Integrating character-level and word-level representation for affect in Arabic tweets

Alharbi, Abdullah I.; Smith, Phillip; Lee, Mark

DOI:
10.1016/j.datak.2021.101973

Document Version
Peer reviewed version

Citation for published version (Harvard):
Alharbi, AI, Smith, P & Lee, M 2022, 'Integrating character-level and word-level representation for affect in Arabic tweets', *Data and Knowledge Engineering*, vol. 138, 101973. https://doi.org/10.1016/j.datak.2021.101973

Integrating Character-level and Word-level Representation for Affect in Arabic Tweets

Abdullah I. Alharbi[1,2], Phillip Smith[1] and Mark Lee[1]

[1] School of Computer Science, University of Birmingham, Birmingham, UK
[2] Faculty of Computing and Information Technology in Rabigh, King Abdulaziz University , Jeddah, Saudi Arabia
{aia784, p.smith.7, m.g.lee}@cs.bham.ac.uk
aamalharbe@kau.edu.sa

Abstract. Affect tasks, which range from sentiment polarity classification to finer grained sentiment strength and emotional intensity detection, have become of increasing interest due to the vast amount of user-generated content and advanced learning models. Word representation models have been leveraged effectively within a variety of natural language processing tasks. However, these models are not always effective in the context of social media. When dealing with social media posts in Arabic, the use of Arabic dialects needs to be considered. Although using informal text to train word-level models can lead to the identification of words that convey the same meaning, these models are unable to capture the full extent of the words that are used in the real world due to out-of-vocabulary (OOV) words. The inability to identify such words is one of the main limitations of word-level models. One approach of overcoming OOV is through the use of character-level embeddings as they can effectively learn the vectors of word parts or character n-grams. This study uses a combination of character-level and word-level models to identify the most effective methods by which affective Arabic words in tweets can be represented semantically and morphologically. We evaluate our generated models and the proposed method by integrating them in a supervised learning framework that was used for a range of affect tasks and other related tasks. Our findings reveal that the developed models surpassed the performance of state-of-the-art Arabic pre-trained word embeddings over eight datasets. In addition, our models enhance previous state-of-the-art outcomes on tasks involving Arabic emotion intensity, outperforming the top-systems that used advanced ensemble learning models and several additional features.

Keywords: Word-level embeddings · Character-level embeddings · Arabic tweets · Affect tasks.

1 Introduction

Language is not only employed by human beings for expressing emotions or sentiment; it is also used to display the intensity of such feelings. The term "affect" refers to a variety of categorisations related to emotions, which range from classifying sentiments (positive-negative) to finer grained categorisation of how strong a sentiment or emotion is (e.g., extreme sadness, mild sadness). It is a significant challenge to detect affect in text, particularly when looking at social media, e.g. Twitter, because the language employed is constrained by numerical limits and is also highly informal, using both symbols and slang.

However, it is an even greater challenge when looking at languages that have a rich morphology, such as Arabic [5]. Arabic social media users usually employ a variety of dialects and sub-dialects when communicating. While Modern Standard Arabic (MSA) has certain standards and rules, Arabic dialects used on social media usually lack such rules and standards. Thus, when examining Arabic affect in tweets, we need to develop tools and resources that can offer a better understanding and interpretation of the varied linguistic forms employed.

One of the central techniques in Natural Language Processing (NLP) is word embedding [17,48,9,27,26]. Word embedding employs dense vectors for representing words that project into continuous vector space, which reduces dimension numbers [29]. Nevertheless, such models can be ineffective when used with Arabic tweets. When we attempt to train the word-level models with informal language, it has been demonstrated that such models have difficulty in recognising a variety of forms of the same words that share meanings. Such unknown words, referred to as out-of-vocabulary (OOV) words, are one of the primary limitations of word-level models. In contrast, using character-level embedding can be effective in overcoming OOV words by employing their capacity for learning character n-grams (word parts). Nevertheless, character-level embedding is so sensitive that the model will encode every variant of the morphology of the word that has greater closeness within the embedded space than words which are similar semantically. Table 1 illustrates a pair of examples of affect words in Arabic dialects متنرفز (mtnrfz)[1] and مروق (mrwq), where word similarity is generally derived from character-level morphology and word-level semantics.

Table 1. Most similar words of different affect words using character and word level embeddings.

Example of a negative query term: متنرفز mtnrfz (uptigh)		Example of a positive query term: مروق mrwq (relaxed)	
Character-level model	Word-level model	Character-level model	Word-level model
متنرفزه mtnrfz (uptight-feminine)	معصب mESb (angry)	ومروق wmrwq (and relaxed)	مصحصح mSHSH (mindful)
متنرفزين mtnrfzyn (uptight-plural)	متوتر mtwtr (tense)	مروقه mrwqh (relaxed-feminine)	ومروق wmrwq (and relaxed)
نتنرفز ntnrfz (uptight-present verb)	متضايق mtDAyq (annoyed)	ومروقه wmrwqh (and relaxed-feminine)	فايق fAyq (awake)
بيتنرفز bytnrfz (uptight-future verb)	منفس mnfs (furious)	رايق rAyq (relaxed)	مفلل mfll (restful)
تتنرفز ttnrfz (uptight- feminine verb)	مضغوط mDgwT (enraged)	رايقه rAyqh (relaxed-feminine)	مستانس mstAns (happy)

In this research, we take advantage of character- and word-level models to discover an effective means of representing Arabic affect in tweets; the resulting model is called Affect

[1] Buckwalter's transliteration is used to represent Arabic orthography with morphological information (Buckwalter, 2004)

Character and Word Embeddings (ACWE). Initially, each model was trained with a large collection of tweets that were specifically selected to ensure demonstrable variations in affect terms from different Arabic dialects. A novel method was then used to concatenate the two models so that each word was represented semantically and morphologically. The ACWE model was evaluated by applying it as an input feature under a supervised learning framework using four benchmark datasets from SemEval-2018 Task 1 (Affect in Tweets) [31] and other tasks that were closely related to the affect area of study (offensive, hate speech and sarcasm detection). Our method advances a state-of-the-art approach to the task of Arabic in emotional intensity, and it outperformed top systems that used combinations of deep neural networks and several other features. Additionally, our method obtained superior outcomes compared with other Arabic pre-trained word embedding models. ACWE has been released for use in pre-trained word embeddings for applications and research relying on Arabic sentiment and emotion analysis and related tasks.

This research presents the following contributions:

- Word- and character-level embeddings are created for Arabic affect tasks in informal social media.
- A novel method combining character- and word-level embeddings for Arabic affect tasks is proposed.
- Systematic analysis is performed to determine the effectiveness of applying pre-processing techniques to a large training corpus prior to generating word embeddings, a study that has not previously been examined for noisy user-generated text.
- ACWE has been released for use as a pre-trained word embedding model for applications and research involving sentiment and emotion analysis of Arabic.
- We used eight datasets to evaluate the performance of our models after using them as input features into different machine and deep learning approaches.

The remainder of this paper is organised as follows. Section 2 provides an overview of the related literature. Section 3 provides a detailed discussion of how the data we used was collected and pre-processed. Section 4 explains our methodology for generating word-level and character-level embeddings and also how they are combined. Section 5 describes the experimental setup which includes the datasets, off-the-shelf pre-trained word embeddings and supervised learning models. Section 6 presents the results of using the experimental models on the downstream tasks. The main findings of our research are discussed in Section 7. Finally, Section 8 concludes the paper and provides some suggested future directions.

