JIAHE GENG* and YANAN MAO*

A Knowledge-Enhanced Pre-trained Language Model for Medical Text Mining

Pre-trained models are an eye-catching area in language text mining. Due to the fact that applying the natural language processing model cannot be directly used in the biomedical field, we aim to propose a pre-trained language model to utilize the knowledge from the MTL-bioinformatics knowledge base. In general, the system promised high accuracy on words but cost enormous time on training the model.

Additional Key Words and Phrases: MML, RoBERTa

ACM Reference Format:
Jiahe Geng and Yanan Mao. 2018. A Knowledge-Enhanced Pre-trained Language Model for Medical Text Mining. *Proc. ACM Meas. Anal. Comput. Syst.* 37, 4, Article 111 (August 2018), 9 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Transfer learning [2] has a long history of being used to address new issues with a small number of samples rather than training models from the start with large amounts of data. Since neural network models do not require manual labeling or statistics, but instead automatically learn low-level numerals or continuous vectors as task features, a series of attempts in the NLP field have been made, combined with a pre-training model and fine-tuning procedure.

Transfer learning study aims to use previously gained knowledge to address new challenges and possibly improve results. More crucially, transfer learning allows you to learn from a variety of sources and then apply that information to the target job. Although the data fields and task settings of the source and target tasks may differ, the knowledge necessary to process these tasks, in the same way, is the same.

Experts make full use of large-scale unlabeled corpus to give multidimensional language knowledge for NLP tasks after realizing the potential of neural networks and transfer learning[3]. As a bridge between the source and target tasks, various pre-training strategies have been developed. To be more specific, these models are pre-trained on data from a variety of different sources. Precode the knowledge and then utilize it to train the target task model with it. We take use of the chance to investigate pre-training models (PTMs) in the era of deep learning and gain a thorough understanding of the models.

If we had to sum up what pretraining does in one statement, I believe it would be "use as much training data as possible to extract as many common features as possible so that the model is not burdened with learning for a specific job." Because annotated resources are limited, but unannotated resources are plentiful. Because there is so little relevant training data for a given task, the model is unable to learn and synthesize useful rules from it.

*Both authors contributed equally to this research.

Authors' address: Jiahe Geng; Yanan Mao, Carleton University.

2476-1249/2018/8-ART111 $15.00
https://doi.org/XXXXXXX.XXXXXXX

The focus of the early stages of research is on pre-training shallow layers of networks to capture the semantics of words, such as Word2Vec. However, because each word is represented by only one vector, expressive polysemy in diverse contexts is severely limited. For example, "spring" might refer to the act of leaping; it could also refer to a location where a torrent of water emerges from the earth; or it could refer to a season of the year. The term "spring" has a very different connotation from the word "vector."

Until the transformer is introduced, the model cannot be fully trained for NLP tasks. Transtransformeformer is quickly becoming the standard structure for natural language interpretation and creation due to its exceptional characteristics. A transformer is a feature extractor that consists of a set of encoding components (Encode) and decoding components (Decode). The encoder converts the input sequence to a single potential vector that represents the entire sequence, which is then passed to the decoder, who converts it to the desired target sequence. This model has two key features: position embedding and a multi-head self-attention mechanism.

Later, it was a turning point in the development of the pre-training model's backbone neural structure, BERT, and GPT. GPT excels in natural language creation, whereas BERT excels at natural language comprehension. Following BERT, other better models based on them, such as RoBERTa [4]and ALBERT, were proposed. We chose RoBERTa as our model because of the following significant enhancements:

- Larger number of model parameters, larger batch size, more training data.
- Remove the Next Sentence Prediction(NSP) task. The NSP must have two sentences for input, but for 512-dimensional data, most of the two sentences cannot be filled, resulting in a large amount of data space waste.
- Dynamic MASK: Random MASK is performed every time a training example is fed to the model.
- Text encoding (character level encoding): BERT: Character-level BPE Vocabulary of Size 30K RoBERTa: Byte Characters 50K.

