Qamosy at KSAA-RD shared task: Semi Decoder Architecture for Reverse Dictionary with SBERT Encoder

Serry Sibaee
Serrytowork@gmail.com

Samar Ahmad
Samar.sass6@gmail.com

Ibrahim Khurfan
ibraheemkhurfan@gmail.com

Vian Sabeeh
Middle Technical University
viantalal@mtu.edu.iq

Ahmed Bahaaulddin
Middle Technical University
ahmedbahaaulddin@mtu.edu.iq

Hanan M. Belhaj
The Libyan Academy
h.belhaj@it.lam.edu.ly

Abdullah I. Alharbi
King Abdulaziz University
aamalharbe@kau.edu.sa

Abstract

A reverse dictionary takes a descriptive phrase of a particular concept and returns words with definitions that align with that phrase. While many reverse dictionaries cater to languages such as English and are readily available online or have been developed by researchers, there is a notable lack of similar resources for the Arabic language. This paper describes our participation in the Arabic Reverse Dictionary shared task. Our proposed method consists of two main steps: First, we convert word definitions into multidimensional vectors. Then, we train these encoded vectors using the SemiDecoder model for our target task. Our system secured 2nd place based on the Rank metric for both embeddings (Electra and Sgns).

1 Introduction

A reverse dictionary takes a phrase describing a specific concept as input and provides words whose definitions match that entered phrase. In contrast, a regular or common (forward) dictionary function contains word-to-meaning or definition mappings, which represents a useful solution for readers when encountering unfamiliar words in a text. For example, a forward dictionary would tell the user that (تحنن عليه - He felt compassion for him) means (ترحم، تعطف عليه ورحمه - have mercy on him) whereas a reverse dictionary allows the user to input the phrase (ترحم، تعطف عليه ورحمه - have mercy on him) and would likely produce the word(تحنن عليه - He felt compassion for him) along with other words having similar meanings as the output.

Reverse dictionaries offer significant practical value; primarily, they are highly effective in resolving the tip-of-the-tongue phenomenon which we encounter every day; people have difficulties finding the precise word to convey their thoughts, despite being on their tongue tip. As a result, they use phrases to explain the word or the concept. This challenge could stem from memory retrieval issues or a limited understanding of a particular language. Such a predicament is widespread, especially when someone is endeavoring to learn a new language and has a restricted number of vocabulary words or people who write frequently and seek a word that precisely matches their intended thought or expression.

Regarding natural language processing (NLP), reverse dictionaries serve various purposes. One of these is assessing sentence representation quality. Additionally, they prove advantageous in tasks related to text-to-entity mapping, such as question answering and information retrieval. Moreover, Reverse dictionaries consider not only the individual meanings of words but also how those meanings change when combined. Many words have synonyms, making determining the exact match for a given definition difficult. For instance, the input "to come together" could correspond to various options like "meet," "gather," "assemble," and more. Consequently, reverse dictionaries offer several potential word options rather than one possible word.

Numerous reverse dictionaries have been available online or have been created by researchers catering to different languages like English, Japanese, Turkish, French, and Persian. However, a noticeable absence of equivalent resources can be observed in the Arabic language. This shortage could stem from the lack of appropriate or substantial datasets containing words and their respective definitions which entails significant efforts in collecting and structuring language data. This paper, outlines our contribution to the Arabic Reverse Dictionary shared task. Our approach involves two phases: first, transforming word definitions into multidimensional vectors, and then training these vectors with the Simi-Decoder model for the intended task.

