applied sciences

MDPI

Article

Arabic Gloss WSD Using BERT

Mohammed El-Razzaz, Mohamed Waleed Fakhr * and Fahima A. Maghraby

College of Computing and Information Technology, Arab Academy for Science, Technology, and Maritime Transport, Cairo P.O. Box 2033, Egypt; mohammed.elrzzaz@gmail.com (M.E.-R.); fahima@aast.edu (F.A.M.)
* Correspondence: waleedf@aast.edu

Abstract: Word Sense Disambiguation (WSD) aims to predict the correct *sense* of a word given its context. This problem is of extreme importance in Arabic, as written words can be highly ambiguous; 43% of diacritized words have multiple interpretations and the percentage increases to 72% for non-diacritized words. Nevertheless, most Arabic written text does not have diacritical marks. Gloss-based WSD methods measure the semantic similarity or the overlap between the context of a target word that needs to be disambiguated and the dictionary definition of that word (gloss of the word). Arabic gloss WSD suffers from a lack of context-gloss datasets. In this paper, we present an Arabic gloss-based WSD technique. We utilize the celebrated Bidirectional Encoder Representation from Transformers (BERT) to build two models that can efficiently perform Arabic WSD. These models can be trained with few training samples since they utilize BERT models that were pretrained on a large Arabic corpus. Our experimental results show that our models outperform two of the most recent gloss-based WSDs when we test them against the same test data used to evaluate our model. Additionally, our model achieves an F1-score of 89% compared to the best-reported F1-score of 85% for knowledge-based Arabic WSD. Another contribution of this paper is introducing a context-gloss benchmark that may help to overcome the lack of a standardized benchmark for Arabic gloss-based WSD.

Keywords: WSD; BERT; Arabic; context gloss

check for updates

Citation: El-Razzaz, M.; Fakhr, M.W.; Maghraby, F.A. Arabic Gloss WSD Using BERT. *Appl. Sci.* **2021**, *11*, 2567. https://doi.org/10.3390/app11062567

Academic Editor: Andrea Prati

Received: 10 February 2021
Accepted: 8 March 2021
Published: 13 March 2021

1. Introduction

Words in spoken and written languages can have multiple meanings for each word and, thus, could be ambiguous. The intended meaning of a word (*sense*) depends strongly on its context. Word Sense Disambiguation (WSD) is the task of predicting the correct sense of a word that has multiple senses given its context [1]. Many Computational Linguistics' applications depend on the performance of WSD, such as Machine Translation, Text Summarization, Information Retrieval, and Question Answering [2]. The Arabic language is particularly challenging for WSD. It was shown in [3] that 43% of Arabic words that have diacritic marks are ambiguous. Moreover, that percentage increases to 72% when the diacritic marks are missing, which is the case for most written Arabic text [4]. This is compared to 32% for the English language [5].

We note here three main challenges in Arabic WSD. First, most written text does not have diacritic marks. As a result, the level of ambiguity for each word can increase. For example, the word "عَلَمْ" (meaning flag) and the word "عِلْمْ" (meaning science) are both written as " علم" . Secondly, even when diacritic marks are present, many words can have several possible part-of-speech (POS) tags that lead to different meanings. For example, the word "عَرَضْ" can have the POS tag *verb* (leading to the meaning of "showing something"). It can also have the POS tag *noun* (leading to the meaning of "impermanent"). Third, a word with both a POS tag and diacritic marks can still have several senses depending on its context. For example, the word "إِسْتَعْرَضَ" with POS tag *verb* has the meaning of "parade" in the context "استعرض نفسه" while having the meaning of "inspect" in the context

"استعرض الجند". In addition, Arabic is an agglutinative language that adds parts of speech to the stems or roots of words to give new meanings [6]. For instance, the word "بالشارع" (in the street) has a stem "شارع" (a street) and two prefixes, "بِ" (in) and "ال" (the), which is the definite article. This problem has been addressed in the preprocessing phase in our work, which will be discussed shortly.

Bidirectional Encoder Representation from Transformers (BERT) [7] is a neural language model based on transformer encoders [8]. Recent Arabic Computational Linguistics' studies show a large performance enhancement when pretrained BERT models are fine-tuned to perform many Arabic Natural Language Processing (NLP) tasks, for example, Question Answering (QA), Sentiment Analysis (SA), and Named Entity Recognition (NER) [9,10]. This performance enhancement stems from the fact that BERT can generate contextual word embeddings for each word in a context. These embeddings were shown to encode syntactic, semantic, and long-distance dependencies of the sentences [11], thus making them ideal for many NLP tasks.

Contextual embedding methods (e.g., BERT [7], Embeddings from Language Models (ELMO) [12], and Generative Pre-trained Transformer (GPT) [13]) learn sequence-level semantics by considering the sequence of all the words in the input sentence. In particular, BERT is trained to predict the masked word(s) of the input sentence; to do this, BERT learns *self-attention* to weigh the relationship between each word in the input sentence and the other words in the same sentence. Each word then has a vector that represents the relationship with other words in the input sentence. Those vectors are used to generate word embedding, so the generated embeddings for each word depends on the other words in a given sentence. This is unlike Glove [14] and Word2Vec [15] that represent the word using a fixed embedding, regardless of its context. These traditional models depend on the cooccurrence of these words in the hole corpus. If two words usually have similar words in different contexts, they will have a similar representation.

