FinMegatron: Large Financial Domain Language Models

Xianchao Wu
NVIDIA
xianchaow@nvidia.com, wuxianchao@gmail.com

Abstract

General domain pretrained large-scale language models, such as BERT and GPT3, have achieved state-of-the-art results among numerous NLP classification and generation applications. This pretraining technology is also willing to be used in vertical domains, such as finance. The downstream applications include financial event extraction from news, summarization, and causal inferencing. In this paper, we propose large-scale pretrained BERT models for financial domain in English and Japanese languages. The original datasets come from professional financial news. We empirically study the factors of sub-word vocabulary set, model size and their impacts to the downstream financial NLP applications. The code and pretrained models are released from https://github.com/NVIDIA/Megatron-LM.

1 Introduction

Large-scale pretrained contextual language models (Qiu et al., 2020), such as ELMo (Peters et al., 2018), BERT (Devlin et al., 2019), GPT (Radford et al., 2018), ALBERT (Lan et al., 2020), Megatron (Shoeybi et al., 2019), and GPT3 (Brown et al., 2020), have achieved outstanding results in numerous classification and generation applications of NLP field. Leveraging the encoder blocks of *transformers* (Vaswani et al., 2017), BERT (Devlin et al., 2019) proposed the prediction of masked words or subwords and next sentences. These pretrained models only require large-scale textual documents as inputs without any additional manual annotations. Parameters include the number of heads used in multi-head self-attentions and the hidden size of point-wise feed-forward networks used in transformer's encoder blocks. Comparing with this non-autoregressive framework of utilizing the encoder blocks, GPT models employ the decoder blocks of transformers. That is, a sentence is taken as input and the model is trained by predicting its word step-by-step, one-by-one. In order to predict the next word, mask tensors are constructed by limiting the usage of left-hand-side visible words in a sentence. A list of NLP applications, such as machine translation (Stahlberg, 2020), machine reading comprehension (Zhang et al., 2020), document summarization (El-Kassas et al., 2021), sentiment analysis (Zhang et al., 2018), have achieved significantly better results in recent years, thank to these large-scale pretrained models (such as NVIDIA's Megatron GPT2 which contains 8.3 billion parameters and OpenAI's GPT3 which contains 160 billion parameters).

On the other hand, vertical domains, such as bioinformatics and finance, have large collections of textual documents, such as technical research papers, analysis reports and financial news. These domains require domain specific knowledge to tackle down-stream applications. In bioinformatics domain, there are researches such as NVIDIA's BioMegatron (Shin et al., 2020) and Microsoft's domain-specific language models for Biomedical NLP (Gu et al., 2021). We focus on building finance-domain pretrained language models in this paper. In this finance domain, there is limited work on quite limited languages. FinBERT (Liu et al., 2020) was proposed for financial text-mining in English language. 61GB textual data are collected from English wikipedia and BooksCorpus, FinancialWeb (13GB, 3.31B words), FinancialWeb (24GB, 6.38B words), YahooFinance (19GB, 4.71B words), and RedditFinanceQA (5GB, 1.62B words). In addition, FinBERT has two versions, *LARGE* and *BASE*, which share the same model settings of transformer and pre-training hyper-parameters as BERT. NTT-data is one of the few companies that are known of developing financial BERT for Japanese language[1]. NTT-data used 12.7GB textual data with a vocabulary size of 32K

[1] https://www.nttdata.com/jp/ja/news/release/2020/071000/

	FinBERT	NTT-Data	FinMegatron
Language	En	Jp	En, Jp
Vocab	30K	32K	customize
Parameter	330M	330M	customize
Model open	No	No	Yes
Code open	No	No	Yes
Model parallel	NA	NA	Yes

Table 1: Comparison of BERT in financial domain.

that includes both words and sub-words (such as Japanese characters, kana). The default setting of BERT-LARGE is used in NTT-data's financial BERT as well.

2 FinMegatron Configurations

We name our pretrained language models in financial domain as *FinMegatron*, in which *Fin* stands for financial domain and *Megatron*[2] (Shoeybi et al., 2019) is our general pretrained language models with model parallelism with open-source code written in PyTorch[3], support mixed precision training (fp16 and fp32) [4] under Apex[5] and parallelism of data, pipeline and tensor.

Table 1 lists the differences of our FinMegatron and two baselines, FinBERT (Liu et al., 2020) and NTT-Data's BERT. We support both English and Japanese languages in which Japanese tokenizer such as Mecab with IPAdict is integrated in our code. In addition, we support customizing vocabulary set so that vertical domain's named entities can be better covered by training Byte-Pair Encoding (BPE)[6] and WordPiece (Kudo, 2018) under the given large-scale textual dataset.

