Lecture 10: BERT

Andrei Arsene Simion

Columbia University

April 5, 2023

Outline

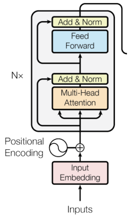
- BERT
- ROBERTA
- Knowledge Distillation
- Distil BERT
- Sentence BERT

Outline

- BERT
- ROBERTA
- Knowledge Distillation
- Distil-BERT
- Sentence BERT

BERT

- We saw Encoder-Decoder Transformers can be used to get SOTA Machine Translation results
- By training to predict the next word, we also saw GPT-1 can be applied to to generate text and also do other tasks (via fine tuning)
- Goal: How can we apply transformers to get bidirectional-contextual embeddings?
  - GPT-1 is a Decoder, so any token's embedding is only looking at things to its left (past)
  - We could have two GPT-1 models, one going left and another going right to get a weakly "bidirectional" model
- Can we apply just the Encoder to this task?



BERT: An Encoder Transformer

- BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
- BERT was a model that uses the Transformer Encoder architecture in novel way
    - An Encoder essentially takes in some input and outputs the same input in a refined way
    - Imagine we insert a set of tokens and run them through a Transformer Encoder
    - Given bidirectional Encoder's context, it is unclear what the objective function would be; we know everything to the left and right
        - For ELMo, notice that the model is weakly bidirectional

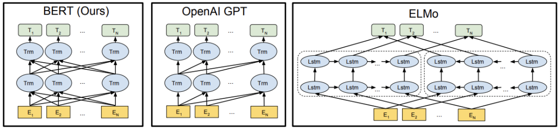
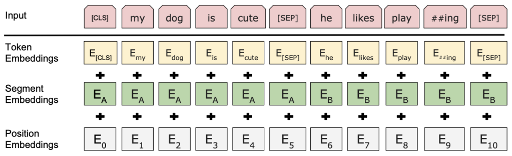


Figure 3: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTMs to generate features for downstream tasks. Among the three, only BERT representations are jointly conditioned on both left and right context in all layers. In addition to the architecture differences, BERT and OpenAI GPT are fine-tuning approaches, while ELMo is a feature-based approach.

BERT Input

- Input of BERT is a pair of 2 segments (A, B)
  - Tokenize A and B and insert 3 special tokens: [CLS] before A, and 2 [SEP] tokens, one between A and B and another after B
  - Add in positional embeddings which contain information related to where a token is
  - Use segment embeddings to denote if we are looking at a token in the first or second sentence
- One question we can ask (NSP): Does B follow A?
  - NSP = Next Sentence Prediction

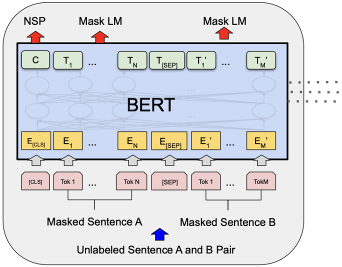


Input
[CLS] my dog is cute [SEP] he likes play ##ing [SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

Masked Language Modeling (MLM)

- In the 1950s, one Language Modeling NLP objective was the Cloze task: Fill in the missing blank
- "I am ___ to the store but my car is acting up."
  - Probably you can guess that "driving" should go in the above
- Essentially, BERT took MLM and NSP as the objective
  - MLM: Mask some words. What are the missing words?
  - Next Sentence Prediction: Pick two segments as A and B. Does B follow A?



Pre-training

Objective Details

- An input to BERT might look like below
  - $x_{original}$ = "[CLS] my dog is nice [SEP] he likes play ##ing [SEP]"
  - x = "[CLS] my dog is [MASK] [SEP] he [MASK] play ##ing [SEP]"
  - y = (1, "nice", "likes")
- You only predict masked tokens and the 1/0 NSP task on top of [CLS]
- How much do you mask?
  - Authors proposed masking 15% of all tokens
    - Of these, 80% become mask in the training data
    - 10% remain as is but have a loss term
    - 10% get a random word inserted on the x side and the loss needs to recover the right word
  - The authors figured that doing the above would force the model to be on its toes: it needs to get good representations for everything not just tokens which have "[MLM]"
  - At test time, "[MLM]" is not used, so the above tries to make training $\sim$ test

What can MLM teach a model?