WSD is classified into three main categories: supervised WSD, unsupervised WSD, and knowledge-based WSD. Knowledge-based WSD methods usually utilize language resources such as knowledge-graphs and dictionaries to solve the WSD problem. Knowledge-based WSD can be further subcategorized into graph-based approaches and gloss-based approaches [16]. Graph-based approaches utilize language knowledge-graphs, such as WordNet, to represent the sense by its surrounding tokens in that graph. Gloss-based approaches measure the semantic similarity or the overlap between the context of the target word and the dictionary definition of that word (gloss of the word). Gloss-based methods require context-gloss datasets. In this paper, we explore utilizing pretrained BERT models to better represent gloss and context information in gloss-based Arabic WSDs. Fortunately, there are pretrained BERT models available for Arabic [9,10]. We utilized these pretrained models to build two gloss-based WSD models. The first model uses the pretrained BERT models as a feature extractor without fine-tuning BERT layers to generate a contextual word embedding of the target word in its context. We also used it to generate a sentence vector representation of the gloss sentence. These representations' vectors were then fed to a trainable dense layer to perform supervised WSD. In the second model, we fine-tuned BERT layers by training them with a *sentence pair classification* objective. Our experiments showed that we outperform many of the state-of-the-art Arabic knowledge-based WSDs. The evaluation of WSD models suffers from a lack of standard benchmark datasets [16]. Each author tests on a different and small (around 100 ambiguous words) dataset, which makes it difficult to compare results. We attempt to fix this issue by creating a publicly available benchmark dataset. We hope that such a dataset can help ease evaluating and comparing different WSD techniques.

The rest of the paper is organized as follows. In Section 2, we discuss the related work. Section 3 introduces the context-gloss benchmark. The background is presented in Section 4. Section 5 describes the proposed models. In Sections 6–8, we present our experimental setup, results, and discussion. Finally, we conclude our paper in Section 9.

2. Related Work

As mentioned previously, knowledge-based WSD is categorized into graph-based approaches and gloss-based approaches [16]. Graph-based approaches depend on language knowledge graphs such as WordNet to represent the sense by its surrounding tokens in that graph. For example, the authors in [17] introduced a *retrofit* algorithm that utilized Arabic WordNet along with *Glove* [14] and *Word2Vec* [15] to generate context-dependent word vectors that carry sense information. Finally, they used cosine similarity between the generated word vector and vectors generated for WordNet synsets of that word and selected the sense with the highest similarity score. In [18], the authors utilized Arabic WordNet (AWN) to map words to *concepts*. A concept was defined as a specific word sense. They further utilized the English WordNet to extend the Arabic one by employing Machine Translation for the missing words in AWN. A concept was then selected for a target word by comparing the context of the target word to neighbors of the concept in the extended WordNet. Concepts with high match to the context were selected as the correct sense. In [19], the authors used AWN to generate a list of words that can be potentially ambiguous. For each ambiguous word, they found Wikipedia articles that correspond to the different senses of those words. Those articles were then converted to real vectors by tf-idf [20]. They also generated a context vector by tf-idf of the word in its context. Finally, they used cosine similarity between the two vectors and selected the sense with the highest similarity score.

The second type of knowledge base approaches is gloss-based approaches. These methods measure the semantic similarity or the overlap between the context of the target word and the dictionary definition of that word (gloss of the word). The authors in [21] used the *Lesk* algorithm to measure the relatedness of the target word's context to its definition. They used AWN as a source for the word gloss. The authors of [22] combined unsupervised and knowledge-based approaches. They used a string matching algorithm to match the root of salient words that appear in many contexts of a particular sense with the gloss definition. The authors of [23] employed word2vec [15] to generate a vector representation for the target word's context sentence and the gloss sentence. Then, cosine similarity was used to match the gloss vector and the context vector. The authors in [24] used Rough Set Theory and semantic short text similarity to measure the semantic relatedness between the target word's context and multiple possible concepts (gloss). Recently, The authors in [25] used FLAIR [26], a character-level language model, to generate sentence representation vectors for both context and gloss sentences. The sentence's vector was calculated by taking the mean of its word vectors, causing a loss of information, especially when the sequence is long. Cosine similarity was then used to measure the similarity between the word context sentence and the gloss sentence.

3. Benchmark

Arabic gloss WSD suffers from the lack of context-gloss pair datasets [16]. Arabic Word Net (AWN), which is the only available gloss dataset, contains only word senses with gloss definitions but lacks context examples for these senses [27]. Due to this limitation in AWN, we created a new Arabic WSD benchmark. This dataset is extracted from the "Modern Standard Arabic Dictionary" [28]. To the best of our knowledge, our benchmark is the only available Arabic context-gloss pair dataset. It consists of 15,549 senses for 5347 unique words with an average of 3 senses for each word. Figure 1 shows the histogram of sense counts per word. We can see that most of the words (4000+) have 2 to 4 senses, that about 750 word are between 4–6 senses per word, and that the count decreases as the number of senses increases. Each record in the dataset is a tuple of three elements: a word sense, a context example, and a definition of that word sense. In Table 1, we present the statistics of the dataset. This dataset is available online (https://github.com/MElrazzaz/Arabic-word-sense-disambiguation-bench-mark.git, accessed on 10 March 2021). Table 2 shows examples of records in the benchmark's dataset. We believe that the benchmark can contribute to standardizing the evaluation of WSD models.

Table 1. The benchmark statistics.

# Words	# Senses	Average Sense per Word	Max Senses per Word
5347	15,549	3	40

Table 2. Benchmark example.

Word	Gloss	Context-Example
عرض (show)	عرض الموضوع له بسطه وطرحه ليطلعه عليه (Show the topic to him, simplify the topic to show it)	عرض خطة بحثه (Show his research plan)
عرض (honor)	ما يمدح ويذم من الإنسان نفسه وحسبه أو فيمن يلزمه أمره (What a person praises and defames in himself or in whom he is obligated to)	طعن في عرض فلان (Insult someone honor)

