Fast Vocabulary Transfer for Language Model Compression

Leonidas Gee
Expert.ai, Italy
lgee@expert.ai

Andrea Zugarini
Expert.ai, Italy
azugarini@expert.ai

Leonardo Rigutini
Expert.ai, Italy
and
University of Siena
lrigutini@expert.ai

Paolo Torroni
Department of Computer Science
and Engineering,
University of Bologna
paolo.torroni@unibo.it

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) (Devlin et al., 2018; Liu et al., 2019; He et al., 2020). Because larger LMs, on average, exhibit higher accuracy, a common trend has been to increase the model's size. Some LMs like GPT-3 (Brown et al., 2020) and BLOOM[1] 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 (Gupta et al., 2015; Shen et al., 2020). Notably, Knowledge Distillation (KD) (Hinton et al., 2015) 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 (Sennrich et al., 2016; Schuster and Nakajima, 2012; Kudo and Richardson, 2018) 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 to-

[1] https://bigscience.huggingface.co/blog/bloom

Proceedings of EMNLP 2022 Industry Track, pages 419–426
December 9–11, 2020.

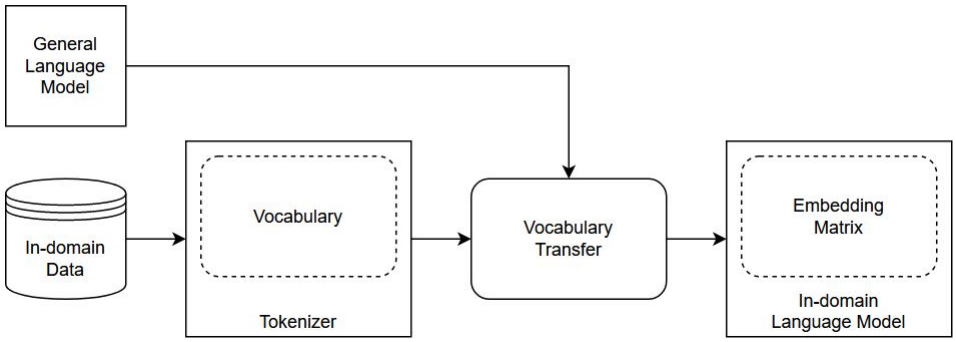

Figure 1: Sketch of the VT procedure. First, the vocabulary is constructed on the in-domain data, then an embedding is assigned to each token, transferring information from the pre-trained representations of the general-purpose language model.

kenization, 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 (Vaswani et al., 2017). 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 Works

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 (Gupta et al., 2015; Shen et al., 2020), pruning (Zhu and Gupta, 2017; Michel et al., 2019) knowledge distillation (Sanh et al., 2019; Jiao et al., 2020; Wang et al., 2020), and combinations thereof (Polino et al., 2018).

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% smaller size than BERT and retains 97% of its language understanding capabilities. This enables a 60% inference-time speedup. Further compression was achieved by Jiao et al. (2020) by adding transformer-layer, prediction-layer and embedding-layer distillation. The resulting model, TinyBERT, is 10 times smaller than BERT, with only four layers and reduced embeddings sizes. Related methods were proposed (Sun et al., 2020; Wang et al., 2020), achieving similar compression rates. All these works focus on the distillation of general-purpose language models. Gordon and Duh (2020) investigated the interaction between KD and Domain Adaptation.

