Leveraging Knowledge-Based Features with Multilevel Attention Mechanisms for Short Arabic Text Classification

Iyad AlAgha

Faculty of Information Technology, the Islamic University of Gaza, Gaza, Palestine, P.O. Box 108
Corresponding author: Iyad AlAgha (e-mail: ialagha@iugaza.edu.ps).

ABSTRACT With the wide spread of short texts through social media platforms, there has become a growing need for effective methods for short-text classification. However, short-text classification has always been challenging due to the ambiguity and the data sparsity of the short text. A common solution is to enrich the short text with additional semantic features extracted from external knowledge, such as Wikipedia, to help the classifier better decide on the correct class. Most existing works, however, focused on text written in English and benefited from the existence of entity-linking tools based on English-based knowledge bases. When it comes to the Arabic language, the exploitation of external knowledge to support the classification of Arabic short text has not been widely explored. This work presents an approach for the classification of short Arabic text that exploits both the Wikipedia-based features and the attention mechanism for effective classification. First, Wikipedia entities mentioned in the short text are identified. Then, Wikipedia categories associated with the identified entities are retrieved and filtered to retain only the most relevant categories. A deep learning model with multiple attention mechanisms is then used to encode the short text and the associated category set. Finally, the short text and category representations are combined together to be fed into the classification layer. The use of the attentive model with category filtering leads to highlighting the most important features while reducing the effect of improper features. Finally, the proposed model is evaluated by comparing it with several deep learning models.

INDEX TERMS short text classification, Arabic, deep learning, attention mechanism, Wikipedia

I. INTRODUCTION

With the proliferation of social media content, huge volumes of short texts are increasingly being shared and used for different purposes. Therefore, there has become an emerging need for classification techniques that can effectively classify these short texts. Short text classification is widely applied in information retrieval [1], sentiment analysis [2], and recommendation systems [3]. However, short texts have unique characteristics that pose challenges compared to long text classification. Unlike the long texts, short texts are usually noisier, less topic-specific, and lacks contextual features that enable for a good similarity measure [4]. In addition, short texts are usually rather ambiguous because they contain polysemes and typos.

Conventional machine learning techniques do not often achieve desirable results in short-text classification tasks [5]. This is due to the data sparsity of the short text that leads to a sparse vector space. These techniques also ignore the potential semantic correlations between words when classifying the short texts [6]. More recently, deep learning techniques drawn much attentions in the field of natural language processing (NLP). They have surpassed classical machine learning approaches in various text classification tasks.

Existing methods from the literature have often tackled the problem of short text classification by extending the short text with information retrieved from search engines [7, 8] or external knowledge bases such as Wikipedia, and WordNet [10-13]. The intention is to enrich the short text with related information that provides extra features to support the classification process. Most existing works, however, focused on text written in English and benefited from the existence of entity-linking tools to annotate words in the short text with relevant entities from external knowledge [14-16]. However, when it comes to the Arabic language, little work has been done to exploit external knowledge bases to support the classification of Arabic short text. This is mainly due to the
lack of comprehensive knowledge resources in Arabic such as those available in English, besides the lack of tools that facilitate access and information retrieval from external knowledge such as the Arabic Wikipedia. This work presents a deep learning approach for short Arabic text classification that exploits both knowledge-based features extracted from Wikipedia and the attention mechanism. It integrates both the contextual features of the short text and the conceptual features of Wikipedia categories into deep neural networks to improve the classification results. It also employs the attention mechanism to improve reasoning by assigning relative importance to the words in the short text and the extracted Wikipedia categories. The method starts by performing entity linking on the short text in order to identify Wikipedia entities mentioned in the text. Subsequently, Wikipedia categories associated with identified entities are extracted, filtered and used to extend the features of the short text. A deep learning model with multiple attention mechanisms is used to encode the short text and the associated category set. Finally, the short text representation is combined with the knowledge representation to be fed into the classification layer.

Fig. 1 depicts an example that illustrates the process of extending the short text with information retrieved from Wikipedia. The words “Family”, “Facebook” and “AI” in the given sentence are first mapped to corresponding Wikipedia entities. Then, the Wikipedia categories associated with these entities are used to extend the short text. Only the last entity should be considered because it is the only entity relevant to the given text, while the other entities should be ignored to avoid incorporating incorrect information. The second problem is the varying number of Wikipedia categories that may be associated with each word in the text after the entity-linking process. The difference in the number of categories per word may lead to a bias towards a particular class. In addition, some categories are too generic so that they do not provide the discriminative features needed for the classifier. Incorporating the features of all retrieved categories into the short-text representation may present noise and obscure the semantics of the original text. Assume the following sentence as an example: "لعبة الرصاص للاعب الكرة (Cristiano Ronaldo)". The Wikipedia entity that corresponds to “Cristiano Ronaldo” comes under more than fifty different Wikipedia categories, most of which are related to sports. Other words in the sentence, such as "pencil" and "drawing" are both associated with several categories in total. Incorporating features of all categories into the classification process will likely lead to classifying the text under the "sports" class, while it is originally labeled “art”. Therefore, there is a need to filter categories to choose those that are most relevant to the context of the text, and to avoid biasing one class over another. We propose a solution that addresses the aforementioned problems at two levels: the knowledge level, and the deep learning level. At the knowledge level, categories extracted from Wikipedia following the entity-linking process are assessed and filtered by calculating the Information Content (IC) of the category. The IC of a category indicates the specificity of the category, and can be determined any analyzing the taxonomy of Wikipedia’s categories [17-19]. In general, top-level categories are generic and have low information content values and therefore should be ignored because they may not provide enough discriminative features for the classification process. In contrast, specialized and down-level categories are more descriptive, and hence could have more influence on the classification process.

At the deep learning model, we aim to train our model to focus on more important words and categories, and ignore irrelevant ones by using the attention mechanism. Two attention mechanisms are employed for this purpose at both the short text level and the category level: category-to-category attention is used to reweight the importance of Wikipedia categories based on the relative semantic relations between them. Then, category-to-sentence attention is used to signify words in the short text that best correspond to the category set. In addition, the attention mechanisms help to minimize the impact of noisy and irrelevant words and categories, and hence improve the classification performance.

Experiments were conducted on three datasets to evaluate the effectiveness of our method. We also explored the performance of our method using different parameters and word embedding models. An ablation study was also...
conducted to analyze the contributions of the different components of our method.

II. RELATED WORKS

Several approaches on short text classification used conventional machine learning techniques such as Support Vector Machine [20, 21] and Naïve Bayes [22]. These works employ the Bag of Words (BOW) representation [23] and mainly use the principles of statistics to classify. Some works attempted to expand the short text features by utilizing noun phrases, part-of-speech tags [24], dependency parsing [25] and tree kernels [26]. Other works exploited information extracted from search engines [7, 9] or background knowledge [27, 28] to expand the features of the short text, but these solutions do not solve the problem of data sparsity.

