Fast Vocabulary Transfer for Language Model Compression

Abstract

Real-world business applications require a trade-off between language model performance and size. We propose a new method for model compression that relies on vocabulary transfer. We evaluate the method on various vertical domains and downstream tasks. Our results indicate that vocabulary transfer can be effectively used in combination with other compression techniques, yielding a significant reduction in model size and inference time while marginally compromising on performance.

1 Introduction

In the last few years, many NLP applications have been relying more and more on large pre-trained Language Models (LM). Because larger LMs, on average, exhibit higher accuracy, a common trend has been to increase the model's size. Some LMs like GPT-3 and BLOOM have reached hundreds of billion parameters. However, these models' superior performance comes at the cost of a steep increase in computational footprint, both for development and for inference, ultimately hampering their adoption in real-world business use-cases. Besides models that only a few hi-tech giants can afford, like GPT-3, even smaller LMs with hundreds of million parameters could be too expensive or infeasible for certain products. For one thing, despite being tremendously cheaper than their bigger cousins, fine-tuning, deploying and maintaining large numbers of such models (one for each downstream task) soon becomes too expensive. Furthermore, latency and/or hardware requirements may limit their applicability to specific use-cases. For all these reasons, significant efforts - in both academic and industry-driven research - are oriented towards the designing of solutions to drastically reduce the costs of LMs.

Recently, several attempts have been made to make these models smaller, faster and cheaper, while retaining most of their original performance. Knowledge Distillation (KD) is a teacher-student framework, whereby the teacher consists of a pre-trained large model and the student of a smaller one. The teacher-student framework requires that both the teacher and the student estimate the same probability distribution. While the outcome is a smaller model, yet, this procedure constrains the student to operate with the same vocabulary as the teacher in the context of Language Modeling.

In this work, we explore a method for further reducing an LM's size by compressing its vocabulary through the training of a tokenizer in the downstream task domain. The tokenizer is a crucial part of modern LMs. In particular, moving from word to subword- level, the tokenization solves two problems: vocabulary explosion and unknown words. Moreover, the capability to tokenize text effectively in any domain is key for the massive adoption of pre-trained general-purpose LMs fine-tuned on downstream tasks. Indeed, tokenizers are still able to process out-of-distribution texts at the cost of producing frequent word splits into multiple tokens.

However, the language varies significantly in vertical domains or, more generally, in different topics. Hence, ad-hoc tokenizers, trained on the domain statistics, may perform a more efficient tokenization, reducing on average the length of the tokenized sequences. This is important since compact and meaningful inputs could reduce computational costs, while improving performance. Indeed, memory and time complexity of attention layers grows quadratically with respect to the sequence length.

Furthermore, a vertical tokenizer may require a smaller vocabulary, which also affects the size of the embedding matrix, hence further reducing the model's size.

Following this intuition, we propose a Vocabulary Transfer (VT) technique to adapt LMs to in-domain, smaller tokenizers, in order to further compress and accelerate them. This technique is complementary to the aforementioned model compression methods and independent of the type of tokenizer. As a matter of fact, we apply it in combination with KD.

Our experiments show that VT achieves an inference speed-up between x1.07 and x1.40, depending on the downstream task, with a limited performance drop, and that a combination of VT with KD yields an overall reduction up to x2.76.

The paper is organized as follows. After reviewing related works in Section 2, we present the methodology in Section 3, we then outline the experiments in Section 4 and draw our conclusions in Section 5.

2 Related Work

The goal of Model Compression is to shrink and optimize neural architectures, while retaining most of their initial performance. Research on LM compression has been carried out following a variety of approaches like quantization, pruning knowledge distillation, and combinations thereof.

A most popular distillation approach in NLP was proposed by Sanh et al. (2019). The obtained model, called DistilBERT, is a smaller version of BERT, with the same architecture but half the layers, trained to imitate the full output distribution of the teacher (a pre-trained BERT model). DistilBERT has a 40

Little focus has been devoted thus far to the role of tokenization in the context of model compression. Even in domain adaptation, the vocabulary was kept the same. Both the versatility of the subword-level tokenization, and the constraints imposed by the teacher- student framework (same output distribution), discouraged such investigations. Recently, Samenko et al. (2021) presented an approach for transferring the vocabulary of an LM into a new vocabulary learned from new domain, with the purpose of boosting the performance of the fine-tuned model. To the best of our knowledge, we are the first to study VT in the scope of model compression.