2 Related Work

Previously, researchers used a traditional approach for tackling the reverse dictionary problem, called semantic analysis using WordNet (Méndez et al., 2013). To determine how similar two words are, they made use of semantic similarity measurements. They used similarity between a word and an input phrase using a distance-based similarity measure. This measurement was considered necessary to determine connections between the term and the input words in the graph (Thorat and Choudhari, 2016). Recently, many researchers have been using embedding techniques in conjunction with neural networks and Deep learning(DL) to improve the generation of reverse dictionaries. Pilehvar (2019) used a combination of Bidirectional Long Short-Term Memory (BiLSTM) and cascade forward neural network (CFNN) to improve the neural reverse dictionary (NRD's) performance; outperforming a commercial reverse dictionary system (OneLook[1]) in various metrics. To find whether a proposed neural network framework is universally effective across all languages, Bendahman et al. (2022) used sequential models with a variety of neural networks, such as embedding networks, denser networks and Long Short-Term Memory LSTM networks. In (Chen and Zhao, 2022), the authors present a model that can be seen as a neural dictionary with two-way indexing and querying, embedding both words and definitions within a common semantic space. Their approach involves separate encoder and decoder networks for words and definitions. These networks are complemented by a shared layer that aligns them within the same representation space. In (Agrawal et al., 2021), they combine Continuous Bag-of-Words (CBOW) model and recurrent neural network (RNN) to employ a reverse dictionary that considers both word order and context. Another group of researchers focused on the concept of attention to better understand the context and meaning of the text. In (Hedderich et al., 2019), they used attention mechanisms to integrate multi-sense embedding using LSTM and contextual word embedding (Bidirectional Encoder Representations from Transformers (BERT) to enhance performance in the reverse dictionary task. As for (Malekzadeh et al., 2021), they utilised different models to simulate the functionality of a reverse dictionary. These included a Bag of Words (BOW) model, an RNN model with additive attention, and a BiLSTM model. Each of these was used to map a descriptive phrase to their corresponding words. Others (Qi et al., 2020; Zhang et al., 2020) used a sentence encoder based on a BiLSTM with an attention mechanism along with four characteristic predictors. These predictors assist in identifying the part-of-speech, morphemes, word category, and other relevant information.

[1]https://www.onelook.com/thesaurus/

3 Methodology

In this section, we will start describing the dataset used for our work. Then, we will explain our approach, divided into two primary steps. The first is to represent or encode the inputs (the definitions of the words) as multidimensional vectors. The second stage is to train the encoded inputs using a Simi-Decoder model for our downstream task.

3.1 Data Description

The used dataset is created and released by the shared task's organizers. They were chosen from the LMF Contemporary Arabic dictionary [2] and subsequently revised and refined by our annotation team. The total entries for all sets are 58,010 (Train: 45200, Dev: 6400 and Test: 6410). The datasets are in JSON format, comprising multiple examples. Each example within this dataset has six main elements. The "id" element indicates a language-specific unique identifier for a target "word". The "gloss" element provides a traditional dictionary definition, which is the source for the RD task. "enId" links to an identifier in the English dictionary. The remaining elements, namely "sgns" and "electra", represent different types of embeddings given as float arrays. Specifically, "sgns" relates to word2vec's skip-gram embeddings, while "electra" is tied to Transformer-based embeddings. Both can be targets in the RD task.

3.2 Encoder: encoding the input

In the first part of the work, we used the Sentence Transformer (SBERT) (Reimers and Gurevych, 2019) to represent the input (words' definitions). SBERT is a framework designed for generating

[2]https://lindat.cz/repository/xmlui/handle/11372/LRT-1944?locale-attribute=en

Model	NO.	h1	h2	h3	h4	Activation	Output	Dropout	Epochs
Electra	1	1024*6	1024*6	-	-	ReLU	256	-	100
	2	512*8	512*4	512*2	512	GELU	256	-	300
	3	512*8	512*4	512*2	512	GELU	256	-	1000
	4	512*8	512*4	512*2	512	GELU	256	-	2000
	5	512*8	512*6	512*4	512	GELU	256	0.65	4000
	6	512*8	512*4	512*2	512	LeakyReLU	256	-	2000
	7	512*8	512*6	512*4	512	GELU	256	0.60	1000
sgns	1	1024*6	1024*6	-	-	ReLU	300	-	100
	2	512*8	512*4	512*2	512	GELU	300	-	1000

Table 1: Arciticture Semi-Decoder MLP

fixed-length embeddings for sentences, optimizing for semantic similarity and efficiency over traditional BERT models. It is optimized for processing multiple sentences simultaneously, ensuring faster results. Its training structure prioritizes semantic similarity, meaning similar sentences have close vector representations. Furthermore, SBERT offers a range of pre-trained models for different tasks and languages, including Arabic.

Using SBERT in our task, the size of the encoded inputs is (d=512) for every definition of the words. This approach helped to make the training easier and more efficient by not worrying about the input size. We used The 'distiluse-base-multilingual-cased' model, which has proven effective in generating dependable embeddings across multiple languages (Reimers and Gurevych, 2020), making it an ideal choice for our focus on Arabic. The output of this step is encoded inputs that will be passed to the next stage, as can be seen in Figure 1.