2 Related Works

2.1 Pre-trained Word Embedding Models

Most studies on Arabic word embedding focus on the application of word-level models [46,45,3,8] and, to a smaller extent, on character-level models [7]. An early work aimed to build word-level embeddings for Arabic [46]. They employed three techniques (CBOW, skip-gram [29] and GloVe [39]) to create word-level representations within a vector space for MSA. To pre-train their word embedding models, they used a significant corpus of Arabic texts (5.8 billion words) collected from a number of sources, such as documents

that had been translated, news articles, the Arabic Gigaword corpus [38] and Arabic Wikipedia. This model holds 300-dimensional vectors consisting of 6 million words and phrases.

AraVec [45] has one of the most common collections of open-source word embeddings, comprising six separate word embedding models for use with Arabic. The training data were derived from three sources, namely, Twitter, Wikipedia and Common Crawl. Similar to [46], they employed CBOW and skip-gram to learn word representations for Arabic NLP applications.

Recently, Abu Farha and Magdy [3] generated the largest word-level embedding model using 250 million Arabic tweets. While numerous words were used to train the models, they could not identify the same words in different forms that were employed in real human speech, as a result of the limited nature of these word-level models. In general, the effectiveness of the embedding depends on the task [40] and is considerably affected by the variety of words that are related to the task in hand [14].

To the best of our knowledge, there is no Arabic pre-trained character-level embedding model targeting Arabic dialects. In addition, we are not aware of any research work that has investigated the impact of pre-processing techniques on the generated word embedding models. The only exception is [10], who systematically studied different pre-processing factors in well-formed English datasets (such as Wikipedia and News). Our work generated character- and word-level embedding models for Arabic informal social media (noisy user-generated text). Additionally, we systematically compared the impact of the pre-processing on the effectiveness of the generated models (character- and word-level) over eight downstream tasks.

2.2 Combining Character- and Word-level Embeddings

Dos et al. [43] performed initial attempts at combining character- and word-level information in English; they offered a deep neural network architecture that can undertake Sentiment Analysis (SA) using sentence-, word- and character-level representations. They used a pre-trained word-level embedding model employing word2vec, but they did not use a pre-trained character-level model. Instead, character vectors were initialised using random sampling.

Recently, [24] proposed a word embedding model (CharCNN) for integrating word representations from word- and character-level models to capture morphological information, such as word suffixes and prefixes. Unlike the model of [43], CharCNN represents a fully conversational network with no max pooling layer for superior semantic information capture in character chunks. To the best of our knowledge, no study has attempted a combination of the two levels (character and word) for generating word representations specifically to Arabic affect tasks in informal text.

Unlike the aforementioned researchers, we separately pre-trained character n-grams and a word embedding model on a large dataset to learn semantics and morphology separately. We then combined them as input features in a supervised learning framework for the downstream tasks.

2.3 Sentiment and Emotion Analysis

Over the last decade, researchers have given considerable attention to Arabic SA due to the large quantity of data available from Arabic social media that reflect opinions and sentiments. Early SA experimentation in Arabic generally centred around expensive

customised elements in which feature representation needed human input to achieve the required accuracy levels [18,33,11]. In recent times, the standard of SA in Arabic has improved through the use of substantial pre-trained language models [3]. Classification of Arabic emotions is intricately linked with these efforts, aiming to identify various emotions contained in texts, such as happiness, surprise, disgust and anger [6,19]. Such endeavours have developed in ways similar to the models and resources of SA.

Although considerable research has been performed to analyse emotions and sentiments, studies that examine emotional intensity or level are less common [30]. Notable works on detecting emotional intensity can be found in SemEval 2018 Task 1 (Affect in Tweets) [31]. Most of the teams that performed well in the competition combined deep neural network tweet representation and features originating from the current lexicons related to emotions and sentiments. AffecThor [1] emerged as the most successful way of conducting tasks related to Arabic emotional intensity. These researchers suggested a system employing different handcrafted lexicons alongside a pre-trained word-level embedding model that was created using 4 million tweets. Such feature representations were employed for input; training was performed with ensemble deep network architectures (bidirectional long-short term memory [BiLSTM] with attention and conversational neural network [CNN] with max pooling).

The EiTAKA system [21] demonstrated the second best performance, presenting an ensemble model that combines two approaches. The first approach, N-Channels ConvNet, is a deep learning methodology; the second one is an XGBoost regressor centred around a connection of features based on lexicons and embedding. This combined technique assisted in delivering improved performance for the emotional intensity challenges with the final model. The third place for emotional intensity regression was given to the system suggested by the UNCC [2], and third place for emotional intensity classification went to the EMA system [12]. These systems both employed pre-trained word-level embeddings (AraVec) within supervised learning frameworks. The EMA system selected applied stems as their processing methods rather than lemmas, as Arabic users of Twitter generally employ dialectical Arabic, and most Arabic morphological analysers undergo training using MSA data.

3 Data Collection and Pre-processing

3.1 Data Collection

The size and variety of the training dataset are major aspects to be considered in improving word embedding quality. For this research, 10 million tweets were gathered using the Twitter API. We aimed to collect tweets that contained 1) various affect-associated words and 2) different Arabic dialects. Enriching our data with such varieties can improve the effectiveness of the generated word embedding models to target Arabic affect in social media.

First, to ensure that these tweets covered a range of affect-associated words, we initially employed the English NRC lexicon [32] for a selection of 63 words[2], which represent different levels and intensity of emotions. The lexicon includes common English terms that are associated with emotions to different degrees. Each term has a real-valued affect intensity score and its corresponding emotion (e.g. anger, fear and joy). We then translated these words into Arabic using Reverso context[3], an online translation service. This tool

[2] These are words that directly convey meanings of sentiments or emotions, such as *anger* and *rage*. They are not words that indirectly convey sentiments, such as *dead* and *tears*.

[3] `http://context.reverso.net`

was also used to find synonyms for the selected words, thereby extending the range of terms from 63 to 228. At this point, the collection of terms covered MSA affect words, which is a predicted outcome from English-to-Arabic translation.

To ensure that the collected tweets would reflect a range of dialects, we employed an MSA term list to search for dialect synonyms in two online dictionaries (Mo3jam[4] and Atlas Allhajaat[5]), which extended the term list by adding 217 new dialectical affect words. Additionally, emojis could be used as a universal language, according to [23]. We chose the 30 most popular emojis from the sentiment scores established in [23], and these were fed into the list of terms. Lastly, we made the assumption that any tweet from a particular Arabic-speaking country would most likely be written using a dialect of the country from which it originated. We retrieved tweets including every identified term (around 500 terms) by using the Twitter Search API and inputting the geolocations of various Arab countries.

3.2 Data Pre-processing

Data collected from Twitter generally includes content that is not useful for affect classification tasks such as mentions, links and unknown symbols. This type of 'noise' has to be treated before training models in order to reduce both the noise and the size of the vector space [25,44]. In this study, we applied different pre-processing techniques to investigate their impact on affect tasks. These methods were integrated at the word embedding generation phase and the downstream task stage (classification/regression datasets). Figure 1 illustrates the stages and steps of applying these pre-processing methods. We studied the following pre-processing techniques, which we believed are the most important for Arabic content in social media:

- **Cleaning (*clean) :*** Common text pre-processing methods, such as removal of unknown symbols, other language letters, diacritics, punctuation marks and URLs, were applied in the first instance.

- **Normalisation of letters (*norm):*** Letters that appeared in different forms in the original tweets were rendered into a single form. For example, the 'hamza' on characters {أ,إ} was replaced with {ا}, while the 't marbouta' {ة} was replaced with {ه}.

