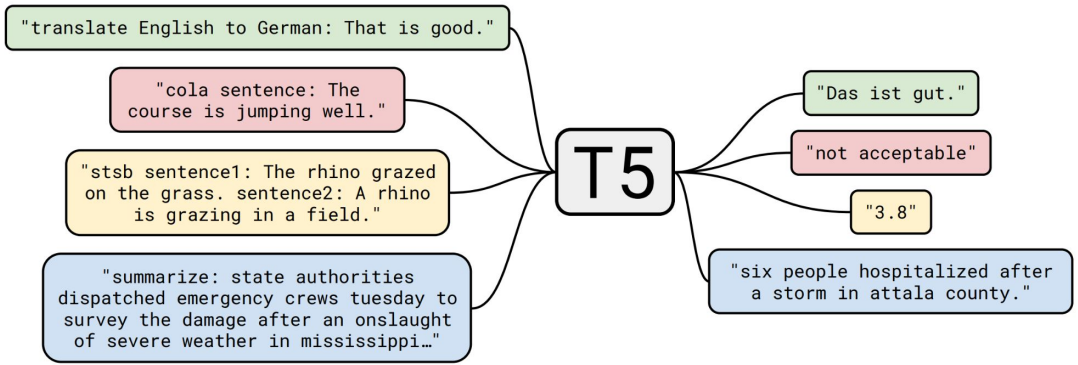


Figure 1: A diagram of our text-to-text framework. Every task we consider—including translation, question answering, and classification—is cast as feeding our model text as input and training it to generate some target text. This allows us to use the same model, loss function, hyperparameters, etc. across 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”.

T5 (Raffel et al., 2019)

Model: Transformer Encoder-Decoder

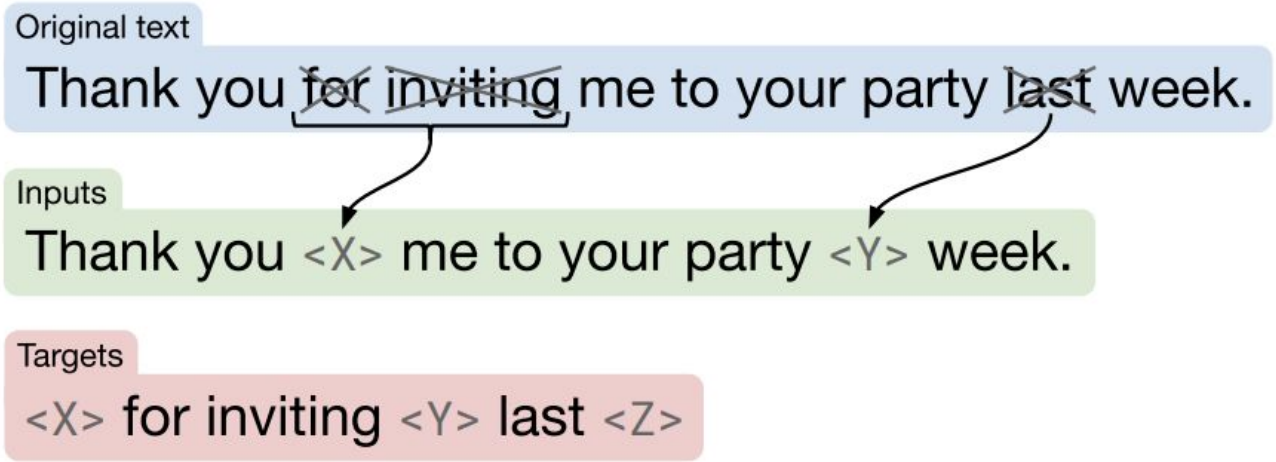
Data: C4 (Colossal Clean Crawled Corpus), clean English text scraped from the we (750GB).

Training Objective

- **Span Corruption (denoising)**

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>

Span Corruption (denoising)



T5 (Raffel et al., 2019)

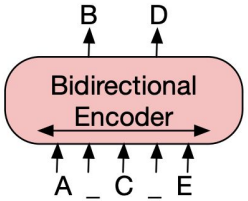
T5 can be used for both classification (GLUE) and generation tasks such as Summarization (CNNDM), Machine Translation (EnDe, EnFr, EnRo).

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**

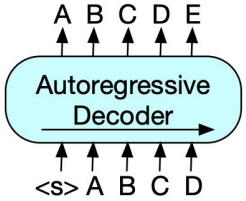
Table 5: Comparison of variants of the BERT-style pre-training objective. In the first two variants, the model is trained to reconstruct the original uncorrupted text segment. In the latter two, the model only predicts the sequence of corrupted tokens.

BART (Lewis et al., 2019)

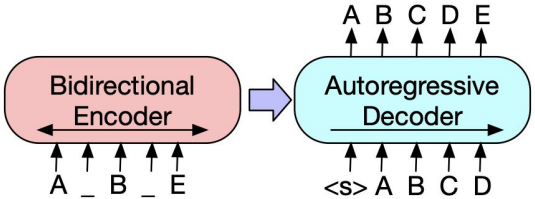
BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension.



(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.

Figure 1: A schematic comparison of BART with BERT (Devlin et al., 2019) and GPT (Radford et al., 2018).

SciBERT (Beltagy et al., 2020)

SciBERT: A Pretrained Language Model for Scientific Text.

Keep pretraining BERT on in-domain data improves the end task performance.

