Robust QA with Task-Adaptive Pretraining

Stanford CS224N Default Project

Jeong-O Jeong
Department of Computer Science
Stanford University
djeongo@stanford.edu

Abstract

It is often hard to find a lot of labeled data to train a QA (question answering) model. One possible approach to overcome this challenge is to use TAPT (task-adaptive pretraining) in which the model is pretrained further using the unlabeled data from the task itself [1]. We implement the TAPT technique to make a QA model perform robustly on a task with low-resource training data by first pertaining on the larger unlabeled data set. We then fine tune the model with a smaller labeled dataset. The results are mixed. Although a preliminary model that is pretrained on just the out-of-domain train *oodomain_train* data performed better than the baseline, additional pretraining using more unlabeled out-of-domain data performed worse than expected.

1 Key Information to include

- Mentor: N/A
- External Collaborators (if you have any): N/A
- Sharing project: N/A

2 Introduction

Performing well on low-resource task is difficult, because large language models such as BERT require a lot of data to train. However, it is expensive to obtain labeled data, and they are often unavailable. One possible way to address this problem is to further pre-train the model with additional data in order to make the language model perform better on the downstream tasks [2]. However, performing additional pre-training on large corpus of data can be expensive as well. The TAPT technique shows that if the pre-training data is chosen to be specific to the task at hand, it gives significant improvement to the task performance even if the amount of data is small. We implement the TAPT technique with various types of pretraining data to evaluate its effectiveness in improving performance on low-resource out-of-domain QA task.

3 Related Work

The main TAPT paper [1] considers two techniques, DAPT (domain-adaptive pretraining) and TAPT (task-adaptive pretraining), for further pretraining a language model. DAPT consists of further pretraining the language model on the data in the same domain as that of the task, while TAPT consists of directly training on the task data itself, but using it as *unlabeled* data. They found that DAPT improves on the off-the-shelf RoBERTa baseline model across all domains, and that TAPT can be competitive with DAPT even with a very small amount of data. The paper shows that combining both DAPT and TAPT results in the best performance.

They also explore data selection method for optimal transfer learning using the VAMPIRE embedding [3]. Similar work [4] uses cosine similarity of sentences to cluster corpus domains, which can help with data selection for task-specific pretraining.

The TAPT technique fits into the wider context of task-adaptive pretraining of an existing language model to improve performance on a specific NLP task. Task-adaptive pretraining has been studied in [5] where they perform general LM pretraining followed by task-specific LM fine-tuning to improve performance.

4 Approach

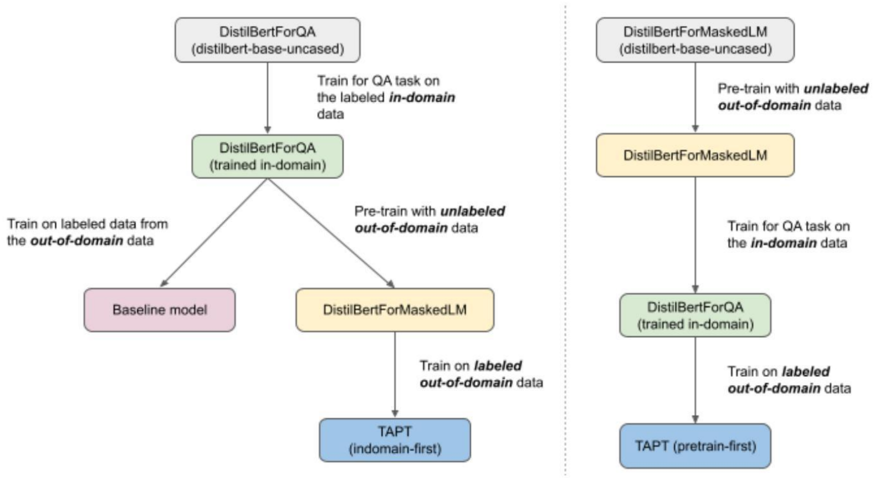
DistilBertForQA (distilbert-base-uncased)
Train for QA task on the labeled ***in-domain*** data
DistilBertForQA (trained in-domain)
Train on labeled data from the ***out-of-domain*** data
Pre-train with ***unlabeled out-of-domain*** data
Baseline model
DistilBertForMaskedLM
Train on ***labeled out-of-domain*** data
TAPT (indomain-first)
DistilBertForMaskedLM (distilbert-base-uncased)
Pre-train with ***unlabeled out-of-domain*** data
DistilBertForMaskedLM
Train for QA task on the ***in-domain*** data
DistilBertForQA (trained in-domain)
Train on ***labeled out-of-domain*** data
TAPT (pretrain-first)

