TransferLearning

MIPT
11.03.2021
Anton Emelianov, Alena Fenogenova.

2021

https://github.com/king-menin/mipt-nlp2021

Today

- Transfer learning
- Pretraining models:
  - BERT and Bertology
  - GPTs
  - T5
- Evaluation
  - Benchmarks (SuperGLUE)
  - Probing

Transfer learning. Motivation

MY TASK

- I have small dataset
- Task specific data (with labels)
- I need a good quality! O_o

SOME OTHER TASK

- I have huge amount of data
- Unlabeled data
- No task-specific, just general domain

The general idea of transfer learning -

to "transfer" knowledge from one task/model to another.

Transfer learning. Pretrain models

Two main aspects of pretrained models:

1. From Words to Words-in-Context

Instead of representing individual words, we can learn to represent words **along with the context they are used in.**

- representations of language
- put context (because for know Meaning we need to have a context)

2. Pretraining/Finetuning

From replacing only word embeddings in task-specific models to replacing entire task-specific models

Transfer learning. Representations

My friend is an **apple** guy, but I prefer Samsung.

My friend likes **apples**, but I am allergic for it.

Idea:

If we use word embeddings, the vector for **apple** will contain information about the general notion of apple.

We need vectors that represent not just words, but ***words in context***!

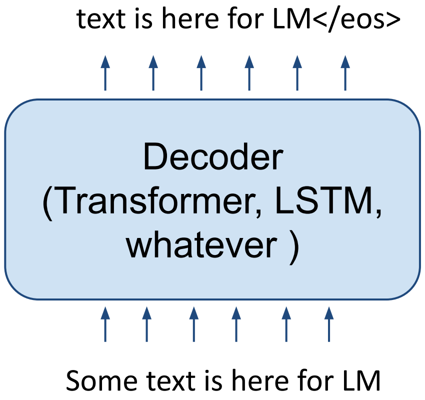
Encoding words with context:

- CoVe: Contextualized Word Vectors Learned in Translation Learned in Translation: Contextualized Word Vectors
- ELMo Embeddings from Language Models Deep contextualized word representations
- see next ...

The Pretraining / Finetuning Paradigm

Pretrain through language modeling

Train a NN to perform on a large amount of text. Save the network parameters

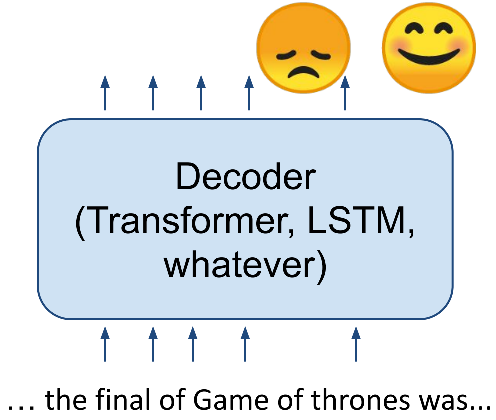


Fine-tune

Take a pretrained model

Train for the specific task (e.g., sentiment classification) with a rather small learning rate

Make good quality using LM knowledge



Transfer learning. Word structure and subword models

type	**word**	**vocab mapping**	**word**	**vocab mapping**
common words	cool	cool (index)	cool	cool
	learn	learn(index)	learn	learn
variations	coool	UNK	coool	coo# ol##
misspelkings	laern	UNK	laern	la## ern##
novel items	Bertology	UNK	Bertology	Bert## ology

We want a tokenization scheme that:

- can deal with an infinite potential vocabulary via a finite list of known words
- don’t break everything into single characters since character-level tokenization (can lose some of the meaning and semantic niceties of the word level).

Hybrid between word-level and character-level tokenization = **subword tokenization**.

- *BPE* (https://arxiv.org/pdf/1508.07909.pdf)
- *WordPiece* (https://static.googleusercontent.com/media/research.google.com/ja//pubs/archive/37842.pdf)
- *SentencePiece* (https://www.aclweb.org/anthology/D18-2012.pdf) unicode characters are grouped together using either a unigram language model

Transfer learning. BPE

Byte Pair Encoding

- The dominant modern paradigm is to learn a vocabulary of parts of words (subword tokens).
- At training and testing time, each word is split into a sequence of known subwords.

BPE relies on a pre-tokenizer that splits the training data into words. The algorithm starts with a character-level tokenization and then iteratively merges most frequent pairs for N iterations.

Byte-pair encoding algo:

1. Training part = learn BPE rules, make merge table:
   a. count pairs of symbols: how many times each pair occurs together in the training data;
   b. find the most frequent pair of symbols;
   c. merge this pair - add a merge to the merge table, and the new token to the vocabulary.
2. Inference = apply learned rules
   a. segment text in characters
   b. among all possible merges at this step, find the highest* merge in the table;
   c. apply this merge

*the merges that are higher in the table were more frequent in the data

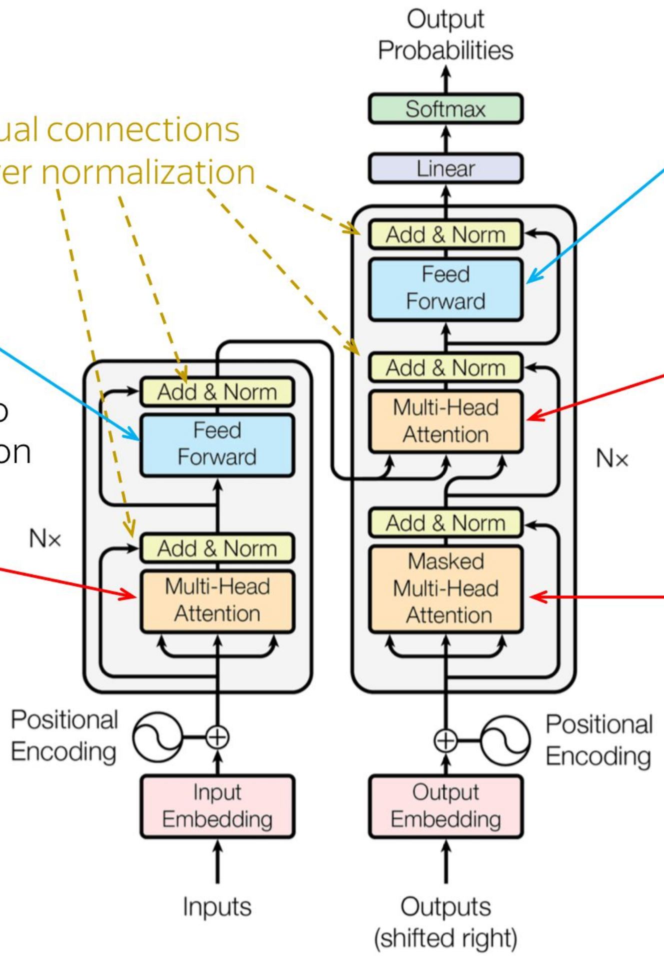
Transfer learning. Transformers

Residual connections and layer normalization

Feed-forward network: after taking information from other tokens, take a moment to think and process this information

Encoder self-attention: tokens look at each other

queries, keys, values are computed from encoder states



Feed-forward network: after taking information from other tokens, take a moment to think and process this information

Decoder-encoder attention: target token looks at the source

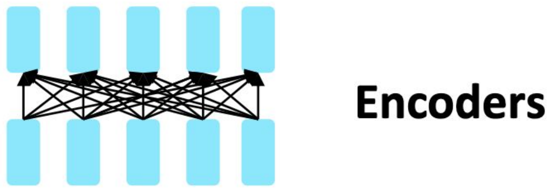
queries – from decoder states; keys and values from encoder states