Field	Task	Dataset	SOTA	BERT-Base		SCIBERT	
				Frozen	Finetune	Frozen	Finetune
Bio	NER	BC5CDR (Li et al., 2016)	88.85[7]	85.08	86.72	88.73	**90.01**
		JNLPBA (Collier and Kim, 2004)	**78.58**	74.05	76.09	75.77	77.28
		NCBI-disease (Dogan et al., 2014)	**89.36**	84.06	86.88	86.39	88.57
	PICO	EBM-NLP (Nye et al., 2018)	66.30	61.44	71.53	68.30	**72.28**
	DEP	GENIA (Kim et al., 2003) - LAS	**91.92**	90.22	90.33	90.36	90.43
		GENIA (Kim et al., 2003) - UAS	**92.84**	91.84	91.89	92.00	91.99
	REL	ChemProt (Kringelum et al., 2016)	76.68	68.21	79.14	75.03	**83.64**
CS	NER	SciERC (Luan et al., 2018)	64.20	63.58	65.24	65.77	**67.57**
	REL	SciERC (Luan et al., 2018)	n/a	72.74	78.71	75.25	**79.97**
	CLS	ACL-ARC (Jurgens et al., 2018)	67.9	62.04	63.91	60.74	**70.98**
Multi	CLS	Paper Field	n/a	63.64	65.37	64.38	**65.71**
		SciCite (Cohan et al., 2019)	84.0	84.31	84.85	**85.42**	**85.49**
Average				73.58	77.16	76.01	79.27

Table 1: Test performances of all BERT variants on all tasks and datasets. **Bold** indicates the SOTA result (multiple

REALM (Guu et al., 2020)

REALM: **Retrieval-Augmented** Language Model Pre-Training

These pre-trained models, such as BERT and RoBERTa, have been shown to *memorize a surprising amount of world knowledge*, such as “the birthplace of Francesco Bartolomeo Conti”, “the developer of JDK” and “the owner of Border TV”. … these models memorize knowledge *implicitly* — i.e., world knowledge is captured in an abstract way in the model weights …

Instead, what if there was a method for pre-training that could access knowledge *explicitly*, e.g., by referencing an additional large external text corpus, in order to achieve accurate results without increasing the model size or complexity?

https://ai.googleblog.com/2020/08/realm-integrating-retrieval-into.html

REALM (Guu et al., 2020)

Open-Domain Question Answering

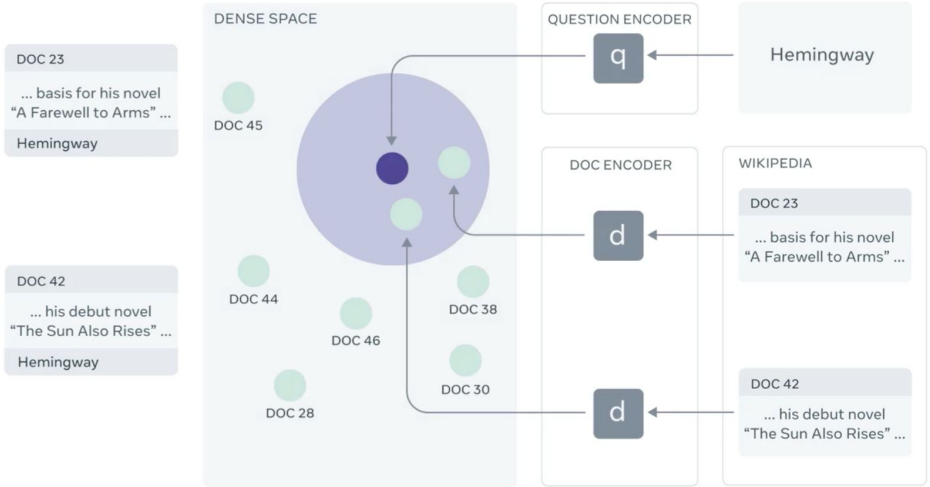
Table 1. Test results on Open-QA benchmarks. The number of train/test examples are shown in paretheses below each benchmark. Predictions are evaluated with exact match against any reference answer. Sparse retrieval denotes methods that use sparse features such as TF-IDF and BM25. Our model, REALM, outperforms all existing systems.

Name	**Architectures**	**Pre-training**	**NQ** (79k/4k)	**WQ** (3k/2k)	**CT** (1k /1k)	**# params**
BERT-Baseline (Lee et al., 2019)	Sparse Retr.+Transformer	BERT	26.5	17.7	21.3	110m
T5 (base) (Roberts et al., 2020)	Transformer Seq2Seq	T5 (Multitask)	27.0	29.1	-	223m
T5 (large) (Roberts et al., 2020)	Transformer Seq2Seq	T5 (Multitask)	29.8	32.2	-	738m
T5 (11b) (Roberts et al., 2020)	Transformer Seq2Seq	T5 (Multitask)	34.5	37.4	-	11318m
DrQA (Chen et al., 2017)	Sparse Retr.+DocReader	N/A	-	20.7	25.7	34m
HardEM (Min et al., 2019a)	Sparse Retr.+Transformer	BERT	28.1	-	-	110m
GraphRetriever (Min et al., 2019b)	GraphRetriever+Transformer	BERT	31.8	31.6	-	110m
PathRetriever (Asai et al., 2019)	PathRetriever+Transformer	MLM	32.6	-	-	110m
ORQA (Lee et al., 2019)	Dense Retr.+Transformer	ICT+BERT	33.3	36.4	30.1	330m
Ours ($\mathcal{X}$ = Wikipedia, $\mathcal{Z}$ = Wikipedia)	Dense Retr.+Transformer	REALM	39.2	40.2	**46.8**	330m
Ours ($\mathcal{X}$ = CC-News, $\mathcal{Z}$ = Wikipedia)	Dense Retr.+Transformer	REALM	**40.4**	**40.7**	42.9	330m

RAG (Lewis et al., 2020)

RAG: Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks

REALM is mostly used for short-span QA, but RAG can be used for generation-based QA.



https://ai.facebook.com/blog/retrieval-augmented-generation-streamlining-the-creation-of-intelligent-natural-language-processing-models/

RAG (Lewis et al., 2020)

