# Transfer learning and pre-trained models

# Today

- Transfer learning
  - Contextualized representations
  - Pre-training + fine-tuning
  - Tokenization. Sub-word Segmentation
- Training objectives and architecture variations
  - BERT, GPTs, T5
- Evaluation (Benchmarks, Probing)
- Compression of pretrained models

# 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

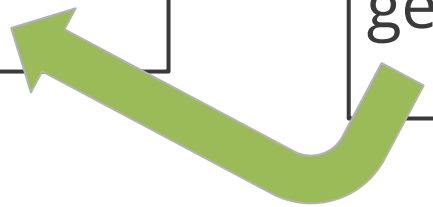
# Transfer learning. Motivation

**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



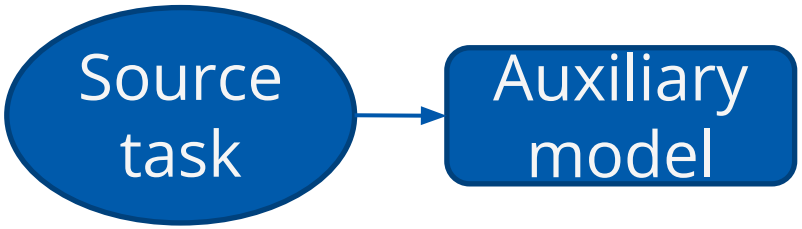
The general idea of transfer learning is

to "transfer" knowledge from one model to another.

# Transfer task

We have:

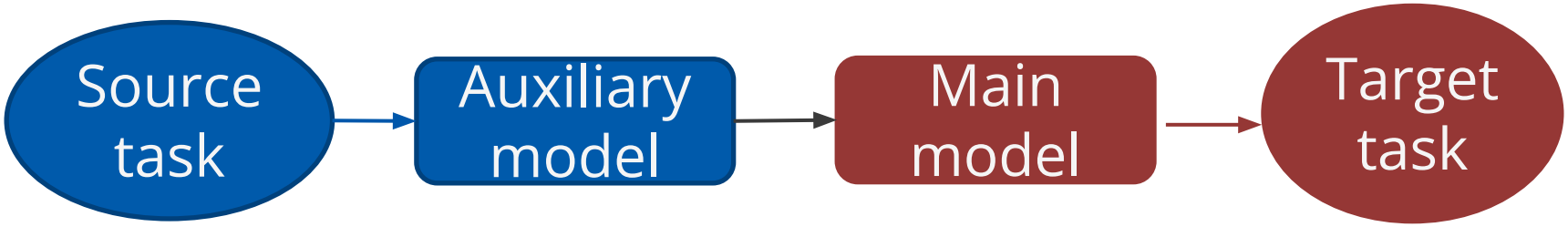
1) some source task
2) some auxiliary model we take knowledge from



# Transfer task

We have:

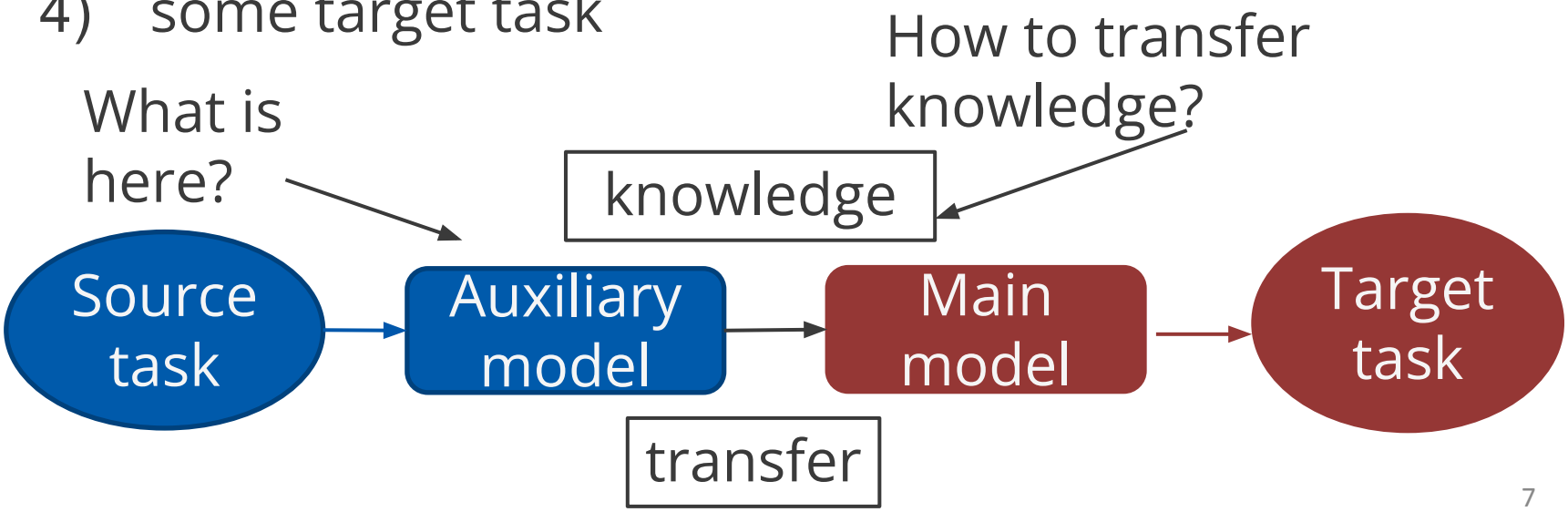
1) some source task
2) some auxiliary model we take knowledge from
3) main model we want to add knowledge to
4) some target task



# Transfer task

We have:

1) some source task
2) some auxiliary model we take knowledge from
3) main model we want to add knowledge to
4) some target task

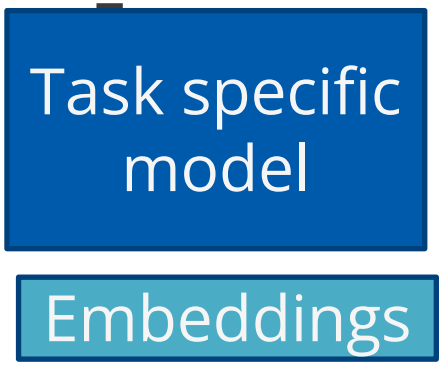


# How to transfer?

1) We can take pretrained word embeddings Word2vec, FastText, GloVE, etc.

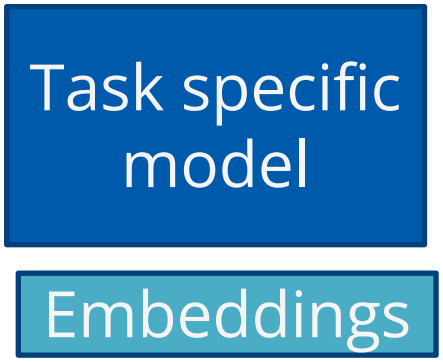
2) Through the embeddings, we "transfer" the knowledge of their training data to our task-specific model

**BEFOR**




Trained from scratch with the model

**AFTER**




Trained before, saw a lot of data, know a lot about language

# Pros and contrs

**1)**

+ unlabeled data

+ know relationships between words

+ can be specific to the task, can be not

- vector is the same in different contexts

**2)**

+ better than from scratch

- we do not change the model

- task specific model features stay

# How to transfer?

1) Initizialize with some pretrained model

2) Change the entire task-specific model to a single model which can be fine-tuned for all tasks

**BEFOR**


Pre-trained embeddings, task specific model

**AFTER**


Pre-trained model that saw a lot and can be trained further to any task

# Transfer learning

**The two ideas of good pretrained model:**

- what is encoded:

  word embeddings ⟹ context representation

- usage for downstream tasks:

  task-specific models ⟹ make model universal

# Words in contexts

Context representation of words

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 contexts)

# Word representation

*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!

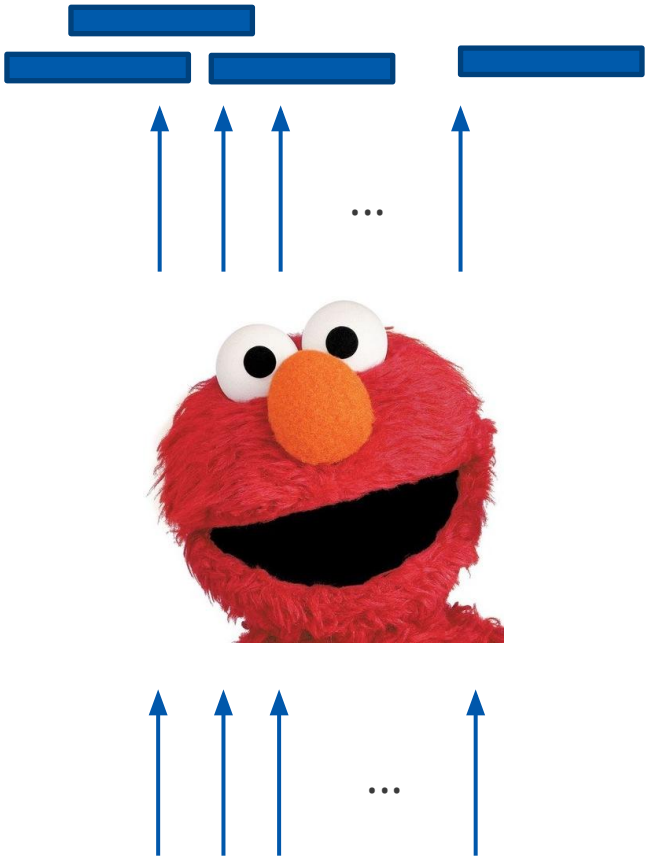
# ELMO

- Deep contextualized word representations (ELMO).
- Data: unlabelled data
- Task: language modeling

We can transfer knowledge on task-specific models (for example classification task).

# ELMO

- Input: tokens
- Output: word in context representation
- Sentence embedding: averaging vector representations of words

