CatBERT: An Incrementally Trained Language Representation Model for E-Commerce Applications

Tejaswini Mallavarapu
Kennesaw State University
Georgia, USA
tmallava@students.kennesaw.edu

Ying Xie
Kennesaw State University
Georgia, USA
yxie2@kennesaw.edu

Simon Hughes
The Home Depot, USA
simon_hughes@homedepot.com

ABSTRACT

Contextual pre-trained language models, such as BERT, have made significant gains in various NLP tasks by training in a self-supervised manner. However, these models are not explicitly aware of domain-specific knowledge, which is essential for downstream applications in many domains, such as tasks in e-commerce scenarios. In this paper, we propose an incremental language model that is called CatBERT to be trained from scratch on e-commerce catalog data through an incremental training process. Experimental studies showed that our proposed incremental training process incrementally incorporates richer semantic information embedded in the catalog data to the CatBERT model and each of such increments contributes positively to the improvement of the model performance in downstream Information Retrieval applications. Our experimental studies further showed that the proposed CatBERT, which is trained from scratch on e-commerce catalog data, outperformed pre-trained BERT that was fine-tuned on the same catalog data using the same incremental training process.

KEYWORDS

Natural Language Processing, Deep Learning, E-Commerce, Transformers, BERT, Incremental Training, Information Retrieval

ACM Reference Format:
Tejaswini Mallavarapu, Ying Xie, and Simon Hughes. 2022. CatBERT: An Incrementally Trained Language Representation Model for E-Commerce Applications . In *Proceedings of Feb, 2022 (Interactive And Scalable Information Retrieval Methods For Ecommerce '22).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Pre-trained language models, such as ELMo [4], GPT [10], BERT [2], RoBERTa [7], and XLNet [12], have achieved significant advances in a variety of downstream natural language processing (NLP) tasks. These models are typically pre-trained using an either bi-directional or auto-regressive process on large-scale text repositories. Among the recent self-supervised pre-trained language models, BERT has emerged as a very powerful method due to its simplicity and superior performance than other models [2]. More specifically, BERT achieved state-of-the-art results in a variety of NLP tasks such as sentiment analysis, language inference, sentence similarities, text classification and more [7]. The success of BERT is largely attributed to its powerful attention-based Transformer and a well-designed pre-training strategy for modeling context inherent in text sequence.

Pre-trained BERT typically encodes rich semantic patterns from the general language. In order to adapt BERT for applications in a specific domain, several domain adaptive BERT models, such as BioBERT [6], SciBERT [1], PubMedBERT [3] and, ClinicalBERT [5], have been proposed. All these models further trained a pre-trained BERT on large-scale domain text repositories. For example, BioBERT was generated by further training the pre-trained BERT on biomedical corpus. BioBERT was demonstrated to be outperforming general pre-trained BERT and previous state-of-the-art models on some critical Natural language Modeling (NLP) tasks such as named entity recognition (NER), question answering (QA) and, relation extraction (RE). Similarly, SciBERT and PubMedBERT were further trained on large text repositories of scientific publications and PubMed articles respectively and better supports downstream applications in its own domain than general pre-trained BERT.

In this paper, we are concerned with developing a domain specific BERT model for E-Commerce applications. More specifically, we would like to train a BERT model on product catalog data such that it can be applied to a variety of downstream e-commerce applications such as product search. There are two options for building such a BERT model. One is to further train a general pre-trained BERT on product catalog data. The second option is to train a randomly initiated BERT from scratch on catalog data. Given our observation that the vocabulary and language structures used in product catalog may be quite different from general language that a pre-trained BERT was trained upon, we hypothesize that the second option, i.e., training a randomly initiated BERT from scratch directly on catalog data, could mitigate negative effects that are caused by irrelevant language patterns learned from general language and thus yield better performance in supporting downstream applications. To illustrate this hypothesis, lets look at a sample piece of product information obtained from Home Depot website.

$< id >'$ 204790568$' < /id >< title >$ *somerset assembled 24 x 30 x 12 in. single door hinge right wall kitchen blind corner cabinet in manganite* $< /title >< brands >$ *outdoor cooking* $< /brands >< L_1 >$ *kitchen* $< /L_1 >< L_2 >$ *kitchen cabinets* $< /L_2 >< L_3 >$ *in stock kitchen cabinets* $< /L_3 >$.