With recent advances in deep learning, a growing number of works have employed neural models that have the potential to solve the data sparsity problem by creating a dense and low-dimensional representation of the short text [29, 30]. For example, Zhang, et al. [31] presented an empirical study on character-level convolutional neural networks (CNNs) for text classification. They showed that working at the character level is particularly effective to lean abnormal character combinations, which are common in short texts, such as misspellings and emoticons. Wang, et al. [32] proposed a method that models short texts using word embeddings clustering and CNN. They performed clustering of word embeddings to group similar words together so that the CNN can achieve better performance.

Some works combined different types of neural networks in an attempt to improve the classification performance. For example, Hu, et al. [30] proposed a model that combines CNN with SVM to classify short texts. Their idea is to benefit from the advantage of CNN in capturing features between consecutive words through convolution processing, and the advantage of SVM in classifying data in the case of limited samples. Several works [29, 33, 34] introduced models that combine CNN with LSTM, which is effective in connecting the extracted features. Xian-yan, et al. [35] proposed a model based on latent Dirichlet allocation (LDA) and BiLSTM-CNN layers to solve the multilingual short text classification problem. LDA is used to extract topic vectors that correspond to the short text. These topic vectors are then combined with the word embeddings to expand the features of the short text. Despite the advantage of the aforementioned deep learning methods, they often perform well on large datasets, but the ambiguity of the short text brings a challenge to these methods because words are too few to reason over the dataset. Recently, incorporating information from external knowledge resources into neural models has been shown to improve the performance of neural networks on short text classification tasks [36]. For example, Wang, et al. [37] combined the words in the short text and relevant concepts extracted from a knowledge base to generate joint embedding, which is then fed into a CNN for classification. Similarly, Alam, et al. [38] combined the short text with relevant entities from Wikipedia to generate a word-entity embedding that is then fed into a CNN-based model. Marivate and Sefara [11] studied the effect of text augmentation on the performance of text classification. They examined different approaches to text augmentation, such as augmenting the text with synonyms or semantically similar words at either the source text or the embedding levels. The aforementioned deep learning methods have made significant progress in solving the ambiguity and data sparsity in the short text. However, these methods ignore the semantic similarity between words, as well as the relative importance of knowledge-based entities. On the other hand, the attention mechanism has made a great breakthrough in several tasks such as machine translation and computer vision. It has shown its efficiency in text classification tasks by highlighting relevant features of the input data. For example, Yang, et al. [39] proposed a hierarchical attention structure to classify texts, where word-level and sentence-level attentions are used to pay attention to more important information. Yin, et al. [40] proposed a CNN-based model and employed the attention mechanism at the character-level to achieve a better performance for short text classification. Other methods tried to better capture contextual information through sequential text classification that uses RNNs with attention mechanism. For example, Xie, et al. [41] used the BiLSTM, which is a special RNN architectural, to obtain contextual semantic features for classifying short texts, and to capture more important information in sentences using the self-attention mechanism. [42] proposed a model that leverages bidirectional multi-residual LSTM with attention module. They showed that their model outperforms other conventional RNN models.

Some works combined the attention mechanism with information extracted background knowledge to improve the classification performance. The intention is to extend the short text features with knowledge-based features and then pay importance to more discriminative features that can affect the classification performance. For example, Chen, et al. [43] incorporated contextual information from the short text with related conceptual information from the YAGO ontology [44]. They utilized deep neural networks with two attention mechanisms: one to consider the semantic similarity between the concepts and the short text, and the other to consider the relative similarity between concepts. Liu, et al. [45] proposed a short text classification approach that extends the features of short texts with information extracted from Probase knowledge [46]. They used a deep learning model on Temporal Convolutional Network (TCN) and CNN to improve the parallelization of the model. They also used two self-attention mechanisms at the word and concept levels. The previous efforts focused solely on English text, and benefited from the presence of rich knowledge sources expressed in English, and the pertaining entity-linking tools. This work explores the use of the attention mechanism and
information extracted from the Arabic Wikipedia to improve the classification of short Arabic texts. When it comes to the classification of Arabic texts, most efforts have focused on the classification of long texts and documents [47-49]. Few efforts, however, have tackled short Arabic text classification using traditional machine learning techniques such as SVM [50], decision trees [50] and Naïve bays [51, 52]. Recently, there has been a growing interest in adopting deep learning models to classify short Arabic texts [53]. However, most of the proposed models handled the short text in specific contexts such as sentiment analysis [54-56], fake news detection [57-59] or question answering [60]. These models perform well on large datasets, but they do not address the ambiguity and the data sparsity of short Arabic texts, which are likely to impede the performance when the datasets are relatively small. Alternatively, this work contributes to the field of short Arabic text classification by building and evaluating a model that: 1) exploits background knowledge such as Wikipedia to enrich the semantic representation of the short text, and 2) uses the attention mechanism at different levels to give importance to context-relevant words that affect the performance of classification, as well as reduce the effect of improper words.

In the following subsections, we first introduce the problem definition and hypothesis, and then present our model that combines Wikipedia-based features with the attention mechanism to classify Arabic short texts.

### III. PROBLEM DEFINITION

The problem of short Arabic text classification can be formulated as the following: A dataset of short Arabic texts can be formally represented as $D = (S, L)$, where $S$ denotes the set of sentences and $L$ denotes the set of labels assigned to sentences. $S$ is represented as the set $\{s_1, s_2, ..., s_{|S|}\}$, where each $s_i \in S$ is a short Arabic text consisting of a sequence of words $(w_1, w_2, ..., w_{|S|})$. $L$ is represented as the set $\{l_1, l_2, ..., l_{|L|}\}$, where each $l_j \in L$ is a predefined label. Given a sentence $s$, its words are first annotated with relevant Arabic Wikipedia entities by using an entity-linking tool. Wikipedia categories associated with identified entities are then extracted and ranked according to their information content values. Top-ranked categories are then used to enrich the semantic features of the short text as follows: Let $C_s = (c_1, c_2, ..., c_{|C_s|})$ be the set of Wikipedia categories that correspond to words in the sentence $s$, where $c_i$ is a Wikipedia category and $|C_s|$ denotes the size of the of category set. We aim to train a model that maps each short text $s \in S$ to a ground-truth label $l_j \in L$. To train the model, $s$ should be converted to a feature space that integrates the short text and the corresponding Wikipedia categories. Our hypothesis is that the integration of knowledge-based features from Wikipedia with a deep learning model will improve the performance of short Arabic text classification. We also anticipate that the use of the attention mechanism and category filtering will improve the performance by highlighting the important features that affect the classification process while reducing the impact of irrelevant features.

### IV. THE PROPOSED METHOD

Fig. 2 illustrates our method for short Arabic text classification. Given the sequence of words in the short text $(w_1, w_2, ..., w_{|S|})$ and the sequence of top relevant categories $(c_1, c_2, ..., c_{|C_s|})$, the two sequences are projected into the vector space by using an embedding layer. Then, the embeddings are encoded by being fed into neural network layers to obtain the final short text representation $(a_{w_1}, a_{w_2}, ..., a_{w_{|S|}})$ and the final category set representation $(a_{c_1}, a_{c_2}, ..., a_{c_{|C_s|}})$. Multiple attention mechanisms are used throughout the encoding process, including self-attention on the short text, self-attention on the category set, and category-to-sentence attention. The two aforementioned representations are then concatenated to fuse the short text and the knowledge-based information. Then, the combination is fed into the output layer, which is a dense layer that takes the number of class labels available as its output dimension. Finally, a softmax activation function is used to compute the probability distribution of class labels. The following sections explain each step in our method in detail.

