MISSION at KSAA-CAD 2024: AraT5 with Arabic Reverse Dictionary

THAMER MASEER ALHARBI
tamr4947@gmail.com

Abstract

This research paper presents our approach for the KSAA-CAD 2024 competition, focusing on Arabic Reverse Dictionary (RD) task (Alshammari et al., 2024). Leveraging the functionalities of the Arabic Reverse Dictionary, our system allows users to input glosses and retrieve corresponding words. We provide all associated notebooks and developed models on GitHub and Hugging face, respectively. Our task entails working with a dataset comprising dictionary data and word embedding vectors, utilizing three different architectures of contextualized word embeddings: AraELECTRA, AraBERTv2, and camelBERT-MSA. We fine-tune the AraT5v2-base-1024 model for predicting each embedding, considering various hyperparameters for training and validation. Evaluation metrics include ranking accuracy, mean squared error (MSE), and cosine similarity. The results demonstrate the effectiveness of our approach on both development and test datasets, showcasing promising performance across different embedding types.

1 Introduction

This research paper is prepared for KSAA-CAD 2024 (Alshammari et al., 2024) and describes our approach[1] as part of our contribution to this competition. All associated notebooks can be accessed on GitHub[2], and the developed models are available on Hugging face [3]. This task focuses on Arabic Reverse Dictionary word and gloss to word embedding, leveraging the Arabic Reverse Dictionary's functionality. It enables users to input glosses and receive corresponding words for better comprehension. For instance, if you forget a word, you can describe it, and the model will provide either the exact word or the closest synonym. This process facilitates gloss to word embedding. The task involves working with a dataset comprising dictionary data and word embedding vectors, utilizing three different architectures of contextualized word embeddings. The dataset includes three dictionaries of Contemporary Arabic Language: C̈ontemporary Arabic Language Dictionary" by Ahmed Mokhtar Omar (Omar, 2008), "Mu'jam Arriyadh" dictionary of Arabic contemporary language (Altamimi et al., 2023), "Al Wassit LMF Arabic Dictionary" (Namly, 2015)(Alshammari et al., 2024).

These dictionaries conform to the ISO Lexical Markup Framework (LMF) standard (Francopoulo et al., 2006) and are based on lemmas, including glosses, part of speech (POS) tags, and examples. Previous experiments indicated that fixed word embeddings like word2vec did not perform well, prompting a shift to contextualized word embeddings using advanced models such as Electra(Clark et al., 2020), BERT(Devlin et al., 2018), AraELECTRA(Antoun et al., 2021), AraBERTv2(Antoun et al.), and camelBERT-MSA(Inoue et al., 2021). AraELECTRA, based on the ELECTRA framework (Clark et al., 2020), trains a discriminator model, while AraBERTv2 uses Farasa segmentation(Darwish and Mubarak, 2016) and camelBERT-MSA is pretrained on Modern Standard Arabic (MSA) corpora, built on the BERT architecture (Alshammari et al., 2024).

2 Datasets

A simplified version of the input data is shown in Table 1, and the data in JSON format is illustrated in Figure 1. The target output of embeddings, as shown in Figure 1, utilizes the following models:

- AraELECTRA is represented as Electra (Antoun et al., 2021).

[1] https://github.com/thxa/KSAA-CAD2024/blob/main/RD_TASK.ipynb
[2] https://github.com/thxa/KSAA-CAD2024
[3] https://huggingface.co/7H4M3R/AraT5-multi-word-embedding/tree/main

- AraBERTv2 is represented as bertSEG (Antoun et al.).
- camelBERT-MSA is represented as bertMSA (Inoue et al., 2021).

The statistics of the dataset are summarized in Table 2.

id	ar.962714
word	أَكْمَدَ
pos	V
gloss	غَمَّ وَأَمْرَضَ القَلْبَ

Table 1: Example of definition in Arabic.

```
{
"id" : "ar.962714",
"word" : "أَكْمَدَ",
"pos" : "V",
"gloss" : "غَمَّ وَأَمْرَضَ القَلْبَ",
"electra" : [0.22,...], // length 256
"bertseg" : [0.13, ...], // length 768
"bertmsa" : [-1.05,...] // length 768
}
```

Figure 1: Example of data in JSON format.

Task	Dev	Train	Test
RD entries	3921	31372	3922

Table 2: Data Statistics.

3 Metrcies

The model evaluation process follows a hierarchical approach using multiple metrics.

3.1 Ranking Metric

The ranking metric evaluates how accurately the model ranks predictions against the ground truth (Alshammari et al., 2024). The following equations describe the computation steps:

First, normalize the vectors:

$$\mathbf{x}_{norm} = \frac{\mathbf{x}}{||\mathbf{x}||} \tag{1}$$

Then, compute the similarity matrix between the normalized predictions and the targets:

$$\mathbf{A} = \mathbf{x}_{norm,preds} \cdot \mathbf{x}_{norm,targets}^{\top} \tag{2}$$

Next, calculate the rank matrix:

$$\mathbf{r} = \sum(\mathbf{A} \geq diag(\mathbf{A})),\ axis = 1 \tag{3}$$

Compute the mean rank:

$$m = mean(\mathbf{r}) \tag{4}$$

Finally, normalize the mean rank by the number of elements N:

$$n = \frac{m}{N} \tag{5}$$

The equations are derived from the code provided in appendix B.