Decoder self-attention (masked): tokens look at the previous tokens

queries, keys, values are computed from decoder states

Transfer learning

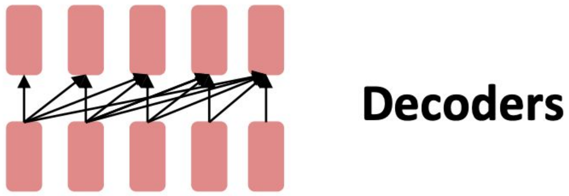
The neural architecture influences the three types of pretraining, and natural use cases:



1) Encoder part

   (Gets bidirectional context – can condition on future)

   Spoiler: like BERT



2) Language models. Decoder part

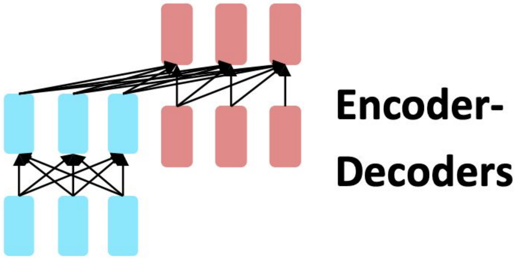
   (Nice to generate from; can’t condition on future words)

   Spoiler: like GPTS

3) Encoder-decoder architecture

   (best from both parts)

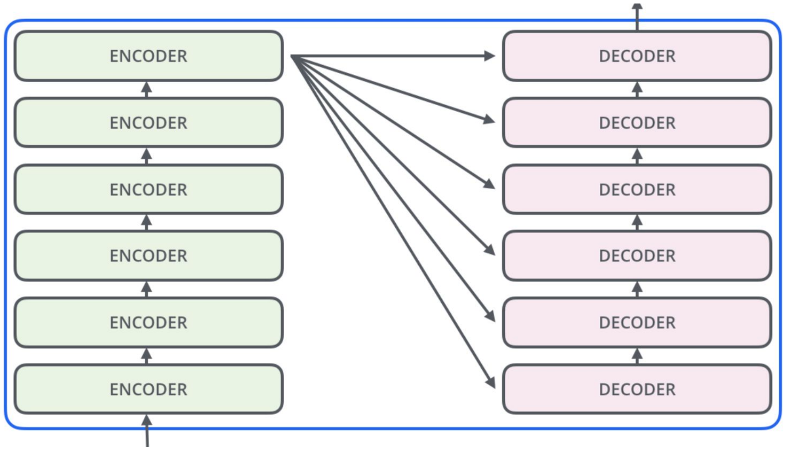
   Spoiler: like T5 for example



BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Transformer:

$$d_k = \frac{d_{model}}{A} = \frac{512}{8} = 64$$



Model architecture is the same

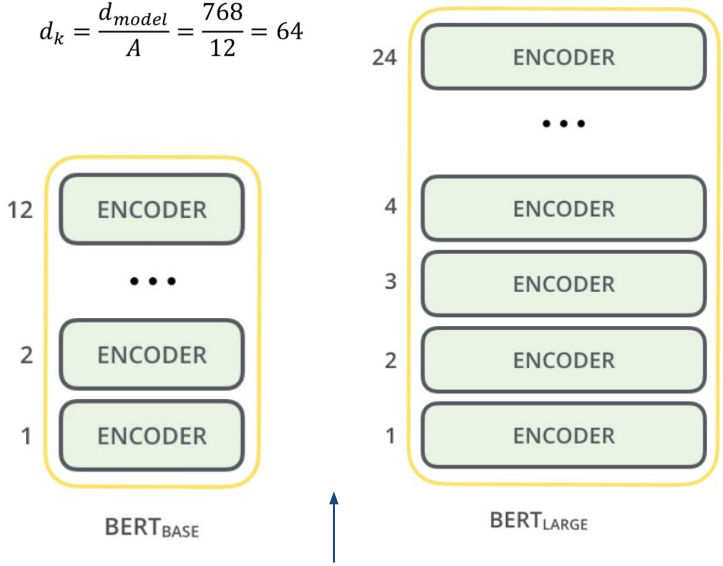
Training objectives are different:

- NSP
- MLM

Bert:

$$d_k = \frac{d_{model}}{A} = \frac{1024}{16} = 64$$

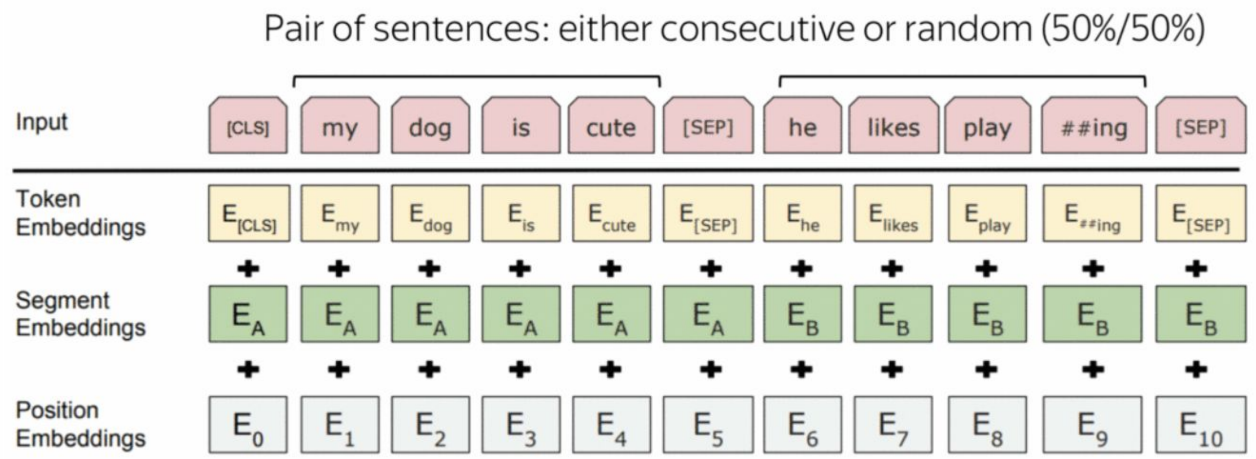
$$d_k = \frac{d_{model}}{A} = \frac{768}{12} = 64$$



[CLS] some other input tokens ...

BERT. Pretraining. Input. NSP

BERT see pairs of sentences separated with a special token-separator **[SEP]**



BERT was trained to **predict whether one sentence follows the other or is randomly sampled => Next Sentence Prediction (NSP) Objective**.

From the final-layer representation of the special token **[CLS]**, the model predicts whether the two sentences are consecutive sentences in some text or not.

Binary classification. In training, 50% of examples real sentences extracted from training texts and 50% - a random pair of sentences.

