Introduction

In recent years, deep neural networks have been applied in various fields and achieved lots of the state-of-the-art (SOTA) performance. In the learning-to-rank (LTR) field, deep learning also breakthroughs lots of challenges. First I introduce the fundamental neural network architecture which is adopted by lots of SOTA natural language processing (NLP) models. It is BERT (Devlin et al., 2018) [1] architecture. Based on a public MS MARCO [2] benchmark and BERT architecture models, “Passage Re-ranking with BERT” (Nogueira and Cho, 2019) [3], “TFR-BERT” (Han et al., 2020) [4] and “HDCT” (Dai and Callan, 2020) [5] are picked for explanations and comparison.

BERT

The landscape of NLP has changed after BERT was introduced. BERT stands for Bidirectional Encoder Representations from Transformers. The inputs are a sequence of text and outputs are a sequence of vectors. To achieve it, BERT leverages transformer architecture (Vaswani et al., 2017) [6] to learn the text representations. It allows every token to “see” the rest of tokens in order to calculate the contextual token embeddings. In other words, transformer architecture simply takes a weighted average over all the rest of the token's vectors. The following sections will cover detailed explanations.

The first step is breaking down a sequence of text into a sequence of tokens. To let BERT handle a variety of down-stream tasks, Devlin et al. (2018) split a single word into multiple subwords. It allows BERT to handle unseen words, proper nouns (e.g. company name), shorthands (e.g. emoji) and misspelling text. After that a reversed token needed to be added. The first token is “[CLS]” while “[SEP]” token is added into separate different segments and the last token.

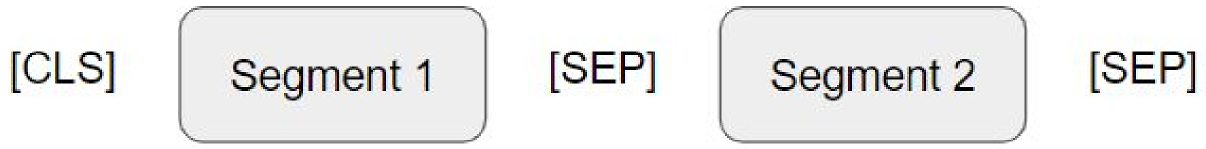

Figure 1: Example of BERT’s text transformation.

Later on, 3 embeddings will be generated to compute the text representations for model training. They are token embeddings, segment embeddings and position embeddings. Token embeddings refer to vectors representing the token itself. Segment embeddings indicate whether it is first segment or second segment. Finally, position embeddings mean the position of the token in this input.

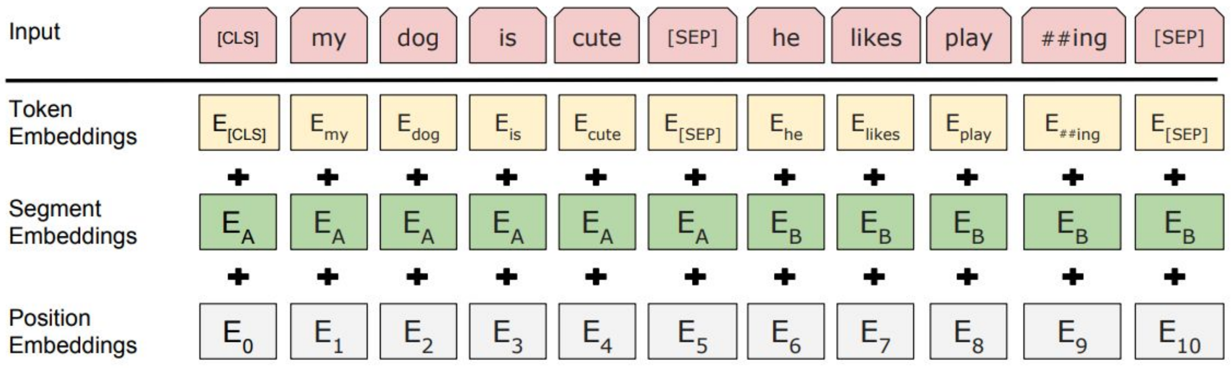
Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8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

Figure 2: Example of BERT's embeddings (Devlin et al., 2018)

The fundamental concept of transformer architecture is the self-attention operation. Self-attention is a sequence of vector inputs and computing a series of vector outputs. To produce every vector output, the corresponding input weighted average over all the rest of the token's vectors. It is a breakthrough when comparing to word2vec (Mikolov et al., 2013) [7] and Long-short-term-memory (Hochreiter and Schmidhuber, 1995) [8]. Expect the maximum token limitation (it is 512 in general), every token can reach out to the rest of tokens. word2vec is designed to reach out a predefined number of surrounding words. On the other hand, LSTM has a vanishing gradient problem which leads word cannot reach out far away words.

