BERT: Contextual Representations for Better Text Understanding

CS 293S UCSB Tao Yang 2020

Outline

- Why BERT
- BERT Architecture
 - Self-attention computation
 - How to pre-train
- BERT Applications
 - Classification, question answering, ad-hoc search

Word Embeddings and Contextual Representations

Word embeddings are the basis of deep learning for text understanding and NLP

Word embeddings (word2vec, GloVe) are often *pre-trained* on text corpus from co-occurrence statistics

Problem: Word embeddings are applied in a context free manner The taste of this gaple is good $\longrightarrow [0.9, -.2, .6, ...]$

The taste of this <mark>apple</mark> is good This <mark>apple</mark> phone looks good

Solution: Train contextual representations on text corpus

[0.1, -.2, .6, ...] [0.3, .1, .3, ...]



3

BERT: Bidirectional Encoder Representations for Transformers

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova Google, Proc of NAACL 2019

- BERT base 12 layers (transformer blocks), 768hidden, 12 attention heads, and 110 million parameters.
- BERT Large 24 layers, 1024hidden, 16 attention heads and, 340 million parameters.
- BERT computation is expensive Takes many days to train

BERT architecture is a multi-layer bidirectional Transformer encoder.



The transformer layers of BERT

Also called encoder



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

More about Attention

"The Attention is All You Need" paper, A. Vaswani et al. NIPS 2017

Self-attention is the method the Transformer uses to detect how other words relevant to the current word being processed

Example: "The animal didn't cross the street because it was too tired"

What does "it" in this sentence refer to? self-attention allows it to associate "it" with "animal".



Transformer Computation for Self Attention

So for each input token, we create a Query vector, a Key vector, and a Value vector.



Transformer Computation for Self-Attention: Compute Query, Key, and Value Matrices

- For 2 tokens x_1 , x_2 , they form two rows of matrix X.
- Multiply with 3 weight matrices, we get query, key, value matrices Q, K, V
- Q contains q_1 , q_2 K contains k_1 , k_2 V contains v_1 , v_2



Example: Output of Self Attention



9

For 2 tokens x_1 , x_2 of dimension d_k , output z_1 , z_2

 z_1 represents interaction of q_1 with k_1 , and interaction of q_1 with k_2



Multi-headed Attention: Assume 8 Heads

 \bigcap

Repeat the same attention computation for different heads Finally concatenate them together multiplied by weight matrix

5) Concatenate the resulting Z matrices, 1) This is our 2) We embed 3) Split into 8 heads. 4) Calculate attention input sentence* each word* We multiply X or using the resulting then multiply with weight matrix W^o to R with weight matrices produce the output of the layer O/K/V matrices **W**₀Q X W₀ĸ Thinking WoV Machines Wo W₁Q V₁K * In all encoders other than #0, W₁V we don't need embedding. We start directly with the output of the encoder right below this one **₩**-2 N-ĸ W-∕V

Outline

- Why BERT
- BERT Architecture
 - Self-attention computation
 - How to pre-train



- BERT Applications
 - Classification, question answering, ad-hoc search

BERT Architecture and How to Pre-Train

- Scan books (800M words) and Wikipedia (2,500M words)
- Unsupervised training
 - Masked language model: Predicted a masked word in a sentence
 - "my dog is cute"
 - Next sentence prediction (NSP):
 Sentence A appears
 Sentence B
 - my dog is cute, he likes playing



Pre-training

How to pre-train BERT

Input token combines 3 embeddings.



Embedding dimension of each token: 768 for BERT base model. #input tokens≤512

How to Use BERT for Applications

- Start with a pre-trained BERT model
- Add some computation layer on top of the core BERT model
 - Fine-tune with your own dataset by converting application input into the specific format used for BERT model

Class

Example for sequence classification: Does a sentence contain personal attack?

• Final hidden state **[CLS]** is the fixeddimensional pooled representation of the input sequence, i.e. a sentence embedding capturing trained semantics





BERT for Question Answering

Given a question and a context paragraph, the model predicts a start and an end token from a reference paragraph that most likely answers the question.



