WOJCIECH PRATKOWIECKI
J. CHOROWSKI, UWR

ELMO, BERT & GPT-2

WORD EMBEDDING: NO CONTEXT

$$\begin{bmatrix} 0.8 \\ 0.3 \\ 0.1 \end{bmatrix}$$

$$\begin{bmatrix} 0.7 \\ 0.5 \\ 0.1 \end{bmatrix}$$

$$\begin{bmatrix} 0.1 \\ 0.2 \\ 0.8 \end{bmatrix}$$

WORD EMBEDDING : NO CONTEXT

$$\begin{bmatrix} 0.8 \\ 0.3 \\ 0.1 \end{bmatrix}$$

$$\begin{bmatrix} 0.7 \\ 0.5 \\ 0.1 \end{bmatrix}$$

$$\begin{bmatrix} 0.1 \\ 0.2 \\ 0.8 \end{bmatrix}$$

ELMO

ELMO

- **ELMo** = Embeddings from Language Models
- Instead of using a fixed embedding for each word, ELMo looks at the entire sentence before assigning each word an embedding
- Uses bidirectional language model (biLM)

ELMO

- Given a token sequence $(t_1, t_2, \ldots, t_N)$, a forward language model maximizes:

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k | t_1, t_2, \ldots, t_{k-1})$$

- A backward model predicts the previous token given the context:

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k | t_{k+1}, t_{k+2}, \ldots, t_N)$$

ELMO

- models complex characteristics of word use (e.g., syntax and semantics) and linguistic context (i.e., to model polysemy)
- High-level layers of the model captures context-dependent aspects of the word
- Low-level layers model aspect of syntax (can be used to POS tagging)

ELMO

- For each token t_k ELMo a L-layer biLM computes 2L+1 representations:

$$R_k = \left\{ \mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} | j = 1, \dots, L \right\} = \left\{ \mathbf{h}_{k,j}^{LM} | j = 0, \dots, L \right\}$$

  - $\mathbf{h}_{k,0}^{LM}$ is the token layer and $\mathbf{h}_{k,j}^{LM} = \left[\overrightarrow{\mathbf{h}}_{k,j}^{LM}; \overleftarrow{\mathbf{h}}_{k,j}^{LM} \right]$

ELMO

- ELMo concatenates all layers into a single vector

$$\mathbf{ELMo}_k = E(R_k; \mathbf{\Theta}_e)$$

- For specific task we compute:

