Small is the New Big: Pre-finetuned Compact Models are Better for Asynchronous Active Learning

Dantong Liu
Amazon.com, Inc
Sunnyvale, United States
lidanton@amazon.com

Kaushik Pavani
Amazon.com, Inc
Seattle, United States
sripava@amazon.com

Sunny Dasgupta
Amazon.com, Inc
Seattle, United States
sunnyd@amazon.com

Abstract

We examine the effects of model size and pre-finetuning in an active learning setting where classifiers are trained from scratch on 14 binary and 3 multi-class text classification tasks. We make an important observation that, in realistic active learning settings, where the human annotator and the active learning system operate in asynchronous mode, a compact pre-finetuned 1-layer transformer model with 4.2 million parameters is 30% more label efficient when compared to the larger 24-layer 84 million parameter transformer model. Further, in line with previous studies, we note that pre-finetuning transformer models on related tasks improves label efficiency of downstream tasks by 12%-50%. The compact pre-finetuned model does not require GPUs, making it a viable solution for large-scale real-time inference with cheaper CPU options.

1 Introduction

Active learning is a popular approach used to reduce the manual labeling effort required to train a classifier. In active learning, we iteratively acquire labels from an annotator and use them to train a classifier.

Most existing academic literature (Huang and Zhou, 2013; Shao et al., 2019) on active learning assumes that manual labeling process can only happen after the model update is complete, making the active learning loop 'synchronous'. In practice, this implies that human annotators have to wait until an active learning iteration (training on labeled data and inference on all unlabeled data) process is complete before they can provide more labels. As pointed out by (Huang et al., 2021), in realistic production settings, 'synchronous' active learning will significantly decrease annotators' productivity. To this end, typical production systems such as Sagemaker GT (sag) employ 'asynchronous' active learning setup where the human annotators continuously provide annotations while the training and inference happen in the background.

Pre-finetuning proposed by Aghajanyan *et al* (2021) is a stage after pre-training to further refine representations before end-task finetuning. The purpose of the pre-finetuning step is to increase the similarity between data used for pre-training and downstream finetuning tasks (Phang et al., 2018; Pruksachatkun et al., 2020; Gururangan et al., 2020). Aghajanyan *et al* pose pre-finetuning as a Multi-task learning (MTL) problem on 47 tasks, and their experiments show that incorporating pre-finetuning to RoBERTa and BART models yields consistent improvements in downstream task finetuning, particularly in the low data regime.

In this work, we examine the effects of model size and pre-finetuning in a realistic asynchronous active learning setting on a diverse set of 14 binary and 3 multi-class text classification tasks. Our contributions are three-fold:

1. We present evidence that a small transformer model is ideal for use in large scale environments with asynchronous active learning setting. With a given training and inference infrastructure, large models, counter-intuitively, can increase the number of labeled data required to achieve precision/recall targets set by customers because of their slow training/inference speeds.
2. We conduct an extensive study surrounding the label efficiency of standard pre-trained representations and their respective pre-finetuned counterparts. We show empirical evidence that pre-finetuning helps to reduce the number of labeled data required to build transformer-based classifiers.
3. We present evidence that pre-finetuning can be formulated as a large-scale multi-label classification problem, which enables us to pre-finetune on a large corpus of 2664 classification tasks. This technique helps us learn

Proceedings of The Fourth Workshop on Simple and Efficient Natural Language Processing (SustaiNLP), pages 110–120

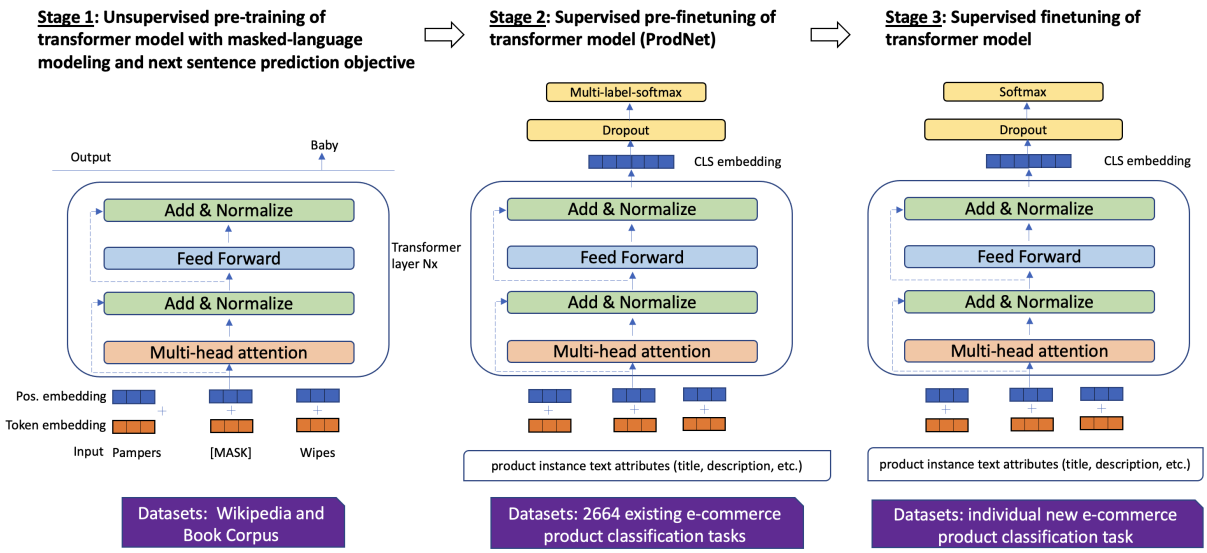


Figure 1: Besides standard pre-training (Stage 1) and then finetuning (Stage 3), our training procedure includes an intermediate pre-finetuning step (Stage 2) where we create ProdNet by training transformer model on data curated from thousands of existing classifiers.