From this example, we can see that product catalog data are different from general English from the following perspectives

Interactive And Scalable Information Retrieval Methods For Ecommerce '22, AZ, USA,

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

- Product catalog doesn't have sentence structures; instead, it contains multiple sections of product attributes, which are likely phrases or single words.
- Product title can be a long phrase that includes multiple consecutive adjectives and/or numeric dimensions.
- Vocabulary used in product catalog is much more restricted than general language.

The study reported in [13] is aligned with our observation on catalog data by demonstrating that further training pre-trained BERT on product catalog only generates limited improvement on model performance over pre-trained BERT itself. Furthermore, we performed a detailed comparative study on these two options, with the experimental results supporting our hypothesis. Please refer to Section III - D for our experimental results.

Once choosing to train randomly initiated BERT, rather than pre-trained Bert, on product catalog data, we need to determine an optimal training process. A straightforward way of training is to view the whole piece of product information as one input sequence, based on which we apply masked language modeling (MLM) [11] approach to train the BERT model. However, this way of training tends to ignore some important hierarchical relationship embedded in the product information, such as the relationship between product and brand, and the relationship between product and different levels of taxonomy. Therefore, we design a new incremental training strategy for BERT to encode comprehensive information embedded in product catalog, including vocabulary, phrases, and hierarchical relationships. More specifically, the incremental training strategy involves multiple rounds of training. The first round uses MLM to train BERT on the $< title >$ section of each product, so that the BERT learns the catalog vocabulary and phrases. Then the following rounds of learning lays classification layers upon BERT to predict higher level concepts such as product brand and different levels of taxonomy. Our experimental studies show that more rounds of learning that incorporates more information from catalog to BERT, better the model performance on downstream search applications.

In summary, we propose an incremental language model that is called CatBERT to be trained from scratch on e-commerce catalog data through an incremental training process. Experimental studies showed that our proposed incremental training process incrementally incorporates richer semantic information embedded in the catalog data to the CatBERT model and each of such increments contributes positively to the improvement of the model performance in downstream Information Retrieval applications. Our experimental studies further showed that the proposed CatBERT, which is trained from scratch on e-commerce catalog data, outperformed pre-trained BERT that was fine-tuned on the same catalog data using the same incremental training process.

The rest of the paper is organized as follows. Section 2 introduces briefly about BERT model and our novel CatBERT model architecture and training methods in detail. In Section 3, we present a brief overview of data preparation, experimental results and visualizations of our experiments by reporting the performance of proposed models over state of art models. Finally, we conclude our work and discuss future work in Section 4.

2 METHOD

In this section, we introduce CatBERT, an incremental language model that is trained from scratch on product catalog data using an incremental training process for the purpose of supporting downstream e-commerce applications. Given that CatBERT model is based on BERT, we briefly introduce BERT in the following subsection before explaining CatBERT in details.

2.1 BERT Model

Bert (Bidirectional Encoder Representations from Transformers) is a deep neural network that uses multi-layer bidirectional transformer encoders to learn to represent the text. BERT base model comprises 12 layers of transformer blocks each with 768 hidden units, and 12 attention heads. It is pre-trained on open-domain large text corpus using combined losses, one for predicting masked tokens and the other for predicting whether the second part of the input follows the first part. BERT reads an input sequence with a fixed length and returns an embedding for each token in the input sequence. The embedding outputs by a pre-trained BERT for each token encodes how other tokens in the input sequence are attended to this token. The output embedding then can be used as input for downstream applications.

2.2 CatBERT

CatBERT starts with randomly initiated BERT model. The overall training process of CatBERT is illustrated in Figure 1. As the figure shows, the first phase of training enables the CatBERT to learn vocabulary and phrases from product titles using MLM. We denote the resulting model of this training as $CatBERT\_v0$. Then, each following phase of training will incrementally incorporates certain high-level concepts of products to the CatBERT by laying a fully connected classification layer on top of previous version of CatBERT. Using the Home Depot product catalog as an example, the second phase of training learns to classify each product to the brand it belongs to. The input of this phase of training is individual product title, same as the first phase. The output is a label that represents a brand. The training loss of this phase not only modifies the weights the classification layer, but also back-propagates to the underneath BERT architecture to modifies the parameters, through which the relationship between product and brand has been incorporated into CatBERT. We denote the resulting model (after removing the added classification layer) as $CatBERT\_v1.0$. Similarly, the following phases of training learn to classify products to other high-level concepts. Again, for the example of the Home Depot catalog, $CatBERT\_v1.1$, $CatBERT\_v1.2$, and $CatBERT\_v1.3$ denote the version of CatBERT after learning to classify products to taxonomy level 1, 2, and 3 respectively.