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

  - s^{task} are softmax weights,
  - γ^{task} is used to scale the entire vector

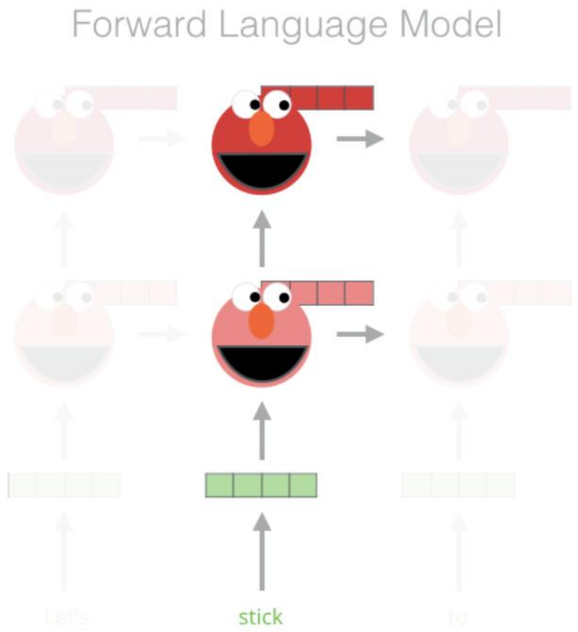
ELMO

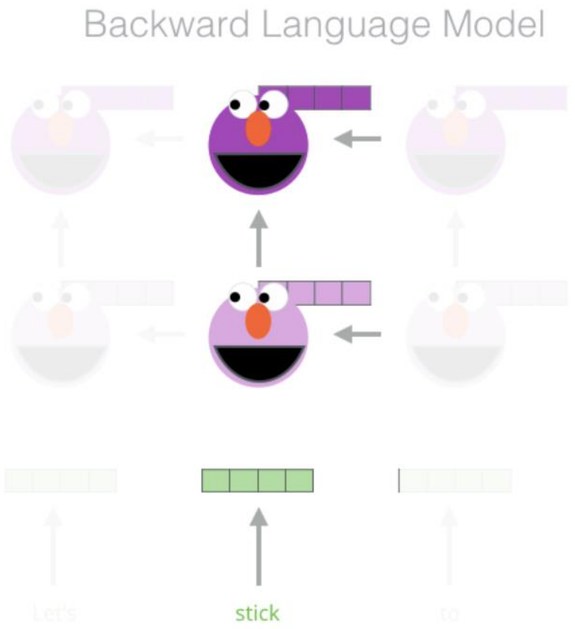
1- Concatenate hidden layers

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

3- Sum the (now weighted) vectors

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





USING PRE-TRAINED MODEL

- To use ELMo for specific, supervised task we run the biLM and record all layers representations for each word
- We only need to train the linear combination of these representations

EXPERIMENTS

Task	Previous SOTA		Our baseline	ELMo + baseline	Increase (absolute/ relative)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

Table 1: Test set comparison of ELMo enhanced neural models with state-of-the-art single model baselines across six benchmark NLP tasks. The performance metric varies across tasks – accuracy for SNLI and SST-5; F_1 for SQuAD, SRL and NER; average F_1 for Coref. Due to the small test sizes for NER and SST-5, we report the mean and standard deviation across five runs with different random seeds. The "increase" column lists both the absolute and relative improvements over our baseline.