3.3 Decoder: Semi-Decoder MLP

In the second part of the work, we use many Multi-Layer Perceptron(MLP) architecture (summarized in Table 1 as a decoder model to transform the inputs with the outputs (which have two dense vector dimensions for them (Electra = 265 d) and (sgns = 300 d). Due to the limited time and resources, We started our experiments with Electra embeddings, and then we selected the best-performing architecture to employ them with Sgns embeddings. The semi-decoder is a Deep Neural Network with four hidden layers where the first hidden layer has 8 times the input, the second has 4 times, the third has 2 times, and the fourth has the same as the input that will be projected to the size of the output. The dropout mentioned rate is between every hidden layer before the activation.

Figure 1 illustrates the main idea behind our proposed framework. The process can be explained mathematically as follows:

$$F : x_{(input)} \rightarrow \widehat{y}_{(output)} \tag{1}$$

where the $x_{(input)} \in R^{512}$ and the $\widehat{y}_{(output)} \in R^{o}$ where $o \in \{256, 300\}$ the dimensions of the two types of outputs (electra and sgns). F is a neural network with 4 hidden layers (this is the defult design while we also trained two networks with 2 hidden layers). x is the input vectors and $\hat{y}$ is the predicted vector (d 256 or 300)

$$E(t) = x \tag{2}$$

E is a function representing the encoder model and t is the tokens, where E will do the following:

1. tokenize the text
2. feed the encoder the tokens IDs
3. output $\sum_{i}^{i=n}(t_i)$ (max polling where n is the maximum number of tokens and if less it will be padded

$$H_{i}^{512\times 512*m} \tag{3}$$

H is the size of hidden layer.

$$m \in \{8, 4, 2, 1\} \tag{4}$$

The x_{input} is output of a pre-trained encoder model called "SBERT" as follow:

Then the semi-Decoder:

$$D(x) = \widehat{y} \tag{5}$$

D is a function representing the semi-decoder model

then the loss function $L(\widehat{y}, y)$ which is $MSE(\widehat{y}, y)$ will update the weights of the network

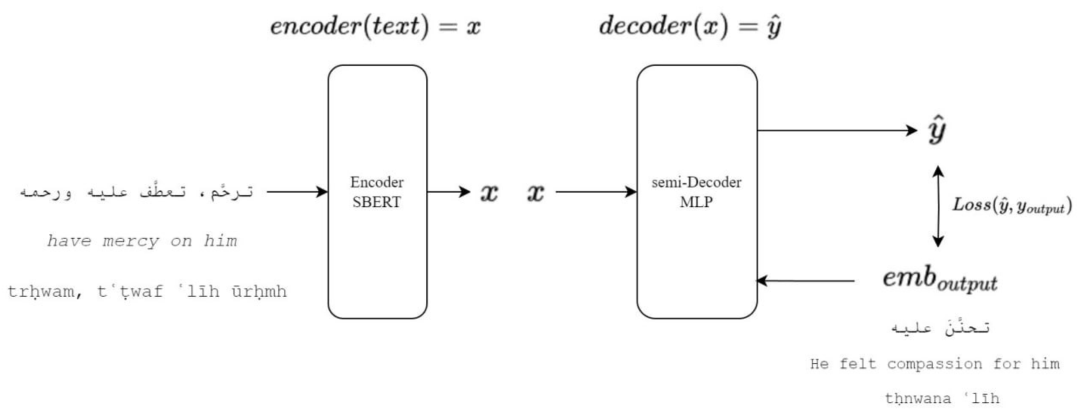

Figure 1: An example of a word's definition as input and a target word as output to show the overview of our proposed framework.

4 Result

We began our initial experiments with the development dataset. Afterward, we selected the most efficient method to produce predictions for the test dataset. Our evaluation relied on the official evaluation metric supported by the event organizers: Mean Square Error (MSE), Cosine similarity, and Rank, which evaluates the model's ability to order predictions in relation to actual values. The primary evaluation metric is Rank. If models yield similar results based on this metric, the mean squared error (MSE) is then employed as a secondary measure. In cases where further differentiation is required, cosine similarity serves as the tertiary metric.

To clarify, we experimented with seven different architecture setups to train the model, taking into account the number of embedding layers and drop-off. We chose the best model on Electra (Model NO. 3) to apply them to Sgns, in addition to the baseline model (Model NO. 1). Table 2 presents the results on the test dataset. It can be seen that Model number 3 has the highest performance (Rank = 28.05%) followed by the models 7 and 4. As for Sgns embeddings, model 2 achieved the best result with Rank of 30.78%.

The concept behind our approach is modifying the standard encoder-decoder architecture by truncating its latter section, which we have called the 'semi-decoder'. Due to the extensive scale of our model, an epoch range of 100-300 was inadequate for training. When the epochs exceeded 2000, overfitting issues emerged with our test data. This observation led us to conclude that the optimal epoch range is between 1000 and 2000. Specifically, the 2000-epoch mark resulted in a 'semi-overfitting' situation that delivered the most promising outcomes.