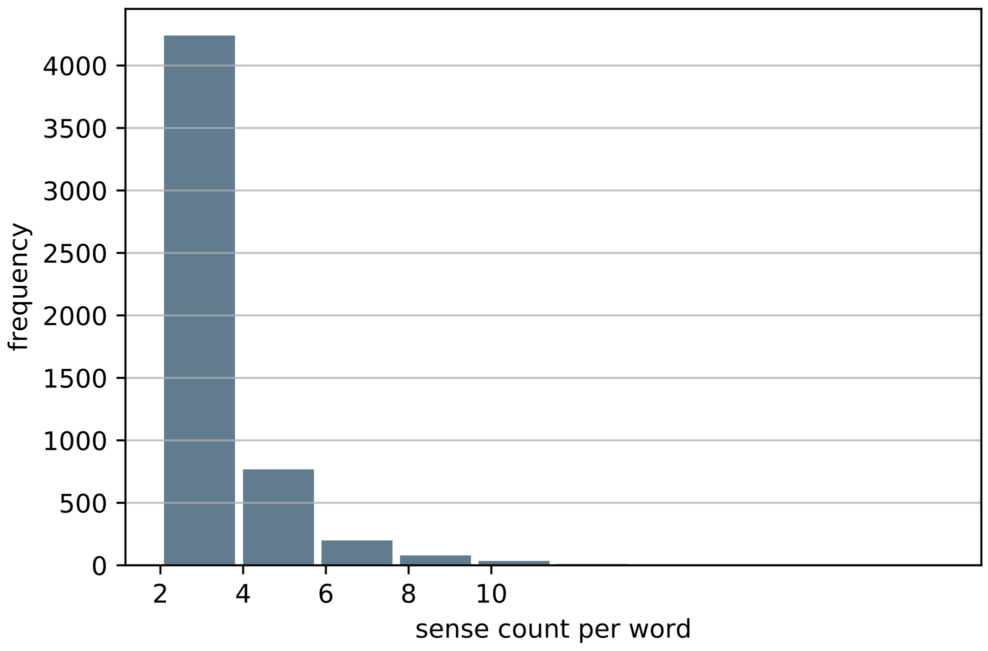

Figure 1. Histogram of benchmark senses per word.

4. Background

In this section, we review the background needed to introduce our approach. We begin by reviewing transformers [8], which are a substantial part of BERT [7]. We then review BERT as it is one of the building blocks of our approach.

4.1. Transformers

A transformer [8] is a celebrated Neural Network (NN) architecture that was introduced to approach translation tasks. Figure 2 depicts the transformer's main components. It consists of two components: an encoder and a decoder. The encoder is used to encode a source sentence into a sequence of real vectors. The decoder, on the other hand, has the goal of decoding a set of vectors into a target sentence. Both the encoder and the decoder consist of 6 blocks. Each of these blocks consists of two layers: the first layer is a *multi-head attention* layer, while the second is a fully connected layer. The layers are equipped with a residual connection that connects the input of a layer to the output of that layer. After each block, a Layer Norm [29] is applied.

Now, we elaborate more on the multi-head attention layer. The intuition behind this layer is that the sense of a given word in a sentence is usually affected by different words in the same sentence (self-attention) or the target sentence, with different degrees. This intuitive idea is implemented in the attention layer by associating three vectors: a key k, a query q, and a value v with each word. The degree of dependence between a given word w_s and a target word w_t is quantified by the dot product between the query of the word q_{w_s} and the key of the target word k_{w_t}. Finally, the word is represented by a weighted average of the values of target words, using $\langle q_{w_s}, k_{w_t} \rangle$ as weights. The weights are further normalized and a Softmax is applied to ensure that the weights sum to 1 before the weighted average is taken. To obtain a compact formula, keys of target words are put in rows of a matrix K, queries are arranged similarly into a matrix Q, while V denotes the concatenation of values of target words. Then, the layer output is realized using Equation (1)

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{(QK^T)}{\sqrt{d_k}}\right)V, \tag{1}$$

where d_k represents the dimensionality of the vector k.

As can be noticed in the above computation of the attention-based word representation, the position of a word in the target sequence is lost as a result of the weighted average over targeted words. This can be alleviated by adding a vector to each word representation that encodes the position of the given word. Such a process is called Positional Encoding (PE). The PE vector is achieved using Equation (2).

$$PE_{(\text{pos},k)} = \begin{cases} \sin\left(\frac{\text{pos}}{10^{4k/d_k}}\right), \text{ if } k \text{ is even}, \\ \cos\left(\frac{\text{pos}}{10^{4k/d_k}}\right), \text{ if } k \text{ is odd}, \end{cases} \tag{2}$$

where k is the index of the value calculated of for the positional vector and pos is the position of the word.

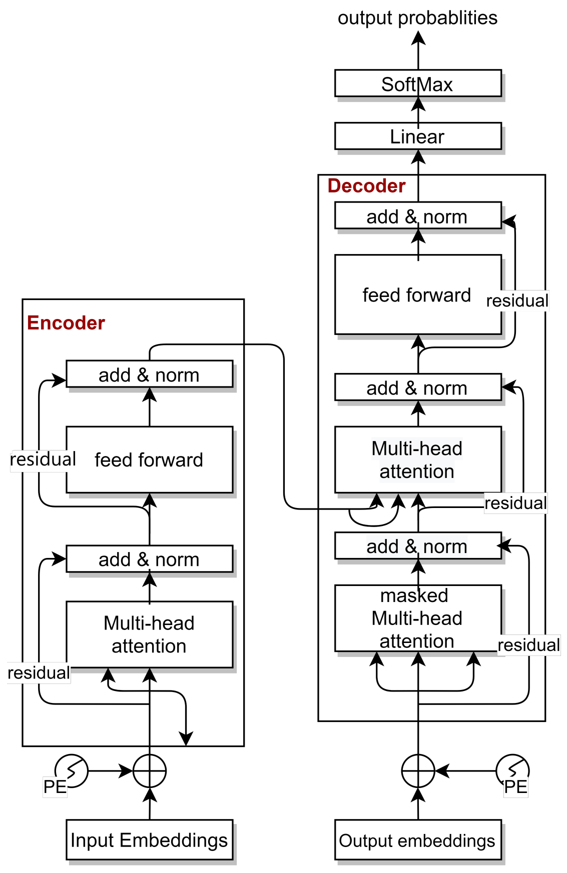

Figure 2. Transformer architecture, multi-head attention layer architecture, and scaled dot-product attention unit.