from thousands of tasks simultaneously. To the best of our knowledge, this is the first work to realize the gain of pre-finetuning under a restricted latency budget in a large-scale asynchronous active learning setting.

2 Pre-Finetuning

2.1 Transformer Models

The details of various transformer architectures used in this study are shown in Table 2 in Appendix A.1. We perform pre-finetuning on BERT-variants models on e-commerce product classification tasks to create ProdNet variants (ProdNet-1L, ProdNet-2L, ProdNet-4L, ProdNet-6L, and ProdNet-24L-P). For example, we pre-finetune BERT-6L to create ProdNet-6L. The training process for creating ProdNet-based classifiers is illustrated in Fig. 1. We note the model parameters, training speed, and inference latency of various transformer models used in the study in Table 1. Note, the ProdNet variants have the same model architecture with the corresponding non-pre-finetuned counterparts (BERT variants). One major difference between BERT-24L-P model and the other BERT variants is that BERT-24L-P is unsupervised pre-trained using internal e-commerce product data while other BERT variants are pre-trained using public datasets. For the sake of readability, most of experiment results only include the results of 3 model pairs (underscored in Table 1), and we have verified that our conclusions hold for all 5 model pairs.

2.2 Pre-finetuning Datasets

To pre-finetune transformers, we selected 2664 proprietary binary classifiers created from Feb'20 to Sep'21 to classify the e-commerce products. We leverage human labeled training data from the selected binary classifiers, and aggregate all training samples from 2664 binary classifiers. Note that one instance may be a *member* of multiple binary classifiers. For e.g., a bundle instance with an 'eraser' and a 'ruler' may be a member of both 'eraser' and 'ruler' classifiers. Appendix A.2 shows the details of data used for pre-finetuning.

For each instance (product data), we use the *item_name* and *product_description* both for pre-finetuning and finetuning. We focused on the text attributes in this work, and the idea of creating ProdNet is readily generalized to image and multi-modal attributes.

2.3 Methodology

In the existing literature, a common approach for pre-finetuning is multi-task learning (MTL), and the number of tasks used has been fairly limited, e.g., 49 tasks were used in (Aghajanyan et al., 2021). Even at this scale, scientists have reported 'negative transfer', where the learning from upstream tasks can reduce accuracy of downstream tasks. Since we aim to learn from a large set of 2664 binary classification tasks, the traditional MTL approach does not scale. To this end, we formulate the pre-finetuning as a large multi-label classification problem using multi-label softmax loss

Models → Parameters ↓	BERT-1L ProdNet- 1L	BERT-2L ProdNet- 2L	BERT-4L ProdNet-4L	BERT-6L ProdNet-6L	BERT-24L-P ProdNet- 24L-P
# Trans. layers	1	2	4	6	24
Hidden layer size	128	128	312	768	256
# Attention head	2	2	12	12	16
Parameters (MM)	4.2	4.4	14.5	66	84
CPU infer. latency	1.09	2.02	15.3	86	100
GPU infer. latency	0.16	0.18	0.4	8.6	10.6
GPU training time	28	32	54	197	233

Table 1: Inference latency (reported in milli-seconds) is the inference latency per instance computed on CPU (ml.c5.2xlarge) and GPU (ml.g4dn.xlarge) instances with batch size as 32 and max sequence length as 128. Training time is reported in seconds on a GPU (ml.g4dn.xlarge) instance with 2000 labeled data. We focus on three model pairs (underscored) for most of our experiments: BERT-1L vs. ProdNet-1L, BERT-2L vs. ProdNet-2L, and BERT-24L-P vs. ProdNet-24L-P.

function recommended by Mahajan *et al* (2018). While the solution is not generic to all multi-task problems with heterogeneous tasks, it is ideal for our use case since all our tasks of interest are binary classification tasks, which can be combined to create a multi-label dataset. The details about multi-label softmax loss and our experiments are shown in Appendix A.3.

3 Finetuning

We perform asynchronous active learning experiments on the selected finetuning tasks to examine the effects of model size and pre-finetuning in an active learning setting.

3.1 Active Learning

To measure the label efficiency of a classifier, we employ pool-based active learning setting (detailed in Appendix A.4) (Lewis and Gale, 1994; Settles, 2009; Gal et al., 2017). In each active learning iteration, we perform two operations: 1) judiciously select a subset of unlabeled instances for the data pool and send them to the annotator, and 2) train a classifier using new and the previously labeled instances. We continue active learning until convergence criteria are achieved. We use two convergence criteria for experiments which exactly mimics a production setup: 1) the estimated recall/precision for each class-of-interest should be no smaller than the business-specified targets, and 2) predictions on unlabeled data should have stabilized (Bloodgood and Vijay-Shanker, 2014).