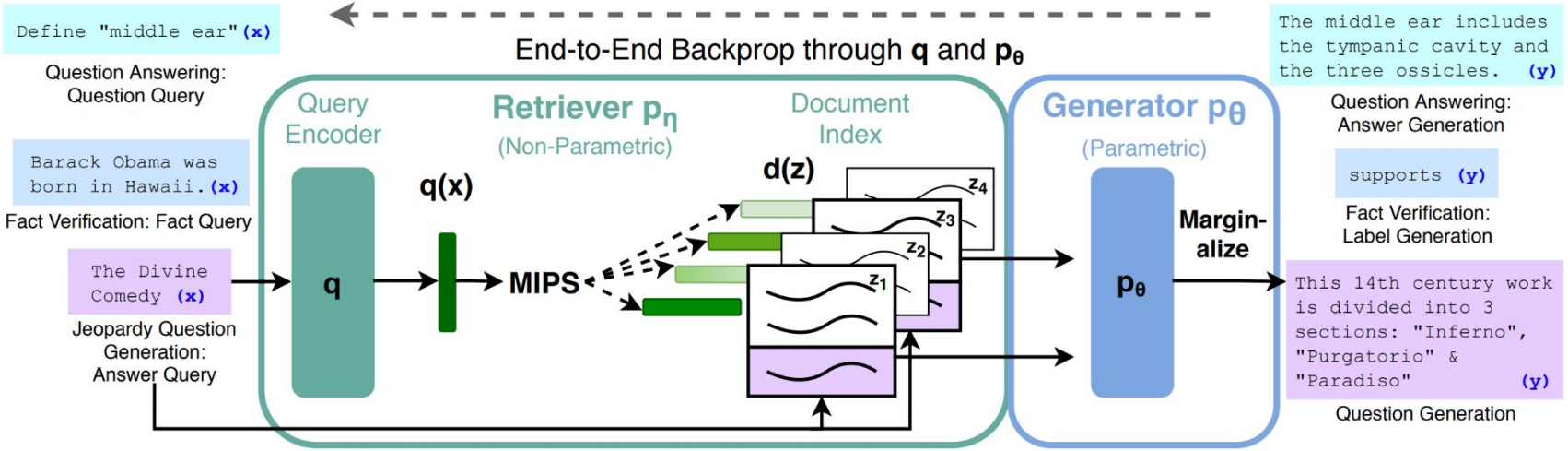


Figure 1: Overview of our approach. We combine a pre-trained retriever (*Query Encoder* + *Document Index*) with a pre-trained seq2seq model (*Generator*) and fine-tune end-to-end. For query x, we use Maximum Inner Product Search (MIPS) to find the top-K documents z_i. For final prediction y, we treat z as a latent variable and marginalize over seq2seq predictions given different documents.

ALBERT (Lan et al. 2019)

ALBERT: A Lite BERT for Self-supervised Learning of Language Representations

Can we have a smaller model but with equal performance?
Two techniques to reduce the number of parameters.

- Factorized embedding: Decompose the large vocabulary embedding matrix into two small matrices.
- Cross-layer parameter sharing.

Model		Parameters	Layers	Hidden	Embedding	Parameter-sharing
BERT	base	108M	12	768	768	False
	large	334M	24	1024	1024	False
ALBERT	base	12M	12	768	128	True
	large	18M	24	1024	128	True
	xlarge	60M	24	2048	128	True
	xxlarge	235M	12	4096	128	True

Table 1: The configurations of the main BERT and ALBERT models analyzed in this paper.

DistilBERT (Sanh et al. 2019)

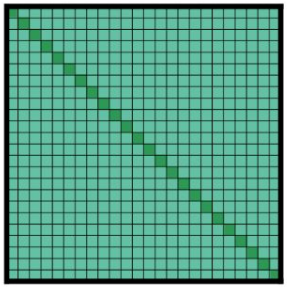
DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter

Use knowledge **distillation** during the pre-training phase.

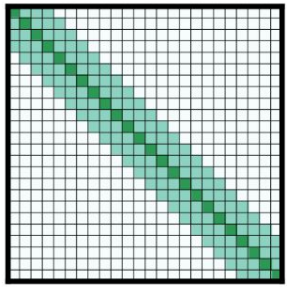
Knowledge distillation [Bucila et al., 2006, Hinton et al., 2015] is a compression technique in which a compact model - **the student** - is trained to reproduce the behaviour of a larger model - **the teacher** - or an ensemble of models.

Longformer (Beltagy et al., 2020)

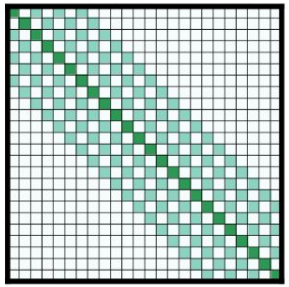
Longformer: The Long-Document Transformer



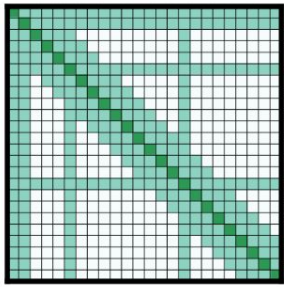
(a) Full n^2 attention



(b) Sliding window attention



(c) Dilated sliding window



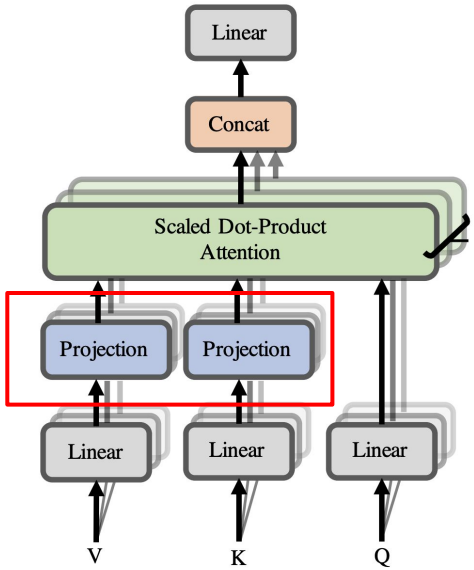
(d) Global+sliding window

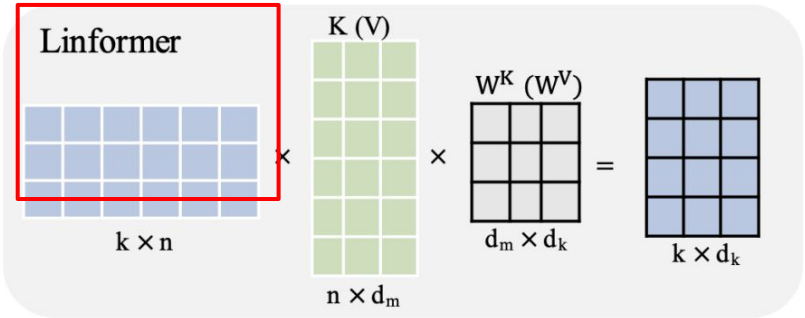
Figure 2: Comparing the full self-attention pattern and the configuration of attention patterns in our Longformer.