- What can we learn from the below?
- "Columbia University is located in ___, New York."
  - "New York"
  - Learning Trivia!
- "I started ___ car and drove to the restaurant"
  - "my"
  - "the"
  - "his"
  - Syntactic categories!
- "I walked across the street, and checked the traffic over ___ shoulder."
  - "my"
  - This seems to imply some sort of coreference!

Notes on BERT

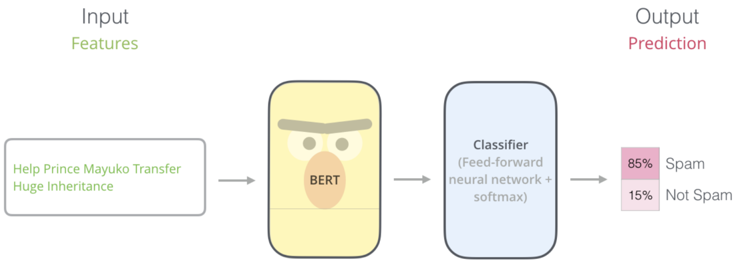
- A better example for data might be as below
- $R(word)$ means the word got replaced with a random word
- $I(word)$ means the word is the same
  - $x_{original}$ = "[CLS] my dog is an amazing animal [SEP] he likes play ##ing [SEP]"
  - x = "[CLS] my I(dog) is an [MASK] R(car) [SEP] he [MASK] play ##ing [SEP]"
  - y = (1, "dog", "amazing", "animal", "likes")
- We get several MLM CrossEntropy loss terms (depending on how many words we masked out)
- The NSP task gives a BinaryCrossEntropy term
- Basically, for each token that you need to predict, get its feature representation vector $H \in \mathbb{R}^{d_{model}}$ and then do $softmax(ReLU(WH + b))$ where $W \in \mathbb{R}^{V \times d_{model}}$ and $b \in \mathbb{R}^{V}$
- For [CLS] and NSP, we do the same but V is replaced by 2 since you want to predict 1/0

Fine Tuning: Start with LM $\rightarrow$ {NER, Classification, MT}

- Recall the idea of fine tuning
- We want to solve some Task 1 (Classification, NER, MT)
- To do this, we first pick another Task 2 (LM)
- We optimize $\min_\theta Loss_{LM}(\theta)$
- We take the parameters $\hat{\theta}_{LM}$ and use it to seed another optimization whose goal is to solve Task 1
- We optimize $\min_\theta Loss_{NER}(\theta)$
  - Typically, Task 2 seeds the word embeddings or features fed into Task 1
  - I.e. $Loss_{NER}$ is fed features by inserting data through the model gotten from optimizing $Loss_{LM}$
  - Task 2 as a feature extractor: do not back propagate gradients into Task 1's architecture, leave it static
  - Task 2 fine tuning: back propagate gradients both into the new task specific "head" AND Task 2's model

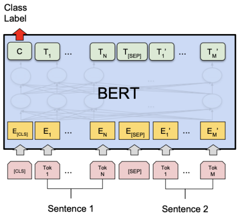
Fine Tuning BERT: General Ideas

- Why does fine tuning work?
  - Solving a problem directly might be hard - we need good inputs
  - Via fine tuning, we are giving the model aimed at solving (main) Task 1 a head start
  - Neural models are highly con convex, so initialization matters
  - We learn general features from that Task 2 (LM), and then either use them (static) or fine tune them to the specific task
- Typically, the (main) Task 1 is framed as a combination of Task 2 ' s architecture and a Task 1 specific "head"
  - Task 1: Classification
  - Task 2: Some BERT variant

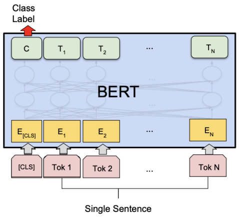

Input
Features
Output
Prediction
Help Prince Mayuko Transfer Huge Inheritance
BERT
Classifier
(Feed-forward neural network + softmax)
85% Spam
15% Not Spam

Fine Tuning BERT: Paper Architectures

- Big idea of Fine tuning BERT is either
  - Using the [CLS] token embedding C with a classification head to predict 1/0 tasks
  - Using the Embeddings of each token T_i to predict spans of an answer (SQUAD)
  - Using BERT for NER: Named-Entity Recognition



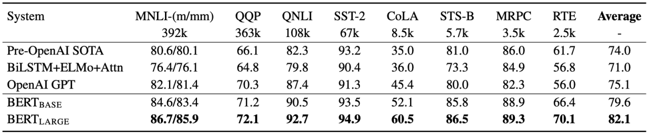
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



