Arabic Compact Language Modelling for Resource Limited Devices

Zaid Alyafeai
KFUPM, Dhahran, Saudi Arabia
`g201080740@kfupm.edu.sa`

Irfan Ahmad
KFUPM, Dhahran, Saudi Arabia
`irfan.ahmad@kfupm.edu.sa`

Abstract

Natural language modelling has gained a lot of interest recently. The current state-of-the-art results are achieved by first training a very large language model and then fine-tuning it on multiple tasks. However, there is little work on smaller more compact language models for resource-limited devices or applications. Not to mention, how to efficiently train such models for a low-resource language like Arabic. In this paper, we investigate how such models can be trained in a compact way for Arabic. We also show how distillation and quantization can be applied to create even smaller models. Our experiments show that our largest model which is 2x smaller than the baseline can achieve better results on multiple tasks with 2x less data for pretraining.

1 Introduction

Language Modelling refers to the process of designing a model that can mimic language understanding in humans. The trained models can then be fine-tuned on multiple tasks by using them as feature extractors by adding extra layers at thend of such models. This is a difficult task since, language understanding and comprehension require huge amounts of data. Recently, advances in machine learning and particularly deep learning is proving significant in different tasks like text classification, sentiment analysis, question answering, summarization and language inference using transformer-based architectures (Devlin et al., 2018). This approach requires training a deep architecture that models the language using unsupervised techniques on unlabelled data. The outcome of this method is a large model that has a huge memory footprint. This limits the deployment of such models on resource-limited devices because of memory constraints. Furthermore, the model needs to be retrained to make it useful for other more interesting tasks. Recently, a lot of approaches have emerged in making these deep architectures more practical using methods like distillation (Sun et al., 2020) and quantization (Zafrir et al., 2019). Unfortunately, most of the work that is available in the literature targets English language because of the availability of data and research development tools. In this work we train a compact Arabic language model that can be fine-tuned on different tasks. Moreover, we make the model efficient for embedded devices with relatively a small loss in accuracy.

The paper is organized as follows: In section 2, we revise the literature for related work. In section 3, we describe our methodology where we discuss the pretraining dataset, the tasks used for evaluation and our training procedure for the teacher and student models. Finally, we summarize the experiments and discuss the results.

2 Related Work

Deep architectures like GPT (Radford et al., 2018), BERT (Devlin et al., 2018) and T5 (Raffel et al., 2019) require a huge amount of data and computation. Moreover, such language models are very large and cannot be deployed on embedded devices due to memory constraints. Recently, many approaches have been applied in order to make such models smaller and more practical. Mainly, the literature focuses on two main approaches which are quantization and distillation.

Quantization is defined as reducing the memory footprint by mapping parameters to low dimensional spaces. For instance, we can map `float32` to `int16`. This is usually done at the deployment phase where the model is already trained. Many studies have been published that adopt this approach. Zafrir et al. applied this approach to BERT in order to create a quantized 8 bit model (Q8bert) (Zafrir et al., 2019). More recently, a more advanced approach called Hessian-based ultra low precision quantization has been applied to BERT (Q-BERT) (Shen et al., 2020). In another work, the authors designed an architecture called GOBO that targets quantizing attention-based models for language understanding (Zadeh and Moshovos, 2020).

On the other hand, another approach is called distillation which mimics the task of a teacher and student for training smaller language models (Alkhulaifi et al., 2020). These methods have been developed in computer vision but they were recently extended to language modelling. The teacher model, usually a large model, is a pretrained model that can be used to distil the knowledge to a smaller model called the student model. The student model is a much smaller architecture. The distillation process has been applied in many fields in machine learning and takes many forms like teacher-student optimization (Yang et al., 2019), teacher assistant (Mirzadeh et al., 2019), contrastive representation distillation (Tian et al., 2019) and patient knowledge distillation (Sun et al., 2019). Other approaches make the process task-agnostic like the work by Sun et al. in their architecture called mobileBERT (Sun et al., 2020).