Question: How many parameters does BERT-large have?

Reference Text: BERT-large is really big... it has 24 layers and an embedding size of 1,024, for a total of 340M parameters! Altogether it is 1.34GB, so expect it to take a couple minutes to download to your Colab instance.

BERT for Question Answering



BERT for Question Answering: Find the beginning and end of an answer



https://mccormickml.com/2020/03/10/questionanswering-with-a-fine-tuned-BERT/

BERT Performance for QA

- Standford Question Answering Dataset (SQuAD) is a collection of 100k crowdsourced question/answer pairs
- Given a question and a paragraph from Wikipedia containing the answer, predict the answer text span in the paragraph.

System	Dev		Test			
	EM	F1	EM	F1		
Leaderboard (Oct 8th, 2018)						
Human	-	-	82.3	91.2		
#1 Ensemble - nlnet	-	-	86.0	91.7		
#2 Ensemble - QANet	-	-	84.5	90.5		
#1 Single - nlnet	-	-	83.5	90.1		
#2 Single - QANet	-	-	82.5	89.3		
Published						
BiDAF+ELMo (Single)	-	85.8	-	-		
R.M. Reader (Single)	78.9	86.3	79.5	86.6		
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5		
Ours						
BERT _{BASE} (Single)	80.8	88.5	-	-		
BERT _{LARGE} (Single)	84.1	90.9	-	-		
BERT _{LARGE} (Ensemble)	85.8	91.8	-	-		
BERT _{LARGE} (Sgl.+TriviaQA)	84.2	91.1	85.1	91.8		
BERT _{LARGE} (Ens.+TriviaQA)	86.2	92.2	87.4	93.2		

SQuAD 1,1 results from Google 2019 BERT paper by Jacob Devlin et al.

Integrate BERT with KNRM and ConvKNRM for Document Ranking

n 1

[BCVGN] 0.4587

[BCVGN] 0.5259

CEDR:			Robust04		
Contextualized	Ranker	Input Representation	P@20	nDCG@20	
	BM25	n/a	0.3123	0.4140	
Embeddings for	SDM [13]	n/a	0.3749	0.4353	
Document	TREC-Best	n/a	0.4386	0.5030	
	ConvKNRM	GloVe	0.3349	0.3806	
Ranking, SIGIR	Vanilla BERT	BERT (fine-tuned)	[BC] 0.4042	[BC] 0.4541	
2019 by	PACRR	GloVe	0.3535	[C] 0.4043	
	PACRR	ELMo	[C] 0.3554	[C] 0.4101	
wacAvaney,	PACRR	BERT	[C] 0.3650	[C] 0.4200	
Yates Cohan	PACRR	BERT (fine-tuned)	[BCVG] 0.4492	[BCVG] 0.5135	
	CEDR-PACRR	BERT (fine-tuned)	[BCVG] 0.4559	[BCVG] 0.5150	
Gonarian	KNRM	GloVe	0.3408	0.3871	
Inject RERT	KNRM	ELMo	[C] 0.3517	[CG] 0.4089	
	KNRM	BERT	[BCG] 0.3817	[CG] 0.4318	
embeddings to	KNRM	BERT (fine-tuned)	[BCG] 0.4221	[BCVG] 0.4858	
KNPM and Conv	CEDR-KNRM	BERT (fine-tuned)	[BCVGN] 0.4667	[BCVGN] 0.5381	
	DRMM	GloVe	0.2892	0.3040	
KNRM	DRMM	ELMo	0.2867	0.3137	
	DRMM	BERT	0.2878	0.3194	
Rodustu4 is one	DRMM	BERT (fine-tuned)	[CG] 0.3641	[CG] 0.4135	

BERT (fine-tuned)

CEDR-DRMM

used in HW1



- Empirical results from BERT are very impressive for various NLP tasks
- With contextual pretraining in a big dataset, bigger == better
 - Good results on pre-training is >1,000x to 100,000 more expensive than supervised training.
- Challenges
 - Extremely expensive to run BERT
 - Active research for ad-hoc search queries