3 Vocabulary Transfer

Let us consider a LM, trained on a general-purpose domain D_{gen} and associated with a vocabulary V_{gen}. Such a vocabulary is used by the LM's tokenizer in order to produce an encoding of the input string via an embedding matrix E_{gen} defined on V_{gen}. More specifically, a tokenizer is a function that maps a textual string into a sequence of symbols of a given vocabulary V. Let T be a tokenizer associated with a vocabulary V and a string s, we have $T : s \rightarrow (t_1, \ldots, t_n), t_i \in V, \forall i = 1, \ldots, n$. Hence, the vocabulary of the tokenizer determines how words in a text are split, whether as words, sub-words, or even characters. These symbols, which define the LM's vocabulary, are statistically determined by training the tokenizer to learn the distribution of a dataset.

Now, let us consider a vertical domain Din, also referred as in-domain. For the reasons discussed earlier, a vocabulary Vin specialized on Din itself better fits the language distribution than Vgen. Unfortunately, with a new vocabulary, embedding representations associated with the tokens of Vgen would be lost. Thus, VT aims to initialize Vin by re-using most of the information learned from the LM pre-trained on Dgen. Once the new tokenizer Tin has been trained on the in-domain dataset Din using a given vocabulary size, Tin will be different from the LM's tokenizer Tgen. However, the two tokenizers' vocabularies Vgen and Vin may still have a large portion of their symbols in common. Our objective is to transfer most of the information from Vgen into Vin. To this end, we first define a mapping between each symbol in Vin and a set of symbols in Vgen. Then, we define an assignment criterion, based on the mapping, to obtain the embeddings for the tokens of Tin.

One such criterion, called Vocabulary Initialization with Partial Inheritance (VIPI), was defined by Samenko et al. (2021). Whenever a token is in V_{in} but not in V_{gen}, VIPI calculates all the partitions of the new token with tokens from V_{gen}, then takes the minimal partitions and finally averages them to obtain an embedding for the new token. Differently, we define a simplified implementation of

VIPI called FVT for Fast Vocabulary Transfer. Instead of calculating all tokenizations, FVT uses a straightforward assignment mechanism, whereby each token $t_i \in V_{\text{in}}$ is partitioned using T_{gen}. If t_i belongs to both vocabularies, $t_i \in V_{\text{in}} \cap V_{\text{gen}}$, then $T_{\text{gen}}(t_i) = t_i$ and the in-domain LM embedding.

$$Ein(ti) = Egen(ti). \tag{1}$$

If instead $t_i \in V_{\text{in}} \setminus V_{\text{gen}}$, then the in-domain embedding is the average of the embeddings associated with the tokens produced by T_{gen}:

$$Ein(t:) = \frac{1}{|T_{gen}(t_i)|} \sum_{t_j \in T_{gen}(t_i)} E_{gen}(t_j) \tag{2}$$

Please notice that Equation (2) is a generalization of Equation (1). Indeed, in case $t_i \in V_{\text{in}} \cap V_{\text{gen}}$, Equation (2) falls back to Equation (1).

Once embeddings are initialized with FVT, we adjust the model's weights by training it with MLM on the in-domain data before fine-tuning it on the downstream task. MLM eases adaptation and has already been found to be beneficial in (Samenko et al., 2021). We observed this trend as well during preliminary experiments, therefore we kept such a tuning stage in all our experiments.

As a baseline model, we also implement a method called Partial Vocabulary Transfer (PVT), whereby only the tokens belonging to both vocabularies $t_i \in V_{\text{in}} \cap V_{\text{gen}}$ are initialized with pre-trained embeddings, while unseen new tokens are randomly initialized.

3.1 Distillation

VT can be combined with other model compression methods like quantization, pruning and KD. For some of the methods, the combination is trivial, since they have no impact on the vocabulary. KD, however, requires the vocabularies of the student and teacher to be aligned. Hence, its integration with VT is non-trivial. Accordingly, we set up a KD procedure with VT, in order to determine the effects of applying both VT and KD to an LM.

Our distillation consists of two steps. In the first step, we replicate the distillation process used in (Sanh et al., 2019) for DistilBERT, in which the number of layers of the encoder is halved and a triple loss-function is applied: a distillation loss, a MLM loss, and a cosine embedding loss. However, unlike the original setup, we do not remove the token-type embeddings and pooler. after distilling the student on Dgen, we further distil the student using Din. However, instead of adapting the teacher before the second distillation, we simply distil the student a second time on the in-domain dataset. Finally, we apply VT using either FVT or PVT and fine-tune the student model on the in-domain datasets.