For experiments, we use a bot (in lieu of a human annotator) to do the labeling job. To simulate realistic production scenario, we adopt asynchronous active-learning and labeling *i.e.,* the bot keeps providing labels regardless of the progress of training /inference/query acquisition process in the active learning loop. Throughout our experiments, the bot provides 3000 labels per day to mimic the labeling speed of a human annotator.

3.2 Finetuning Datasets

The finetuning datasets were sourced from proprietary binary and multi-class e-commerce product classification tasks created from Oct'21 to Mar'22. We deliberately selected classification tasks created after Oct'21 to 1) simulate real-world scenario where the pre-finetuned model will be used for new classification tasks, and 2) avoid any overlap with the datasets used for pre-finetuning. We selected 14 diverse binary classification tasks with the positive class prevalence ranging from 0.13 to 0.88. We also selected 3 multi-class classification datasets with class cardinalities 10, 8, and 3, respectively. The datasets have a long-tail distribution in terms of class sizes. Since we use a bot for the labeling job to avoid the human-in-the-loop, we need fully-labeled datasets of the finetuning tasks as the source of labels used by the bot, and we also need to compute metrics on data pool, which acts as the test data for active learning experiments. To curate fully-labeled datasets, we employed trained human annotators to manually label all instances in data pool. Note the data pool size of the 17 selected finetuning tasks are ranging from 8M to 17K.

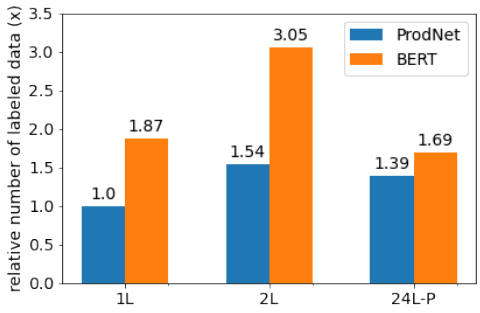
3.3 Metrics

The key metric of interest is the number of labeled data required for the active learning process to

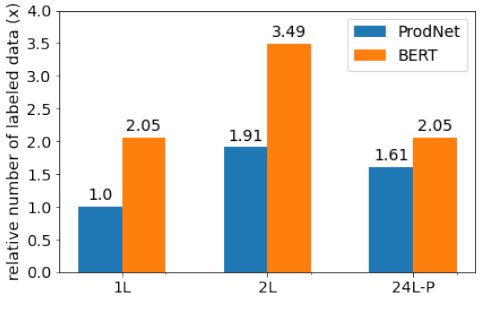
converge, since our goal is to require as few human annotations as possible to build a classifier which satisfies precision/recall targets set by customers. For each active learning experiment, we track the progression of class-level *recall*, and *precision* computed on data pool. We summarize the class-level metrics by reporting *macro*-recall and *macro*-precision. We choose *macro* over *micro* averaging because performance of each class is equally important for our use-case.

4 Experiments Results

Results on binary and multi-class classification tasks: Fig. 2 (a), and (b) show the relative number of labeled data required for active learning experiments to converge averaged across 14 binary classification tasks and 3 multi-class classification tasks, respectively. The results of individual task (including the targeted/achieved precision/recall and number of required labels) are reported in Table 7 and 8 in Appendix A.7. We observed that



(a)



(b)

Figure 2: Average relative number of labeled data required for active learning experiments to converge (a) 14 binary classification tasks, and (b) 3 multi-class classification tasks.

- On average ProdNet-1L requires the least number of labeled data for model convergence among all the benchmarking models, increasing the label efficiency by 40% and 30% compared with ProdNet-2L and ProdNet-24L-P, respectively.
- The pre-finetuned models (ProdNet) consistently reduces the number of labeled data required for model convergence compared to the non-pre-finetuned counterparts. On average, ProdNet-1L requires 51% of labels required by BERT-1L classifier. Similarly, ProdNet-2L requires 50% of the labels that BERT-2L needs, and ProdNet-24L-P requires 88% of labels that BERT-24L-P model takes.
- The gain of pre-finetuning BERT-24L-P model is smaller when compared with the gain of pre-finetuning BERT-1L and BERT-2L. This is intuitive as BERT-24L-P model is unsupervised pre-trained using our internal product data, and it has already learnt internal product related information. In contrast, BERT-1L and BERT-2L are pre-trained using public corpora without the product specific information.
- Interestingly, even without the effect of pre-finetuning, BERT-1L is more label efficient than BERT-2L, which demonstrates the importance of smaller model size in the asynchronous active learning setting.