4.2. *Bidirectional Encoder Representation from Transformers*

BERT [7] is a language model that is based on the architecture of the transformers reviewed above. As can be seen in Figure 3, BERT's architecture consists of a stacked set of transformer encoders. A key aspect of BERT is that it is designed to be trained in an unsupervised manner with no particular task at hand. To facilitate training, two objectives were proposed to optimize BERT. (1) Given a sentence with a particular word, it is *masked* (replaced with a placeholder) and then such a word is predicted. (2) Given two sentences, say X and Y, the order of such sentences is predicted; that is, whether Y appears before X or vice versa is predicted. The model that learned in this manner is shown in [11] to encode phrase-level information in the lower layers, while in the middle layers, it encodes surface, syntactic, and semantic features. Moreover, the top layers of the model encode long-distance dependencies in the sentence.

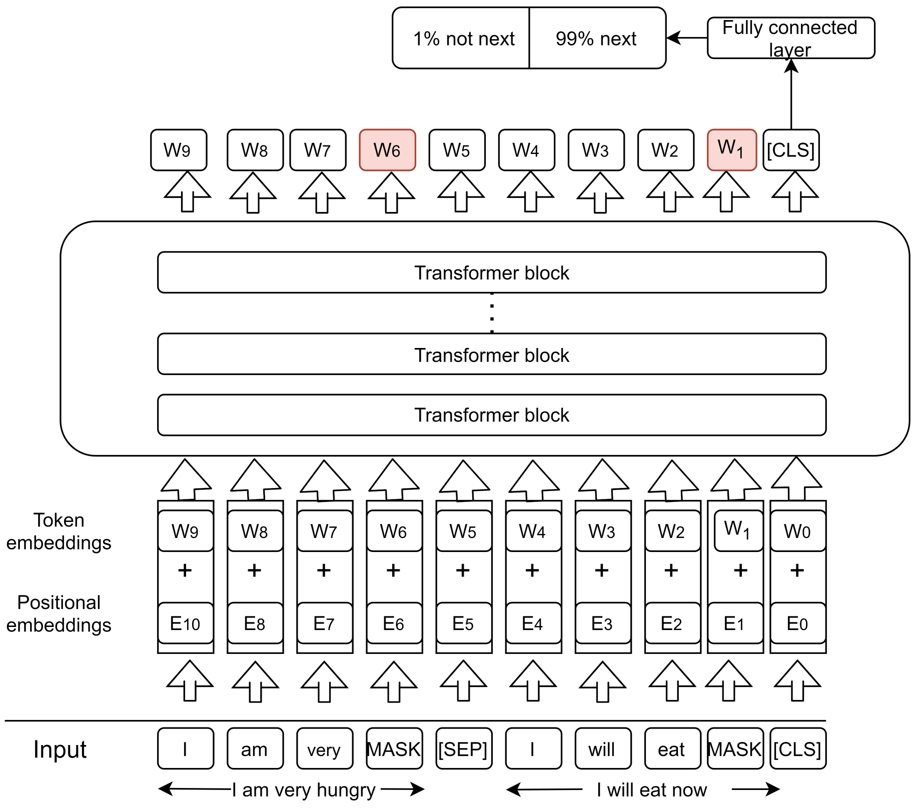

Figure 3. Bidirectional Encoder Representation from Transformers (BERT) training input and BERT training tasks: the first task is to predict masked words, and the second task makes the next sentence prediction.

4.3. Arabic Pretrained BERT Models

Recent studies on Arabic NLP showed that using BERT achieved state-of-the-art results on many NLP tasks. For instance, Question Answering (QA), Named Entity Recognition (NER), and Sentiment Analysis [9,10]. These studies trained two BERT models, namely AraBERTv2 [9] and ARBERT [10]. These models were trained on large Arabic corpora. The details of these corpora as well as the number parameters of each model are shown in Table 3. Since these models already contributed to the state-of-the-art results in many tasks, we use them as building blocks for our proposed models.

Table 3. Arabic pretrained models.

	Training Data			Vocab		Configs
Model	**Sentence**	**Size**	**No. of Tokens**	**Tokenization**	**Size**	**of Params**
AraBERTv2	200 M	77 GB	8.6 B	Sentence piece	60 K	of 135
ARBERT	Several (6 sources)	61 GB	6.5 B	Word piece	100 K	of 163

4.4. Transfer Learning with BERT

In this section, we discuss how to utilize BERT pretrained models to approach NLP tasks. There are three common strategies, as shown in Figure 4. The first strategy is to tune all layers, that is, to train the entire pretrained model with a new task while the pretrained model parameters are used as initializations. Second, we fine-tuned only some layers while keeping the weights of the rest of the network unchanged. Another strategy is to freeze the entire architecture while attaching a set of layers to the output of the pretrained model. During training, only the parameters of the new layers are allowed to change.

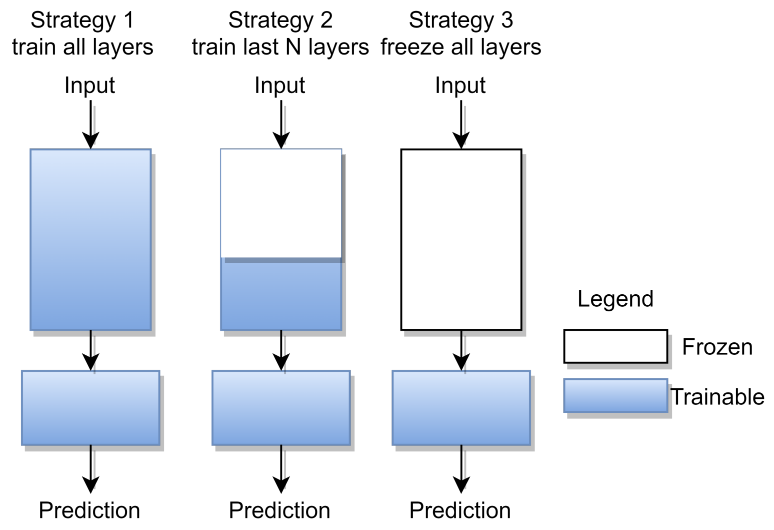

Figure 4. Transfer learning strategies.