- **Elongated words (elong):** Social media users often repeat some letters for emphasis, such as 'hapyyyyy' and 'saaad'. This nonstandard writing could be treated by removing the repeated characters.

- **Hashtag segmentation (hashSeg):** Hashtags are used to draw attention to words or phrases that are trending, such as #sad and #fun. While it is common to remove both the hash symbol and words, we removed the hash symbol but kept the words. Users sometimes express their emotions using hashtags, so it was considered useful to retain them. In addition, Arabic Twitter users typically combine multiple words as one hashtag; thus, we segmented such forms to be treated as individual words.

[4] http://en.mo3jam.com
[5] http://atlasallhajaat.com

– **Emoji removal (emoji):** We applied this method to remove emojis from the text. By default, emojis are retained.

– **Stemming (stem):** We used this technique to reduce a word to its root form. We used an open source Python toolkit for Arabic (CAMeL tools) [36] to stem the target text.

We systematically investigated the impact of these pre-processing techniques individually. We grouped either some of them or all of them before generating word embeddings and when targeting the downstream tasks. To maximize the stability of the outcome, we carefully considered the sequence of the aforementioned pre-processing techniques. For example, hashtag segmentation had to be applied prior to stemming in order to stem individual words coming from hashtags. We used the following order to combine the above mentioned methods: *clean*, *norm*, *hashSeg*, *elong*, *emojis* and *stem*.

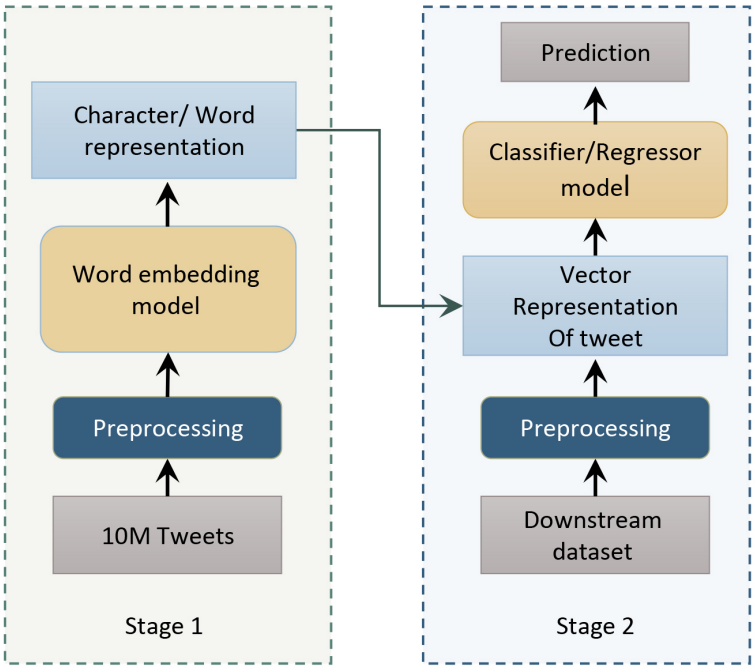

Fig. 1. The stages and steps of applying the pre-processing techniques.

4 Embedding Models

A large collection of tweets containing many Arabic affect-related words was retrieved and pre-processed to generate a language model at both character and word level. Word embeddings are learned representations of text, with words of similar meanings represented in similar ways. An essential element of this methodology is the concept of employing dense distributed representations for every word. Here, each word is encoded to a real-valued vector with a few hundred dimensions. Given a large corpus, there are different models and levels available for learning word embeddings. We first used the word2vec

model [29] and fastText model [13] for word- and character-level embeddings, respectively. We leveraged these pre-trained embeddings as an input feature after combining them with a novel concatenation approach. These main steps are detailed in the following subsections.

4.1 Word-level Embeddings (WE)

We used the word2vec algorithm [29] to learn individual words and their embeddings from the harvested data. Word2vec adopts two learning techniques, namely, the continuous bag-of-words (CBOW) and skip-gram (SG) models. The abstract architectures of the CBOW and SG models are shown in Figure 2. Using a simple neural network, the SG model is trained by predicting the words surrounding a given target word and minimises the following loss function:

$$E = -log(p(\vec{w}_t) \mid \vec{W}_t)) \tag{1}$$

where w_t represents the given word, and the words coming before and after the target word (window) are denoted by W_t. All of the inputs and outputs are of the same size and encoded with one-hot coding. The CBOW model works in a similar way, but rather than predict the context on the basis of the target word, it predicts the target word on the basis of the surrounding words .

Both models (CBOW and SG) were trained on the collection of tweets that was retrieved to create affect word embeddings. The Gensim library[6] was used to implement the word2vec models. Every tweet was assumed to represent a sentence, with the input for the word-level model being a list of pre-processed tweets that were tokenised into words. We examined different pre-processing methods to study their impact on the effectiveness of the generated models. One of the primary parameters for training the models was (window), which is the maximum distance between the target word and the surrounding context. We compared different values (3, 5 and 7) to select the parameter value that best improves the performance of the models. We also compared different vector sizes (300, 200 and 100) to study the impact of these factors on the final generated models.

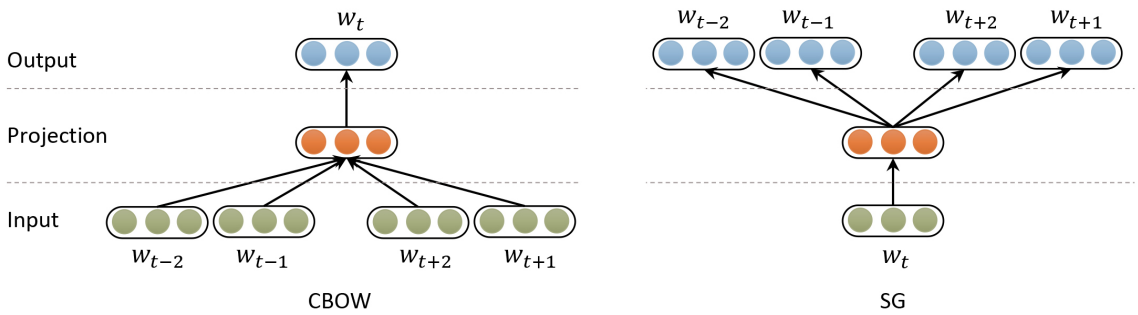

Fig. 2. The general architecture of CBOW and SG models [28].

[6] http://radimrehurek.com/gensim/models/word2vec.html

4.2 Character-level Embeddings (CE)

The wide variation in the form of Arabic dialect words leads to the OOV problem. Therefore, effective resources and tools are needed to better understand and treat these various linguistic forms when targeting affect tasks in Arabic tweets. We employed a character n-grams model (fastText) [13] to learn the morphological features present in each word. FastText differs from word2vec in that it can learn vectors for character n-grams. Thus, fastText can identify words that are similar in meaning but have different word formations. The input for this CE model was a composed of n-grams for each word in a given tweet. For example: the token متنرفز (mtnrfz) was deemed composed of 2- or 3-grams as follows:

'<m', 'mt' , 'tn' , 'nr', 'rf', 'fz', 'z>'. 2-grams
'<mt', 'mtn' , 'tnr' , 'nrf', 'rfz', 'fz>'. 3-grams.

The '<' and '>' are special symbols which were appended to indicate the token start and end. After training the model, we obtained the embeddings for all the n-grams given retrieved tweets. The word representation vector for a given token can be taken by the sum of its n-grams. Using this character-level information enable the model to represent a rare word since it is strongly likely that some of its n-grams can be found in other words.

