DOWNSTREAM DATASETS MAKE SURPRISINGLY GOOD PRETRAINING CORPORA

Anonymous authors
Paper under double-blind review

ABSTRACT

For most natural language processing tasks, the dominant practice is to finetune large pretrained transformer models (e.g., BERT) using smaller downstream datasets. Despite the success of this approach, it remains unclear to what extent these gains are attributable to the massive background corpora employed for pretraining versus to the pretraining objectives themselves. This paper introduces a large-scale study of *self-pretraining*, where the same (downstream) training data is used for both pretraining and finetuning. In experiments addressing both ELECTRA and RoBERTa models and 10 distinct downstream datasets, we observe that self-pretraining rivals standard pretraining on the BookWiki corpus (despite using around $10\times$–$500\times$ less data), outperforming the latter on 7 and 5 datasets, respectively. Surprisingly, these task-specific pretrained models often perform well on other tasks, including the GLUE benchmark. Our results suggest that in many scenarios, performance gains attributable to pretraining are driven primarily by the pretraining objective itself and are not always attributable to the incorporation of massive datasets. These findings are especially relevant in light of concerns about intellectual property and offensive content in web-scale pretraining data.

1 INTRODUCTION

For training predictive models operating on natural language data, the current best practice is to *pretrain* models on large unlabeled *upstream* corpora to optimize self-supervised objectives, for example, masked language modeling (MLM); the resulting weights are then used to initialize models that are subsequently trained (*finetuned*) on the labeled *downstream* data available for the task at hand. Large-scale pretrained models typically provide significant performance boosts when compared to models trained directly on the downstream task (with random initializations) (Peters et al., 2018; Devlin et al., 2019; Chiang & Lee, 2020; Krishna et al., 2021). Upstream corpora tend to be significantly larger than the downstream corpora and the success of this approach is often attributed to its ability to leverage these massive upstream corpora (Liu et al., 2019; Yang et al., 2019). For example, the seminal BERT model (Devlin et al., 2019) was pretrained using the BookWiki corpus which is a combination of English Wikipedia and BooksCorpus (Zhu et al., 2015), totaling 13GB of plain text. Subsequent models have moved on to web-scale data. For example, XLNet (Yang et al., 2019), RoBERTa (Liu et al., 2019), and T5 (Raffel et al., 2020)), were trained on 158GB, 160GB and 750GB of data, respectively.

As upstream corpus size and downstream performance have gone up, popular attempts at explaining these gains have focused on themes of "knowledge transfer" from the upstream corpus, attributing them to shared linguistic structure, semantics (Lina et al., 2019; Tenney et al., 2019), and facts about the world (Petroni et al., 2019). However, since the introduction of large-scale pretraining corpora occurred together with the invention of self-supervised pretraining objectives (e.g. masked language modeling (Devlin et al., 2019) and replaced token detection (Clark et al., 2019)), it remains unclear to what extent large-scale corpora are integral to these leaps in performance. For several tasks, especially summarization, recent works have managed to achieve surprising performance gains in settings where the upstream corpus is created synthetically with arbitrary symbols, but the pretraining objective is designed to capture some of the structure of the task (Krishna et al., 2021; Wu et al., 2022).

In this work, we ask just how much of pretraining's benefits could be realized in the absence of upstream corpora by pretraining directly on the downstream corpora (with the same self-supervised

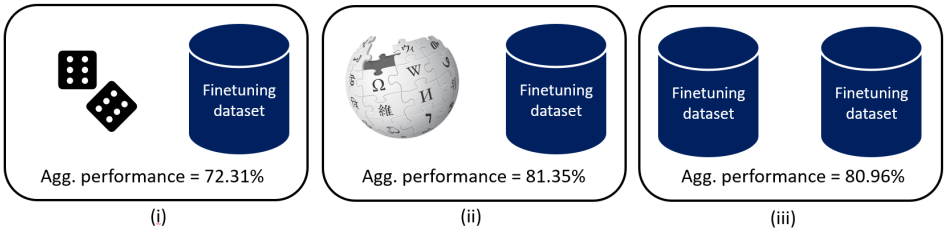

Figure 1: Aggregate performance of an ELECTRA model across 10 finetuning datasets when it is (i) randomly initialized (ii) pretrained on upstream corpus (BookWiki) (iii) pretrained on the finetuning dataset itself

objectives). We find that this approach, which we call *self-pretraining*, often rivals the performance boosts conferred by *off-the-shelf* models pretrained on large upstream corpora (Figure 1), even outperforming them on 7 out of 10 datasets. While prior research has shown that *additional* self-supervised pretraining of off-the-shelf models using the downstream data can give further gains (Gururangan et al., 2020), our study goes further, showing that even when starting from random initializations, self-pretraining can rival standard practices. Since self-pretraining requires the same data that must already be available for downstream finetuning, the benefits of pretraining in this case cannot be attributed to *knowledge transfer*. Instead, these benefits can only be attributed to the pretraining objective, which, via some unknown mechanism, primes the model parameters such that subsequent finetuned models generalize better to the underlying population. While we note that similar observations have been made in the computer vision community (El-Nouby et al., 2021), we argue that it is especially important to establish these phenomena in the language domain, for which building on self-supervised pretrained models is now the ubiquitous practice of the vast majority of practitioners.