ANALYSIS: REGULARIZATION

- L2 $\lambda \| \mathbf{w} \|_2^2$ penalty was added to the loss
- Large λ, like $\lambda = 1$, reduces weighting function to a simple average over layers
- small values, like $\lambda = 0.001$ allow weights to vary

Task	Baseline	Last Only	All layers	
			λ=1	λ=0.001
SQuAD	80.8	84.7	85.0	**85.2**
SNLI	88.1	89.1	89.3	**89.5**
SRL	81.6	84.1	84.6	**84.8**

Table 2: Development set performance for SQuAD, SNLI and SRL comparing using all layers of the biLM (with different choices of regularization strength λ) to just the top layer.

ANALYSIS: INFORMATION CAPTURE BY LAYERS

- The first layer of biLM encodes some context of the word
- The second layer captures syntax information

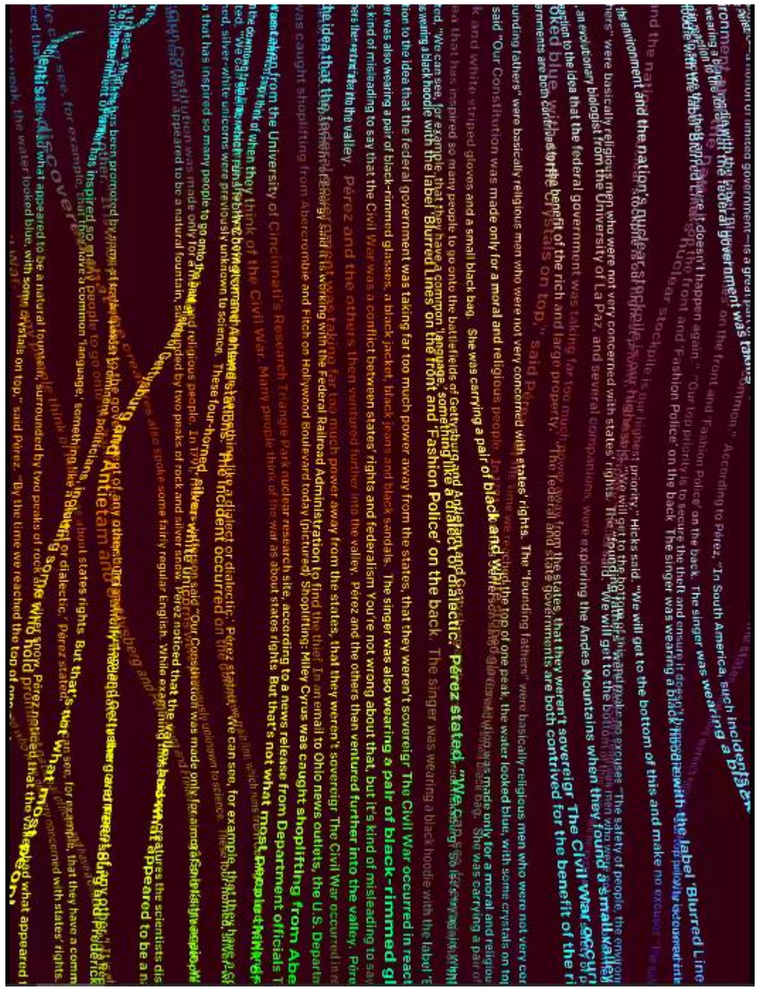
Model	F_1
WordNet 1st Sense Baseline	65.9
Raganato et al. (2017a)	69.9
Iacobacci et al. (2016)	**70.1**
CoVe, First Layer	59.4
CoVe, Second Layer	*64.7*
biLM, First layer	67.4
biLM, Second layer	*69.0*