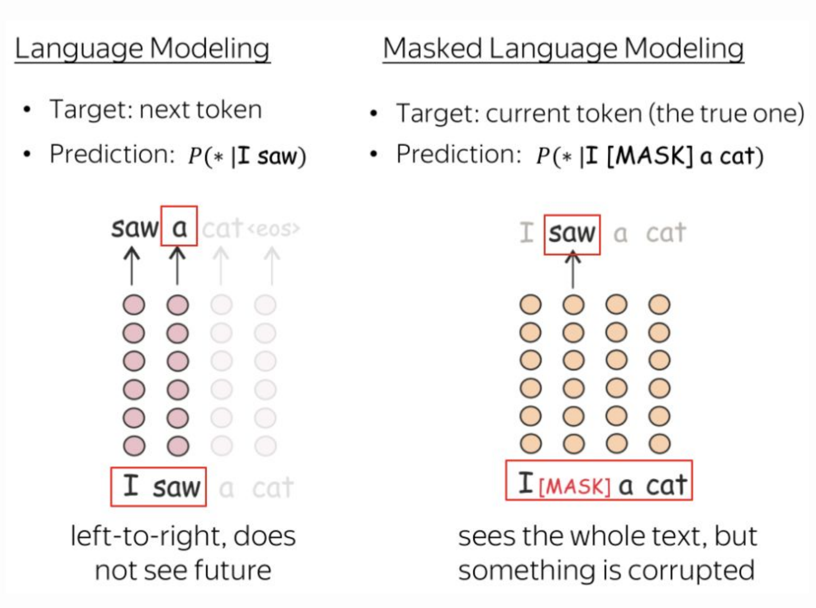
BERT. Masked Language Modeling Objective

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

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

Only add loss terms from words that are "masked out." If x~ is the masked version of we're learning P_theta(x|x~).

1. Select some tokens random 15% of (sub)word tokens
2. Replace selected 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 t (but still predict it!)
3. Predict selected tokens



Language Modeling

- Target: next token
- Prediction: $P(* \mid \text{I saw})$

left-to-right, does not see future

Masked Language Modeling

- Target: current token (the true one)
- Prediction: $P(* \mid \text{I [MASK] a cat})$

sees the whole text, but something is corrupted

BERT

Paper: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Parameters:

- 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.

Dataset:

- BooksCorpus (800 million words)
- English Wikipedia (2,500 million words)

Pretraining:

- BERT was pretrained with 64 TPU chips for a total of 4 days.
- (TPUs are special tensor operation acceleration hardware)

Tokenizer:

- WordPiece

BERT. Usage

Task-specific:

1. Single sentence classification

To classify individual sentences, feed the data and predict the label from the final representation of the **[CLS]** token.

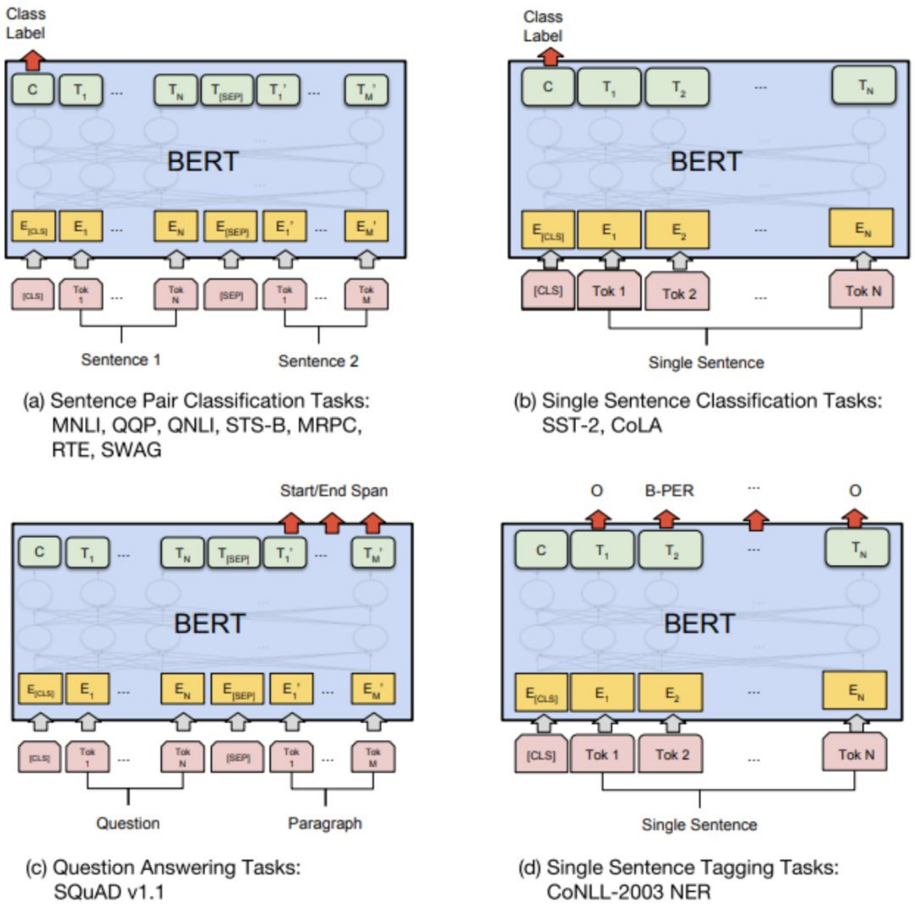
2. Sentence pair classification

Feed the data as you did in training (two sentences with SEP). Predict the label from the final representation of the **[CLS]** token

BERT for feature extraction:

use the pre-trained BERT to create

contextualized word embeddings



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

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

(c) Question Answering Tasks: SQuAD v1.1

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

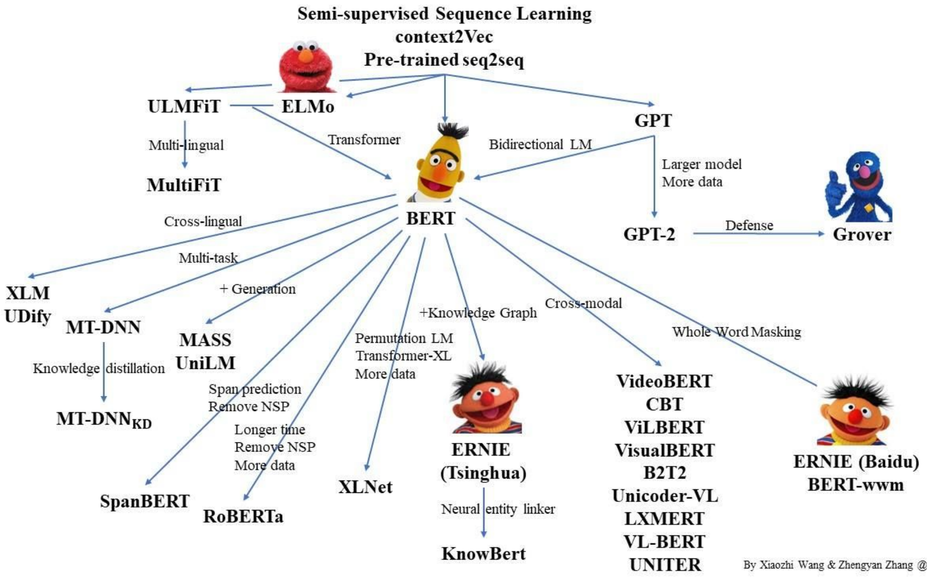
BERTology

RoBERTa:

- Facebook AI research
- More data: 160GB of text instead of the 16GB dataset originally used to train BERT.
- Longer training: increasing the number of iterations from 100K to 300K and then further to 500K.
- Larger batches: 8K instead of 256 in the original BERT base model.
- Larger byte-level BPE vocabulary with 50K subword units.
- Removing the next sequence prediction objective from the training procedure.
- Dynamically changing the masking pattern applied to the training data.