Passage Re-ranking with BERT

Nogueira and Cho introceuded Passage Re-ranking with BERT in 2019. They used BERT to convert text to vectors for model training. Nogueira and Cho used the same notation by Devlin et al. (2018). They treat query and passage as first segment and second segment respectively. As mentioned in the previous section, BERT has a limitation on long sequence inputs, input text will be truncated. Query text will be truncated if the number of token exceeds 64. Passage text will be truncated too if separator tokens (i.e. [CLS] and [SEP]), query text and passatge text exceed 512 tokens.

As shown in the figure 3, query text and passage text are surrounded by "[CLS]" and "[SEP]" which are reserved tokens mentioned in the previous section.

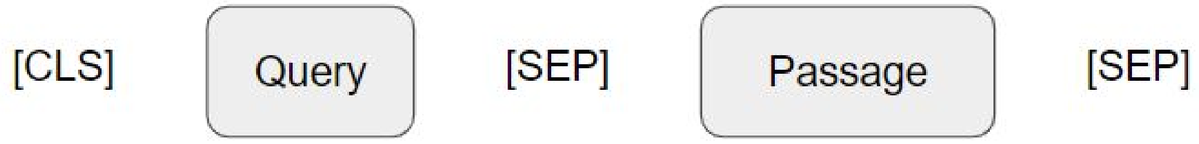

Figure 3: Example of BERT's text transformation in LTR

Vectors become model input and estimate a score of how relevant a passage is to query. In other words, it is a pointwise approach.

TFR-BERT

Han et al. (2020) proposed TFR-BERT (TensorFlow Ranking - BERT) to tackle LTR problem. A generic framework which leverages BERT representations to rank documents according to query. TFR-BERT (Han et al., 2020) can be divided into text representation and scoring stages.

Same as setup in Passage Re-ranking with BERT (Nogueira and Cho, 2019), they treat query text as first segment while passage text is second segment (figure 3). Passage text will be truncated if separator tokens (i.e. [CLS] and [SEP]), query text and passatge text exceed 512 tokens.

By design of Han et al. (2020), this framework supports pointwise, pairwise and listwise approaches. The interesting part is that they ensembled three approaches to build a ranking model and achieving #24 best performance in a public MS MARCO benchmark for the re-ranking task as of October 4, 2020.

HDCT

Dai and Callan proposed Context-aware Hierarchical Document Term (HDCT) in 2019. Unlike previous models, HDCT leaveges BERT to learn the importance of term weight in passage. After that HDCT computer the term weight per document and persist into inverted indexing.

First of all, a document will be splitted into passages and fitting passages into the BERT model to predict term importance. To train the BERT model, Dai and Callan came up with 3 approaches to build a training label for the BERT model training. They are context-based weak-supervision strategy, relevance-based supervision strategy and pseudo-relevant feedback based weak supervision strategy. All of the labels are a pair of word and term weights. As adopting the BERT model, there is limitation of input tokens. According to different studies, 300 words is the optimal number of words to represent a text. Therefore, Dai and Callan decided to truncate the passage text to 300 words. Later on, all passage’ term importances under the same document will be aggregated and generate an inverted indexing for query.