The purpose of a language model is to calculate the likelihood of a word sequence. RoBERTa's pre-training strategy is the Masked Language Model (MLM). MLM conceals some tokens from the original sequence and then allows the model to predict these masked tokens, as the name implies.

Thus, based on our understanding, the pre-training language model is a multi-layer network structure that is pre-trained to initialize the downstream task model's multi-layer network structure, which can learn both shallow and deep knowledge. Pre-trained language models are a type of dynamic text representation that adapts the text representation to the current situation. The modified text representation can better reflect the word's specific meaning in context and successfully address the polysemy problem.

2 METHODOLOGY

2.1 BERT

The main structure of BERT, which is based on transformer architecture, is a bi-direction depth transformer. The structure consists of a stack of multi-layer transformers with two different stages for pre-training and fine-tuning BERT to specific activities. According to the paper[1], the BERT, large-scale PTMs with hundreds of millions of parameters are capable of capturing polysemic disambiguation. When the amount of PTMs grows larger, lexical and syntactic structures, as well as factual knowledge from the text, are affected. The enormous PTMs' language knowledge is brought to great performance in downstream NLP tasks by find-tunning the big-scale PTMs using a large number of samples.

Convolutional neural networks or recursive neural networks are often used sequence models, and the best model also includes an encoder-decoder framework and an attention mechanism. As a result, a new model transformer based on the attention mechanism is offered, replacing the existing model structure.

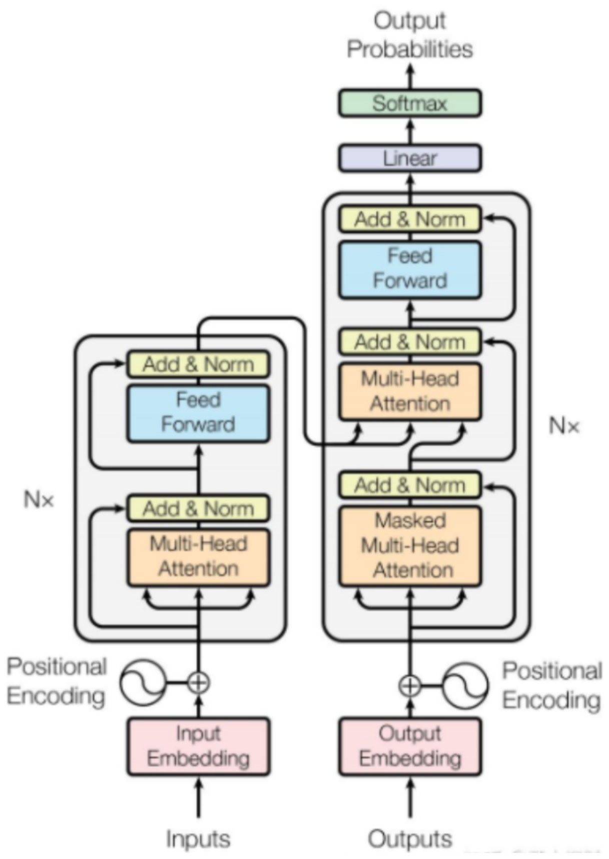

Fig. 1. The Structure of BERT

2.1.1 Multi-head attention. The input from each head is passed to the attention mechanism via multi-head attention. The embedding vector for words has been segmented. Each attention mechanism can balance the probable deviation of the exact attention mechanism by optimizing the varied properties of each word. As a result, the term's meaning can have a wider range of expression.

The self-attention mechanism does not adopt the sequence structure of RNN so that the model can be trained in parallel and have global information. Multi-head attention mechanism can capture various multidimensional correlation coefficients between words.