3.2 Mean Squared Error (MSE)

Measures the average of the squares of the errors between predicted and actual values, providing insight into the model's overall accuracy (PyTorch, 2024b).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{6}$$

3.3 Cosine Similarity

Assesses the cosine of the angle between two non-zero vectors, providing additional insights into the similarity between predicted and actual values based on their direction (PyTorch, 2024a).

$$cosine\_similarity(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_1 \cdot \mathbf{x}_2}{||\mathbf{x}_1||||\mathbf{x}_2||} \tag{7}$$

4 System

4.1 Hardware

We utilized Google Colab[4] with GPU acceleration, specifically NVIDIA A100-SXM4-40GB, for training the model.

4.2 Model

We fine-tune the AraT5v2-base-1024 model (Elmadany et al., 2023). The model undergoes three separate fine-tuning processes, each aimed at predicting different embeddings (e.g., AraELECTRA (Antoun et al., 2021), AraBERTv2 (Antoun et al.), camelBERT-MSA(Inoue et al., 2021)). The model consists of the T5 architecture (Raffel et al., 2023), followed by a final linear layer for embedding generation. The inputs to the model are token IDs (tokenized input sequences or glosses), attention masks (to avoid attending to padding tokens), and labels (target word tokens used to compute the loss

[4]`https://colab.research.google.com/`

during training). The outputs from the model are the sequence-to-sequence loss used for training and the final word embedding generated by the model as shown in 2.

4.3 Train and Validation

Table 3 lists the hyperparameters used for both training and validation. The loss function employed is Mean Squared Error (MSE) 3.2, and model selection is based on cosine similarity 3.3 evaluated on the validation data at each epoch. The only difference between the models is the number of epochs, due to budget constraints. A batch size of 32 was chosen to avoid crashes due to limited computing power, with the other hyperparameters kept the same as the baseline.

Here is an explanation of each hyperparameter:

- EPOCHS: Number of complete passes through the training dataset.
- BATCH SIZE: Number of training examples used in one iteration.
- LEARNING RATE: Step size at each iteration while moving toward a minimum of the loss function.
- BETA1: Exponential decay rate for the first moment estimates in the AdamW optimizer.
- BETA2: Exponential decay rate for the second moment estimates in the AdamW optimizer.
- WEIGHT DECAY: Regularization term to prevent overfitting by penalizing large weights.
- Optimizer: AdamW, which is a variant of the Adam optimizer that includes weight decay for better regularization.

The models for predicting embeddings are deployed on Hugging Face as follows:

- Electra model[5] (trained for 20 epochs).
- bertMSA model[6] (trained for 10 epochs).
- bertSEG model[7] (trained for 10 epochs).

[5]https://huggingface.co/7H4M3R/AraT5-multi-word-embedding/blob/main/best_model_electra_v3.pt

[6]https://huggingface.co/7H4M3R/AraT5-multi-word-embedding/blob/main/best_model_bertmsa_v1.pt

[7]https://huggingface.co/7H4M3R/AraT5-multi-word-embedding/blob/main/best_model_bertseg_v1.pt

Hyper Parameter	Value
EPOCHS	10,20
BATCH_SIZE	32
LEARNING_RATE	1.0e-4
BETA1	0.9
BETA2	0.999
WEIGHT_DECAY	1.0e-6
Optimizer	AdamW

Table 3: Hyper Parameters.

5 Result

The results are as shown in table[8] 4.

Embedding	Data	Rank	MSE	Cos
Electra	Dev	0.2469	0.2297	0.5515
	Test	0.2482	0.2298	0.5507
bertMSA	Dev	0.3334	0.3257	0.7188
	Test	0.3315	0.3224	0.7219
bertSEG	Dev	0.4126	0.0777	0.7745
	Test	0.4165	0.0781	0.7731

Table 4: Results on DevSet and TestSet for RD Task.

5.1 Analysis

From the table 4, we observe the following:

- Electra shows the lowest rank and MSE but also the lowest cosine similarity, indicating it may not perform as well in capturing semantic similarity compared to the other models.
- bertMSA shows a balanced performance with moderate scores in all metrics.
- bertSEG achieves the highest cosine similarity, suggesting better semantic representation, but it has higher rank and MSE values.

6 Discussion

The results from Table 4 indicate varying performance levels across the different embeddings when predicted using the AraT5-base-1024 model. The results highlight the trade-offs between different embedding models. A model like Electra, which performs well in ranking and error metrics, might not capture semantic similarities as effectively as bertSEG. Choosing the right model depends on the specific requirements of the task. For tasks that require precise semantic understanding, models with

[8]https://codalab.lisn.upsaclay.fr/competitions/18510#results

higher cosine similarity like bertSEG are preferable. For tasks where ranking and minimizing errors are more important, models like Electra may be more suitable. bertMSA offers a balanced performance, making it a strong candidate for general-purpose use across a variety of NLP tasks.