Figure 1: Model training approaches. Left: TAPT (indomain-first), Right: TAPT (pretrain-first)

4.1 Baseline

The baseline model is a *DistilBertForQuestionAnswering* model that is trained on the high-resource dataset and then further trained using the low-resource out-of-domain dataset. The implementation for the baseline model is obtained from [6]. Figure 1 summarizes the approaches in generating the baseline and the TAPT models.

4.2 TAPT approach

The approach we take is to perform TAPT using the unlabeled data from the low-resource out-of-domain training dataset. Specfically, we further pretrain a *DistilBertForQuestionAnswering* model [7] that is already trained on a question answering task with the in-domain datasets that are high-resource, as shown in the left half of Figure 1. We also try pretraining the base *DistilBertForMaskedLM* model with unlabeled out-of-domain data first, and then train with the in-domain high-resource train dataset, as shown in the right half of Figure 1. Finally, we train the pre-trained models with the low-resource out-of-domain labeled training data.

4.3 Pretraining

We implement the masked LM pretraining method. We use the standard masked LM cross-entropy loss described in the original BERT paper [8] with 15% mask probability. The words are randomly masked with 15% probability and the model is trained to guess the correct word for the masked tokens. Following the original masked LM training procedure, for 80% of the time, we replace it with the mask token, for 10% of the time, we replace with a random token, and 10% of the time we leave it unchanged.

4.3.1 Pretraining data

We try different pretraining data in the following list to see the effect of data used for TAPT.

- *oodomain_train* as unlabeled data
- *oodomain_val* as unlabeled data
- *oodomain_test* as unlabeled data
- Combination of *oodomain_train*, *oodomain_val*, *oodomain_test* as unlabeled data
- Curated unlabeled data from the original *oodomain* corpora

To use *oodomain_train* for pretraining, we chunk the 248 passages available in the dataset into 512-length tokens with stride of 128. This results in 273 sentences of length 512 tokens. We perform the same process for *oodomain_val* and *oodomain_test*. We also try combining all of *oodomain* available to generate 9,535 sentences of length 512 tokens. Finally, we try using data from the original corpora of *duorc*, *race*, and *relation_extraction* data sets to generate even more unlabeled pretraining training data. However, instead of using the entire corpora, we curate the most similar sentences using cosine-similarity of sentence embeddings as described in the next section.

4.3.2 Curating pre-training data using cosine-similarity of sentence embeddings

Since the provided *oodomain_train* data is only a subset of the corpora from which they are derived, it might be helpful to incorporate more data from the original corpora. We use the unlabeled data as descried in Table 1 from the original corpora to augment the pretraining dataset.

Corpus	Data
DuoRC [9]	5,133 paraphrase passages (train)
RACE [10]	18,728 passages from middle and high (train)
Relation extraction [11]	107,765 unique sentences from "positive_examples"