5. Materials and Methods

In this section, we present our proposed approach to gloss WSD. As stated above, we utilized a pretrained BERT to design our models. Our first model uses BERT as a feature extractor, that is, the parameters of BERT layers are not retrained. On the other hand, the second model fine-tunes BERT layers by training it with a *sentence pair classification* objective. We experimented with both AraBERTv2 [9] and ARBERT [9] as our pretrained BERT. In the rest of this section, we present the details of our proposed models. The code and the data needed to replicate our proposed method are available on GitHub (https://github.com/MElrazzaz/Arabic-word-sense-disambiguation-bench-mark.git, accessed on 10 March 2021).

5.1. Model I

Model I uses the pretrained BERT model as a feature extractor. The intuition here is that, since we only have a few samples, fine-tuning the model can be affected by the noise in those samples. Therefore, training only our added layers will reduce the complexity of the model and make efficient use of the training sample size.

Figure 5 shows the Model I architecture. The architecture consists of the concatenation of two BERT outputs. These two outputs are the features extracted from a pretrained BERT. We then applied a fully connected layer on top of those concatenated outputs. Finally, a softmax layer was attached on top of the fully connected layer to provide means of training that layer. The two feature vectors were extracted in a specific manner to represent the sense of the word as defined in the gloss and its sense as defined by its context in a given example. To that end, we fed the word gloss to our pretrained BERT model. We then used the embedding of the '[CLS]' token as the feature vector representing the sense of the word from the gloss perspective. The second feature vector, on the other hand, was generated by feeding the pretrained BERT with an example sentence of this word sense. The extracted features, in this case, were, however, the embedding of that word averaged over the last N BERT layer(s). We then concatenated those feature vectors and applied a fully connected layer.

The model was trained with the objective of classifying whether the sense in the example context of the target word matches the sense defined by the gloss of the target word. To train such an objective, we created a dataset with pairs of gloss and examples. A record with a positive class has both the gloss and the example matching, while a negative record has the example and gloss not matching. For an example of those records, see Table 4. Our novel dataset consists of 31 K records, 15,549 positive examples, and 15,549 negative examples.

Table 4. The training data for Model I.

Word	Gloss	Context-Example	Label
عرض (show)	عرض الموضوع له بسطه وطرحه ليطلعه عليه (Show the topic to him, simplify the topic to show it)	عرض خطة بحثه (Show his research plan)	1
عرض (honor)	عرض الموضوع له بسطه وطرحه ليطلعه عليه (Show the topic to him, simplify the topic to show it)	طعن في عرض فلان (Insult someone honor)	0

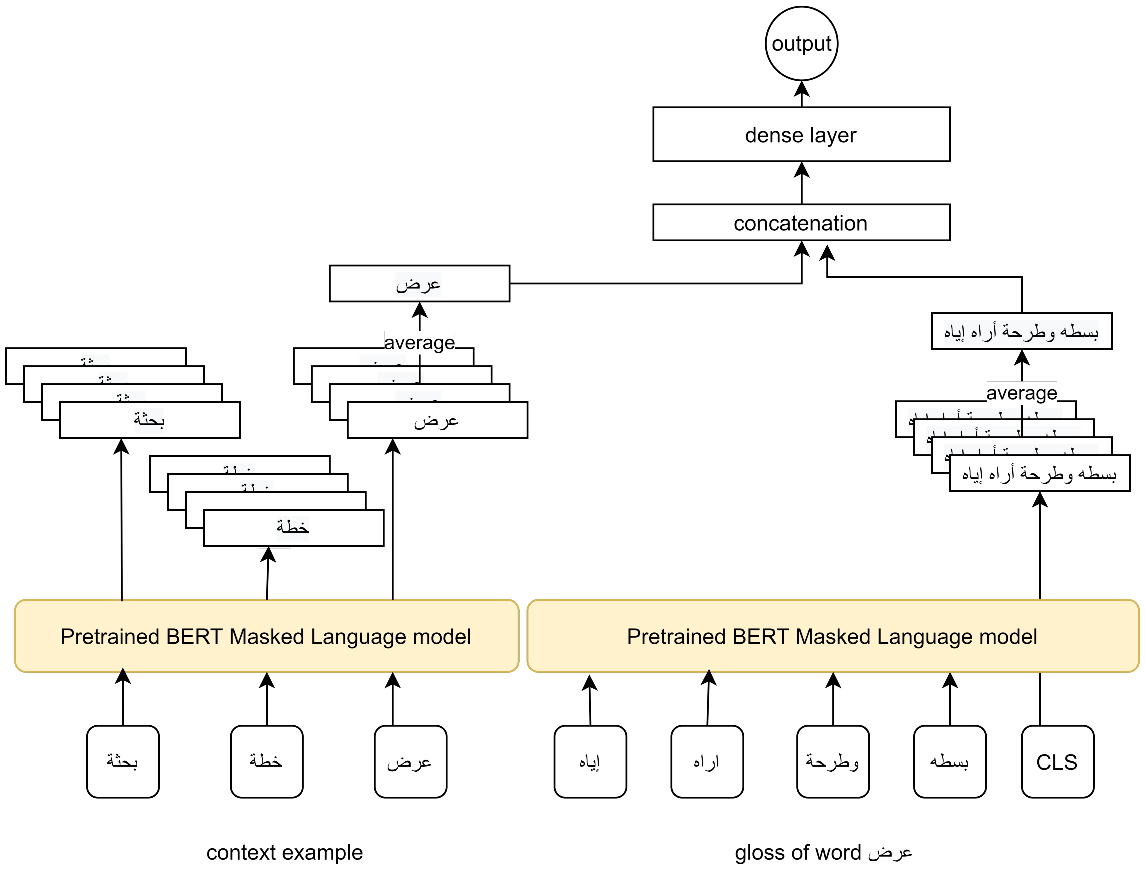

Figure 5. Model I architecture.

5.2. Model II

Unlike Model I, in Model II, we fine-tuned all BERT layers. Again, the training objective was to perform a sentence pair classification task. In this model, however, only one BERT architecture was used. The input to the BERT model, in this case, was two sentences separated by a separator token '[SEP]'. The first sentence was a context example, while the other sentence was the gloss of the target word. A sequence classification layer with a Gaussian Error Linear Units (GELU) activation function was added on top of the BERT layers to classify whether the context sentence and the gloss definition were related. Examples of training set record are show in Table 5. For each context-gloss pair, a '[SEP]' was added between the two sentences and a '[CLS]' token was added at the beginning of the two sentences. The classification token '[CLS]' was used to present an encoding on the total sequence, and therefore, the classification layer was attached on top of the '[CLS]' embedding. Figure 6 shows Model II architecture.

