PART-OF-SPEECH MODELS COMPRESSION METHODS FOR ON-DEVICE GRAPHEME-TO-PHONEME CONVERSION

Marek Kubis[1], *Maxime Méloux*[2], *Paweł Skórzewski*[1], *Marcin Lewandowski*[2], *Gunu Jho*[3], *Hyoungmin Park*[3]

[1]Adam Mickiewicz University, Poznań, Poland
[2]Samsung R&D Institute Poland, Warsaw, Poland
[3]Mobile Communications Business, Samsung Electronics, Republic of Korea

ABSTRACT

The paper investigates methods of compressing part-of-speech models that are developed for an on-device grapheme-to-phoneme conversion module. The performance of part-of-speech models is analyzed under different compression regimes. The evaluation is done with respect to French, German and Italian datasets that consist of TTS input prompts. The study shows that a proper selection of a compression method reduces the model size significantly without deteriorating the grapheme-to-phoneme conversion performance.

***Index Terms*—** part-of-speech tagging, model compression, grapheme-to-phoneme conversion

1. INTRODUCTION

Although end-to-end text-to-speech synthesis has gained considerable attention in recent years, the possibility to process the input and output of a grapheme-to-phoneme conversion is still invaluable in the industrial setting, especially in the case of heavily constrained, embedded systems. Augmentation of graphemes with corresponding part-of-speech (POS) tags to improve the accuracy of grapheme-to-phoneme conversion models is an example of such a pre-processing procedure. However, state-of-the-art POS tagging models require a considerable amount of memory that is not available in the case of on-device speech synthesis systems. This paper investigates methods of compressing part-of-speech models for on-device grapheme-to-phoneme conversion. We implement a set of POS tagging models based on Transformer architecture [1, 2] and study their performance under different compression regimes. We show that a proper combination of compression methods can reduce the model size significantly without deteriorating the grapheme-to-phoneme conversion performance.

2. RELATED WORK

The compression of machine learning models is the subject of many publications. Due to limited space, we discuss below only the most significant methods in this context.

Knowledge distillation is a machine learning method in which a smaller model (called the student model) is trained to imitate the behavior of a larger model (called the teacher model) [3]. This way, the student model can learn a more efficient representation of the teacher model. DistilBERT [4] – a distilled version of BERT [2] – is an example of a natural language model created using knowledge distillation. It shares its general architecture with BERT, but it has half as many network layers. As a result, DistilBERT has been reported to be about 40% smaller and 70% faster than BERT, without significant performance loss on downstream tasks like sentiment classification or question answering.

Deep networks can be trained using a restricted fixed-point number representation with almost no degradation in the classification accuracy [5]. Quantization encompasses a variety of techniques for performing computations at lower bandwidths than floating point precision, such as quantization-aware training (QAT), static quantization and dynamic quantization. These methods can be used to efficiently compress even large models such as BERT [6]. Quantization reduces model size at the expense of its accuracy after training by decreasing the precision of the numeric representation of the model's weights and/or activations. Deep compositional code learning (DCCL) [7] is a compression method that relies on multi-codebook quantization. Each word is represented by a code composed of multiple discrete numbers. Then, the embedding vectors represent the codes rather than the words. The codes are learned so that similar words have similar codes. Derivative methods include AdaComp (adaptive compression) [8], where different code lengths are adaptively assigned by training on downstream tasks to improve the downstream task accuracy of a compressed model.

Pruning is a technique used to reduce models' size. It can therefore increase performance and decrease model size on other NLP tasks [9, 10]. It has been successfully applied to BERT [11]. Neural network pruning consists in removing some less important parts of the network (e.g., weights, neurons, layers) to reduce its size. Various importance criteria can help determine which parts of the network can be removed without sacrificing too much accuracy.

Dimensionality reduction is a way of representing multi-dimensional data in a lower-dimensional space. It has many applications, including the compression of word embeddings for NLP tasks [12].