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

Results on Classification

- $\text{BERT}_{BASE} = 110$ M Parameters
- $\text{BERT}_{LARGE} = 340$ M Parameters $\sim$ GPT-1
- For each task related to classification, the authors used the $[CLS]$ token with a $1/0$ "head on top"
  - Get the token representation of [CLS] H
  - Apply softmax and a nonlinearity to get to dimension 2: $softmax(ReLU(WH + b))$
  - Use a CrossEntropy loss



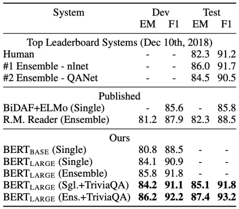
System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1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

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

- RTE: amazing boost on very little data!

Results on Squad 1.0

- SQUAD: Find the answer to the question in some document; the answer is inside the document between indices i and j
- Introduce two new embeddings S, E for start and end span
- Probability token i is the start of a span is $P_i = \frac{\exp^{ST_i}}{\sum_j e^{ST_j}}$; similar formula applied to the end of a span
- Training objective is the sum of the log-likelihoods of the correct start and end tokens
- At prediction time, a span $\hat{s}_{i,j} = \max_{j \geq i}\{ST_i + ET_j\}$ is the score of a span with $j \geq i$; highest span score is the prediction



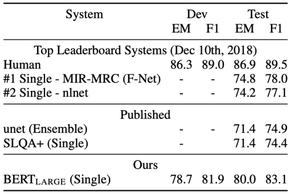
System	Dev EM	Dev F1	Test EM	Test F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
Published				
BiDAF+ELMo (Single)	-	85.6	-	85.8
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
BERT$_{\text{BASE}}$ (Single)	80.8	88.5	-	-
BERT$_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
BERT$_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
BERT$_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
BERT$_{\text{LARGE}}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

Table 2: SQuAD 1.1 results. The BERT ensemble is 7x systems which use different pre-training checkpoints and fine-tuning seeds.

Results on Squad 2.0

- SQUAD 2.0: Like 1.0, but now the answer need not be in the document
- For questions that do not have an answer, their spans start and end at the [CLS] token with embedding C
- A Null answer has score given by $s_{null} = SC + EC$, so this is compared with all the other answers
- Fix τ. A span (i, j) is selected only if

$$\hat{s}_{i,j} = \max_{j \geq i}\{ST_i + ET_j\} > s_{null} + \tau$$

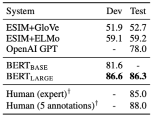


System	Dev		Test	
	EM	F1	EM	F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	86.3	89.0	86.9	89.5
#1 Single - MIR-MRC (F-Net)	-	-	74.8	78.0
#2 Single - nlnet	-	-	74.2	77.1
Published				
unet (Ensemble)	-	-	71.4	74.9
SLQA+ (Single)	-		71.4	74.4
Ours				
BERT$_{\text{LARGE}}$ (Single)	78.7	81.9	80.0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

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

Results on SWAG

- SWAG is a dataset where you are given a sentence and asked to choose among 4 plausible next sentences
- For each element in SWAG, select the source sentence and 1 of each of the 4 other sentences to get 4 inputs to BERT
- Grab the [CLS] token embedding C and and score one sentence continuing another via the score $softmax(ReLU(CW + b))$ where $W \in \mathbb{R}^{d_{model} \times 2}$, $b \in \mathbb{R}^2$
- Use a BinaryCrossEntropy loss which is 1 for the right choice and 0 for any incorrect choice; at predict time, use this head to make a selection

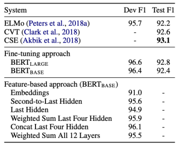


System	Dev	Test
ESIM+GloVe	51.9	52.7
ESIM+ELMo	59.1	59.2
OpenAI GPT	-	78.0
BERT$_{BASE}$	81.6	-
BERT$_{LARGE}$	**86.6**	**86.3**
Human (expert)[†]	-	85.0
Human (5 annotations)[†]	-	88.0

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

Results on NER + Feature Extractor

- In NER: You want to tag each word in a sentence with a certain (9) tag: B/I-PER, B/I-MISC, B/I-ORG, B/I-LOC, O
- NER is usually a task done by adding a CRF layer on top of your model
- For the authors, they do not fine-tune BERT but just add a classification head on top of every token to predict 1 of 5 outcomes B-PER, B-MISC, B-ORG, B-LOC, O
- No fine tuning, no Markov / DP-based decoding: it still does really well!