'<' and '>' are special symbols appended to indicate the token start and end, respectively. After training the model, we obtained the embeddings for all the n-grams given the retrieved tweets. The word representation vector for a given token can be taken by the sum of its n-grams. The use of this character-level information enables the model to represent a rare word, as it is strongly likely that some of its n-grams can be found in other words.

As in the word-level model generation, we examined different pre-processing methods to study their impact on the effectiveness of the generated models. The Gensim library[7] was employed to implement the fastText model. The input of the character-level model was a list of a bag of character n-grams for each tweet. We adopted the identical primary parameters used for WE. Additionally, in order to control character n-gram length, we examined different values of n (2 and 3).

4.3 Affect Character and Word Embeddings (ACWE)

At this stage, we have two pre-trained models: character-level *CE* and word-level *WE*. As explained in the Introduction, while *CE* seems to encode all variants of a word's morphology closely in the embedded space, *WE* seems to give more importance to semantic similarity. To take advantage of both models, we propose ACWE, a novel approach that aims to concatenate these two pre-trained embeddings; hence, it can be used as an input feature for a range of sentiment, emotion and related downstream tasks.

Given a tweet t_i that has a sequence of words $\{w_1, w_2, ..., w_n\}$, our goal is to morphologically and semantically represent each word in each tweet $w_i \in t_i$ as an n-dimensional continuous vector. To achieve this goal, we assumed that each word $w_i \in t_i$ is represented semantically by *WE(w_i)* and morphologically by *CE(w_i)*, where *WE(w_i)* is the word embedding of w_i, while *CE(w_i)* is the character embedding of w_i. The *ACWE(w_i)* method is used to concatenate both embeddings, and it can be obtained in the following cases:

[7] `http://radimrehurek.com/gensim/models/fasttext.html`

$$ACWE(w_i) = \begin{cases} CE(w_i) \bigoplus WE(w_i), & \text{if } w_i \in (CE|V|, WE|V|) \\ CE(w_i) \bigoplus WE(find\_alternative(w_i)), & \text{if } w_i \notin (WE|V|) \\ zeros\ \ of(CE + WE)\ dimensions & \text{otherwise} \end{cases} \quad (2)$$

The first case is a direct concatenation of *CE(w_i)* and *WE(w_i)*, and it arises if w_i can be found in both embeddings. However, if w_i cannot be found in *WE*[8], we assume this is due to variants in the given word's morphology. Consequently, instead of using a vector of zeros for unseen w_i, it will be replaced by another word's morphology that can be realised by *WE*. Alternative words can be obtained using *find_alternative(w_i)*, which aims to find an alternative word to be represented by (w_i). Finally, if w_i cannot be determined using *CE* and *WE*, it will be represented by a vector of zeros.

find_alternative(w_i) is a method that aims to find the most similar word on the basis of 1) the cosine similarity of the w_i vector and the vectors for each word in *CE* and 2) the most similar word that shares the maximum number of letters. To identify the most similar word on the basis of the cosine similarity, we applied the (most_similar) function from Gensim to find the five most similar words. This function was used to compute the cosine similarity between the weight vectors of the given unseen word and the vectors for each word in *CE*. From these potential candidates, which are likely to be different variants of the unseen word, we selected the word that shared the maximum number of characters and could be recognised by *WE*. Table 2 presents three examples of three unseen words that could not be found by *WE*; they were replaced using the *find_alternative(w_i)* method.

Table 2. Examples of unseen words and the steps of how to find the alternative words.

Step	Examples of OOV from WE		
	زعلاتك zElAAtk (your upsets)	ومتحلطمه wmtHlTmh (and feel broken-feminine)	هفضحه hfDHh (will expose him)
The five most similar words using CE	زعلاان zElAAn (upset) زعلاتك zElAtk (your upsets) زعلاانه zElAAnh (upset-feminine) زعلاتي zElAty (my upsets) زعلاتك zElAtk (your upsets)	متحلطمه mtHlTmh (feel broken-feminine) ومتحلطم wmtHlTm ((and feel broken) متحلطم mtHlTm (feel broken) تحلطمه tHlTmh (his broken feeling) لحلطمه lHlTmh (will break his feeling)	هفضح hfDH (will expose) افضحه AfDHh (expose him) هفضحك hfDHk (will expose you) بتفضحه btfDHh (he will be exposed) هتفضح htfDH (will expose)
The selected word that shared the maximum number of characters	زعلاتك zElAtk (your upsets)	متحلطمه mtHlTmh (feels broken-feminine)	هفضح hfDH (will expose)

[8] As explained in the Introduction, WE cannot account for OOV words.

5 Experimental Setup

In this section, we provide information about the datasets used to evaluate our models, the official metrics for each task, and an overview of the state-of-the-art pre-trained Arabic word embeddings compared against our models and supervised learning models used word embedding models as input feature.

5.1 Datasets

We evaluated our models using different affect tasks in the SemEval 2018 task 1 (Affect in Tweets) datasets [31]. We selected these datasets because of the variety of affect tasks and Arabic dialects present in the data. In addition, other related downstream tasks, described below, were used to evaluate the robustness of the models. In total, we used eight datasets in our experiments as follows:

- **Emotion Intensity Regression Task (*EI-reg*):** In this task, there are four subsets for each emotion (anger, fear, sadness and joy). When given an emotion and a tweet, the goal is to determine the emotional intensity (EI) that most accurately is expressed by the target tweet. The data contains 1800 Arabic tweets divided by three sets: a training, dev and test set for each emotion. The EI-reg task was annotated by a real-valus scores, ranging from zero (the lowest intensity) to one (the highest intensity).

- **Emotion Intensity ordinal classification Task (*EI-oc*):** This task is similar to EI-reg, however, it aims at predicting EI classes ranging from 0 to 3, where 0 refers to an unrelated emotion, 1 for the lowest EI and 3 for the highest EI that can be inferred. Table 3 presents the details of the dataset of EI-reg and EI-oc.

Table 3. Number of tweets in *EI-reg* and *EI-oc* datasets and the statistics of the datasets splits.

Task	Emotion	Labels	Train	Dev	Test	Total
EI-reg/ EI-oc	anger	o to 1 (real-value)/ 0,1,2,3 (classes)	877	150	373	1,400
	fear		882	146	372	1,400
	joy		728	224	448	1,400
	sadness		889	141	370	1,400

- **Valence Intensity regression Task (*V-reg*):** When given a tweet, the task was to predict the valence (V) that most effectively represents the tweeter's valance or sentiment using a real-value score. The V-reg task scores range from 0 to 1, from most negative to most positive.

- **Valence Intensity ordinal classification Task (*V-oc*):** The aim in this task is to classify a given tweet to one of the seven class labels, ranging from -3 (very negative) to +3 (very positive), where 0 indicates neutrality. Table 4 presents the details of the dataset of EI-reg and EI-oc.

Table 4. Number of tweets in *V-reg* and *V-oc* datasets and the statistics of the datasets splits.

Task	Labels	Train	Dev	Test	Total
V-reg/V-oc	real-value/ 7 classes	932	138	730	1,800

– **ArSentiment (*ArSen*):** [4] generated using a combination of SemEval's 2017 [41] and ASTD [35] datasets. The dataset that was produced consisted of 10,547 tweets, of which 8,075 were extracted from the SemEval's dataset, and the remaining 2,472 were extracted from ASTD. The extracted tweets were subsequently reannotated into three sentiment classes: positive, negative, or neutral. The tweets were composed in various Arabic dialects and were annotated using Amazon's Mechanical Turk.