To understand differences in predictions with different pretraining strategies (i.e., between self-pretrained and off-the-shelf models), we analyse the errors made by these models on the same downstream data. Despite similar performance of these models, we find that self-pretrained and off-the-shelf models make significantly less correlated errors when compared to two independently finetuned models pretrained with either strategy. However, we observe that these uncorrelated mistakes do not transfer to improvements in the ensemble performance.

We find that models pretrained on one downstream dataset often perform surprisingly well when finetuned to other downstream datasets. Notably, the downstream datasets in our study come from a wide variety of domains such as news, online forums, tweets, reviews etc (Table 1). Nevertheless, we find that pretraining on any of these downstream datasets delivers significant performance gains on most datasets (greater than half of off-the-shelf model's gains in 88% of cases) irrespective of domain. However, the best performance on a downstream dataset is usually achieved by the model pretrained on that dataset itself. Models pretrained on downstream datasets perform well on the GLUE benchmark too, despite having considerably less long-term dependencies as compared to standard upstream corpora. For example, the MNLI corpus consists of 2-sentence input texts that are concatenated in random order. However, we note that further reducing the length of dependencies by shuffling the sentences in the self-pretraining corpus reduces the GLUE score significantly.

We propose a heuristic that, given a corpus of randomly ordered sentences, simply re-orders them to produce a corpus which can achieve a higher GLUE score. Our heuristic creates a corpus as follows: starting with a random sentence, we greedily append to the corpus (without replacement) by choosing the available sentence with maximal TF-IDF cosine similarity to the most recently appended sentence. This heuristic, when applied to our downstream datasets, leads to increases of up to 13 GLUE points, and sometimes performs even better than the corpus with the original order of sentences in the dataset.

Overall, our contributions can be summarized as follows:

- Demonstration of self-pretraining's efficacy across 10 downstream tasks and two pretraining techniques (ELECTRA and RoBERTa), with comparisons to off-the-shelf pretrained models.
- Analysis of the out-of-distribution performance of models pretrained on one downstream dataset and finetuned on other downstream datasets including the GLUE benchmark.

- Examination of the role of long-term dependencies in pretraining text towards increasing the GLUE score, along with a simple heuristic to introduce them artificially, leading to performance gains.

2 RELATED WORK

Self-Pretraining in Computer Vision Most relevant to our work, two recent/concurrent works in computer vision explore self-pretraining (He et al., 2022; El-Nouby et al., 2021). In a contemporary work, He et al. (2022) showed that pretraining with a Masked AutoEncoder (MAE) objective (analogue of MLM objective for image datasets) significantly boosts the performance of ViT models on the Imagenet-1K dataset. In another related paper, El-Nouby et al. (2021) showed that pretraining solely on small-scale downstream datasets for object detection and segmentation tasks reaches the performance of Imagenet-pretrained models. Our work establishes that a similar phenomenon is observed for NLP tasks too across a wide range of datasets.

Pretraining on Downstream Data in NLP *Task-Adaptive PreTraining* (TAPT (Gururangan et al., 2020)) consists of taking off-the-shelf pretrained models like BERT and RoBERTa and engaging in further pretraining on the downstream datasets before finetuning them to the task at hand. TAPT has been shown to improve performance of off-the-shelf models in a variety of works (Logeswaran et al., 2019; Han & Eisenstein, 2019; Chakrabarty et al., 2019). By contrast, our work pretrains models *only* on the downstream dataset, enabling a head-to-head comparison between the performance of off-the-shelf and self-pretrained models, and (in some situations) challenging the necessity of upstream corpora altogether.

Claims about Knowledge transfer Many works claim that pretraining extracts generally useful *knowledge* from the upstream corpus such as linguistic patterns (Lina et al., 2019; Tenney et al., 2019; Manning et al., 2020) and facts (Petroni et al., 2019), and that this accounts for the performance gains that they enjoy on downstream tasks. Several works, e.g., in the *probing* literature (Tenney et al., 2019; Manning et al., 2020; Petroni et al., 2019), demonstrate that from the internal representations of a model, it is easy (e.g., via linear models) to predict certain linguistic features or real-world facts. However, these studies do not clarify the mechanism by which these observations relate to performance gains on downstream tasks. Tenney et al. (2019) recognizes this limitation, stating *"the observation of a (linguistic) pattern does not tell us how it is used"*. Our work suggests that to the extent that such knowledge extraction plays a role in pretraining's benefits, sufficient knowledge is often present in the downstream dataset and need not be *transferred* from huge upstream corpora.

Challenges to the Knowledge Transfer Narrative Multiple previous works have questioned whether knowledge transfer can fully account for the efficacy of pretraining. Improvements in performance on downstream NLP tasks have resulted from pretraining on other modalities like music and code (Papadimitriou & Jurafsky, 2020), sequences of meaningless symbols (Chiang & Lee, 2020; Krishna et al., 2021; Wu et al., 2022), and language denatured via shuffling of words (Sinha et al., 2021). On the other hand, models pretrained on language have shown improved performance on tasks dealing with other modalities such as image classification Lu et al. (2021) and reinforcement learning for games Reid et al. (2022). By contrast, we show that without surplus upstream data of any modality, self-pretraining alone can often perform comparably or even better than standard pretraining with a large upstream corpus. In a similar vein with these papers, our work suggests that a large portion of pretraining's success may come from alternative, unexplored mechanisms which have more to do with the pretraining objective than knowledge transfer from upstream corpora.