In order to support customized vocabulary size, we reuse the code *build_vocab.py* in Tohoku-U's BERT[7] for general domain. For example, in our Japanese BERT model, we set the vocabulary size to be 40k.

Besides the traditional usage of mini-batch for data-parallelism, we also leverage the tensor and pipeline parallelism. Tensor parallelism means that we can cut one tensor by its last dimension into several tensors and each tensor allocated in one GPU. For example, if one tensor's shape is (batch size=b, sequence length=s, hidden size=h) and we use two GPUs to respectively store half of it. Then each GPU will have a tensor of (b, s, h/2). We can define an activate function to implement this forward cutting and backward gathering.

For example, to implement the multi-layer perceptron (MLP) layer (i.e., affine linear projection of from h to 4*h and then back to hidden-size used in point-wise feed-forward layer) in transformer, we define two types of parallelism, column-parallel and row-parallel. For an input tensor X, column-parallel linear layer performs $Y = XA+b$ where matrix A is separated by columns alike $A = [A_1, A_2]$ (when p=2). Then we have $Y_1 = XA_1+b$ and $Y_2 = XA_2 + b$. In forward process, X is not changed and broadcast to each GPU, in backward process, PyTorch's *all-reduce* is performed to sum-up tensor values from different GPUs and sync their values to be the same and up-to-date. Typically, in the first affine linear projection from h to 4h, A_i will take a shape of (h, 4h/p) where p is the number of parallel process or number of GPUs for tensor parallelism. So we will have Y_i's shape of (b, s, h) * (h, 4h/p) = (b, s, 4h/p). The second affine linear projection is a row-parallel linear layer with original A to be (4h, h) and here cut by rows to be (4h/p, h). This means that now A is cut by rows and $A = [A_1, A_2]^T$ (when p=2).

To implement the tensor-parallelism for the multi-head self-attention layer, we can first separate by the number of heads and then each head's tensor can be further paralleled by being separated following the final hidden size dimension. There are four linear layers in the multi-head self-attention layer, the first three linear layers are to project the input X into Q, K, and V tensors and the forth is a simple linear projection of the output of the multi-head self-attention tensor. We can combine the first three linear layers into one layer with weight matrix to be from the original 3*(h, h) to now (h, 3h). Then we can make use of the column-parallelism linear layer used in MLP layer. For the forth layer, it is actually a row-parallelism layer used in MLP layer.

There are 24 encoder blocks in BERT-LARGE and each block contains a multi-head self-attention layer followed by a point-wise feed-forward layer. We can further perform pipeline-parallelism to separate each For example, splitting a model with 24 transformer layers across 4 stages would mean each stage gets 6 transformer layers each. Through this

[2] https://github.com/NVIDIA/Megatron-LM
[3] https://pytorch.org/
[4] https://docs.nvidia.com/deeplearning/performance/mixed-precision-training/index.html
[5] https://github.com/NVIDIA/apex
[6] https://en.wikipedia.org/wiki/Byte_pair_encoding
[7] https://github.com/cl-tohoku/bert-japanese

	English	Japanese
file size	15GB	13GB
documents	5,516,610	5,221,226
sentences	97,122,204	40,467,789
tokens	2.38B	1.06B
vocab	7.18M	364K
vocab-bert	30K	40K
model size	335M	346M

Table 2: Statistical Information of the English and Japanese datasets for pre-training.

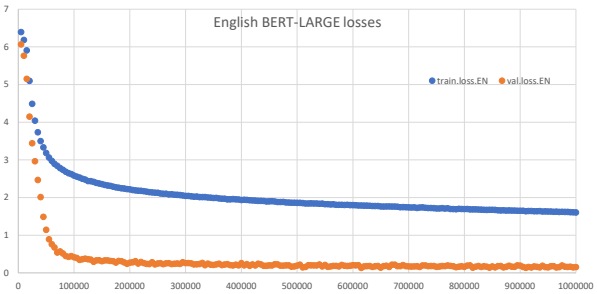

Figure 1: Training and validating losses for the English finanical BERT-LARGE.

