ArabicTransformer: Efficient Large Arabic Language Model with Funnel Transformer and ELECTRA Objective

Sultan Alrowili
University of Delaware
Newark, Delaware, USA
alrowili@udel.edu

K. Vijay-Shanker
University of Delaware
Newark, Delaware, USA
vijay@udel.edu

Abstract

Pre-training Transformer-based models such as BERT and ELECTRA on a collection of Arabic corpora, demonstrated by both AraBERT and AraELECTRA, shows an impressive result on downstream tasks. However, pre-training Transformer-based language models is computationally expensive, especially for large-scale models. Recently, Funnel Transformer has addressed the sequential redundancy inside Transformer architecture by compressing the sequence of hidden states, leading to a significant reduction in the pre-training cost. This paper empirically studies the performance and efficiency of building an Arabic language model with Funnel Transformer and ELECTRA objective. We find that our model achieves state-of-the-art results on several Arabic downstream tasks despite using less computational resources compared to other BERT-based models.

1 Introduction

The introduction of Transformer and attention mechanism (Vaswani et al., 2017) have achieved significant success by exploiting transfer learning. Bidirectional Encoder Representations from Transformers BERT (Devlin et al., 2019), builds upon the idea of pre-training a Transformer with self-attention on large amounts of unlabeled text. Then, leverage the idea of transfer learning to fine-tune the pre-trained language model on downstream tasks. BERT has achieved impressive performance gains against its predecessor Bi-LSTM (Huang et al., 2015) on many downstream tasks. In the Arabic domain, both AraBERT (Antoun et al., 2020) and AraELECTRA (Antoun et al., 2021) have adapted BERT and ELECTRA (Clark et al., 2020b) models to the Arabic language and show impressive results on downstream tasks.

However, pre-training Transformer-based models, especially at a large scale, requires enormous computational resources. This issue motivates us to investigate a solution to reduce the cost of pre-training Transformer-based models. Reducing the cost to train Arabic language models will help accelerate research advancement in Arabic language processing. Additionally, this will help researchers with limited resources to fine-tune large models.

Several techniques in the literature have suggested solutions to reduce the cost of pre-training and fine-tuning, including cross-layer parameter sharing with ALBERT (Lan et al., 2020) and distillation (Sanh et al., 2020). Distillation and similar techniques have a detrimental effect on performance since they aim to reduce the parameter size. On the other hand, the fine-tuning and inference time for the ALBERT model, especially for $\text{ALBERT}_{\text{xlarge}}$ and $\text{ALBERT}_{\text{xxlarge}}$ scale is significantly higher than $\text{BERT}_{\text{Large}}$ and $\text{ELECTRA}_{\text{Large}}$ as a result of having more hidden layer size. Thus, we seek alternative architectures that could increase the scale of the model without adding additional cost to the pre-training.

Funnel Transformer (Dai et al., 2020) introduces a novel solution to address the cost of pre-training by reconstructing the Transformer architecture using pooling and up-sampling techniques. Additionally, ELECTRA speeds up the pre-training by introducing a new objective function, employing a small generator model trained with maximum likelihood. This study investigates the effect of pre-training Funnel Transformer with ELECTRA objective on the performance of Arabic downstream tasks. Our results show that we achieve state-of-the-art results with less computational resources than existing Arabic language models described in the literature. Thus, our contributions in this paper include :

- We pretrain ArabicTransformer on a large collection of unlabeled Arabic corpora with Funnel Transformer and ELECTRA objective that requires significantly less time and resources than state-of-the-art models.

Findings of the Association for Computational Linguistics: EMNLP 2021, pages 1255–1261
November 7–11, 2021.