From the research perspective, Arabic language modelling is under-represented in the literature let alone implementing compression techniques. Noticeably, there were a few main attempts to adapt deep models for Arabic like Hulmuna (ElJundi et al., 2019) which retrains ELMo (Peters et al., 2018) for Arabic and fine-tunes it on different tasks. Moreover, we have AraBERT which retrains BERT for Arabic (Antoun et al., 2020). More recently, other larger variants have been introduced namely ARBERT and MARBERT (Abdul-Mageed et al., 2020). These models are comparatively very large and are trained with a huge amount of data usually larger than 20 GB of text and the models are typically more than 100 million parameters in size. Arabic poses its own challenges when considered for natural language processing tasks. Arabic is a morphologically rich language where words can take different representations by connecting different prefixes, infixes and suffixes. Hence words need better representations in order to tackle the out-of-vocabulary problem (Gerz et al., 2018) by applying different preprocessing approaches. For instance, segmentation slightly achieves a better performance compared to direct tokenization (Antoun et al., 2020). Not to mention applying preprocessing to deal the linguistic parts like diacritics, normalization, cleaning, mixed foreign characters, etc.

In this work we consider applying distillation and quantization on Arabic language models for resource-limited devices. Up to our knowledge, there is no previous work that applies compression directly on Arabic. There are some related work that applies distillation on a cross-lingual level (Cui et al., 2017) but it is evident from the literature that monolingual models are more powerful (Antoun et al., 2020), (Martin et al., 2019).

3 Experimentation

In this section we explain in details our approach that we take for training compact language models for Arabic. First we train a large teacher model in an unsupervised fashion on a large corpus. Then we distil the knowledge from the teacher model to the student model and evaluate it by fine-tuning on different tasks. Finally, we use quantization as a final step for reducing the memory footprint of our distilled student models. For the comparison we use the first version of AraBERT as the baseline for evaluating our models.

3.1 Pretraining Dateset

We collected a wide variety of datasets in order to cover as many tasks as possible. We used four different datasets for pretraining. Mainly, we collected 10 GB from the 1.5 billion words corpus (El-Khair, 2016), 1.5 GB from wikipedia, 1 GB books from Alshamila (shamela, 2005) and around 500 MB from Twitter. We clean all the datasets by removing diacritics and applying normalization (mapping different forms of a character to a single character). We then converted the dataset to `tfrecords` for faster training. In Table 1, we present the total number of tokens for each dataset. We use the WordPiece algorithm (Devlin et al., 2018) for tokenization with a fixed sequence length of 512. We fixed the max sequence length to 512 with max predictions per sequence as 20. The masked tokens probability is set to 15 % with a duplication factor of size 5.

3.2 Tasks

We evaluate the models on a variety of datasets and tasks from the literature. Some datasets are not split by the authors so we split them randomly into training and testing sets. In all the experiments we

Table 1: Number of tokens in each dataset. B:billions and M:millions.

Dataset	Number of Tokens
Billion Words Corpus	1.3 B
Wikipedia	148 M
Alshamila	112 M
Twitter	68 M

use the same split to test the models.

1. **Hard**: Named the Arabic Reviews Dataset (Elnagar et al., 2018), The dataset consists of 93,700 reviews in total where we have 46,850 reviews for each positive and negative sentiments.

2. **ASTD**: called the Arabic Sentiment Tweets Dataset which contains 10,000 tweets written in both MSA and Egyptian dialect (Nabil et al., 2015). We used the balanced dataset on binary sentiment classification.

3. **ArSenTD-Lev**: The Arabic Sentiment Twitter Dataset for Levantine dialect (ArSenTD-LEV) (Baly et al., 2018) contains 4,000 tweets written in Arabic and equally collected from Lebanon, Jordan, Syria. and Palestine.