Do bigger models mean fewer labels? Fig. 3 illustrates the relative number of labeled data required for active learning experiments to converge versus number of parameters in the model on *FEE* dataset. The results show an interesting phenomenon that the number of labeled data required for active learning experiments to converge increases with the rising model parameter size. This is counter-intuitive, as larger models are usually better than smaller models for cases where the training and test datasets are fixed in academic settings. However, in an asynchronous active learning setting, larger models take longer to train and infer (*e.g.*, ProdNet-24L-P is $\sim$ 100 times slower than ProdNet-1L in CPU inference), thereby forcing human annotators to label 'stale' data. As such, larger models miss the opportunity to assist query acquisition module to select unlabeled instances effectively, and unnecessarily accumulate excessive labeled data from annotators. Smaller models, especially when pre-finetuned on 1000s of previously authored classifiers, provide a viable alternative with fast classifier authoring-time and low inference-cost. In addition, the compact pre-finetuned model does not require GPUs for inference, making it a viable solution for real-time

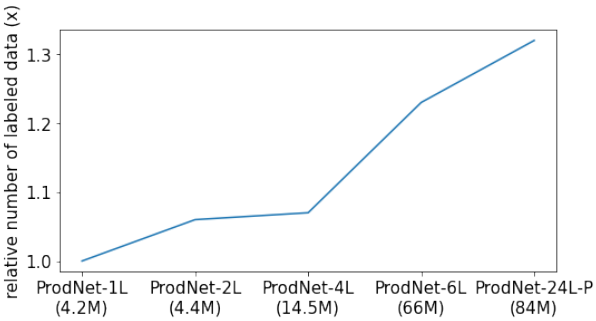


Figure 3: Relative number of labeled data required for active learning experiments to converge with model sizes on *FEE* dataset. *e.g.*, ProdNet-1L classifier requires x labels to converge while ProdNet-2L needs $1.06x$ labels to converge.

large-scale inference with cheaper CPU options.

Why does ProdNet learn faster than the non-pre-finetuned counterpart? Although researchers have cautioned against using attention as a reliable means of model interpretability (Serrano and Smith, 2019; Jain and Wallace, 2019), several recent works use attention weights to partially explain what words/tokens that are most influential to the model (Galassi et al., 2019; Letarte et al., 2018; Vashishth et al., 2019; Clark et al., 2019). To get an intuitive understanding if pre-finetuning helps, we illustrate attention weights[1] of the `[CLS]` token in the last layer for pre-finetuned (ProdNet-2L) and non-pre-finetuned (BERT-2L) models as shown in Fig.4. We choose to visualize `[CLS]` token since in downstream task finetuning, we pass the last layer `[CLS]` representation to a task-specific feed-forward layer and train the classifier end-to-end. It is worth mentioning that we visualize the attention of `[CLS]` token in the original ProdNet-2L (Stage 2 in Fig 1) and BERT-2L (Stage 1 in Fig 1) models without downstream tasks finetuning. Our rationale is that if a model is able to pay attention to the key information in downstream tasks before finetuning, then the model might learn faster when finetuned with the downstream task data (few-shot learning). The input text `Veterinary Formula Flea and Tick Spray for Dogs` is from the downstream task **PetCare** which aims to classify pet-care products designed to treat fleas, ticks, ringworm, or other parasites. The orange and blue color demonstrate the 2 attention heads of ProdNet-2L and BERT-2L. The darker color indicates more attention. In this example, we can observe that ProdNet-2L is able

[1]We used BertViz for attention visualization https://github.com/jessevig/bertviz.

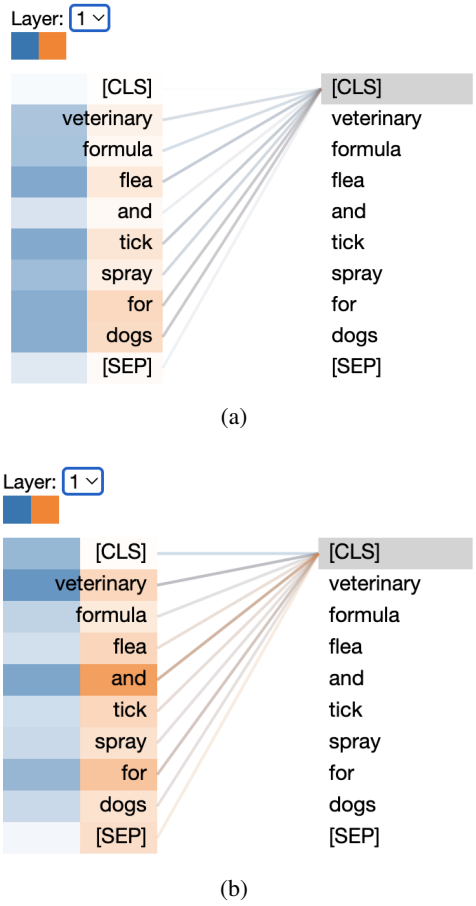


Figure 4: Attention visualization of (a) ProdNet-2L, and (b) BERT-2L

to pay more attention to `flea` and `tick` compared with BERT-2L. We hypothesize that pre-finetuning on relevant e-commerce product classification data helps the model understand potentially important words for the downstream task, thereby allowing the model to learn the downstream task faster.

Appendix A.8 shows additional results on the attention analysis.

