Multi-word Tokenization for Sequence Compression

Leonidas Gee
University of Sussex, United Kingdom
jg717@sussex.ac.uk

Leonardo Rigutini
expert.ai, Siena, Italy
lrigutini@expert.ai

Marco Ernandes
expert.ai, Siena, Italy
mernandes@expert.ai

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

Abstract

Large Language Models have proven highly successful at modelling a variety of tasks. However, this comes at a steep computational cost that hinders wider industrial uptake. In this paper, we present MWT: a Multi-Word Tokenizer that goes beyond word boundaries by representing frequent multi-word expressions as single tokens. MWTs produce a more compact and efficient tokenization that yields two benefits: (1) Increase in performance due to a greater coverage of input data given a fixed sequence length budget; (2) Faster and lighter inference due to the ability to reduce the sequence length with negligible drops in performance. Our results show that MWT is more robust across shorter sequence lengths, thus allowing for major speedups via early sequence truncation.

1 Introduction

The field of Natural Language Processing (NLP) has seen major breakthroughs with the advent of Large Language Models (LLMs) (Vaswani et al., 2017; Devlin et al., 2018; Touvron et al., 2023; OpenAI, 2023). Despite their successes, LLMs like ChatGPT (OpenAI, 2023; Brown et al., 2020) possess hundreds of billions of parameters that entail enormous computational cost by design. Traditional model compression methods such as Knowledge Distillation (Hinton et al., 2015), Pruning (Michel et al., 2019; Zhu and Gupta, 2017), and Quantization (Shen et al., 2020; Gupta et al., 2015) have focused on creating lighter models either by shrinking the architectural size or by reducing the number of FLOPs.

Recently, LLMs have been shown to produce impressive performance on inputs that have been carefully designed to contain all the necessary information for a given instruction. As such, there is an increasing trend in designing longer and longer prompts that has led to a significant rise in computational cost. To address this, interest has grown in compressing the input sequences from the tokenizer (Gee et al., 2022; Mu et al., 2023; Petrov et al., 2023). Indeed, various works have shown the importance of tokenization in determining the length of a sequence in specialized domains (Gee et al., 2022) or on underrepresented languages (Petrov et al., 2023).

In this paper, we propose a method for reducing the computational cost of a LLM by compressing the textual inputs using Multi-Word Tokenizers (MWTs). To achieves this, we enrich the vocabulary of the tokenizer with statistically determined multi-word expressions. By encoding the frequent n-grams with single tokens, the sequences produced are both shorter and more informative, thus allowing for major speedups via early sequence truncation. Additionally, MWTs are shown to be compatible with the aforementioned traditional compression methods. Experimentally, we assess MWTs on three text classification datasets. We show how our approach still performs well when combined with distilled models (Sanh et al., 2019) and other sequence compression techniques (Gee et al., 2022). The code for our paper is publicly available[1].

The rest of the paper is organized as follows. First, we review the related works in Section 2. Then, we describe our approach in Section 3 and present the experiments in Section 4. Finally, we draw our conclusions in Section 5.

2 Related Works

Most model compression research falls into one of the following categories: Knowledge Distillation (Hinton et al., 2015; Sanh et al., 2019; Jiao et al., 2020; Wang et al., 2020; Sun et al., 2020), Pruning (Zhu and Gupta, 2017; Michel et al., 2019), and Quantization (Shen et al., 2020). The family of approaches is somewhat complementary and

[1] https://github.com/LeonidasY/fast-vocabulary-transfer/tree/emnlp2023

Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track, pages 612–621
December 6-10, 2023 ©2023 Association for Computational Linguistics

Input: an energizable member is operably coupled to the outer sleeve .

$\mathcal{T}_{gen}$: an, en, ##er, ##gi, ##zable, member, is, opera, ##bly, coupled, to, the, outer, sleeve, .

$\mathcal{T}_{gen}^{1000}$: an, en, ##er, ##gi, ##zable, **member_is**, opera, ##bly, **coupled_to**, **the_outer**, sleeve, .

$\mathcal{T}_{100}$: an, **energizable**, member, is, **operably**, coupled, to, the, outer, sleeve, .

$\mathcal{T}_{100}^{1000}$: an, **energizable**, **member_is**, **operably**, **coupled_to**, **the_outer**, sleeve, .