Albert:

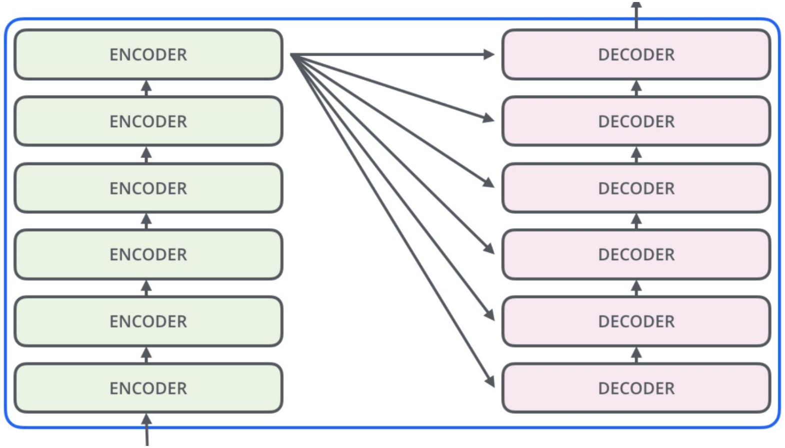
- Google research
- It is not reasonable to further improve language models by making them larger.
- Two parameter-reduction techniques:
  - **factorized embedding parameterization**, where the size of the hidden layers is separated from the size of vocabulary embeddings by decomposing the large vocabulary-embedding matrix into two small matrices;
  - **cross-layer parameter sharing** to prevent the number of parameters from growing with the depth of the network.
- Improved by introducing the self-supervised loss for **sentence-order prediction** to address BERT's limitations with regard to inter-sentence coherence.



DistilBERT, SpanBERT, KnowBERT, etc.

GPT. Generative Pre-Training for Language Understanding

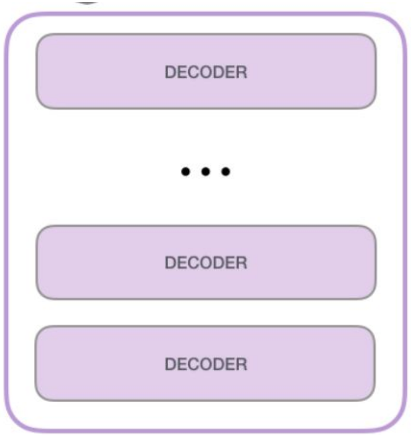
Transformer:



OpenAI. GPT:

left-to-right language model

The architecture is a 12-layer Transformer decoder



- GPT-1: Improving Language Understanding by Generative Pre-Training
- GPT-2: Language Models are Unsupervised Multitask Learners
- GPT-3: Language Models are Few-Shot Learners

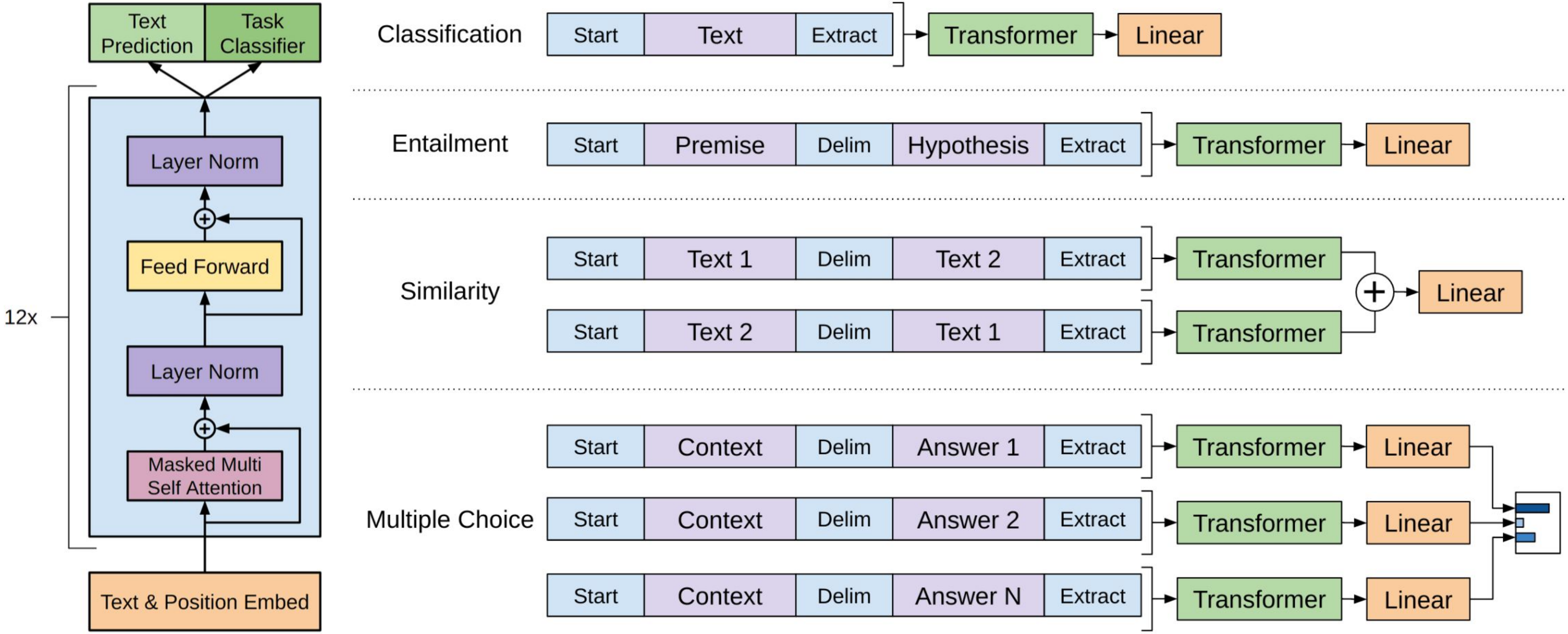
GPT

1) Unsupervised Language Modelling

$$L_{xent} = -\sum_{t=1}^{n} \log(p(y_t|y_{<t}))$$

2) Supervised Fine-Tuning $L = L_{xent} + \lambda \cdot L_{task}.$

The linear classifier is applied to the representation of the [EXTRACT] token.



GPTs

1

Paper: Improving Language Understanding by Generative Pre-Training

Parameters:

117M parameters

Decoder with 12 layers. 768-dimensional hidden states, 3072-dimensional feed-forward hidden layers

Dataset:

BooksCorpus: over 7000 unique books

Tokenizer:

BPE with 40,000 merges

2

Paper: Language Models are Unsupervised Multitask Learners

Parameters:

1.5 billion parameters

48 layers and used 1600 dimensional vectors for word embedding

Larger batch size of 512 and larger context window of 1024 tokens

Dataset:

WebText, had 40GB of text data from over 8 million documents.

Tokenizer:

BPE 50,257

3

Paper: Improving Language Understanding by Generative Pre-Training

Parameters:

175 billion parameters

96 layers each layer has 96 attention heads

Size of word embeddings 12888.