#### A. ENTITY LINKING AND CATEGORY EXTRACTION

The aim of this phase is to retrieve Wikipedia categories relevant to the input text. This can be achieved by two main steps: 1) entity-linking, and 2) category extraction and...
filtering. Entity-linking to the knowledge base is the process of identifying knowledge entities mentioned in the text. In this work, we perform entity-linking on the Arabic Wikipedia. Although there are plenty of tools and approaches that can be used to annotate texts with Wikipedia entities [61-64], most of these tools use the English version of Wikipedia, while the entity-linking to the Arabic Wikipedia is not well supported. Thus, we use the entity-linking approach proposed in [65], in which the entity-linking is performed based on a local indexed version of the Arabic Wikipedia’s XML dump file. The contents of the dump file were retrieved and indexed using the Apache Lucene search engine to enable rapid access to and query over Wikipedia information. The used entity-linking approach also tackles the problem of article disambiguation by leveraging semantic similarity measures to filter out ambiguous entities. After identifying relevant Wikipedia entities, the categories associated with these entities are retrieved. These categories link related entities under a common topic and thus can provide useful topical information to enrich the semantic features of the short text.

B. CATEGORY FILTERING
As discussed in the introduction, the number of categories extracted after the entity-linking process varies largely from one entity to another, resulting in a bias towards certain words in the short text over others. In addition, categories of one entity may vary based on the generality: Categories that come from the category top levels of the taxonomy are more general and abstract. On the contrary, the categories from the lower levels are more concrete and specialized. In general, specialized categories are more descriptive of entities and thus can better distinguish between words in the short text than top-level categories [17]. Therefore, we aim to filter the retrieved categories to maintain only the most relevant ones that are likely to improve the classification performance. In this work, a simple approach is adopted to assess the relevance of categories by computing the Information Content (IC). The IC of a category indicates the specificity of that category and can be computed based on the hierarchical structure of categories modelled in Wikipedia [66]:

\[ IC(c) = \frac{\log(\text{maxDepth}(c))}{\log(\text{graphMaxDepth})} \]  

(1)

Where \( IC(c) \) is the information content of a category \( c \), \( \text{maxDepth}(c) \) is the depth of \( c \) in the entire taxonomy of categories in the Arabic Wikipedia, and \( \text{graphMaxDepth} \) is the maximum depth of the taxonomy of categories. The logarithm in Equation 1 is used to respond to skewness towards large depth values. This measure indicates that the larger the depth of \( c \), the larger IC value it has. It gives a value that ranges from 0 to 1, where 1 denotes the highest specificity. In general, top-level categories are of low depth, are often general and contain less information value compared to high-depth categories. For each word in the short text that is mapped to a Wikipedia entity, the IC values of the corresponding categories are computed. Categories with highest IC values are maintained while other categories are ignored. We set a threshold value to indicate the number of categories to be retained for each word. In the experimental part of this work, different threshold values are examined to determine the most appropriate one.

C. SHORT TEXT AND CATEGORY EMBEDDINGS
After retrieving the set of categories relevant to the input short text, both the short text and the category set are fed into an embedding layer to be transformed into embedding vectors. Let us use \( \tilde{s} = \langle \tilde{w}_1, \tilde{w}_2, ..., \tilde{w}_{|s|} \rangle \) to represent the embedding vectors of the sentence \( s \), where \( \tilde{w}_i \) is the embedding vector of the word \( w_i \in s \), and \( |s| \) represents the length of the sentence. Let us also use \( \tilde{C}_s = \langle \tilde{c}_1, \tilde{c}_2, ..., \tilde{c}_{|C_s|} \rangle \) to represent the embedding vectors of the Wikipedia categories in \( C_s \), where \( \tilde{c}_j \) is the embedding vector of the category \( c_j \in C_s \).

For Arabic word embedding, we experimented with several popular pre-trained static word embedding models including FastText [67] and AraVec [68]. In the experimental part we report on the influence of using different word embedding models on the classification results.

D. ENCODING OF THE SHORT TEXT
Given a sentence, not all context words have the same contribution to the semantics of a sentence. To address this issue, the self-attention mechanism is used to extract these more important words by giving them a higher weight to increase their importance. Self-attention is a case of attention that captures the important parts in the sentence itself and only requires a single sequence to compute its representation. Before applying the self-attention mechanism, a standard BiLSTM is used to process the sequence of word embeddings. The reason of using a BiLSTM layer before the self-attention layer is explained as follows: First, word embeddings cannot be fed directly into the self-attention layer because the latter uses weighted sum to generate output vectors, making its representational power limited [43]. The BiLSTM layer can capture the contextual information of the sequence before applying self-attention, the thing that can further increase the expressive power of the attentional network. Second, the embedding vectors in the sequence \( \tilde{s} \) are independent on each other. Thus, the BiLSTM will help to gain some dependency between adjacent words within a single sentence before applying self-attention [69].

Self-attention is applied on top of the BiLSTM layer as proposed by [69]: First, the BiLSTM processes \( \tilde{s} \) to obtain the \( \tilde{s} = \langle \tilde{w}_1, \tilde{w}_2, ..., \tilde{w}_{|s|} \rangle \). Essentially, \( \tilde{s} \) represents the whole BiLSTM hidden states from all time steps, and each hidden state \( \tilde{w}_i \) represents \( i \)-th word in the sentence together with its contextual information.

Afterwards, the self-attention layer takes \( \tilde{s} \) as input, and outputs a vector of weights \( \tilde{A} = \langle \alpha_1, \alpha_2, ..., \alpha_{|s|} \rangle \). This is performed as the following: Each hidden state \( \tilde{w}_i \in \tilde{s} \) is fed into a simple Multi-Layer Perceptron to get a new hidden representation \( \tilde{u}_i \) as shown in Equation 2. \( W_w \) and \( b_w \) are
parameters to be learned. Then a weight value $\alpha_i$ that indicates the importance of the $i$-th word is calculated for $\vec{w}_i$ as in Equation 3, given $u_i$ and a word-level context vector $\vec{u}_w$. $u_w$ is a high dimensional representation that is randomly initialized and jointly learned in the training process to judge the importance of different words in the sentence. Equation 3 uses the softmax function to ensure that all the computed weights sum up to 1. Finally, the attention-weighted sum $a_{w_i}$ is calculated for each word as in Equation 4. The expectation is that the words in the sentence that are more important should have larger weights:

\[ u_i = \tanh \left( W_w \vec{w}_i + b_w \right) \]
\[ \alpha_i = \frac{\exp (u_i^T u_w)}{\sum_{i=1}^{|\vec{w}|} \exp (u_i^T u_w)} \]
\[ a_{w_i} = \sum_{i=1}^{|\vec{w}|} \alpha_i \cdot \vec{w}_i \]

\section*{E. ENCODING OF THE CATEGORY SET}

Similarly to the encoding of the sentence, we use a standard BiLSTM to process the embedding vectors of categories in $C$, to obtain $\vec{c}_s = (\vec{c}_1, \vec{c}_2, ..., \vec{c}_{|C|})$, where $\vec{c}_j$ denotes the embedding vector of with contextual information of the $j$-th category in $C$.