Our choice of applying VT after KD is based on findings by Kim and Hassan (2020), that different input embedding spaces will produce different output embedding spaces. This difference in spaces is not conducive to knowledge transfer during distillation. Hence, if VT were to be applied first to the student, its input embedding space would differ greatly from that of the pre-trained teacher during distillation.

4 Experiments

In the experiments we measure the impact of FVT on three main KPIs: quality (F1 score), size of the models and speedup in inference.

4.1 Experimental Setup

We consider for all our experiments the pre-trained cased version of BERTbase as our pre-trained language model. Its tokenizer is composed of 28996 wordpieces. We then define four vocabulary sizes for retraining our tokenizers. Specifically, we take the original vocabulary size and define it as a vocabulary size of 100 percent. We subsequently reduce this size to 75percent, 50percent, and 25percent, From now on, we will refer to such tokenizers as T100, T75, T50, T25 respectively, while the original vocabulary will be called Tgen.

Models are fine-tuned for 10 epochs with early stopping on the downstream task. We set the initial learning rate to 3×10^{-5} and batch size to 64 for each task. The sequence length is set to 64 for ADE and CoNLL03 and 128 for LEDGAR. Each configuration is repeated 3 times with different random initializations. MLM is performed for one epoch.

4.2 Datasets

To best assess the effectiveness of VT, we apply it on three different tasks from three heterogeneous linguistic domains: medical (ADE), legal (LEDGAR) and news (CoNLL03). Table 4 reports the dataset statistics.

ADE. The Adverse Drug Events (ADE) corpus is a binary sentence

classification dataset in the medical domain. This domain is particularly suitable for investigating the benefits of VT, since documents are characterized by the presence of frequent technical terms, such as drug and disease names, that are usually rare in common language. Domain-specific words are usually split into multiple tokens, yielding longer sequences and breaking the semantics of a word into multiple pieces. An example is shown in Figure 2.

LEDGAR. LEDGAR is a document classification corpus of legal provisions in contracts from the US Securities and Exchange Commission (SEC). The dataset is annotated with 100 different mutually-exclusive labels. It is also part of LexGLUE, a benchmark for legal language understanding.

CoNLL03. CoNLL03 is a popular Named Entity Recognition (NER) benchmark. It is made of news stories from the Reuters corpus. We chose this corpus because, differently from ADE and LEDGAR, the news domain typically uses a more standard language, hence we expect its distribution to differ less from the one captured by a general-purpose tokenizers in the web. Statistics in Table 1 confirms this hypothesis. We can observe that the sequence compression gain obtained with domain- specific tokenizers is less significant with respect to LEDGAR and ADE.

Table 1: Number of examples of each dataset.

Dataset	Train	Validation	Test
ADE	16716	3344	836
LEDGAR	60000	10000	10000
CoNLL03	14042	3251	3454

4.3 Results

We report an extensive evaluation of FVT on different setups and perspectives.

In-domain Tokenization. By retraining the tokenizer on the in-domain dataset, the average number of tokens per sequence decreases since the learned distribution reduces the number of word splits, as shown in Table 1. In the medical domain, which is particularly specialized, we notice a remarkable 32

Table 2: Average sequence length on the three datasets with different tokenizers. Tgen is the generic tokenizer (BERT cased), the same in each corpus, while T percent are the tokenizers trained in the vertical domain itself.

Dataset	Tgen	T100	T75	T50	T25
ADE	31	21	22	23	26
LEDGAR	155	131	131	132	135
CoNLL03	19	17	17	18	20

Vocabulary Transfer. From the results shown in Tables 2 and 3, we note a few interesting findings. First, FVT vectors initialization method consistently outperforms the baseline PVT, which confirms the positive contribution of Equation 2. Second, transferring vocabulary with FVT causes limited drops in performance, especially in LEDGAR (the largest one), where F1 slightly increases despite a 75

Table 3: F1 results on the three benchmarks. A pre- trained language model fine-tuned on the task (Tgen) is compared with models having differently sized in-domain tokenizers (T100, T75, T50, T25) adapted by transferring information with FVT or PVT.

