# Multi-Channel Embedding Convolutional Neural Network Model for Arabic Sentiment Classification

ABDELGHANI DAHOU, SHENGWU XIONG, JUNWEI ZHOU, and MOHAMED ABD ELAZIZ, Wuhan University of Technology, China

With the advent of social network services, Arabs' opinions on the web have attracted many researchers in recent years toward detecting and classifying sentiments in Arabic tweets and reviews. However, the impact of word embeddings vectors (WEVs) initialization and dataset balance on Arabic sentiment classification using deep learning has not been thoroughly studied. In this article, a multi-channel embedding convolutional neural network (MCE-CNN) is proposed to improve Arabic sentiment classification by learning sentiment features from different text domains, word, and character n-grams levels. MCE-CNN encodes a combination of different pre-trained word embeddings into the embedding block at each embedding channel and trains these channels in parallel. Besides, a separate feature extraction module implemented in a CNN block is used to extract more relevant sentiment features. These channels and blocks help to start training on high-quality WEVs and fine-tuning them. The performance of MCE-CNN is evaluated on several standard balanced and imbalanced datasets to reflect real-world use cases. Experimental results show that MCE-CNN provides a high classification accuracy and benefits from the second embedding channel on both standard Arabic and dialectal Arabic text, which outperforms state-of-the-art methods.

CCS Concepts: • **Information systems** → **Sentiment analysis**; **Language models**; • **Computing methodologies** → **Neural networks**;

Additional Key Words and Phrases: Arabic language, Arabic sentiment classification, Arabic word embeddings, neural language models, convolutional neural network, multi-channel, deep learning

**ACM Reference format:**
Abdelghani Dahou, Shengwu Xiong, Junwei Zhou, and Mohamed Abd Elaziz. 2019. Multi-Channel Embedding Convolutional Neural Network Model for Arabic Sentiment Classification. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* 18, 4, Article 41 (May 2019), 23 pages.
https://doi.org/10.1145/3314941

## 1 INTRODUCTION

Currently, people get used to expressing their opinions on all sorts of trending platforms such as social networking services, blogs, and e-commerce websites, generating vast amounts of opinion

This work was in part supported by the National Key Research and Development Program of China (Grant No. 2017YFB1402203), the National Natural Science Foundation of China (Grant No. 61601337), the Defense Industrial Technology Development Program (Grant No. JCKY2018110C165), Hubei Provincial Natural Science Foundation of China (Grant No. 2017CFA012), and the Key Technical Innovation Project of Hubei Provence of China (Grant No. 2017AAA122).
Authors' addresses: A. Dahou, S. Xiong (corresponding author), J. Zhou, and M. A. Elaziz, School of Computer Science and Technology, Wuhan University of Technology, 122 Luoshi Road, Wuhan, Hubei, 430070, China; emails: {dahou, xiongsw}@whut.edu.cn, junweizhou@msn.com, abd_el_aziz_m@yahoo.com.

2375-4699/2019/05-ART41 $15.00
https://doi.org/10.1145/3314941

data. Since the interest of a wide range of real-world applications in harvesting public opinion, sentiment analysis (SA), and its applications in opinionated web content has attracted much attention and leads to one of the most growing research fields in natural language processing (NLP). SA, or opinion mining, is the study of subjective information conveyed in an opinion expressed by a writer, whether at the document level (Yessenalina et al. 2010), the sentence level (Al Sallab et al. 2015), the word/term level (Engonopoulos et al. 2011), or the aspect level (Chifu et al. 2015; Mohammad et al. 2017).