Table 5: All-words fine grained WSD F_1. For CoVe and the biLM, we report scores for both the first and second layer biLSTMs.

Model	Acc.
Collobert et al. (2011)	97.3
Ma and Hovy (2016)	97.6
Ling et al. (2015)	**97.8**
CoVe, First Layer	*93.3*
CoVe, Second Layer	92.8
biLM, First Layer	*97.3*
biLM, Second Layer	96.8

Table 6: Test set POS tagging accuracies for PTB. For CoVe and the biLM, we report scores for both the first and second layer biLSTMs.

GPT 1 & 2



GPT

- Transformer Language Model
- Unsupervised pre-training on a huuuge corpus
- Pre-trained model is learned once and then used for many various NLP tasks.

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$

GPT

- Supervised fine-tuning - single linear layer is learned for given task. Input token sequence $(x^1, x^2, \ldots, x^m)$ is passed through pre-trained model to obtain h^m
- for label y: $P(y|x^1, \ldots, x^m) = \text{softmax}\left(h^m \cdot W_y\right)$
- We want to maximize $L_2(C) = \sum_{(x,y)} \log P(y|x^1, \ldots, x^m)$
- Objective for the whole model can be specified as:

$$L_3(C) = L_2(C) + \lambda \cdot L_1(C)$$

GPT

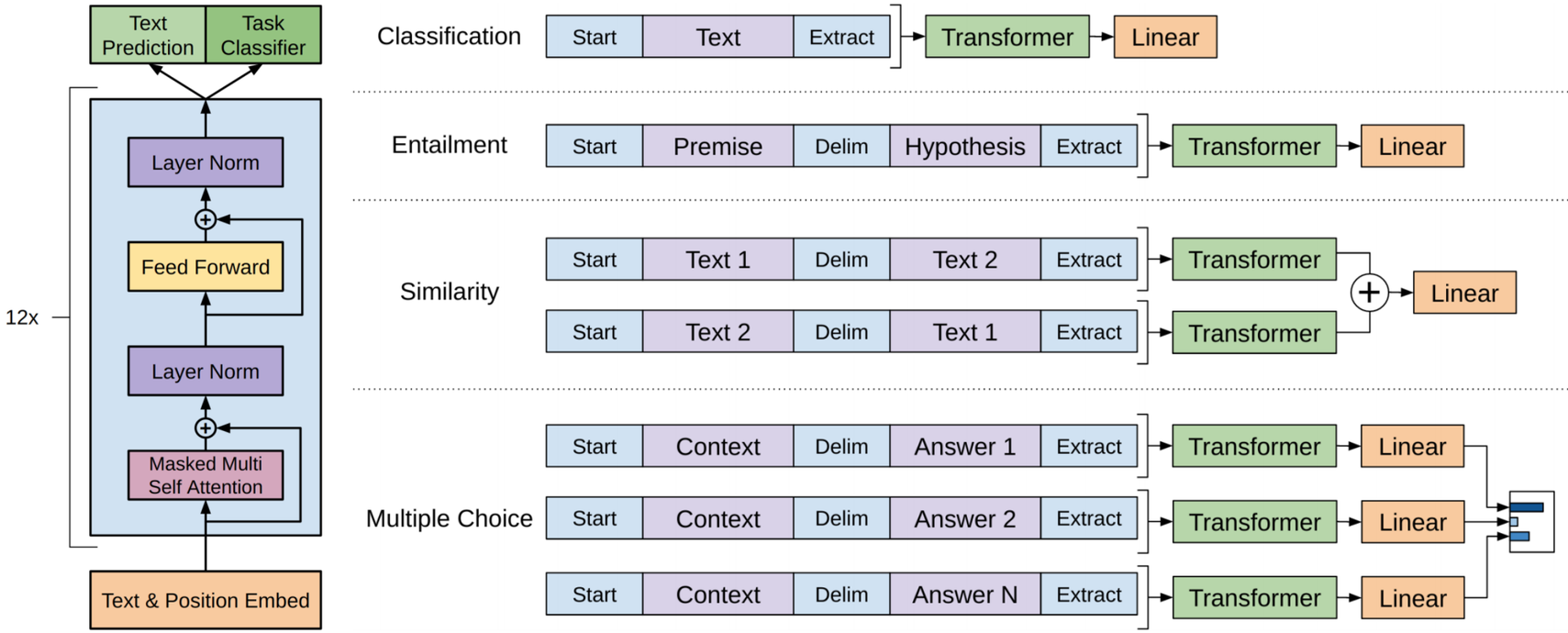


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

GPT-2

- Trained on dataset of 8 million web pages
- „GPT-2 is a direct scale-up of GPT, with more than 10X the parameters and trained on more than 10X the amount of data.”
- OpenAI did not release the dataset, training code, or GPT-2 model weights „*due to concerns about large language models being used to generate deceptive, biased, or abusive language at scale”*

GPT-2

SYSTEM PROMPT (HUMAN-WRITTEN)

Legolas and Gimli advanced on the orcs, raising their weapons with a harrowing war cry.

MODEL COMPLETION (MACHINE-WRITTEN, FIRST TRY)

The orcs' response was a deafening onslaught of claws, claws, and claws; even Elrond was forced to retreat. "You are in good hands, dwarf," said Gimli, who had been among the first to charge at the orcs; it took only two words before their opponents were reduced to a blood-soaked quagmire, and the dwarf took his first kill of the night. The battle lasted for hours until two of the largest Orcs attempted to overwhelm Aragorn. When they finally stopped, they lay defeated and lifeless for miles and miles.

"I take nothing," said Aragorn. "But I give my word, at my peril and mine, that I will never forget this day of horror. None of us will forget. Ever!"

"I'll never forget it!" cried Gimli, who had been in the thick of the battle but hadn't taken part in it. One of the wounded orcs he had carried off, he was the only one of the survivors who remained uninjured. "We'll keep the memory of that day of evil, and the war with it, alive as long as we live, my friends!"

"Then we'll keep it alive as long as we live," added Legolas. "And we won't forget the first great battle of the night, even if we may have forgotten the final defeat."

"I agree," Gandalf said, "but we will all remember it as the last battle in Middle-earth, and the first great battle of the new day."