Figure 1: Tokenization using generic $\mathcal{T}_{gen}$ and adapted $\mathcal{T}_{100}$ tokenizers. $\mathcal{T}_{gen}^{1000}$ and $\mathcal{T}_{100}^{1000}$ are extended with the top-1000 bigrams. Tokens obtained with domain-adaptation or MWT are highlighted in orange and blue respectively. MWTs are shown to be highly complementary to existing tokenizers for sequence compression.

can be applied individually or jointly. Each approach alters the model's size to obtain a more efficient architecture. Differently, other works such as FlashAttention (Dao et al., 2022) seek to optimize a model's implementation. In particular, LLMs are sped up by reducing the number of memory accesses for the self-attention mechanism.

Sequence Compression. An emerging direction for reducing the cost of LLMs involves the designing of shorter input sequences. Prompting techniques such as Mu et al. (2023) compress repetitive lengthy prompts into gist tokens. Other works emphasize the role of tokenization in sequence compression. In Petrov et al. (2023), the authors show how the tokenizer of most LLMs strongly favor the English language over other languages. For underrepresented languages, the same translated sentence may consist of inputs that are up to 15 times longer. Analogously, Gee et al. (2022) investigated the tokenization efficiency of general-purpose tokenizers in vertical domains such as medicine and law. They proposed a transfer learning technique that adapts the vocabulary of a LLM to specific language domains. An effect of a dedicated vocabulary is a more efficient tokenization that reduces the number of sub-word tokens in a sequence.

In this work, we push this effect further, going beyond word boundaries by introducing Multi-Word Expressions (MWEs) in the form of n-grams into the tokenizer as shown in Figure 1. The underlying intuition behind this is that a more compact tokenization can save computations by allowing the model to process shorter sequences without a significant loss of information. The usage of MWEs is not novel with several works (Lample et al., 2018; Otani et al., 2020; Kumar and Thawani, 2022) introducing phrases or n-grams to improve the quality of machine translation. In Kumar and Thawani (2022), the authors generalized BPE (Sennrich et al., 2016) to multi-word tokens. However, to the best of our knowledge, we are the first to investigate MWEs in the context of sequence compression.

3 Multi-word Tokenizer

Tokenization is a necessary step in the feeding of textual data to a LLM. Typically, tokenizers split a text into a sequence of symbols which can be entire words or only subparts. To do this, a vocabulary is first constructed by statistically learning the most frequent tokens from a large general-purpose corpus (Sennrich et al., 2016; Schuster and Nakajima, 2012; Kudo and Richardson, 2018). The resulting tokenizer can then be used to segment an input text by greedily looking for the solution with the least number of tokens. Building upon this, we inject into the tokenizer new symbols formed by n-grams of words. We do this by first selecting the most frequent n-grams to include in its vocabulary. Then, we place an n-gram merging step within the tokenization pipeline as sketched in Figure 2. The added n-grams will be treated as single tokens further down the tokenization pipeline.

N-gram Selection. In order to maximize the sequence reduction, we statistically estimate the top-K most frequent n-grams in a reference training corpus. Although the approach is greedy, hence sub-optimal, it still effectively yields significant compression while being extremely fast and easy to compute. More formally, given a corpus $\mathcal{D}$ and $N \geq 2$, we compute all the possible n-grams $g_n \in \mathcal{D}$, where $n = 2, \ldots, N$. Then, we count their frequency $f(g_n), \forall g_n \in \mathcal{D}$. The K most frequent n-grams $\mathcal{G}_K$ are included in the vocabulary $\mathcal{V} \leftarrow \mathcal{V} \cup \mathcal{G}_K$ of the tokenizer $\mathcal{T}$.