CatBERT is designed to incorporate comprehensive information on product. In this study, $CatBERT\_v0$ encodes basic descriptions of products; $CatBERT\_v1.X$ encodes high-level concepts of products, such as brands and taxonomy. However, CatBERT can continue to be trained to incorporate more information on products. For example, in the future, $CatBERT\_v2.X$ can be trained to encode product image information; $CatBERT\_v3.X$ can learn to encode product-user interactions; and so on.

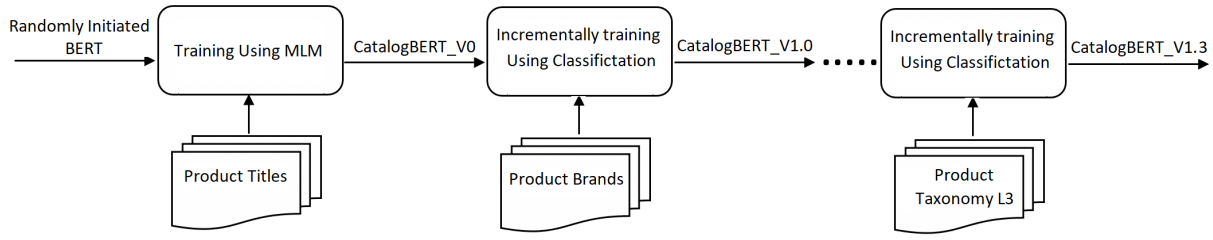

Figure 1: Overview of training of CatBERT

The type of product information that needs to be Incorporated into CatBERT also depends on what types of downstream applications it supports. In this study, we apply CatBERT to text based product search. For this application, we tested $CatBERT\_v0$ and $CatBERT\_v1.X$. Our hypothesis is that $CatBERT\_v(y)$ has better performance supporting text based product search than $CatBERT\_v(x)$ if $2 > y > x$.

3 EXPERIMENT SETUP

The goal of our experimental study is to validate the following two hypotheses we mentioned in the previous sections.

- $CatBERT\_v(y)$ has better performance supporting text based product search than $CatBERT\_v(x)$ if $2 > y > x$.
- training a randomly initiated BERT from scratch directly on catalog data yields better performance in supporting search applications than using pre-trained BERT that is further trained on catalog data in the same way.

To validate the first hypothesis, we compare the following versions of CatBERT on there performance on product search.

- $CatBERT\_v0$ - randomly initialized BERT trained on product title using MLM
- $CatBERT\_v1.0$ incrementally trained on product brands using classification
- $CatBERT\_v1.1$ incrementally trained on product taxonomy level 1 using classification
- $CatBERT\_v1.2$ incrementally trained on product taxonomy level 2 using classification
- $CatBERT\_v1.3$ incrementally trained on product taxonomy level 3 using classification

To validate the second hypothesis, we further train a pre-trained BERT on product catalog data using the same incremental training process to form the following versions. Then we compare $CatBERT\_v(x)$ and $BERT\_v(x)$ on each search senario.

- $BERT\_v0$ - pre-trained BERT further trained on product title using MLM
- $BERT\_v1.0$ incrementally trained on product brands using classification
- $BERT\_v1.1$ incrementally trained on product taxonomy level 1 using classification
- $BERT\_v1.2$ incrementally trained on product taxonomy level 2 using classification
- $BERT\_v1.3$ incrementally trained on product taxonomy level 3 using classification

3.1 Dataset

We use the product catalog of the Home Depot for our experimental studies. This data set consists of roughly 600k products in JSON format. The product attributes in the dataset as follows:

- ID: refers to the product identification number
- Title: is the product name
- Brand: refers to the brand of a product
- Taxonomy: is the website-specific hierarchical taxonomy levels for a product

3.2 Model Building

All product titles are used to train both $CatBERT\_v0$ and $BERT\_v0$ using MLM. For incrementally training all $CatBERT_v1.X$s and $BERT_v1.X$s using classification, we split the product data into: 400K training, 100k validation, and 100k testing. Validation data is used for selecting values for model hyperparameters and determining early stopping of training; Test data is used to evaluating classification performance. We build all the models using the PyTorch [8] Huggingface [9] deep learning framework. All the models are trained with Adam optimizer [**?**] with learning rate to be 0.0005, batch size to be 32, and epoch number to be 10.