Little focus has been devoted thus far to the role of tokenization in the context of model compression. Even in domain adaptation (Gordon and Duh, 2020), 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 $\mathcal{D}_{gen}$ and associated with a vocabulary $\mathcal{V}_{gen}$. 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 $\mathcal{V}_{gen}$. More specifically, a tokenizer is a function that maps a textual string into a sequence of symbols of a given vocabulary $\mathcal{V}$. Let $\mathcal{T}$ be a tokenizer associated with a vocabulary $\mathcal{V}$ and a string $\boldsymbol{s}$, we have $\mathcal{T} : \boldsymbol{s} \rightarrow (\boldsymbol{t}_1, \ldots, \boldsymbol{t}_n), \boldsymbol{t}_i \in \mathcal{V}, \forall i = 1, \ldots, n$. Hence, the vocabulary of the tokenizer determines how words in a text are split, whether as words, subwords, 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 $\mathcal{D}_{in}$, also referred as *in-domain*. For the reasons discussed earlier, a vocabulary $\mathcal{V}_{in}$ specialized on $\mathcal{D}_{in}$ itself better fits the language distribution than $\mathcal{V}_{gen}$. Unfortunately, with a new vocabulary, embedding representations associated with the tokens of $\mathcal{V}_{gen}$ would be lost. Thus, VT aims to initialize $\mathcal{V}_{in}$ by re-using most of the information learned from the LM pre-trained on $\mathcal{D}_{gen}$. Once the new tokenizer $\mathcal{T}_{in}$ has been trained on the in-domain dataset $\mathcal{D}_{in}$ using a given vocabulary size, $\mathcal{T}_{in}$ will be different from the LM's tokenizer $\mathcal{T}_{gen}$. However, the two tokenizers' vocabularies $\mathcal{V}_{gen}$ and $\mathcal{V}_{in}$ may still have a large portion of their symbols in common. Our objective is to transfer most of the information from $\mathcal{V}_{gen}$ into $\mathcal{V}_{in}$. To this end, we first define a mapping between each symbol in $\mathcal{V}_{in}$ and a set of symbols in $\mathcal{V}_{gen}$. Then, we define an assignment criterion, based on the mapping, to obtain the embeddings for the tokens of $\mathcal{T}_{in}$. The existence of a meaningful mapping is the only underlying assumption for VT, which holds for all the phonographic languages.

One such criterion, called Vocabulary Initialization with Partial Inheritance (VIPI), was defined by Samenko et al. (2021). Whenever a token is in $\mathcal{V}_{in}$ but not in $\mathcal{V}_{gen}$, VIPI calculates all the partitions of the new token with tokens from $\mathcal{V}_{gen}$, 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 $t_i \in \mathcal{V}_{in}$ is partitioned using $\mathcal{T}_{gen}$. If t_i belongs to both vocabularies, $t_i \in \mathcal{V}_{in} \cap \mathcal{V}_{gen}$, then $\mathcal{T}_{gen}(t_i) = t_i$ and the in-domain LM embedding $E_{in}(t_i)$ is the same as the embedding in the general LM:

$$E_{in}(t_i) = E_{gen}(t_i). \quad (1)$$

If instead $t_i \in \mathcal{V}_{in} \setminus \mathcal{V}_{gen}$, then the in-domain embedding is the average of the embeddings associated with the tokens produced by $\mathcal{T}_{gen}$:

$$E_{in}(t_i) = \frac{1}{|\mathcal{T}_{gen}(t_i)|} \cdot \sum_{t_j \in \mathcal{T}_{gen}(t_i)} E_{gen}(t_j). \quad (2)$$

Please notice that Equation 2 is a generalization of Equation 1. Indeed, in case $t_i \in \mathcal{V}_{in} \cap \mathcal{V}_{gen}$, 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),

Input:
`He was initially treated with interferon alfa.`

$\mathcal{T}_{gen}$**:**
`He, was, initially, treated, with,` **`inter`**`,`**`##fer`**`,`
`##on``,` **`al`**`,` **`##fa`**`, .`

$\mathcal{T}_{100}$**:**
`He, was, initially, treated, with,` **`interferon`**`,`
`alfa``, .`

Figure 2: Example of different tokenizations using a pre-trained or an adapted tokenizer. In the latter case, domain-specific words are not broken down into multiple word pieces.

Dataset	$\mathcal{T}_{gen}$	$\mathcal{T}_{100}$	$\mathcal{T}_{75}$	$\mathcal{T}_{50}$	$\mathcal{T}_{25}$
ADE	31	21	22	23	26
LEDGAR	155	131	131	132	135
CoNLL03	19	17	17	18	20

Table 1: Average sequence length on the three datasets with different tokenizers. $\mathcal{T}_{gen}$ is the generic tokenizer (BERT cased), the same in each corpus, while $\mathcal{T}_{\%}$ are the tokenizers trained in the vertical domain itself, where $\%$ indicates the percentage of the original vocabulary size that has been set for training it.

whereby only the tokens belonging to both vocabularies $t_i \in \mathcal{V}_{in} \cap \mathcal{V}_{gen}$ are initialized with pre-trained embeddings, while unseen new tokens are randomly initialized.