To give more weight to important categories, two stages of attention are used: 1) a category-to-sentence attention, and 2) self-attention. At the first stage, a category-to-sentence attention mechanism is used to signify which Wikipedia categories have the closest similarity to one of the sentence words and are hence critical for classifying the sentence. Recall that the mapping of words to Wikipedia entities is likely to result in ambiguous and irrelevant categories. Thus, the proposed category-to-sentence attention aims to pay attention to context-related categories and reduce the importance of ambiguous and less descriptive categories.

The category-to-sentence attention is applied as follows: First, each $\vec{w}_i \in \vec{w}$ and $\vec{c}_j \in \vec{c}_s$, which are the outputs of the BiLSTM layers, are fed into a layer with tanh as its activation function to get $\tau_j$ and $\phi_i$. Then, the attention weight $\beta_i$ is calculated as shown in Equation 7, which demonstrates how well the $i$-th word in the sentence matches the $j$-th category. A larger $\beta_i$ means that the $i$-th category is semantically more similar to the sentence.

\[ \tau_j = \tanh \left( W \vec{c}_j + b \right) \]
\[ \phi_i = \tanh \left( W \vec{w}_i + b \right) \]
\[ \beta_i = \frac{\exp (\tau_j^T \phi_i)}{\sum_{i=1}^{|\vec{w}|} \exp (\tau_j^T \phi_i)} \]

The second level of attention is self-attention that aims to measure the importance of each Wikipedia category with respect to the whole category set. First, the BiLSTM output vector $\vec{c}_j$ is fed into a simple Multi-Layer Perceptron to get a new hidden representation $h_j$, as in Equation 8. $W_e$ and $b_e$ are parameters to be learned. Then a weight value $\gamma_j$ that indicates the importance of the $j$-th category is calculated using Equation 9, giving $h_j$ and a word-level context vector $\vec{u}_e$. $h_e$ is a high dimensional representation that is randomly initialized and jointly learned in the training process to judge the importance of different categories in the category set.

\[ h_i = \tanh \left( W_e \vec{c}_i + b_e \right) \]
\[ \gamma_j = \frac{\exp (h_j^T h_e)}{\sum_{i=1}^{|\vec{c}|} \exp (h_i^T h_e)} \]

So far, we computed two attention weights for each category $c_i$: $\beta_i$ which indicates the importance of the $i$-th category with respect to the sentence $s$, and $\gamma_j$ which indicates the importance of the $j$-th category with respect to the entire category set. To obtain the final weight of attention for each category, we combine $\beta_i$ and $\gamma_j$ using the following formula:

\[ \delta_i = \text{softmax}(\epsilon \beta_i + (1 - \epsilon) \gamma_j) \]

where the parameter $\epsilon \in [0, 1]$ is a weighting factor used to adjust the importance of the two attention weights $\beta_i$ and $\gamma_j$. Instead of setting $\epsilon$ manually, we use a neural network so that value of $\epsilon$ can be learned automatically during the training process.

We calculate $\epsilon$ by the following equations:

\[ \sigma_i = F(\beta_i)^T F(\gamma_j) \]
\[ \epsilon_i = \sigma(W^T \sigma_i + b) \]

where $F(\cdot)$ is a single non-linear layer with tanh as its activation function. $W$ and $b$ are parameters to be learned, and $\sigma$ is a sigmoid function to obtain a value in the range from 0 to 1. Finally, the attention-weighted sum $a_{c_i}$ of the category $c_i$ is calculated using the following equation:

\[ a_{c_i} = \sum_{i=1}^{|C|} \delta_i \cdot \vec{c}_i \]

\section*{F. COMBINATION OF THE SHORT TEXT AND CATEGORY SET REPRESENTATIONS}

Let $A_\delta$ be the sequence of attention-weighted representations of the words in the sentence, where $A_\delta = (a_{w_1}, a_{w_2}, ..., a_{w_{|\vec{w}|}})$, and $a_{w_i}$ is calculated using Equation 4. Let also $A_c$ be the sequence of attention-weighted representations of the Wikipedia categories, where $A_c = (a_{c_1}, a_{c_2}, ..., a_{c_{|C|}})$, and $a_{c_i}$ is calculated using Equation 13. Then, $A_\delta$ and $A_c$ are concatenated to produce the final representation of the short text that is enriched with knowledge-based features as the following:

\[ A = A_\delta \oplus A_c \]

where $\oplus$ denotes the concatenation operation.

\section*{G. OUTPUT LAYER AND MODEL TRAINING}

The combined representation obtained from Equation 14 is classified using the classification layer, which is a dense layer with the size of the number of classes of the dataset under...
consideration with a softmax activation function. At last, the class label \( l_i \in L \) with the maximum probability is chosen as the predicted label for sentence \( s \). The cross-entropy is employed as the training loss:

\[
    \text{Loss} = -\frac{1}{L} \sum_{i=1}^{L} y \log(\hat{y})
\]

where \( y \) is the ground-truth label of \( s \), and \( \hat{y} \) is the predicted probability distribution on all classes.

V. EXPERIMENTS

A. DATASETS

The datasets we used to evaluate our method are introduced in the following. Statistics on the datasets are shown in Table I.

- **ArSarcasm**\(^2\): The ArSarcasm dataset consists of 15,548 tweets divided into 12,548 training tweets and 3,000 testing tweets. It was used and released as part of the datasets so that words in each short text are mapped to the aspect of the Twitter user profile, which are: art-and-entertainment, politics, human-rights, business-and-economy, education, sports, health, science-and-technology, environment, spiritual and other. The “other” class was ignored because it contains a very small number of samples. We used a version of this dataset that was preprocessed and filtered by [72]. They filtered out noisy Twitter accounts whose tweets cannot be assigned to any category. The final dataset contains 115,692 tweets from 36 twitter accounts. The dataset was divided into 80% for training and 20% for testing. Note that these datasets were selected from the literature because they vary in terms of the scale, domain, and the intended classification (binary vs. multi-class classification). This variety allows us to better assess the performance of our method.

- **ArSAS**\(^2\): This is a manually annotated dataset for Arabic speech-act and sentiment analysis [71]. It contains around 21K tweets that cover many different topics, and most of them are in dialectal Arabic. Each of the tweets is labeled with one of the following: positive (4,643 tweets), negative (7,840 tweets), neutral (7,279 tweets), or mixed (6,495 tweets). We used the same splits provided in this dataset for training and testing.

- **AITD** (Arabic Influencer Twitter Dataset)\(^3\): This dataset was created by collecting tweets posted by 60 Arab influencers on Twitter [72]. A Twitter application programming interface was used to collect the last 3,200 tweets for each Twitter account. Tweets are categorized into 11 categories based on the aspect of the Twitter user profile, which are: art-and-entertainment, politics, human-rights, business-and-economy, education, sports, health, science-and-technology, environment, spiritual and other. The “other” class was ignored because it contains a very small number of samples. We used a version of this dataset that was preprocessed and filtered by [72]. They filtered out noisy Twitter accounts whose tweets cannot be assigned to any category. The final dataset contains 115,692 tweets from 36 twitter accounts. The dataset was divided into 80% for training and 20% for testing. Note that these datasets were selected from the literature because they vary in terms of the scale, domain, and the intended classification (binary vs. multi-class classification). This variety allows us to better assess the performance of our method.