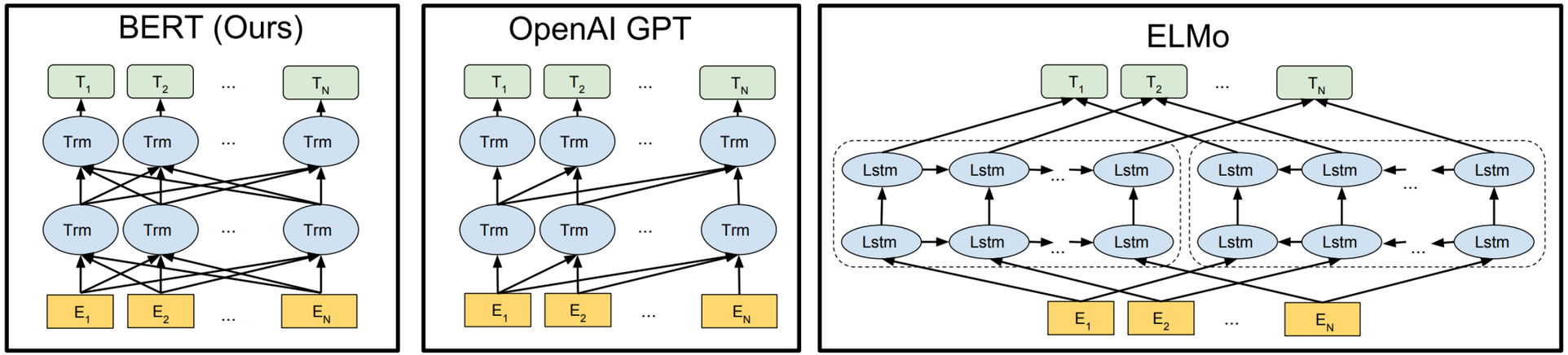
Aragorn drew his sword, and the Battle of Fangorn was won. As they marched out through the thicket the morning mist cleared, and the day turned to dusk.

BERT

BERT

- BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers.
- Bert uses pre-trained bidirectional model for language representations
- To adjust to given task fine-fine-tuning is made, which requires low computational cost
- Trained on BookCorpus (800M words) and English Wikipedia (2500M words)

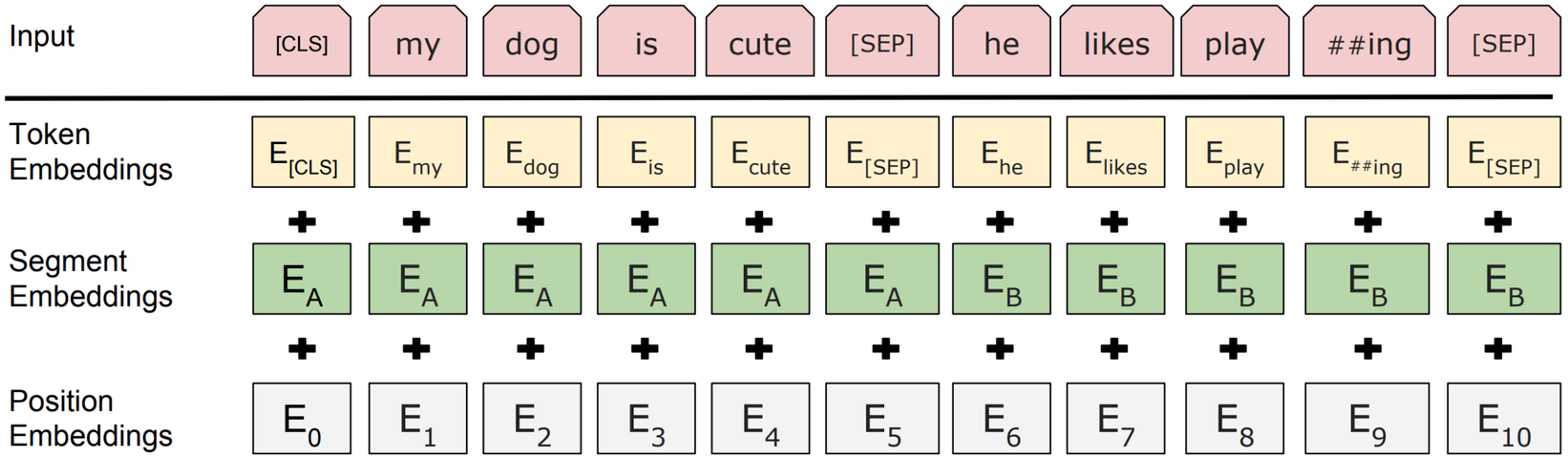
BERT



BERT: INPUT REPRESENTATION

- WordPiece embedding is used at the very beginning
- Positional embedding is used with maximal sequence length = 512
- First token is always [CLS] - its output is used in classification tasks, for other is ignored
- If two sequences are to be processed they are separated with [SEP] token