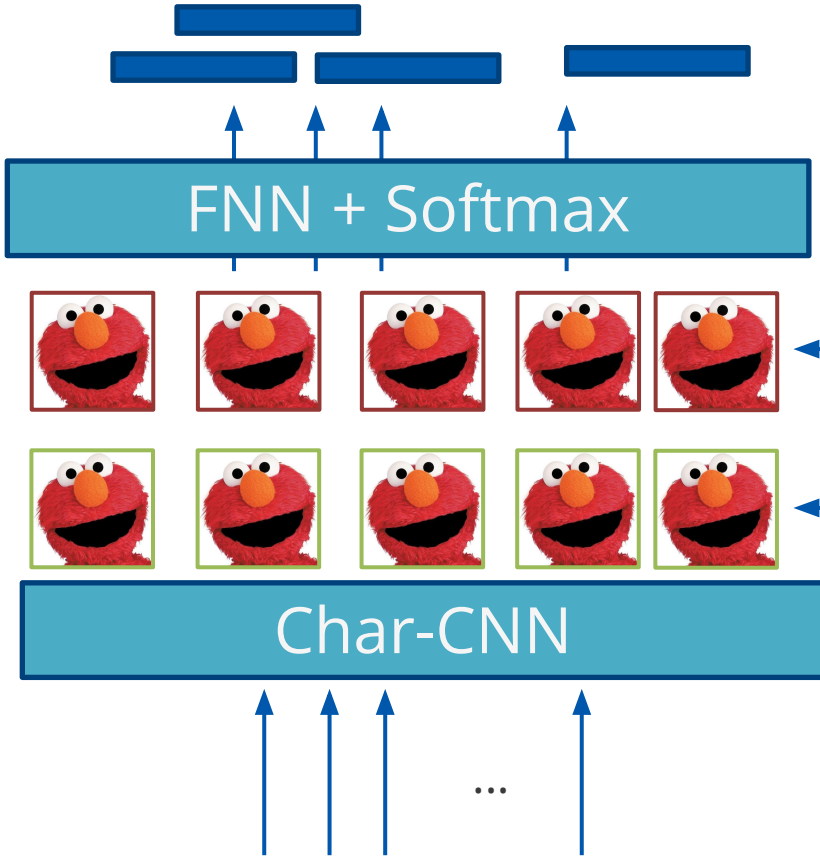



Some input tokens
ELMO

# ELMO

- Model Training: 2 LSTMs

Standard LM $P(word_i|word_{i-1})$




2-layer backward LSTM

2-layer forward LSTM

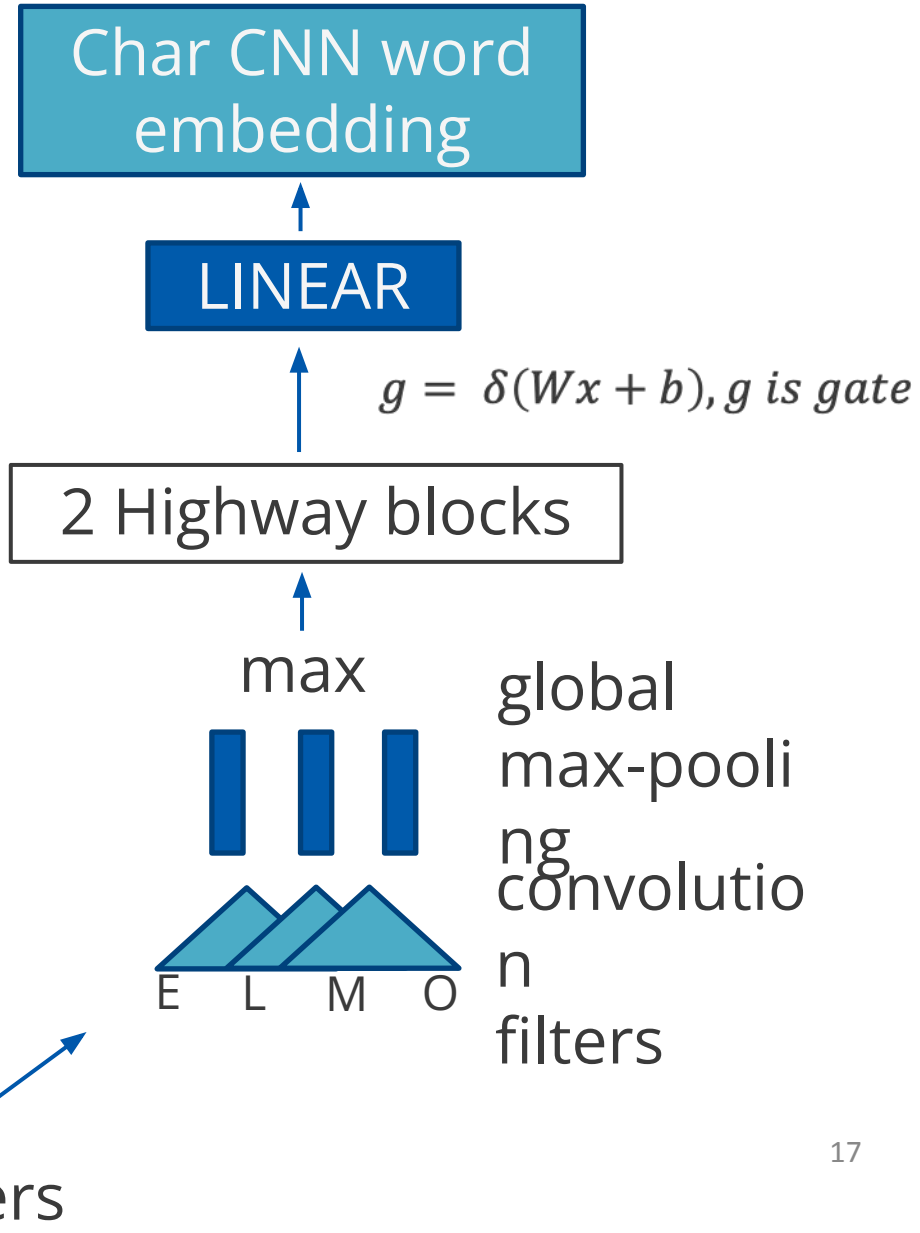
Some input tokens ELMO

# ELMO

Size of char embeddings vocabulary is **262 chars** because:

char ids 0-255 come from utf-8 encoding bytes

assign 256-300 to special chars (bos, eos. pad ...)

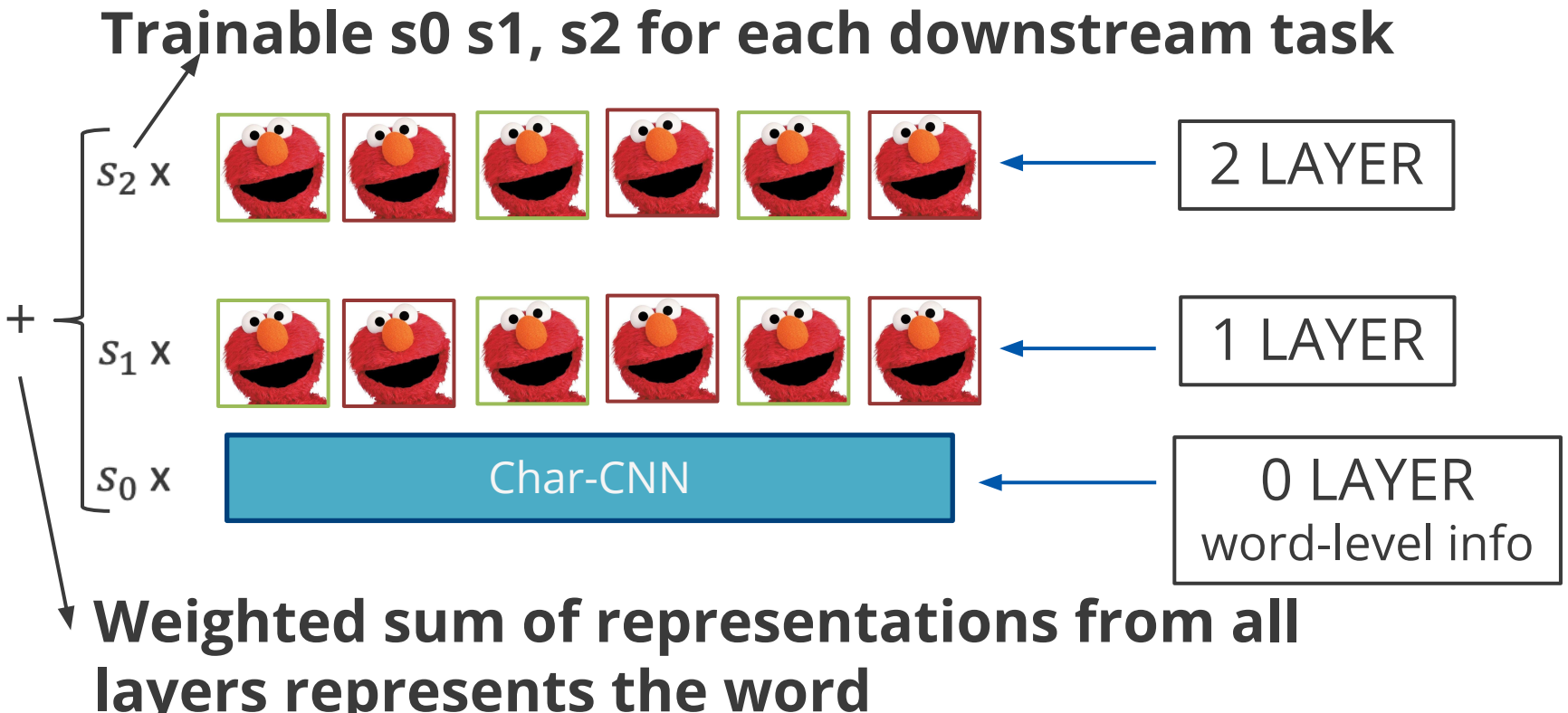


# ELMO

*layer 0* - output of the character-level CNN;

*layer 1* - concatenated representations from layer 1 of both forward and backward LMs;

*layer 2* - concatenated representations from layer 2 of both forward and backward LMs.



# ELMO. Summary

- **Model Training:**

Forward and Backward LSTM-LMs on top of a char-CNN.

- **Full word embedding:**

weighted sum of 2 word representations and char-cnn.

+ We can represent out-of-vocabulary (OOV) words
+ Words have meaning depending of the context
+ We can use different layers of the model
+ We can get sentence embedding

# Pre-training/fine-tuning

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

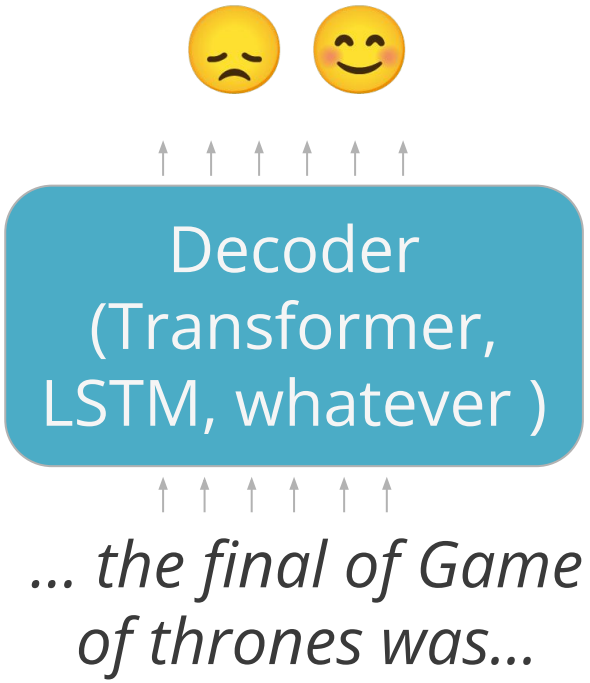
- **Pre-train** through language modeling
- Train a NN to perform on a large amount of text
- Save the network parameters

# Pre-training/fine-tuning

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

**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



# Sub-word segmentation

# OOV problems

*FastText copes with OOV problems via char ngrams*

*ELMO uses Char-CNN embeddings*

We want from pretrained models a tokenization procedure that:

- Solve OOV problems
- Doesn’t make our vocabulary fix
- Applied for new words

# Word structure and sub-word 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 |

# Word structure and sub-word models

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

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.

# BPE

Byte-pair encoding algo:

1. Training part = learn BPE rules, make merge table:
   - count pairs of symbols: how many times each pair occurs together in the training set;

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



("h" "u" "g", 10), ("p" "u" "g", 5),
("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "u" "g" "s", 5)

# BPE

Byte-pair encoding algo:

1. Training part = learn BPE rules, make merge table:
   - count pairs of symbols: how many times each pair occurs together in the training set;
   - find the most frequent pairs of symbols;

("h" "u" "g", 10), ("p" "u" "g", 5),
("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "u" "g" "s", 5)

("h" "ug", 10), ("p" "ug", 5), ("p" "u" "n",
12),
("b" "u" "n", 4), ("h" "ug" "s", 5)

# BPE

Byte-pair encoding algo:

1. Training part = learn BPE rules, make merge table:
   - count pairs of symbols: how many times each pair occurs together in the training set;
   - find the most frequent pairs of symbols;
   - merge this pair - add a merge to the merge table.