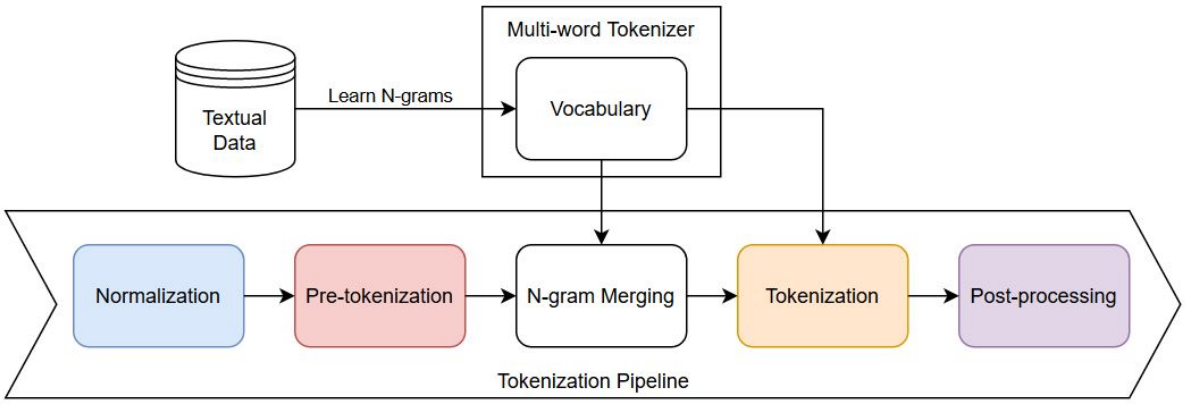

Figure 2: Sketch of the Multi-word Tokenizer pipeline. First, n-grams are statistically learned from the training set. Then, the top-K n-grams are added to the vocabulary of the tokenizer. N-grams are merged from left to right within a sequence after pre-tokenization.

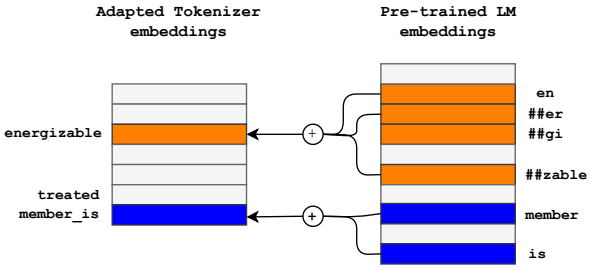

Figure 3: Fast Vocabulary Transfer. The pre-trained embeddings of existing tokens are combined to form the embeddings of the newly adapted vocabulary.

Fast Vocabulary Transfer. Given that the vocabulary of the tokenizer has changed, the newly added symbols $\mathcal{G}_K$ must be included into the embedding matrix of the language model as well. To avoid retraining the entire model from scratch which is highly resource-demanding, or a random initialization of new tokens which would perform poorly, we make use of Fast Vocabulary Transfer (FVT) instead (Gee et al., 2022).

FVT is a transfer learning technique that assigns embeddings to new tokens by combining existing elements of the embedding matrix as shown in Figure 3. After initializing the multi-word embeddings with FVT, we found it beneficial to tune the model with Masked-Language Modeling (MLM) as done by Gee et al. (2022). We believe this is helpful as it aids the model in further readjusting the embeddings of the new tokens.

4 Experiments

Given a fixed number of tokens, a more compact input sequence preserves a greater amount of information. This can be used to either achieve a better performance with limited benefits in speedup, or vice versa, i.e. making the model faster with negligible drops in performance. The experiments aim to analyze how these two aspects interact with one another. We focus on text classification as it is a problem of particular interest for many industry-oriented applications.

4.1 Experimental Setup

Our experiments were conducted on the cased versions of BERT$_{base}$ (Devlin et al., 2018) and DistilBERT$_{base}$ (Sanh et al., 2019). Additionally, we consider an adapted tokenizer with a vocabulary size equal to that of the generic tokenizer from a pre-trained model as done by Gee et al. (2022). We refer to the generic and adapted tokenizers as $\mathcal{T}_{gen}$ and $\mathcal{T}_{100}$ respectively. Both tokenizers are extended with the top-K n-grams of 1000, 2500, and 5000. Overall, we compare eight different tokenizers indicated as: $\mathcal{T}_{gen}, \mathcal{T}_{gen}^{1000}, \mathcal{T}_{gen}^{2500}, \mathcal{T}_{gen}^{5000}$ and $\mathcal{T}_{100}, \mathcal{T}_{100}^{1000}, \mathcal{T}_{100}^{2500}, \mathcal{T}_{100}^{5000}$.

Implementation Details. We train each model with 5 different random initializations. The macro-F1 and inference speedup are measured as metrics. The average of all 5 initializations is taken as the final value of each metric. The inference speedup measurements were done on a V100-PCIE GPU with 16GBs of dedicated RAM.