Table 5. Model II training data.

Context-Definition Pairs of the Target Word	Label	Word
[CLS] 'gloss sentence' [SEP] 'example sentence'	1	the word
[CLS] 'gloss sentence' [SEP] 'wrong example sentence'	0	the word

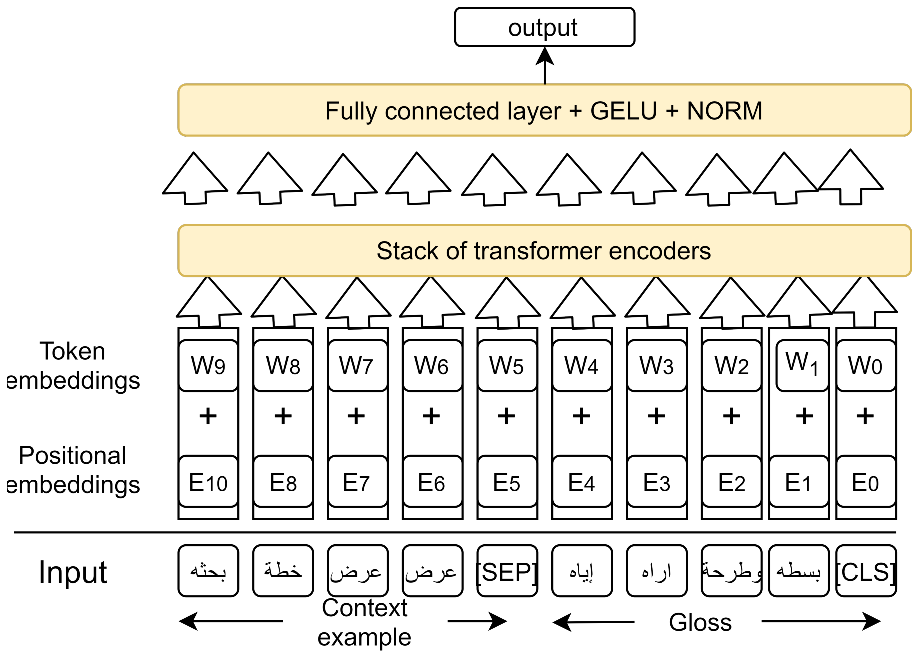
output
Fully connected layer + GELU + NORM
Stack of transformer encoders
Token embeddings
Positional embeddings
W9 W8 W7 W6 W5 W4 W3 W2 W1 W0
E10 E8 E7 E6 E5 E4 E3 E2 E1 E0
Input
بحثه خطة عرض عرض [SEP] إياه اراه وطرحة بسطه [CLS]
Context example
Gloss

Figure 6. Model II architecture.

6. Experiments

In this section, we present the experimental setup for building and evaluating our proposed Arabic WSD models.

6.1. Evaluation Dataset

To evaluate the performance of the proposed models, the benchmark dataset was divided into 60%, 20%, and 20% for training, validation, and testing, respectively. The details of the test data are shown in Table 6.

Table 6. Test data statistics.

No. Words	No. Senses	No. Positive Examples	No. Negative Examples
3601	6246	1800	1801

We preprocessed both the word sense definition and the context examples to match the input expected by the BERT models. Diacritic marks were removed from the input sentences. Moreover, we applied tokenization to the input sentences. We used the Farasa tokenizer [30].

6.2. Evaluation Metrics

The goal of the model was to find the correct word sense from a given example. We report three evaluation metrics, namely, precision, recall, and F1-score. These metrics are defined as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}},$$
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$
$$\text{F1-Score} = \frac{2 * (\text{Precision} * \text{Recall})}{\text{Precision} + \text{Recall}},$$

where TP denotes the true positive count, that is, the number of correctly classified sentence pairs when the context example matches the word sense definition, while FP is the false positive count, that is, the number of falsely classified sentence pairs. TN is the number of correctly classified sentence pairs when the word sense definition does not match its context, while FN reflects the times when the classification is incorrect.

6.3. Training Configuration

We now present the details of our training procedures. For Model I, we trained the model with 20 epochs. The features extracted from BERT were the average of the last 4 layers. We used the Adam [31] optimization method with a batch size of 8 and a learning rate of 0.00002. The learning objective was to minimize the binary cross-entropy loss. These learning parameters are summarized in Table 7. Model II was trained with the same parameters. However, we optimized also the BERT layers. Moreover, we appended the targeted words to the sequence of input to drive the model to learn to disambiguate this specific word from the full sequence.

Table 7. The hyperparameter configurations for Model I and Model II.

Model	Optimizer	Learning Rate	Epochs	Layers to Represent Target Word
Model I	Adam	0.00002	20	last 4
Model II	Adam	0.00002	2	-

7. Results

In this section, we present our experimental results. Table 8 summarizes the results of Model I and Model II on the test set as measured by precision, recall, and the F1-score. For Model I and Model II, we experimented with three different configurations as follows:

- AraBERTv2 as a pretrained BERT model without tokenizing the input sentences.
- AraBERTv2 as a pretrained BERT model with tokenizing the input sentences.
- ArBERT as a pretrained BERT model without tokenizing the input sentences.

As can be seen in Table 8, both configurations that use AraBERTv2 seem to outperform ArBERT as the pretained BERT model. This holds for both Model I and Model II. For the tokenization configuration, Model II enhances the precision from 92% to 96% while decreasing the recall from 87% to 67%, which affects the F1-score and hence decreases from 89% to 79%. On the other hand, Model I is adversely affected by tokenization, leading to a decrease in precision from 69% to 67% and in F1-score from 74% to 72%. We can also see that Model II always outperforms Model I.