– **Arabic Sarcasm detection (*ArSarc*):** The *ArSen* dataset was also used to apply a new annotation that could be used to detect sarcasm. Tweets were labeled with either a sarcasm and not sarcasm tag, where 16% were labelled as being sarcastic (1,682 tweets). Every tweet was examined and annotated by three separate annotators, who achieved an 86.7% agreement level. Table 5 presents an overview of the dataset size and label distribution.

Table 5. Number of tweets in *ArSen* and *ArSarc* tasks and distribution of classes.

Task	Label	Train	Test	Total	Class %
ArSen	Positive	1,362	316	1,678	16%
	Negative	2,813	716	3,529	33%
	Neutral	4,262	1,078	5,340	51%
ArSarc	False	7,100	1,765	8,865	84%
	True	1,337	345	1,682	16%

– **Arabic Offensive Language identification (*Off*):** The dataset consisted of 10,000 tweets that were extracted using the Twitter API between April and May 2019. A native speaker who was knowledgeable in multiple dialects manually annotated the extracted tweets using the 'OFF' (offensive) and 'NOT OFF' (not offensive) labels. The dataset was split into three sub-groups: training (70% of the tweets), dev (10% of the tweets), and test (20% of the tweets) [34,47].

– **Arabic Hate Speech detection (*HS*):** *Off* dataset also was employed to develop a new annotation for the detection of hate speech. Hate speech tweets were deemed to be those that contained threats or insults that were leveraged at a group of people due to their gender, nationality, religion, sports affiliations, ethnicity, political affiliations, or other common characteristics. Tweets were labelled with 'HS' if they were deemed

to contain hate speech and 'NOT HS' if they did not contain hate speech. The way in which the targeted classes were distributed were highly imbalanced; only 5% of the tweets were labelled as containing hate speech [34]. Table 6 presents an overview of the dataset and the associated label distribution.

Table 6. Number of tweets in *Off* and *HS* tasks and distribution of classes.

Task	Label	Train	Dev	Test	Total	Class %
Off	NOT OFF	5,590	821	1,598	8,009	80%
	OFF	1,410	179	402	1,991	20%
HS	NOT HS	6,639	956	1,899	9,494	95%
	HS	361	44	101	506	5%

5.2 Evaluation Metrics

For each aforementioned dataset, we follweed the evaluation metric provided by the organisers. The metrics used for evaluating our models over these eight datasets are as follows:

- **Pearson:** Pearson's correlation coefficient aims to calculate the correlation between the score predicted by our system and the score given by the test data. Pearson is the official metric for the affect tasks (*V-oc*, *V-reg*, *EI-oc* and *EI-reg*). For EI tasks, we calculated the average (macro-average) for all four emotions to obtain the final result for each task.

- **Macro F1-score:** F1 can be interpreted as a weighted average of precision and recall, where 1 refers the best result and 0 for the worst one. Macro F1-score calculates the average of the F1 score of each class. We adopt the same official metric provided by the organisers and researchers for tasks: *ArSen*, *ArSarc*, *offens* and *HS*.

5.3 Pre-trained Word Embeddings

To evaluate the effectiveness of WE, CE and ACWE, we used three Arabic pre-trained word embeddings which are (to the best of our knowledge) the most commonly available resources released to the research community as free to use as the following:

- **Ara2Vec [45]:** Ara2Vec consists of six different word embedding models derived from different sources. These are word-level models that aim to learn word representations for general NLP tasks. We employed the model that was trained on Twitter data because our downstream tasks contained tweets.

- **Mazajak [3]:** Mazajak is a word-level model, and it is considered the largest Arabic word-level embedding . A total of 250 million Arabic tweets were used to build this language model.

– **Altwyan [8]:** They trained their model using a corpus from a variety of public text contents, most of which were news articles (150 million words) and consumer reviews (40 million words).

Table 7 presents a summary of important information about each of these models with their sizes and pre-trained corpora.

Table 7. Different pre-trained Arabic word embeddings used for experimental evaluation.

Model	No. of Words	Corpus	Size
Ara2Vec	4,347,845	General - Twitter	77M Tweets
Mazajak	1,476,715	Sentiment - Twitter	250M Tweets
Altwaian	159,175	Sentiment - Twitter	190M words
Our generated Arabic word Embeddings			
WE	626,212	Affect - Twitter	10M Tweets
CE	441,025	Affect - Twitter	10M Tweets

5.4 Supervised Learning Models

To evaluate the language models, we incorporated them into various supervised learning framework settings for the aforementioned tasks. First, we pre-processed the datasets using the techniques described in Section 3.2. Then, we adopted different approaches to predict real-value scores for the regression tasks and class categories for classification tasks. To this end, we used and compared the following supervised learning models.

– **Logistic Regression:** We used logistic regression as the baseline for comparing the impact of the pre-processing techniques. We then reported the results to consider the use of the best performing methods for more advanced training models.

– **XGBoost:** The extreme gradient boosting (XGBoost) learning model [15] is a state of-the-art method in machine learning [37] for a number of regression tasks. This is an algorithm of decision trees in which new trees correct the errors of trees which are already part of the model. Trees are added to the model until no further changes can be made. Regularisation is incorporated into the XGBoost algorithm to control over-fitting. This model is frequently employed for different problems because it performs excellently on a wide range of significant challenges. In this study, we input tweet vector representations obtained from an average of real-value word vectors for every word with matching vector representations derived from the pre-trained embeddings.

– **CNN:** We adopted the convolutional neural network (CNN) proposed by [22]. In this deep learning architecture, the pre-trained word embeddings were used to initialise the embedding layer weights. These weights were then updated during the training process to make them appropriate for the tasks. Then, different filters and kernels were applied to generate features that were then max pooled. Finally, these features were passed to the fully connected softmax layer for the downstream tasks. Additionally, we used the CNN architecture proposed by [49], which derives various features from different word

embedding models. However, instead of passing word embeddings models separately, we combined them using our ACWE model which were concatenated after multi-group of filters and max-pooling layers. We named this architecture by MG-CNN and it is illustrated in Figure 4.

– **LSTM:** We employed Long Short-Term Memory (LSTM) [20], which is an enhanced form of a recurrent neural network. We used the pre-trained embedding models to initialise the weights of the embedding layer. These weights were updated during training to fine-tune them to each task, and this was connected to the rest of the layers in the networks. Finally, a dense layer with one output was introduced by exploiting a sigmoid activation function. For all the other layers of the network, the ReLU activation function was utilised. We used the Adam optimiser as an optimisation function for the network.

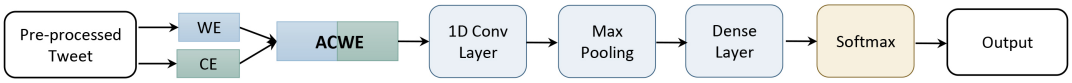

Fig. 3. The general architecture of CNN model.

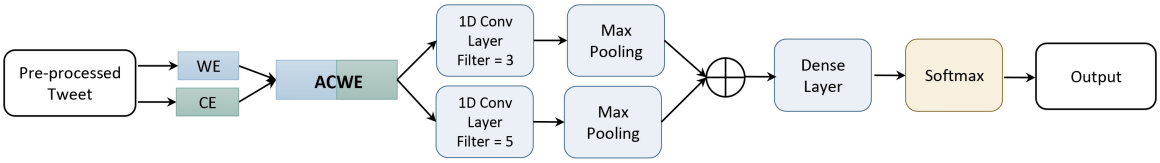

Fig. 4. The general architecture of MG-CNN model.