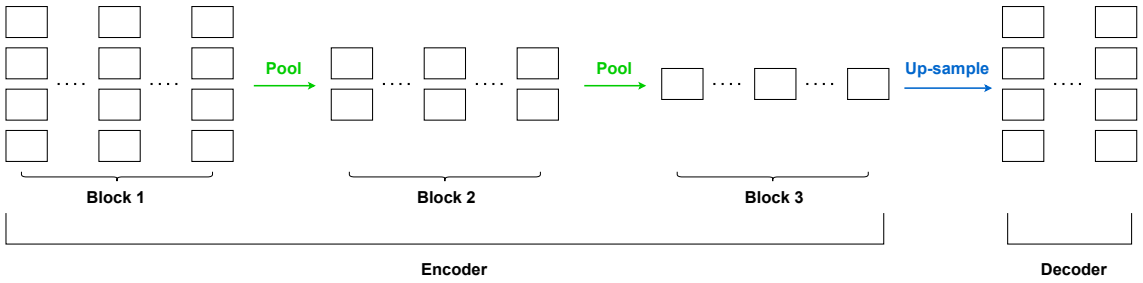

Figure 1: Overview of Funnel Transformer Architecture. Figure adapted from (Dai et al., 2020).

- We fine-tune and evaluate our model on a suite of Arabic downstream tasks, including question answering and sentiment analysis tasks showing that we achieve state-of-the-art performance on several downstream tasks.

- We released our models to the research community along with our GitHub repository [1].

2 Related Work

2.1 ELECTRA

The loss function inside the BERT model consists of semi-supervised learning objectives that aim to capture the contextual representation of an unstructured unlabeled dataset. This loss function in BERT has two objectives: Masked Language Model MLM and Next Sentence Prediction NSP. Several studies have investigated the effect of those two objectives on language model perplexity (Liu et al., 2019), (Lan et al., 2020) . ELECTRA (Clark et al., 2020b), reconstructed the BERT model's loss function based on game theory concepts , particularly the GAN (Goodfellow et al., 2014) and MaskGAN (Fedus et al., 2018) models. In ELECTRA, the loss function is formed as a zero-sum game where the goal of the discriminator and generator is to reach the Nash equilibrium point. This point represents the convergence of the language model to the optimal solution. As a result of having a binary loss function, the ELECTRA model's learning curve is higher than the MLM objective.

2.2 Funnel Transformer

The ELECTRA paper only introduces novelty to the loss function without significant changes to the Transformer architecture. The major problem with Transformer architecture is the sequential redundancy within its structure. This redundancy adds additional pre-training cost to the language model. Funnel Transformer reconstructed the Transformer's architecture to address the redundancy issue.

The key idea is to use a pooling technique to compress the full sequence of hidden states in the encoder part through a series of blocks. Then recover the full sequence representation in the decoder part using an up-sampling technique. A common configuration for the block layout, as shown by (Dai et al., 2020) consists of 3 blocks and a hidden layer size of 768 for base-scale models. For example, an architecture with B6-6-6 design has three blocks where each has 6 layers of a hidden size of 768. A model with a B4-4-4 design consists of three blocks where each has 4 layers of hidden size of 768. Figure 1 shows a high-level illustration of Funnel Transformer architecture.

Funnel Transformer with this novel design managed to save more FLOPs. The saved FLOPs can be used either to increase the model parameters or to speed up the pre-training process (Dai et al., 2020). Results of Funnel Transformer on English domain show significant performance leap, especially at base scale. These results motivate us to investigate the cost and efficiency of pre-training Funnel Transformer in the Arabic domain.

3 Pre-Training the Language Model

3.1 Dataset

We pretrain our models using a collection of large Arabic corpora (45GB) including :

- Arabic Wikipedia dump 1.3GB .

- Abu El-Khair corpus 14GB (El-khair, 2016).

- Unshuffled Arabic Oscar dataset 30GB (Ortiz Suárez et al., 2020).

[1] We released our code and our models at https://github.com/salrowili/ArabicTransformer .