3 EXPERIMENTAL SETUP

Our experiments center around the ELECTRA model (Clark et al., 2019) and the RoBERTa-base model (Liu et al., 2019). On the broadest set of experiments, for which we can only afford to train one model, we employ ELECTRA because it performs better than BERT/RoBERTa given comparable compute budgets (Clark et al., 2019). In particular, we use the small variant of ELECTRA (14 million parameters), which performs similarly to BERT-base on GLUE (difference of about 2 points) while training much faster (Clark et al., 2019). However, we replicate many of these results on the larger RoBERTa-base model revealing similar results and thus establishing the generality of our findings.

Dataset	Size (MB)	Classes	Domain	Task
AGNews (Zhang et al., 2015)	27	4	News	topic classification
QQP (Wang et al., 2018)	43	2	Online forum questions	paraphrase detection
Jigsaw Toxicity (Kaggle.com, 2018)	59	6	Wikipedia comments	toxicity detection
MNLI (Williams et al., 2018)	65	3	Diverse	natural language inference
Sentiment140 (Go et al., 2009)	114	5	Tweets	sentiment classification
PAWS (Zhang et al., 2019)	139	2	Wikipedia	paraphrase detection
DBPedia14 (Zhang et al., 2015)	151	14	Wikipedia	topic classification
Discovery (Sileo et al., 2019)	293	174	Web crawl	discourse marker prediction
Yahoo Answertopics (Zhang et al., 2015)	461	10	Online forum answers	topic classification
Amazon Polarity (Zhang et al., 2015)	1427	2	Product reviews	sentiment classification

Table 1: The suite of downstream datasets used in this work along with their training set sizes

During pretraining, a text sequence is fed into the model with some tokens masked out. While MLM-only models like RoBERTa only have a *generator* network that predicts the content of the masked tokens, ELECTRA has an additional discriminator module that predicts if those predictions were correct. Both the generator and the discriminator networks' parameters are updated simultaneously during pretraining. After pretraining, the generator is discarded and the discriminator is used as an encoder for finetuning on downstream tasks.

We experimented with 10 different downstream datasets (Table 1). We chose these datasets in our testbed to span different dataset sizes ranging from 27 megabytes to about 1.4 gigabytes of text in the training split. These datasets are for different tasks such as topic classification, sentiment classification, natural language inference etc., and are created using data sourced from diverse domains. Most of them are multi-class classification tasks except Jigsaw Toxicity which is a multi-label classification task, and Sentiment140 which is modeled as a regression task. For finetuning a pretrained model on any dataset, we passed the input through the model, took the vector representation of the CLS token in the final layer, and passed it through a classification head with one hidden layer to get the output.

4 SELF-PRETRAINING PERFORMANCE

In our first set of experiments, we compare self-pretraining's performance with other pretraining techniques. For each dataset, we pretrain an ELECTRA model on text from its training split and then finetune it on the same training data using the associated labels. To create a pretraining corpus from a downstream dataset, we concatenate the input text from each of the examples, assembling them in random order. We evaluate the performance of each finetuned model on the corresponding dataset's test split[1]. For all datasets, we evaluate performance by accuracy, except for Sentiment140 and Jigsaw Toxicity, for which we use Pearson correlation and micro-averaged AUC scores, respectively (these are not multi-class classification problems).

Notably, all self-pretrained models deliver significant performance boosts on their respective datasets (Table 2), and over half of them perform even better than the off-the-shelf model. We measured a model's *benefit* as the performance boost that it achieves over a randomly initialized model, divided by the boost achieved by the off-the-shelf ELECTRA model against the same baseline. The average benefit of self-pretraining across all datasets is 103.70%. We do not see a clear correlation between the size of the dataset and the performance of self-pretraining. For example, the highest benefit of 131.33% is achieved for the smallest dataset (AGNews), which is merely 27MB in size, while the minimum benefit is achieved on the Discovery dataset, which is the third largest dataset measuring 293MB.

We also evaluated the alternate pretraining technique *TAPT* as described in Gururangan et al. (2020). In this technique, we take the off-the-shelf ELECTRA model, which has already been pretrained on the upstream BookWiki corpus, and further pretrain it on the downstream dataset for 100 epochs. Self-pretraining outperforms TAPT on 6 datasets, notably including the two datasets where it outperformed the off-the-shelf models by the greatest benefit margin - *AGNews* and *Yahoo Answertopics*. Interestingly, TAPT performs worse than off-the-shelf model on the same 3 datasets where self-pretraining performs worse than off-the-shelf model (except Sentiment140). None of the three pretraining approaches seem to be uniformly better than any other.

[1]For QQP and MNLI we just use the validation split because test set labels are private.

Finally, we also evaluate the self-pretrained models on the GLUE benchmark and report results on the dev set [2]. The performance of the models on their pretraining dataset does not seem to be correlated with its GLUE score. The GLUE score also does not monotonically go up with increasing dataset size, indicating that the data domain makes some difference. For example, the Amazon Polarity corpus scores just 66.14 on GLUE despite being about 1.4 gigabytes in size, while AGNews which is 27MB in size, scores 74.30. The highest GLUE score is achieved by pretraining on Yahoo Answertopics.