5 Conclusion

In this paper, we present empirical evidence on 14 binary and 3 multi-class text classification tasks that compact transformer models consistently reduce number of labeled data required to build new classifiers in realistic asynchronous active learning settings when compared to larger models. Smaller models take less time for training and inference, and allow active learning query acquisition module to select next batch of informative instances more frequently, thereby allowing the classifier to learn fast. Further, we conclude that pre-finetuning helps compact models to learn even faster.

References

Use amazon sagemaker ground truth to label data https://docs.aws.amazon.com/sagemaker/latest/dg/sms.html. In *Amazon Web Services*.

Armen Aghajanyan, Anchit Gupta, Akshat Shrivastava, Xilun Chen, Luke Zettlemoyer, and Sonal Gupta. 2021. Muppet: Massive multi-task representations with pre-finetuning. *CoRR*, abs/2101.11038.

Michael Bloodgood and K. Vijay-Shanker. 2014. A method for stopping active learning based on stabilizing predictions and the need for user-adjustable stopping. *CoRR*, abs/1409.5165.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of bert's attention. *CoRR*, abs/1906.04341.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. 2017. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pages 1183–1192. PMLR.

Andrea Galassi, Marco Lippi, and Paolo Torroni. 2019. Attention, please! A critical review of neural attention models in natural language processing. *CoRR*, abs/1902.02181.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks.

Sheng-Jun Huang and Zhi-Hua Zhou. 2013. Active query driven by uncertainty and diversity for incremental multi-label learning. In *2013 IEEE 13th International Conference on Data Mining*, pages 1079–1084.

Sheng-Jun Huang, Chen-Chen Zong, Kun-Peng Ning, and Hai-Bo Ye. 2021. Asynchronous active learning with distributed label querying. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2570–2576. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not explanation. *CoRR*, abs/1902.10186.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling BERT for natural language understanding. *CoRR*, abs/1909.10351.

Gaël Letarte, Frédérik Paradis, Philippe Giguère, and François Laviolette. 2018. Importance of self-attention for sentiment analysis. In *BlackboxNLP@EMNLP*.

David D Lewis and William A Gale. 1994. A sequential algorithm for training text classifiers. In *SIGIR'94*, pages 3–12. Springer.

Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. 2018. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Jason Phang, Thibault Févry, and Samuel R. Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *CoRR*, abs/1811.01088.

Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel R. Bowman. 2020. Intermediate-task transfer learning with pretrained models for natural language understanding: When and why does it work?

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Sofia Serrano and Noah A. Smith. 2019. Is attention interpretable? *CoRR*, abs/1906.03731.

Burr Settles. 2009. Active learning literature survey.

Jingyu Shao, Qing Wang, and Fangbing Liu. 2019. Learning to sample: an active learning framework. *CoRR*, abs/1909.03585.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: The impact of student initialization on knowledge distillation. *CoRR*, abs/1908.08962.

Shikhar Vashishth, Shyam Upadhyay, Gaurav Singh Tomar, and Manaal Faruqui. 2019. Attention interpretability across NLP tasks. *CoRR*, abs/1909.11218.

A Appendix

A.1 Transformer models in study

Table 2 introduces the transformer models used in this study. We choose the BERT based models in our study as they are well-established models, and we focus on investigating the impact of different model sizes in the asynchronous active learning settings. The study can be easily extended to other transformer based models (e.g, RoBERTa)

A.2 Pre-finetuning dataset

Table 3 illustrates the dataset that we used to pre-finetune transformers.

Table 4 shows dataset statistics for the supervised pre-finetuning.

A.3 multi-label softmax

Due to the multi-label property of the pre-finetuning data, we formulate the pre-finetuning as a multi-label classification problem using multi-label softmax loss function recommended by Mahajan *et al* (Mahajan et al., 2018). multi-label softmax loss computes probabilities over all labels in the label space using a softmax activation and is trained to minimize the cross-entropy between the predicted softmax distribution and the target distribution of each instance. The target is a vector with k non-zero entries, each set to $1/k$ corresponding to the $k \geq 1$ labels for the instance. We experimented both the conventional per-class sigmoid outputs with binary cross entropy loss and multi-label softmax. Results show using multi-label softmax loss function improves the top-1 accuracy by 20% compared with the model using per-class sigmoid outputs with binary cross entropy loss.

A.4 Active learning strategy

To measure the label efficiency of a classifier $\mathcal{M}$, we employ pool-based active learning setting (Algorithm 1) (Lewis and Gale, 1994)(Settles, 2009)(Gal et al., 2017), consisting of a seed set of labeled instances $(x, y) \in \mathcal{D}_{\text{seed}}$ ($|\mathcal{D}_{\text{seed}}| = 100$) to initialize the classifier $\mathcal{M}$ in the first iteration, an unlabeled pool of data $\mathcal{D}_{\text{pool}}$, and a query acquisition function $\mathcal{A}(x, \mathcal{M})$ that ranks the next set of unlabeled instance $x \in \mathcal{D}_{\text{pool}}$ to be sent to the annotator. In each active learning iteration, we perform two operations: 1) judiciously select a subset of unlabeled instances and send them to the annotator, and 2) train a classifier using new and the previously labeled instances. To pick the next set of unlabeled instances for annotation, we take the trained classifier to perform inference on all instances in unlabeled data pool, rank them based on entropy score (calculated from the model prediction score) and select instances which have high entropy score (these are confusing instances that are most likely to increase the accuracy of the classifier in the next active learning iteration). We continue active learning until convergence criteria are achieved. We use two convergence criteria for experiments which exactly mimics a production setup: 1) the estimated recall/precision for each class-of-interest should be equal to or greater than the corresponding business-specified targets, and 2) predictions on unlabeled data should have stabilized (Bloodgood and Vijay-Shanker, 2014).