Following Gee et al. (2022), we first apply one epoch of MLM using the in-domain dataset. Next, the model is fine-tuned for 10 epochs with early stopping on the downstream task. We set the initial learning rate to $3 \cdot 10^{-5}$ for both MLM and downstream fine-tuning, while the batch size is set

Dataset	$\mathcal{T}_{gen}$	$\mathcal{T}_{gen}^{1000}$	$\mathcal{T}_{gen}^{2500}$	$\mathcal{T}_{gen}^{5000}$	$\mathcal{T}_{100}$	$\mathcal{T}_{100}^{1000}$	$\mathcal{T}_{100}^{2500}$	$\mathcal{T}_{100}^{5000}$
ADE	31	26	25	23	21	18	17	16
LEDGAR	155	118	107	98	131	97	90	84
PATENT	134	110	105	100	118	94	90	86

Table 1: Average sequence length from tokenization. The generic $\mathcal{T}_{gen}$ and adapted $\mathcal{T}_{100}$ tokenizers are extended with varying top-Ks of 1000, 2500, and 5000.

to 8 and 32 for MLM and downstream fine-tuning respectively.

Choice of N. An important hyperparameter is N, i.e. the maximum number of words constituting an n-gram. In our experiments, N is set to 2 as we believe that using bigrams only provides better generalization properties. Increasing the value of N may lead to an overspecialization of n-grams which could overfit on small textual corpora.

4.2 Datasets

To determine the effectiveness of MWTs, we select 3 different text classification tasks from diverse linguistic domains, namely medical (ADE), legal (LEDGAR), and tech (PATENT).

ADE. A sentence classification dataset of determining whether a sentence is Adverse Drug Event (ADE)-related or not (Gurulingappa et al., 2012). The sentences are characterized by the presence of medical terminologies of drugs and their adverse effects. We use the same train, validation, and test splits as in Gee et al. (2022).

LEDGAR. A document classification dataset of contracts obtained from the US Securities and Exchange Commission (SEC) filings (Tuggener et al., 2020). The task is to determine whether the main topic of the contract provision from a set of 100 mutually-exclusive labels. The dataset is also part of LexGLUE (Chalkidis et al., 2022), which is a benchmark for legal language understanding.

PATENT. A document classification dataset[2] of US patent applications filed under the Cooperative Patent Classification (CPC) code (Sharma et al., 2019). A human written abstractive summary is provided for each patent application. The task is to determine the category that a patent application belongs to from 9 unbalanced classes.

[2] https://huggingface.co/datasets/ccdv/patent-classification

4.3 Results

Preliminary Analysis. Before measuring the effects of MWTs on LLMs, we analyze how the average sequence length changes for each dataset depending on the tokenizer. From Table 1, increasing the top-K most frequent n-grams naturally yields a greater compression. However, even a 1000 bigrams is enough to achieve a reduction of about 20%. When multi-words are combined with an adapted tokenizer $\mathcal{T}_{100}$, the joint sequence narrowing effects appear to be highly complementary, achieving a compression rate close to 50% in ADE. In practice, a 50% reduction means that on average we can store the same amount of text in half the sequence length. Consequently, we could in principle reduce a LLM's maximum sequence length by a factor of 2.

Multi-word Tokenization. As a first evaluation, we assess the macro-F1 and inference speedups achieved by fine-tuned BERT models with multi-word tokenizers: $\mathcal{T}_{gen}^{1000}, \mathcal{T}_{gen}^{2500}, \mathcal{T}_{gen}^{5000}$. The pre-trained BERT with a generic tokenizer $\mathcal{T}_{gen}$ is considered as the reference model. From Table 2, MWTs are shown to either improve the reference performance or induce a relatively negligible degradation. At the same time, the sequence compression from MWTs yields a natural speedup that depending on the dataset varies from about x1.1 to x1.4.

MWT and Domain Adaptation. Additionally, we investigate the application of MWTs with tokenizers adapted to the dataset: $\mathcal{T}_{100}^{1000}, \mathcal{T}_{100}^{2500}, \mathcal{T}_{100}^{5000}$. With the exception of PATENT, most models are shown to achieve significant inference speedups of up to x1.8 with minimal degradation in performance from Table 2. We hypothesize that this is due to the fact that the language domain of PATENT is not as specialized as ADE and LEDGAR, which reduces the benefits of using an adapted tokenizer.