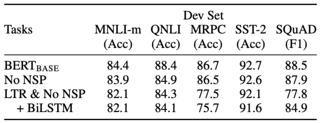
System	Dev F1	Test F1
ELMo (Peters et al., 2018a)	95.7	92.2
CVT (Clark et al., 2018)	-	92.6
CSE (Akbik et al., 2018)	-	**93.1**
Fine-tuning approach		
$\text{BERT}_{\text{LARGE}}$	96.6	92.8
$\text{BERT}_{\text{BASE}}$	96.4	92.4
Feature-based approach ($\text{BERT}_{\text{BASE}}$)		
Embeddings	91.0	-
Second-to-Last Hidden	95.6	-
Last Hidden	94.9	-
Weighted Sum Last Four Hidden	95.9	-
Concat Last Four Hidden	96.1	-
Weighted Sum All 12 Layers	95.5	-

Table 7: CoNLL-2003 Named Entity Recognition results. Hyperparameters were selected using the Dev set. The reported Dev and Test scores are averaged over 5 random restarts using those hyperparameters.

Wolff	B-PER
,	O
currently	O
a	O
journalist	O
in	O
Argentina	B-LOC
,	O
played	O
with	O
Del	B-PER
Bosque	I-PER

Ablation Studies

- Generally, each step is important, with the bidirectional nature of BERT being most important
- NSP is the "least" important

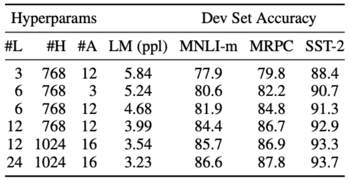


	Dev Set				
Tasks	MNLI-m (Acc)	QNLI (Acc)	MRPC (Acc)	SST-2 (Acc)	SQuAD (F1)
BERT$_{\text{BASE}}$	84.4	88.4	86.7	92.7	88.5
No NSP	83.9	84.9	86.5	92.6	87.9
LTR & No NSP	82.1	84.3	77.5	92.1	77.8
+ BiLSTM	82.1	84.1	75.7	91.6	84.9

Table 5: Ablation over the pre-training tasks using the BERT$_{\text{BASE}}$ architecture. "No NSP" is trained without the next sentence prediction task. "LTR & No NSP" is trained as a left-to-right LM without the next sentence prediction, like OpenAI GPT. "+ BiLSTM" adds a randomly initialized BiLSTM on top of the "LTR + No NSP" model during fine-tuning.

Size Experiments

- As you make the model larger, it performs better



Hyperparams				Dev Set Accuracy		
#L	#H	#A	LM (ppl)	MNLI-m	MRPC	SST-2
3	768	12	5.84	77.9	79.8	88.4
6	768	3	5.24	80.6	82.2	90.7
6	768	12	4.68	81.9	84.8	91.3
12	768	12	3.99	84.4	86.7	92.9
12	1024	16	3.54	85.7	86.9	93.3
24	1024	16	3.23	86.6	87.8	93.7

Table 6: Ablation over BERT model size. #L = the number of layers; #H = hidden size; #A = number of attention heads. “LM (ppl)” is the masked LM perplexity of held-out training data.

Outline

- BERT
- ROBERTA
- Knowledge Distillation
- Distil-BERT
- Sentence BERT

ROBERTA

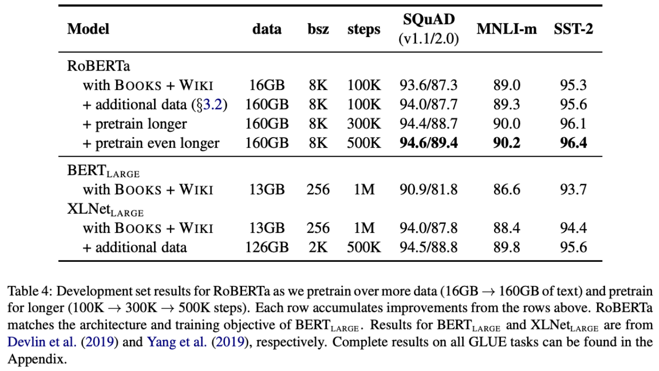
- ROBERTA: Was BERT trained the right way and will training longer make it do better?
- Some items about the Optimization of BERT
  - GELU Activation $\sim$ ReLU
  - 1 M updates, using batches of size 256 tokens and maximum sequence length 512
  - ADAM optimizer, using $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e - 6$ and L_2 weight decay 0.01
  - Warmed up learning rate 10000 steps to a peak of $1e - 4$, then linearly decayed
  - 16 GB of text used, a combination of BOOKCORPUS and WIKIPEDIA
  - Can you tweak the above to improve performance?

