Learning of BERT and MT-DNN

BERT

the first paper is BERT. At the first of this paper, the author introduces the BERT, which is the State-of-the-Art in all NLU tasks. The most special of BERT is that it focuses on the contextual information in each layer.

After that the author discusses two model (EMLO and OpenAI GPT), both of them have pre-training and fine-tuning two stages, but at the pre-training stages all of them can't attention the context-before and context-after at the same time, although EMLO combines the left-to -right and right-to-left training result.

The Model of BERT like OpenAI GPT, while it uses the left and right information in a sentence.

INPUT:

The input of BERT include three part(Token Embedding , Segment Embedding, Position Embedding) of a sentence. A sentence always begins with [CLS] and end with [SEP], What's more if there are two sentences, we connect the sentences with [SEP].

Pre-training Tasks

Pre-training include two tasks: Masked LM and Next Sentence Prediction.

Masked LM:

The author masks 15% of all WordPiece tokens in each sequence at random. Due to the word mask hardly be seen in the fine-tuning, so the author not always uses the word[mask] to replace masked words. The model uses the following produce:

80% of the time: Replace the word with the [MASK] token

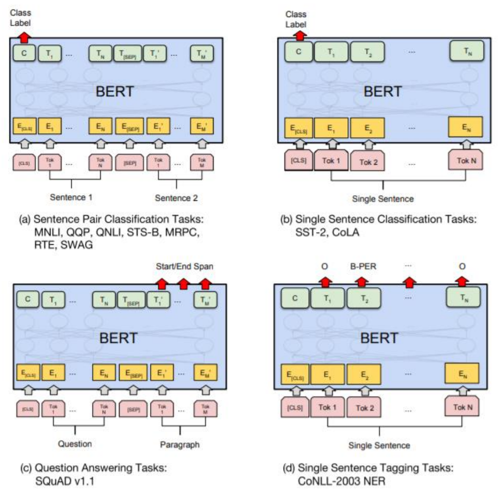
10% of the time: Replace the word with a random word

10% of the time: Keep the word unchanged

Next Sentence Prediction:

This task is pay an attention to some tasks in really life like QA and NLI. Author uses this task to help model understand the relationship of two sentence. Author choose some a pair of sentences (A and B), 50% of the time B is the actual next sentence of A, while others are the random sentences. Author trains the pre-training model to predict weather B is the next sentence of A. What's more the author also chooses the not next sentences at random.

In the following part the author also introduces some detail about pre-training and fine-tuning. And how to use BERT in the main tasks of NLP.


Class Label
BERT
Sentence 1
Sentence 2
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG
Class Label
BERT
Single Sentence
(b) Single Sentence Classification Tasks: SST-2, CoLA
Start/End Span
BERT
Question
Paragraph
(c) Question Answering Tasks: SQuAD v1.1
O
B-PER
O
BERT
Single Sentence
(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

At the last the Author use his model and achieve the best result of all NLP tasks. The Author also show some details of experiment and do some controlled experiments.

MT-DNN.

In the paper the Author think MTL need less data and leads to more general representations to help new tasks and domain. What's more BERT performs outstanding in all tasks of NLP, So Author want to combine the MTL and BERT. He uses BERT and MTL to pre-train and fine-tune respectively and gets a good result. The MT-DNN can share some params in different tasks, rather than BERT which need fine-tune independently in different tasks.

Author first introduce the MTL which originate from the idea that "people often apply the knowledge learned from previous tasks to help learn a new task and can make use of the data more effective" .Then the Author introduce the tasks of this model.

MT-DNN Model

MT-DNN have two big layers. In the lower layers are shared layers, the higher layers are tasks special layers. At the following I will tell this model from low layer to high layer. The input X is a sentence or a pair of sentences, like BERT's input, which starts with [CLS] and end with [SEP] . it used [SEP] divide two sentences. Then the lexicon encoder maps X into a sequence of input embedding vectors, one for each token, constructed by summing the corresponding word, segment, and positional embeddings. After that we use the output of lexicon encoder as the input of Transformer encode. The output of this layer is l2 which are contextual embedding vectors. The output L2 is shared by different tasks.

output:

Single-Sentence Classification Output: x is the embedding of the token [CLS] WSST is the task-specific parameter matrix.

$$\mathbf{Pr(c|X) = softmax(\boldsymbol{W}_{\boldsymbol{SST}}^{\boldsymbol{T}}.x)}$$

Text Similarity Output: g is sigmoid function. x is the embedding of the token [CLS]. Wsst is the task-special vector. Pairwise Text Classification Output: the Author uses SAN model.

$$\mathbf{Sim(X1, X2) = g(\boldsymbol{W}_{\boldsymbol{SST}}.x)}$$

Relevance Ranking Output: use QNLI task as example,

$$\mathbf{Rel(Q, A) = g(\boldsymbol{w}_{\boldsymbol{QNLI}}^{\boldsymbol{T}}.x)},$$

x is the output of [CLS], W QNLI also is a special vector.