3. MODELS

3.1. Part-of-speech tagging

For POS tagging, we use the Transformer model [1] – a neural architecture that manifests reduced training costs compared to traditional RNN-based sequence-to-sequence models, by forgoing recurrent layers in favor of pure attention. Since Transformer achieves state-of-the-art results in sequence transduction tasks, we refrained from implementing other models.

In order to maintain comparable model sizes across different languages before compression, we use a common, pre-trained, multilingual model as the base for all POS taggers. Our POS tagging models trained for French, Italian and German consist of the BERT multilingual base model (cased) [2] followed by a single, fully-connected neural network layer set on top of the hidden-states output to form the token classification head. This architecture is simple and effective for sequential tagging tasks [13]. Since the focus of this study is compression performance, we closely follow the training procedure to estimate POS tagging models and refrain from further boosting the tagging performance by manipulating hyper-parameters.

3.2. Compression

For experiments, we selected diverse methods addressing both the compression of a language model and the compression of parameter matrices of the final model: knowledge distillation, dynamic quantization, and pruning. We did not implement dimensionality reduction methods because, in the case of a pre-trained Transformer-based model, these techniques have to be applied layer by layer, which is hard to perform reasonably considering inter-layer connections.

For knowledge distillation, we replace the multilingual BERT model with its corresponding DistilBERT version in the experiments. DistilBERT's authors showed that DistilBERT retains decent performance in a classification task (IMDb sentiment classification [14] – 0.6% worse while being 40% smaller) and a question answering task (SQuAD v1.1 [15] – 3.9 points worse) [4]. We expect that BERT's task-agnostic generalization capabilities are successfully captured into a smaller model, and that the subsequent training can mitigate the potential loss of POS tagging accuracy.

For Transformer-like models, loading weights from memory is the main factor affecting model execution time, rather than the computation of matrix multiplications. Therefore, we quantize the weights of all linear and normalization layers in the model to 8-bit integers; then the activations are dynamically quantized during inference.

We apply the pruning by zeroing tensor weights and biases of linear layers, in a way similar to that described in [9], but instead of the value of the weights, we use L1-norm as the importance criterion. We treat the percentage of weights to be pruned (denoted by k) as a parameter. After training, we replace a given percentage (k) of all tensor weights with zeros. The values zeroed out are the ones with the lowest L1-norm. We convert the resulting model to sparse matrix representation to efficiently take advantage of the pruning and reduce the memory and disk footprint. In our implementation, a non-sparse tensor of dimensions $(m_1, m_2, \ldots, m_n)$ has a fixed memory size of $S \prod_i m_i$ bits, where $S = 32$ in general and $S = 8$ for 8-bit quantized tensors. Sparse tensors are stored in a coordinate list (COO) format and have a memory size of $(8n + S)(1 - k)$. For both sparse and non-sparse tensors, there is an additional overhead from storing other tensor data that we consider negligible. Tensors are therefore only effectively made smaller in memory if $k > 1 - 4m/5$ (non-quantized) or $k > 1 - m/2$ (8-bit quantized) for biases, and for $k > 1 - 2mn/3$ (non-quantized) or $k > 1 - mn/3$ (quantized) for weights. Since we apply the sparse conversion to the model globally, this typically only results in effective compression for a high enough value of k, measured empirically to be $k_{\text{min}} \approx 1/3$.

We test four different types of pruning: (1) without retraining, (2) retraining and repruning once after the initial pruning, (3) repeated pruning and retraining until the model converges, and (4) iterative pruning, where k is progressively increased and the model repeatedly retrained and pruned after each increase until convergence is achieved.

The motivation for these different retraining methods is to give the model progressively stronger ways to adapt to its pruned configuration and to learn how not to use the pruned weights. For retraining and pruning, we retrain the model 10 times for 10 epochs each, with a smaller learning rate of 10^{-5}. For iterative pruning, we first train the model without pruning ($k = 0$). We then increase k by 0.05 (or 0.10 for a target value $k_f > 0.5$), prune the model and retrain it. This last step is repeated until the desired pruning rate is achieved.

4. EXPERIMENTS

A reliable comparison of the quality of different model compression methods is not easy. One not only has to take into account the actual compression ratio, but also its impact on the model's performance in the downstream tasks.