2.1.2 Feed-forward network. A feed-forward network has two linear layers and is fully connected. The model's ability can be improved by learning more nonlinear features.

$$PE(pos, 2i) = sin(\frac{pos}{1000^{2i/d_{model}}})$$
$$PE(pos, 2i+1) = cos(\frac{pos}{1000^{2i/d_{model}}})$$

The token position number in the sequence is labeled using positional encoding.

The pos represente the position number, and the value range is the integers between 0 to the max queue length minus 1.

d_{model} is the dimension of the position vector has the same value as the hidden state dimension of the entire model.

PE is a matrix where the number of rows is the maximum sequence length and the number of columns in d_{model}. The value of PE is a scalar.

Because the Transformer encoder structure lacks word position information processing, a position encoder must be added after the Embedding layer. Even though the vectors will be nonsensical, this is plainly a disadvantage. The ability to scale the absolute positions of natural numbers to smaller values, allowing for faster convergence during following gradient descent. And create the periodic function so that the period may be utilised to show the distance between two words.

The vector of $PE(pos + k, 2i + 1)$, for example, can be written as a linear representation of $PE(pos, 2i + 1)$. $PE(1, 2i + 1)$ is fixed due to its periodicity. $PE(k, 2i + 1)$ can be inferred as follows.

$$sin(\alpha + \beta) = sin\alpha cos\beta + con\alpha sin\beta$$
$$con(\alpha + \beta) = con\alpha cos\beta - sin\alpha sin\beta$$

And generate position values:

$$PE(pos + k, 2i) = PE(pos, 2i) * PE(k, 2i + 1) + PE(pos, 2i + 1) * PE(k, 2i)$$
$$PE(pos + k, 2i + 1) = PE(pos, 2i + 1) * PE(k, 2i + 1) - PE(pos, 2i) * PE(k, 2i)$$

2.1.3 Masks. We introduce two types of masks:

- Sequence mask to prevent decoder from seeing future messages.
- Padding mask reset the value 0 when dealing with pad for attention.

At the end of the training each epoch, the attention mask which reflects the position of the mask, will be returned.

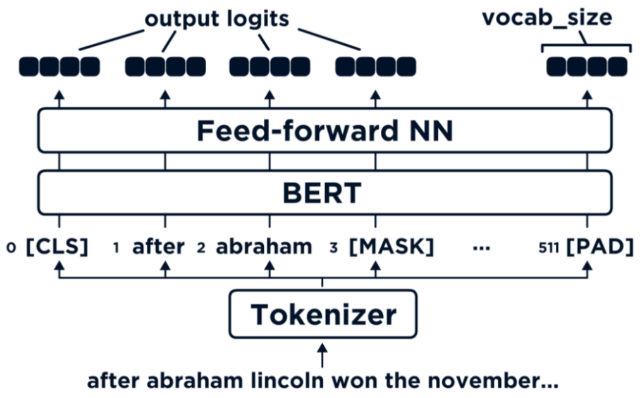

Fig. 2. The Tokens

2.1.4 Layer Normalization. The mean value is 0 and the standard deviation is 1 regardless of the number of features. The objective is to avoid overfitting. All deep networks require this network layer as a minimum. The value estimated over numerous layers may be excessively big or too small as the number of network layers rises, resulting in aberrant learning processes and poor model convergence. To keep the feature value within a suitable range, a normalisation layer is required.

2.1.5 Decoder Layer. Each decoder layer, as a component unit of the decoder, extracts features based on the input to the desired direction, i.e. the decoding process. This section contains a number of decoders.

3 ROBERTA

Based on BERT, after modifying it with a bigger dataset, more arguments, and a larger batch, Roberta is proposed in paper[1].

3.1 The Dynamic Mask

Roberta copied 10 copies of the pre-trained data and randomly selected 15% of Tokens for each mask. That is, there are 10 different ways to mask the same sentence. Each data is then trained with N/10 epochs. This means that in the training of these N epochs, the tokens masked by each sequence will change.