B. DATA PREPROCESSING

The datasets were preprocessed through the following step: First, short texts were tokenized using the Farasa Arabic NLP toolkit [73]. Second, text parts that are not part of the language semantic structure were moved, including URLs, numeric expressions, punctuation and all tweet specific tokens (e.g. mentions, retweets, emojis, and hashtags). Third, Arabic diacritics were removed, and Arabic letters were normalized to convert the text into a more convenient and standard form. The Stanford Arabic Word Segmenter [74] was used to apply orthographic normalization to raw Arabic text. Elongation removal was also performed by removing repeated letters and reducing words into their standard forms.

C. ENTITY LINKING OF DATASETS

Entity linking to Arabic Wikipedia was performed on all datasets so that words in each short text are mapped to relevant Wikipedia entities. The two columns to the right of Table I summarize the results of the entity-linking process: One column shows the average number of Wikipedia entities identified per tweet in each dataset. The last column shows the average number of categories associated with the identified entities per tweet in each dataset.

D. COMPARED MODELS

We compare our deep learning model with five different models that have been used for Arabic text classification in previous works. These models are as follows:

1. https://github.com/iabufarha/ArSarcasm-v2
2. https://homepages.inf.ed.ac.uk/wmagdy/resources.htm
3. https://github.com/shammur/Arabic_news_text_classification_datasets
• BiLSTM: we used the BiLSTM architecture shown in Fig. 3(A), which was used in [54] and [75].
• CNN: we used the CNN model shown in Fig. 3(B), which was used in [75].
• CNN-LSTM: This combined architecture consists of a CNN layer with max pooling followed by LSTM layer as shown in Fig. 3(C). This model was used in [76] and [54]. The motivation to have such a model is that the CNN works as a feature extractor, then the LSTM works on the features extracted from the CNN to capture dependencies in the generated sequence.
• AraBERT\(^4\): This is an Arabic pre-trained language model provided by Antoun, et al. [77] based on Google’s BERT architecture [78]. AraBERT is trained on large Arabic datasets that include Arabic Wikipedia and news articles from all over the Arab region. We use ArabBERT (v0.2/2) because it is trained on more data compared to AraBERT (v0.1/1).
• QARiB\(^5\): The QCRI Arabic and Dialectal BERT (QARiB) model is an Arabic BERT model that was developed by Abdelali, et al. [79]. QARiB is implemented based on the BERT model and is trained on 420 million tweets and 180 million sentences of text.

![Figure 3](image)

FIGURE 3. The architectures of deep learning models used in our experiments

E. EXPERIMENTAL SETTINGS

While AraBERT and QARiB utilize their own embeddings, we used a FastText model [67] to generate word embeddings for other deep learning models, including ours. FastText is trained on portions of Wikipedia and CommonCrawl corpora written in Modern Standard Arabic (MSA), with vector dimensionality of 300. If a word is unknown, its embedding will be randomly initialized. AraBERT and QARiB were fine-tuned by adding a fully connected layer and a softmax layer after the pre-trained model. Then they were trained for a small number of epochs to adjust the weights for the specific tasks. The hyper-parameters for the deep learning experiments are shown in Table II. For all experiments, Adam optimizer [80] was used for parameter optimization, with a learning rate of 0.0001. In our model, the hyper-parameter \(\epsilon\) in Equation 10 is automatically learned, as this can achieve better classification results than using a fixed value.

I. RESULTS AND DISCUSSION

Table III summarizes the evaluation results in terms of accuracy, precision, recall, and F1 for our model and other deep learning models on the datasets. The performance of all models varies depending on the dataset. We use McNemar’s test for statistical significance test. We can observe that our model outperforms all other models on three of four tasks (ArSarcasm- sarcasm detection, ArSarcasm- sentiment analysis, AITD). The main reason is that our method enriches the semantics of short texts with features from Wikipedia, the thing that improves the classification performance. This reason was verified through the ablation study reported in Section V and which shows that the elimination of the knowledge-based categories causes the performance to drop noticeably.

When it comes to the ArSAS dataset, the best performance was achieved by QARiB followed by our model. We believe that this could be due to the large number of dialectical texts in this ArSAS, which could not be mapped to relevant Wikipedia entities. On average, ArSAS has the lowest number of matched entities and associated categories per each short text compared to other datasets (refer to Table I). Another possible reason for the better performance of QARiB is that it is trained on large volumes of both formal (news) and informal (tweets) data, and thus can better capture peculiarities of different platforms such as Twitter [72].

Comparing results across the datasets, it can be noticed that all models achieved the highest performance results in the sentiment analysis task on the ArSAS dataset. We think this is because ArSAS dataset is more balanced than the other two datasets. For example, 82% of the tweets in the ArSarcasm dataset are non-sarcastic compared to only 18% sarcastic.

| Model       | # LSTM cells | Recurrent dropout | Output dropout | # filters | Filter size | Pooling size | Hidden size | Batch size |
|-------------|--------------|-------------------|----------------|-----------|-------------|--------------|-------------|------------|
| Ours        | 64           | 0.2               | 0.2            | -         | -           | -            | 128         | 64         |
| BiLSTM      | 64           | 0.2               | 0.2            | -         | -           | -            | 128         | 64         |
| CNN         | -            | -                 | -              | 300       | 3           | 2            | 256         | 64         |
| CNN-LSTM    | 128          | 0.2               | 0.2            | 300       | 3           | 2            | 128         | 64         |

\(^4\) https://github.com/aub-mind/arabert
\(^5\) https://github.com/qcri/QARiB
tweets. AITD data also has a large variance in the distribution of the classes.

In general, our model and the language-based models (QARiB and AraBERT) perform significantly better than other deep learning models. This is because the language-based models are pre-trained on huge volumes of data, while our model utilizes knowledge-based features to enrich the short text. Another reason for the superiority of the aforementioned models is that they are all attentive models that can better capture powerful features by focusing on more important words and conceptual features.