Algorithm 1: Active learning setup used in production and for experiments.

Input: Pool of unlabeled data $\mathcal{D}_{\text{pool}}$, Batch size B, Initial labeled dataset $\mathcal{S}$ (could be empty), Business targets on precision and recall
Output: Trained classifier $\mathcal{M}$ that meets business targets
while *convergence criteria are not met* **do**
  Rank instances from $\mathcal{D}_{\text{pool}}$ using query strategy $\mathcal{A}(x, \mathcal{M})$.
  Label top-ranked B instances, add them to $\mathcal{S}$. (*Comment*: In production, human annotators provide labels, while in experiments a bot does the labeling job)
  Remove $\mathcal{S}$ from unlabeled data $\mathcal{D}_{\text{pool}} \leftarrow \mathcal{D}_{\text{pool}} \setminus \mathcal{S}$.
  (asynchronous process) Train the classifier $\mathcal{M}$ on $\mathcal{S}$.
  (asynchronous process) Predict $\mathcal{D}_{\text{pool}}$ using the classifier.
  (asynchronous process) Estimate current precision/recall using out-of-fold scores (k-fold cross validation)
end

A.5 Data distribution in multi-class datasets

Table 5 presents the data distribution of the three multi-class datasets.

Architecture	Description
BERT-24L-P	It is a internal BERT-based language model (84 million parameters). It is pre-trained on multilingual, multi-locale and multimodal (text + structured fields) e-commerce product data, and then pre-finetuned using the multi-task soft labels generated from the previously pre-trained teacher model with 500 million parameters.
BERT-6L	It is widely known as DistilBERT (Sanh et al., 2019), which is a 6-layer transformer model unsupervised pre-trained using English Wikipedia and Toronto Book Corpus.
BERT-4L	It is known as TinyBERT (Jiao et al., 2019). We used the 4 layer TinyBERT, which is distilled from the 12-layer BERT teacher model unsupervised pre-trained using English Wikipedia and Toronto Book Corpus.
BERT-2L	It is a 2-layer transformer model (Turc et al., 2019) unsupervised pre-trained with a masked language modeling objective on Book Corpus and English Wikipedia.
BERT-1L	In order to minimize the model size and investigate the impact of small transformer model, we extracted the first transformer layer from the BERT-2L and made it as a BERT-1L model.

Table 2: Various transformer models under study (The model artifacts of BERT-6L, BERT-4L, BERT-2L used in this study are available in HuggingFace.).

Classifier→ product instance↓	C1	C2	C3	...	C2664
A1	1	0	0	...	0
A2	0	1	1	...	0
...	...	...	...	...	...
An	1	0	0	...	0

Table 3: Illustration of dataset used to pre-finetune transformers. The rows of the table represent product instances, and the columns represent the 2664 binary classifiers used in our pre-finetuning. Each cell represents the membership of an product instance to a class. For example, if the class "C1" was created to identify "Face Masks", then product instance A1 was classified as a FaceMask and product instance A2 was not.

L	n_{trn}	n_{val}	n_{tst}	$\bar{n}$	$\bar{L}$
2664	4,607,904	918,894	820,839	2,383	2.62

Table 4: Statistics of the supervised pre-finetuning dataset. L is number of classification tasks. n_{trn}, n_{val}, and n_{tst} refer to number of instances in training, validation and test dataset, respectively. $\bar{n}$ is average number of instances per classification task and $\bar{L}$ is average number of labeled data per instance (product data)

Table 5: Multi-class datasets (*HS*, *FEE*, and *AB*) have long-tail distribution in terms of class sizes. S denotes the relative size of each class. *E.g.*, the class HS-2 has 41.88% of 1.15M products in HS dataset.

HS (1.15M)		FEE (750K)		AB (8.4M)	
Class	$S(\%)$	Class	$S(\%)$	Class	$S(\%)$
HS-0	0.52	T-0	50.56	A-0	18.39
HS-1	0.36	T-1	6.36	A-1	46.58
HS-2	41.88	T-2	30.84	-	-
HS-3	25.81	T-3	10.69	-	-
HS-4	0.30	T-4	0.53	-	-
HS-5	0.44	T-5	0.62	-	-
HS-6	2.36	T-6	0.28	-	-
HS-7	11.10	-	-	-	-
HS-8	14.65	-	-	-	-
not-in-k	2.59	not-in-k	0.12	not-in-k	35.03

Hyperparameters	Pre-finetuning	Finetuning
Loss function	Multi-label softmax	Softmax
Train batch size	32	32
Val batch size	32	-
Dropout factor	0.1	0.3
Max seq length	128	128
Optimizer	AdamW	AdamW
Learning rate	$2e^{-5}$	$2e^{-5}$ /$1e^{-4}$
Weight decay	0.01	0.01
Early stopping	True	False
Max training epoch	10	5

Table 6: Hyperparameters we used in the pre-finetuning and finetuning stages to train various transformer models. In downstream task finetuning, we use learning rate $2e^{-5}$ for BERT-6L and ProdNet-6L, and learning rate $1e^{-4}$ for rest of benchmarking transformer models. In order to fully utilize the labeled data for training, we do not use validation dataset in downstream task finetuning.