3.3 Product Search

We use product search to evaluate the model performance for all $CatBERT_vX$s and $BERT_vX$s. In order to apply CatBERT to product search, we first use CatBERT to index all the products. The indexing process works in the following way. Given a product P, whose product title has n tokens in order $\{t_1, t_2, t_i, ..., t_n\}$, and $e(t_i)$ denotes the embedding for t_i output by a $CatBERT\_vX$ or $BERT\_vX$. Then the index for P can be denoted as $\{e(t_1), e(t_2), e(t_i), ..., e(t_n)\}$. Given a query $Q = \{q_1, q_2, q_j, ..., q_m\}$, we also feed it to a $CatBERT\_vX$ or $BERT\_vX$ to generate its representation as $\{e(q_1), e(q_2), e(q_j), ..., e(q_m)\}$, where $e(q_j)$ is the embedding for the query token q_j output by the model. Now, the similarity between the query Q and the product P can be calculated using the following formulae.

$$\text{similarity}(Q, P) = \sum_j \max_i \frac{e(q_j) \cdot e(t_i)}{\|e(q_j)\|\|e(t_i)\|} \tag{1}$$

3.4 Evaluation of Search Performance

In order to evaluate the performance of all $CatBERT\_vX$ or $BERT\_vX$. We randomly picked 1000 queries that were issued by customers on homedepot.com. Each query is associated with two sets. The first set (denoted as $set1$)contains all ids of those products that

Table 1: Precision@k results for all *CatBERT_vX*s on *set*1

Version	1	4	16	24	48	96
CatBERT_v0	0.63	0.440	0.196	0.143	0.075	0.038
CatBERT_v1.0	0.68	0.530	0.261	0.189	0.097	0.048
CatBERT_v1.1	0.68	0.505	0.214	0.149	0.077	0.038
CatBERT_v1.2	0.75	0.600	0.291	0.215	0.113	0.056
CatBERT_v1.3	0.77	0.640	0.351	0.260	0.137	0.069

Table 2: Recall@k results for all *CatBERT_vX*s on *set*1

Version	1	4	16	24	48	96
CatBERT_v0	0.0578	0.145	0.204	0.211	0.215	0.215
CatBERT_v1.0	0.062	0.185	0.292	0.301	0.303	0.303
CatBERT_v1.1	0.063	0.175	0.247	0.252	0.254	0.254
CatBERT_v1.2	0.070	0.208	0.341	0.354	0.358	0.358
CatBERT_v1.3	0.072	0.230	0.412	0.433	0.440	0.441

Table 3: Precision@k results for all *CatBERT_vX*s on *set*2

Version	1	4	16	24	48	96
CatBERT_v0	0.78	0.648	0.408	0.323	0.194	0.133
CatBERT_v1.0	0.79	0.715	0.515	0.428	0.266	0.138
CatBERT_v1.1	0.85	0.753	0.488	0.387	0.230	0.118
CatBERT_v1.2	0.85	0.783	0.611	0.514	0.333	0.183
CatBERT_v1.3	0.87	0.805	0.669	0.586	0.401	0.219

Table 4: Recall@k results for all *CatBERT_vX*s on *set*2

Version	1	4	16	24	48	96
CatBERT_v0	0.020	0.064	0.147	0.167	0.183	0.185
CatBERT_v1.0	0.02	0.069	0.188	0.227	0.261	0.262
CatBERT_v1.1	0.021	0.075	0.183	0.209	0.234	0.236
CatBERT_v1.2	0.0215	0.078	0.233	0.284	0.334	0.342
CatBERT_v1.3	0.0216	0.081	0.256	0.322	0.397	0.408

were purchased by customers under this query; while the second set (denoted as *set*2) contains all ids of those products that were clicked or purchased by customers under this query. For each of the 1000 queries, let each model return the k most similar products towards this query. Then we will calculate precision@k and recall@k on each of *set*1 and *set*2 respectively. Formally, precision@k and recall@k are defined as follows, where k = 1, 4, 16, 24, 48, or 96 in our study.