Settings	AraELECTRA	ArabicTransformer		AraBERT$_L$
Model-Scale	Base	B4-4-4	B6-6-6	Large
Hidden Layer Size	768	768	768	1024
Vocabulary Size	64K	50K	50K	64K
Corpora Size	77GB	45GB	45GB	77GB
Pre-Segmentation	No	No	No	Yes (v2) - No (v02)
Learning Rate	2e-4	1e-4	4e-4	-
Max Sequence Length	512	512	512	128-512
Batch Size	256	256	1024	13440-2056
Steps	2M	1M	250k	550K
Computational Ratio	1.0x	0.5x	0.5x	7.8x
Pre-Training Hardware	TPUv3-8	TPUv3-8	TPUv3-32	TPUv3-128

Table 1: The structure and hyperparameters of ArabicTransformer models compared to AraELECTRA and AraBERT$_L$. Computational Ratio (C ratio) represents the training steps multiplied by the batch size where the AraELECTRA model is the baseline. AraBERT$_L$ follows a similar approach to (Devlin et al., 2019) by pretraining AraBERT$_L$ initially for 250K steps with a maximum sequence length of 128 and batch size of 13440. Then, they continue the pre-training for addtional 300K steps with a maximum sequence length of 512 and batch size of 2056. Both AraBERTv2$_L$ and AraBERTv02$_L$ have similar hyperparameters except the use of pre-segmentation.

3.2 Environmental Setup

We pretrain our models using the google cloud compute engine and TensorFlow units (TPUs). We use TensorFlow 1.15 (Abadi et al., 2015) and the open-source code of Funnel Transformer .

3.3 Pre-Training Hyperparameters

Table 1 provides our choice of pre-training hyperparameters for our models against both AraELECTRA (Antoun et al., 2021) and AraBERT (Antoun et al., 2020). We build our base model with a structure that consists of a 6-6-6 block layout and 768 hidden layer size. This block layout increases the model parameters up to 1.39x compared to BERT$_{\text{Base}}$ and ELECTRA$_{\text{Base}}$ (Dai et al., 2020). Additionally, we pretrain a smaller model with a 4-4-4 block layout. This model has a similar parameter size to ELECTRA$_{\text{base}}$.

Instead of using a batch size of 256 as proposed in the original paper of ELECTRA and AraELECTRA, we increase the batch size to 1024 and the learning rate to 4e-4 for our B6-6-6 model. Several studies in the literature support the idea of using large batch size since it improves the language model's perplexity (Liu et al., 2019), (You et al., 2020). On the other hand, we use similar pre-training hyperparameters to (Dai et al., 2020) for our B4-4-4 model. We build our vocabulary file with a size of 50K without using Farasa segmenter (Abdelali et al., 2016). Farasa segmenter is a tool that breaks words into stems, suffixes, and prefixes (Antoun et al., 2020).

4 Fine-tuning on Downstream Tasks

4.1 Question Answering

To compare our model with existing models in the literature, we use ARCD (Mozannar et al., 2019) and the Arabic portion of TyDi QA (Clark et al., 2020a). Both ARCD and TyDi QA are in format of SQuADv1.1 dataset (Rajpurkar et al., 2016). Similar to the AraELECTRA and AraBERT approach, we fine-tune our model on both ArabicSQuAD and ARCD training datasets. Then, we evaluate our model on the test portion of the ARCD dataset. Moreover, as is a common practice, we use a preprocessing script developed by the AUB MIND lab, which fixes the position of text spans and handles special characters in the ARCD dataset.

Our baseline models for QA tasks including AraBERTv02$_{\text{large}}$, AraBERTv2$_{\text{large}}$ (Antoun et al., 2020), Arabic-ALBERT$_{\text{xlarge}}$ (Safaya, 2020) and AraELECTRA. We follow the same split of training and development dataset used by AraELECTRA, summarized in Table 2. We only include models that have reported results in the literature for ARCD and TyDi QA in our baseline models.