4. **LABR**: The Large-scale Arabic Book Reviews dataset (Aly and Atiya, 2013). It contains over 63,000 book reviews that were collected from the website Goodreads during the month of March 2013. Each book review comes with the rating (1 to 5) and the text of the review. The ratings 1 and 2 are considered negative while 4 and 5 are considered positive.

5. **AJGT**: The Arabic Jordanian General Tweets dataset (Alomari et al., 2017) consists of 1,800 tweets written in Jordanian dialect and modern standard Arabic (MSA). The tweets were manually annotated as either positive or negative to be used for sentiment analysis tasks.

6. **TEAD**: A large-scale dataset of tweets gathered in 2017. The dataset was annotated using emojis and sentiment lexicons. We only use a small subset of the dataset that contains 2000 positive and 2000 negative labels.

7. **ARCD**: The Arabic Reading Comprehension Dataset (ARCD) is composed of 1,395 questions annotated by crowd workers on Wikipedia articles (Mozannar et al., 2019). The authors also published a translated version of the English SQuAD called the Arabic-SQuAD. The question answering tasks are usually reported using F1 score and exact match (EM) score in literature.

8. **PADT** treebank is based on the Prague Arabic Dependency Treebank (PADT) (Zeman, 2015). We use this dataset for the part-of-speech tagging (PoS) tasks. The treebank is composed of 7,664 sentences (282,384 tokens) and its domain is mainly news. The results are reported using precision, recall and F1.

3.3 Pretraining Models

We train the teacher and student models on TPUv3-8 which distributes the batch size over 8 nodes. Furthermore, TensorFlow 1.15 is used as the main library for training all the models. For the pretraining we use Google Cloud and for fine-tuning we use Google Colab notebooks.

3.3.1 Teacher Model

We follow the same approach as in (Sun et al., 2020) where the teacher model has the same depth as BERT$_{\text{LARGE}}$, i.e., 24 blocks each one containing a multi-head attention and feed-forward network with a normalization layer after each one. However, in order to reduce the number of parameters, a linear transformation is applied to modify the input and the output size to 512 (i.e., bottleneck). The linear transformations are applied after the embeddings and after the end of the fully connected layers of each block. The output size of each layer in each block is named inner block-neck size and it was set to 1024 in the original architecture. However, we train with smaller inner block-neck of size 256 instead of 1024 in order to reduce the model size. Also we train different models with vocabulary sizes of 30K and 50K with hidden sizes 512 and 256, respectively. The number of layers is 24 and the number of attention heads is 4. We use the lamb optimizer (You et al., 2019) with a learning rate 0.0015. We train the model for around 2 million steps with a batch size of 256.

3.3.2 Student Model

For the student model we train two types of models: the first one has 4 feed-forward layers and the smaller model has 1 per each block. The number of layers and attention heads is the same as the teacher model. The model is first trained using knowledge transfer as a combination of attention transfer and feature map transfer. Attention transfer $\mathcal{L}_{AT}$ maps the attention weights from the teacher model to the student model. While the feature map transfer $\mathcal{L}_{FMT}$ maps the weights from each block in the teacher to each block in the student. We train each model for 500K steps for attention transfer and 2 million steps for pretraining distillation. The pretraining distillation objective is defined as combination of BERT training objectives and knowledge distillation objectives

$$\mathcal{L}_{PD} = \alpha \mathcal{L}_{MLM} + (1-\alpha)\mathcal{L}_{KD} + \mathcal{L}_{NSP}$$

Masked language model (**MLM**) is the objective for predicting the masked word in the corpus. Next sentence prediction (**NSP**) optimizes the probability of predicting next sentence. Knowledge distillation (**KD**) is a mixture of training on current corpus and teacher labels. Similar to the teacher model, We use the lamb optimizer, a learning rate 0.0015 and batch size 256. In Table 2, we summarize the hyper-parameters for training during the distillation process. The beta and gamma distillation factors measure the difference between the mean and the variances between the teacher and student models. The hidden distillation factor is used for weighted layer wise distillation and the distillation ground truth parameter is used for the mixed label training.