Transfer	ADE	LEDGAR	CoNLL03
Tgen	90.80	80.93	89.43
T100 + FVT	90.77	80.60	87.87
T75 + FVT	90.40	80.93	87.90
T50 + FVT	90.07	80.93	86.87
T25 + FVT	90.27	81.03	86.17
T100 + PVT	82.57	80.07	84.53
T75 + PVT	82.47	80.33	84.63
T50 + PVT	83.07	80.23	84.43
T25 + PVT	83.57	80.20	83.47

Table 4: F1 results on the three benchmarks. A distilled language model fine-tuned on the task (Tgen) is compared with models having differently sized in-domain tokenizers (T100, T75, T50, T25) adapted by transferring information with FVT or PVT.

	ADE	LEDGAR	CoNLL03
Tgen	90.47	78.37	86.90
T100 + FVT	89.47	78.33	84.63
T75 + FVT	88.57	78.90	84.23
T50 + FVT	88.43	79.30	83.80
T25 + FVT	88.23	78.10	83.13
T100 + PVT	79.13	76.97	81.13
T75 + PVT	78.87	76.93	81.40
T50 + PVT	76.30	77.37	81.63
T25 + PVT	77.90	77.33	79.50

Vocabulary Transfer and Distillation. The results summarized in Table 3 clearly indicate that KD is complementary to VT: there is no harm in applying them together, in terms of performance on the downstream task. Crucially, this guarantees a full exploitation of FVT in the scope of language model compression.

Compression and Efficiency. After showcasing that VT has limited impact on performance, we analyze and discuss its effects on efficiency and model compression. Table 5 reports the relative F1 drop on the downstream task with respect to the original LM (Ž206F1), the relative reduction in model size (Ž206Size) and the speedup gained by FVT alone and by FVT combined with KD for varying vocabulary sizes. Either way, FVT achieves a remarkable 15

Furthermore, the reduced input length enabled by in-domain tokenization brings a reduction in inference time. The more a language is specialized, the higher is the speedup with in-domain tokenizers. This is also confirmed by the experiments, where the major benefits are obtained on the medical domain, with a x1.40 speedup. In CoNLL03 instead where language is much less specialized, speedup reduces and even disappears with T25. Distillation further pushes compression and speedup in any benchmark and setup, up to about 55

In summary, depending on the application needs, VT enables a strategic trade-off between compression rate, inference speed and accuracy.

5 Conclusion

The viability and success of industrial NLP applications often hinges on a delicate trade-off between computational requirements, responsiveness and output quality. Hence, language model compression methods are an active area of research whose practical ramifications are self-evident. One of the factors that greatly contribute to a model's inference speed and memory footprint is vocabulary size. VT has been recently proposed for improving performance, but never so far in the scope of model

Table 5: The first row (Tgen) reports absolute values of the LM fine-tuned on the downstream task without VT or KD. The rows below show values relative to Tgen.

2*Transfer	ADE			LEDGAR			CoNLL03		
	Ž206F1	Ž206Size	Speedup	Ž206F1	Ž206Size	Speedup	Ž206F1	Ž206Size	Speedup
Tgen	90.80	433.32	1.00	80.93	433.62	1.00	89.43	430.98	1.00
T100 + FVT	-0.04	0.00	1.40	-0.41	0.00	1.21	-1.75	0.00	1.07
T75 + FVT	-0.44	-5.14	1.35	0.00	-5.14	1.21	-1.71	-5.17	1.07
T50 + FVT	-0.81	-10.28	1.32	0.00	-10.27	1.10	-2.87	-10.33	1.02
T25 + FVT	-0.59	-15.42	1.20	0.12	-15.41	1.09	-3.65	-15.50	0.99
Distil + T100 + FVT	-1.47	-39.26	2.76	-3.21	-39.24	2.38	-5.37	-39.48	2.11
Distil + T75 + FVT	-2.46	-44.40	2.64	-2.51	-44.37	2.38	-5.81	-44.64	2.11
Distil + T50 + FVT	-2.61	-49.54	2.59	-2.02	-49.51	2.16	-6.30	-49.81	2.01
Distil + T25 + FVT	-2.83	-54.68	2.37	-3.50	-54.64	2.14	-7.04	-54.98	1.96

compression. In this work, we run an extensive experimental study on the application of a lightweight method for VT, called FVT. An analysis conducted on various downstream tasks, application domains, vocabulary sizes and on its possible combination with knowledge distillation indicates that FVT enables a strategic trade-off between compression rate, inference speed and accuracy, especially, but not only, in more specialized domains. Importantly, FVT appears to be orthogonal to other model compression methods.

In the future, we plan to fully integrate Vocabulary Transfer within Knowledge Distillation during the learning process in order to maximize the information transfer.