| Dataset         | Evaluation | BiLSTM | CNN | CNN-LSTM | AraBERT | QARiB | Ours |
|-----------------|------------|--------|-----|----------|---------|-------|------|
| ArSarcasm (sarcasm detection) | Accuracy | 0.6987 | 0.6847 | 0.6717 | 0.7463 | 0.7510 | 0.7805* |
|                 | Precision | 0.6759 | 0.5497 | 0.6360 | 0.6985 | 0.7231 | 0.7693* |
|                 | Recall    | 0.4202 | 0.4185 | 0.5857 | 0.7649 | 0.7447 | 0.7760* |
|                 | F1        | 0.5182 | 0.4752 | 0.6098 | 0.7302 | 0.7337 | 0.7726* |
| ArSarcasm (sentiment analysis) | Accuracy | 0.5973 | 0.5847 | 0.6333 | 0.7104 | 0.7144 | 0.7345* |
|                 | Precision | 0.558  | 0.5764 | 0.5873 | 0.6774 | 0.6927 | 0.7055 |
|                 | Recall    | 0.5813 | 0.4546 | 0.6159 | 0.6988 | 0.7021 | 0.7282* |
|                 | F1        | 0.5694 | 0.5083 | 0.6013 | 0.6879 | 0.6974 | 0.7167* |
| ArSAS           | Accuracy  | 0.91314 | 0.901 | 0.9201 | 0.9236 | 0.9527* | 0.9307 |
|                 | Precision | 0.8785 | 0.8782 | 0.891 | 0.9148 | 0.9299* | 0.9108 |
|                 | Recall    | 0.8917 | 0.8831 | 0.9072 | 0.9113 | 0.9346 | 0.9242 |
|                 | F1        | 0.8851 | 0.8806 | 0.8990 | 0.9130 | 0.9322* | 0.9175 |
| AITD            | Accuracy  | 0.6633 | 0.6501 | 0.6917 | 0.8177 | 0.8355 | 0.8423 |
|                 | Precision | 0.6109 | 0.6353 | 0.6667 | 0.7998 | 0.8198 | 0.8205 |
|                 | Recall    | 0.6533 | 0.6379 | 0.6813 | 0.8082 | 0.8271 | 0.8404* |
|                 | F1        | 0.6314 | 0.6439 | 0.6739 | 0.8040 | 0.8234 | 0.8303 |

**TABLE III**

**Evaluation results in terms of accuracy, precision, recall, and F1. * indicates that the result is statistically significantly better than the best baseline for that dataset at 0.05 significance value based on McNemar test.**

| Dataset         | Model                  | Accuracy | Precision | Recall | F1   |
|-----------------|------------------------|----------|-----------|--------|------|
| ArSarcasm (sarcasm detection) | BiLSTM                     | -        | 0.62      | 0.38   | 0.46 |
|                 | Farha and Magdy [70]    | -        | 0.75      | 0.74   | 0.75 |
|                 | SalamBERT               | -        | 0.73      | 0.72   | 0.73 |
|                 | Husain and Uzuner [81]  | -        | 0.73      | 0.72   | 0.73 |
|                 | AraBERT-v2              | -        | -         | -      | 0.68 |
|                 | Abdul-Mageed, et al. [82]| -       | -         | -      | 0.715|
|                 | MARBERT                 | -        | 0.72      | 0.7    | 0.7 |
|                 | Husain and Uzuner [81]  | -        | 0.72      | 0.71   | 0.71 |
|                 | AraBERT-v2              | -        | 0.72      | 0.71   | 0.71 |
| ArSAS           | AraBERT-v2              | 0.93     | -         | -      | 0.92 |
|                 | Farha and Magdy [75]    | -        | 0.86      | -      | -    |
| AITD            | QARiB                   | -        | 0.86      | -      | -    |

**TABLE IV**

**Top results reported in the literature on the same datasets used in this work.**

**TABLE V**

**The effect of the hyper-parameter ε on our model.**
data will be relevant to the classification task being dealt with. In contrast, our model is trained only on portions of Wikipedia content that is relevant to the dataset in hand. This makes our model less computationally-intense at inference time comparing to the pre-trained models. We also compared the results of our model with experimental results reported in the literature. Table IV shows the results of some studies that used deep learning models over the same datasets. It shows the name of the dataset, the model used, and the performance results as reported in these studies (empty fields have no values mentioned). Comparing these results with ours, we notice the following: 1) Our model provides better performance on all datasets except AITD. 2) The results reported in previous studies are close to the results of our experiments for the same deep learning models. For example, AraBERT in the reported studies achieved F1-scores of 0.73% on ArSarcasm (sarcasm detection), 0.68% on ArSarcasm (sentiment analysis) and 93% on ArSAS. In our experiments, AraBERT achieved F1 scores of 0.73% on ArSarcasm (sarcasm detection), 0.71% on ArSarcasm (sentiment analysis) and 91.3% on ArSAS.

II. Effect of Knowledge Attention

Two attention mechanisms are used when encoding the knowledge from Wikipedia, which are category-to-sentence attention, and self-attention. The parameter \( \varepsilon \in [0, 1] \) is the weighting factor that tunes the contributions of attention weights in Equation 10. We aim to explore the best value of \( \varepsilon \) by varying it manually and observing the influence on the classification performance in terms of F1. Table V shows the F1 scores on the datasets when varying \( \varepsilon \) from 0 to 1 when a step of 0.25. In general, setting \( \varepsilon \) to the 0 or 1 does not lead to best performance results. When \( \varepsilon = 0 \), the model completely ignores the semantic similarity between categories and the short text, leading to the lowest performance on all datasets. The best value of \( \varepsilon \) varies from one task to another: In three of four tasks (ArSarcasm -sarcasm detection, ArSAS and AITD), the best performance is achieved when \( \varepsilon = 0.75 \). In ArSarcasm (sentiment analysis), the method yields best performance when \( \varepsilon = 0.5 \). In general, this result shows that both attention weights are important, since the best performance cannot be achieved if any of them is eliminated. It also seems that category-to-sentence attention has more effect on the performance than self-attention, indicating the potential of knowledge-based features for short-text classification.

III. Effect of Category Filtering

As explained in our method, the IC values of extracted categories after the entity-linking process are calculated in order to retain the most specific and descriptive categories. In the following experiment, we investigate the influence of the number of categories retained for each word on the classification performance. We vary this number from 1 to 10, and observe the influence on our model in terms of F1 score. Table VI shows the effect of the number of categories per word on the classification performance in all tasks. The results show that the classification tasks differ in terms of the number of categories per word that leads to the best performance. This number is 3 for ArSarcasm (sarcasm detection), 4 for ArSarcasm (sentiment analysis), 3 for ArSAS, and 6 for AITD. In general, setting this number too low or too high leads to poor performance in all tasks. If the number of categories is low, the influence of knowledge-based features will become marginal. If the number is high, the performance falls due to the incorporation of many noisy, generic or irrelevant categories. In fact, when the number of categories is too large, the effect of the original text will fade, while the effect of the knowledge-based features will be dominant. This result proves the effectiveness of the proposed category filtering approach for improving the classification performance.

IV. EFFECT OF ARABIC WORD EMBEDDINGS

We further investigate the impact of using different word embeddings on the classification results of our model. We explore three pre-trained models for Arabic word embedding which are:

- AraVec\(^6\) (Skip-gram-Wiki) [68]: This model is trained using Skip-Gram on the portions of Wikipedia and CommonCrawl corpora.
- AraVec (Skip-gram-Twitter) [68]: This model is trained using Skip-Gram Continuous Bag of Words on Twitter text.
- FastText\(^7\) [67]: This model from Facebook AI research lab is pretrained on 294 languages. It is similar to the CBOW architecture in Word2Vec, but it has a hierarchical structure and represents words in dense vector form.

Note that AraVec offers two other models pretrained using (CBOW) on either Wikipedia or Twitter, but they are excluded.

---

\(^6\) https://github.com/bakrianoo/aravec

\(^7\) https://github.com/facebookresearch/fastText
because the Skip-Gram technique outperforms CBOW based on experimental results from previous works [83, 84]. Note also that language-based models (AraBERT and QARiB) use their own word embeddings.