Dataset	**Size(MB)**	**RandInit**	**SelfPretrain**	**Off-the-shelf**	**Benefit %**	**TAPT**	**GLUE**
AGNews	27	91.75	94.34	93.72	131.33	94.07	74.30
QQP	43	82.93	90.66	90.34	104.34	90.64	75.43
Jigsaw Toxicity	59	97.83	98.49	98.53	94.99	98.48	76.65
MNLI	65	65.49	78.39	82.29	76.77	79.26	78.28
Sentiment140	114	63.75	67.04	66.95	102.91	65.65	72.67
PAWS	139	50.00	97.53	97.30	100.49	97.85	74.65
DBPedia14	151	98.59	99.22	99.11	121.17	99.23	70.38
Discovery	293	17.00	22.38	24.55	71.22	23.58	77.26
Yahoo Answertopics	461	61.94	65.26	64.55	127.31	65.05	79.53
Amazon Polarity	1427	93.86	96.27	96.13	106.49	96.16	66.14

Table 2: Performance of ELECTRA-small models pretrained with different techniques on various downstream datasets and on the GLUE benchmark (dev set). For reference, a randomly initialized model scores 53.20 and the off-the-shelf model scores 79.43 on GLUE.

5 Cross Dataset finetuning

In this set of experiments, we investigated if the models pretrained on a dataset are only useful for that specific task, or are they useful across the whole spectrum of tasks that we consider. We took each model pretrained on a dataset in our testbed and finetuned and evaluated it on all other datasets in the testbed. The performance benefits provided in all cases are shown as a heatmap in Figure 2.

We found that for almost all downstream datasets, pretraining on any other dataset provides significant advantage (Figure 2). In most cases, pretraining on the downstream dataset itself performs the best. Among datasets where self-pretraining performs better than off-the-shelf model (i.e. the diagonal entry is greater than 1), pretraining on datasets of larger size does not help further. However, for the datasets where self-pretraining's benefit is much less than 100% (i.e. MNLI and Discovery), pretraining on a larger dataset (e.g., Yahoo Answertopics) performs better than self-pretraining.

Among all the pretrained models, a few models perform consistently good or bad across different downstream datasets (Figure 2). For example, the model pretrained on Yahoo Answertopics gets the highest average score of 0.90 across all datasets, while the PAWS-pretrained model gives the lowest aggregate score of 0.64. Similarly, there are downstream datasets that are benefited consistently by either a large or a small margin by pretraining on different datasets. For example, performance on QQP and PAWS receives huge boosts by pretraining on most datasets. In contrast, performance on sentiment140 is low for most pretrained models, even dropping below 20% for 3 pretraining datasets.

Next, we perform an ablation to investigate that given a fixed dataset to finetune on, is it better to pretrain on the *exact* same data (i.e., using the same set of inputs), or is it better to pretrain on different data with an identical distribution. To test this hypothesis, we divided downstream datasets into two subsets (denoted as A and B) by randomly splitting the training dataset into half. We pretrained one model on each subset and then finetuned them on both subsets separately. The validation and test sets used for finetuning are the same as in the original dataset.

We do not see any consistent benefits with pretraining and finetuning on the same dataset (Table 3). Instead, we found consistent patterns where models pretrained on one split (either A or B) outperformed models pretrained on the other, irrespective of the split used for finetuning. This suggests that the pretraining data has greater influence on the final performance than the finetuning data. Additionally, we observe that finetuning the superior pretrained model, using the downstream split other than the one used for pretraining, performs the best, suggesting overall exposure to more data helps.

[2]Following Clark et al. (2019) we exclude the WNLI task from the results.