Table 9 compares our proposed models against other embedding based models that depend on representing gloss and the context sentences in the vector space such as in our model. We compare our proposed models with two models, Laatar [25] and Laatar [23] which are the most recent Arabic WSD works. The test data of the two models are not available; thus, in order to make a fair comparison, we redeveloped their models and tested them on our benchmark. Laatar [25] used FLAIR [26] to generate word embedding. FLAIR is a pretrained character level language model. Laatar [25] represents each sentence by the mean of its words vectors. The authors in [23] used Word2Vec [15] language model to represent the words of the sentence in the vector space and then they represented the sentence by adding its words vector representation. After calculating the sentences' vectors, both models measure the cosine similarity between the context sentence vector and the gloss sentences vector and choose the gloss with the highest similarity score. Table 9 shows that our models outperform other embedding-based models in terms of precision, recall and F1 score. Table 9 also shows the original paper results of other models according to their test set. We performed McNemar's significance test [32] between Model II and Laatar [25] and between Model II and Laatar [23]; we find that Model II is significantly better than Laatar [25] and Laatar [23] and that the *p*-value is less than 0.01.

Table 8. Model I versus Model II.

Model	BERT Model	Tokenization	Precision	Recall	F1
Model I	AraBERTv2	No	69%	78%	74%
Model I	AraBERTv2	Yes	67%	78%	72%
Model I	ARBERT	No	50%	90%	64%
Model II	AraBERTv2	No	92%	**87%**	**89%**
Model II	AraBERTv2	Yes	**96%**	67%	79%
Model II	ARBERT	No	89%	73%	83%

In Table 10, we present the performance in comparison to other Arabic Knowledge based WSDs.

As shown in Table 10, Model II with non-tokenized inputs outperforms all other Arabic knowledge-based WSD as measured by F1-score.

Table 9. Embedding-based models vs. our models.

Model	Embedding Model	Result Source	Precision	Recall	F1
Model I	AraBERTv2	Our benchmark	69%	78%	74%
Model I	AraBERTv2	Our Benchmark	67%	78%	72%
Model I	ArBERT	Our benchmark	50%	90%	64%
Model II	AraBERTv2	our benchmark	92%	**87%**	**89%**
Model II	AraBERTv2	Our benchmark	**96%**	67%	79%
Model II	ARBERT	Our benchmark	89%	73%	83%
Laatar [25]	FLAIR	Our benchmark	63%	63%	63%
Laatar [25]	FLAIR	Orginal paper result (100 words)	67%	67%	67%
Laatar [23]	word2vec	Our benchmaek	45%	45%	45%
Laatar [23]	word2vec	Original paper test (100 words)	78%	-	-

Table 10. Model II versus other knowledge-based models.

Model	Testset	Precision	Recall	F1
Model II-AraBERTv2	3601 ambiguous words	92%	**87%**	**89%**
Model II-AraBERTv2-tokenized	3601 ambiguous words	**96%**	67%	79%
Model II-ArBERT	3601 ambiguous words	89%	73%	80%
Hadnie [18]	AWN	73%	-	-
Zouaghi [21]	50 ambiguous words	59%	-	-
Menai [33]	-	79%	63%	70%
Bekkali [24]	1217 text	-	-	85%
Bouhriz [34]	manually collected text	74%	-	-

8. Discussion

The results above show that our approach outperforms two of the recent gloss-based models and that the reported results are super-passed previously reported results of knowledge-based WSD. We believe that the reason behind this improvement lies in the use of a pretrained BERT. Indeed, BERT models are trained on a very large dataset to perform masked language modeling and, next, sentence prediction tasks. In order to perform well on these tasks, it has been shown in [11] that BERT models learned both syntactic and semantic features at the middle layers while, at the top layers, BERT learned long-distance dependency information. Therefore, using these pretrained models resulted in transferring this knowledge to Arabic WSD and hence improved the performance and decreased the amount of data needed for these models to learn. Moreover, we exploited BERT's ability to perform well in sentence pair classification tasks by posing the WSD as a sentence pair classification problem. Finally, we used our new benchmark to perform supervised fine-tuning; we believe that this is one of the main drivers to reach these results.

9. Conclusions

In this paper, we introduced the first Arabic context-gloss pairs benchmark. We believe that it can help standardize the evaluation of models in Arabic WSD. We then introduced two Arabic WSD models that benefited from pretrained BERT models. The first model used BERT as a feature extractor without fine-tuning BERT layers to generate two feature vectors. The first vector is a word-contextualized word embedding, while the second vector is a sense representation vector. These two vectors served as inputs to a fully connected layer with a sigmoid activation function to predict whether the word in the context matches the sense in the definition. In the second model, we posed the WSD task as a sentence pair classification task and fine-tuned pretrained BERT using the sequence classification objective. We experimentally showed that our models outperform recent gloss-based WSD systems and that the reported results passed other reported knowledge-based WSD results. Furthermore, we discovered that fine-tuning the model was crucial for achieving good performance.

Author Contributions: Conceptualization, M.E.-R., F.A.M., and M.W.F.; methodology, M.E.-R.; software, M.E.-R.; validation, M.E.-R.; formal analysis, M.E.-R., F.A.M., and M.W.F.; investigation, M.E.-R., F.A.M., and M.W.F.; resources, M.E.-R.; data curation, M.E.-R.; writing—original draft preparation, M.E.-R.; writing—review and editing, F.A.M. and M.W.F.; visualization, M.E.-R., F.A.M., and M.W.F.; supervision, F.A.M. and M.W.F.; project administration, F.A.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data that were used in this paper can be found at https://github.com/MElrazzaz/Arabic-word-sense-disambiguation-bench-mark.git accessed on 10 March 2021.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Navigli, R. Word Sense Disambiguation: A Survey. *ACM Comput. Surv.* **2009**, *41*, 1–69. [CrossRef]
2. Nancy, I.; Jean, V. Word sense disambiguation: The state of the art. *Comput. Linguist.* **1998**, *24*, 1–40.
3. Debili, F.; Achour, H.; Souissi, E. La langue arabe et l'ordinateur: De l'étiquetage grammatical à la voyellation automatique. *Correspondances* **2002**, *71*, 10–28.
4. Alqahtani, S.; Aldarmaki, H.; Diab, M. Homograph disambiguation through selective diacritic restoration. *arXiv* **2019**, arXiv:1912.04479.
5. Britton, B.K. Lexical ambiguity of words used in English text. *Behav. Res. Methods Inst.* **1978**, *10*, 1–7. [CrossRef]
6. Farghaly, A.; Shaalan, K. Arabic natural language processing: Challenges and solutions. *ACM Trans. Asian Lang. Inf. Process. (TALIP)* **2009**, *8*, 1–22. [CrossRef]
7. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. 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, MN, USA, June 2019; pp. 4171–4186.
8. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. *Attention Is All You Need*; Curran Associates Inc.: Red Hook, NY, USA, 2017; pp. 5998–6008.
9. Antoun, W.; Baly, F.; Hajj, H. AraBERT: Transformer-based model for Arabic language understanding. *arXiv* **2020**, arXiv:2003.00104.