It should also be noted that the compression ratio notation varies between different authors. Some authors use the convention of reporting the ratio of the compressed model size to the original model size (i.e., the lower, the better), while others report the fraction (or percentage) of model size reduction (the higher, the better). We use the disk compression rate metric defined as $R_d = 1 - \frac{C_d}{U_d}$, where C_d is the disk size of a compressed model and U_d is the disk size of the

corresponding uncompressed model. By analogy, we define the in-memory compression rate metric as $R_m = 1 - \frac{C_m}{U_m}$, where C_m is the in-memory size of a compressed model and U_m is the in-memory size of the corresponding uncompressed model. For metrics defined this way, their higher value (closer to 1) means a better compression (both for R_d and R_m).

For evaluation, we perform two types of experiments. First, we analyze the impact of compressing the models on POS tagging performance. Second, we perform end-to-end evaluation of POS tagging models inside a Text-To-Speech system to verify if the compressed models have a lower grapheme-to-phoneme conversion accuracy.

4.1. POS tagging performance

We measure the difference between micro-average F_1-scores of the uncompressed and compressed models that share the same architecture (ΔF_1-score in Table 2). The performance is measured on French, German and Italian datasets that consist of exemplary TTS input prompts annotated with POS tags. The tagsets consist of tags that represent coarse designations of word classes, such as noun or verb, in conjunction with fine-grained designations of morphological features, such as gender, number and case that are indispensable in grapheme-to-phoneme conversion. We use 80% of the available data for training, reserving the rest for development and test sets as shown in Table 1.

Table 1. Numbers of sentences and POS tags in the corpus

Language	# Train	# Dev	# Test	# POS Tags
French	6702	1675	2094	90
German	2617	655	818	99
Italian	7261	1817	2272	73

Our results show that knowledge distillation and dynamic quantization generally cause a drop of less than one percent in F_1-score, even when used jointly. Furthermore, their combined usage allows us to reduce the models' in-memory footprint by up to 90% and their on-disk size by up to 40%. One notable exception is the case of French, for which each of these methods performs equally well or better than for German and Italian when used individually, but combining them results in a 7.5% drop in F_1-score. We also show that pruning with no retraining leads to a sharper decrease of 7 to 12% in F_1-score even with a low value of $k = 0.1$, but this decrease is almost entirely compensated by retraining once, and that subsequent repeated or iterative retraining further mitigates this drop. A higher value of $k = 0.5$ (where memory savings become significant) causes a noticeable drop in F_1-score of 5 to 9 percentage points respectively for French and Italian even with iterative retraining. Our German models suffer from a higher drop of 27 percentage points. At $k = 0.9$, our models still retain between 50 and 70 percent of their original F_1-score.

4.2. End-to-end evaluation in TTS frontend

Although the F_1-scores of the tested models do not deteriorate significantly except in the case of extreme pruning, one must consider that minor changes in part-of-speech annotations can have a severe impact on the performance of grapheme-to-phoneme conversion. Therefore, in our second set of experiments, we measure the change in the accuracy of the grapheme-to-phoneme conversion before and after replacing the uncompressed POS tagging model with its compressed counterpart in a TTS system.

Our text-to-speech pipeline is composed of frontend and backend parts. Following a classical TTS design pattern, the frontend is responsible for mapping a sequence of letters to a sequence of phonetic labels that describe an utterance in terms of phonemes, word-stress location, pauses, phrases etc. The backend transforms such a sequence into an acoustic speech signal itself. Therefore, we can isolate the frontend and measure the grapheme-to-phoneme conversion accuracy directly.

The frontend itself encompasses a few distinct processes – a rule-based text normalization; a word-level grapheme-to-phoneme function predicting pronunciation with classification and a regression tree supplemented with manually curated lexica of pronunciation exceptions; a rule-based model of supra-word-level phonetic phenomena. The quality of part-of-speech tagging can influence the quality of grapheme-to-phoneme conversion in multiple ways. Lexicons in general may contain multiple POS-dependent pronunciations of a word. Normalization rules can also rely on POS information. German provides an important example – numbers originally represented with digits can be expanded with different word endings depending on the grammatical gender or case of a neighboring word. In French, on the other hand, part of speech plays a crucial role in rules governing alterations in word pronunciation in different contexts. Most notably, the last consonant of some words may or may not be pronounced (the so-called liaison) depending on, among others, the syntactic role of the following word.