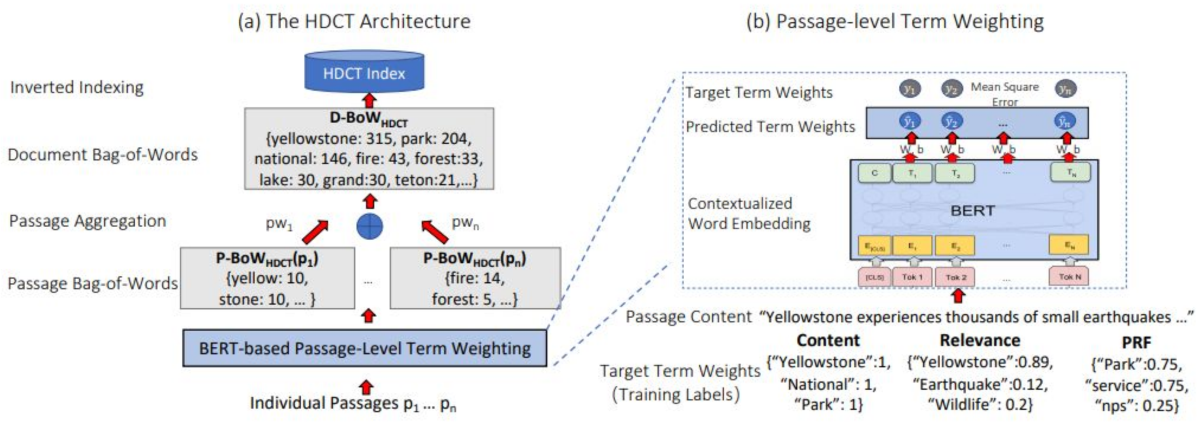
(a) The HDCT Architecture
(b) Passage-level Term Weighting
Inverted Indexing
HDCT Index
Document Bag-of-Words
D-BoW HDCT {yellowstone: 315, park: 204, national: 146, fire: 43, forest:33, lake: 30, grand:30, teton:21,...}
Passage Aggregation
pw1
pwn
Passage Bag-of-Words
P-BoW HDCT(p1) {yellow: 10, stone: 10, ... }
P-BoW HDCT(pn) {fire: 14, forest: 5, ...}
BERT-based Passage-Level Term Weighting
Individual Passages p1 ... pn
Target Term Weights
Mean Square Error
Predicted Term Weights
Contextualized Word Embedding
BERT
Passage Content "Yellowstone experiences thousands of small earthquakes ..."
Content
Relevance
PRF
Target Term Weights (Training Labels)
{"Yellowstone":1, "National": 1, "Park": 1}
{"Yellowstone":0.89, "Earthquake":0.12, "Wildlife": 0.2}
{"Park":0.75, "service":0.75, "nps": 0.25}

Figure 4: The HDCT architecture (Dai and Callen, 2019)

Comparison

Although aforementioned algorithms are built on top of the BERT model, they use different setup and approaches. Basically, "Passage Re-ranking with BERT" and "TFR-BERT" is quite similar, they use the BERT model to predict relevant probability directly while HDCT leverages the BERT model to compute the term weight and stored in inverted indexing for query matching.

	Passage Re-ranking with BERT	TFR-BERT	HDCT
Approach	Pointwise	Pointwise, pairwise and listwise	Pointwise
Input	Concatenate query and passage text	Concatenate query and passage text	Only passage text
Maximum number of query token	64	N/A	N/A
Maximum number of total token	512	512	512
Label	Binary classification (Probability of relevant)	Regression prediction (Probability of relevant)	Term weight

Table 1: Comparison among algorithms

Conclusion

BERT is a breakthrough in the field of NLP. It is just a starting point and practitioners are evaluating the BERT variations such as RoBERTa [9], ELECTRA [10]. I noticed there is a new state-of-the-art model that achieves a better result in MS MARCO benchmark every week.

Han et al. (2020) leverages the "TFR-BERT" framework to build ensemble models to achieve better results. They not only ensemble neural network models (e.g. RoBERTa, ELECTRA) but also approaches (e.g. pointwise, listwise). I foresee that one of the trends is focusing on enquiring a larger data set, tuning parameters and ensemble model complexity models.

Rank	Model	Submission Date	MRR@100 On Eval
1	ColBERT MaxP end-to-end Omar Khattab, Christopher Potts, and Matei Zaharia - Stanford University	Oct 6, 2020	0.384
2	HDCT top100 + BERT-base FirstP (single) Luyu Gao and Zhuyun Dai - LTI, CMU	Sep 9, 2020	0.382
3	BERT-m1 base + classic IR + doc2query Leonid Boytsov - Bosch Center for A	Oct 1, 2020	0.380

Table 2: Top 3 in MS MARCO Full Ranking (as of Oct 6, 2020)

References

1. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.
2. Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. Ms marco: A human generated machine reading comprehension dataset. arXiv preprint arXiv:1611.09268, 2016.
3. Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with BERT. arXiv preprint arXiv:1901.04085, 2019.
4. Shuguang Han, Xuanhui Wang, Michael Bendersky and Marc Najork. Learning-to-rank with BERT in TF-Ranking. arXiv preprint arXiv: 2004.08476, 2020
5. Zhuyun Dai and Jamie Callan. Context-Aware Document Term Weighting for Ad-Hoc Search. WWW' 20. 2020
6. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. arXiv preprint arXiv:1706.03762, 2017.

7. Tomas Mikolov, Greg Corrado, Kai Chen & Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. arXiv preprint arXiv:1301.3781, 2013.
8. Sepp Hochreiter and Jurgen Schmidhuber. Long Short Term Memory. Technical Report FKI-207-95, Technische Universitat Munchen, Munchen, 1995.
9. Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv preprint arXiv: 1907.11692, 2020.
10. Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. arXiv preprint arXiv: 2003.10555, 2020.