Fig. 5. The general architecture of LSTM model.

Table 8. Hyper-parameters utilised for deep learning models.

Model	No. Filters	Filter size	Hidden units	Recurrent dropout	Output dropout
CNN	300	3	256	-	-
MG-CNN	200	3 - 5	64	-	0.5
LSTM	-	-	128	0.2	0.2

6 Results

The results of our experiments were evaluated using Pearson's correlation coefficient and F-measure metrics based on the official metrics for each task. Our results and findings are discussed in the following subsections.

6.1 Effect of Pre-processing Techniques

One of the primary aims of this research is to study the impact of pre-processing techniques on downstream affect tasks. We compared the effect of applying individual pre-processing methods or groups thereof to both stages , as illustrated in Figure 1. Since raw tweets usually contain noisy data, we essentially applied clean as the default pre-processing method. The other pre-processing methods (Section 3.2) were applied besides clean to investigate different scenarios of the pre-processing methods.

The results of our experiments are presented in Table 9. For all the models, the use of norm, elong and hashSeg individually with clean can lead to larger improvements across all datasets compared with those of emoji and stem. Emojis are an important element and convey meanings in affect tasks; therefore omitting them has a negative impact on the results. Although stemming words improved results in previous works in English and Standard Arabic, they did not show a positive effect in our experiments. We believe this is because current stem tools cannot handle Arabic dialects. This can explain why incorporation of all pre-processing methods negatively affected the performance of our models across all the datasets. In particular, application of a simple pre-processing method (clean) alone produced better results compared with those of the combination of all methods. Integration of clean, norm, elong and hashSeg improved the results by an average of 2.5% across all datasets. Finally, the results showed no considerable performance difference between the application of the pre-processing methods in WE, CE and ACWE.

6.2 Comparison with State-of-the-Art Pre-Trained Arabic Word Embeddings

We compared five pre-trained word embeddings (Table 10 and 11), namely, three open-source models and both of our generated models. In addition, we compared these models with the ACWE method. The information presented in Table 4 shows the effectiveness of each model in the supervised framework of performing affect-sensitive tasks. The Pearson's correlation coefficient for *CE* significantly outperformed those of the other models. We believe that the main reason for this was associated with OOV problems. Although these models were trained using a massive corpus, the word-level embeddings could not

Table 9. Performance results for evaluating the impact of pre-processing techniques using WE and CE models cross eight datasets.

Models	Pre-processing	EI-oc	EI-reg	V-oc	V-reg	ArSen	ArSarc	HS	Off	**avg.**
WE	clean	0.444	0.547	0.676	0.663	0.629	0.609	0.672	0.832	0.634
	clean+norm	0.461	0.541	0.700	0.679	0.619	0.611	0.661	0.840	0.639
	clean+elong	0.462	0.557	0.717	0.654	0.633	0.608	0.657	0.846	0.642
	clean+hashSeg	0.452	0.564	0.706	0.661	0.638	0.615	0.672	0.832	0.643
	clean+emoji	0.442	0.534	0.635	0.603	0.613	0.586	0.681	0.840	0.617
	clean+stem	0.458	0.529	0.677	0.632	0.608	0.588	0.636	0.825	0.619
	all	0.443	0.524	0.682	0.626	0.610	0.596	0.620	0.833	0.617
	clean+norm+elong+hashSeg	0.512	0.554	0.712	0.686	0.637	0.615	0.680	0.847	0.655
CE	clean	0.487	0.561	0.705	0.675	0.646	0.640	0.694	0.858	0.658
	clean+norm	0.503	0.539	0.722	0.691	0.657	0.647	0.690	0.857	0.663
	clean+elong	0.495	0.557	0.713	0.678	0.657	0.647	0.721	0.864	0.667
	clean+hashSeg	0.483	0.552	0.724	0.692	0.654	0.648	0.714	0.858	0.665
	clean+emoji	0.477	0.529	0.685	0.638	0.654	0.633	0.710	0.863	0.649
	clean+stem	0.479	0.489	0.708	0.663	0.630	0.621	0.643	0.860	0.637
	all	0.483	0.503	0.704	0.656	0.642	0.624	0.634	0.855	0.638
	clean+norm+elong+hashSeg	0.538	0.557	0.745	0.695	0.660	0.656	0.707	0.869	0.678

realise more than 1200 words from each dataset. Nonetheless, the ACWE method improved the results by 1.3% to 5% across all datasets. This indicates the effectiveness of the proposed method and the importance of leveraging character-level and word-level embeddings in Arabic words in the context of social networks and microblogs.

6.3 Comparison with Various Machine and Deep Learning Algorithms

We conducted experiments to compare the results of using ACWE as an input feature into different machine and deep learning approaches. Table 12 presents the experiments' results on the eight datasets using the official metrics for each task. For the affect tasks, XGBoost achieved the highest result, followed by LSTM. As seen in Table 12, the deep leaning methods performed poorly in the affect tasks compared with the other tasks. This was because the datasets of the affect tasks were small (around 800 tweets for training). Deep learning methods need more data to perform well, which was the case particularly for the tasks sar, Off and HS.

6.4 Comparison with Top Systems Analysing Affect in Tweets

Most of the top-performing systems proposed for this shared task employed ensemble approaches to combine different machine and deep learning models. The majority of these systems employed models based on hand-engineered features, such as the sentiment and emotional lexicons found in the Arabic language. In our work, we only used our embedding models as the input feature for XGBoost, a machine learning classifier or regressor. As shown in Table 13, we achieved competitive results. We outperformed the

Table 10. Pearson correlation coefficient results for our models and state-of-the-art pre-trained Arabic Word Embeddings

Model	EI-reg					EI-oc					V-reg	V-oc
	anger	fear	joy	sad	avg.	anger	fear	joy	sad	avg.		
Ara2Vec	0.556	0.536	0.688	0.641	0.605	0.472	0.526	0.604	0.594	0.549	0.773	0.723
Mazajak	0.555	0.576	0.683	0.623	0.609	0.450	0.512	0.646	0.530	0.534	0.720	0.680
Altwyan	0.297	0.333	0.449	0.497	0.415	0.272	0.312	0.425	0.489	0.375	0.515	0.535
Our generated Arabic word Embeddings												
WE	0.539	0.529	0.653	0.607	0.587	0.479	0.511	0.628	0.556	0.544	0.756	0.702
CE	0.601	0.595	0.704	0.658	0.643	0.511	0.531	0.647	0.606	0.576	0.783	0.731
ACWE	**0.638**	**0.622**	**0.758**	**0.686**	**0.676**	**0.543**	**0.572**	**0.675**	**0.609**	**0.600**	**0.818**	**0.767**

Table 11. F-measure results for our models and state-of-the-art pre-trained Arabic Word Embeddings

Model	ArSen	ArSarc	Off	HS
Ara2Vec	0.665	0.638	0.855	0.685
Mazajak	0.653	0.634	0.847	0.689
Altwyan	0.569	0.524	0.700	0.628
Our generated Arabic word Embeddings				
WE	0.647	0.625	0.857	0.690
CE	0.660	0.646	0.859	0.697
ACWE	**0.671**	**0.659**	**0.864**	**0.735**

Table 12. Performance results for using ACWE as an input feature in varied machine and deep learning algorithms cross eight datasets.