Task	Train	Test
ARCD Mozannar et al. (2019)	49,037	702
TyDiQA Clark et al. (2020a)	14,805	921

Table 2: Summary of Question Answering datasets.

4.2 Sentiment Analysis

Sentiment analysis (SA) task is a text classification task where we classify each sentence (sequence) with a (sentiment) label. Those labels can be either binary or categorical. Our choice for sentiment analysis task including Hotel Arabic-Reviews Dataset (HARD) (Elnagar et al., 2018), Arabic Jordanian General Tweets (AJGT) (Dahou et al., 2019) and ArScarcasmv2 (sentiment shared task) (Abu Farha et al., 2021). Our baseline models for sentiment analysis tasks including XLM-R_{Base};XLM-R_{Large} (Conneau et al., 2020), AraBERTv2$_{Large}$;AraBERTv02$_{Large}$ (Antoun et al., 2020), AraELECTRA (Antoun et al., 2021), ARBERT and MARBERT (Abdul-Mageed et al., 2021). Table 3 summarize the details of the dataset we use for SA tasks.

Task	Labels	Train	Test
HARD	[neg, pos]	84.5k	21.1k
ArSarcasm	[neg, neut, pos]	12.5k	3K
AJGT	[neg, pos]	1.4k	360

Table 3: Summary of sentiment analysis (SA) datasets. (neg: negative , pos:positive , neut: neutral)

4.3 Fine-tuning Hyperparameters

We extensively conduct a grid search to find the best hyperparameters for each task using the TPUv3-8 unit and Tensorflow 1.15. Our grid search space range is : learning rate (2e-5, 3e-5, 4e-5, 5e-5, 6e-5) , batch size (16, 24, 32, 40, 48, 64), layer-wise decay (0.75, 0.8, 1.0), max sequence length (384, 512) and epochs number (2-12). For sentiment analysis tasks, we use 256 as the maximum sequence length. We report our result as the best result out of five different runs for each task, which is a similar approach used by both ELECTRA (Clark et al., 2020b) and BERT (Devlin et al., 2019). We use the following seeds: 123, 1234, 12345, 666, 42 for each run. We define our choices of seeds to improve the reproducibility of results.

5 Results and Discussion

5.1 Pre-Training

Table 4 shows the pre-training time of our models against AraELECTRA. The reduction in cost for both B6-6-6 and B4-4-4 models is a result of using a 0.5x C ratio (batch x steps) compared to AraELECTRA. Additionally, Funnel-transformer architecture contributes to additional reduction from 0.5x to 0.4x (B6-6-6) and from 0.5x to 0.31x for (B4-4-4) model. We have also evaluated our pretrained models on a random Arabic sample (2.5M words with a size of 25MB) from CCNet dataset (Wenzek et al., 2020). Our evaluation shows that the B4-4-4 model has a loss score of 11.58% against 11.12% for the B6-6-6 model.

Model	Hardware	Time	Cost
AraELECTRA	TPUv3-8	24d	1.00x
B6-6-6 (Ours)	TPUv3-32	2d 10h	0.40x
B4-4-4 (Ours)	TPUv3-8	7d 11h	0.31x

Table 4: Pretraining cost of our models compared to AraELECTRA.

5.2 Question Answering

Table 5 shows the performance of our models on QA tasks compared to state-of-the-art models reported by (Antoun et al., 2021).

Model	TyDiQA		ARCD	
	EM	F1	EM	F1
AraBERT02$_{L}$	73.72	86.03	36.89	71.32
AraBERT2$_{L}$	64.49	82.15	34.19	68.12
ArabicALBERT$_{xl}$	71.12	84.59	**37.75**	68.03
AraELECTRA$_{B}$	74.91	86.68	37.03	71.22
Ours B4-4-4	74.70	85.89	31.48	67.70
Ours B6-6-6	**75.35**	**87.21**	36.89	**72.70**

Table 5: Evaluation results of ArabicTransformer compared to SOTA models on QA tasks. We use F1 and exact match (EM) score for both tasks which is a common practice to evaluate task in format of SQuAD1.1. We use reported number by (Antoun et al., 2021) for our baseline models results.