Table 1: Pretraining data

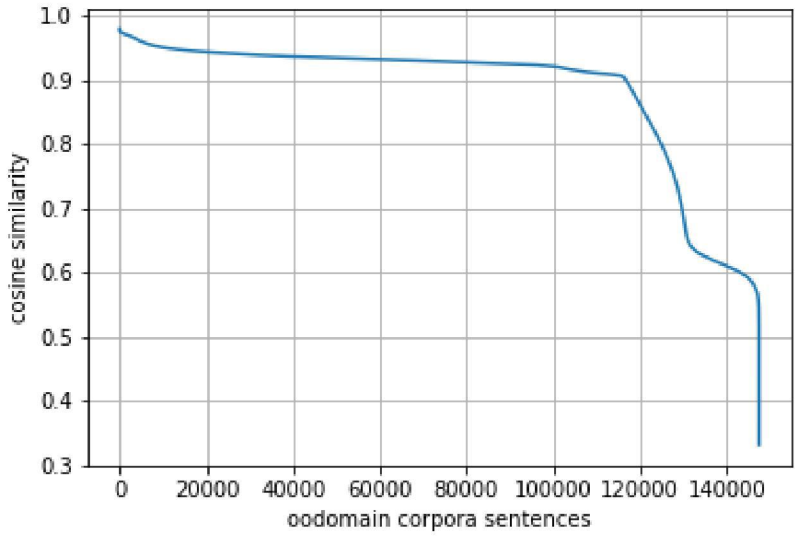

Figure 2: Cosine similarity between oodomain_train and oodomain corpora

As using the entire corpora could negatively affect the performance if a large portion of the data is unrelated to the task, we use cosine similarity of sentence embeddings to curate only the most similar sentences from the corpora [4]. Specifically, we obtain a representative sentence embedding of the *oodomain* data and then compare against all sentence embeddings in the original corpora. In order to obtain the representative sentence embedding of *oodomain* data, we average the last hidden layer outputs of *DistilBertModel* of all 512-length sentences in the *oodomain_train* data to obtain a vector of size $\mathcal{R}^{1\times768}$.

The original *duorc*, *race*, and *relation_extraction* corpora shown in Table 1 are chunked to 512-length sentences resulting in a total of 147,419 sentences. For each sentence, we use *DistilBertModel* to generate the sentence embedding. For each sentence embedding, we compute the cosine similarity against the representative embedding of the *oodomain* data to quantify how similar each sentence is to the *oodomain* training data. We then pick the 1,000, 10,000, and 100,000 most similar sentences for pretraining.

Figure 2 shows the cosine similarities of the sentences in the original corpora to the *oodomain* data, sorted by cosine similarity. It shows that the top 100,000 most simliar sentences have a high similarity value of at least 0.9, but it sharply drops off after around 100,000 most similar sentences. We exclude the sentences with low similarity from pretraining since they could negatively affect the downstream task performance.

5 Experiments

5.1 Data

The datasets used are shown in Table 2. For training a QA model, we use the *indomain_train* data which consists of 5,000 SQuAD, 5,000 NewsQA, and 5,000 Natural Questions question and answer pairs. For fine-tuning a QA model, we use the *oodomain_train* data which consists of 381 question and answer pairs from DuoRC, RACE, and RelationExtraction. For validation while fine-tuning the QA model, we use the *oodomain_val* which consists of 382 question and answer pairs from DuoRC, RACE, and RelationExtraction datasets.

Data	Description	Purpose
indomain_train	15,000 QA pairs from SQuAD, NewsQA and Natural Questions	training a QA model
oodomain_train	381 QA pairs from DuoRC, RACE, and RelationExtraction	fine-tuning the QA model
oodomain_val	382 QA pairs from DuoRC, RACE, and RelationExtraction	evaluating the QA model
duorc	5,133 unlabeled paraphrase passages	pre-training
race	18,728 passages from middle and high	pre-training
relation_extraction	107,765 unique sentences from "positive_examples"	pre-training

Table 2: Data sets and their purposes

5.2 Evaluation method

We use the F1 and EM (Exact Match) scores for evaluation. The F1 score is a harmonic mean of precision and recall. It is a "soft" metric to measure how much overlap exists between the tokens in the predicted answer and the tokens in the expected answer. The EM score is a "hard" metric that indicates whether the predicted answer exactly matches the expected answer. For each question and answer pair, the maximum F1 and EM scores are 1.0 each. The F1 and EM scores shown in the results section in this report are aggregate values across the entire validation or test set which are scaled to a maximum value of 100.

5.3 Experimental details

5.3.1 Baseline

The baseline model is trained for 20 epochs on the out-of-domain dataset with a fixed learning rate of $3 \cdot 10^{-5}$. The baseline achieves F1 score of 49.697 and EM score of 34.555 on the validation leaderboard.