A.6 Hyperparameters

Table 6 is the hyperparameters used in pre-finetuning and finetuning various transformer models.

A.7 Per-task experiment results

Table 7 and Table 8 shows 1) the relative number of labeled data required for various active learning experiments to converge, and 2) the relative (macro) recall/(macro) precision of the models measured on data pool $\mathcal{D}_{\text{pool}}$, in the 14 binary classification tasks and 3 multi-class classification tasks, respectively.

A.8 Why does ProdNet learn faster than the non-pre-finetuned counterpart?

To obtain a global understanding if pre-finetuning helps, we aggregate attention weights computed on 500 positive examples for four classification tasks (listed in Table 7). Specifically, we calculate the average attention for the key phrases a human expert deemed most critical to the classification task. For example, for PetCare products, the key phrases identified are *flea*, *tick*, *worm*, and *parasites*. The results are shown in Table 9. We observe that in general ProdNet-2L is able to pay more attention to the key tokens than BERT-2L. We hypothesize that pre-finetuning on relevant e-commerce product data helps the model understand potentially important words for the downstream task, thereby allowing the model to learn the downstream task faster.

Models→	ProdNet-1L	BERT-1L	ProdNet-2L	BERT-2L	ProdNet-24L-P	BERT-24L-P
Tasks ↓ (R/P target)	Top: relative number of labeled data Bottom: relative recall(R)/precision(P) computed on data pool					
Speakers (r/p)	$\boldsymbol{x}$ 1.11r/p	2.56x 1.10r/1.02p	2.33x 1.11r/1.01p	2.57x 1.07r/p	1.76x 1.11r/1.02p	2.7x 1.11r/1.03p
Pillows (r/p)	$\boldsymbol{x}$ 1.09r/1.01p	1.43x 1.1r/1.01p	1.09x 1.07r/1.01p	2.05x 1.02r/1.02p	1.08x 1.11r/1.02p	1.74x 1.11r/1.03p
Plants (r/p)	$\boldsymbol{x}$ 1.01r/1.02p	1.80x r/1.03p	2.12x r/1.03p	5.87x r/1.04p	1.32x r/1.03p	1.47x 1.01r/1.03p
Bottle (r/p)	$\boldsymbol{x}$ 1.34r/1.01p	1.74x 1.38r/1.01p	1.52x 1.28r/p	2.94x 1.32r/1.01p	1.10x 1.32r/p	1.28x 1.3r/1.03p
Jackets (r/p)	x 1.07r/1.02p	1.32x 1.06r/1.04p	1.79x 1.08r/1.02p	2.40x 1.08r/p	**0.99$\boldsymbol{x}$** 1.1r/p	1.20x 1.13r/1.01p
Belt (r/p)	$\boldsymbol{x}$ 1.06r/1.01p	1.34x 1.05r/1.02p	2.01x 1.06r/1.02p	3.85x 1.05r/1.02p	1.66x 1.06r/1.04p	1.75x 1.06r/1.03p
Postcard (r/p)	$\boldsymbol{x}$ 1.16r/1.01p	1.37x 1.16r/1.01p	2.10x 1.17r/p	3.10x 1.15r/1.02p	1.47x 1.16r/1.02p	1.55x 1.16r/1.03p
PetCare (r/p)	$\boldsymbol{x}$ 1.12r/1.02p	1.57x 1.12r/1.03p	1.25x 1.12r/1.01p	2.12x 1.12r/1.02p	1.74x 1.12r/1.04p	2.17x 1.12r/1.04p
T39253 (r/p)	$\boldsymbol{x}$ 1.06r/p	2.21x 1.07r/1.01p	1.45x 1.04r/p	2.31x 1.05r/p	1.72x 1.08r/1.01p	1.78x 1.08r/1.02p
FireStarter (r/p)	$\boldsymbol{x}$ 1.08r/1.02p	2.98x 1.08r/1.03p	1.22x 1.08r/1.01p	2.01x 1.08r/1.02p	1.89x 1.08r/1.03p	2.32x 1.08r/1.04p
Batteries (r/p)	x 1.25r/p	1.65x 1.24r/1.01p	0.92x 1.14r/1.01p	5.15x 1.24r/1.01p	**0.81$\boldsymbol{x}$** 1.28r/1.01p	1.52x 1.28r/1.03p
Radio (r/p)	$\boldsymbol{x}$ 1.04r/1.04p	1.95x 1.01r/1.04p	1.55x 1.03r/1.04p	2.55x r/1.04p	1.34x 1.02r/1.04p	1.40x 1.03r/1.04p
GDevices (r/p)	x 1.12r/1.01p	2.02x 1.13r/1.02p	**0.96$\boldsymbol{x}$** 1.09r/1.01p	2.40x 1.13r/1.01p	1.29x 1.13r/1.02p	1.45x 1.13r/1.02p
Extg (r/p)	$\boldsymbol{x}$ 1.06r/1.01p	2.37x 1.07r/1.03p	1.27x 1.02r/1.01p	3.51x 1.05r/1.02p	1.34x 1.07r/1.03p	1.46x 1.07r/1.03p