For end-to-end evaluation, we use test sets that consist of utterances paired with verified "gold standard" transcriptions. There are 26251 sentences in the German dataset, 29194 in the French one and 17994 in the Italian one. The test sets were drawn from internal corpora that were originally created as a set of prompts for recording unit selection databases. Reference transcriptions were generated automatically, then checked and corrected by language experts. There is exactly one transcription marked as correct for each sentence. An utterance is assumed to be phonetized correctly if the whole sequence produced by the frontend is identical to the reference. Therefore, if there is more than one proper pronunciation of a given sentence and the frontend with the compressed POS tagging model selects different pronunciation than the frontend with the uncompressed one, the utterance is not accepted, making the evaluation even more challenging and focused on

Table 2. Evaluation results

Language	Compression method	R_d	R_m	ΔF_1-score	ΔG2PAcc. (pp)
French	Distillation	0.240	0.328	$-$**0.001**	$-$0.45
	Dynamic quantization	0.359	**0.907**	$-$0.009	$-$**0.17**
	Pruning ($k = 0.1$, no retraining)	~ 0	~ 0	$-$0.072	$-$2.56
	Pruning ($k = 0.1$, retraining once)	~ 0	~ 0	$-$0.011	$-$0.44
	Pruning ($k = 0.1$, repeated retraining)	~ 0	~ 0	$-$0.009	$-$0.38
	Pruning ($k = 0.1$, iterative retraining)	~ 0	~ 0	$-$0.009	$-$0.34
	Pruning ($k = 0.5$, iterative retraining)	~ 0	0.250	$-$0.088	$-$1.56
	Pruning ($k = 0.9$, iterative retraining)	0.240	0.850	$-$0.296	$-$4.52
	Dynamic quantization, distillation	**0.420**	**0.907**	$-$0.075	$-$0.58
	Dyn. quantization, distillation, pruning ($k = 0.1$, no retraining)	0.419	0.749	$-$0.163	$-$1.78
	Dyn. quantization, distillation, pruning ($k = 0.5$, iter. retraining)	0.419	0.861	$-$0.091	$-$1.23
German	Distillation	0.240	0.328	$-$0.030	$+$0.68
	Dynamic quantization	0.359	**0.907**	$-$0.020	$-$0.54
	Pruning ($k = 0.1$, no retraining)	~ 0	~ 0	$-$0.117	$+$0.38
	Pruning ($k = 0.1$, retraining once)	~ 0	~ 0	$-$0.036	$+$**1.88**
	Pruning ($k = 0.1$, repeated retraining)	~ 0	~ 0	$-$0.027	$+$**1.88**
	Pruning ($k = 0.1$, iterative retraining)	~ 0	~ 0	$-$0.033	$+$1.66
	Pruning ($k = 0.5$, iterative retraining)	~ 0	0.250	$-$0.267	$+$0.85
	Pruning ($k = 0.9$, iterative retraining)	0.240	0.850	$-$0.463	$+$0.30
	Dynamic quantization, distillation	**0.420**	**0.907**	$-$**0.007**	$+$0.76
	Dyn. quantization, distillation, pruning ($k = 0.1$, no retraining)	**0.420**	0.749	$-$0.066	$-$0.20
	Dyn. quantization, distillation, pruning ($k = 0.5$, iter. retraining)	**0.420**	0.861	$-$0.256	$+$1.19
Italian	Distillation	0.240	0.328	$-$**0.001**	$+$0.14
	Dynamic quantization	0.359	**0.907**	$-$0.004	$+$0.08
	Pruning ($k = 0.1$, no retraining)	~ 0	~ 0	$-$0.120	$-$7.23
	Pruning ($k = 0.1$, retraining once)	~ 0	~ 0	$-$0.006	$-$0.16
	Pruning ($k = 0.1$, repeated retraining)	~ 0	~ 0	$-$0.003	$-$0.06
	Pruning ($k = 0.1$, iterative retraining)	~ 0	~ 0	$-$0.002	$+$0.12
	Pruning ($k = 0.5$, iterative retraining)	~ 0	0.250	$-$0.054	$-$0.39
	Pruning ($k = 0.9$, iterative retraining)	0.240	0.850	$-$0.242	$-$0.24
	Dynamic quantization, distillation	**0.420**	**0.907**	$-$0.011	$+$**0.31**
	Dyn. quantization, distillation, pruning ($k = 0.1$, no retraining)	**0.420**	0.749	$-$0.082	$-$0.33
	Dyn. quantization, distillation, pruning ($k = 0.5$, iter. retraining)	**0.420**	0.861	$-$0.055	$-$0.50