Model No.	MSE	COS SIM	Rank
		Electra	
1	18.89%	54.83%	50.00%
2	26.59%	21.91%	50.01%
3	23.56%	51.94%	**28.05%**
4	**17.03%**	**59.08%**	33.31%
5	17.85%	55.57%	48.15%
6	74.20%	-7.98%	37.97%
7	32.33%	46.07%	28.92%
		Sgns	
1	6.59%	21.90%	50.01%
2	**6.50%**	**39.36%**	**30.78%**

Table 2: Performance results for different models on the test set using three evaluation metrics.

4.1 Error Analysis

During the training process, a notable range of effective epochs emerged, spanning from 300 to 2000, wherein discernible patterns were successfully learned. Preceding this pivotal interval, the model's proficiency in capturing intricate patterns appeared limited. However, the subsequent epochs saw an escalated tendency towards overfitting. The employment of the GELU activation function exhibited superior performance. Conversely, the ReLU activation function demonstrated commendable potential for generalization, specifically in contexts characterized by diverse conditions ("sgns"). Nonetheless, for ranking tasks, its efficacy appeared akin to a stochastic outcome. Conversely, the Leaky ReLU activation function exhibited a subdued impact, potentially owing to the specificity of the problem domain. Notably, the application of dropout regularization yielded moderate influence on the model's performance. The chosen model architecture, designed to encapsulate definitions, demonstrated inherent promise, warranting a finer calibration to further explore the nuances of the Arabic language.

5 Conclusion

Our methodology encompasses two fundamental stages. Initially, we encode the word definitions, translating them into multidimensional vector representations. Subsequently, we subject these encoded vectors to training via the Simi-Decoder model to address our designated task. Our system secured a 2nd place based on Rank metric for both embeddings (Electra and Sgns).

Future work could involve collecting more data for training or validation, or providing the service online for public access. Improving our model's performance might be achieved by adopting the BERT or transformer model for training, known for efficient parallel processing and capturing long-term dependencies.

References

Aarchi Agrawal, KS Ashin Shanly, Kavita Vaishnaw, and Mayank Singh. 2021. Reverse dictionary using an improved cbow model. In *Proceedings of the 3rd ACM India Joint International Conference on Data Science & Management of Data (8th ACM IKDD CODS & 26th COMAD)*, pages 420–420.

Nihed Bendahman, Julien Breton, Lina Nicolaieff, Mokhtar Boumedyen Billami, Christophe Bortolaso, and Youssef Miloudi. 2022. Bl. research at semeval-2022 task 1: Deep networks for reverse dictionary using embeddings and lstm autoencoders. In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*, pages 94–100.

Pinzhen Chen and Zheng Zhao. 2022. A unified model for reverse dictionary and definition modelling. *AACL-IJCNLP 2022*, page 8.

Michael A Hedderich, Andrew Yates, Dietrich Klakow, and Gerard de Melo. 2019. Using multi-sense vector embeddings for reverse dictionaries. In *Proceedings of the 13th International Conference on Computational Semantics-Long Papers*, pages 247–258.

Arman Malekzadeh, Amin Gheibi, and Ali Mohades. 2021. Predict: persian reverse dictionary. *arXiv preprint arXiv:2105.00309*.

Oscar Méndez, Hiram Calvo, and Marco A Moreno-Armendáriz. 2013. A reverse dictionary based on semantic analysis using wordnet. In *Advances in Artificial Intelligence and Its Applications: 12th Mexican International Conference on Artificial Intelligence, MICAI 2013, Mexico City, Mexico, November 24-30, 2013, Proceedings, Part I 12*, pages 275–285. Springer.

Mohammad Taher Pilehvar. 2019. On the importance of distinguishing word meaning representations: A case study on reverse dictionary mapping. 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 2151–2156.

Fanchao Qi, Lei Zhang, Yanhui Yang, Zhiyuan Liu, and Maosong Sun. 2020. Wantwords: An open-source online reverse dictionary system. *EMNLP 2020*, page 175.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2020. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Sushrut Thorat and Varad Choudhari. 2016. Implementing a reverse dictionary, based on word definitions, using a node-graph architecture. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2797–2806.

Lei Zhang, Fanchao Qi, Zhiyuan Liu, Yasheng Wang, Qun Liu, and Maosong Sun. 2020. Multi-channel reverse dictionary model. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 312–319.