Linformer (Wang et al., 2020)

Add low-rank linear projections for keys and values (but not for queries).

- k: projection dimension
- n: input length
- k << n





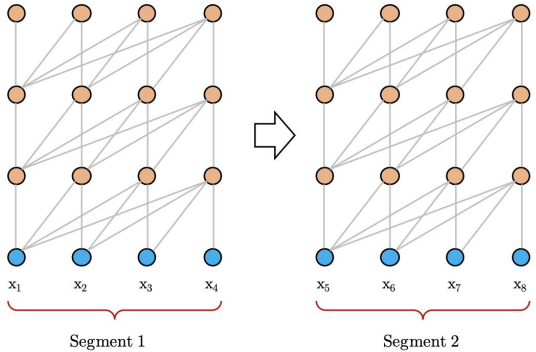
$$\overline{\text{head}_i} = \text{Attention}(QW_i^Q, E_iKW_i^K, F_iVW_i^V)$$
$$= \underbrace{\text{softmax}\left(\frac{QW_i^Q(E_iKW_i^K)^T}{\sqrt{d_k}}\right)}_{\bar{P}:n\times k} \cdot \underbrace{F_iVW_i^V}_{k\times d},$$

Transformer-XL (Dai et al., 2019)

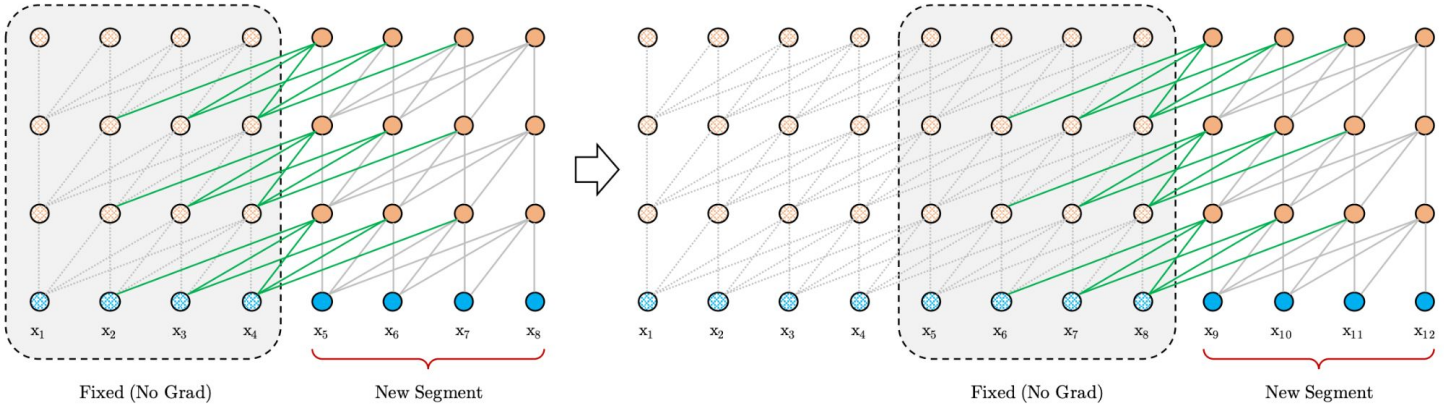
Extend beyond a fixed lengths by using segments with recurrence.

XL means extra long.

Vanilla Transformer



Transformer-XL



XLNet (Yang et al. 2019)

Model: similar as BERT, but

- **Get rid of [MASK] token** by masking attention to consider all possible different orders of prediction (see next slide).
- **Incorporate Transformer-XL** to model longer sequence.

Data: Same as BERT + Giga5, ClueWeb 2012-B, Common Crawl

Training Objective

- Permutation language modeling objective: sample a permutation of the original sentence, and predict the permuted sentence in order (see next slide).
- This combines the idea of autoregressive and autoencoding.

XLNet (Yang et al. 2019)