Training Produce

Like others MT-DNN training produce include two stages: pre-training fine-tuning. For pre-training, Author trains model by two tasks: ML and next sentence prediction.

Algorithm:

Algorithm 1: Training a MT-DNN model.

```
Initialize model parameters $\Theta$ randomly.
Pre-train the shared layers (i.e., the lexicon
  encoder and the transformer encoder).
Set the max number of epoch: $epoch_{max}$.
//Prepare the data for $T$ tasks.
for $t$ in $1, 2, ..., T$ do
    Pack the dataset $t$ into mini-batch: $D_t$.
end
for $epoch$ in $1, 2, ..., epoch_{max}$ do
    1. Merge all the datasets:
      $D = D_1 \cup D_2 ... \cup D_T$
    2. Shuffle $D$
    for $b_t$ in $D$ do
        //$b_t$ is a mini-batch of task $t$.
        3. Compute loss : $L(\Theta)$
           $L(\Theta)$ = Eq. 6 for classification
           $L(\Theta)$ = Eq. 7 for regression
           $L(\Theta)$ = Eq. 8 for ranking
        4. Compute gradient: $\nabla(\Theta)$
        5. Update model: $\Theta = \Theta - \epsilon\nabla(\Theta)$
    end
end
```

Loss funcitons:

classification tasks:

$$-\sum_{c} \mathbb{1}(X, c) \log(P_r(c|X)), \qquad (6)$$

text similarity tasks:

$$(y - \text{Sim}(X_1, X_2))^2, \qquad (7)$$

relevance ranking tasks:

$$-\sum_{(Q,A^+)} P_r(A^+|Q), \qquad (8)$$

$$P_r(A^+|Q) = \frac{\exp(\gamma \text{Rel}(Q, A^+))}{\sum_{A' \in A} \exp(\gamma \text{Rel}(Q, A'))}, \qquad (9)$$

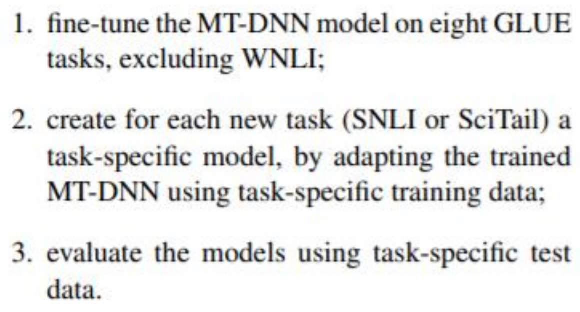
Experiments:

MT-DNN is based on the PyTorch implementation of BERT, what's more the optimizer is Adamax , the learning is 5e-5 , a batch size is 32 and the max size of epochs is 5. The Author use the most famous datasets: GLUE, SNLI, and SciTail. GLUE is used to demonstrate the effective of fine-tuning about MT-DNN. Domain adaptation using SNLI and SciTail.

The MT-DNN get the best result in all NLU tasks exception WNLI base on GLUE datasets. The Author also think that the MT-DNN model is benefit from its flexible modeling framework. Such as the SAN and d the pairwise ranking loss.

MT-DNN also get a good result in domain adaption. Author demonstrate the domain adaption of MT-DNN on NLI task basing on SNLI and Scitail databases. The Author prove that MT-DNN preform better than BERT when we only have a small data set.

Produces:



1. fine-tune the MT-DNN model on eight GLUE tasks, excluding WNLI;
2. create for each new task (SNLI or SciTail) a task-specific model, by adapting the trained MT-DNN using task-specific training data;
3. evaluate the models using task-specific test data.

Above all, during the process of reading those two paper, I have learned many NLP knowledge, at the beginning I spend some time to read the BERT, I found it is difficult to understand it, even though I know each word in the paper. So I decided to spend some time to learn the leading knowledge of NLP. Though I think I don’t understand all details of those knowledge like self-attention, seq-seq, even the BERT. I think I need read the open-source code to understand it. I also learn some things from the two papers. During learning those model: LSTM, Transform, ELMO, OpenAI-GPT,BERT,MT-DNN. Firstly, I found that the breakthrough of NLP hardly current in recent years, when I am learning in college. I never found that the change of a filed is so close to me, which is so exciting. Secondly, I learned that we should combine the strengths of others model, for example the ELMO extract the before and after information of a word, while it uses LSTM to extract it. The OpenAl-GPT uses Transformer to extract the information, while it only extract the context-before information. The BERT combine the strengths of each model.