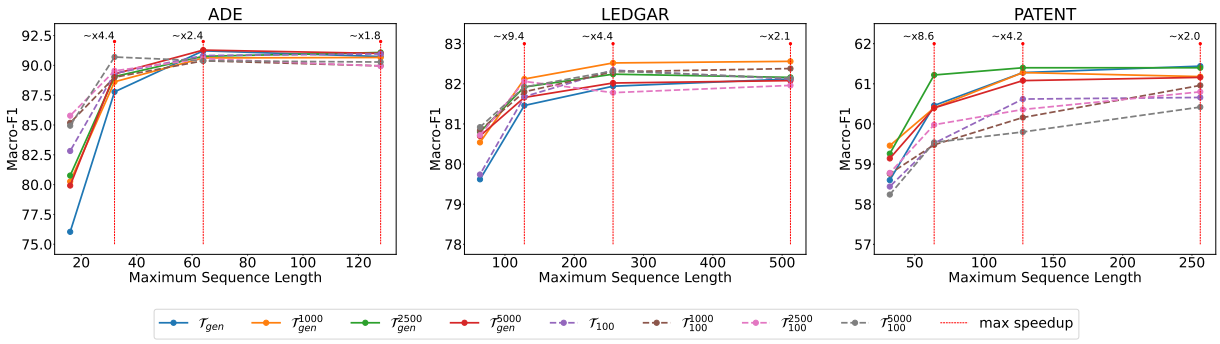

Figure 4: Plot of macro-F1 against maximum sequence length. The generic $\mathcal{T}_{gen}$ and adapted $\mathcal{T}_{100}$ tokenizers are represented by solid and dashed lines respectively. MWTs are shown to be more robust on shorter sequence lengths, thus allowing for major speedups via early sequence truncation.

Method	ADE		LEDGAR		PATENT	
	ΔF1	Speedup	ΔF1	Speedup	ΔF1	Speedup
$\mathcal{T}_{gen}$	90.74 ± 0.84	1.00	82.12 ± 0.33	1.00	**61.44 ± 0.38**	1.00
$\mathcal{T}_{gen}^{1000}$	-0.09 ± 0.70	1.32	**0.54 ± 0.24**	1.14	-0.42 ± 0.54	1.11
$\mathcal{T}_{gen}^{2500}$	**0.37 ± 0.54**	1.38	0.05 ± 0.44	1.23	-0.07 ± 0.46	1.16
$\mathcal{T}_{gen}^{5000}$	0.29 ± 0.68	1.43	-0.05 ± 0.41	1.33	-0.46 ± 0.69	1.19
$\mathcal{T}_{100}$	0.24 ± 0.67	1.51	0.00 ± 0.41	1.10	-1.27 ± 0.39	1.06
$\mathcal{T}_{100}^{1000}$	-0.86 ± 1.21	1.71	0.32 ± 0.58	1.36	-0.78 ± 0.62	1.24
$\mathcal{T}_{100}^{2500}$	-0.88 ± 0.72	1.78	-0.19 ± 0.57	1.47	-1.04 ± 0.42	1.30
$\mathcal{T}_{100}^{5000}$	-0.51 ± 0.65	**1.79**	0.02 ± 0.58	**1.57**	-1.66 ± 0.44	**1.34**

Table 2: Absolute values of BERT fine-tuned on the downstream task using a sequence length of 128, 512 and 256 for ADE, LEDGAR and PATENT respectively. $\mathcal{T}_{gen}$ is shown on the first row, while relative values to $\mathcal{T}_{gen}$ are shown on subsequent rows.

MWT and Truncation. Based on the preliminary analysis, we analyze how truncating sequences with different maximum lengths affects both the performance and inference speedup. Reducing the maximum sequence length has a double impact on the inference speedup given a fixed amount of resources. First, latency linearly grows with respect to the sequence length. Second, reducing the sequence length releases GPU resources that can be used to enlarge the batch size. We consider 4 maximum sequence lengths for each dataset by progressively halving the initial maximum sequence length, i.e. $\{128, 64, 32, 16\}$ for ADE, $\{256, 128, 64, 32\}$ for LEDGAR, and $\{512, 256, 128, 64\}$ for PATENT.