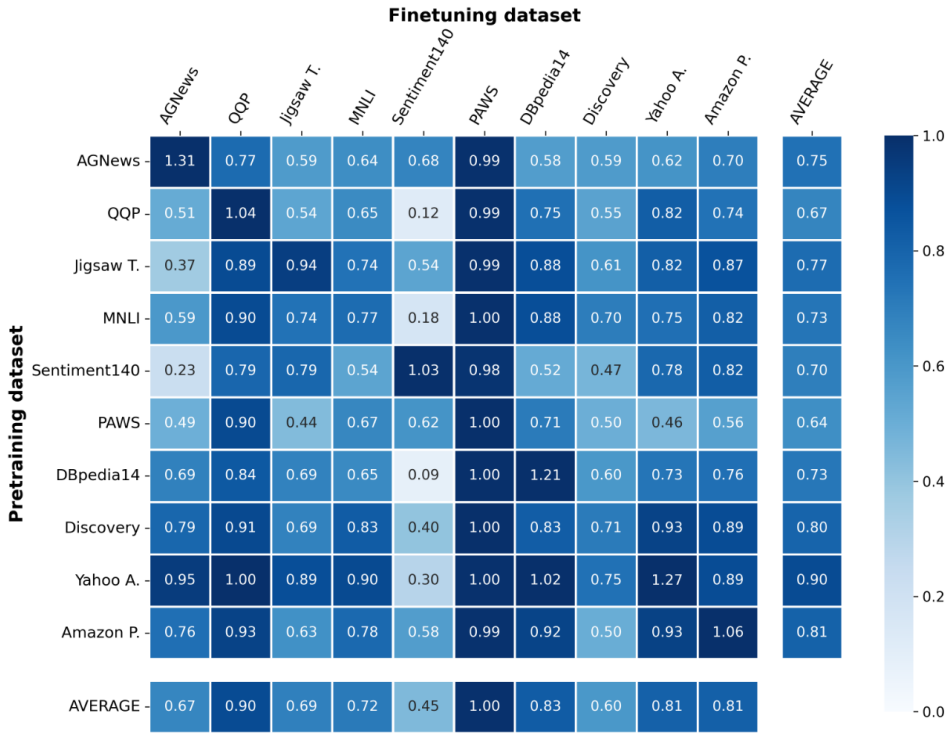

Figure 2: Performance benefits of models pretrained on each dataset, upon finetuning on each downstream dataset. Each value is the ratio of performance gains achieved by model pretrained on the row's dataset vs off-the-shelf model, relative to random initialization, upon finetuning on the column's dataset.

MNLI			QQP			Discovery			Yahoo Answertopics		
	A	B		A	B		A	B		A	B
A	76.00	**76.42**	A	84.28	84.79	A	18.78	18.61	A	64.18	**64.34**
B	75.93	75.05	B	**88.73**	88.41	B	**19.99**	19.98	B	64.09	64.18

Table 3: Performance when splitting the dataset into two equal-sized subsets A and B and then pretraining on one (row) and finetuning on another (column)

6 Difference in outputs of self-pretrained and off-the-shelf models

Since self-pretrained models and off-the-shelf models perform similarly in terms of classification accuracy, a natural question to ask is: *do these models make errors on the same set of inputs?* To answer this question, we investigate the difference in predictions made by models pretrained with different strategies across all multi-class classification tasks. In particular, given model f_A and f_B, we compute *error inconsistency*, defined as follows:

$$\frac{\sum_{i=1}^{n} \left(\mathbb{1}\left[f_A(x_i) \neq y_i \wedge f_B(x_i) = y_i\right] + \mathbb{1}\left[f_A(x_i) = y_i \wedge f_B(x_i) \neq y_i\right]\right)}{n},$$

where $\{x_i, y_i\}_{i=1}^{n}$ is the test set. Intuitively, error inconsistency captures the fraction of examples where exactly one model is correct. This definition has been commonly used to estimate diversity in model prediction (Gontijo-Lopes et al., 2022; Geirhos et al., 2020). Across all the multi-class classification tasks, in addition to computing error inconsistency between self-pretrained and off-the-shelf model, for baseline comparison, we also tabulate error inconsistency between: (i) two independently finetuned versions of a self-pretrained model; and (ii) two independently finetuned versions of the off-the-shelf model.

	Ensemble Accuracy			Error Inconsistency		
Dataset	2×SelfPretrain	2×Offself	SelfPretrain + Offself	2×SelfPretrain	2×Offself	SelfPretrain + Offself
AGNews	94.66	94.17	94.54	1.76	3.50	4.01
QQP	90.92	90.74	91.63	4.57	5.27	8.91
MNLI	78.51	82.37	82.31	6.94	6.42	14.82
PAWS	97.70	97.45	97.75	0.96	1.30	2.07
DBPedia14	99.28	99.19	99.24	0.38	0.48	0.51
Discovery	22.98	25.25	25.02	7.85	9.18	12.66
Yahoo	65.32	64.69	65.64	5.27	5.49	9.55
Amazon	96.40	96.24	96.51	1.26	1.58	2.48

Table 4: Performance of ensemble models of self-pretrained and off-the-shelf models. For ensembling, we aggregate predictions of models after calibration with Temperature Scaling (Guo et al., 2017). We observe that in most of the datasets, SelfPretrain + Off-the-shelf ensembling does not improve over ensembles of two models with the same pre-training strategy, despite relatively higher error inconsistency of SelfPretrain + Off-the-shelf models.

Compared to error inconsistency between two models with the same pretraining dataset, we observe that models trained with different pretraining datasets have high error inconsistency in predictions (Table 4). Note that for models with comparative performance, high error inconsistency highlights the high disagreement in predictions. This demonstrates that while different pretraining datasets produce similarly performing models in terms of overall accuracy, the model predictions are relatively dissimilar. Our observations here align with investigations in vision tasks, where Gontijo-Lopes et al. (2022) observed that different models trained with different pretraining datasets produced uncorrelated errors.

Since different pretraining datasets produce models with uncorrelated errors, we now seek to ensemble these models to check if uncorrelated mistakes can be converted to a correct prediction. When the models make different predictions, in particular, when one model is correct and another is incorrect, the ensemble prediction will be dominated by the model with higher confidence in their prediction. As before, we consider ensembles of (i) two independently finetuned versions of a self-pretrained model; (ii) two independently finetuned off-the-shelf models; and (iii) a finetuned version each of the self-pretrained and off-the-shelf models.

We make the following observations: First, as expected we observe that ensembling improves model performance as compared to a single model (Table 4). Second, despite having larger error inconsistency, we do not observe any significant improvements in ensembles of self-pretrained and off-the-shelf model as compared to ensembles of two models with the same pretraining strategy (Table 4). This is in contrast with findings on vision tasks where Gontijo-Lopes et al. (2022) observed that larger error inconsistency led to larger improvement in ensemble performance.