Table 2: Hyper-parameters for training the student models.

Parameter	**Value**
Beta distillation factor	500
Gamma distillation factor	5
Hidden distillation factor	100
Distillation ground truth	0.5

4 Evaluation

We use fine-tuning on down-stream tasks as the main approach for evaluating our models. In all of our experiments, we follow the same procedure for the teacher and the student models. Note that in order to make the comparison fair to AraBERT, we don't do any preprocessing to our datasets. This is in order to avoid any bias towards our models. This might cause our models to preform worse than expected because some characters in the datasets might cause unknown tokens to show up. Mainly, we train for 5 epochs and we use a learning rate of 3×10^{-5} for the text classification and question answering tasks while we use a learning rate of 2×10^{-3} for the PoS tagging task. In Figure 1, we compare inverted-block BERT for Arabic (IBERTA) for two different vocabulary sizes. To make the comparison fair, we decrease the hidden size to 256 for the model with vocabulary size 50K. We see as the number of iterations increases, the accuracy increases for both models. However, the model with vocabulary size 50K achieve better results across the different iterations compared to the vocabulary size 30K. This is due to the fact that Arabic is morphologically rich language with a huge and sparse vocabulary. We use the model with the vocabulary size 50K as the teacher model and we refer to it as (IBERTA).

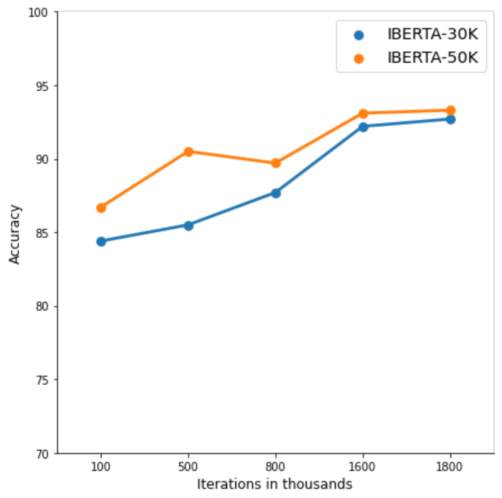

Figure 1: Comparison between the teacher models when fine-tuned in the AJGT task.

In Table 3, we compare IBERTA to AraBERT for different datasets and tasks. We can notice that our teacher model achieves better results on most of the text classification tasks. In tasks that require more token information like question answering

Table 3: Comparing IBERTA with 50K vocabulary size, MBERTA, MBERTA II and AraBERT for multiple tasks.

Dataset	**Metric**	**IBERTA**	**AraBERT**	**MBERTA**	**MBERTA II**
AJGT	Accuracy	**93.3**	89.7	90.8	89.1
HARD	Accuracy	**95.7**	95.4	95.5	95.4
ArSenTD	Accuracy	**55.8**	55.3	55.1	53.6
LABR	Accuracy	87.5	**89.3**	86.7	87.6
TEAD	Accuracy	**68.5**	65.4	66.9	66.6
ASTD	Accuracy	**86.2**	84.6	83.6	81.8
PADT	Precision	**85.2**	84.4	85.2	82.4
	Recall	78.1	**83.0**	79.0	79.9
	F1	78.8	**83.8**	80.5	80.9
ARCD	Exact Match	26.1	**29.6**	21.5	21.1
	F1	58.4	**62.2**	49.9	51.3