Algorithm	EI-oc	EI-reg	V-oc	V-reg	ArSen	ArSarc	Off	HS
XGBoost	**0.600**	**0.676**	**0.767**	**0.818**	0.659	0.659	0.864	0.735
CNN	0.278	0.313	0.458	0.488	0.644	0.676	0.785	0.691
LSTM	0.568	0.640	0.648	0.691	**0.671**	**0.688**	**0.878**	0.701
MG-CNN	0.448	0.505	0.690	0.736	0.636	0.664	0.860	**0.756**

top system in the EI-oc task by 1.3% and ranked second in the remaining tasks. Our goal was not to fully address affect tasks but to demonstrate that the use of a well-generated word embedding model could yield competitive results. We will investigate other features and employ ensemble methods to improve the results in future works.

Table 13. Pearson correlation coefficient results for our ACWE and top systems across all tasks.

Task	1st best	2nd best	Our ACWE
Ei-reg	**0.685**	0.667	0.676
EI-oc	0.587	0.574	**0.600**
V-reg	**0.828**	0.816	0.818
V-oc	**0.809**	0.752	0.767

7 Discussion

Our paper aims to take advantage of both character- and word-level models to discover effective methods of obtaining better representations for affect in tweets in Arabic dialects. To achieve this, we built a large corpus containing a variety of affect words and Arabic dialects. We systematically compared different pre-processing techniques to examine their effect on the effectiveness of the generated word embedding models. Finally, we employed different machine and deep learning algorithms to evaluate our models using eight downstream tasks.

Our experiments with our generated models and off-the-shelf embeddings show the importance of leveraging affect-specific word embedding models as well as the ability of character-level models to overcome the OOV problem. From our observation, about 5%–10% of the words in each dataset could not be identified by the word-level embeddings. Most of these words were Arabic dialects or misspellings, which are common among user-generated text in social media.

Our experiments with different pre-processing techniques show the importance of applying simple methods to clean noisy data (such as user mentions and none Arabic letters). Moreover, emojis and hashtag words are useful and can convey valuable information for model training. Therefore, these words should be segmented instead of being removed. Although stemming words provided better results in MSA, in our work, the stem method negatively impacted the performance of the models.

Future research directions on Arabic affect in tweets are listed as follows.

- Given the success of contextualised word embedding models (BERT [16] as an example) in different NLP tasks, these sophisticated models can be trained on our collected data to improve results.

- Training BERT from scratch is time consuming, and off-the-shelf models (such as AraBERT) may not perform well because such models usually are trained on MSA. Therefore, one possible direction is to enhance these large models with our generated models to target Arabic affect in tweets [42].

– Multi-task learning (MTL) is an approach to inductive transfer that enhances generalisation by using the domain knowledge found in similar tasks as an inductive bias. At present, most studies regard Arabic tasks as individual tasks. Exploiting the relationship between the different affect tasks may enhance findings.

8 Conclusion

In this paper, we take advantage of both character-level and word-level models to discover more effective means of representing Arabic affect in tweets, which we call Affect Character and Word Embeddings (ACWE). We first trained both levels of models on a massive number of tweets, which were collected carefully to ensure that there was significant variation of affect and Arabic dialects in the words. We then employed a novel method that concatenates both levels of models to represent each word morphologically and semantically. We evaluated the effectiveness of our ACWE model by applying it only as a feature under a supervised learning, using eight datasets for affect tasks and related tasks. Our method advances a state-of-the-art approach to the task of discerning Arabic emotional intensity, outperforming the top-performing systems. In addition, our method achieves better results compared to other Arabic pre-trained word embeddings. ACWE has been released to be used in pre-trained word embeddings for applications and research relying on Arabic sentiment and emotion analysis[9].

In future works, we will apply more sophisticated algorithms to improve the quality of our embeddings. Especially, we would like to employ contextualised word embeddings, such as BERT [16]. We would also like to investigate ensemble models to fully target affect tasks.

References

1. Abdou, M., Kulmizev, A., Ginés i Ametllé, J.: AffecThor at SemEval-2018 task 1: A cross-linguistic approach to sentiment intensity quantification in tweets. In: Proceedings of The 12th International Workshop on Semantic Evaluation. pp. 210–217. Association for Computational Linguistics, New Orleans, Louisiana (2018)
2. Abdullah, M., Shaikh, S.: TeamUNCC at SemEval-2018 task 1: Emotion detection in English and Arabic tweets using deep learning. In: Proceedings of The 12th International Workshop on Semantic Evaluation. pp. 350–357. Association for Computational Linguistics, New Orleans, Louisiana (2018)
3. Abu Farha, I., Magdy, W.: Mazajak: An online Arabic sentiment analyser. In: Proceedings of the Fourth Arabic Natural Language Processing Workshop. pp. 192–198. Association for Computational Linguistics, Florence, Italy (2019)
4. Abu Farha, I., Magdy, W.: 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. pp. 32–39. European Language Resource Association, Marseille, France (May 2020)
5. Al-Ayyoub, M., Khamaiseh, A.A., Jararweh, Y., Al-Kabi, M.N.: A comprehensive survey of Arabic sentiment analysis. Information Processing & Management **56**(2), 320–342 (2019)
6. Alswaidan, N., Menai, M.E.B.: Hybrid feature model for emotion recognition in Arabic text. IEEE Access **8**, 37843–37854 (2020)

[9] `https://github.com/nlppaper/ACWE`. This is a temporary link for blind reviews, it will be updated later with the official one.

7. Altowayan, A.A., Elnagar, A.: Improving Arabic sentiment analysis with sentiment-specific embeddings. In: 2017 IEEE International Conference on Big Data (Big Data). pp. 4314–4320. IEEE, Boston (2017)
8. Altowayan, A.A., Tao, L.: Word embeddings for Arabic sentiment analysis. In: 2016 IEEE International Conference on Big Data (Big Data). pp. 3820–3825. IEEE, Washington (2016)
9. Arslan, Y., Küçük, D., Birturk, A.: Twitter sentiment analysis experiments using word embeddings on datasets of various scales. In: Silberztein, M., Atigui, F., Kornyshova, E., Métais, E., Meziane, F. (eds.) Natural Language Processing and Information Systems. pp. 40–47. Springer International Publishing, Cham (2018)
10. Babanejad, N., Agrawal, A., An, A., Papagelis, M.: A comprehensive analysis of preprocessing for word representation learning in affective tasks. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 5799–5810. Association for Computational Linguistics, Online (Jul 2020)
11. Badaro, G., Baly, R., Hajj, H., Habash, N., El-Hajj, W.: A large scale Arabic sentiment lexicon for Arabic opinion mining. In: Proceedings of the EMNLP 2014 workshop on Arabic natural language processing (ANLP). pp. 165–173 (2014)
12. Badaro, G., El Jundi, O., Khaddaj, A., Maarouf, A., Kain, R., Hajj, H., El-Hajj, W.: EMA at SemEval-2018 task 1: Emotion mining for Arabic. In: Proceedings of The 12th International Workshop on Semantic Evaluation. pp. 236–244. Association for Computational Linguistics, New Orleans, Louisiana (Jun 2018)
13. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics **5**, 135–146 (2017)
14. Çano, E., Morisio, M.: Quality of word embeddings on sentiment analysis tasks. In: Natural Language Processing and Information Systems. pp. 332–338. Springer International Publishing, Cham (2017)
15. Chen, T., Guestrin, C.: XGBoost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 785–794. KDD '16, Association for Computing Machinery, New York, NY, USA (2016)
16. 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). pp. 4171–4186. Association for Computational Linguistics, Minneapolis (2019)
17. Devlin, J., Zbib, R., Huang, Z., Lamar, T., Schwartz, R., Makhoul, J.: Fast and robust neural network joint models for statistical machine translation. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1370–1380. Baltimore, Maryland (2014)
18. Eskander, R., Rambow, O.: SLSA: A sentiment lexicon for Standard Arabic. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 2545–2550. Association for Computational Linguistics, Lisbon, Portugal (Sep 2015)
19. Hifny, Y., Ali, A.: Efficient Arabic emotion recognition using deep neural networks. In: ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 6710–6714 (2019)
20. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)