Table VII shows the F1 scores resulting from the aforementioned embedding methods. The results show that the FastText model outperforms other models on all datasets. This may be attributed to the fact that FastText treats each word as composed of character n-grams, and thus can generate better word embeddings for rare words. This also enables FastText to provide word representations that do not appear in training data, and thus can overcome the out of vocabulary problem [85]. Furthermore, we noticed that models trained on Twitter data such as FastText and AraVec (Skip-gram and Twitter) generally outperform the models trained solely on Wikipedia, i.e. AraVec (Skip-gram and Wikipedia). This may be due to the nature of our datasets which mostly consists of tweets written in both MSA and other Arabic dialects.

V. ABLATION STUDY

To evaluate the influence of all components in the proposed method, we conducted an ablation study using variants of our method as follows.

- V-WithoutCategories: In this variant we eliminate the impact of the information extracted from Wikipedia, and consider only the impact of words of short texts in classification. The self-attention mechanism at the text level is still used. This variant is used to verify the effectiveness of knowledge-based features.

- V-WithoutAttention: In this variant, we removed the attention mechanisms, including self-attention and category-to-sentence attention, so that the model no longer pays more attention to important words. Wikipedia categories are still used to enrich text features. This variant is used to verify the effectiveness of the attention mechanism.

- V-WithoutAttentionCategories: In this variant, both Wikipedia-based features and attention mechanisms are eliminated. Word-embeddings are processed only by BiLSTM layers.

Table VIII shows the F1 score values for the variants of our method. In general, the results show that both the categories and the attention mechanism have influence on the performance of the method. The V-WithoutAttentionCategories gives the lowest performance among all other variants. By comparing the influence of components separately, we notice that V-WithoutCategories has lower performance than the V-WithoutAttention on all datasets. This indicates that the knowledge-based features are more influential than the attention mechanism in classifying short texts. This is because short texts tend to be ambiguous without enough contextual information to perform classification. Finally, combining both the attention mechanism and knowledge-based entities resulted in the best classification performance.

VI. ERROR ANALYSIS

Sample errors in the classification results were inspected to identify the reasons of errors. We identified the following types of errors. Where applicable, we present examples from the AITD dataset to demonstrate samples of common errors.

Dialectical Arabic words: One of the main sources of errors comes from the use of dialectical words that do not map to any

| Dataset       | Evaluation | V-WithoutCategories | V-WithoutAttention | V-WithoutAttentionCategories | Ours  |
|---------------|------------|---------------------|--------------------|-----------------------------|-------|
| ArSarcasm     | Precision  | 0.6399              | 0.7015             | 0.6601                      | 0.7693|
|               | Recall     | 0.6466              | 0.7921             | 0.4411                      | 0.7760|
|               | F1         | 0.6432              | 0.7441             | 0.5288                      | 0.7726|
| ArSarcasm     | Precision  | 0.6175              | 0.6805             | 0.5618                      | 0.7055|
| (sentiment    | Recall     | 0.6352              | 0.7184             | 0.6041                      | 0.7282|
| analysis)     | F1         | 0.6262              | 0.6989             | 0.5822                      | 0.7167|
| ArSAS         | Precision  | 0.8801              | 0.8852             | 0.8681                      | 0.9108|
|               | Recall     | 0.9044              | 0.8908             | 0.8813                      | 0.9242|
|               | F1         | 0.8921              | 0.8880             | 0.8747                      | 0.9175|
| AITD          | Precision  | 0.6892              | 0.7889             | 0.6352                      | 0.8205|
|               | Recall     | 0.7171              | 0.8085             | 0.6602                      | 0.8404|
|               | F1         | 0.7029              | 0.7986             | 0.6475                      | 0.8303|

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
Wikipedia entity. Since the Arabic version of Wikipedia is mostly written in modern standard Arabic, many dialectical words could not be annotated with any Wikipedia entity, thus lacked the conceptual information needed for the classification.

Too short texts: some texts are too short, so that they could not map to any Wikipedia entity, leading to incorrect classification of these short texts. Examples of such texts are: “عندما أكتب (Very great)” and “أنا أحبك (Go one. I liked you)”. In particular, very short texts caused many errors as a result of the ambiguity resulting from the entity-linking process: This is the case when the same mention refers to different entities in Wikipedia. Our entity-linking approach, like many other approaches [15, 61, 63], resolves the ambiguity issue by computing the semantic similarity between entities identified in the same text; and then choosing entities that are most similar to each other. However, this approach for ambiguity resolution becomes inapplicable when the text is too short such that only one entity is identified in the text because there will be no other entities to carry out the similarity measure. Assume the following the sentence “من سيظهر باللقب؟ (Who will win the title?)” as an example. The word “title” refers to multiple entities, including the title used before a person’s name, the title of a creative work, and the title in a sports competition. While the last entity is the intended one, our entity-linking approach does not correctly identify it due to the lack of other entities needed for the ambiguity resolution task.

VII. CONCLUSION AND FUTURE WORK

This work proposes a deep learning approach for short Arabic text classification that combines contextual text features with Wikipedia-based features for effective classification. The proposed approach utilizes several mechanisms to focus on important text and knowledge-based features: First, it filters categories extracted from Wikipedia to keep only the most relevant ones. Second, it employs attention mechanisms at both the short text and category levels to reweight the influence of words and categories based on their similarity to the input text. Finally, we conducted experiments on three datasets and the results demonstrated that our approach is more effective in short Arabic text classification compared to most competing deep learning-based methods.

In the future, our aim is to extract and incorporate additional features from knowledge sources that may further improve the classification. For example, we will investigate the use of structured knowledge bases such as DBPedia and domain ontologies to exploit not only the knowledge concepts but also the relationships between concepts and the values of associated properties. Another direction could be to employ our approach in the domain of Arabic question answering. Our hypothesis is that enriching the context of the question with related information from external knowledge can help the model better understand and answer the question.