Sentiment classification (SC) is a fundamental task of SA, which aims at automatically determining whether an opinionated text is being positive, negative, or neutral. Recently, many types of research have been dedicated to SC targeting English and other European languages. However, few studies focus on the Arabic language (ElSahar and El-Beltagy 2015; Nabil et al. 2015). As one of six official and working languages in the United Nations, Arabic has ranked the fifth most-spoken language worldwide. Recent statistics show that more than 422 million people speak Arabic,[1] with more than 290 million as the first language, and 132 million as the second language. On the web, with a growth rate in the number of Internet users of 7,737% within the period (2000-2017),[2] Arabic is ranked the fourth among the top 10 languages used on the Internet. On the other part, Arabic is considered as a Semitic language and a morphologically rich language (Habash 2010). Roots and patterns are a formal representation of Arabic morphology. The root is a sequence of letters consisting of only consonants or long vowels. Moreover, the root is the base of all Arabic words that holds the base meaning of the word. Patterns are used to create a variety of related words by attaching vowels and non-root consonants to the root letters (Neme and Laporte 2013). In addition, there are some differences between Arabic and English that make the proposed method more suitable for Arabic. Mainly, we focus on replacing the need of complex models used during the preprocessing of Arabic text, handcrafted feature engineering, and the need of domain-specific Arabic lexicons by a deep learning SC system that can be more accurate and less time and resource consuming. For further illustration, Arabic words are derived from a root containing three or more letters. The high derivational and inflectional nature of the Arabic language can affect the performance of the model, where one or more root letters may be dropped during the derivation process. Besides, matching the derived words with their root is a majorly difficult problem in Arabic as compared to English. The derivational problem has been tackled in the proposed method using word representations built under different character *n*-grams levels that learn different root lengths as well as word levels. English text preprocessing such as stemming can be applied by removing well-known prefixes and suffixes based on a predefined list or simple extraction rules. In contrast, Arabic possesses a complex morphology (Baly et al. 2017) that can demand more complex rules and patterns to perform stemming or other preprocessing such as part-of-speech tagging (Habash 2010). Keyword matching is inadequate for Arabic classification due to the widespread of Arabic synonyms, which creates an ambiguity (Al-Smadi et al. 2018). In the proposed method, to overcome the preprocessing and keyword matching issues, different convolutional neural network (CNN) blocks and embedding channels trained on general and domain-specific pre-trained word embeddings (PWEs) are joined to learn more relevant features such as domain-specific sentiments. Therefore, developing an accurate SC system for Arabic opinions using deep learning approaches is a challenging task that has vast potential worth investigating.

Deep learning is one of the most popular research fields that has attracted many researchers from the artificial intelligence community. Deep neural networks (DNNs) are known for their

[1] http://www.ethnologue.com/statistics/size.
[2] http://www.internetworldstats.com/stats7.htm.

capability to learn features automatically and refine them during the training. However, the inappropriate initialization of word embeddings with low-quality or random word embeddings vectors (WEVs) in most of the training scenarios might affect the performance of the trained model (Kim 2014). To overcome this issue, a multi-channel embedding convolutional neural network (MCE-CNN) is proposed in this article as an alternative Arabic sentiment classification model. MCE-CNN improves the performance of classification by initializing the embedding layer with a set of WEVs extracted from different PWEs. PWEs are trained on common web domains data, domain-specific data, as well as from the training dataset itself. Unlike the traditional word embeddings initialization based on random values that do not hold any Semantic-Syntactic features or use only one PWE trained on small data, we encode a combination of WEVs extracted from several PWEs into embedding block on two different embedding channels. To evaluate the impact of WEVs initialization on Arabic SC, we investigate the impact of two WEVs initialization methods that are (1) the initialization using word and character *n*-grams level WEVs extracted from several PWEs and (2) random WEVs initialization. Moreover, the impact of deep learning approaches is investigated where neural language models (NLMs) are used to build PWEs, and a CNN is used as a feature extraction block. The impact of balanced and highly imbalanced shapes of each evaluated dataset is analyzed to assess the performance of our Arabic SC systems.

In the first WEVs initialization method, WEVs are learned from a general or a domain-specific PWE. In addition, WEVs can be learned from a dataset PWE (D-PWE) that is built under different character *n*-grams levels to account for the morphological complexity of Arabic, rare words, and out-of-vocabulary (OOV) words. FastText (Bojanowski et al. 2016) with skip-gram (SG) model is used as an NLM system to train and build D-PWE for each dataset separately. In the second method, WEVs are randomly initialized with values in the interval $[-1, 1]$. The randomly initialized WEVs may result in a low classification accuracy compared with WEVs initialized using a PWE. Furthermore, due to the change of terms used on social media platforms, PWEs must frequently be updated to cover new words from different training datasets vocabularies. To overcome these challenges, MCE-CNN provides a set of solutions that are stated as follows:

(1) Initializing MCE-CNN embedding block with a combination of high-quality WEVs extracted from several PWEs after mapping each word from the dataset to the corresponding WEVs in a PWE. If the word does not exist in PWE, we will tag it as an OOV word.
(2) Rather than re-training a PWE to cover new words or OOV words, D-PWE vectors are encoded into the combined WEVs before training. This helps to cover new and OOV words rather than initializing their embedding vectors randomly.
(3) To minimize the number of OOV WEVs, a vocabulary reduction process is adopted by keeping only the top most frequent words in each dataset.
(4) To improve classification performance and account for domain-specific sentiment features, a training process using a domain-specific WEV on a second embedding channel in MCE-CNN is implemented.
(5) To learn more sentiment features from WEVs, a CNN block is trained with different filter sizes that can capture word level *n*-grams features.