ROBERTA vs BERT

- No short sentences
- $\beta_2 = 0.98$
- 160 GB of text, more data; larger batches
- Dynamic Masking: mask at each training iteration, not just once across the data
- NSP
  - NSP Loss, use segments (A, B)
  - NSP Loss, but instead of segments (A, B), use sentences (A, B)
  - No NSP loss, just long sentences that can cross document boundaries
  - No NSP loss, but sentences are from the same document
- Add more tokens

ROBERTA vs BERT

- Seems like using sentences is bad; maybe the model can't learn long range dependencies?
- Seems like NSP loss is not needed, you can just use MLM with longer spans of text
- Seems like BERT was undertrained - more data helps, but so does training!



Model	**data**	**bsz**	**steps**	**SQuAD** (v1.1/2.0)	**MNLI-m**	**SST-2**
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
$\text{XLNet}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4
+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6

Table 4: Development set results for RoBERTa as we pretrain over more data (16GB $\rightarrow$ 160GB of text) and pretrain for longer (100K $\rightarrow$ 300K $\rightarrow$ 500K steps). Each row accumulates improvements from the rows above. RoBERTa matches the architecture and training objective of $\text{BERT}_{\text{LARGE}}$. Results for $\text{BERT}_{\text{LARGE}}$ and $\text{XLNet}_{\text{LARGE}}$ are from Devlin et al. (2019) and Yang et al. (2019), respectively. Complete results on all GLUE tasks can be found in the Appendix.

Outline

- BERT
- ROBERTA
- Knowledge Distillation
- Distil-BERT
- Sentence BERT

Knowledge Distillation

- Suppose we have a large model, which performs really well
- What are some problems?
  - Maybe it is very hard to train this model
  - Maybe the inference speed is prohibitively large
  - Maybe you need the model on a phone ("Edge Device"), and you don't have memory
- Knowledge Distillation offers one way to addresses this issue: you use the larger (teacher) model to help a smaller (student) model learn
- Let's look at the case where the task is Classification with C classes
- Consider one example with C classes and let $\{v_j\}_{j=1}^{C}$ and $\{z_j\}_{j=1}^{C}$ be the logits (pre-softmax) of the larger (teacher) and smaller (student) model
- Let $T > 0$ be a chosen fixed and consider

$$p(T) = softmax(v/T) \text{ and } q(T) = softmax(z/T)$$

- $q = softmax(z)$ as usual, no T

Knowledge Distillation

- What is the effect of T on $p(T) = softmax(v/T)$
  - $T = 1$, do nothing
  - $T \to 0$, this converges to one hot vector, which is 1 at the index where v is maximum
  - $T \to \infty$, this converges to the uniform probability over C
- Example: Suppose we want to predict digits 1 - 10, $p_7 = 0.8$ and $p_1 = 0.1$, $p_i \sim 0$ for $i \neq 7, 1$
  - If we pick $T \to 0$ then $p_7(T), p_1(T)$ go up and the other levels become even more residual
  - Effectively, you might want the student to get information from $1, 7$, but not from other levels, and T allows this
- After you train the teacher, you pick $\alpha > 0$ and minimize:

$$-\alpha \sum_{j=1}^{C} y_j \log q_j - (1-\alpha) T^2 \sum_{j=1}^{C} p(T)_j \log q(T)_j$$

  - α, T, $p(T)$, y are constants here

Knowledge Distillation

- What's the idea of Knowledge Distillation?
  - You want the student to still learn things from the teacher; here, "learns" means same probabilities per target
  - You want the student answers to be similar to the teacher answers
  - Case 1: If the teacher is too sure ($p(1)_j$ is high for one particular j), $T \to \infty$ smooths outs the distribution so $p(T)$ becomes uniform
  - Case 2: If you want to just learn from the teacher directly, you can use $T \sim 0$ and in the limit you use $y_j^t = 1$ if $j = argmax(p_i)$ otherwise $y_j^t = 0$
    - Now, for "hard" Knowledge Distillation you can use

$$-\left(\frac{\sum_{j=1}^{C} y_j \log q_j}{2} + \frac{\sum_{j=1}^{C} y_j^t \log q_j}{2}\right)$$