BERT randomly selected 15% of tokens to be replaced. To eliminate the mismatch between upstream and downstream tasks, the 15% tokens were :(1) replaced with [MASK] 80% of the time, (2) remained unchanged 10% of the time, and (3) replaced with other words 10% of the time.

3.2 Delete the NSP Task

RoBERTa did away with NSP tasks in favor of full-sentence training. Enter a string of sentences up to 512 characters long at a time.

The NSP task is used by BERT to detect if two sentences are sequential. The training data is made up of 50 percent samples that are the following sentence of the same article and 50 percent samples that are two sentences from separate publications.

It has been discovered that reducing NSP loss can equal or slightly improve downstream task performance when compared to the original BERT. This could be due to Bert's use of a single sentence as a unit, preventing the model from learning remote dependencies between words, whereas RoBERTa used multiple consecutive sentences, allowing the model to capture longer dependencies, making it more suitable for downstream tasks involving long sequences.

3.3 Bigger mini-batch, more data, longer training time

- The batch size used by RoBERTa is 8K, and that used by BERT is 256.
- RoBERTa used 160G of data, while BERT used 13G.

The purpose of this is that obviously more training data increases the diversity of the data (vocabulary, syntactic structure, grammatical structure, etc.) and certainly improves model performance

3.4 AdamW

Regularization terms (L2 and L1 regularization) are frequently used in classical SGD to boost the model's generalization capabilities. The gradient result of the regularization term will be added to the gradient computation after the insertion of the L2 regularization term. As a result, if the parameter is huge, the related gradient will be large as well.

However, in Adam's calculation, the minuend would be divided by the gradient square accumulation, resulting in a smaller removed term, preventing Adam from punishing the term with excessive weight. With the same coefficient, weight attenuation updates ownership weight, and the bigger the weight, the greater the penalty. In adamW, the proper method for Weight Decay was added. To ensure that the parameter with too large a parameter value can decay faster, the weight matrix, learning rate, gradient, and the parameter value of the final moment are all added. As a result, AdamW is a popular approach in the hugging face transformer version, which is the one we employed.

4 ACTUAL WORK

4.1 Tools

There are a variety of Pre-training Transformer models, all of which have open-source code, but their implementations are different and it can be difficult to compare them. Huggingface Transformer helps us keep track of popular new models and provides a unified code style to work with BERT, XLNet, GPT, and more. Moreover, it has a model warehouse, where all common pre-training models and fine-tuning models for different tasks can be downloaded conveniently.

Transformers are designed the following ways to ease of use:

- There are only three main classes: Configuration, Models, and Tokenizer.
- All models are loaded using the uniform From_pre-trained function, and the Transformers handle loading, caching, and all other details related to loading the models. And all these Models are managed centrally by Hugging Face Models.
- Based on the three classes above, provide a higher level of pipeline and Trainer or TFTrainer to predict and fine-tune the model with less code.
- Instead of a basic neural network library to build Transformer step by step, it encapsulates common Transformer models into a building block that can be easily used in PyTorch or TensorFlow.

Such as the Configuration (Config) class of Robertaconfig, which holds the relevant (super) parameters of the model. We don't usually have to construct it ourselves. If we do not need to make changes to the model, the configuration will be used automatically when we create the model.

The Tokenizer class of RobertaTokenizer, which holds information such as dictionaries and implements the ability to turn strings into sequences of ids. Tokenizer basically splits words and turns them into integer ids, although some models use subwords. But anyway, the ultimate goal is to turn a piece of text into a sequence of ids. Of course, it must also be able to turn the ID sequence into text.

4.2 Database

We employ the dataset from hugging face, named "medical_questions_pairs". This dataset consists of 3048 similar and dissimilar medical question pairs labeled by Curai's doctors. Each question is shown as one similar and one different pair. The questions are rewritten in different ways and restructure the syntax but maintain the same keywords.

4.3 Evaluation

Changing the number of training epochs allows us to investigate the impact of training data on the model. The more input data there is, the more accurate the model will be, but the time it takes to train the model will also be longer.