VIII. REFERENCES

[1] W. B. Croft, D. Metzler, and T. Strohman, Search engines: Information retrieval in practice. Addison-Wesley Reading, 2010.
[2] X. Sun, J. J. M. T. He, and Applications, “A novel approach to generate a large scale of supervised data for short text sentiment analysis,” vol. 79, no. 9, pp. 5439-5459, 2020.
[3] W. Jing and Y. Bailong, “News Text Classification and Recommendation Technology Based on Wide & Deep-Bert Model,” in 2021 IEEE International Conference on Information Communication and Software Engineering (ICICSIE), 2021, pp. 209-216: IEEE.
[4] G. Song, Y. Ye, X. Du, X. Huang, and S. Bie, “Short text classification: A survey,” Journal of multimedia, vol. 9, no. 5, p. 635, 2014.
[5] C. Li, G. Zhan, and Z. Li, “News text classification based on improved Bi-LSTM-CNN,” in 2018 9th International conference on information technology in medicine and education (ITME), 2018, pp. 890-893: IEEE.
[6] I. Alsmadi and K. H. Gan, “Review of short-text classification,” International Journal of Web Information Systems, 2019.
[7] W. Meng, L. Lanfen, W. Jing, Y. Penghua, L. Jialong, and X. Fei, “Improving short text classification using public search engines,” in International Symposium on Integrated Uncertainty in Knowledge Modelling and Decision Making, 2013, pp. 157-166: Springer.
[8] A.-A. Iyad and A. Ahmed, “Towards Supporting Exploratory Search over the Arabic Web Content: The Case of ArabXplore,” Journal of Information Technology Management, vol. 12, no. 4, pp. 160-179, 2020.
[9] A. Sun, “Short text classification using very few words,” in Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval, 2012, pp. 1145-1146.
[10] J. Li, Y. Cai, Z. Cai, H. Leung, and K. Yang, “Wikipedia based short text classification method,” in International Conference on Database Systems for Advanced Applications, 2017, pp. 275-286: Springer.
[11] V. Marivate and T. Sefara, “Improving short text classification through global augmentation methods,” in International Cross-Domain Conference for Machine Learning and Knowledge Extraction, 2020, pp. 385-399: Springer.
[12] I. AlAgha and Y. Abu-Samra, “Tag Recommendation for Short Arabic Text by Using Latent Semantic Analysis of Wikipedia,” Jordanian Journal of Computer Science and Information Technology, vol. 6, no. 02, pp. 165-181, 2020.
[13] L. Wensen, C. Zewen, W. Jun, and W. Xiaoyi, “Short text classification based on Wikipedia and Word2vec,” in 2016 2nd IEEE International Conference on Computer and Communications (ICCC), 2016, pp. 1195-1200: IEEE.
[14] P. Ferragina and U. Scalaia, “Fast and accurate annotation of short texts with wikipedia pages,” IEEE software, vol. 29, no. 1, pp. 70-75, 2011.
[15] D. Ceccarelli, C. Lucchese, S. Orlando, R. Perez, and S. Trani, “Dexter: an open source framework for entity linking,” in Proceedings of the eighth international workshop on Exploiting semantic annotations in information retrieval, 2013, pp. 17-20.
[16] W. Shen, J. Wang, and J. Han, “Entity linking with a knowledge base: Issues, techniques, and solutions,” IEEE Transactions on Knowledge and Data Engineering, vol. 27, no. 2, pp. 443-460, 2014.
[17] M. B. Aouicha, M. A. Hadj Taieb, and A. B. Hamadou, “Taxonomy-based information content and wordnet-wiktionary-wikipedia glosses for semantic relatedness,” Applied Intelligence, vol. 45, no. 2, pp. 475-511, 2016.
[18] N. Seco, T. Veale, and J. Hayes, “An intrinsic information content metric for semantic similarity in WordNet,” in Ecai, 2004, vol. 16, p. 1089.
[19] D. Sánchez, M. Batet, and D. Isern, “Ontology-based information content computation,” Knowledge-based systems, vol. 24, no. 2, pp. 297-303, 2011.
[20] S. García-Méndez, M. Fernandez-Gavilanes, J. Juncal-Martínez, F. J. González-Castaño, and O. B. Seara, “Identifying banking transaction descriptions via support vector machine short-text classification based on a specialized labelled corpus,” IEEE Access, vol. 8, pp. 61642-61655, 2020.
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3175306, IEEE Access
[61] B. Hachey, W. Radford, J. Nothman, M. Honnibal, and J. R. Curran, "Evaluating entity linking with wikipedia," Artificial intelligence, vol. 194, pp. 130-150, 2013.

[62] A. Moro, A. Raganato, and R.Navigli, "Entity linking meets word sense disambiguation: a unified approach," Transactions of the Association for Computational Linguistics, vol. 2, pp. 231-244, 2014.

[63] J. Lin and A. Zettles, "WikiGUM: Exhaustive entity linking for Wikification in 12 genres," arXiv preprint arXiv:07449, 2021.

[64] X. Liu, Y. Li, H. Wu, M. Zhou, F. Wei, and Y. Lu, "Entity linking for tweets," in Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2013, pp. 1304-1311.

[65] F. Fayad and I. AlAgha, "Automatic Linking of Short Arabic Texts to Wikipedia," M. Sc. Thesis, Faculty of Information Technology, The Islamic University ..., 2013.

[66] I. Traverso, M.-E. Vidal, B. Kämppgen, and Y. Sure-Vetter, "GADES: a graph-based semantic similarity measure," in Proceedings of the 12th International Conference on Semantic Systems, 2016, pp. 101-104.

[67] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," Transactions of the Association for Computational Linguistics, vol. 5, pp. 135-146, 2017.

[68] A. B. Soliman, K. Eissa, and S. R. El-Beltagy, "Aravec: A set of arabic word embedding models for use in arabic nlp," Procedia Computer Science, vol. 117, pp. 256-265, 2017.

[69] Z. Lin et al., "A structured self-attentive sentence embedding," arXiv preprint arXiv:01759, 2017.

[70] I. A. Farha and W. Magdy, "From arabic sentiment analysis to sarcasm detection: The arsarcasm dataset," in Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection, 2020, pp. 32-39.

[71] A. Elmadany, H. Mubarak, and W. Magdy, "Arsas: An arabic speech-act and sentiment corpus of tweets," OSACT, vol. 3, p. 20, 2018.

[72] S. A. Chowdhury, A. Abdelali, K. Darwish, J. Soon-Gyo, J. Salminen, and B. J. Jansen, "Improving Arabic text categorization using transformer training diversification," in Proceedings of the fifth Arabic natural language processing workshop, 2020, pp. 226-236.

[73] A. Abdelali, K. Darwish, N. Durrani, and H. Mubarak, "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, 2016, pp. 11-16.

[74] W. Monroe, S. Green, and C. D. Manning, "Word segmentation of informal Arabic with domain adaptation," in Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), 2014, pp. 206-211.

[75] I. A. Farha and W. Magdy, "A comparative study of effective approaches for Arabic sentiment analysis," Information Processing and Management, vol. 58, no. 2, p. 102438, 2021.

[76] A. M. Alayba, V. Palade, M. England, and R. Iqbal, "Arabic language sentiment analysis on health services," in 2017 1st international workshop on arabic script analysis and recognition (usar), 2017, pp. 114-118: IEEE.

[77] W. Antoun, F. Baly, and H. Hajj, "Arabert: Transformer-based model for arabic language understanding," arXiv preprint arXiv:00104, 2020.

[78] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:04035, 2018.

[79] A. Abdelali, S. Hassan, H. Mubarak, K. Darwish, and Y. Samih, "Pre-training bert on arabic tweets: Practical considerations," arXiv preprint arXiv:10684, 2021.

[80] D. P. Kingma and J. J. a. p. a. Ba, "Adam: A method for stochastic optimization," 2014.

[81] F. Husain and O. Uzuner, "Leveraging offensive language for sarcasm and sentiment detection in Arabic," in Proceedings of the Sixth Arabic Natural Language Processing Workshop, 2021, pp. 364-369.

[82] M. Abdul-Mageed, A. Elmadany, and E. M. B. Nagoudi, "ARBERT & MARBERT: deep bidirectional transformers for Arabic," arXiv preprint arXiv:01759, 2016.