5.3.2 TAPT model

For the "TAPT (indomain-first)" model, we start off with the model that is trained on the *in_domain* QA task. We then further pre-train the model using the unlabeled data from the out-of-domain data sets. The pretraining is performed for 10 epochs using the masked LM loss. Finally, we train the model using the labeled out-of-domain data. The training is performed for 20 epochs with a fixed learning rate of $3 \cdot 10^{-5}$. We evaluate the TAPT model against the out-of-domain validation dataset. We explored different learning rates of $3 \cdot 10^{-6}$, $1 \cdot 10^{-5}$, $2 \cdot 10^{-5}$ as well, but it did not impact the results noticeably.

5.4 Results

The results show that the TAPT model can improve over the baseline model, but not necessarily for all cases. The results are summarized in Table. 3

Model	Pretraining Data	F1 (val)	EM (val)	F1 (test)	EM (test)
Baseline	N/A	49.697	34.555		
TAPT (in-domain first)	oodomain_train	51.182	37.958	**57.700**	40.092
TAPT (in-domain first)	oodomain_val	50.806	36.387	57.124	39.495
TAPT (in-domain first)	oodomain_test	50.16	35.34	57.5	**40.367**
TAPT (in-domain first)	oodomain_train, val, test	47.97	34.03		
TAPT (in-domain first)	1,000 most similar	48.82	35.6		
TAPT (in-domain first)	10,000 most similar	47.61	33.77		
TAPT (in-domain first)	100,000 most similar	40.42	27.23		
TAPT (pretrain first)	10,000 most similar	48.42	33.25		
TAPT (pretrain first)	100,000 most similar	47.97	34.03		

Table 3: Experiment results

The results show that TAPT model with just using the *oodomain_train* data can improve on the baseline model, but it also performed poorly in other cases. This is unexpected because we expect that the more relevant pretraining data we use, the better the model should perform, but the results show that using 100,000 most similar sentences performs worse than using 1,000 most similar sentences. This might be because for the "TAPT (in-domain first)" approach, the weights of the Transfomer layers start to become less compatible with the weights of the QA head layers, since the QA head layers cannot be updated during pretraining while the Transformer layers are changing. The best performing model based on the F1 metric achieves 57.7 and 40.092 for F1 and EM scores, respectively.

5.4.1 Pretrain duration

We look at the impact of number of epochs in pretraining on the downstream task. Unlike training for a downstream task, there is no validation set to use during pretraining. Since it is unclear how to determine how many epochs to pretrain for without the validation set, we evaluate the downstream QA task performance using models pretrained for different number of epochs.

Figure 3 shows how the QA model performs when it is fine tuned on models that got pretrained for different number of epochs. The results indicate that pretraining for multiple epochs actually leads to

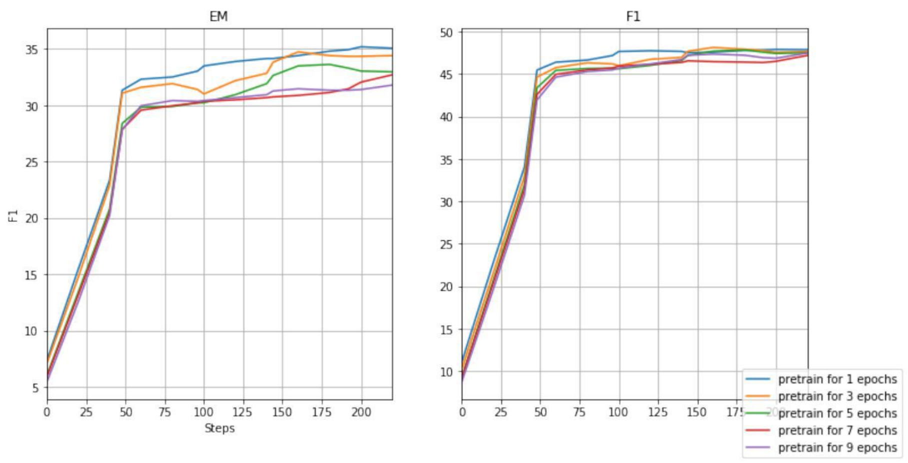

Figure 3: Effect of pretraining epochs

degradation in performance for the downstream task, when using "TAPT (in-domain first)" method. This is likely due to the fact that the Tranformer layers start to drift from the Question and Answer head layer that was trained using *indomain* data. This problem does not appear to happen in the "TAPT (pretrain first)" approach.