10. Abdul-Mageed, M.; Elmadany, A.; Nagoudi, E.M.B. ARBERT & MARBERT: Deep Bidirectional Transformers for Arabic. *arXiv* **2020**, arXiv:2101.01785.
11. Jawahar, G.; Sagot, B.; Seddah, D. What does BERT learn about the structure of language? In Proceedings of the ACL 2019—57th Annual Meeting of the Association for Computational Linguistics, Minneapolis, MN, USA, June 2019.
12. Peters, M.E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; Zettlemoyer, L. Deep contextualized word representations. *arXiv* **2018**, arXiv:1802.05365.
13. Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I. Improving Language Understanding by Generative Pre-Training, Vancouver, Canada. 2018 October. Available online: https://www.cs.ubc.ca/~amuham01/LING530/papers/radford2018improving.pdf (accessed on 10 March 2021).
14. Pennington, J.; Socher, R.; Manning, C.D. Glove: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1532–1543.
15. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781.
16. Elayeb, B. Arabic word sense disambiguation: A review. *Artif. Intell. Rev.* **2019**, *52*, 2475–2532. [CrossRef]
17. Alkhatlan, A.; Kalita, J.; Alhaddad, A. Word sense disambiguation for arabic exploiting arabic wordnet and word embedding. *Procedia Comput. Sci.* **2018**, *142*, 50–60. [CrossRef]
18. Hadni, M.; Ouatik, S.E.A.; Lachkar, A. Word sense disambiguation for Arabic text categorization. *Int. Arab J. Inf. Technol.* **2016**, *13*, 215–222.
19. Alian, M.; Awajan, A.; Al-Kouz, A. Arabic word sense disambiguation using Wikipedia. *Int. J. Comput. Inf. Sci.* **2016**, *12*, 61. [CrossRef]
20. Aizawa, A. An information-theoretic perspective of tf–idf measures. *Inf. Process. Manag.* **2003**, *39*, 45–65. [CrossRef]
21. Zouaghi, A.; Merhbene, L.; Zrigui, M. Word Sense disambiguation for Arabic language using the variants of the Lesk algorithm. In Proceedings of the International Conference on Artificial Intelligence (ICAI), the Steering Committee of the World Congress in Computer Science, Vancouver, BC, Canada, July 2011.
22. Zouaghi, A.; Merhbene, L.; Zrigui, M. A hybrid approach for arabic word sense disambiguation. *Int. J. Comput. Process. Lang.* **2012**, *24*, 133–151. [CrossRef]
23. Laatar, R.; Aloulou, C.; Belghuith, L.H. Word2vec for Arabic word sense disambiguation. In *International Conference on Applications of Natural Language to Information Systems*; Springer: New York, NY, USA, 2018; pp. 308–311.
24. Bekkali, M.; Lachkar, A. Context-based Arabic Word Sense Disambiguation using Short Text Similarity Measure. In Proceedings of the 12th International Conference on Intelligent Systems: Theories and Applications, October 2018; pp. 1–6. Available online: https://dl.acm.org/doi/abs/10.1145/3289402.3289521 (accessed on 10 March 2021).
25. Laatar, R.; Aloulou, C.; Belguith, L.H. Disambiguating Arabic Words According to Their Historical Appearance in the Document Based on Recurrent Neural Networks. *ACM Trans. Asian Low-Resourc. Lang. Inf. Process. (TALLIP)* **2020**, *19*, 1–16. [CrossRef]
26. Akbik, A.; Blythe, D.; Vollgraf, R. Contextual string embeddings for sequence labeling. In Proceedings of the 27th International Conference on Computational Linguistics, Santa Fe, NM, USA, 20–25 August 2018; pp. 1638–1649.
27. Black, W.; Elkateb, S.; Rodriguez, H.; Alkhalifa, M.; Vossen, P.; Pease, A.; Fellbaum, C. Introducing the Arabic wordnet project. In Proceedings of the Third International WordNet Conference, Citeseer, UK, 22–26 January 2006; pp. 295–300.
28. Omar Ahmed Mokhtar. *Modern Standard Arabic Dictionary*; Omar: Alam ALkotob, Egypt, 2008.
29. Ba, J.L.; Kiros, J.R.; Hinton, G.E. Layer Normalization. *Stat* **2016**, *1050*, 21.
30. Abdelali, A.; Darwish, K.; Durrani, N.; Mubarak, H. 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, San Diego, CA, USA, June 2016; pp. 11–16.
31. Kingma, D.; Ba, L. *Adam: A Method for Stochastic Optimization*; The International Conference on Learning Representations (ICLR): San Diego, CA, USA, 2015.
32. Lachenbruch, P.A. McNemar test. *Wiley StatsRef Stat. Ref. Online* **2014**, *5*.
33. Menai, M.E.B. Word sense disambiguation using evolutionary algorithms–Application to Arabic language. *Comput. Hum. Behav.* **2014**, *41*, 92–103. [CrossRef]
34. Bouhriz, N.; Benabbou, F.; Lahmar, E.B. Word sense disambiguation approach for Arabic text. *Int. J. Adv. Comput. Sci. Appl.* **2016**, *7*, 381–385. [CrossRef]