Context window size 2048 tokens

Alternating dense and locally banded sparse attention patterns

Dataset:

five datasets: Common Crawl, WebText2, Books1, Books2 and Wikipedia

Tokenizer:

BPE

GPTs

Pros:

- Task conditioning

  the model is expected to produce different output for same input for different tasks

- Zero-shot and few-shot capabilities

  Zero shot learning is a special case of zero shot task transfer where no examples are provided at all and the model understands the task based on the given instruction

Contrs:

- high quality text, at times it starts losing coherency while formulating long sentences and repeats sequences of text over and over again
- the language biases
- complex and costly inferencing from model due to its heavy architecture, less interpretability of the language and results generated

https://openai.com/blog/better-language-models/

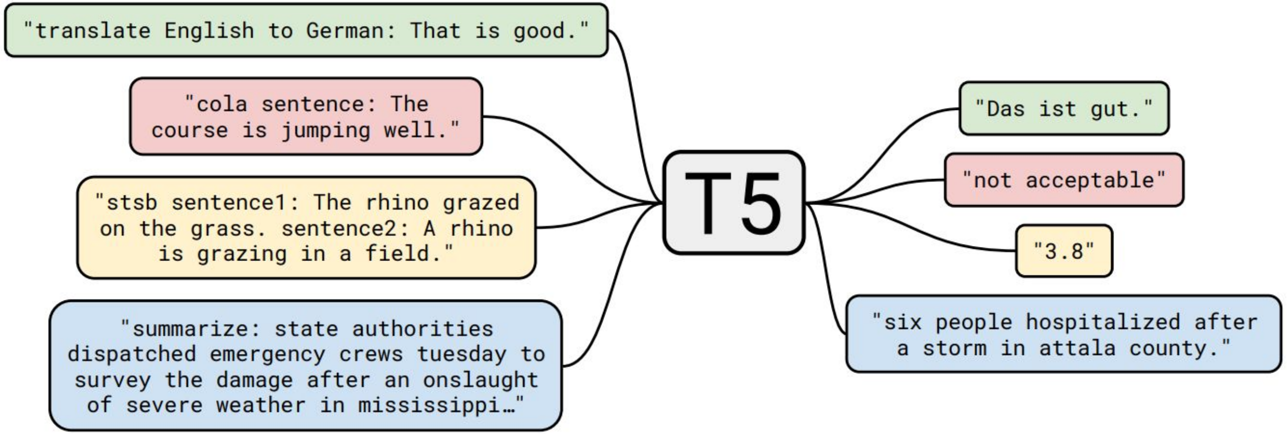
Text-to-Text Transfer Transformer T5

Encoder decoder architecture

Language modeling, but where a prefix of every input is provided to the encoder and is not predicted.

Treat every NLP problem as a **Text-to-text** task

(where the input and output are always text strings)



"translate English to German: That is good."

"cola sentence: The course is jumping well."

"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."

"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi…"

T5

"Das ist gut."

"not acceptable"

"3.8"

"six people hospitalized after a storm in attala county."

T5

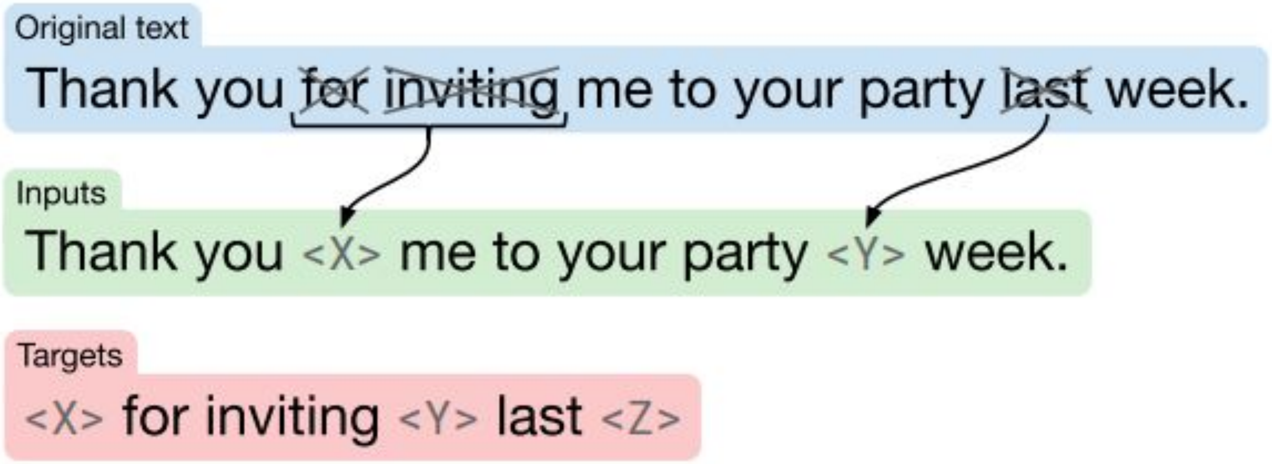
T5 trains with Masked Language Model.

The Original text is transformed into **Input** and **Output** pairs:

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

Final objective is to train a model that inputs text and outputs text; is trained to predict basically sentinel tokens to delineate the dropped out text. Teacher forcing (for training we always need an input sequence and a target sequence)

The targets were designed to produce a sequence, that tries to output one word (itself) through final feed-forward and softmax at the output level.



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>

T5

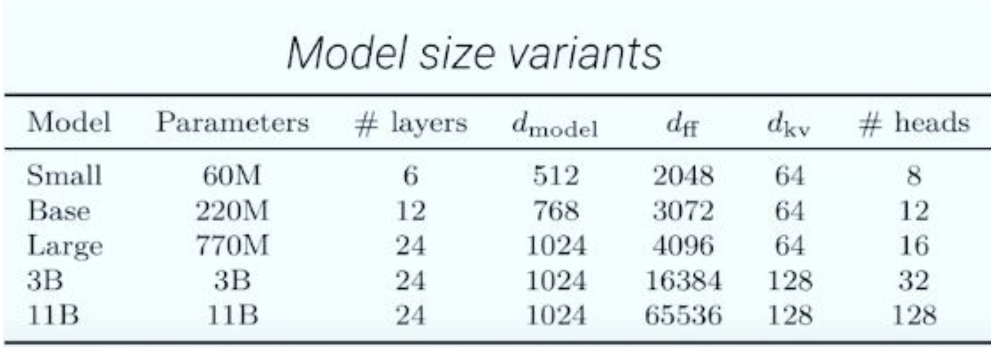
Paper: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

Parameters:

5 variants: small model, base model,

large model, and models

with 3 billion and 11 billion parameters



Model size variants

Model	Parameters	# layers	d_{model}	d_{ff}	d_{kv}	# heads
Small	60M	6	512	2048	64	8
Base	220M	12	768	3072	64	12
Large	770M	24	1024	4096	64	16
3B	3B	24	1024	16384	128	32
11B	11B	24	1024	65536	128	128

Dataset:

Colossal Clean Crawled Corpus (C4). Common Crawl

700GB cleaning in sense of extracting only English text, removing code lines, deduplicating, etc

Fine-tuning tasks: GLUE, CNN/DM(CNN / Daily Mail), SQuAD, SuperGLUE, and translation tasks:

WMT14 EnDe, WMT14 EnFr, and WMT14 EnRo

Tokenizer:

SentencePiece

Limitations of traditional transformers

Quadratic compute in self-attention:

- Computing all pairs of interactions means our computation grows quadratically with the sequence length!
- For recurrent models, it only grew linearly!

=> Linformer (Linformer: Self-Attention with Linear Complexity)

=> BigBird (Big Bird: Transformers for Longer Sequences)
(sparse attention)

Evaluation

- LMs went through the advanced stages of natural language modelling.
- Universal transformers show ability to extract complicated relationships from texts.
- There are a lot of the pretrained models...

HOW TO TEST WHICH MODEL IS BETTER?

- You can use Perplexity!

  Perplexity is the standard evaluation metric for language models. Perplexity is the inverse probability of test set which is normalised by number of words in test set.

  Language models with lower perplexity are considered to better than ones with higher perplexity.