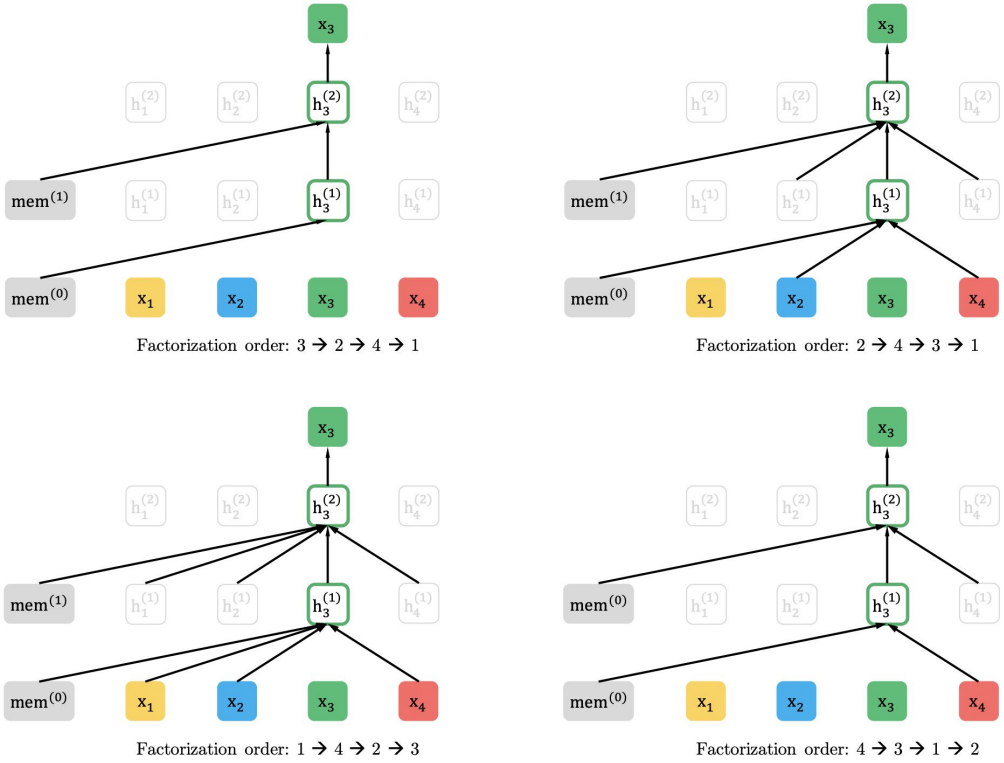
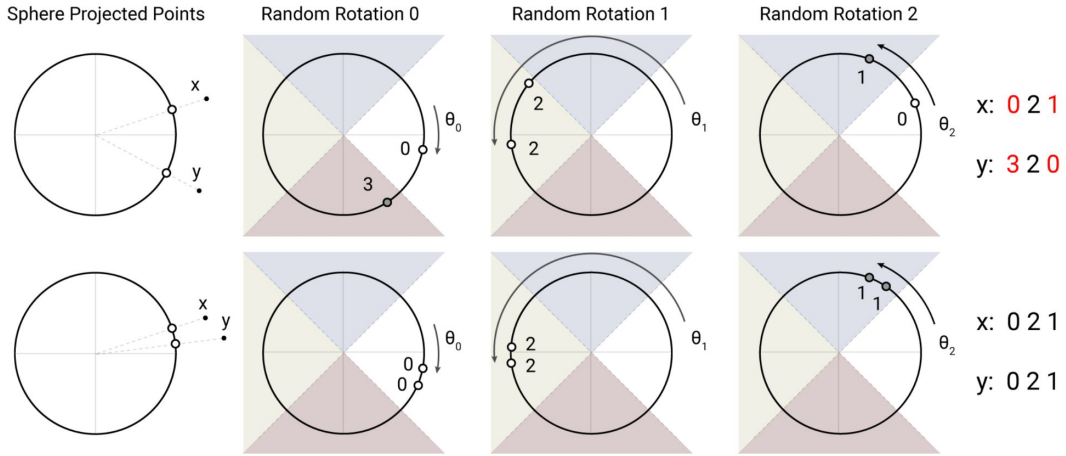


Figure 4: Illustration of the permutation language modeling objective for predicting x_3 given the same input sequence $\mathbf{x}$ but with different factorization orders.

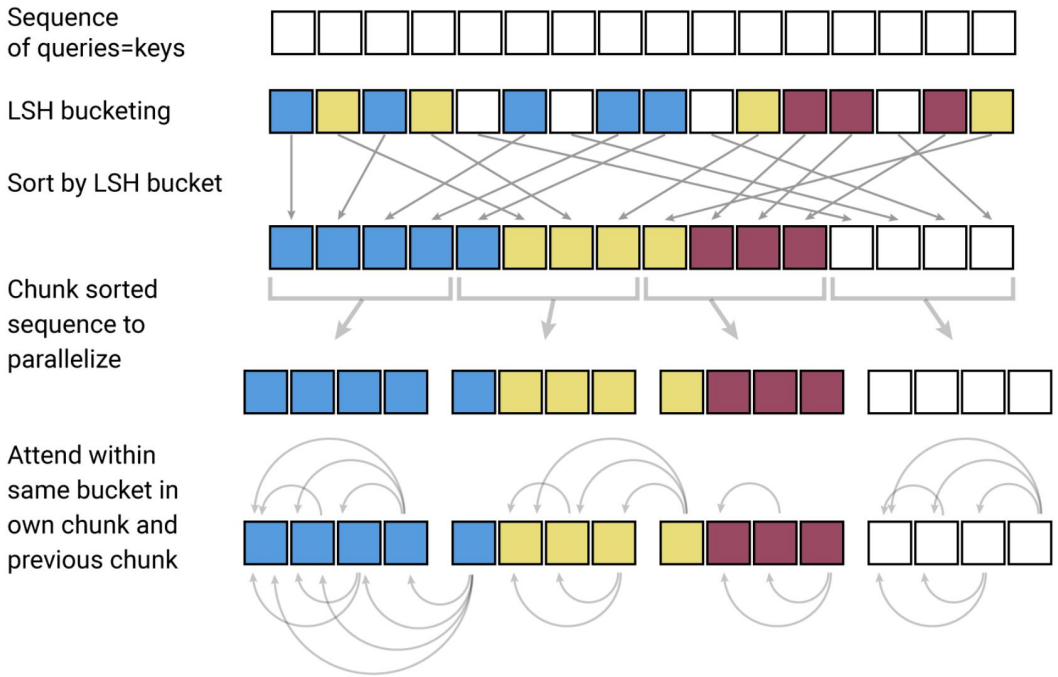
Reformer (Kitaev et al., 2019)

- During attention computation, we are interested in softmax(QK^T).
- Since softmax is dominated by the largest elements, we are only interested in nearest neighbors during attention computation.
- For example, if K is of length 64, for each qi we could only consider a small subset of, say, the 32 or 64 closest keys.
- This is much more efficient, but how can we find the nearest neighbors among the keys?
- Finding nearest neighbors quickly in high-dimensional spaces can be solved by locality-sensitive hashing (LSH).



Reformer (Kitaev et al., 2019)

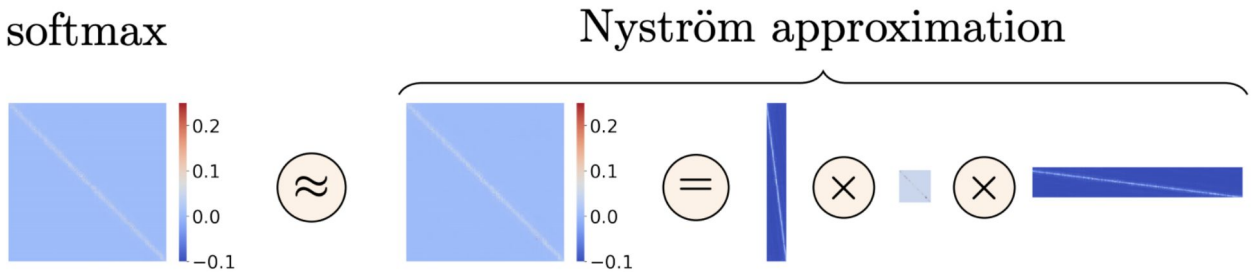
- Replace dot-product attention by one that uses LSH.
- Changing its complexity from $O(L^2)$ to $O(L \log L)$.