BERT


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

BERT: MASKED LM

- There is a problem with biLM - each token is able to indirectly „see itself” in multi-layer model
- We randomly **mask** 15% of input words. BERT task is only to predict the masked words

Input: The man went to the $[MASK]_1$. He bought a $[MASK]_2$ of milk .
Labels: $[MASK]_1$ = store; $[MASK]_2$ = gallon

BERT: MASKED LM

- Mismatch between pre-training and fine-tuning: as [MASK] token is seen only during pre-training
- To mitigate, during pre-training we randomly pick 15% of tokens and:
  - 80% - replace the word with [MASK] token
  - 10% - Replace the word with random word
  - 10% - Keep the word unchanged

BERT: MASKED LM

- If [MASK] was used 90% of the time and random word 10% of the time, the model could learn that observed word is never correct
- If [MASK] was used 90% of the time and the real word 10% of the time, then the model could just trivially copy the non-contextual embedding.
- Random replacement occurs only 1.5% of examples - it should not harm the model’s understanding capability

BERT: NEXT SEQUENCE TRAINING

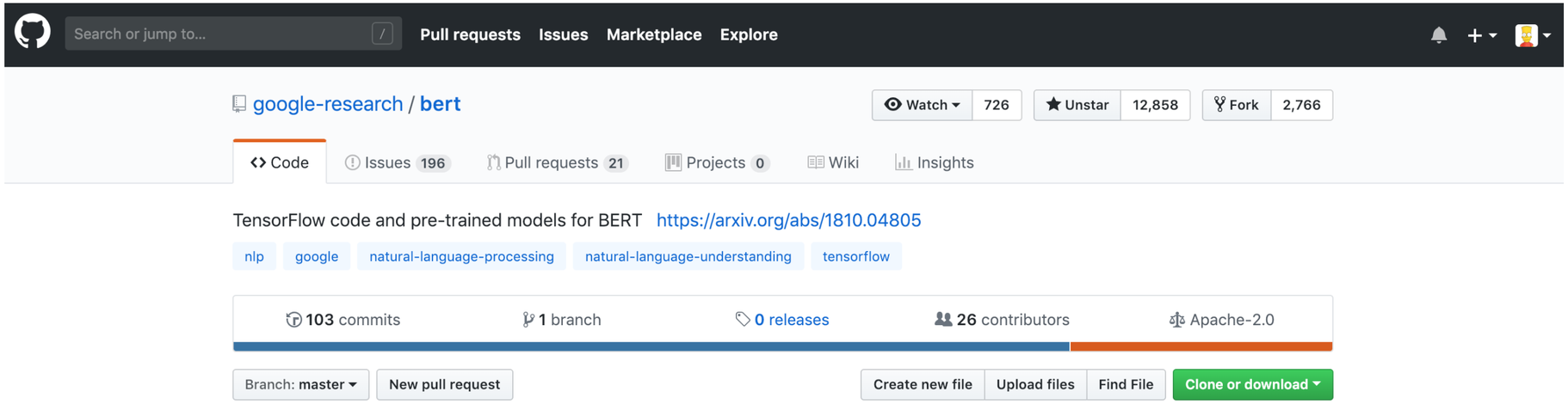
- Many important NLP tasks are based on understanding the relationship between two text sequences
- BERT was trained on *next sentence prediction task* - two sequences **A** and **B** are used,
  - 50% of the time B is actual next sentence that follows A, 50% it is a random sequence from the corpus
- BERT predicts **Next/NotNext** labels with 98% accuracy