("h" "ug", 10), ("p" "ug", 5), ("p" "u" "n", 12),
("b" "u" "n", 4), ("h" "ug" "s", 5)

↓

("hug", 10), ("p@@ug", 5), ("p@@un", 12),
("b@@un", 4), ("hug@@s", 5)

# BPE

Byte-pair encoding algo:

1 Training part = learn BPE rules, make merge table:

2. Inference part = apply learned rules

- segment text in characters
- among all possible merges at this step, find the highest* merge in the table;
- apply this merge

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

| | | |
|---|---|---|
| **New words:** | bug | ["b@@ug"] |
| | mug | [“<unk>ug"] |

# Types of tokenizers

BPE

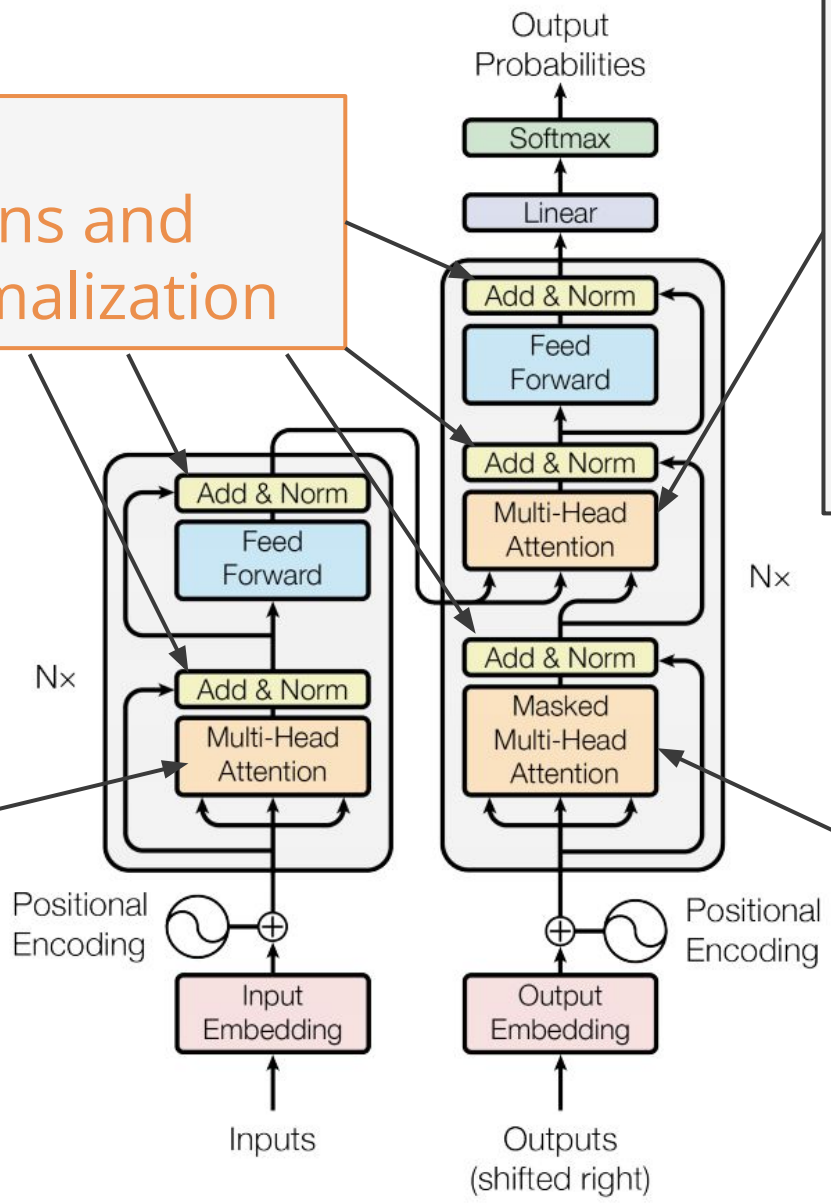
- BBPE (byte-level BPE) - all unicode characters are considered as base characters. Force the base vocabulary to be of size 256 while ensuring that every base character is included in the vocabulary.
- WordPiece. It does not choose the most frequent symbol pair, but the one that maximizes the likelihood of the training data once added to the vocabulary
- SentencePiece treats the input as a raw input stream, thus including the space in the set of characters to use. After itunicodes characters are grouped together using either a unigram language model or BPE.

# Training objectives and architectures

# Transformer

Residual connections and layer normalization

Decoder-Encoder attention: Q – from decoder states, K and V from encoder




Encoder self-attention: Q,K,V are computed from encoder states

Decoder (masked) self-attention: Q,K,V are computed from decoder states

# Architecture variations

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

1. Encoder part

   (Gets bidirectional context – can condition on future)

Encoders

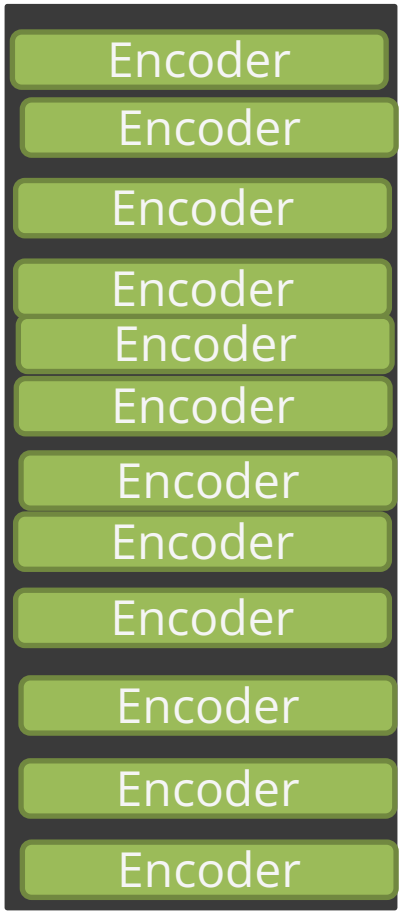
# BERT

Model architecture: transformer's encoder

No task specific models

Training objectives are different:

- NSP
- MLM

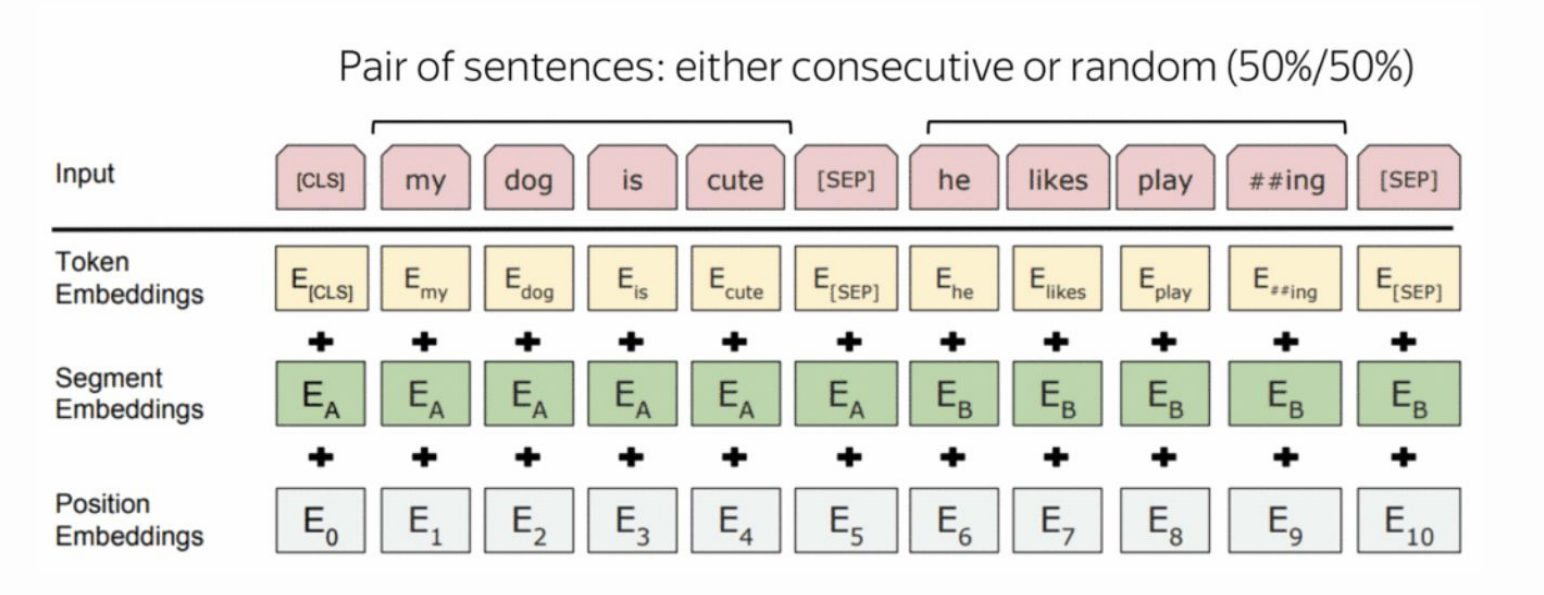



Bert-base (12 encoders)

# BERT. Next Sentence Prediction (NSP) Objective.

BERT was trained to predict whether one sentence follows the other or is randomly sampled.

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






arxiv.org

# BERT. 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
- 50% - a random pair of sentences.

# BERT. Masked LM objective

Idea of Masked LM:

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

Only add loss terms from words that are "masked out."
If $x\sim$ is the masked version of $x$, we're learning $P_\theta(x|x\sim)$

# BERT. Masked LM objective

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 time (but still predict it!)
3. Predict selected tokens

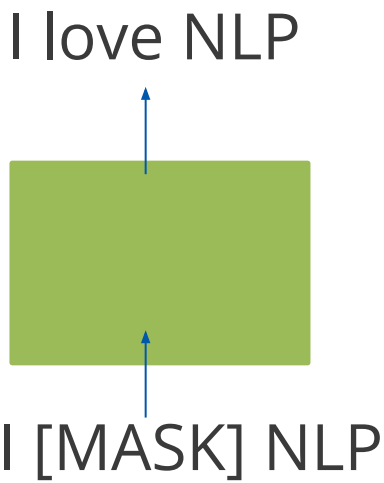
# BERT. MLM

**Masked LM:**

the goal is to predict some tokens in a sentence/text based on unmasked parts of this text.

Target: current token (Masked)

Prediction: $P(*|I\ [MASK]\ NLP)$




Sees the whole text but some tokens are masked

# BERT. Summary

**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)

# BERT. Summary

**Pretraining:**

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

**Tokenizer:**

- WordPiece

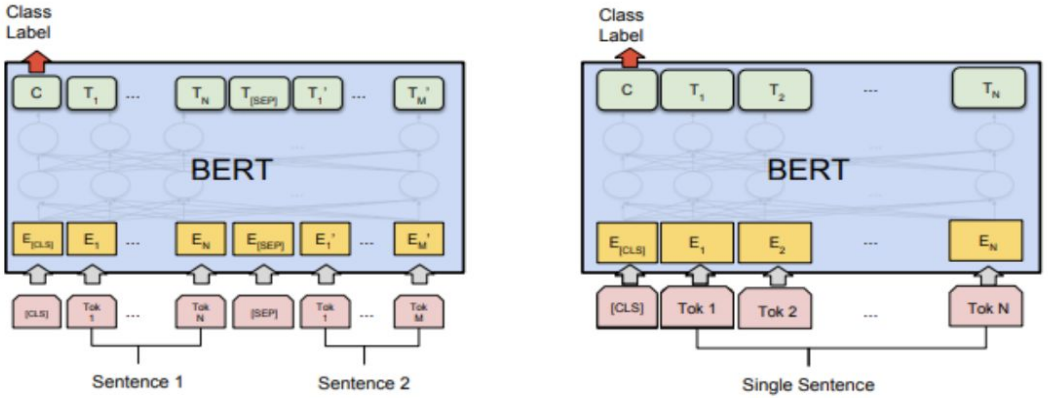
# BERT. Usage

Task-specific:

- Single sentence classification
- Sentence pair classification
- Single sentence tagging
- Question Answering

BERT for feature extraction:
use the pre-trained BERT to create contextualized word embeddings