- Benchmark approach

  Testing general intellectual “abilities” in a text format

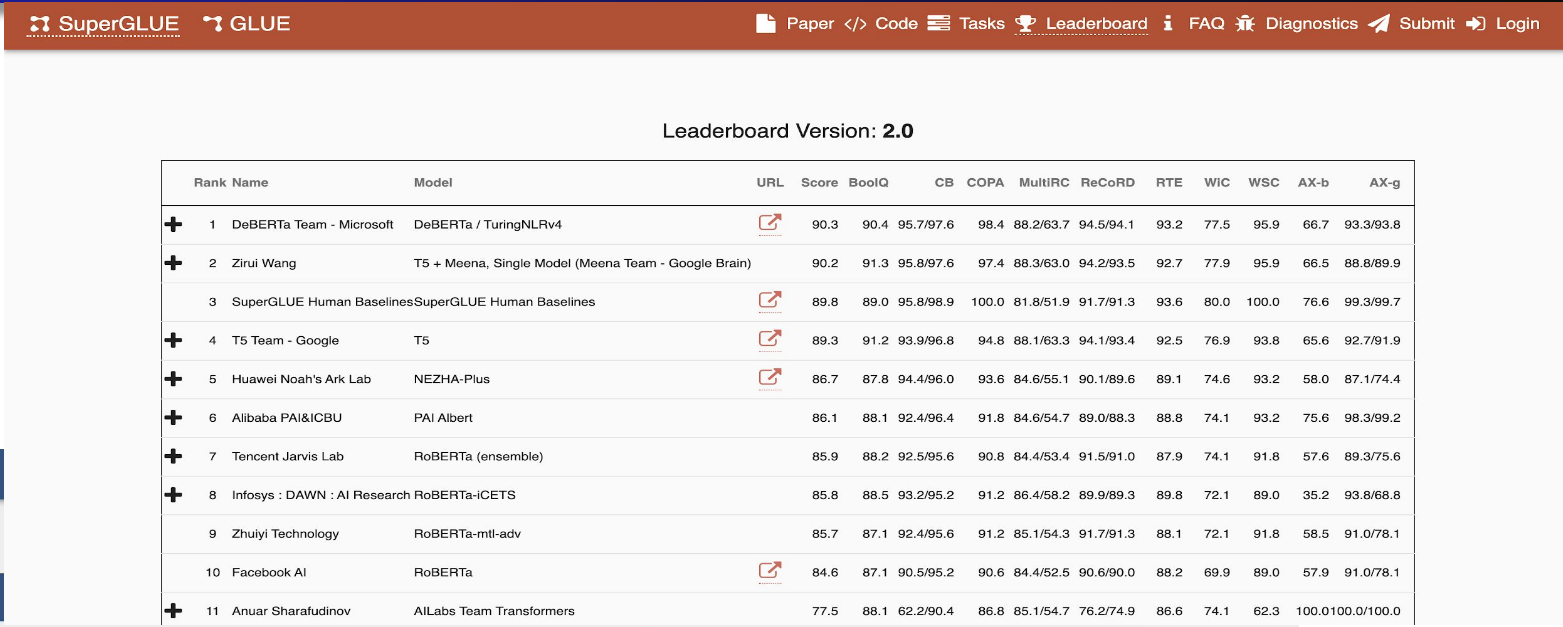
Evaluation. Benchmarks

GLUE - 2019

(Human baseline - 15 place)

SuperGLUE - 2020

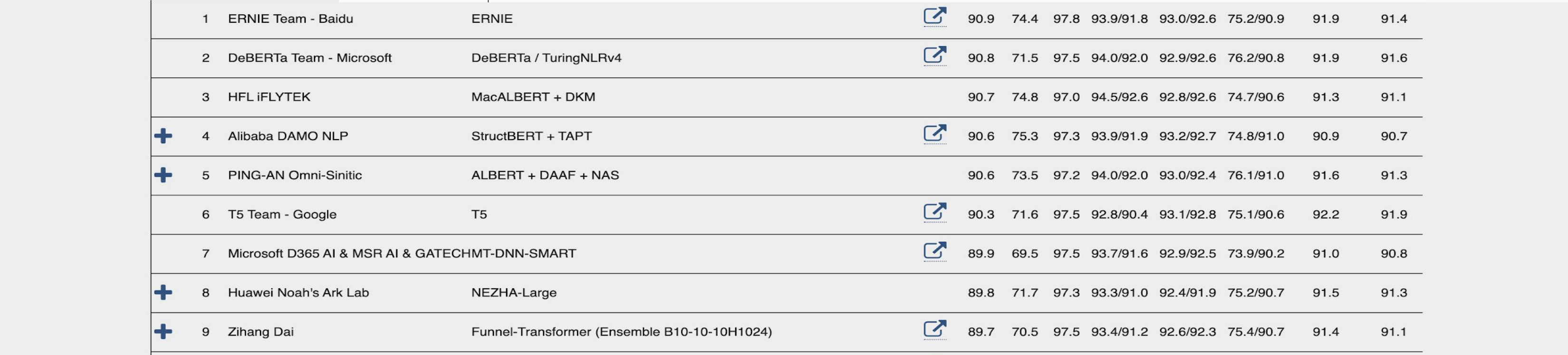
(Human baseline is already beaten)



Leaderboard Version: **2.0**

	Rank	Name	Model	URL	Score	BoolQ	CB	COPA	MultiRC	ReCoRD	RTE	WiC	WSC	AX-b	AX-g
+	1	DeBERTa Team - Microsoft	DeBERTa / TuringNLRv4	↗	90.3	90.4	95.7/97.6	98.4	88.2/63.7	94.5/94.1	93.2	77.5	95.9	66.7	93.3/93.8
+	2	Zirui Wang	T5 + Meena, Single Model (Meena Team - Google Brain)		90.2	91.3	95.8/97.6	97.4	88.3/63.0	94.2/93.5	92.7	77.9	95.9	66.5	88.8/89.9
	3	SuperGLUE Human Baselines	SuperGLUE Human Baselines	↗	89.8	89.0	95.8/98.9	100.0	81.8/51.9	91.7/91.3	93.6	80.0	100.0	76.6	99.3/99.7
+	4	T5 Team - Google	T5	↗	89.3	91.2	93.9/96.8	94.8	88.1/63.3	94.1/93.4	92.5	76.9	93.8	65.6	92.7/91.9
+	5	Huawei Noah's Ark Lab	NEZHA-Plus	↗	86.7	87.8	94.4/96.0	93.6	84.6/55.1	90.1/89.6	89.1	74.6	93.2	58.0	87.1/74.4
+	6	Alibaba PAI&ICBU	PAI Albert		86.1	88.1	92.4/96.4	91.8	84.6/54.7	89.0/88.3	88.8	74.1	93.2	75.6	98.3/99.2
+	7	Tencent Jarvis Lab	RoBERTa (ensemble)		85.9	88.2	92.5/95.6	90.8	84.4/53.4	91.5/91.0	87.9	74.1	91.8	57.6	89.3/75.6
+	8	Infosys : DAWN : AI Research	RoBERTa-iCETS		85.8	88.5	93.2/95.2	91.2	86.4/58.2	89.9/89.3	89.8	72.1	89.0	35.2	93.8/68.8
	9	Zhuiyi Technology	RoBERTa-mtl-adv		85.7	87.1	92.4/95.6	91.2	85.1/54.3	91.7/91.3	88.1	72.1	91.8	58.5	91.0/78.1
	10	Facebook AI	RoBERTa	↗	84.6	87.1	90.5/95.2	90.6	84.4/52.5	90.6/90.0	88.2	69.9	89.0	57.9	91.0/78.1
+	11	Anuar Sharafudinov	AILabs Team Transformers		77.5	88.1	62.2/90.4	86.8	85.1/54.7	76.2/74.9	86.6	74.1	62.3	100.0	100.0/100.0