6 Analysis

The following question and answer pairs show some of the examples where the baseline model's F1 score was 0, while the TAPT model's score was 1.0. It is interesting that the baseline model provided wrong answers for the passages with highly technical terms, while the TAPT model provided correct answers. This improvement over baseline on the specific domain about chromosome might be attributed to the fact that the TAPT model has been further pretrained on the relevant data, while the baseline model has not been.

1.
   - **F1** - Baseline: 0.0, TAPT: 1.0
   - **Context** - In humans, the gene RUNX1 is 260 kilobases (kb) in length, and is located on chromosome 21 (21q22.12).
   - **Question** - What is the name of RUNX1's chromosome?
   - **Answer(Baseline)** - 21q22.12
   - **Answer (TAPT)** - chromosome 21
   - **Answer (Correct)** - chromosome 21
2.
   - **F1** - Baseline: 0.0, TAPT: 1.0
   - **Context** - By genomic sequence analysis, the FOXP3 gene maps to the p arm of the X chromosome (specifically, Xp11.23).
   - **Question** - Which chromosone can you find FOXP3?
   - **Answer(Baseline)** - Xp11.23
   - **Answer (TAPT)** - X chromosome
   - **Answer (Correct)** - X chromosome
3.
   - **F1** - Baseline: 0.0, TAPT: 1.0
   - **Context** - In follicular lymphoma, a chromosomal translocation commonly occurs between the fourteenth and the eighteenth chromosomes – t(14;18) – which places the Bcl-2 gene from chromosome 18 next to the immunoglobulin heavy chain locus on chromosome 14.
   - **Question** - Which chromosone can you find Bcl-2?
   - **Answer(Baseline)** - immunoglobulin
   - **Answer (TAPT)** - chromosome 18
   - **Answer (Correct)** - chromosome 18

7 Conclusion

We implemented the TAPT technique with different pretraining data to improve performance on a low-resource QA task. In addition to using the provided dataset, we tried augmenting the dataset by curating additional data from the original corpora using cosine-similarity metrics.

Results show that although TAPT showed improved performance when using only *oodomain_train*, *oodomain_val*, or *oodomain_test* as unlabeled data set for pretraining, it showed worse performance when using more data for pretraining. This was unexpected because intuitively, the more relevant data is used for pretraining, the better model should have performed. Specifically, pretraining the model with the combination of *oodomain_train*, *oodomain_val*, and *oodomain_test* led to worse performance than using each one separately. However, this might be because the approach "TAPT (in-domain first)" was potentially flawed. The right approach to explore more extensive might have been the "TAPT (pretrain first)" approach. However, it was too expensive to perform the in-domain training multiple times after each different pre-training data set, so that approach could not be pursued more thoroughly.

For future work, we could try additional techniques for augmenting the training dataset such as back translation and word substitution with synonyms. We could also explore the use of DAPT (domain-adaptive pretraining) to curate more pretraining data and use "TAPT (pretrain first)" approach instead of "TAPT (indomain-first)" approach. Another possibility is to use span masked LM technique [12] for pretraining instead of the plain masked LM technique.

References

[1] 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*, pages 8342–8360, 2020.

[2] 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.

[3] Suchin Gururangan, Tam Dang, Dallas Card, and Noah A. Smith. Variational pretraining for semi-supervised text classification. In *Proceedings of ACL*, 2019.

[4] Roee Aharoni and Yoav Goldberg. Unsupervised domain clusters in pretrained language models. *arXiv preprint arXiv:2004.02105*, 2020.

[5] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

[6] Murty Shikhar. Starter code for robustqa track. `https://github.com/MurtyShikhar/robustqa`.

[7] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[9] Amrita Saha, Rahul Aralikatte, Mitesh M. Khapra, and Karthik Sankaranarayanan. DuoRC: Towards Complex Language Understanding with Paraphrased Reading Comprehension. In *Meeting of the Association for Computational Linguistics (ACL)*, 2018.

[10] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*, 2017.

[11] Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. *arXiv preprint arXiv:1706.04115*, 2017.

[12] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77, 2020.