From Figure 4, we can see the performance of $\mathcal{T}_{gen}$ dropping more rapidly than MWTs as truncation increases (maximum sequence length decreases). In the extreme 8-times truncation, the performance of $\mathcal{T}_{gen}$ falls dramatically for both ADE and LEDGAR. However, MWTs are shown to be more robust to truncation, hence their degradation in performance is smoother and without sudden collapses. In both ADE and LEDGAR, a 4-times truncation leads to nearly identical or better performance, while bringing significant inference speedups of ∼x2.4 and ∼x4.4 respectively. If a certain performance degradation is acceptable, the inference speedup can be maximized, reaching up to ∼x9.4 in LEDGAR.

MWT and Distillation. Additionally, we investigate the interaction between sequence compression and knowledge distillation in Table 3. To this end, we utilize a DistilBERT model with MWTs. For simplicity, we restrict our analysis to LEDGAR and to a single multi-word tokenizer $\mathcal{T}_{gen}^{2500}$ on different maximum sequence lengths. From the table, our MWT is shown to retain most of its performance with a quarter of the sequence length and an in-

ference speedup of ∼x8.8. Even with an extreme sequence truncation to only 64 tokens, we can still achieve a ∼x18.1 inference speedup with only a 2.7% drop in relative performance.

Model	Length	ΔF1	Speedup
$\mathcal{T}_{gen}$	512	82.12	1.00
Distil. + $\mathcal{T}_{gen}$	512	-0.78	2.43
Distil. + $\mathcal{T}_{gen}^{2500}$	128	-0.32	8.81
Distil. + $\mathcal{T}_{gen}^{2500}$	64	-2.70	18.13

Table 3: The macro-F1 and inference speedup results on LEDGAR with DistilBERT. MWTs are shown to be highly compatible with distilled models.

5 Conclusion

In this work, we proposed a sequence compression approach that reduces textual inputs by exploiting the use of multi-word expressions drawn from the training set according to their top-K frequencies. We conducted an investigation on 3 different datasets by evaluating each model in conjunction with other compression methods (Gee et al., 2022; Sanh et al., 2019). Our approach is shown to be highly robust to shorter sequence lengths, thus yielding a more than x4 reduction in computational cost with negligible drops in performance. In the future, we expect to extend our analysis to other language models and tasks such as language generation in the scope of sequence compression.

6 Limitations

As demonstrated in the paper, MWTs work well on text classification problems. Despite not having conducted experiments on generative tasks, there are no limitations in extending MWTs to them. Differently, the application of MWTs to token classification problems can be challenging. Specifically, when merging multiple words together, it is unclear how to label such fused tokens.

Acknowledgements

This work was supported by the IBRIDAI project, a project financed by the Regional Operational Program "FESR 2014-2020" of Emilia Romagna (Italy), resolution of the Regional Council n. 863/2021.

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.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359.

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*.

Leonidas Gee, Andrea Zugarini, Leonardo Rigutini, and Paolo Torroni. 2022. Fast vocabulary transfer for language model compression. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 409–416, Abu Dhabi, UAE. 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.

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.

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*.

Dipesh Kumar and Avijit Thawani. 2022. Bpe beyond word boundary: How not to use multi word expressions in neural machine translation. In *Proceedings of the Third Workshop on Insights from Negative Results in NLP*, pages 172–179.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-based & neural unsupervised machine translation. *arXiv preprint arXiv:1804.07755*.

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

Jesse Mu, Xiang Lisa Li, and Noah Goodman. 2023. Learning to compress prompts with gist tokens. *arXiv preprint arXiv:2304.08467*.

OpenAI. 2023. Gpt-4 technical report.

Naoki Otani, Satoru Ozaki, Xingyuan Zhao, Yucen Li, Micaelah St Johns, and Lori Levin. 2020. Pre-tokenization of multi-word expressions in cross-lingual word embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4451–4464, Online. Association for Computational Linguistics.

Aleksandar Petrov, Emanuele La Malfa, Philip HS Torr, and Adel Bibi. 2023. Language model tokenizers introduce unfairness between languages. *arXiv preprint arXiv:2305.15425*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

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.

Eva Sharma, Chen Li, and Lu Wang. 2019. BIGPATENT: A large-scale dataset for abstractive and coherent summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2204–2213, Florence, Italy. Association for Computational Linguistics.

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*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

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*.

A Further Details

A.1 Results

We tabulate the complete results for BERT and DistilBERT on ADE, LEDGAR, and PATENT in Tables 4 and 5 respectively. The values in each table are averaged across 5 seeds.