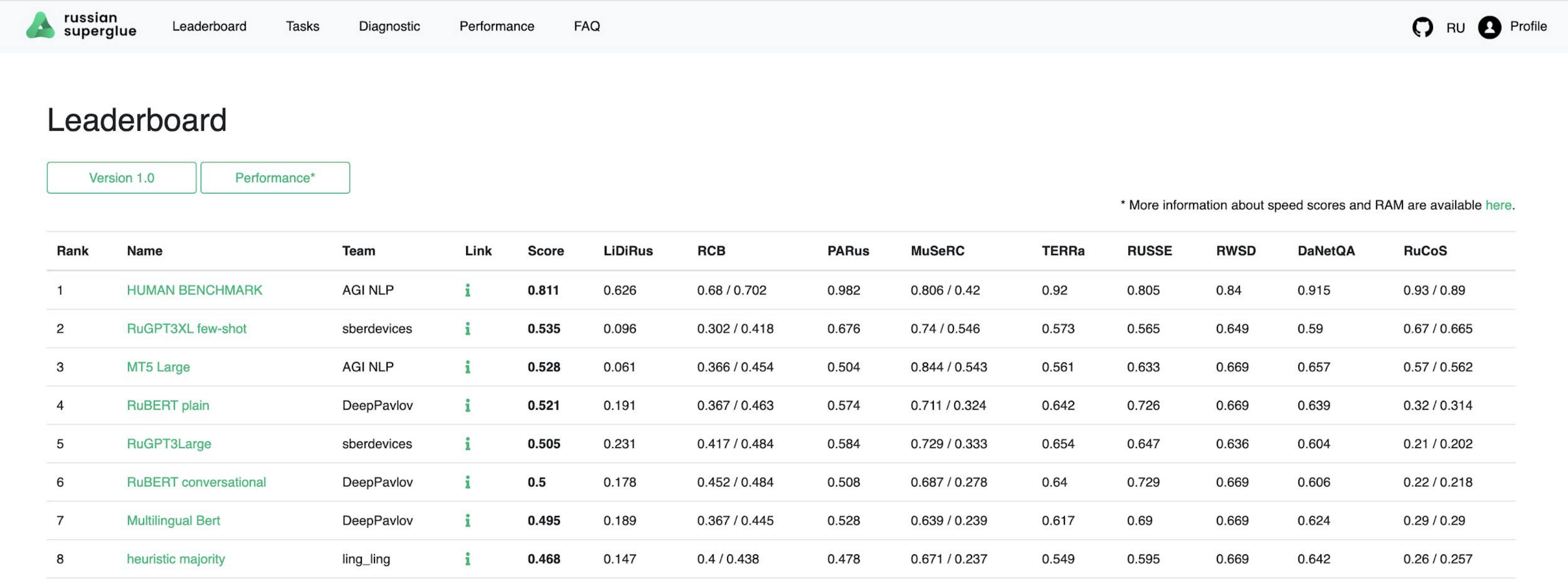
GLUE SuperGLUE



	Rank	Name	Model									
	1	ERNIE Team - Baidu	ERNIE	↗	90.9	74.4	97.8	93.9/91.8	93.0/92.6	75.2/90.9	91.9	91.4
	2	DeBERTa Team - Microsoft	DeBERTa / TuringNLRv4	↗	90.8	71.5	97.5	94.0/92.0	92.9/92.6	76.2/90.8	91.9	91.6
	3	HFL iFLYTEK	MacALBERT + DKM		90.7	74.8	97.0	94.5/92.6	92.8/92.6	74.7/90.6	91.3	91.1
+	4	Alibaba DAMO NLP	StructBERT + TAPT	↗	90.6	75.3	97.3	93.9/91.9	93.2/92.7	74.8/91.0	90.9	90.7
+	5	PING-AN Omni-Sinitic	ALBERT + DAAF + NAS		90.6	73.5	97.2	94.0/92.0	93.0/92.4	76.1/91.0	91.6	91.3
	6	T5 Team - Google	T5	↗	90.3	71.6	97.5	92.8/90.4	93.1/92.8	75.1/90.6	92.2	91.9
	7	Microsoft D365 AI & MSR AI & GATECH	MT-DNN-SMART	↗	89.9	69.5	97.5	93.7/91.6	92.9/92.5	73.9/90.2	91.0	90.8
+	8	Huawei Noah's Ark Lab	NEZHA-Large		89.8	71.7	97.3	93.3/91.0	92.4/91.9	75.2/90.7	91.5	91.3
+	9	Zihang Dai	Funnel-Transformer (Ensemble B10-10-10H1024)	↗	89.7	70.5	97.5	93.4/91.2	92.6/92.3	75.4/90.7	91.4	91.1

Evaluation. Benchmarks

- A benchmark of 8 sentence- or sentence-pair language understanding tasks
- A diagnostic dataset designed to evaluate and analyze model performance with respect to a wide range of linguistic phenomena found in natural language,
- A public leaderboard for tracking performance on the benchmark;



Leaderboard

Version 1.0 | Performance*

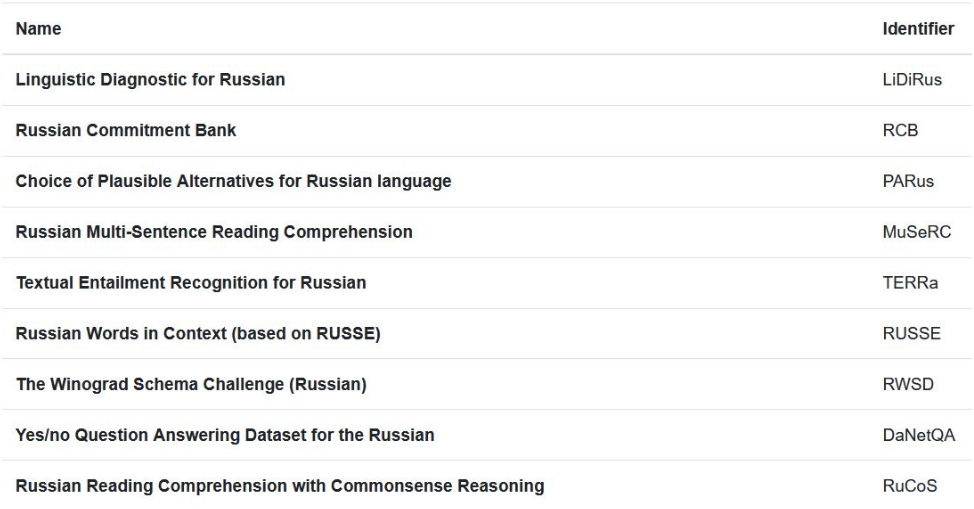
* More information about speed scores and RAM are available here.