Transfer	ADE	LEDGAR	CoNLL03
$\mathcal{T}_{gen}$	**90.80**	80.93	**89.43**
$\mathcal{T}_{100}$ + FVT	90.77	80.60	87.87
$\mathcal{T}_{75}$ + FVT	90.40	80.93	87.90
$\mathcal{T}_{50}$ + FVT	90.07	80.93	86.87
$\mathcal{T}_{25}$ + FVT	90.27	**81.03**	86.17
$\mathcal{T}_{100}$ + PVT	82.57	80.07	84.53
$\mathcal{T}_{75}$ + PVT	82.47	80.33	84.63
$\mathcal{T}_{50}$ + PVT	83.07	80.23	84.43
$\mathcal{T}_{25}$ + PVT	83.57	80.20	83.47

Table 2: F1 results on the three benchmarks. A pre-trained language model fine-tuned on the task ($\mathcal{T}_{gen}$) is compared with models having differently sized in-domain tokenizers ($\mathcal{T}_{100}, \mathcal{T}_{75}, \mathcal{T}_{50}, \mathcal{T}_{25}$) adapted by transferring information with FVT or PVT.

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. Inspired by Gordon and Duh (2020), after distilling the student on $\mathcal{D}_{gen}$, we further distil the student using $\mathcal{D}_{in}$. 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.

	Distillation		
Transfer	**ADE**	**LEDGAR**	**CoNLL03**
$\mathcal{T}_{gen}$	**90.47**	78.37	**86.90**
$\mathcal{T}_{100}$ + FVT	89.47	78.33	84.63
$\mathcal{T}_{75}$ + FVT	88.57	78.90	84.23
$\mathcal{T}_{50}$ + FVT	88.43	**79.30**	83.80
$\mathcal{T}_{25}$ + FVT	88.23	78.10	83.13
$\mathcal{T}_{100}$ + PVT	79.13	76.97	81.13
$\mathcal{T}_{75}$ + PVT	78.87	76.93	81.40
$\mathcal{T}_{50}$ + PVT	76.30	77.37	81.63
$\mathcal{T}_{25}$ + PVT	77.90	77.33	79.50

Table 3: F1 results on the three benchmarks. A distilled language model fine-tuned on the task ($\mathcal{T}_{gen}$) is compared with models having differently sized in-domain tokenizers ($\mathcal{T}_{100}, \mathcal{T}_{75}, \mathcal{T}_{50}, \mathcal{T}_{25}$) adapted by transferring information with FVT or PVT.

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 BERT$_{base}$ (Devlin et al., 2018) 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%. We subsequently reduce this size to 75%, 50%, and 25%. From now on, we will refer to such tokenizers as $\mathcal{T}_{100}, \mathcal{T}_{75}, \mathcal{T}_{50}, \mathcal{T}_{25}$ respectively, while the original vocabulary will be called $\mathcal{T}_{gen}$.

Models are fine-tuned for 10 epochs with early stopping on the downstream task. We set the initial learning rate to $3 \cdot 10^{-5}$ 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

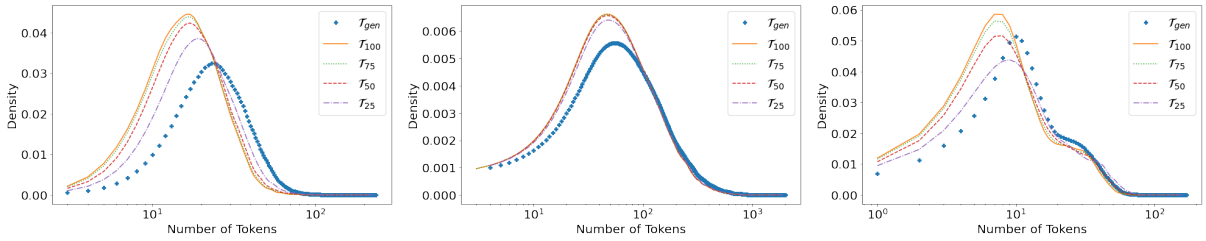

Figure 3: Sequence length distribution of each tokenizer on ADE, LEDGAR and CoNLL03 (left to right).