21. Jabreel, M., Moreno, A.: EiTAKA at SemEval-2018 task 1: an ensemble of n-channels ConvNet and XGboost regressors for emotion analysis of tweets. In: Proceedings of The 12th International Workshop on Semantic Evaluation. pp. 193–199. Association for Computational Linguistics, New Orleans, Louisiana (2018)
22. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1746–1751. Association for Computational Linguistics, Doha, Qatar (Oct 2014)
23. Kralj Novak, P., Smailović, J., Sluban, B., Mozetič, I.: Sentiment of emojis. PLoS ONE **10**(12), 1–22 (2015)
24. Lei, Z., Yang, Y., Yang, M., Liu, Y.: A multi-sentiment-resource enhanced attention network for sentiment classification. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). pp. 758–763. Association for Computational Linguistics, Melbourne, Australia (Jul 2018)
25. Li, Q., Shah, S., Liu, X., Nourbakhsh, A.: Data sets: word embeddings learned from tweets and general data. In: Proceedings of the Eleventh International Conference on Web and Social Media (ICWSM-17). pp. 428–436. AAAI Press, Montréal, Canada (2017)
26. Li, X., Rao, Y., Xie, H., Liu, X., Wong, T.L., Wang, F.L.: Social emotion classification based on noise-aware training. Data Knowledge Engineering **123**, 101605 (2019)
27. Mahmoud, A., Zrigui, M.: Deep neural network models for paraphrased text classification in the Arabic language. In: Métais, E., Meziane, F., Vadera, S., Sugumaran, V., Saraee, M. (eds.) Natural Language Processing and Information Systems. pp. 3–16. Springer International Publishing, Cham (2019)
28. Mikolov, T., Corrado, G., Chen, K., Dean, J.: Efficient Estimation of Word Representations in Vector Space. Proceedings of the International Conference on Learning Representations (ICLR 2013) pp. 1–12 (2013)
29. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2. p. 3111–3119. NIPS'13, Curran Associates Inc., Red Hook, NY, USA (2013)
30. Mohammad, S., Bravo-Marquez, F.: Emotion intensities in tweets. In: Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017). pp. 65–77. Association for Computational Linguistics, Vancouver, Canada (Aug 2017)
31. Mohammad, S., Bravo-Marquez, F., Salameh, M., Kiritchenko, S.: SemEval-2018 task 1: Affect in Tweets. In: Proceedings of The 12th International Workshop on Semantic Evaluation. pp. 1–17. Association for Computational Linguistics, New Orleans, Louisiana (2018)
32. Mohammad, S.M.: Word affect intensities. In: Proceedings of the 11th Edition of the Language Resources and Evaluation Conference (LREC-2018). pp. 174–183. Miyazaki, Japan (2018)
33. Mourad, A., Darwish, K.: Subjectivity and sentiment analysis of modern standard Arabic and Arabic microblogs. In: Proceedings of the 4th workshop on computational approaches to subjectivity, sentiment and social media analysis. pp. 55–64 (2013)
34. Mubarak, H., Darwish, K., Magdy, W., Elsayed, T., Al-Khalifa, H.: Overview of OSACT4 Arabic offensive language detection shared task. In: Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection. pp. 48–52. European Language Resource Association, Marseille, France (May 2020)

35. Nabil, M., Aly, M., Atiya, A.: ASTD: Arabic sentiment tweets dataset. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 2515–2519. Association for Computational Linguistics, Lisbon, Portugal (Sep 2015)
36. Obeid, O., Zalmout, N., Khalifa, S., Taji, D., Oudah, M., Alhafni, B., Inoue, G., Eryani, F., Erdmann, A., Habash, N.: CAMeL tools: An open source python toolkit for Arabic natural language processing. In: Proceedings of the 12th Language Resources and Evaluation Conference. pp. 7022–7032. European Language Resources Association, Marseille, France (May 2020)
37. Orzechowski, P., La Cava, W., Moore, J.H.: Where are we now? a large benchmark study of recent symbolic regression methods. In: Proceedings of the Genetic and Evolutionary Computation Conference. p. 1183–1190. GECCO '18, Association for Computing Machinery, New York, NY, USA (2018)
38. Parker, R., Graff, D., Chen, K., Kong, J., Maeda, K.: Arabic gigaword fifth edition robert parker, david graff, ke chen, junbo kong, kazuaki maeda, `https://catalog.ldc.upenn.edu/LDC2011T11`
39. Pennington, J., Socher, R., Manning, C.: GloVe: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543. Association for Computational Linguistics, Doha, Qatar (Oct 2014)
40. Qu, L., Ferraro, G., Zhou, L., Hou, W., Schneider, N., Baldwin, T.: Big data small data, in domain out-of domain, known word unknown word: The impact of word representations on sequence labelling tasks. In: Proceedings of the Nineteenth Conference on Computational Natural Language Learning. pp. 83–93. Association for Computational Linguistics, Beijing, China (Jul 2015)
41. Rosenthal, S., Farra, N., Nakov, P.: SemEval-2017 task 4: Sentiment analysis in Twitter. In: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017). pp. 502–518. Association for Computational Linguistics, Vancouver, Canada (Aug 2017)
42. Roy, A., Pan, S.: Incorporating extra knowledge to enhance word embedding. In: Bessiere, C. (ed.) Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20. pp. 4929–4935. International Joint Conferences on Artificial Intelligence Organization (7 2020)
43. dos Santos, C., Gatti, M.: Deep convolutional neural networks for sentiment analysis of short texts. In: Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers. pp. 69–78. Dublin City University and Association for Computational Linguistics, Dublin, Ireland (Aug 2014)
44. Singh, T., Kumari, M.: Role of text pre-processing in Twitter sentiment analysis. Procedia Computer Science **89**, 549–554 (2016)
45. Soliman, A.B., Eissa, K., El-Beltagy, S.R.: AraVec: A set of Arabic word embedding models for use in Arabic NLP. Procedia Computer Science **117**, 256–265 (2017)
46. Zahran, M.A., Magooda, A., Mahgoub, A.Y., Raafat, H., Rashwan, M., Atyia, A.: Word representations in vector space and their applications for Arabic. In: Gelbukh, A. (ed.) Computational Linguistics and Intelligent Text Processing. pp. 430–443. Springer International Publishing, Cham (2015)
47. Zampieri, M., Nakov, P., Rosenthal, S., Atanasova, P., Karadzhov, G., Mubarak, H., Derczynski, L., Pitenis, Z., Çöltekin, Ç.: SemEval-2020 task 12: Multilingual offensive language identification in social media (OffensEval 2020). In: Proceedings of the Fourteenth Workshop on Semantic Evaluation. pp. 1425–1447. International Committee for Computational Linguistics, Barcelona (online) (Dec 2020)

48. Zhang, J., Liu, S., Li, M., Zhou, M., Zong, C.: Bilingually-constrained phrase embeddings for machine translation. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 111–121. Association for Computational Linguistics, Baltimore, Maryland (2014)
49. Zhang, Y., Roller, S., Wallace, B.C.: MGNC-CNN: A simple approach to exploiting multiple word embeddings for sentence classification. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 1522–1527. Association for Computational Linguistics, San Diego, California (Jun 2016)