Outline

- BERT
- ROBERTA
- Knowledge Distillation
- Distil-BERT
- Sentence BERT

Distil-BERT

- Can we get a much smaller model than BERT with performance that is on par?
  - The teacher model is BERT$_{BASE}$, from the BERT paper
- The authors of Distil-BERT tried to do just this
- Their loss L had three terms
  - $L_{ce} = \sum_j p(T)_j \log q(T)_j$ where $p(T)_i$, $q(T)_i$ are the teacher, student probabilities
  - L_{mlm}, a MLM objective on the student
  - L_{cos}, an objective that tends to align the student and teacher hidden states
- Upshot: Their model is 97% as effective as BERT$_{BASE}$, 40% smaller, and 60% faster

Distil-BERT

- Distil-BERT ablation study on GLUE = lower is better

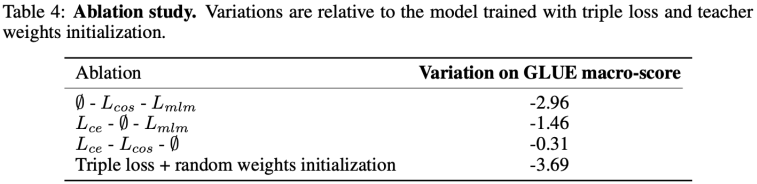


Table 4: **Ablation study.** Variations are relative to the model trained with triple loss and teacher weights initialization.

Ablation	**Variation on GLUE macro-score**
$\emptyset$ - L_{cos} - L_{mlm}	-2.96
L_{ce} - $\emptyset$ - L_{mlm}	-1.46
L_{ce} - L_{cos} - $\emptyset$	-0.31
Triple loss + random weights initialization	-3.69

- Results were very positive given the size

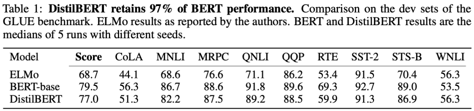


Table 1: **DistilBERT retains 97% of BERT performance.** Comparison on the dev sets of the GLUE benchmark. ELMo results as reported by the authors. BERT and DistilBERT results are the medians of 5 runs with different seeds.

Model	**Score**	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	WNLI
ELMo	68.7	44.1	68.6	76.6	71.1	86.2	53.4	91.5	70.4	56.3
BERT-base	79.5	56.3	86.7	88.6	91.8	89.6	69.3	92.7	89.0	53.5
DistilBERT	77.0	51.3	82.2	87.5	89.2	88.5	59.9	91.3	86.9	56.3

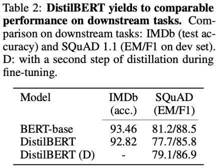


Table 2: **DistilBERT yields to comparable performance on downstream tasks.** Comparison on downstream tasks: IMDb (test accuracy) and SQuAD 1.1 (EM/F1 on dev set). D: with a second step of distillation during fine-tuning.

Model	IMDb (acc.)	SQuAD (EM/F1)
BERT-base	93.46	81.2/88.5
DistilBERT	92.82	77.7/85.8
DistilBERT (D)	-	79.1/86.9

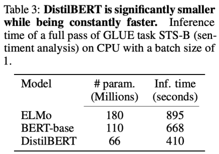
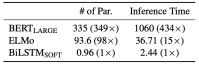


Table 3: **DistilBERT is significantly smaller while being constantly faster.** Inference time of a full pass of GLUE task STS-B (sentiment analysis) on CPU with a batch size of 1.

Model	# param. (Millions)	Inf. time (seconds)
ELMo	180	895
BERT-base	110	668
DistilBERT	66	410

Distil-BERT w BiLSTM

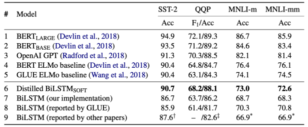
- Another distillation method looks if we can fit BERT into a smaller BiLSTM
- For this work, the authors argue that the distillation objective should be $\sum_i ||z_i - v_i||$ where here z, v are the pre-softmax logits for the student and teacher
- The authors use just the objective above, no more
- The BiLSTM model is more efficient



	# of Par.	Inference Time
BERT$_{LARGE}$	335 (349×)	1060 (434×)
ELMo	93.6 (98×)	36.71 (15×)
BiLSTM$_{SOFT}$	0.96 (1×)	2.44 (1×)

Table 2: Single-sentence model size and inference speed on SST-2. # of Par. denotes number of millions of parameters, and inference time is in seconds.