(LEDGAR) and news (CoNLL03). Table 4 reports the dataset statistics.

ADE. The Adverse Drug Events (ADE) corpus (Gurulingappa et al., 2012) 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 (Tuggener et al., 2020) 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 (Chalkidis et al., 2022), a benchmark for legal language understanding.

CoNLL03. CoNLL03 (Tjong Kim Sang and De Meulder, 2003) 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.

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% reduction of the average number of tokens per sequence. We expect this to yield a noticeable impact on inference time speedup. Furthermore, we can notice in Figure 3 that the sequence length distribution shifts to the left for the learned tokenizers. It can also be observed that by reducing the vocabulary size of the *in-domain* tokenizer, the sequence length distribution will begin to shift back to the right. Indeed, with fewer tokens in its vocabulary, the tokenizer will need to break down words more frequently into subwords.

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

Table 4: Number of examples of each dataset.

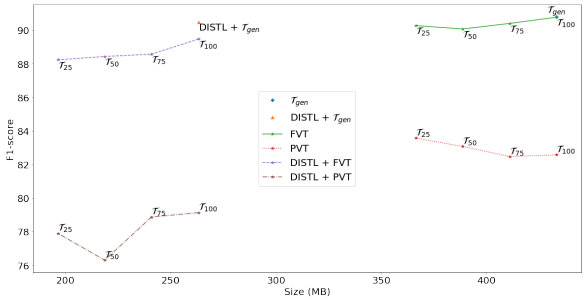

Figure 4: F1-score vs model size of VT with or without KD on ADE. VT and KD together can further compress a LM's size in exchange for a limited performance drop. FVT is better than PVT. A smaller vocabulary size does not always imply a lower performance.

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. Sec-

Transfer	ADE			LEDGAR			CoNLL03		
	ΔF1	**ΔSize**	**Speedup**	**ΔF1**	**ΔSize**	**Speedup**	**ΔF1**	**ΔSize**	**Speedup**
$\mathcal{T}_{gen}$	90.80	433.32	1.00	80.93	433.62	1.00	89.43	430.98	1.00
$\mathcal{T}_{100}$ + FVT	**-0.04**	0.00	1.40	-0.41	0.00	1.21	-1.75	0.00	1.07
$\mathcal{T}_{75}$ + FVT	-0.44	-5.14	1.35	0.00	-5.14	1.21	**-1.71**	-5.17	1.07
$\mathcal{T}_{50}$ + FVT	-0.81	-10.28	1.32	0.00	-10.27	1.10	-2.87	-10.33	1.02
$\mathcal{T}_{25}$ + FVT	-0.59	-15.42	1.20	**0.12**	-15.41	1.09	-3.65	-15.50	0.99
Distil + $\mathcal{T}_{gen}$	-0.36	-39.26	1.97	-3.16	-39.24	1.97	-2.83	-39.48	1.95
Distil + $\mathcal{T}_{100}$ + FVT	-1.47	-39.26	**2.76**	-3.21	-39.24	**2.38**	-5.37	-39.48	**2.11**
Distil + $\mathcal{T}_{75}$ + FVT	-2.46	-44.40	2.64	-2.51	-44.37	**2.38**	-5.81	-44.64	**2.11**
Distil + $\mathcal{T}_{50}$ + FVT	-2.61	-49.54	2.59	-2.02	-49.51	2.16	-6.30	-49.81	2.01
Distil + $\mathcal{T}_{25}$ + FVT	-2.83	**-54.68**	2.37	-3.50	**-54.64**	2.14	-7.04	**-54.98**	1.96

Table 5: The first row ($\mathcal{T}_{gen}$) reports absolute values of the LM fine-tuned on the downstream task without VT or KD. The rows below show values relative to $\mathcal{T}_{gen}$.

ond, transferring vocabulary with FVT causes limited drops in performance, especially in LEDGAR (the largest one), where F1 slightly increases despite a 75% vocabulary reduction. We observed a higher degradation in CoNLL03. We believe this is due to the less specialized nature of the news domain, whereby the benefits of adapting the vocabulary to it are reduced. Overall, the effects of FVT on model performance do not have a steadily decreasing trend, as it might be presumed when reducing the vocabulary size, as also evident from Figure 4. In some cases, somewhat surprisingly, reducing the vocabulary size yields better model performance. In other cases, a 50% vocabulary size reduction yields better results than a full scale reduction or no reduction. Hence, vocabulary size should be considered as a hyper-parameter, where the model selection criteria may vary depending on the application's KPIs, such as acceptable F1 drop, disk occupation and delay constraints.