# BERT. Usage




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

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

arxiv.org

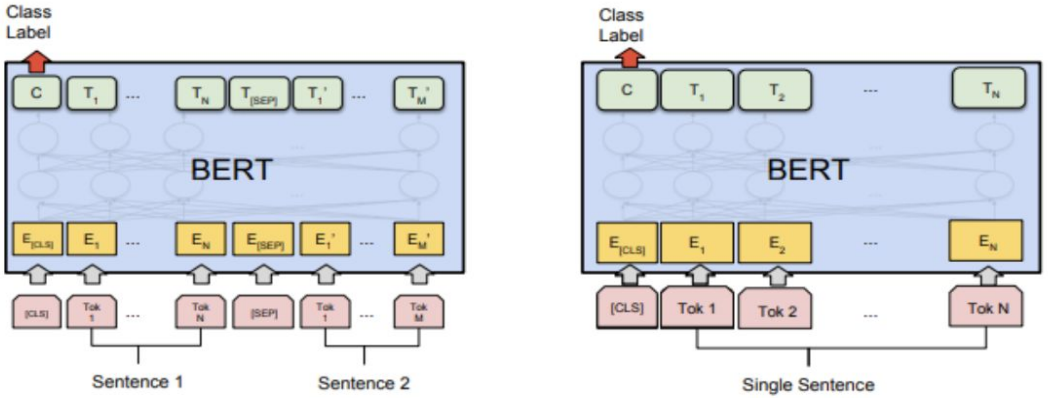
## Sentence pair classification

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

## Single sentence classification

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

# BERT. Usage




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

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

arxiv.org

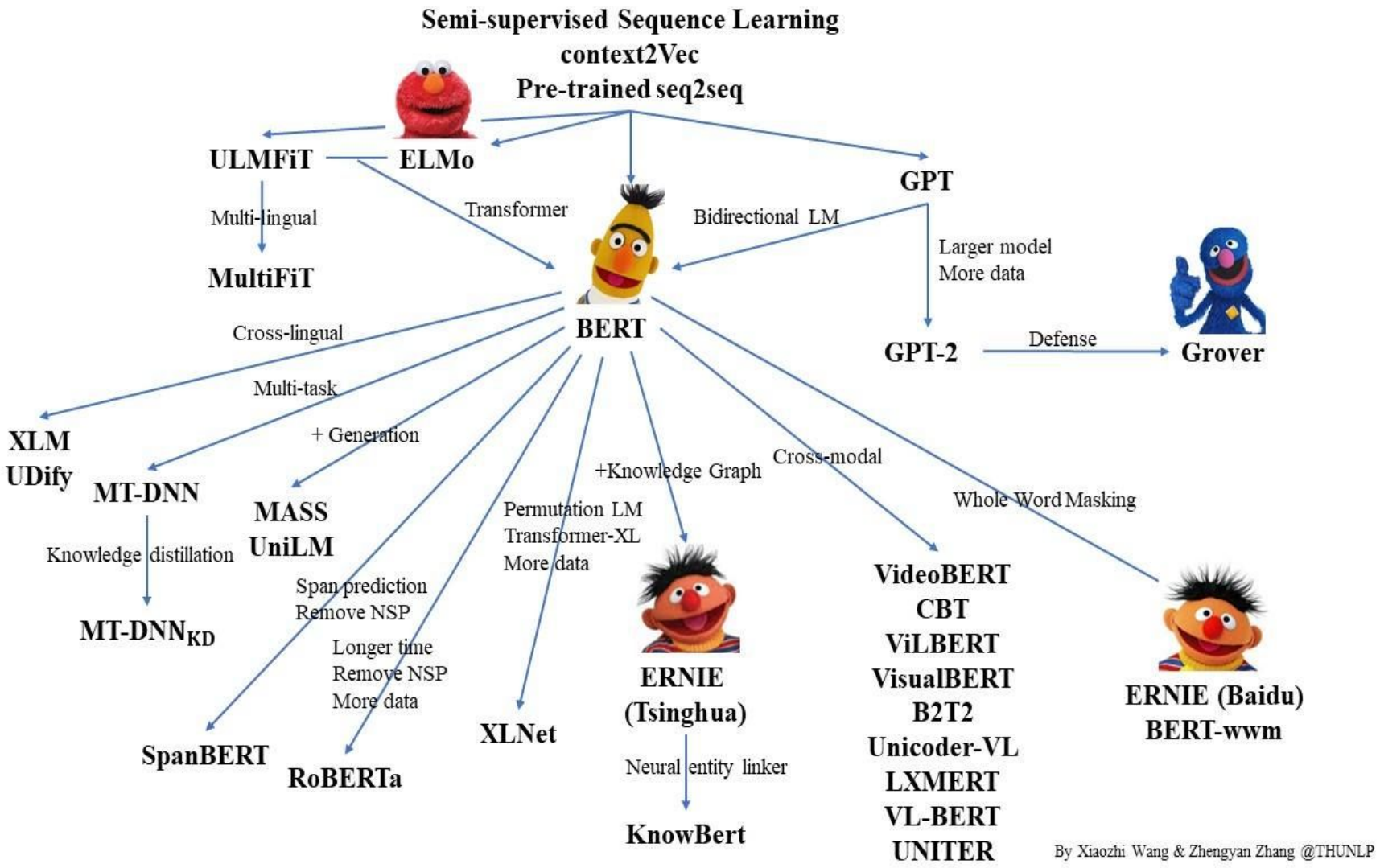
## Question Answering

Feed question and paragraph into BERT. for each token in the passage use final BERT representations to predict whether this token is the start or the end of the correct segment

## Single sentence tagging

No second sentence, predict tags for each token.

# BERTology. Sesame street



# 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.

# BERTology

**Albert**:

- Google research
- It is not reasonable to further improve language models by making them larger.
- Two parameter-reduction techniques:

# BERTology

**Albert**:

- 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.

# GPTs

# Architecture variations

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

- Encoder part (Gets bidirectional context – can condition on future)
- Decoder part. LMs (Nice to generate from; can't condition on future words

Decoders

# GPT

Model architecture:

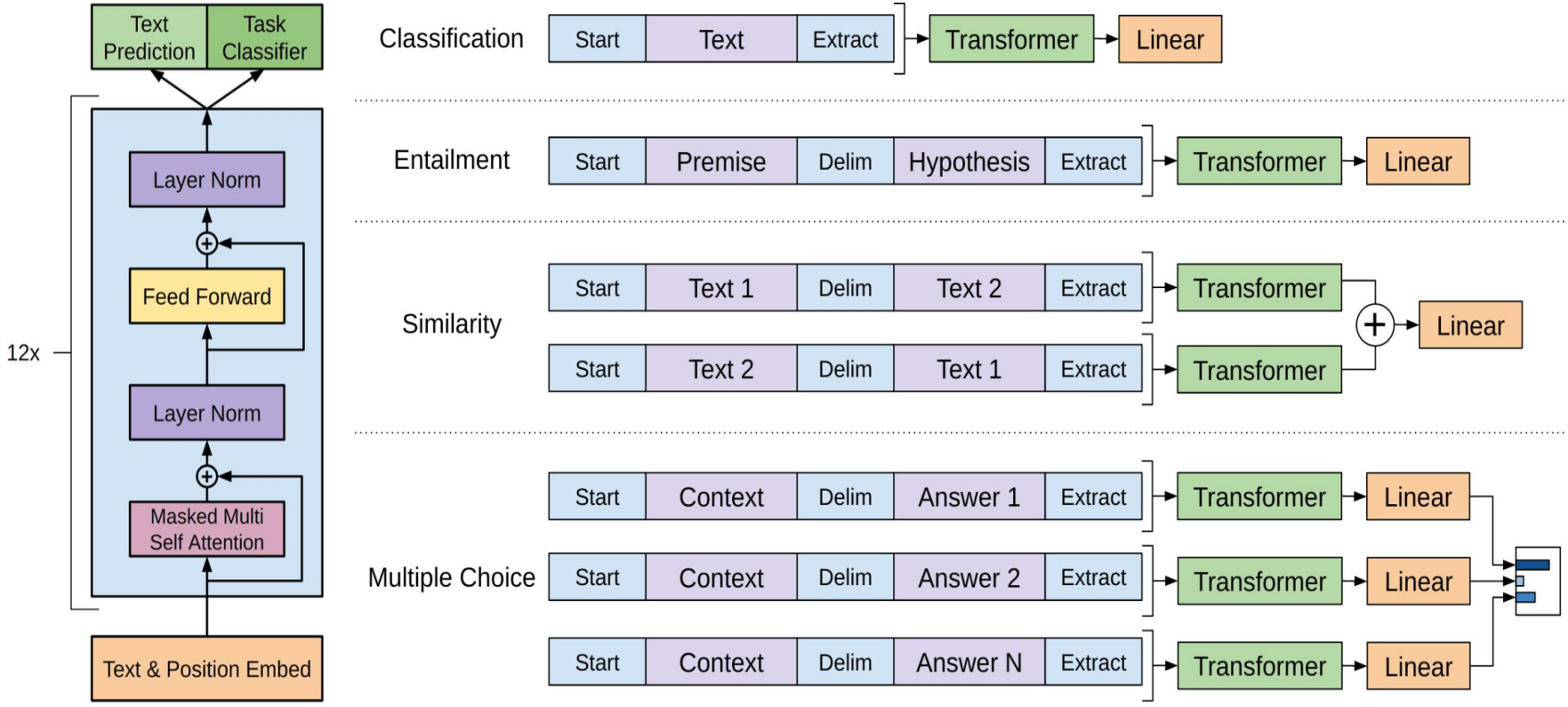
transformer's decoder (12 decoders)

Left-to-right model

Autoregressive

# GPT

- Unsupervised Language Modelling $L_{lm} = -\sum_{t=1}^{n} \log(p(y_t|y_{<t}))$
- Supervised Fine-Tuning $L = L_{lm} + \lambda \cdot L_{task}$
- 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

# GPTs

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**: BBPE 50,257

# GPTs

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**: BBPE

# GPT

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

  no examples are provided at all and the model understands the task based on the given instruction

  Few-shot learning

  classify new data when you have only a few training samples with supervised information

# GPT

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

# Encoder-Decoders

# Architecture variations

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

- Encoder part (Gets bidirectional context – can condition on future)
- Decoder part. LMs (Nice to generate from; can't condition on future words
- Encoder-decoder architecture(best from both parts)

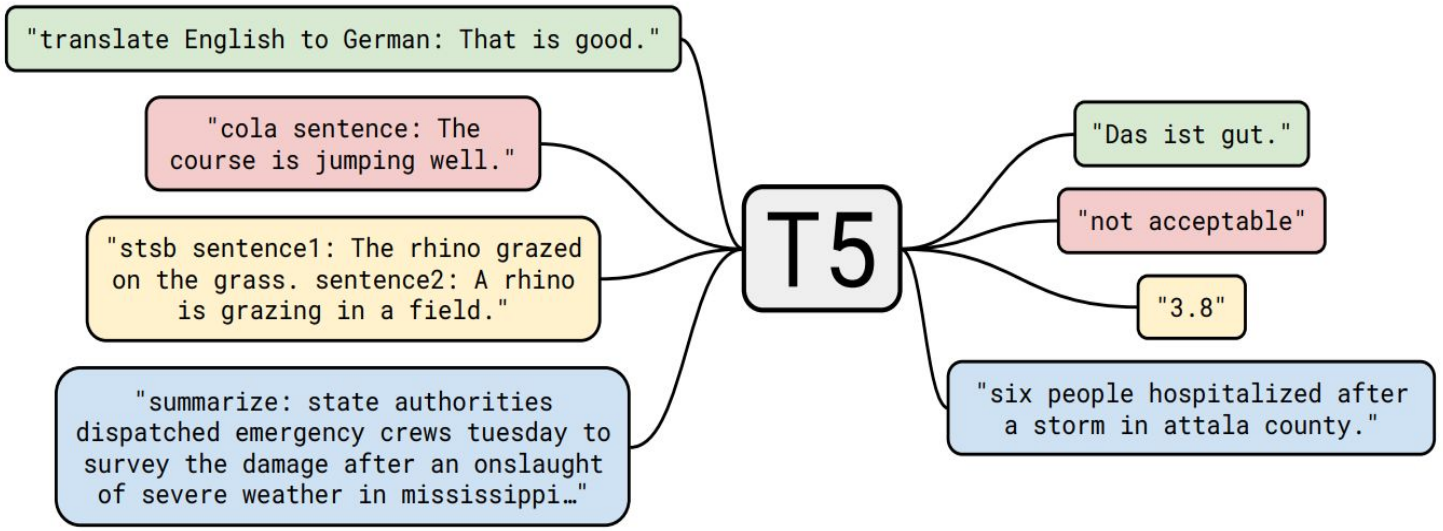
Mbart, T5, etc.