7 The Role of Long-term Dependencies in Pretraining Corpora

Virtually all pretrained models like BERT, ELECTRA, XLNet are pretrained using documents which have dependencies across long spans of sentences. Devlin et al. (2019) mention in their work : *"It is critical to use a document-level corpus rather than a shuffled sentence-level corpus (...) in order to extract long contiguous sequences."* However, many of our pretraining corpora made from downstream datasets have shorter and relatively superficial dependencies between sentences. For example, in the MNLI or QQP corpora, the only dependencies are that neighboring sentences may be premise-hypothesis pairs or potential paraphrase candidates, respectively.

Despite the lack of long-term narrative dependencies, many pretraining corpora achieve high performance boosts on the GLUE language understanding benchmark (Table 5). For example, MNLI achieves around 96% of the performance boost of the off-the-shelf model (Table 5). Interestingly, even though the cross-sentence dependencies in MNLI and QQP corpora are superficial, removing them by shuffling the sentences leads to a large drop in performance (Table 5). This suggests that

Pretraining Dataset	Random	Standard	TF-IDF(Ours)
None (RandomInit)	-	53.20	-
Sentiment140	-	72.67	75.29
DBpedia14	72.82	70.38	75.44
Discovery	71.79	77.26	78.94
MNLI	62.80	78.28	76.33
QQP	71.09	75.43	69.57
BookWiki (Off-the-shelf)	-	79.43	-

Table 5: GLUE scores achieved by different strategies for ordering sentences from the downstream dataset used for pretraining. Random: randomly ordered sentences; Standard: sentences within a datapoint occur contiguously in original order; TF-IDF: sentences reordered using content similarity.

while having long-term dependencies in the pretraining corpus is important, some benefit can even be gained by less complex dependencies than what's seen in typical documents.

Inspired by the results, we test if GLUE performance can be improved by artificially introducing superficial dependencies in a corpus of unordered sentences. We rearrange the sentences in the sentence-shuffled versions of pretraining corpora to encourage content overlap among neighboring sentences, and see if this can recover some of the drop in performance that occurred due to shuffling. Our algorithm creates the corpus by iteratively appending sentences to it, such that at each step the new sentence is the one with maximum TF-IDF similarity with the previous sentence. We also make sure that the algorithm does not accidentally recover the original order of sentences (e.g. by matching the premise-hypothesis pairs originally in the MNLI dataset).

We experiment with 5 different datasets and find that the sentence-reordering scheme improves performance compared to random sentence order for all of them except QQP. For Discovery and DBPedia14 datasets, it scores even higher than our *standard* sentence ordering scheme which preserves the adjacency and order of sentences within each datapoint. This shows that introducing long-term dependencies among neighboring sentences of the form of some content overlap, can potentially improve GLUE score, without introducing any new information or narrative structure.

8 Ablations with other pretraining architectures

We conducted our experiments so far with ELECTRA-small architecture because it is faster to pretrain than other popular models, yet delivers good downstream performance (Clark et al., 2019) (e.g. comparable to BERT-base on GLUE benchmark). But we also conduct experiments with more architectures and model sizes to test the efficacy of self-pretraining more broadly.

We experiment with the RoBERTa model which uses only the masked language modeling objective, rather than ELECTRA's complex objective. We use the RoBERTa-base architecture, which has a much larger parameter count of 110 million, compared to ELECTRA-small's 14 million. Due to resource constraints, we pretrained the RoBERTa models for fewer iterations as outlined in Warstadt et al. (2020). For reference, we pretrain a RoBERTa-base model on the BookWiki corpus for the same number of iterations. Our results show again that self-pretraining performs comparably to pretraining on BookWiki corpus, delivering over 85% of pretraining benefit on 9 out of 10 datasets, and outperforming the model pretrained on BookWiki corpus (Table 6) on 5 datasets.

In addition to experimenting with a *base*-sized architecture (110M parameters), we also experiment with architectures which are even smaller than ELECTRA-small. We train ELECTRA models of smaller size by either reducing the number of layers in the generator and discriminator, or reducing the hidden dimension of the discriminator[3]. As the models get smaller, self-pretraining continues to significantly outperform random initialization and often outperforms pretraining on BookWiki corpus (Figure 3). Interestingly, the relative performance of self-pretrained and BookWiki-pretrained models tends to stay the same across model size. For example, for QQP self-pretraining is always best and for MNLI BookWiki-pretraining is always best irrespective of number of layers or hidden size.

[3]In ELECTRA, the generator's hidden size is already much smaller than that of the discriminator by design. So we do not reduce it further, in order to have a reasonably well-performing generator.

Dataset	RandInit	SelfPretrain	BookWiki	Benefit %	TAPT
AGNews	91.91	94.28	94.22	102.27	94.07
QQP	76.50	88.68	90.18	89.05	90.64
Jigsaw Toxicity	97.32	97.72	98.03	56.02	98.48
MNLI	31.82	75.12	80.90	88.23	79.26
Sentiment140	56.68	68.55	60.19	338.26	65.65
PAWS	50.00	97.34	97.08	100.55	97.85
DBPedia14	98.57	99.21	99.24	95.98	99.23
Discovery	17.36	25.85	26.30	94.91	23.58
Yahoo Answertopics	61.11	65.96	64.58	139.80	65.05
Amazon Polarity	89.02	96.68	96.11	108.13	96.16

Table 6: Performance of RoBERTa-base models pretrained with different techniques on downstream datasets.