Performer (Choromanski et al., 2020)

Other Efficient Transformers

- Sparse Transformers (Child et al., 2019)
  - Introduce sparse factorizations of the attention matrix.
  - Reduce quadratic complexity to $O(n \sqrt{n})$.
- Routing Transformers (Roy et al., 2020)
  - Content-based sparse attention.
  - Tokens are assigned to different clusters.
  - Attention is performed only within each cluster.
- Nystromformer (Xiong et al., 2021)
  - Approximate self-attention using the Nystrom method.



Efficient Transformers: A Survey (Tay et al., 2020)

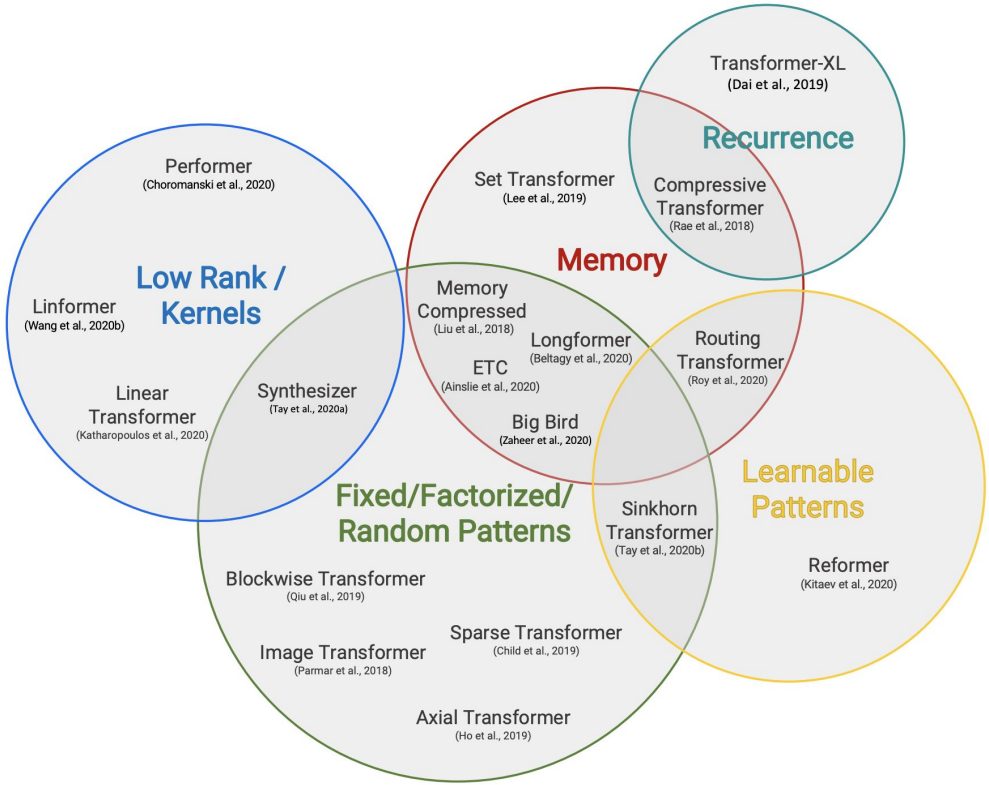
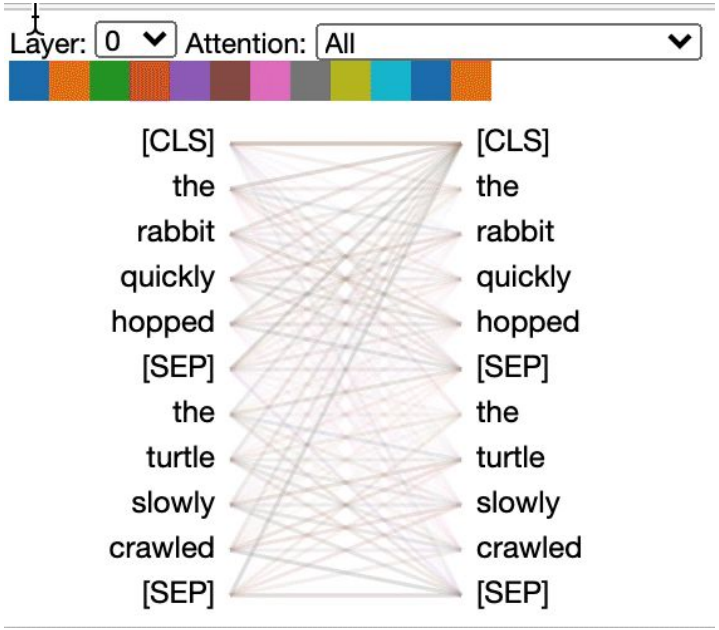


Figure 2: Taxonomy of Efficient Transformer Architectures.