7 Future Work

- Conduct more extensive testing on different datasets and tasks to gain a deeper understanding of the strengths and weaknesses of each embedding model.
- Explore hybrid models or ensemble approaches that combine the strengths of different embeddings to potentially yield better performance across all metrics.
- Fine-tune these models on more specific datasets or tasks to improve their performance further, especially in capturing the nuances of the Arabic language.
- Investigate new architectures such as Mamba (Gu and Dao, 2024) or KAN (Liu et al., 2024), or create a new architecture.

8 Conclusion

The study highlights the varying strengths and weaknesses of different Arabic embeddings predicted using the AraT5-base-1024 model. By understanding these differences, practitioners can make more informed choices about which embedding model to use for specific NLP tasks.

Acknowledgments

I would like to acknowledge the organizers of KSAA-CAD for their support, and I extend my thanks to all the teams from the previous version for their contributions.

References

Rawan Al-Matham, Waad Alshammari, Abdulrahman AlOsaimy, Sarah Alhumoud, Asma Wazrah, Afrah Altamimi, Halah Alharbi, and Abdullah Alaifi. 2023. KSAA-RD shared task: Arabic reverse dictionary. In *Proceedings of ArabicNLP 2023*, pages 450–460, Singapore (Hybrid). Association for Computational Linguistics.

Waad Alshammari, Amal Almazrua, Asma Al Wazrah, Rawan Almatham, Muneera Alhoshan, Abdulrahman AlOsaimy, and Alfaifi Abdullah Altamimi, Afrah and. 2024. KSAA-CAD: Contemporary Arabic dictionary shared task. In *Proceedings of the 2nd Arabic Natural Language Processing Conference (Arabic-NLP), Part of the ACL 2024*. Association for Computational Linguistics.

Wissam Antoun, Fady Baly, and Hazem Hajj. Arabert: Transformer-based model for arabic language understanding. In *LREC 2020 Workshop Language Resources and Evaluation Conference 11–16 May 2020*, page 9.

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.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *Preprint*, arXiv:2003.10555.

Kareem Darwish and Hamdy Mubarak. 2016. Farasa: A new fast and accurate Arabic word segmenter. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1070–1074, Portorož, Slovenia. European Language Resources Association (ELRA).

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

Ahmed Elbakry, Mohamed Gabr, Muhammad El-Nokrashy, and Badr AlKhamissi. 2023. Rosetta stone at KSAA-RD shared task: A hop from language modeling to word–definition alignment. In *Proceedings of ArabicNLP 2023*, pages 477–482, Singapore (Hybrid). Association for Computational Linguistics.

AbdelRahim Elmadany, El Moatez Billah Nagoudi, and Muhammad Abdul-Mageed. 2023. Octopus: A multitask model and toolkit for Arabic natural language generation. In *Proceedings of ArabicNLP 2023*, pages 232–243, Singapore (Hybrid). Association for Computational Linguistics.

Gil Francopoulo, Monte George, Nicoletta Calzolari, Monica Monachini, Nuria Bel, Mandy Pet, and Claudia Soria. 2006. Lexical markup framework (LMF). In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces. *Preprint*, arXiv:2312.00752.

Go Inoue, Bashar Alhafni, Nurpeiis Baimukan, Houda Bouamor, and Nizar Habash. 2021. The interplay of variant, size, and task type in Arabic pre-trained language models. In *Proceedings of the Sixth Arabic Natural Language Processing Workshop*, Kyiv, Ukraine (Online). Association for Computational Linguistics.

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. 2024. Kan: Kolmogorov-arnold networks. *Preprint*, arXiv:2404.19756.

PyTorch. 2024a. Cosine similarity. Accessed: 2024-06-27.

PyTorch. 2024b. Mean squared error loss. Accessed: 2024-06-27.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the limits of transfer learning with a unified text-to-text transformer. *Preprint*, arXiv:1910.10683.

A Model Architecture

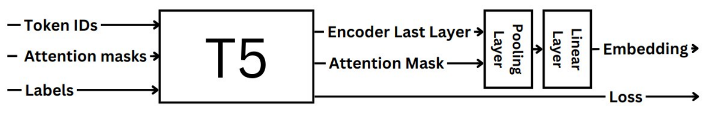

Figure 2: Model Architecture

B Ranking Metric

The code is derived from the baseline (Alshammari et al., 2024).

```
def rank_cosine(preds, targets):
    assocs = F.normalize(preds) @ F.normalize(targets).T
    refs = torch.diagonal(assocs, 0).unsqueeze(1)
    ranks = (assocs >= refs).sum(1).float()
    assert ranks.numel() == preds.size(0)
    ranks = ranks.mean().item()
    return ranks / preds.size(0)
```