Model	Maximum Sequence Length			
	128	**64**	**32**	**16**
$\mathcal{T}_{gen}$	90.74 ± 0.84	91.22 ± 0.74	87.78 ± 0.74	76.04 ± 2.09
$\mathcal{T}_{gen}^{1000}$	90.66 ± 0.70	90.62 ± 0.41	88.62 ± 0.41	80.26 ± 0.91
$\mathcal{T}_{gen}^{2500}$	91.08 ± 0.54	90.76 ± 0.87	89.06 ± 0.87	80.76 ± 0.93
$\mathcal{T}_{gen}^{5000}$	91.00 ± 0.68	91.28 ± 0.62	89.28 ± 0.62	79.92 ± 1.42
$\mathcal{T}_{100}$	90.96 ± 0.67	90.82 ± 0.71	89.32 ± 0.71	82.82 ± 0.85
$\mathcal{T}_{100}^{1000}$	89.96 ± 1.21	90.38 ± 0.48	89.00 ± 0.48	85.18 ± 1.11
$\mathcal{T}_{100}^{2500}$	89.94 ± 0.72	90.56 ± 0.61	89.54 ± 0.61	85.78 ± 0.72
$\mathcal{T}_{100}^{5000}$	90.28 ± 0.65	90.38 ± 0.75	90.70 ± 0.75	84.94 ± 0.45

(a) ADE

Model	Maximum Sequence Length			
	512	**256**	**128**	**64**
$\mathcal{T}_{gen}$	82.12 ± 0.33	81.94 ± 0.36	81.46 ± 0.39	79.62 ± 0.56
$\mathcal{T}_{gen}^{1000}$	82.56 ± 0.24	82.52 ± 0.35	82.12 ± 0.40	80.54 ± 0.37
$\mathcal{T}_{gen}^{2500}$	82.16 ± 0.44	82.24 ± 0.40	81.92 ± 0.54	80.80 ± 0.57
$\mathcal{T}_{gen}^{5000}$	82.08 ± 0.41	82.02 ± 0.20	81.66 ± 0.19	80.70 ± 0.16
$\mathcal{T}_{100}$	82.12 ± 0.41	82.34 ± 0.21	81.68 ± 0.43	79.74 ± 0.66
$\mathcal{T}_{100}^{1000}$	82.38 ± 0.58	82.30 ± 0.68	81.80 ± 0.34	80.84 ± 0.23
$\mathcal{T}_{100}^{2500}$	81.96 ± 0.57	81.78 ± 0.60	82.06 ± 0.35	80.72 ± 0.57
$\mathcal{T}_{100}^{5000}$	82.14 ± 0.58	82.32 ± 0.35	81.92 ± 0.31	80.92 ± 0.71

(b) LEDGAR

Model	Maximum Sequence Length			
	256	**128**	**64**	**32**
$\mathcal{T}_{gen}$	61.44 ± 0.38	61.28 ± 0.37	60.46 ± 0.24	58.60 ± 0.60
$\mathcal{T}_{gen}^{1000}$	61.18 ± 0.54	61.28 ± 0.36	60.40 ± 0.45	59.46 ± 0.50
$\mathcal{T}_{gen}^{2500}$	61.40 ± 0.46	61.40 ± 0.69	61.22 ± 0.68	59.26 ± 0.42
$\mathcal{T}_{gen}^{5000}$	61.16 ± 0.69	61.08 ± 0.49	60.40 ± 0.71	59.14 ± 0.44
$\mathcal{T}_{100}$	60.66 ± 0.39	60.62 ± 1.04	59.52 ± 0.63	58.44 ± 0.63
$\mathcal{T}_{100}^{1000}$	60.96 ± 0.62	60.16 ± 0.68	59.48 ± 0.25	58.76 ± 0.63
$\mathcal{T}_{100}^{2500}$	60.80 ± 0.42	60.36 ± 1.02	59.98 ± 1.15	58.78 ± 0.58
$\mathcal{T}_{100}^{5000}$	60.42 ± 0.44	59.80 ± 0.73	59.54 ± 0.46	58.24 ± 1.76

(c) PATENT

Table 4: Model performance of BERT averaged across 5 seeds.