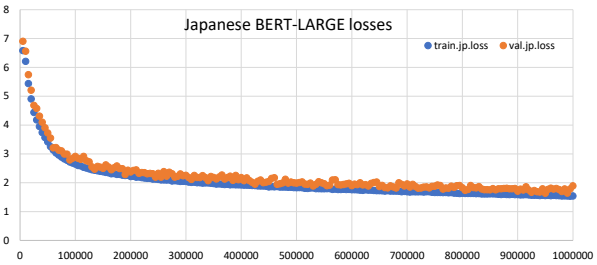

Figure 2: Training and validating losses for the Japanese finanical BERT-LARGE.

way, the forward and backward computing can be paralleled across each stage.

In our BERT models, we also follow the default settings of BERT-LARGE. The network contains 24 layers in which each layer contains a multi-head self-attention layer and a point-wise feed-forward layer with residual adding and layer normalization included as well. We set the max position embedding and maximum sequence length to be 512, number of attention heads to be 16, hidden size to be 1024. The whole dataset is separated into three subsets: 94.9% for training, 5% for validating and 1% for testing.

3 Experiments

3.1 Datasets

We extract finance related documents from large-scale multilingual datasets, such as Google's C4[8] and Facebook's CC-100[9] which are all originally collected from the web. In order to perform the filtering, we keep a large finance word list for each language. The word lists are collected from the web.

Table 2 lists the major statistical information of the English and Japanese datasets of financial domain. An independent document stands for a complete web page or a whole news or even a whole analysis report.

3.2 Training Details

Figure 1 and 2 illustrates the train/validation losses during training the BERT-LARGE models. We set 1 million iterations (with time costing to be 190ms/iteration) for the English dataset and 1 million iterations (with time costing to be 200ms/iteration) for the Japanese dataset.

For the English training, we employ a NVIDIA DGX-1 workstation with 8 V100 cards[10] (16GB memory each, micro-batch size to be 4 and global batch size to be 32), and for the Japanese training, we employ a NVIDIA DGX-1 workstation with 4 V100 cards (16GB memory each, micro-batch size to be 4 and global batch size to be 16). All these workstations come from a large-scale DGX-SUPERPOD[11] in NVIDIA. In addition, note that our open-source code is suitable of using large-scale textual datasets as well to train on NVIDIA's large-scale clusters of DGX-SUPERPOD with as many as 3,072 A100 GPUs[12]. The losses in English models drop from 6.5 around to less than 2 and finally at around 1.6 for training and 0.2 for validating. Finally, the 1% test set's loss is 0.224. On the other hand, the losses in Japanese models drop from 7 around to around 2 in both the training and validating. Finally, the 1% test set's loss is 1.648.

4 Conclusion

We have trained two BERT-LARGE BERT models in English and Japanese for financial domain using the Megatron open-source toolkit in NVIDIA. Our

[8] https://www.tensorflow.org/datasets/catalog/c4

[9] http://data.statmt.org/cc-100/

[10] https://www.nvidia.com/en-us/data-center/v100/

[11] https://www.nvidia.com/en-us/data-center/dgx-superpod/

[12] https://github.com/NVIDIA/Megatron-LM

code and model support both customized vocabulary sizes and model paralleling among multi-GPU of single-node and even multi-nodes. Considering that this is still a work in progress, we will include the training of GPT models as well for English and Japanese financial domain. Finally, down-stream NLP applications such as market understanding (Wu, 2020) are also to be fine-tuned again with these pretrained BERT models.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Wafaa S. El-Kassas, Cherif R. Salama, Ahmed A. Rafea, and Hoda K. Mohamed. 2021. Automatic text summarization: A comprehensive survey. *Expert Systems with Applications*, 165:113679.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2021. Domain-specific language model pretraining for biomedical natural language processing.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Zhuang Liu, Degen Huang, Kaiyu Huang, Zhuang Li, and Jun Zhao. 2020. Finbert: A pre-trained financial language representation model for financial text mining. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 4513–4519. International Joint Conferences on Artificial Intelligence Organization. Special Track on AI in FinTech.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Hoo-Chang Shin, Yang Zhang, Evelina Bakhturina, Raul Puri, Mostofa Patwary, Mohammad Shoeybi, and Raghav Mani. 2020. Biomegatron: Larger biomedical domain language model.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *CoRR*, abs/1909.08053.

Felix Stahlberg. 2020. Neural machine translation: A review and survey.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Xianchao Wu. 2020. Event-driven learning of systematic behaviours in stock markets. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2434–2444, Online. Association for Computational Linguistics.

Lei Zhang, Shuai Wang, and Bing Liu. 2018. Deep learning for sentiment analysis : A survey.

Zhuosheng Zhang, Hai Zhao, and Rui Wang. 2020. Machine reading comprehension: The role of contextualized language models and beyond.