full reproducibility. The evaluation results are reported in Table 2 (ΔG2PAcc. column). One can observe that the decrease in the grapheme-to-phoneme conversion accuracy of the TTS frontend that encompasses the uncompressed POS tagging model is less than 1 percentage point for most models except for the pruned ones. Applying pruned models does affect G2P performance for German and French but not for Italian. The Italian frontend mostly uses POS labels for looking up pronunciation in lexicons and for a few transformations accounting for inter-word phenomena like clitics. German and French frontends use POS information far more extensively via hand-crafted general textual and phonetic transformations. We hypothesize that this makes them more sensitive to the mistakes of POS tagger models.

5. CONCLUSION

We analyzed the performance of compression methods applied to Transformer-based part-of-speech tagging models used in the grapheme-to-phoneme conversion task. Among the investigated methods, a combination of dynamic quantization and knowledge distillation achieved the best results for German and Italian. In the case of French, we achieved the best results by using dynamic quantization solely. These compression schemes lead to POS models that are significantly smaller than their uncompressed counterparts while maintaining almost the same POS tagging performance. Furthermore, the accuracy of the grapheme-to-phoneme conversion procedure that utilizes the quantized and distilled models did not change significantly.

6. ACKNOWLEDGEMENTS

This research was partially funded by the "Research in the field of Natural Language Processing 2021" project, a cooperation between Adam Mickiewicz University and Samsung Electronics.

7. REFERENCES

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. 2017, vol. 30, Curran Associates, Inc.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota, June 2019, pp. 4171–4186, Association for Computational Linguistics.

[3] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, "Distilling the knowledge in a neural network," 2015.

[4] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," 2020.

[5] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan, "Deep learning with limited numerical precision," 2015.

[6] Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat, "Q8BERT: quantized 8bit BERT," *CoRR*, vol. abs/1910.06188, 2019.

[7] Raphael Shu and Hideki Nakayama, "Compressing word embeddings via deep compositional code learning," 2017.

[8] Yeachan Kim, Kang-Min Kim, and SangKeun Lee, "Adaptive compression of word embeddings," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, July 2020, pp. 3950–3959, Association for Computational Linguistics.

[9] Shijian Tang and Jiang Han, "A pruning based method to learn both weights and connections for lstm," in *Proceedings of the Advances in Neural Information Processing Systems, NIPS*, Montreal, QC, Canada, 2015, pp. 7–12.

[10] Abigail See, Minh-Thang Luong, and Christopher D. Manning, *Compression of Neural Machine Translation Models via Pruning*, Abigail See, 2016.

[11] Mitchell A. Gordon, Kevin Duh, and Nicholas Andrews, "Compressing BERT: Studying the effects of weight pruning on transfer learning," 2020.

[12] Vikas Raunak, Vivek Gupta, and Florian Metze, "Effective dimensionality reduction for word embeddings," in *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, 2019, pp. 235–243.

[13] 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*, Online, Oct. 2020, pp. 38–45, Association for Computational Linguistics.

[14] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, 2011, pp. 142–150.

[15] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250*, 2016.