and PoS tagging, our models perform worse. We believe that our model fails in token-based tasks because AraBERT was trained on a much larger dataset and has larger vocabulary size; more by around 14K tokens compared to our vocabulary size. Note that we only consider AraBERT without any pre-segmentation. On the right side of Table 3, we show the results for the distilled student models. The larger model (MBERTA) uses 4 feed-forward networks while the smaller model (MBERTA II) uses only 1. We notice that it achieves comparable results for Hard, ArsenTD and LABR. It also shows better results in terms of precision for the PoS tagging task. However, it achieves much worse results on question answering task. We believe that the reason behind that is the complexity of the task and the quality of the dataset which has some automatically created translations. In Table 4, we compare all the models in terms of the number of parameters, on-disk space and quantized models. We observe that our teacher model is around half of the size of AraBERT but achieves comparable results. We also notice that we can reduce the model size further by applying quantization using the `tflite` library. This has a big impact especially for resource limited devices. The models can be compressed into even smaller sizes depending on the memory/storage constraints. Of course, further compression will lead to some deterioration in accuracy and other performance measures.

5 Conclusion

In this paper we trained compact Arabic language models through pretraining, distillation and quantization. We showed that such compact language models can achieve comparable results to state-of-the-art models even though they have significantly a fewer number of parameters and while trained on smaller datasets. We benchmark the models on six different classification datasets, one part-of-speech tagging dataset and one question answering dataset. We believe such smaller models can be very practical because they enable deploying such models on resource-limited devices like mobile phones and embedded devices. Moreover, we showed that using smaller datasets with smaller vocabulary, we can achieve reasonable results compared to other models in the literature. This shows that carefully curated datasets with high diversity even if smaller can get some good results. We believe that we can achieve better results by applying some hyper-parameter tuning through approaches like grid search. It is also interesting to look at how the parameters of mobileBERT distillation affect the training process and if changing some of them from the original model can achieve better results for Arabic. Also, a closer look at the distillation process can give us some estimate about the bias towards a certain linguistic phenomenon (Hooker et al., 2020).

Table 4: Memory consumption and size on disk for the fine-tuned and quantized models. M:millions, MB:Megabyte, GB:Gigabyte.

	AraBERT	**IBERTA**	**MBERTA**	**MBERTA II**
Base model number of parameters	135.2 M	66.7 M	23.7 M	**14.2 M**
Fine-tuned model's size on disk	1.6 GB	763 MB	272 MB	**163 MB**
Fine-tuned quantized model's size on disk	516 MB	255 MB	91 MB	**55 MB**

Acknowledgements

We would like to thank Zhiqing Sun (author of mobileBERT paper) for his suggestions and insights to train the models for Arabic. Also, we would like to thank Paige Bailey and Zak Stone for providing the TPU access as part of the TensorFlow Research Cloud (TFRC) program to train our models.

References

Abdul-Mageed, M., Elmadany, A., and Nagoudi, E. M. B. (2020). Arbert & marbert: Deep bidirectional transformers for arabic. *arXiv preprint arXiv:2101.01785*.

Abid, A., Ali, W., Farooq, M. S., Farooq, U., Sabir, N., and Abid, K. (2020). Semi-automatic classification and duplicate detection from human loss news corpus. *IEEE Access*.

Alkhulaifi, A., Alsahli, F., and Ahmad, I. (2020). Knowledge distillation in deep learning and its applications. *arXiv preprint arXiv:2007.09029*.

Alomari, K. M., ElSherif, H. M., and Shaalan, K. (2017). Arabic tweets sentimental analysis using machine learning. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 602–610. Springer.

Aly, M. and Atiya, A. (2013). Labr: A large scale arabic book reviews dataset. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 494–498.

Antoun, W., Baly, F., and Hajj, H. (2020). Arabert: Transformer-based model for arabic language understanding. *arXiv preprint arXiv:2003.00104*.

Baly, R., Khaddaj, A., Hajj, H., El-Hajj, W., and Bashir Shaban, K. (2018). Arsentd-lev: A multi-topic corpus for target-based sentiment analysis in arabic levantine tweets. *OSACT3*.

Cui, J., Kingsbury, B., Ramabhadran, B., Saon, G., Sercu, T., Audhkhasi, K., Sethy, A., Nussbaum-Thom, M., and Rosenberg, A. (2017). Knowledge distillation across ensembles of multilingual models for low-resource languages. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4825–4829. IEEE.