- The Distilled BiLSTM (BiLSTM$_{SOFT}$) > ELMo?



#	Model	SST-2	QQP	MNLI-m	MNLI-mm
		Acc	F$_1$/Acc	Acc	Acc
1	BERT$_{LARGE}$ (Devlin et al., 2018)	94.9	72.1/89.3	86.7	85.9
2	BERT$_{BASE}$ (Devlin et al., 2018)	93.5	71.2/89.2	84.6	83.4
3	OpenAI GPT (Radford et al., 2018)	91.3	70.3/88.5	82.1	81.4
4	BERT ELMo baseline (Devlin et al., 2018)	90.4	64.8/84.7	76.4	76.1
5	GLUE ELMo baseline (Wang et al., 2018)	90.4	63.1/84.3	74.1	74.5
6	Distilled BiLSTM$_{SOFT}$	**90.7**	**68.2/88.1**	**73.0**	**72.6**
7	BiLSTM (our implementation)	86.7	63.7/86.2	68.7	68.3
8	BiLSTM (reported by GLUE)	85.9	61.4/81.7	70.3	70.8
9	BiLSTM (reported by other papers)	87.6†	– /82.6‡	66.9*	66.9*

Table 1: Test results on different datasets. The BiLSTM results reported by other papers are drawn from Zhou et al. (2016),† Wang et al. (2017),‡ and Williams et al. (2017).* All of our test results are obtained from the GLUE benchmark website.

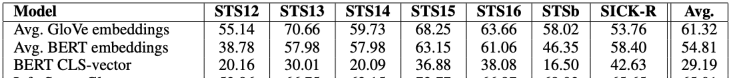
Outline

- BERT
- ROBERTA
- Knowledge Distillation
- Distil-BERT
- Sentence BERT

Main Idea

- Goal: How do we get better Sentence level embeddings from BERT?
- Typically, one can do any of:
  - Use the [CLS] token embedding
  - Use the mean of the embeddings for all other tokens in the sentence
- Is this the right thing to do?
- What is wrong with doing this?
- Remember: The above is if you use BERT directly, feeding sentences through and getting a representation and then doing similarity search

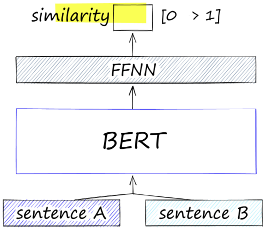
BERT for sentence embeddings: GloVe is better



Model	STS12	STS13	STS14	STS15	STS16	STSb	SICK-R	Avg.
Avg. GloVe embeddings	55.14	70.66	59.73	68.25	63.66	58.02	53.76	61.32
Avg. BERT embeddings	38.78	57.98	57.98	63.15	61.06	46.35	58.40	54.81
BERT CLS-vector	20.16	30.01	20.09	36.88	38.08	16.50	42.63	29.19

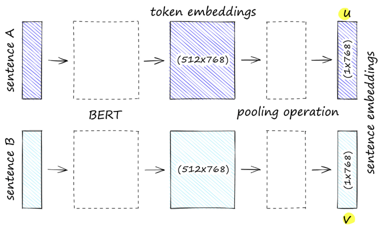
- It was found that using CLS or mean pooling to summarize a sentence was not much stronger than using GloVe from 2014!
  - In fact, on many tasks averaged GloVe embeddings did better
  - CLS did not do much better than average pooling, and min or max pooling did not not do much better either
- Is there a better way?

Using BERT for sentence embeddings



- What about using a dual Encoder?
- For any two sentences A or B, feed them as a pair into a BERT model with a special head on top, output the similarity score; fine tune
- This performs well, but is very slow
  - If you have n sentences, you need to do $\binom{n}{2}$ computations
  - On 100 K sentences this takes 5 billion comparisons - many computations on a small data set
  - On a 10 K sentences, using the above idea takes 65 hours - infeasible

Using BERT for sentence embeddings



- What about a Siamese network?
  - I.e. for two sentences A and B feed them separately through the *same* BERT model and add a task head on top
  - The key: if we do this right, the embeddings u and v above can be computed beforehand and we can leverage fast search tools (FAISS) to do the similarity search
    - For any A (with vector u), we can find the nearest neighbor C, where C is a sentence whose embedding minimizes $\min_w u^\intercal w$
  - We don't have to feed pairs together in the BERT model
  - We can just use BERT as an embedding module