Our base-scale model (B6-6-6) outperforms AraELECTRA on both TyDi QA and ARCD tasks. This performance improvement is due to the fact that B6-6-6 has larger parameter size (1.39x) than ELECTRA$_{Base}$ architecture. Furthermore, our small model (B4-4-4) has a competitive performance against AraELECTRA and AraBERT$_{L}$ on the TyDi QA task, especially on the exact match (EM) metric. The discrepancy in performance between ARCD and TyDi QA tasks is due to the poor quality of the training dataset that we use for the ARCD task. This training dataset uses the Arabic Translation of SQuAD1.1 dataset (Antoun et al., 2021).

5.3 Sentiment Analysis

Table 6 summarizes the performance of Arabic-Transformer against SOTA models on sentiment analysis tasks. In both HARD and ArScarcasm

Task	HARD	AJGT	Scarcasm	
Metric	Acc.	Acc.	Acc.	$F1_{PN}$
$XLM\text{-}R_B$	95.7	89.4	64.3	66.1
$XLM\text{-}R_L$	96.0	91.9	67.8	69.9
$AraBERT02_L$	96.4	94.5	69.5	71.8
$AraBERT2_L$	96.5	**96.4**	70.0	72.4
$ARBERT_B$	96.1	94.4	67.3	69.5
$MARBERT_B$	96.2	96.1	69.3	72.4
$AraELECT_B$	96.4	95.0	69.6	72.3
Ours B4-4-4	96.5	95.0	70.4	72.8
Ours B6-6-6	**96.6**	95.0	**70.8**	**74.0**

Table 6: Evaluation results of our models compared to SOTA models. $F1_{PN}$ score takes only positive and negative classes in calculation excluding neutral class. For HARD and AJGT tasks, we use reported numbers of XLM-R, ARBERT and MARBERT (Abdul-Mageed et al., 2021). For ArScarcasm task we use the reported numbers by (Farha and Magdy, 2021). We reproduced AraELECTRA results on all tasks and $AraBERT_L$ models on HARD and AJGT tasks.

tasks, our models perform better than other state-of-the-art models, including larger models such as $XLM\text{-}R_L$ and $AraBERTv2_L$. However, our models perform worse on the AJGT task. We attribute this performance to the fact that the AJGT task has a relatively smaller dataset than HARD and ArScarcasm. Therefore, it is more sensitive to hyperparameter tuning, leading to a significant performance fluctuation.

5.4 Pre-Segmentation

$AraBERTv2_L$, in contrast to other models in Table 5 and Table 6, uses Farasa segmenter. Although $AraBERTv2_L$ outperforms AraELECTRA on the ArScarcasm task, $AraBERTv2_L$ performs worse on QA tasks despite having a 7.5x computational ratio compared to AraELECTRA. The performance of AraELECTRA, $AraBERTv02_L$ and our models against $AraBERTv2_L$ on the QA task suggests that pre-segmentation do not always lead to better performance on span-based QA tasks. In contrast, pre-segmentation contributes to the performance improvement of AraBERTv2 on sentiment analysis tasks against AraBERTv02, especially on the ArScarcasm task.

5.5 Efficiency of Fine-Tuning

Table 7 shows the fine-tuning time of our models compared to $AraELECTRA_{base}$. In addition to improvment in fine-tuning speed, we also observe that B4-4-4 uses less memory consumption than AraELECTRA.

Model	Time / Ratio	#Params
$AraELECTRA_B$	25:31 (1.00x)	1.00x
Ours (B4-4-4)	18:27 (0.72x)	1.00x
Ours (B6-6-6)	27:24 (1.07x)	1.39x

Table 7: Fine-Tuning time of our models compared to SOTA models. We finetune all models on HARD dataset for 3 epochs and with a batch size of 32 using V100 16GB Tesla GPU with PyTorch (FP16 - O2). Parameters ratio does not include embedding matrix.