BERT: EXPERIMENTS

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

BERT

- Google has open-sourced pre-trained BERT model
- Also available in Google Colab Notebook



NLP LEADERBOARD (AROUND MARCH 2019)

GLUE Tasks Leaderboard FAQ Diagnostics Submit Login

Rank	Name	Model	URL	Score	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m	MNLI-mm	QNLI	RTE	WNLI	AX
1	GLUE Human Baselines	GLUE Human Baselines		87.1	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4	92.0	92.8	91.2	93.6	95.9	-
2	王玮	ALICE large (Alibaba DAMO NLP)		82.9	61.6	95.2	91.1/87.7	89.6/88.6	74.0/90.4	87.9	87.4	95.4	80.9	65.1	40.7
3	Microsoft D365 AI & MSR AI	MT-DNNv2 (BigBird)		82.9	62.5	95.6	91.1/88.2	89.5/88.8	72.7/89.6	86.7	86.0	94.9	81.4	65.1	40.3
4	Jason Phang	BERT on STILTs		82.0	62.1	94.3	90.2/86.6	88.7/88.3	71.9/89.4	86.4	85.6	92.7	80.1	65.1	28.3
5	Jacob Devlin	BERT: 24-layers, 16-heads, 1024-hidden		80.5	60.5	94.9	89.3/85.4	87.6/86.5	72.1/89.3	86.7	85.9	92.7	70.1	65.1	39.6
6	Neil Houlsby	BERT + Single-task Adapters		80.2	59.2	94.3	88.7/84.3	87.3/86.1	71.5/89.4	85.4	85.0	92.4	71.6	65.1	9.2
7	Alec Radford	Singletask Pretrain Transformer		72.8	45.4	91.3	82.3/75.7	82.0/80.0	70.3/88.5	82.1	81.4	-	56.0	53.4	29.8
8	Samuel Bowman	BiLSTM+ELMo+Attn		70.5	36.0	90.4	84.9/77.9	75.1/73.3	64.8/84.7	76.4	76.1	-	56.8	65.1	26.5

NLP LEADERBOARD (AROUND MARC

GLUE SuperGLUE Paper Code Tasks Leaderboard FAQ Diagnostics Submit Login

Rank	Name	Model	URL	Score	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m	MNLI-mm	QNLI	RTE	WNLI	AX
1	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	97.5	91.7	94.5	51.2
2	ERNIE Team - Baidu	ERNIE		90.4	74.4	97.5	93.5/91.4	93.0/92.6	75.2/90.9	91.4	91.0	96.6	90.9	94.5	51.7
3	Alibaba DAMO NLP	StructBERT		90.3	75.3	97.1	93.9/91.9	93.0/92.5	74.8/91.0	90.9	90.7	96.4	90.2	94.5	49.1
4	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	96.9	92.8	94.5	53.1
5	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	99.2	89.7	94.5	50.2
6	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	95.8	89.8	91.8	50.7
7	Huawei Noah's Ark Lab	NEZHA-Large		88.7	67.4	97.2	93.2/91.0	92.2/91.6	74.1/90.2	90.8	90.2	95.7	88.5	93.2	45.0
8	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	95.6	88.7	89.0	50.1
9	Junjie Yang	HIRE-RoBERTa							92.4/92.0	74.3/90.2	90.7	90.4	95.5	87.9	89.0	49.3

Click on a submission to see more information

BIBLIOGRAPHY

- *Efficient Estimation of Word Representations in Vector Space*, Mikolov et al., 2013
- *Distributed Representations of Words and Phrases and their Compositionality,* Mikolov et al., 2013
- *Deep contextualized word representations*, Peters et al., 2018
- *Neural Machine Translation by Jointly Learning to Align And Translate*, Bahdanau et al., 2016
- *Attention Is All You Need*, Vaswani et al., 2017
- *Improving Language Understanding by Generative Pre-Training*, Radford et al., 2018
- *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,* Devlin et al., 2018
- *Language Models are Unsupervised Multitask Learners,* Radford et al., 2019

Thank you
For your attention