Devlin, J. et al. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. preprint, arXiv.

El-Khair, I. A. (2016). 1.5 billion words arabic corpus. *arXiv preprint arXiv:1611.04033*.

ElJundi, O., Antoun, W., El Droubi, N., Hajj, H., El-Hajj, W., and Shaban, K. (2019). hulmona: The universal language model in arabic. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 68–77.

Elnagar, A., Khalifa, Y. S., and Einea, A. (2018). Hotel arabic-reviews dataset construction for sentiment analysis applications. In *Intelligent Natural Language Processing: Trends and Applications*, pages 35–52. Springer.

Gerz, D., Vulić, I., Ponti, E., Naradowsky, J., Reichart, R., and Korhonen, A. (2018). Language modeling for morphologically rich languages: Character-aware modeling for word-level prediction. *Transactions of the Association for Computational Linguistics*, 6:451–465.

Hooker, S., Moorosi, N., Clark, G., Bengio, S., and Denton, E. (2020). Characterising bias in compressed models. *arXiv preprint arXiv:2010.03058*.

Kudo, T. and Richardson, J. (2018). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Martin, L., Muller, B., Suárez, P. J. O., Dupont, Y., Romary, L., de la Clergerie, É. V., Seddah, D., and Sagot, B. (2019). Camembert: a tasty french language model. *arXiv preprint arXiv:1911.03894*.

Mirzadeh, S.-I., Farajtabar, M., Li, A., Levine, N., Matsukawa, A., and Ghasemzadeh, H. (2019). Improved knowledge distillation via teacher assistant. *arXiv preprint arXiv:1902.03393*.

Mozannar, H., Hajal, K. E., Maamary, E., and Hajj, H. (2019). Neural arabic question answering. *arXiv preprint arXiv:1906.05394*.

Nabil, M., Aly, M., and Atiya, A. (2015). Astd: Arabic sentiment tweets dataset. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 2515–2519.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Radford, A. et al. (2018). *Improving language understanding by generative pre-training*. URL understanding paper. pdf.

Raffel, C. et al. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. preprint, arXiv.

shamela (2005). shamela.

Shen, S., Dong, Z., Ye, J., Ma, L., Yao, Z., Gholami, A., Mahoney, M. W., and Keutzer, K. (2020). Q-bert: Hessian based ultra low precision quantization of bert. In *AAAI*, pages 8815–8821.

Sun, S., Cheng, Y., Gan, Z., and Liu, J. (2019). Patient knowledge distillation for bert model compression. *arXiv preprint arXiv:1908.09355*.

Sun, Z., Yu, H., Song, X., Liu, R., Yang, Y., and Zhou, D. (2020). Mobilebert: a compact task-agnostic bert for resource-limited devices. *arXiv preprint arXiv:2004.02984*.

Tian, Y., Krishnan, D., and Isola, P. (2019). Contrastive representation distillation. *arXiv preprint arXiv:1910.10699*.

Yang, C., Xie, L., Su, C., and Yuille, A. L. (2019). Snapshot distillation: Teacher-student optimization in one generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2859–2868.

You, Y., Li, J., Reddi, S., Hseu, J., Kumar, S., Bhojanapalli, S., Song, X., Demmel, J., Keutzer, K., and Hsieh, C.-J. (2019). Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*.

Zadeh, A. H. and Moshovos, A. (2020). Gobo: Quantizing attention-based nlp models for low latency and energy efficient inference. *arXiv preprint arXiv:2005.03842*.

Zafrir, O., Boudoukh, G., Izsak, P., and Wasserblat, M. (2019). Q8bert: Quantized 8bit bert. *arXiv preprint arXiv:1910.06188*.

Zeman, D. (2015). Universal dependencies. https://github.com/UniversalDependencies/UD_Arabic-PADT.