Analysis on BERT

BertViz is an interactive tool for visualizing attention in Transformer language models such as BERT, GPT-2, or T5. Check out its code and demo here: https://github.com/jessevig/bertviz



Analysis on BERT

What Does BERT Look At? An Analysis of BERT's Attention (Clark et al., 2019)

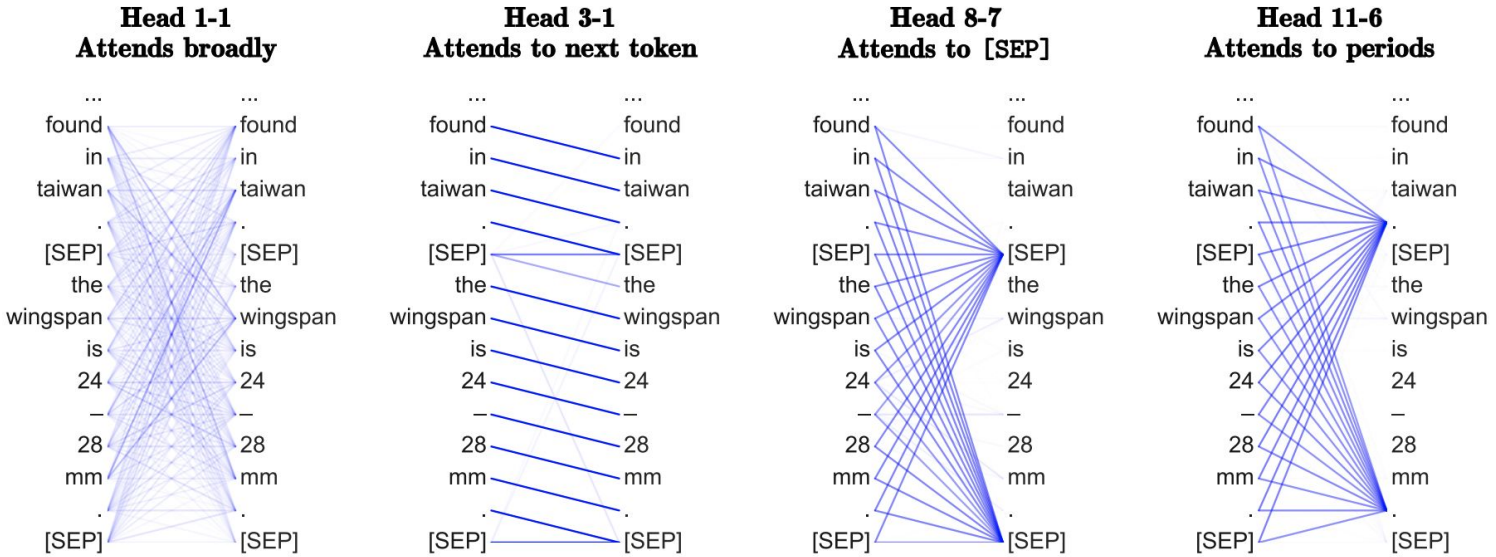


Figure 1: Examples of heads exhibiting the patterns discussed in Section 3. The darkness of a line indicates the strength of the attention weight (some attention weights are so low they are invisible).

Analysis on BERT

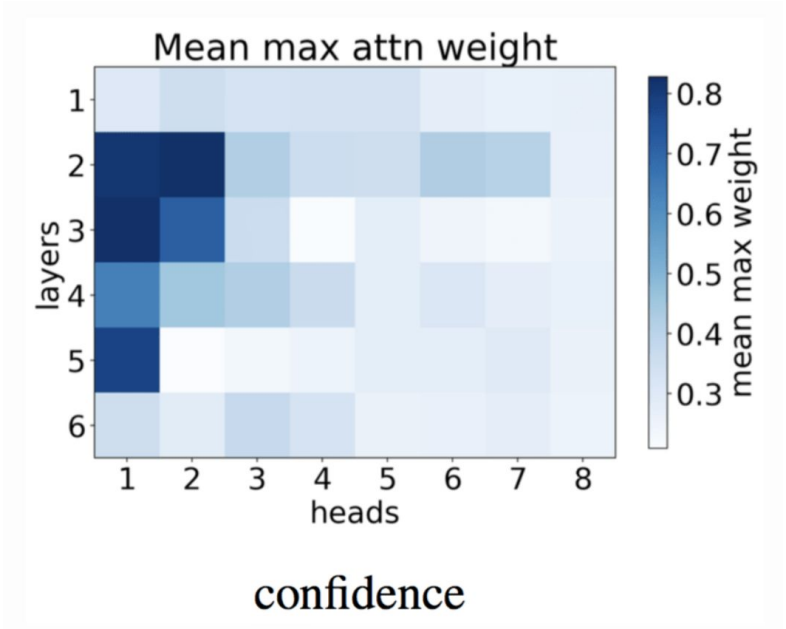
Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned (Voita et al., 2019)

- Analyze Multi-head self-attention in Neural Machine Translation
- Most important and confident heads play consistent and often linguistically-interpretable roles.
- Can **prune** less important heads.

Analysis on BERT

Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned (Voita et al., 2019)

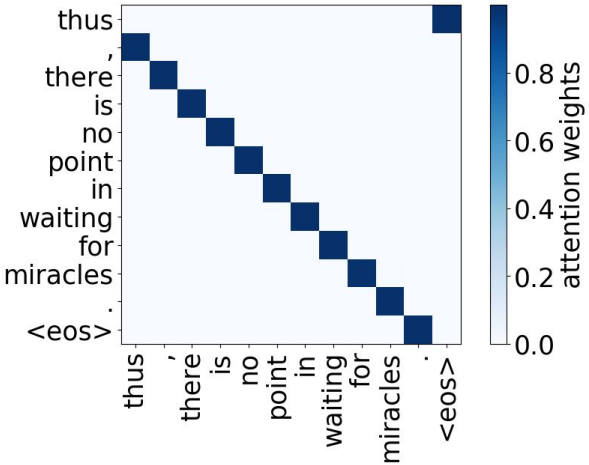
- Heads have different "confidence", measured as average of its maximum attention weight.



Analysis on BERT

Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned (Voita et al., 2019)

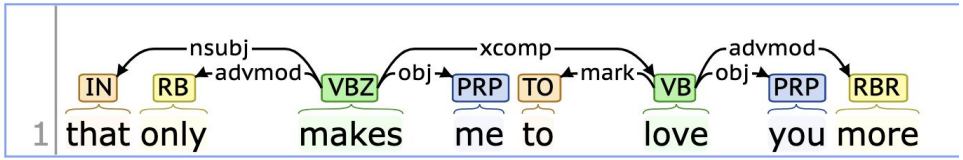
- There are different types of heads.
- **Positional heads**: We refer to a head as “positional” if at least 90% of the time its maximum attention weight is assigned to a specific relative position (in practice either -1 or +1, i.e. attention to adjacent tokens).