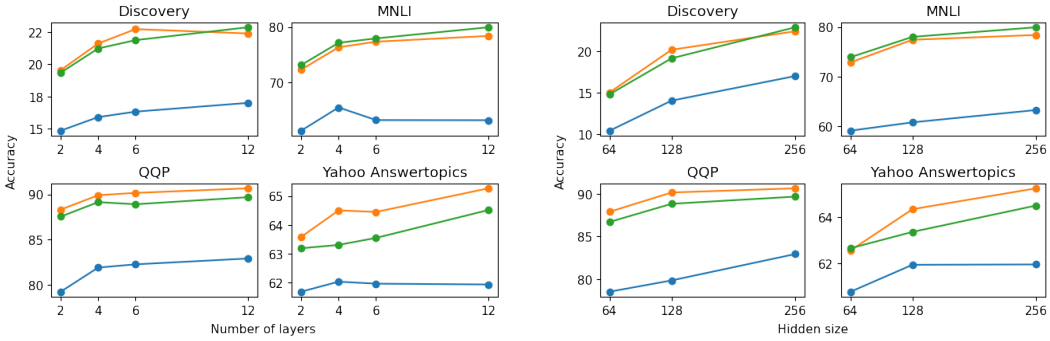

Figure 3: Variation in performance of ELECTRA models with change in number of layers and hidden size (— randomly initialized, — self-pretrained, — BookWiki-pretrained)

9 Conclusion and Future Work

In this work, we showed that pretraining models only on text from the downstream dataset performs comparably to pretraining on a huge upstream corpus for a wide variety of datasets. The errors made by such *self-pretrained* models on the downstream tasks are significantly different from the ones made by the *off-the-shelf* models pretrained on upstream corpora. Our results suggest that the importance of learning from surplus upstream data for improving downstream task performance may have been overestimated.

We found that the quantity and quality of data required for pretraining to provide significant benefit (over a randomly initialized model trained only with a supervised loss) is quite low. Downstream datasets which are tiny in comparison to typical upstream corpora, still function as useful pretraining corpora for getting performance gains across a wide range of datasets. Additionally, many of these datasets (e.g. MNLI, QQP) do not contain high quality long-term dependencies and yet manage to perform well even on the GLUE benchmark for language understanding. A possible future work could be to characterize the nature of long-term dependencies in pretraining corpora and quantify their impact on downstream performance boosts.

Since self-pretraining does not involve any upstream corpus, it prevents exposure of the model to potentially undesirable contents in the large upstream corpus, while still delivering large performance benefits. Research has demonstrated the negative influence of web-sourced pretraining corpora on models, such as generating toxic language (Gehman et al., 2020) or reflecting racial biases in predictions (Ahn & Oh, 2021). In future work, we hope to compare how self-pretrained models and off-the-shelf models perform on these negative measures such as toxicity and social biases.

REFERENCES

Jaimeen Ahn and Alice Oh. Mitigating language-dependent ethnic bias in bert. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 533–549, 2021.

Tuhin Chakrabarty, Christopher Hidey, and Kathleen McKeown. Imho fine-tuning improves claim detection. In *Proceedings of NAACL-HLT*, pp. 558–563, 2019.

Cheng-Han Chiang and Hung-yi Lee. Pre-training a language model without human language. *arXiv preprint arXiv:2012.11995*, 2020.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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)*, pp. 4171–4186, 2019.

Alaaeldin El-Nouby, Gautier Izacard, Hugo Touvron, Ivan Laptev, Hervé Jegou, and Edouard Grave. Are large-scale datasets necessary for self-supervised pre-training? *arXiv preprint arXiv:2112.10740*, 2021.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A Smith. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 3356–3369, 2020.

Robert Geirhos, Kristof Meding, and Felix A Wichmann. Beyond accuracy: quantifying trial-by-trial behaviour of cnns and humans by measuring error consistency. *Advances in Neural Information Processing Systems*, 33:13890–13902, 2020.

Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford*, 1(12):2009, 2009.

Raphael Gontijo-Lopes, Yann Dauphin, and Ekin D Cubuk. No one representation to rule them all: Overlapping features of training methods. In *International Conference on Learning Representations (ICLR)*, 2022.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8342–8360, 2020.

Xiaochuang Han and Jacob Eisenstein. Unsupervised domain adaptation of contextualized embeddings for sequence labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4238–4248, 2019.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16000–16009, 2022.

Kaggle.com. Toxic comment classification challenge: Identify and classify toxic online comments. `https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge`, 2018.

Kundan Krishna, Jeffrey P Bigham, and Zachary C Lipton. Does pretraining for summarization require knowledge transfer? In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 3178–3189, 2021.

Yongjie Lina, Yi Chern Tana, and Robert Frankb. Open sesame: Getting inside bert's linguistic knowledge. *ACL 2019*, pp. 241, 2019.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. 2019.

Lajanugen Logeswaran, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Jacob Devlin, and Honglak Lee. Zero-shot entity linking by reading entity descriptions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3449–3460, 2019.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Pretrained transformers as universal computation engines. *arXiv preprint arXiv:2103.05247*, 2021.

Christopher D Manning, Kevin Clark, John Hewitt, Urvashi Khandelwal, and Omer Levy. Emergent linguistic structure in artificial neural networks trained by self-supervision. *Proceedings of the National Academy of Sciences*, 117(48):30046–30054, 2020.