# T5

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)



# T5

T5 trains with Denoising or Text Infilling objective.

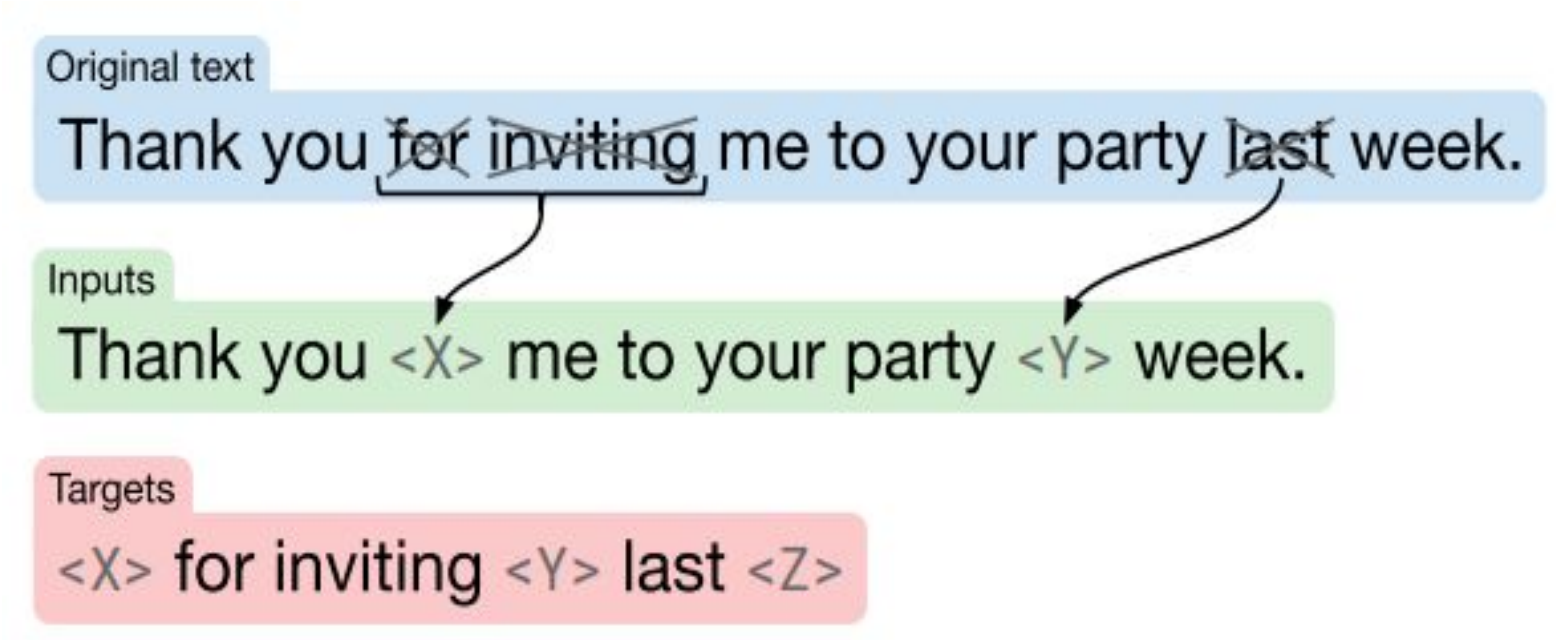
The original text is transformed into **Input** and **Output** pairs:

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

Final objective is trained to predict basically sentinel tokens to delineate the dropped out text.

# T5

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.




T5's mask language modeling (Raffel et al., 2019)

# 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

**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

# Summary. Pretrained models

| Method | Architecture | Objective |
|---|---|---|
| ELMO | LSTM | LM |
| GPTs | Decoder | LM |
| BERT | Encoder | MLM, NPS |
| RoBERTa | Encoder | MLM |
| ALBERT | Encoder | MLM, SOP |
| T5 | Encoder-Decoder | Text Infilling |

# Training objectives

| Objective | Inputs | Targets |
|---|---|---|
| LM | [START] | I am happy to join with you today |
| MLM | I am [MASK] to join with you [MASK] | happy today |
| NSP | Sent1 [SEP] Next Sent or Sent1 [SEP] Random Sent | Next Sent/Random Sent |
| SOP | Sent1 [SEP] Sent2 or Sent2 [SEP] Sent1 | in order/reversed |
| Discriminator (o/r) | I am thrilled to study with you today | o o r o r o o o |
| PLM | happy join with | today am I to you |
| seq2seq LM | I am happy to | join with you today |
| Span Mask | I am [MASK] [MASK] [MASK] with you today | happy to join |
| Text Infilling | I am [MASK] with you today | happy to join |
| Sent Shuffling | today you am I join with happy to | I am happy to join with you today |
| TLM | How [MASK] you [SEP] [MASK] vas-tu | are Comment |

# Evaluation

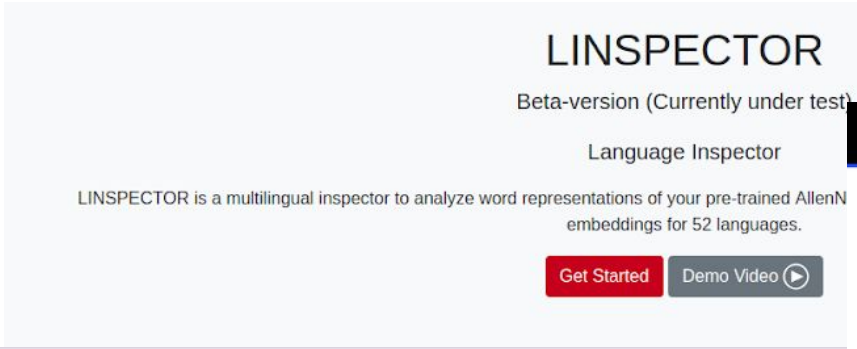
# Evaluation

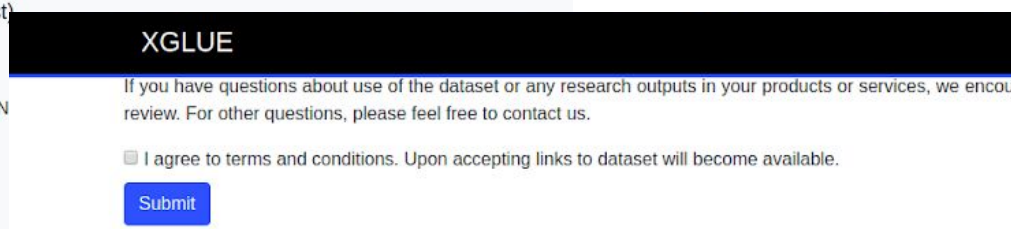
- LMs went through the advanced stages of natural language modelling.
- Universal transformers show ability to extract complicated relationships from texts.
- There are a lot of the pretrained models…

*How to test which model is better?*

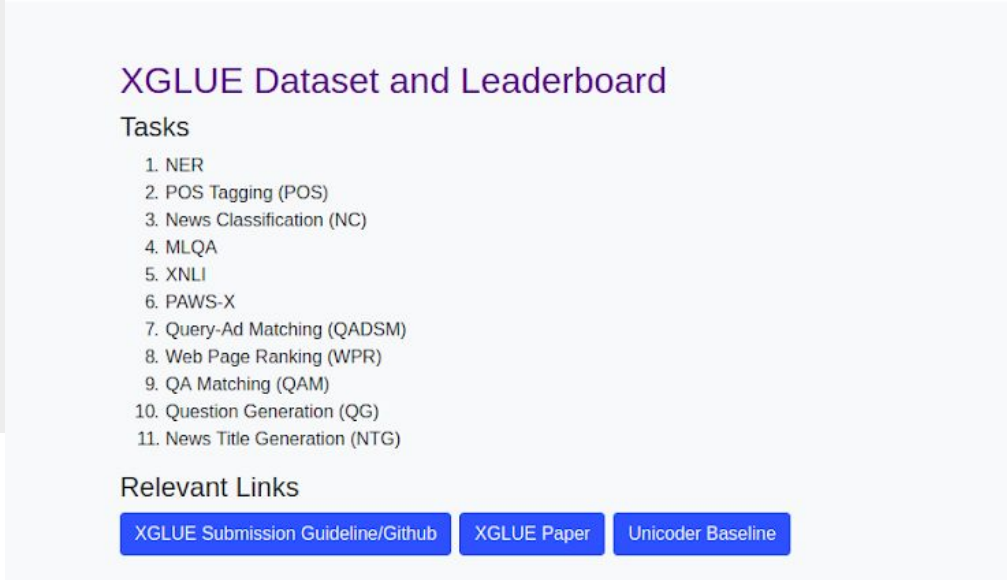
- Perplexity
- Benchmark approach
- Probing