6 Conclusion

We introduce Arabic Transformer, a pretrained Arabic language representation model based on Funnel Transformer and ELECTRA objective. We show that we achieve state-of-the-art results on several Arabic downstream tasks, including question answering and sentiment analysis tasks. Additionally, we show that our models are computationally efficient and pretrained using significantly less resources than state-of-the-art models. For future work, we plan to investigate different designs of the Funnel Transformer, including larger models such as (B8-8-8).

7 Acknowledgement

We would like to acknowledge the support we have from Tensorflow Research Cloud (TFRC) team to grant us access to TPUv3 units. The authors also would like to thank anonymous reviewers from EMNLP21 for their constructive feedback on our initial manuscript.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan,

Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Ahmed Abdelali, Kareem Darwish, Nadir Durrani, and Hamdy Mubarak. 2016. Farasa: A fast and furious segmenter for Arabic. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 11–16, San Diego, California. Association for Computational Linguistics.

Muhammad Abdul-Mageed, AbdelRahim Elmadany, and El Moatez Billah Nagoudi. 2021. ARBERT & MARBERT: Deep bidirectional transformers for Arabic. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7088–7105, Online. Association for Computational Linguistics.

Ibrahim Abu Farha, Wajdi Zaghouani, and Walid Magdy. 2021. Overview of the wanlp 2021 shared task on sarcasm and sentiment detection in arabic. In *Proceedings of the Sixth Arabic Natural Language Processing Workshop*.

Wissam Antoun, Fady Baly, and Hazem Hajj. 2020. AraBERT: Transformer-based model for Arabic language understanding. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*, pages 9–15, Marseille, France. European Language Resource Association.

Wissam Antoun, Fady Baly, and Hazem Hajj. 2021. AraELECTRA: Pre-training text discriminators for Arabic language understanding. In *Proceedings of the Sixth Arabic Natural Language Processing Workshop*, pages 191–195, Kyiv, Ukraine (Virtual). Association for Computational Linguistics.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020a. TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020b. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Abdelghani Dahou, Mohamed Abd Elaziz, Junwei Zhou, Shengwu Xiong, and Rodolfo Zunino. 2019. Arabic sentiment classification using convolutional neural network and differential evolution algorithm. *Intell. Neuroscience*, 2019.

Zihang Dai, Guokun Lai, Yiming Yang, and Quoc Le. 2020. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. In *Advances in Neural Information Processing Systems*, volume 33, pages 4271–4282. Curran Associates, Inc.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ibrahim Abu El-khair. 2016. 1.5 billion words arabic corpus.

Ashraf Elnagar, Yasmin S. Khalifa, and Anas Einea. 2018. *Hotel Arabic-Reviews Dataset Construction for Sentiment Analysis Applications*, pages 35–52. Springer International Publishing, Cham.

Abu Farha and Walid Magdy. 2021. Benchmarking transformer-based language models for arabic sentiment and sarcasm detection. In *Proceedings of the Sixth Arabic Natural Language Processing Workshop*.

William Fedus, Ian Goodfellow, and Andrew M. Dai. 2018. MaskGAN: Better text generation via filling in the . In *International Conference on Learning Representations*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27, pages 2672–2680. Curran Associates, Inc.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

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.

Hussein Mozannar, Elie Maamary, Karl El Hajal, and Hazem Hajj. 2019. Neural Arabic question answering. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 108–118, Florence, Italy. Association for Computational Linguistics.

Pedro Javier Ortiz Suárez, Laurent Romary, and Benoît Sagot. 2020. A monolingual approach to contextualized word embeddings for mid-resource languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1703–1714, Online. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Ali Safaya. 2020. Arabic-albert.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. 2020. CCNet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4003–4012, Marseille, France. European Language Resources Association.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. 2020. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations*.