Table 7: For each classification task in the first column, we report the relative number of labeled data required to converge the active learning process by various classifiers. We also report the relative recall and precision measured on data pool $\mathcal{D}_{\text{pool}}$ as "recall/precision". For each classification task, we denoted the recall/precision targets set by customers as (r/p), and the number of labeled data required to converge with ProdNet-1L classifier as x. (Note: different classification tasks may have different recall/precision targets, and they require different number of labeled data to converge the experiments with ProdNet-1L classifier. Since we do not compare performance across different classification tasks, we use the same letters for the denotation of different tasks.) For example, on Speakers dataset the recall/precision targets are set as (r/p), the ProdNet-1L classifier achieved 1.11r recall at p precision when the active learning experiment converged at x labels, while the BERT-1L classifier achieved 1.10r recall at 1.02p precision when the active learning experiments converged at 2.56x labels. For each classification task, we highlight the least number of labeled data required for convergence in **bold**.

Models→	ProdNet-1L	BERT-1L	ProdNet-2L	BERT-2L	ProdNet-24L-P	BERT-24L-P
Tasks ↓ (R/P target)	Top: relative number of labeled data Bottom: relative macro-recall/macro-precision computed on data pool					
HS (r/p)	$\boldsymbol{x}$ 1.15r/1.13p	2.71x 1.16r/1.17p	2.30x 1.13r/1.12p	3.48x 1.13r/1.11p	1.08x 1.16r/1.17p	1.09x 1.16r/1.17p
FEE (r/p)	$\boldsymbol{x}$ 1.06r/1.07p	1.84x 1.05r/1.08p	1.06x 1.05r/1.07p	3.12x 1.05r/1.06p	1.32x 1.06r/1.07p	1.35x 1.05r/1.08p
AB (r/p)	$\boldsymbol{x}$ 1.08r/p	1.60x 1.12r/p	2.39x 1.04r/p	3.89x 1.08r/p	2.44x 1.08r/1.01p	2.73x 1.05r/1.02p

Table 8: For each multi-class classification task in the first column, we report the relative number of labeled data required to converge the active learning process by various classifiers. We also report relative macro-recall and macro-precision measured on data pool $\mathcal{D}_{\text{pool}}$. For example, on HS dataset the macro-recall/macro-precision targets are set as (r/p), the ProdNet-1L classifier achieved 1.15r (macro-recall) at 1.13p (macro-precision) when the active learning experiment converged at x labels, while the BERT-1L classifier achieved 1.16r (macro-recall) at 1.17p (macro-precision) when the active learning experiment converged at 2.71x labels. For each classification task, we highlight the least number of labeled data required for convergence in **bold**.

Models→		ProdNet-2L		BERT-2L	
Class rationale ↓	Key token	Average attention	Token ranking	Average attention	Token ranking
Task: PetCare					
To classify pet-care products designed to prevent fleas, ticks, ringworm, or other parasites.	flea	**0.45**	**15%**	0.36	50%
	worm	**0.44**	**16.7%**	0.42	37.7%
	tick	**0.43**	**17.4%**	0.39	44.3%
	parasites	**0.38**	**24.2%**	0.33	56.2%
Task: Speakers					
To classify wireless battery operated portable speakers.	blue	**0.70**	**4.9%**	0.39	52.6%
	portable	**0.60**	**11.2%**	0.37	57.2%
	speaker	**0.40**	**35.3%**	0.30	69.7%
	wireless	0.29	52.1%	**0.48**	**34.7%**
Task: Pillow					
To classify pillows and cushions with feather fillings or inserts.	cushion	**0.42**	**23.1%**	0.25	68.4%
	pillow	**0.60**	**24.0%**	0.25	65.9%
	feather	0.13	75.1%	**0.45**	**32.0%**
Task: Seed					
To classify seeds which are used for growing plants.	seed	**0.61**	**13.0%**	0.31	50.4%
	plant	**0.60**	**13.6%**	0.47	19.6%
	fruits	**0.42**	**36.0%**	0.38	36.1%
	vegetable	**0.27**	**63.4%**	0.24	64.8%

Table 9: Global aggregation of attention values for key phrases of a classification task. We take 500 positive samples to the original ProdNet-2L and BERT-2L models without downstream task finetuning. We calculate the average attention of each token that the `[CLS]` token pays to in the last transformer layer. The average attention (higher the better) received by the identified key tokens is noted in the column *Avg. attention*. We also rank all the tokens in the 500 positive samples based on their average received attention, and calculate the rank of identified key tokens (smaller the better). Key tokens were identified by a trained expert who provided the labeled data required to train the classifiers.