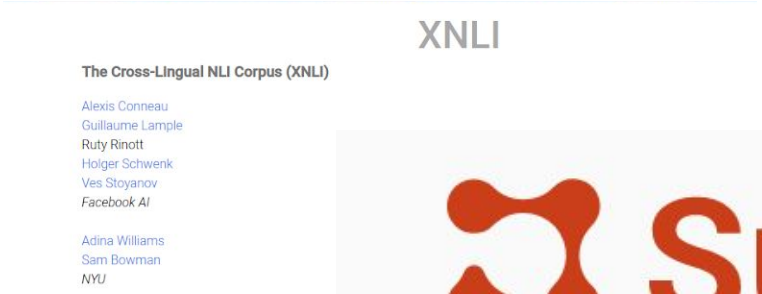
# Evaluation. Benchmark



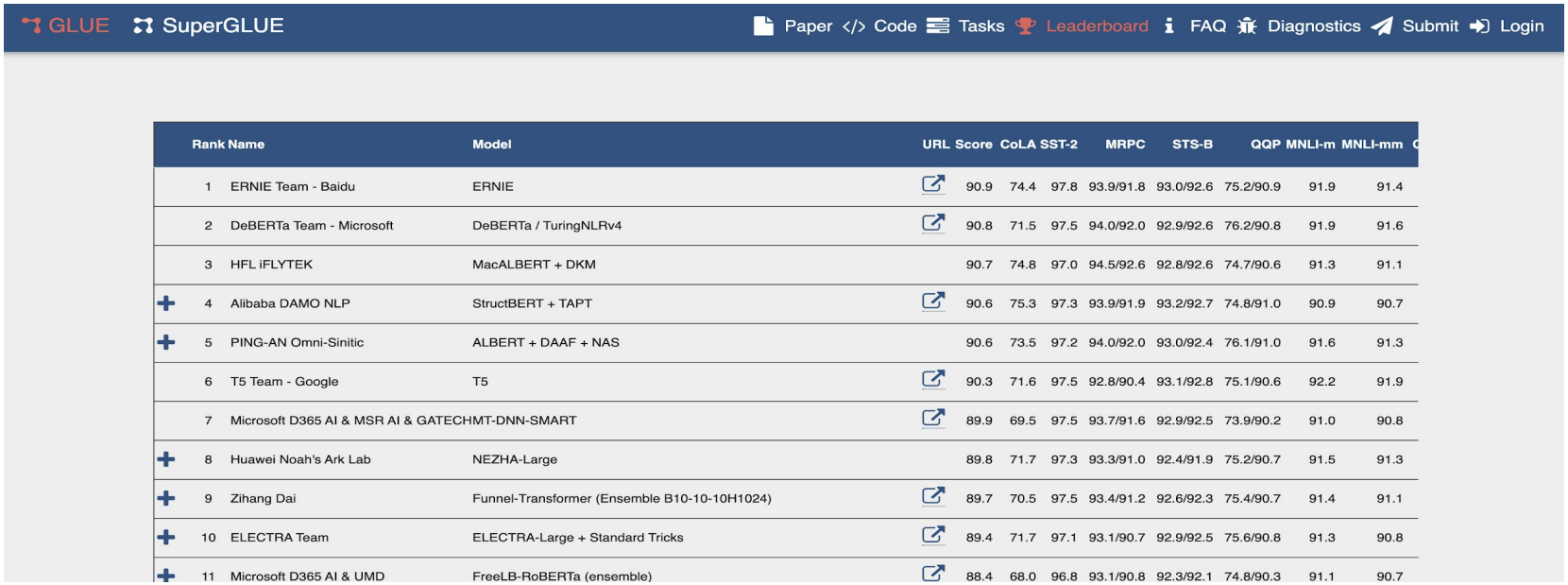








# Evaluation. Benchmark



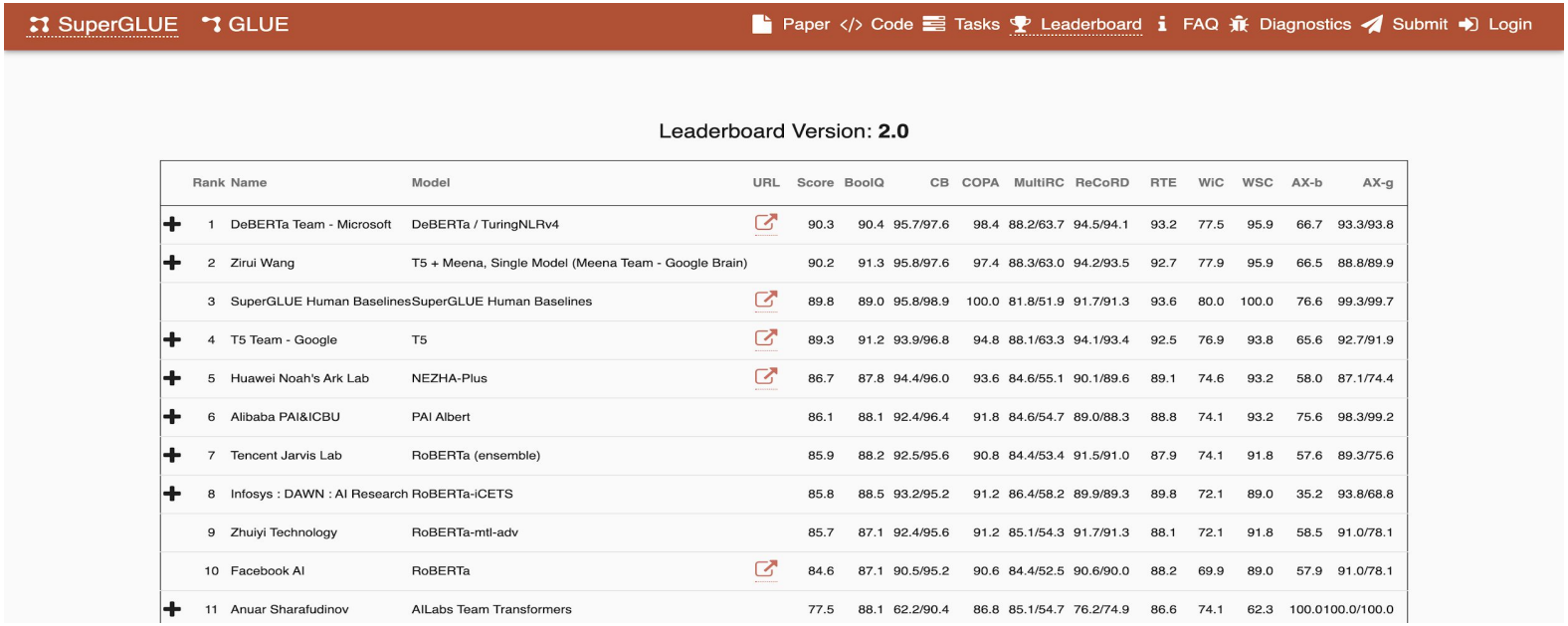

| Rank | Name | Model | URL | Score | CoLA | SST-2 | MRPC | STS-B | QQP | MNLI-m | MNLI-mm |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ERNIE Team - Baidu | ERNIE | | 90.9 | 74.4 | 97.8 | 93.9/91.8 | 93.0/92.6 | 75.2/90.9 | 91.9 | 91.4 |
| 2 | DeBERTa Team - Microsoft | DeBERTa / TuringNLRv4 | | 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 |
| 10 | ELECTRA Team | ELECTRA-Large + Standard Tricks | | 89.4 | 71.7 | 97.1 | 93.1/90.7 | 92.9/92.5 | 75.6/90.8 | 91.3 | 90.8 |
| 11 | Microsoft D365 AI & UMD | FreeLB-RoBERTa (ensemble) | | 88.4 | 68.0 | 96.8 | 93.1/90.8 | 92.3/92.1 | 74.8/90.3 | 91.1 | 90.7 |

GLUE - 2019

(Human baseline - 15 place)

Down-stream tasks: sentiment, COLA, paraphrasing, NLIs, Winograd, etc.

# Evaluation. Benchmark




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 |

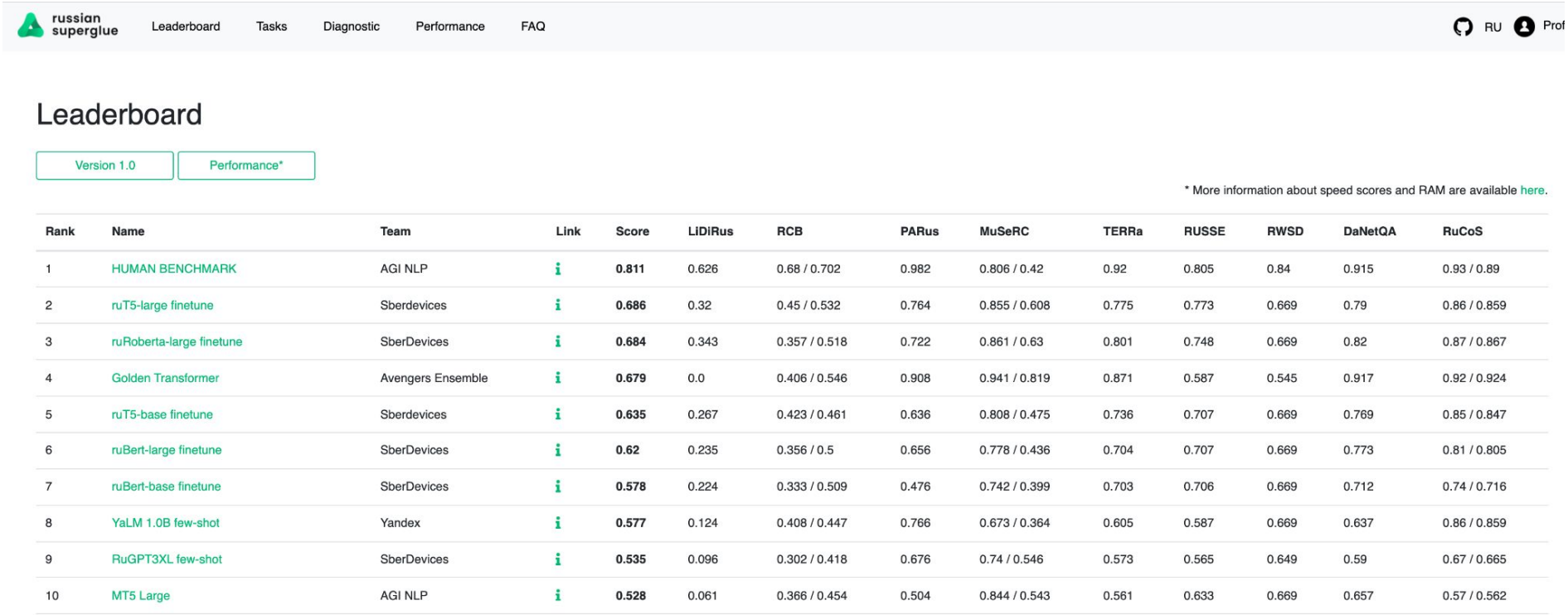
SuperGLUE – 2020

(Human baseline is already beaten)

Still the most usable benchmark for NLU.

**Groups of tasks:** Textual Entailment, Common Sense, World Knowledge, Machine Reading, Logic

# Evaluation. 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 | ruT5-large finetune | Sberdevices | i | **0.686** | 0.32 | 0.45 / 0.532 | 0.764 | 0.855 / 0.608 | 0.775 | 0.773 | 0.669 | 0.79 | 0.86 / 0.859 |
| 3 | ruRoberta-large finetune | SberDevices | i | **0.684** | 0.343 | 0.357 / 0.518 | 0.722 | 0.861 / 0.63 | 0.801 | 0.748 | 0.669 | 0.82 | 0.87 / 0.867 |
| 4 | Golden Transformer | Avengers Ensemble | i | **0.679** | 0.0 | 0.406 / 0.546 | 0.908 | 0.941 / 0.819 | 0.871 | 0.587 | 0.545 | 0.917 | 0.92 / 0.924 |
| 5 | ruT5-base finetune | Sberdevices | i | **0.635** | 0.267 | 0.423 / 0.461 | 0.636 | 0.808 / 0.475 | 0.736 | 0.707 | 0.669 | 0.769 | 0.85 / 0.847 |
| 6 | ruBert-large finetune | SberDevices | i | **0.62** | 0.235 | 0.356 / 0.5 | 0.656 | 0.778 / 0.436 | 0.704 | 0.707 | 0.669 | 0.773 | 0.81 / 0.805 |
| 7 | ruBert-base finetune | SberDevices | i | **0.578** | 0.224 | 0.333 / 0.509 | 0.476 | 0.742 / 0.399 | 0.703 | 0.706 | 0.669 | 0.712 | 0.74 / 0.716 |
| 8 | YaLM 1.0B few-shot | Yandex | i | **0.577** | 0.124 | 0.408 / 0.447 | 0.766 | 0.673 / 0.364 | 0.605 | 0.587 | 0.669 | 0.637 | 0.86 / 0.859 |
| 9 | 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 |
| 10 | 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 |

Russian SuperGLUE

https://russiansuperglue.com/

# Evaluation. Benchmark

1. Textual Entailment & NLI: *TERRa, RCB*
2. Diagnostics: *LiDiRus*
3. Common Sense: *RUSSe, PARus*
4. World  Knowledge: *DaNetQA*
5. Machine Reading: *MuSeRC, RuCoS*
6. 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 |

# Evaluation. Benchmark

**TERRA**

**Analogue in SuperGLUE**: *RTE* dataset.
**Task group**: Natural language inference
**Dataset size:**
2616 train / 307 val / 3198 test examples
**Data Source:** Data extracted from *Taiga web-corpus*
**Task:** given two text fragments (premise and hypothesis), say whether the meaning of one text is entailed from the other

**Example**

**Premise:** Автор поста написал в комментарии, что прорвалась канализация.

**Hypothesis:** Автор поста написал про канализацию.

**Label:** Entailment

# Evaluation. Benchmark

**TERRa**

**Analogue in SuperGLUE**: *RTE* dataset.
**Task group**: Natural language inference
**Dataset size:**
2616 train / 307 val / 3198 test examples
**Data Source:** Data extracted from *Taiga web-corpus*
**Task:** given two text fragments (premise and hypothesis), say whether the meaning of one text is entailed from the other

**Example**

**Premise:** Автор поста написал в комментарии, что прорвалась канализация.

**Hypothesis:** Автор поста написал про канализацию.

**Label:** Entailment

# Evaluation. Benchmark

**Russian Commitment bank(RCB)**

**Analogue in SuperGLUE**: *CB* dataset.
**Task group**: Natural language inference
**Dataset size:** 438/220/348
**Data Source:** Dataset corresponds to CommonBank dataset. Similarly to the design of TERRa dataset, we filtered out Taiga with a number of rules and manually post processed the extracted passages.
**Task:** The Russian Commitment Bank is a corpus of naturally occurring discourses whose final sentence contains an embedded clause predicate under an entailment canceling operator (question, modal, negation, antecedent of conditional)

**Example**

**Text:** Сумма ущерба составила одну тысячу рублей. Уточняется, что на место происшествия выехала следственная группа, которая установила личность злоумышленника. Им оказался местный житель, ранее судимый за подобное правонарушение.