Model	Maximum Sequence Length			
	128	**64**	**32**	**16**
Distil. + $\mathcal{T}_{gen}$	90.66 ± 0.69	91.66 ± 0.43	87.56 ± 1.64	74.78 ± 1.50
Distil. + $\mathcal{T}_{gen}^{1000}$	90.18 ± 0.89	90.44 ± 0.73	88.16 ± 0.81	78.74 ± 0.88
Distil. + $\mathcal{T}_{gen}^{2500}$	91.08 ± 0.28	90.64 ± 0.53	88.30 ± 0.96	79.24 ± 1.37
Distil. + $\mathcal{T}_{gen}^{5000}$	89.60 ± 0.92	90.22 ± 1.11	88.06 ± 0.79	79.52 ± 1.16
Distil. + $\mathcal{T}_{100}$	90.52 ± 0.48	89.76 ± 0.84	88.54 ± 1.01	81.16 ± 0.91
Distil. + $\mathcal{T}_{100}^{1000}$	88.26 ± 0.86	89.10 ± 0.44	88.52 ± 0.68	82.84 ± 0.35
Distil. + $\mathcal{T}_{100}^{2500}$	88.58 ± 1.20	89.10 ± 1.18	89.32 ± 1.01	83.38 ± 0.62
Distil. + $\mathcal{T}_{100}^{5000}$	87.68 ± 0.92	87.94 ± 1.22	87.88 ± 0.55	82.84 ± 0.77

(a) ADE

Model	Maximum Sequence Length			
	512	**256**	**128**	**64**
Distil. + $\mathcal{T}_{gen}$	81.48 ± 0.52	81.12 ± 0.50	81.18 ± 0.31	79.22 ± 0.29
Distil. + $\mathcal{T}_{gen}^{1000}$	82.02 ± 0.83	82.30 ± 0.31	81.56 ± 0.44	80.20 ± 0.41
Distil. + $\mathcal{T}_{gen}^{2500}$	81.74 ± 0.23	81.36 ± 0.25	81.86 ± 0.18	79.90 ± 1.01
Distil. + $\mathcal{T}_{gen}^{5000}$	81.38 ± 0.52	81.62 ± 0.29	81.60 ± 0.29	80.34 ± 0.28
Distil. + $\mathcal{T}_{100}$	81.42 ± 0.70	81.60 ± 0.12	81.50 ± 0.48	80.02 ± 0.54
Distil. + $\mathcal{T}_{100}^{1000}$	81.42 ± 0.59	80.90 ± 0.68	81.98 ± 0.18	80.62 ± 0.47
Distil. + $\mathcal{T}_{100}^{2500}$	81.80 ± 0.17	81.36 ± 0.30	82.06 ± 0.27	80.46 ± 0.38
Distil. + $\mathcal{T}_{100}^{5000}$	81.58 ± 0.57	81.34 ± 0.42	81.92 ± 0.18	80.82 ± 0.43

(b) LEDGAR

Model	Maximum Sequence Length			
	256	**128**	**64**	**32**
Distil. + $\mathcal{T}_{gen}$	60.88 ± 0.61	60.98 ± 0.67	59.88 ± 0.57	57.72 ± 0.71
Distil. + $\mathcal{T}_{gen}^{1000}$	60.58 ± 0.31	59.92 ± 0.63	59.94 ± 0.94	58.36 ± 0.62
Distil. + $\mathcal{T}_{gen}^{2500}$	59.96 ± 0.75	59.94 ± 0.43	59.90 ± 0.65	58.16 ± 0.61
Distil. + $\mathcal{T}_{gen}^{5000}$	59.86 ± 0.61	60.10 ± 0.88	59.26 ± 0.53	58.46 ± 0.52
Distil. + $\mathcal{T}_{100}$	59.58 ± 0.77	59.22 ± 0.59	58.10 ± 0.70	57.22 ± 0.59
Distil. + $\mathcal{T}_{100}^{1000}$	59.52 ± 0.49	59.88 ± 0.54	58.72 ± 0.47	57.42 ± 0.72
Distil. + $\mathcal{T}_{100}^{2500}$	59.04 ± 0.32	58.82 ± 0.95	57.58 ± 0.53	56.76 ± 0.47
Distil. + $\mathcal{T}_{100}^{5000}$	59.82 ± 0.57	58.74 ± 0.40	58.76 ± 0.59	57.30 ± 1.01

(c) PATENT

Table 5: Model performance of DistilBERT averaged across 5 seeds.