Using BERT for sentence embeddings

- Given the Siamese network above, for any two sentences A and B we get two embeddings u and v and then we can
  - Add a classification objective via softmax:
  $$softmax(W(u, v, |u - v|))$$
    - Here, two sentences can be 3 classes: entailment, contradiction, and neutral
    - We can get this data from SNLI datasets, and back propagate gradients as needed to modify BERT so it gives better embeddings
  - Add a regression objective via a similarity measure:
  $$||1 - cos(u, v)||^2$$
  - Optimize the Siamese classical objective:
  $$max(||u_a - u_p|| - ||u_a - u_n|| + \epsilon, 0)$$
    - Here, u_x is the embedding for some sentence x explicitly
    - a is the anchor sentence and p is the positive sentence while n is the negative sentence
    - The idea is we want the distance between p and a to be at least ϵ less than the distance between n and a

Training and Inference with Sentence BERT

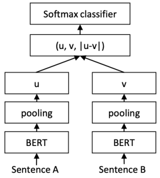


Figure 1: SBERT architecture with classification objective function, e.g., for fine-tuning on SNLI dataset. The two BERT networks have tied weights (siamese network structure).

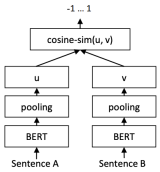
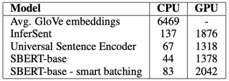


Figure 2: SBERT architecture at inference, for example, to compute similarity scores. This architecture is also used with the regression objective function.

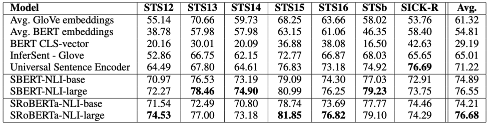
- Training
  - Take BERT or ROBERTA to start and fine-tune on SNLI / MultiNLI data sentence pairs data where each pair is classified as contradiction, entailment, and neutral
- Inference
  - Take sentences and run them through the fine-tuned model, cache embedding results
  - When you compare two sentences, evaluate the cosine similarity
  - If a sentence is new, run it through BERT

Using BERT for sentence embeddings



Model	CPU	GPU
Avg. GloVe embeddings	6469	-
InferSent	137	1876
Universal Sentence Encoder	67	1318
SBERT-base	44	1378
SBERT-base - smart batching	83	2042

Table 7: Computation speed (sentences per second) of sentence embedding methods. Higher is better.



Model	STS12	STS13	STS14	STS15	STS16	STSb	SICK-R	Avg.
Avg. GloVe embeddings	55.14	70.66	59.73	68.25	63.66	58.02	53.76	61.32
Avg. BERT embeddings	38.78	57.98	57.98	63.15	61.06	46.35	58.40	54.81
BERT CLS-vector	20.16	30.01	20.09	36.88	38.08	16.50	42.63	29.19
InferSent - Glove	52.86	66.75	62.15	72.77	66.87	68.03	65.65	65.01
Universal Sentence Encoder	64.49	67.80	64.61	76.83	73.18	74.92	**76.69**	71.22
SBERT-NLI-base	70.97	76.53	73.19	79.09	74.30	77.03	72.91	74.89
SBERT-NLI-large	72.27	**78.46**	**74.90**	80.99	76.25	**79.23**	73.75	76.55
SRoBERTa-NLI-base	71.54	72.49	70.80	78.74	73.69	77.77	74.46	74.21
SRoBERTa-NLI-large	**74.53**	77.00	73.18	**81.85**	**76.82**	79.10	74.29	**76.68**

Table 1: Spearman rank correlation ρ between the cosine similarity of sentence representations and the gold labels for various Textual Similarity (STS) tasks. Performance is reported by convention as $\rho \times 100$. STS12-STS16: SemEval 2012-2016, STSb: STSbenchmark, SICK-R: SICK relatedness dataset.

- Inference Quality: They did numerous studies and found that the output sentences significantly did better than BERT / ROBERTA and GLOVE across many tasks
- Inference Speed: Still slower than GLOVE lookups, but smart batching allows for very fast GPU processing in the worst case

References

- BERT paper
- Pinecone
- BERT pictures by Jay Allamar
- Good video on KD
- KD paper
- Roberta
- Distil BERT
- Sentence BERT
- BERT + BiLSTM KD