**Hypothesis:** Ранее местный житель совершал подобное правонарушение.

**Label:** Entailment

# Evaluation. Benchmark

**RUSSE**

**Analogue in SuperGLUE**: WiC
**Task group**: Common Sense
**Dataset size:** 19845 train, 8508 dev, 12151 test examples
**Task:** Given two sentences, each containing an anchor word, define, whether the word is used in the same sense, or not.
**Data source:** Re-use of RUSSE dataset [Panchenko et al., 2018], in which ambiguous words are annotated with sense labels.

**Example**

**Context 1:** Бурые ковровые дорожки заглушали шаги.

**Context 2:** Приятели решили выпить на дорожку в местном баре.

**Sense match:** False

*1.Panchenko, A., Lopukhina, A., Ustalov, D., Lopukhin, K., Arefyev, N., Leontyev, A., Loukachevitch, N.: RUSSE'2018: A Shared Task on Word Sense Induction for the Russian Language. (2018)*

# Evaluation. Benchmark

**PARUS**

**Analogue in SuperGLUE**: *COPA*

**Task group**: Common Sense

**Task:** Choose the best of plausible alternatives for the question based on the text.

**Data source:** PARus is constructed as a translation of COPA dataset from SuperGLUE and edited by professional editors. The data split from COPA is retained.

**Dataset size:** 500/100/400 1000 sentences

**Example**

**Premise:** Чиновника обвинили в растрате

**Question:** *Что было дальше?*

**Alternative 1:** *Его сняли с должности*
**Alternative 2:** *Он начал кампанию по переизбранию*
**Correct Alternative:** 1

# Evaluation. Benchmark

**DaNetQA**

**Analogue in SuperGLUE**: BoolQ
**Task group**: Machine Reading
**Dataset size:** 800 train, 200 dev, 200 test examples; 562 (~59%) unique questions
**Task:** Given a passage, answer a yes/no question to it.
**Data source**

1) Crowdsourced questions are used as queries to Wikipedia
2) Wikipedia pages are retrieved via Google API
3) Passages are retrieved by Deep Pavlov SQuAD models
4) Crowd workers answer the questions based on the passages

**Example**

**Passage:** В период с 1969 по 1972 год по программе «Аполлон» было выполнено 6 полётов с посадкой на Луне.
**Question:** *Был ли человек на луне?*
**Answer:** Yes

# Evaluation. Benchmark

**MULTIRC**

**Analogue in SuperGLUE**: MultiRC
**Task group**: Machine reading
**Task:** Reading comprehension challenge, questions can be answered only based on multiple sentences from the paragraph.
**Dataset size:** 500/100/322
**Data source**
+800 paragraphs ~6k questions
5 different domains collected from open sources:

1) elementary school texts
2) news
3) fiction stories
4) fairy tales
5) brief annotations of TV series and books

**Example**

**Paragraph:** (1) Мужская сборная команда Норвегии по биатлону в рамках этапа Кубка мира в немецком Оберхофе выиграла эстафетную гонку. (2) Вторыми стали французы, а бронзу получила немецкая команда. (3) Российские биатлонисты не смогли побороться даже за четвертое место, отстав от норвежцев более чем на две минуты. (4) Это худший результат сборной России в текущем сезоне. (5) Четвёртыми в Оберхофе стали австрийцы. (6) В составе сборной Норвегии на четвёртый этап вышел легендарный Уле-Эйнар Бьорндален. (7) Впрочем, Норвегия с самого начала гонки была в числе лидеров, успешно проведя все четыре этапа. (8) За сборную России в Оберхофе выступали Иван Черезов, Антон Шипулин, Евгений Устюгов и Максим Чудов. (9) Гонка не задалась уже с самого начала: если на стрельбе из положения лежа Черезов был точен, то из положения стоя он допустил несколько промахов, в резуль- тате чего ему пришлось бежать один дополнительный круг. (10) После этого отставание российской команды от соперников только увеличива- лось. (11) Напомним, что днем ранее российские биатлонистки выиграли свою эстафету. (12) В составе сборной России выступали Анна Богалий- Титовец, Анна Булыгина, Ольга Медведцева и Светлана Слепцова. (13) Они опередили своих основных соперниц - немок - всего на 0,3 секунды.

**Question:** На сколько секунд женская команда опередила своих соперниц?

**Candidate answers:** Всего на 0,3 секунды. (*T*), На 0,3 секунды. (*T*), *На секунду.* (F), На секунды. (*F*)

# Evaluation. Benchmark

**RUCOS**

**Analogue in SuperGLUE**:

ReCoRD

**Task group**: Machine Reading

**Dataset size**

72193 train / 4370 val / 4147 test

**Data source:**

Lenta & Deutsche Welle

**Task:** Find the correct entity in the paragraph that best fits the placeholder in the query.

**Example**

**Passage:** Мать двух мальчиков, брошенных отцом в московском аэропорту *Шереметьево*, забрала их. Об этом сообщили *ТАСС* в пресс-службе министерства образования и науки *Хабаровского края*. Сейчас младший ребенок посещает детский сад, а старший ходит в школу. В учебных заведе- ниях с ними по необходимости работают штатные психологи. Также министерство социальной защиты населения рассматривает вопрос о бесплатном оздоровлении детей в летнее время. Через несколько дней после того, как *Виктор Гаврилов* бросил своих детей в аэропорту, он явился с повинной к следователям в городе *Батайске* *Ростовской области*.

**Query:** 26 января <placeholder> бросил сыновей в возрасте пяти и семи лет в Шереметьево.

**Correct Entities:** Виктор Гаврилов

# Evaluation. Benchmark

**Analogue in SuperGLUE**: Winograd Schema Challenge
**Task group**: Logic and reasoning
**Dataset size:** 606 train/204 val/154 test pairs of sentences
**Data Source:** Editing and translation of the Winograd Schema Challenge
**Task:** Each sentence has two objects for coreference and segment markup

**Example**

**Text:** *Кубок не помещается в коричневый чемодан, потому что он слишком большой.*

**Question:** *Кто слишком большой? Кубок илиЧемодан*

**Coreference:** True

# Evaluation. Benchmark

*Diagnostic dataset: c*overing 33 linguistic phenomena

- **Lexical Semantics:** lexical entailment, factivity, quantifiers, named entities, symmetry etx.;
- **Logic:** negation and double negation, intervals or numbers, temporal, conjunction and disjunction, conditionals, universal and existential;
- **Predicate-Argument Structure:**
  - anaphora and coreference, coordination scope, active or passive voice, ellipsis or implicits,
  - nominalization, relative clauses, datives, genitives and partitives;
- **Knowledge:** common sense, world knowledge

# Evaluation. Probing

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

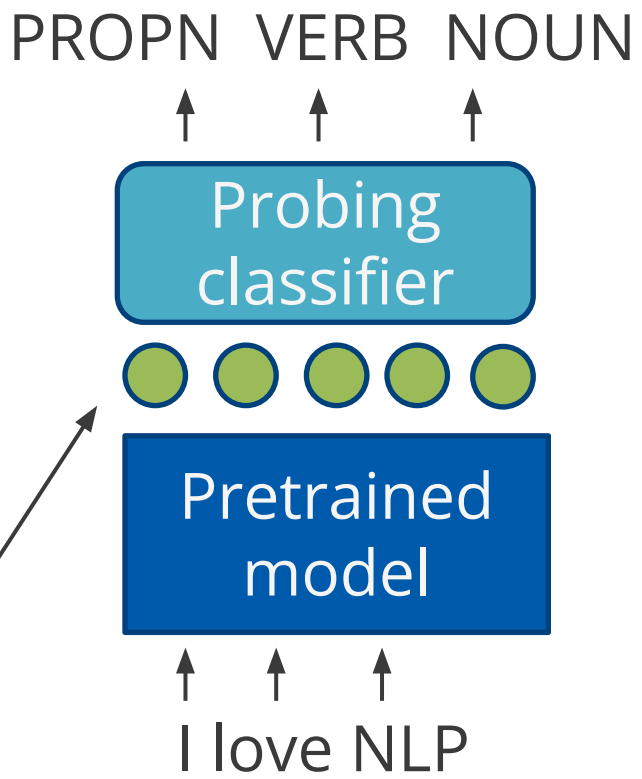
The field of model interpretation is based on the concept of **probing tasks** which capture different linguistic properties.

| | |
|---|---|
| *Models are great* | 3 |
| *Models are great* | Subject number |

The most classic approach is to use a probing classifier which is trained to predict the property.

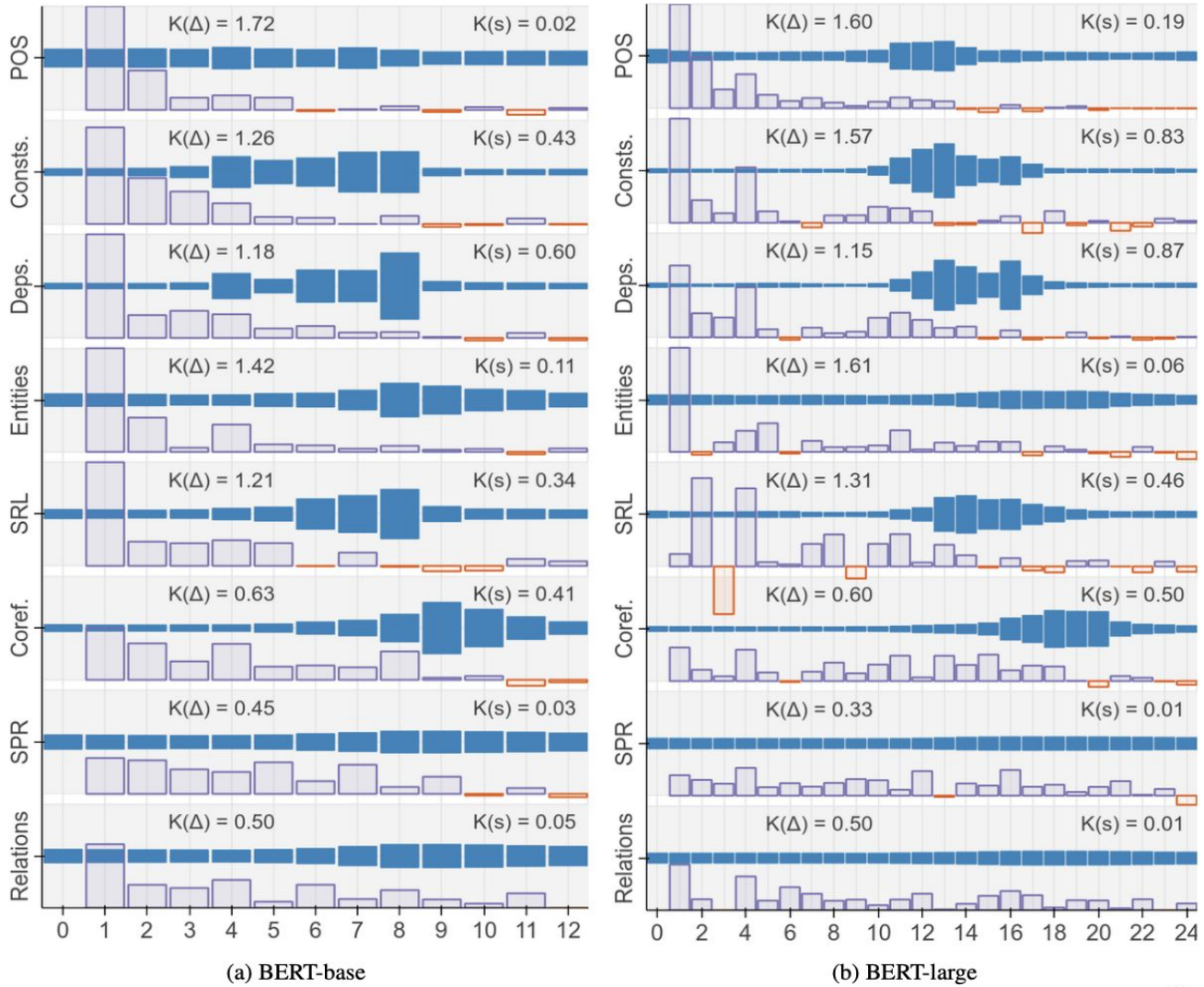
# Evaluation. Probing

- Freeze the weights of the model
- Produce embeddings for the probing tasks
- Train a classifier to predict some linguistic labels using these embeddings
- Use the classifier's accuracy as a measure of how well the representations encode the properties.