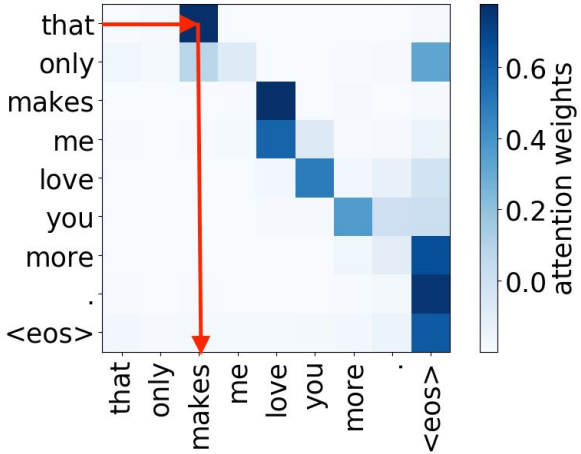
Analysis on BERT

[Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned](#) (Voita et al., 2019)

- There are different types of heads.
- **Syntactic heads:** comparing its attention weights to a predicted dependency structure generated using CoreNLP.



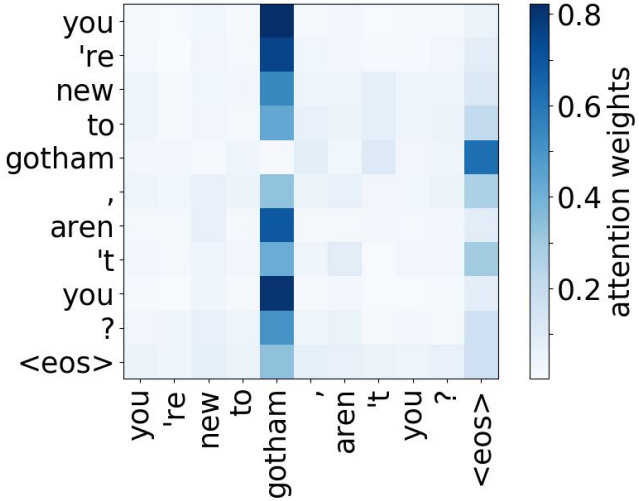
https://corenlp.run/



Analysis on BERT

Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned (Voita et al., 2019)

- There are different types of heads.
- **Rare tokens**: For all models, we find a head pointing to the least frequent tokens in a sentence.



Analysis on BERT

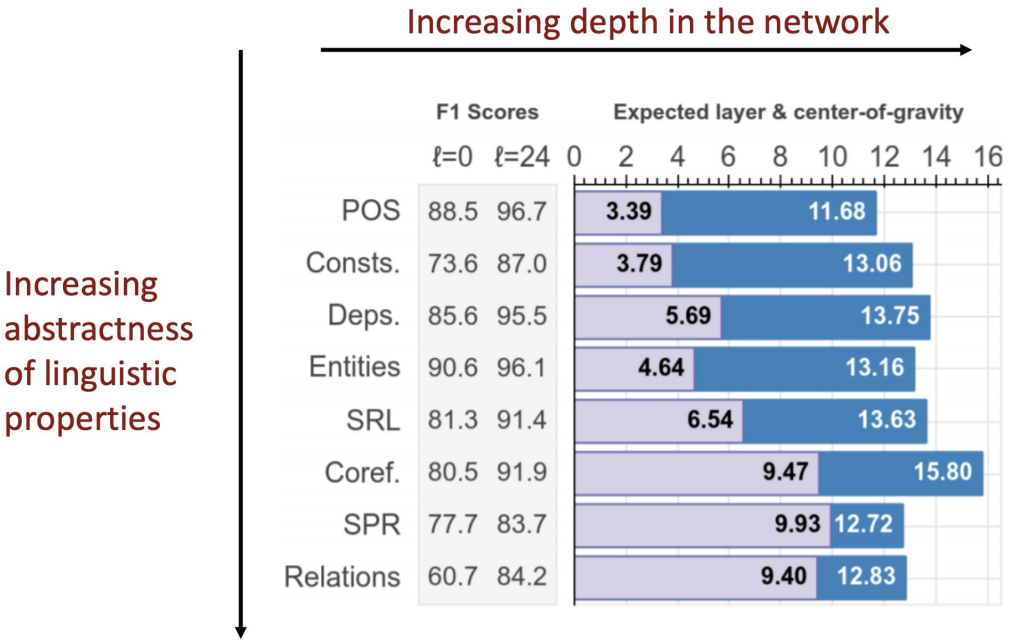
Are Sixteen Heads Really Better than One? (Michel, et al., 2019)

- Make the surprising observation that even if models have been trained using multiple heads, in practice, **a large percentage of attention heads can be removed** at test time without significantly impacting performance.
- In fact, some layers can even be reduced to a **single head.**

Analysis on BERT

BERT Rediscovers the Classical NLP Pipeline (Tenney et al., 2019)

- Quantify where linguistic information is captured within the network.
- Find that the model represents the steps of the traditional NLP pipeline in an interpretable and localizable way, and that the regions responsible for each step appear in the expected sequence: POS tagging, parsing, NER, semantic roles, then coreference



BERT Code Demo

Hugging Face Transformers library
Hugging Face course

Code Demo

- BERT for predicting masked tokens.
- BART for summarization
- GPT-2 for text generation.

Can a LM ever truly understand natural language?

https://newsletter.ruder.io/issues/palm-dall-e-2-chinchilla-chain-of-thought-prompting-values-and-culture-in-nlp-845878?utm_campaign=Issue&utm_content=view_in_browser&utm_medium=email&utm_source=NLP+News

https://newsletter.ruder.io/issues/ml-and-nlp-starter-toolkit-low-resource-nlp-toolkit-can-a-lm-understand-natural-language-the-next-generation-of-nlp-benchmarks-254211?utm_campaign=NLP%20News&utm_medium=email&utm_source=Revue%20newsletter

https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00412/107385/Provable-Limitations-of-Acquiring-Meaning-from?utm_campaign=NLP%20News&utm_medium=email&utm_source=Revue%20newsletter

https://arxiv.org/abs/2004.10151