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 (ΔF1), the relative reduction in model size (ΔSize) and the speedup gained by FVT alone and by FVT combined with KD for varying vocabulary sizes. Either way, FVT achieves a remarkable 15%+ reduction with respect to BERT's learnable parameters, with almost no loss in F1.

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 $\mathcal{T}_{25}$. Distillation further pushes compression and speedup in any benchmark and setup, up to about 55% (of which 15% due to VT) and x2.75 respectively.

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 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. We also plan to define a unified metric that combines all the KPIs, to facilitate model selection.

Acknowledgments

This work is part of the SCUDO project, which was funded by the "FESR 2014-2020" Regional Operational Program of the Tuscany Region (Italy), Call 2: "Research and development projects of the SMEs".

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901.

Ilias Chalkidis, Abhik Jana, Dirk Hartung, Michael Bommarito, Ion Androutsopoulos, Daniel Katz, and Nikolaos Aletras. 2022. LexGLUE: A benchmark dataset for legal language understanding in English. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 4310–4330, Dublin, Ireland. Association for Computational Linguistics.

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

Mitchell Gordon and Kevin Duh. 2020. Distill, adapt, distill: Training small, in-domain models for neural machine translation. In Proceedings of the Fourth Workshop on Neural Generation and Translation, pages 110–118, Online. Association for Computational Linguistics.

Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. 2015. Deep learning with limited numerical precision. In International conference on machine learning, pages 1737–1746. PMLR.

Harsha Gurulingappa, Abdul Mateen Rajput, Angus Roberts, Juliane Fluck, Martin Hofmann-Apitius, and Luca Toldo. 2012. Development of a benchmark corpus to support the automatic extraction of drug-related adverse effects from medical case reports. Journal of Biomedical Informatics, 45(5):885 – 892. Text Mining and Natural Language Processing in Pharmacogenomics.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. DeBERTa: Decoding-enhanced BERT with disentangled attention. arXiv preprint arXiv:2006.03654.

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2(7).

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In Findings of the Association for Computational Linguistics: EMNLP 2020, pages 4163–4174.

Young Jin Kim and Hany Hassan. 2020. FastFormers: Highly efficient transformer models for natural language understanding. In Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing, pages 149–158, Online. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. arXiv preprint arXiv:1808.06226.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. arXiv preprint arXiv:1907.11692.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? Advances in neural information processing systems, 32.

Antonio Polino, Razvan Pascanu, and Dan Alistarh. 2018. Model compression via distillation and quantization. arXiv preprint arXiv:1802.05668.

Igor Samenko, Alexey Tikhonov, Borislav Kozlovskii, and Ivan P Yamshchikov. 2021. Fine-tuning transformers: Vocabulary transfer. arXiv preprint arXiv:2112.14569.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In 2012 IEEE international conference on acoustics, speech and signal processing (ICASSP), pages 5149–5152. IEEE.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1715–1725.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2020. Q-BERT: Hessian based ultra low precision quantization of BERT. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 8815–8821.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. MobileBERT: a compact task-agnostic BERT for resource-limited devices. arXiv preprint arXiv:2004.02984.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003, pages 142–147.

Don Tuggener, Pius von Däniken, Thomas Peetz, and Mark Cieliebak. 2020. Ledgar: A large-scale multi-label corpus for text classification of legal provisions in contracts. In Proceedings of the 12th Language Resources and Evaluation Conference, pages 1235–1241.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. Advances in neural information processing systems, 30.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. Advances in Neural Information Processing Systems, 33:5776–5788.

Michael Zhu and Suyog Gupta. 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression. arXiv preprint arXiv:1710.01878.