$$\text{Recall@k} = \frac{\#\text{ of relevant items @k}}{\#\text{of total relevant items}} \tag{2}$$

$$\text{Precision@k} = \frac{\#\text{of relevant items @k}}{\#\quad\text{of total items}} \tag{3}$$

3.5 Experimental Results

Results of precison@k and recall@k on *set*1 for all *CatBERT_vX*s are shown in Table 1 and 2. Results of precison@k and recall@k on *set*2 for all *CatBERT_vX*s are shown in Table 3 and 4. We further

Table 5: precison@k results for all *BERT_vX*s on *set*1 evaluation data

Version	1	4	16	24	48	96
BERT_v0	0.64	0.530	0.348	0.287	0.170	0.089
BERT_v1.0	0.78	0.693	0.469	0.383	0.245	0.130
BERT_v1.1	0.73	0.620	0.431	0.346	0.203	0.101
BERT_v1.2	0.84	0.738	0.558	0.476	0.306	0.163
BERT_v1.3	0.86	0.773	0.627	0.551	0.383	0.211

Table 6: precison@k results for all *BERT_vX*s on *set*2 evaluation data

Version	1	4	16	24	48	96
BERT_v0	0.46	0.3500	0.176	0.123	0.065	0.033
BERT_v1.0	0.61	0.478	0.222	0.160	0.085	0.043
BERT_v1.1	0.55	0.433	0.191	0.132	0.067	0.034
BERT_v1.2	0.69	0.548	0.262	0.192	0.10	0.05
BERT_v1.3	0.71	0.635	0.341	0.255	0.14	0.070

Table 7: Recall@k results for all *BERT_vX*s on *set*1 evaluation data

Version	1	4	16	24	48	96
BERT_v0	0.039	0.11	0.187	0.189	0.191	0.191
BERT_v1.0	0.052	0.154	0.245	0.253	0.257	0.257
BERT_v1.1	0.048	0.147	0.220	0.223	0.225	0.225
BERT_v1.2	0.062	0.195	0.317	0.328	0.332	0.332
BERT_v1.3	0.064	0.225	0.392	0.411	0.422	0.422

Table 8: Recall@k results for all *BERT_vX*s on *set*2 evaluation data

Version	1	4	16	24	48	96
BERT_v0	0.017	0.053	0.131	0.155	0.170	0.171
BERT_v1.0	0.020	0.069	0.172	0.20	0.234	0.234
BERT_v1.1	0.018	0.060	0.161	0.187	0.208	0.208
BERT_v1.2	0.021	0.072	0.211	0.263	0.308	0.313
BERT_v1.3	0.022	0.076	0.235	0.299	0.372	0.383

show precision-recall curves for all *CatBERT_vX*s on *set*1 in Fig. 2, and on *set*2 in Fig. 3. As can be seen, *CatBERT_v*(y) has better performance supporting text based product search than *CatBERT_v*(x) when $y > x$, with one exception that the precision of *CatBERT_v*1.0 has better precision than *CatBERT_v*1.1 at low recall on *set*2.

Results of precison@k and recall@k on *set*1 for all *BERT_vX*s are shown in Table 6 and 7. Results of precison@k and recall@k on *set*2 for all *BERT_vX*s are shown in Table 5 and 8. We further compare the precision-recall of each pair of *CatBERT_VX*s and *BERT_VX*s on *set*1 in Fig. 4; on *set*2 in Fig. 5. The comparison clearly validated that training a randomly initiated BERT from scratch directly on catalog data yields better performance in supporting search than using pre-trained BERT that is further trained on catalog data in the same way.