It should be noted that our LOSS calculation only calculates the [MASK] part, and the above rules can guarantee robustness. The accuracy on BERT comes from paper[5]. The sharp jitter of the convergence curve is probably due to random mask, but the convergence speed is affected by the number of preset functions and variables.

Table 1. Accuracies on Dataset

Model	1 epoch	2 epoch
RoBERTa	40.53%	51.22%
BERT	average level: 82% +	

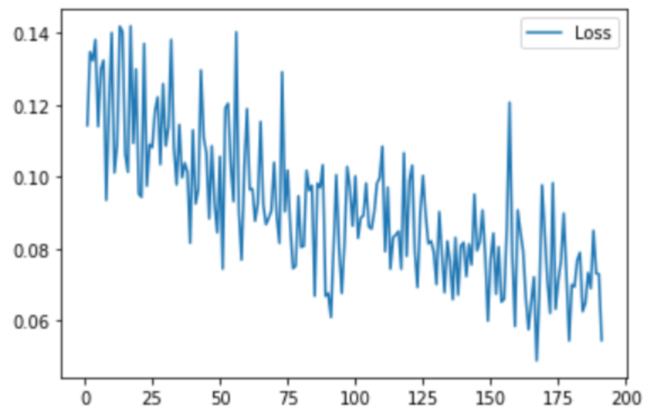

Fig. 3. Loss

Table 2. Run Time on Dataset

Model	1 epoch	2 epoch
RoBERTa	2.30 h and up	7.30 h and up
BERT	average level: 23 days	

5 CONCLUSION

We propose a pre-train language model for Medical Text Mining. The project aims to build a workable model that would return high-similar words in the medical field and figure out the sentences that share the same meanings.

Following the description of our initial hypothesis and digging deeper into the Roberta model, we finally present the source code and the paper of this project.

We introduce Roberta, an MML-based language representation model for pre-processed biomedical text mining, in this research. As a result, our model is far from the upper limit of data processing, and even 15 hours of data training falls far short of the over-fitting norm. We tried to change our minds, reducing the model's training depth and the number of layers in the neural network. The number of layers cannot be modified because the present model is directly cited. We try to simply set up the LSTM model to reduce the required training amount, but we have a feeling that the model's accurate rate has decreased, which isn't useful for improving the model.

The LSTM model, on the other hand, works with the default database: classification of film reviews as good or terrible, with an accuracy rate of over 70%. We only maintain the discussion about the conclusion, not the code, because it isn't directly related to this project.

As a result, we've come to the following conclusions on the impact of data sets and data similarity on pre-processing model correctness. The database used in this project, the required training time, and the preliminary training estimate in days can all help to improve the model's accuracy. Although MML reduces the time required for data regression from a random standpoint, existing pre-training language models have too many parameters, a high model complexity, a hefty model, a high training cost, and a higher black box degree.

6 ACKNOWLEDGEMENT

The project is completed based on the hard work of this team. We have learned a lot, not only from the free resources and papers we have found on the internet, but learned from each other. Effective communication significantly shortened the time of discussion within the group, enabling us to fully improve and reflect on the system we created in this project.

And further thanks to professor Ahmed EI-Roby, who led us to start this project.

The source code could be found: https://github.com/Ymao70115/COMP5118_FInal_Project

REFERENCES

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Proceedings of NeurIPS, pages 5998– 6008.

[2] Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. IEEE TKDE, 22(10):1345–1359.

[3] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. Science China Technological Sciences, 63:1872—- 1897.

[4] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020d. Roberta: A robustly optimized bert pretraining approach. In Proceedings of ICLR.

[5] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, Jaewoo Kang, BioBERT: a pre-trained biomedical language representation model for biomedical text mining, Bioinformatics, Volume 36, Issue 4, 15 February 2020, Pages 1234–1240, https://doi.org/10.1093/bioinformatics/btz682