Isabel Papadimitriou and Dan Jurafsky. Learning music helps you read: Using transfer to study linguistic structure in language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6829–6839, 2020.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 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)*, pp. 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL `https://aclanthology.org/N18-1202`.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2463–2473, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

Machel Reid, Yutaro Yamada, and Shixiang Shane Gu. Can wikipedia help offline reinforcement learning? *arXiv preprint arXiv:2201.12122*, 2022.

Damien Sileo, Tim Van De Cruys, Camille Pradel, and Philippe Muller. Mining discourse markers for unsupervised sentence representation learning. 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)*, pp. 3477–3486, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/N19-1351`.

Koustuv Sinha, Robin Jia, Dieuwke Hupkes, Joelle Pineau, Adina Williams, and Douwe Kiela. Masked language modeling and the distributional hypothesis: Order word matters pre-training for little. *arXiv preprint arXiv:2104.06644*, 2021.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4593–4601, 2019.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, 2018.

Alex Warstadt, Yian Zhang, Xiaocheng Li, Haokun Liu, and Samuel Bowman. Learning which features matter: Roberta acquires a preference for linguistic generalizations (eventually). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 217–235, 2020.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1112–1122, 2018.

Yuhuai Wu, Felix Li, and Percy Liang. Insights into pre-training via simpler synthetic tasks. *arXiv preprint arXiv:2206.10139*, 2022.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *NIPS*, 2015.

Yuan Zhang, Jason Baldridge, and Luheng He. Paws: Paraphrase adversaries from word scrambling. 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)*, pp. 1298–1308, 2019.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pp. 19–27, 2015.

A Appendix

A.1 Implementation details for pretraining and finetuning

Hyperparameters for pretraining For pretraining ELECTRA-small models, we use the standard hyperparameters (Table 7) as described in Clark et al. (2019). For the Roberta-base models, training with the standard hyperparameters with our computing resources would be prohibitively slow, and so we used hyperparameters from Warstadt et al. (2020) which require lesser time to train (Table 7). For task-adaptive pretraining(TAPT), we follow Gururangan et al. (2020) and further pretrain off-the-shelf models for 100 epochs on the downstream task's training set, with the first 6% of the resulting total updates used for learning rate warmup.

Hyperparameters for finetuning For finetuning the models on the 10 downstream datasets, we use hyperparameters as shown in Table 8. We use the AdamW optimizer (Loshchilov & Hutter, 2018) for finetuning. We use early stopping based on validation set performance. The validation metric used is mean squared error for the sentiment140 dataset (regression), average binary crossentropy for the jigsaw dataset (multi-label classification), and accuracy for all other datasets (multi-class classification). The patience parameter for early stopping is set to 3 epochs. For finetuning ELECTRA-small models on the GLUE datasets, we use the standard learning rate of 1e-4 following Clark et al. (2019).

Details about use of downstream datasets All downstream datasets used in this paper were sourced from the Huggingface library[4]. For the Yahoo Answertopics dataset, we use only the text from the answer (not the question) as input to the models (both for pretraining and finetuning). For the PAWS dataset, we use the version called "Unlabeled $\text{PAWS}_{\text{wiki}}$" in Zhang et al. (2019), which is actually *not* unlabeled but has silver labels. We preferred that version over others because of its larger size. For datasets which had a train and test split but no validation split (e.g. Yahoo Answertopics), we extracted 5000 random datapoints from the the train split to make the validation split. If a dataset had a train and validation split but no test split (e.g. Unlabeled $\text{PAWS}_{\text{wiki}}$), we designated the validation

[4] https://huggingface.co/docs/datasets/index

split to be the test split, and created a new validation set by extracting 5000 random datapoints from the train set.

Hyperparameter	**ELECTRA**	**Roberta**
Size (Parameter count)	Small (14M)	Base (110M)
Training steps	1M	100K
Warmup steps	10K	6K
Batch size	128	512
Peak learning rate	5e-4	5e-4
Sequence length	128	512

Table 7: Hyperparameters used for pretraining models

Hyperparameter	**ELECTRA**	**Roberta**
Training epochs	20	20
Batch size	32	32
Learning rate	{1e-4,1e-5}	2e-5
Max sequence length	512	512

Table 8: Hyperparameters used for finetuning models on 10 downstream tasks

A.2 Software packages and hardware used

For pretraining ELECTRA models, we used Nvidia's implementation of the ELECTRA codebase [5], run using Nvidia's Tensorflow cotainer image 21.07 [6]. For pretraining Roberta models, we used the official implementation in the Fairseq library[7]. For finetuning experiments, we used the AllenNLP library for training and evaluation routines, coupled with the Huggingface library for the model architectures.

We used a collection of Nvidia V100 (32GB) and A6000(48GB) GPUs for our experiments. Pretraining an ELECTRA-small model takes around 1.5 days on 2 GPUs while pretraining a Roberta-base model takes around 1.5 days on 4 GPUs.

[5] https://github.com/NVIDIA/DeepLearningExamples/tree/master/TensorFlow2/LanguageModeling/ELECTRA

[6] https://docs.nvidia.com/deeplearning/frameworks/tensorflow-release-notes/rel_21-07.html

[7] https://github.com/facebookresearch/fairseq