Rank	Name	Team	Link	Score	LiDiRus	RCB	PARus	MuSeRC	TERRa	RUSSE	RWSD	DaNetQA	RuCoS
1	HUMAN BENCHMARK	AGI NLP	ℹ	**0.811**	0.626	0.68 / 0.702	0.982	0.806 / 0.42	0.92	0.805	0.84	0.915	0.93 / 0.89
2	RuGPT3XL few-shot	sberdevices	ℹ	**0.535**	0.096	0.302 / 0.418	0.676	0.74 / 0.546	0.573	0.565	0.649	0.59	0.67 / 0.665
3	MT5 Large	AGI NLP	ℹ	**0.528**	0.061	0.366 / 0.454	0.504	0.844 / 0.543	0.561	0.633	0.669	0.657	0.57 / 0.562
4	RuBERT plain	DeepPavlov	ℹ	**0.521**	0.191	0.367 / 0.463	0.574	0.711 / 0.324	0.642	0.726	0.669	0.639	0.32 / 0.314
5	RuGPT3Large	sberdevices	ℹ	**0.505**	0.231	0.417 / 0.484	0.584	0.729 / 0.333	0.654	0.647	0.636	0.604	0.21 / 0.202
6	RuBERT conversational	DeepPavlov	ℹ	**0.5**	0.178	0.452 / 0.484	0.508	0.687 / 0.278	0.64	0.729	0.669	0.606	0.22 / 0.218
7	Multilingual Bert	DeepPavlov	ℹ	**0.495**	0.189	0.367 / 0.445	0.528	0.639 / 0.239	0.617	0.69	0.669	0.624	0.29 / 0.29
8	heuristic majority	ling_ling	ℹ	**0.468**	0.147	0.4 / 0.438	0.478	0.671 / 0.237	0.549	0.595	0.669	0.642	0.26 / 0.257

Evaluation. Russian SuperGLUE

Six groups of tasks:

- Textual Entailment & NLI: *TERRa, RCB*
- Diagnostics: *LiDiRus*
- Common Sense: *RUSSe, PARus*
- World Knowledge: *DaNetQA*
- Machine Reading: *MuSeRC, RuCoS*
- Logic: *RWSD*



Name	Identifier
Linguistic Diagnostic for Russian	LiDiRus
Russian Commitment Bank	RCB
Choice of Plausible Alternatives for Russian language	PARus
Russian Multi-Sentence Reading Comprehension	MuSeRC
Textual Entailment Recognition for Russian	TERRa
Russian Words in Context (based on RUSSE)	RUSSE
The Winograd Schema Challenge (Russian)	RWSD
Yes/no Question Answering Dataset for the Russian	DaNetQA
Russian Reading Comprehension with Commonsense Reasoning	RuCoS

https://russiansuperglue.com/

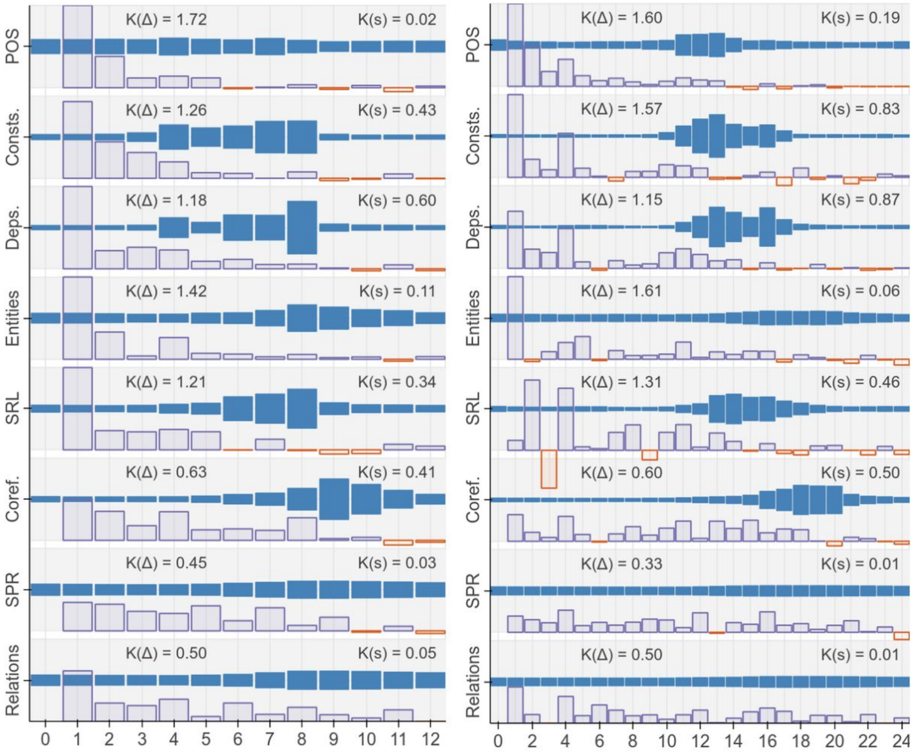
Evaluation. Probing

What do representations in the model learn? Do they learn to encode some linguistic features?

The most popular approach is to use probing classifiers (make probes, probing tasks, diagnostic classifiers).

<u>What should we do?</u>

- **feed data** to the network we explore;
- **get vector representations** of this data;
- **train a classifier to predict some linguistic labels** from these representations (but the model itself is frozen and is used only to produce representations);
- **use the classifier's accuracy** as a measure of how well representations encode labels.



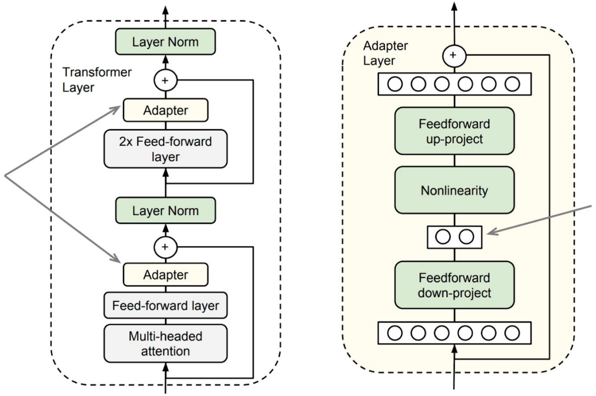
(a) BERT-base (b) BERT-large

Figure A.3: Layer-wise metrics on BERT-base (left) and BERT-large (right). Solid (blue) are mixing weights $s_\tau^{(\ell)}$; outlined (purple) are differential scores $\Delta_\tau^{(\ell)}$, normalized for each task. Horizontal axis is encoder layer.

Adapters

Fine-tuning Contrs:

For each downstream task you need to fine-tune a separate copy of your pretrained model => you end up with a lot of large models - this is highly inefficient



Only these are trained, everything else is fixed and is the same for all tasks

Small hidden size, i.e. an adaptor has only a few parameters (which is good!)

Adapters

Parameter-Efficient Transfer Learning for NLP proposed transfer with adapter modules.

In this setting, the parameters of the original model are fixed, and one has to train only a few trainable parameters per task: these new task-specific parameters are called **adaptors**.

With adapter modules, transfer becomes very efficient: the largest part, the pretrained model, is shared between all downstream tasks.

Adapter module = two-layer feed-forward network with a nonlinearity. What is important, the hidden dimensionality of this network is small, which means that the total number of parameters in the adapter is also small. This is what makes adaptors very efficient.

https://adapterhub.ml/

References

References

BERT

- paper: https://arxiv.org/pdf/1810.04805.pdf
- http://jalammar.github.io/illustrated-bert/
- See video: https://www.youtube.com/channel/UCoRX98PLOsaN8PtekB9kWrw/videos
- Как приготовить **RuBERT** https://arxiv.org/pdf/1905.07213.pdf

Gpts

- GPT-1: Improving Language Understanding by Generative Pre-Training
- GPT-2: Language Models are Unsupervised Multitask Learners
- GPT-3: Language Models are Few-Shot Learners
- http://jalammar.github.io/illustrated-gpt2/
- https://medium.com/walmartglobaltech/the-journey-of-open-ai-gpt-models-32d95b7b7fb2

T5

paper: https://arxiv.org/pdf/1910.10683.pdf

Read https://medium.com/analytics-vidhya/t5-a-detailed-explanation-a0ac9bc53e51

Work with models:

- Hugging face: https://huggingface.co/models
- Transformers library: https://github.com/huggingface/transformers
- Russian GPTS: https://github.com/sberbank-ai/ru-gpts/tree/master
- Tokenizers: https://huggingface.co/transformers/tokenizer_summary.html