With the proposed MCE-CNN, we can combine different PWEs to cover new and OOV words from different datasets by mapping them to WEVs with high quality. Furthermore, MCE-CNN can be forced to learn additional sentiment features from the text domain of a dataset by using domain-specific WEVs on a second embedding channel and CNN block. To evaluate the performance of MCE-CNN, we performed a set of experiments on various Arabic datasets. Experiments show that

Table 1. List of Abbreviations

| Abbreviation | Definition | Abbreviation | Definition |
|---|---|---|---|
| CBOW | Continuous Bag of Word Model | NLP | Naturel Language Processing |
| C-PWE | Primary PWE trained on web crawled data | NS | Negative Sampling |
| CNN | Convolutional Neural Network | OOV | Out-of-vocabulary Word |
| DAE | Deep Autoencoder | POS | Part-of-Speech |
| DBN | Deep Belief Networks | PWEs | Pre-trained Word Embeddings |
| DNN | Deep Neural Networks | RAE | Recursive Autoencoder |
| D-PWE | Dataset PWE | ReLU | Rectified Linear Unit |
| FC | Fully Connected | RNTN | Recursive Neural Tensor Network |
| GBC | Gradient Boosting Classifier | SA | Sentiment Analysis |
| Glove | Global Vectors Model | SC | Sentiment Classification |
| k-NN | k-nearest Neighbor | SCNN | Single Embedding Channel CNN |
| LR | Logistic Regression | SG | Skip-gram Model |
| LSTM | Long Short-term Memory | SGD | Stochastic Gradient Descent |
| MCE-CNN | Multi-channel Embedding Convolutional Neural Network | SMOTE | Synthetic Minority Over-sampling Technique |
| MLP | Multi-layer Perceptron | SVM | Support Vector Machines |
| MSA | Modern Standard Arabic | T-PWE | Twitter |
| NER | Named Entity Recognition | WEVs | Word Embeddings Vectors |
| NLMs | Neural Language Models | W-PWE | World Wide Web PWE (domain-specific PWE) |

MCE-CNN has a better performance than existing approaches for Arabic sentiment classification tasks on differently balanced and highly imbalanced datasets from two text domains.

The rest of the article is organized as follows. The related work will be introduced in Section 2. Section 3 describes the proposed deep learning architecture for sentiment classification of Arabic text. Experiments and analyses will be given in Section 4. The conclusion will be drawn in Section 5. Furthermore, to make the readability of the article more clear, we have listed the important abbreviations and their corresponding explanations in Table 1.

## 2 RELATED WORK

In this section, existing works related to the sentiment classification focusing on the Arabic language are reviewed. We start by reviewing works using NLMs as word embedding building systems. NLMs represent individual words of a language as word vectors onto a lower dimensional vector space. Then, works related to Arabic sentiment classification that use supervised machine learning or deep learning methods are introduced. These methods can be used to generate semantic and syntactic representations of texts and perform various tasks in NLP.

### 2.1 Neural Language Models

There are several NLMs to construct word embeddings that rely on machine learning and deep learning methods proposed in recently published papers. Mikolov et al. (2013a) proposed Word2vec in which a distributed representation of a word is learned using feedforward neural network. The learned word representations capture semantic-syntactic relationships in a simple way. Word2vec is a predictive model consisting of two learning models that are continuous bag of word (CBOW) and SG. The training objective of CBOW is to maximize the conditional probability of predicting a central word as output given the input context words surrounding it. Whereas, the training

objective of SG is to predict the surrounding words of an input word from training samples. SG model encountered some issues dealing with the large-scale neural network during the training process. The network needs to update weight matrices of hidden and output layers for each training sample. Moreover, the training will be slow if gradient descent is used with a neural network on a large size of word vocabulary. Authors in Mikolov et al. (2013b) extended the SG model to speed up learning by adopting several extensions such as sub-sampling of high-frequency words, negative sampling (NS), and dealing with frequent word pairs or phrases such as persons, city, and country names as a single word and they assigned unique word vectors to these word pairs. SG has been investigated in our previous work (Dahou et al. 2016) to improve Arabic word embeddings by using an alternative phrase mining strategy since infrequent phrases with low quality were present in the Arabic word embeddings. In general, these techniques (i.e., sub-sampling and NS) result in reducing the computation complexity of the learning process and improving the quality WEVs. Authors in Pennington et al. (2014) presented a count-based model to learn word representations named global vectors (Glove). The Glove model applies matrix factorization methods to reduce the dimensionality of the co-occurrence counts matrix. Recently, Bojanowski et al. (2016) proposed FastText, which trains WEVs on subwords units, taking into consideration the morphology of a language. The key difference between the previously mentioned models and FastText is that SG, CBOW, and Glove treat words as the smallest unit during the learning phase. On the other hand, FastText treats each word as composed of character *n*-grams. So the vector for a word can be learned from the sum of its *n*-grams, as well as the complete word. For that, FastText can learn WEVs for rare words rather than ignore them like Word2vec when applying the sub-sampling technique. Hence, it covers more OOV words even if the word does not exist in the training corpus.