**Representations: weights for layers**

# Evaluation. Probing




(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.

# Compression of pretrained models

# Compression of pretrained models

Pretrained models are huge

The number of parameters are growing (the quality as well)

=> The more parameters, the more difficult it is to train, does not fit into memory, time-resource etc.

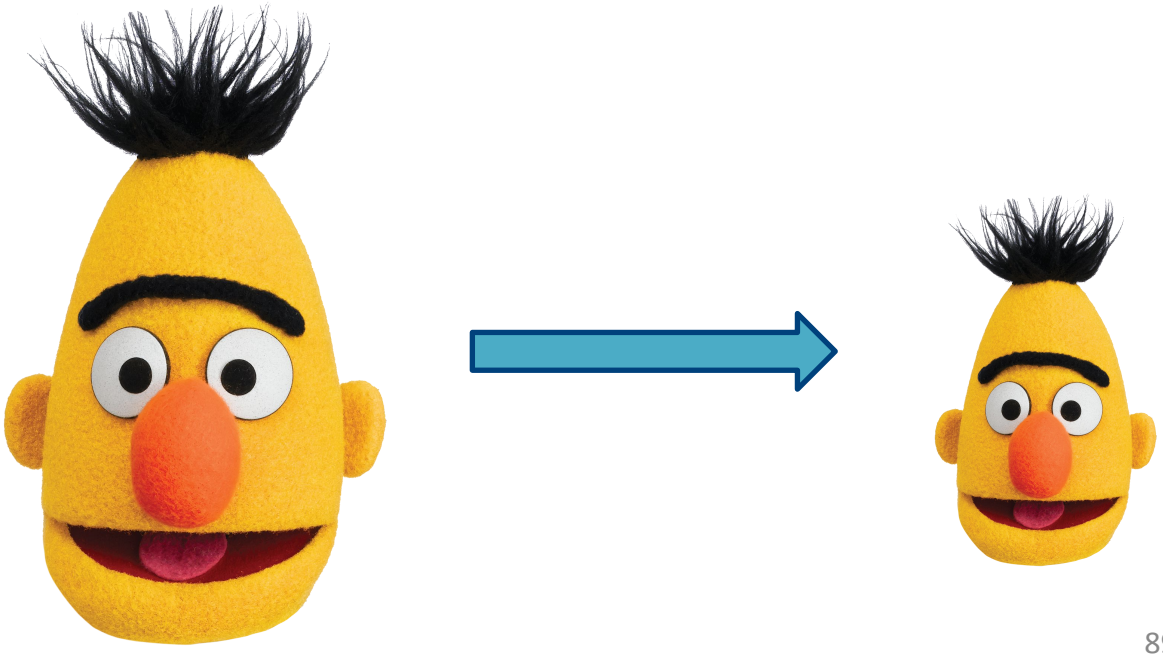
We want to **reduce the size of the model**, but left performance the same.

Getting faster/smaller networks is important for running models on mobile devices.

# Types

Three techniques of how to reduce model sizes:

- Quantization
- Pruning
- Distillation



# Quantization

**Quantization** is a conversion technique that can reduce model size while also improving CPU and hardware accelerator latency, with little degradation in model accuracy.

- introducing approximations
- resulting networks have slightly less accuracy

# Quantization

*Example: Q8Bert*

- All weights of the model we convert in Int8 (-127 to 128)
- Quantization is used during the model tuning
- The amount of memory used by the model is reduced by 4 times

=> inference time increase till 3.7 times

=> quality is kept on the level 99% of BERT on GLUE



arxiv.org

# Pruning

**Pruning** is a model compression technique that allows us to compress the model to a smaller size with zero or marginal loss of accuracy.

Pruning eliminates the weights with low magnitude. Both original and pruned model has the same architecture, with the pruned model being sparser.

+ not require training a model from scratch

- Weight pruning
  (set individual weights in the weight matrix to zero)
- Heads pruning (cut some heads)
- Unit/Neuron pruning

# Pruning

*Example: Poor Man's BERT*

Delete some weights in model.

- Delete some of the weights using different strategies:
  - only upper/bottom layers
  - even/odd layers
  - random
- After deleting, we tune model on target task

=> quality is kept on the level 98% of BERT on GLUE

# Pruning

*Example: Are sixteen heads better that one?*

Origin Bert has 16 heads in Multi-head Attention.

Delete some of them.

- Some attention heads we multiply on 0. We exclude them
- Some heads are really excess, as some of the heads create noise

⇒ inference time speed the model on 17% (- 50% of heads)

⇒ quality stays the same

# Distillation

**Knowledge Distillation** is a procedure for model compression, in which a small (student) model is trained to match a large pre-trained (teacher).

Knowledge is transferred from the teacher model to the student by minimizing a loss function, aimed at matching softened teacher logits as well as ground-truth labels.

# Distillation

**Step 1:**

Take big pre-trained model (for example BERT). Teacher

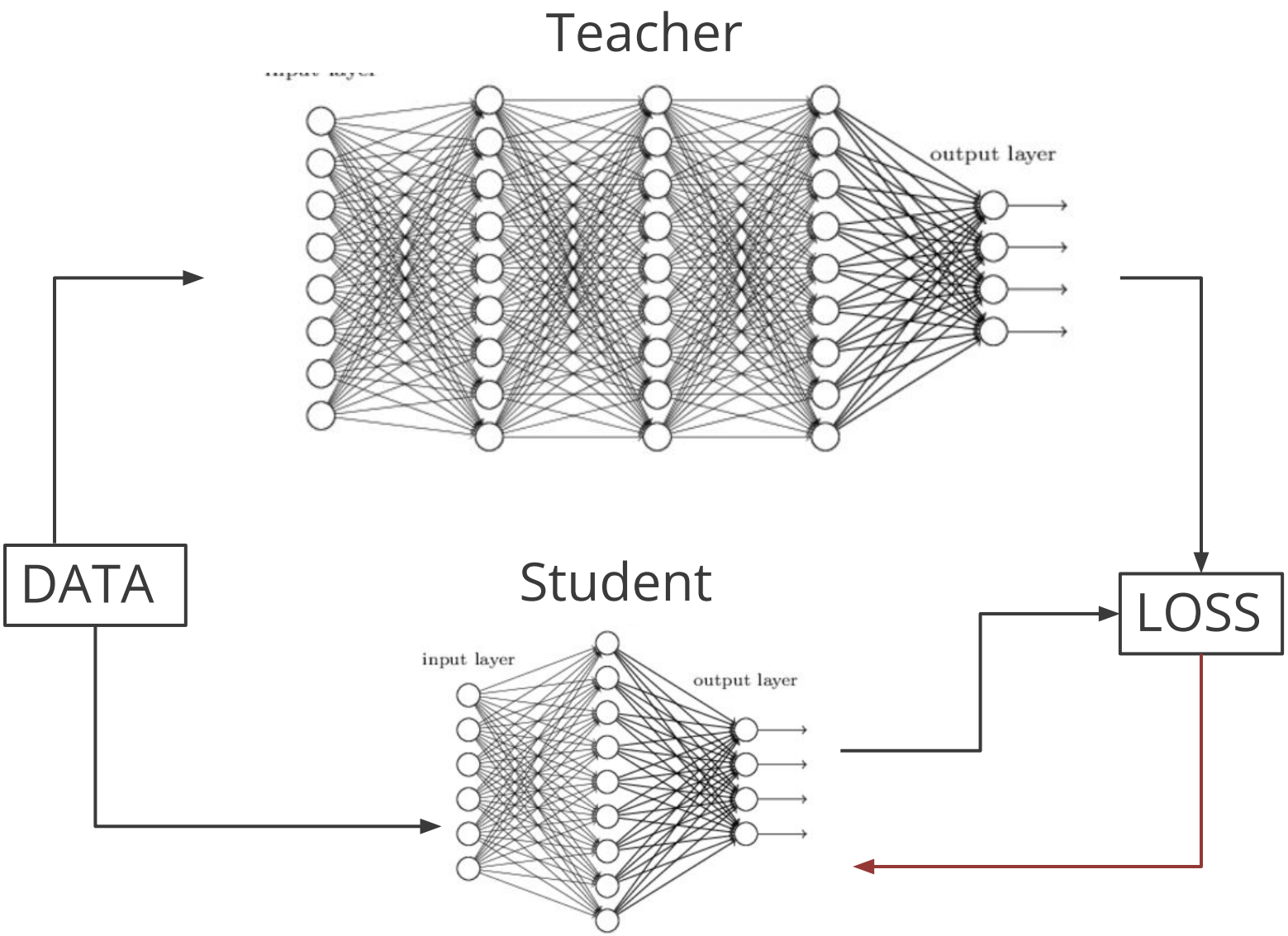
**Step 2:**

Train small model (Student) to reproduce predictions of a large model. Any train task, for example MLM.

**Step 3:**

Take predictions from Teacher. Train only Student model

# Distillation



# Distillation

*Example: DistilBERT*

- KL-divergence instead cross-entropy. Distance between two distributions (they should be similar between teacher and student distributions).
- Too times less parameters than in original BERT
- Performance on GLUE only 5% lower
- DistillBERT can be further be fine-tuned

# Summary

| | Quantization | Pruning | Distillation |
|---|---|---|---|
| Model size | reduce | reduce | reduce |
| Weights | the same | less | less |
| Inference speed | faster | faster | faster |
| Fine-tuning | No | NO | YES |

# Related work

- BERTs:
  - ruberts: https://huggingface.co/DeepPavlov
- GPTs:
  - demo gpt: https://russiannlp.github.io/rugpt-demo/
  - gpt models: https://github.com/sberbank-ai/ru-gpts
- Models for Russian: https://huggingface.co/sberbank-ai
- model zoo: https://github.com/sberbank-ai/model-zoo

- Evaluation:
  - Probing https://github.com/RussianNLP/rusenteval
  - Superglue https://super.gluebenchmark.com/
  - Russian SuperGLUE https://russiansuperglue.com/

- Surveys about Transformers:
  - https://arxiv.org/abs/2106.04554
  - https://arxiv.org/abs/2009.06732

- Stanford lectures:
  - http://web.stanford.edu/class/cs224n/slides/cs224n-2021-lecture09-transformers.pdf
  - http://web.stanford.edu/class/cs224n/slides/cs224n-2021-lecture10-pretraining.pdf