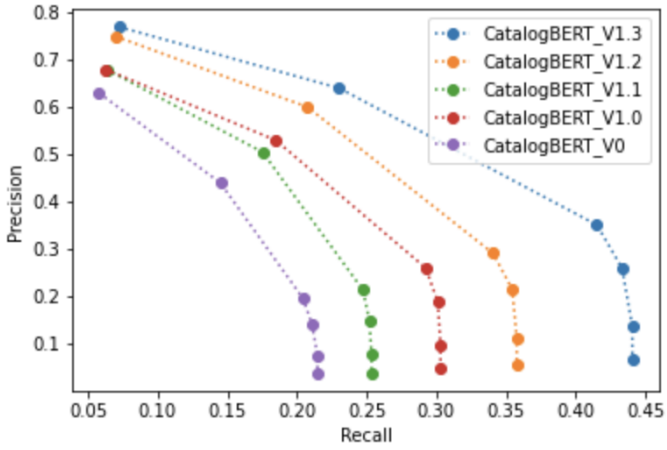

Figure 2: Comparison of Precision-Recall Curves of CatBERT Versions on *set*1 Evaluation Data

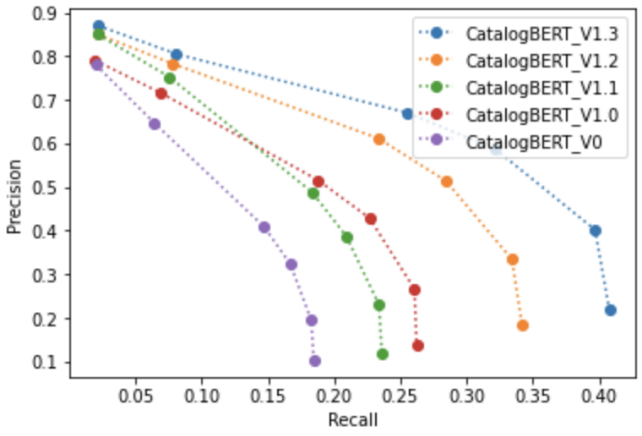

Figure 3: Comparison of Precision-Recall Curves of CatBERT Versions on set2 Evaluation Data

4 CONCLUSION AND FUTURE WORK

In this paper, we proposed CatBERT, an incrementally trained language representation model for E-commerce applications. Starting with randomly initiated BERT, CatBERT is trained first on product titles using MLM and then incrementally trained on high level product concepts using classification. Experimental studies show that each increment of training boosts the performance of the model on downstream search applications. Moreover, CatBERT outperforms pre-trained BERT that is further trained on product catalog using the same incremental training.

The proposed CatBERT is an open-ended model. For future work, we will continue to upgrade CatBERT with more information on products, such as product images and product-user interactions, and apply it to multiple downstream applications including automatic question answering and product recommendation. We will also specifically tune CatBERT for search applications using bi-encoder [JCR21] or poly-encoder [SH19] architecture on product transaction data in order to further improve online search performance.

REFERENCES

[1] Iz Beltagy, Kyle Lo, and Arman Cohan. 2020. SCIBERT: A pretrained language model for scientific text. *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference.* https://doi.org/10.18653/v1/d19-1371

[2] Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference* 1.

[3] Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2022. Domain-Specific Language Model Pretraining for Biomedical Natural Language Processing. *ACM Transactions on Computing for Healthcare* 3 (1 2022), 1–23. Issue 1. https://doi.org/10.1145/3458754

[4] Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)* 1. https://doi.org/10.18653/v1/p18-1031

[5] Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. 2019. ClinicalBERT: Modeling Clinical Notes and Predicting Hospital Readmission. (4 2019). http://arxiv.org/abs/1904.05342

[6] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. BioBERT: A pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* 36 (2 2020), 1234–1240. Issue 4. https://doi.org/10.1093/bioinformatics/btz682

[7] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. http://arxiv.org/abs/1907.11692

[8] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf

[9] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *CoRR* abs/1910.03771 (2019). arXiv:1910.03771 http://arxiv.org/abs/1910.03771

[10] Xinghao Wu and Moritz Lode. 2020. Language Models are Unsupervised Multitask Learners (Summarization). *OpenAI Blog* 1 (2020). Issue May.

[11] Atsuki Yamaguchi, George Chrysostomou, Katerina Margatina, and Nikolaos Aletras. 2021. Frustratingly Simple Pretraining Alternatives to Masked Language Modeling. (9 2021). http://arxiv.org/abs/2109.01819

[12] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. (6 2019). http://arxiv.org/abs/1906.08237

[13] Denghui Zhang, Zixuan Yuan, Yanchi Liu, Zuohui Fu, Fuzhen Zhuang, Pengyang Wang, Haifeng Chen, and Hui Xiong. 2020. E-BERT: A Phrase and Product Knowledge Enhanced Language Model for E-commerce. (9 2020). http://arxiv.org/abs/2009.02835