The previously mentioned techniques performed well on various NLP tasks based on deep learning approaches. Few attempts have been carried out to learn word embeddings for the Arabic language using the methods mentioned above. Al-Rfou et al. (2013) trained various word embeddings in more than 100 languages including the Arabic language on Wikipedia data using their system Polyglot. They tested the performance of word embeddings on the part of speech (POS) tagging task. Zahran et al. (2015) conducted several experiments to build Arabic word embeddings trained on a large dataset using CBOW, SG, and Glove. They evaluated the quality of these word representations on query expansion for information retrieval and short answer grading. Dahou et al. (2016) focused on building word embeddings with better quality using a sizable web-crawled corpus to perform Arabic sentiment classification. By training these word embeddings on top of a CNN, the results were encouraging. Soliman et al. (2017) proposed a set of different Arabic word embeddings named AraVec, which mainly trained on Tweets (Twt), World Wide Web pages (WWW), and Wikipedia (Wiki) using both CBOW and SG. They performed experiments utilizing these word embeddings on semantic textual similarity task. Elrazzaz et al. (2017) built a benchmark to perform an intrinsic evaluation of the Arabic word embeddings.

Several major key issues are related to the Arabic word embeddings based on the highlighted techniques in the previous paragraph. Starting by the evaluation of the quality of Arabic word embeddings where, in most cases, an intrinsic evaluation and extrinsic evaluation are needed. Zahran et al. (2015) used a translated English benchmark to perform intrinsic evaluation on Arabic word embeddings, which may not be the best strategy to evaluate Arabic embeddings. Dahou et al. (2016) corrected the analogy questions from Zahran et al. (2015) and added new analogy questions to overcome the inadequacy of the English questions translation for the Arabic language. However, intrinsic evaluation can not reflect the quality of Arabic word embeddings unless an extrinsic evaluation is performed such as sentiment classification, query expansion, part-of-speech tagging, and short answer grading. Using Arabic word embeddings to tackle NLP tasks shows promising results on differently available datasets (Al-Azani and El-Alfy 2017a). The lack of the

data needed to build Arabic word embeddings is another major issue that results in few recent studies attempted to build word embeddings for the Arabic language. It is important to use large training data to obtain meaningful embeddings (Elrazzaz et al. 2017).

### 2.2 Arabic Sentiment Classification

Recently, sentiment classification became one of the most motivating research areas among the NLP community. Some tools and applications have been applied to Arabic sentiment classification. For example, Altowayan and Tao (2016) trained six machine learning binary classifiers to perform subjectivity and sentiment detection in both standard and dialectal Arabic. They used word embeddings features extracted from a relatively small data containing around 190 million words to train all classifiers. Al Sallab et al. (2015) investigated the usage of DNN, deep belief networks (DBN), deep autoencoder (DAE), and recursive autoencoder (RAE) to perform Arabic sentiment classification. Bag-of-Words features were used as the input data representation to each model with a combination with standard Arabic sentiment lexicon features. DAE gives a better representation of the input sparse vector and RAE gives the best F1 score among other deep learning methods.

Al-Azani and El-Alfy (2017b) used Syria tweets dataset (STD) as an evaluation dataset to investigate the problem of imbalanced datasets for sentiment polarity determination. They compared several classifiers and ensembles with and without synthetic minority over-sampling technique (SMOTE). They used several machine learning classifiers such as k-nearest neighbor (k-NN), support vector machines (SVM), logistic regression (LR), stochastic gradient descent (SGD), Gaussian naïve Bayes, and decision trees. They used voting, bagging, boosting, stacking, and random forests as ensemble learning techniques, and gradient boosting classifier (GBC) and AdaBoost classifier as instances of boosting ensemble learning. Baly et al. (Al-Sallab et al. 2017) proposed A Recursive Deep Learning Model for Opinion Mining in Arabic (AROMA) as a recursive deep learning framework to overcome the shortage of RAE algorithm handling Arabic language morphological complexity. Furthermore, Baly et al. (2017) used a recursive neural tensor network (RNTN) for Arabic sentiment analysis, where the results show that their RNTN outperforms well-known classifiers such as SVM, RAE, and long short-term memory (LSTM).

Alayba et al. (2017) built a dataset of opinions on health services collected from Twitter. They used DNN and CNN for the classification task. The results of experiments showed that an SVM classifier outperformed deep learning approaches. El-Beltagy et al. (2017) used CNN, LR, and multilayer perceptron (MLP) to tackle three subtasks of SemEval-2017 Task 4 in SemEval-2017 competition. Al-Azani and El-Alfy (2017a) used word embeddings on top of CNN and LSTM. The results of their experiments show that LSTM gets best scores among other proposed models on differently available datasets. Although few types of research were done on Arabic sentiment classification using deep learning techniques, most of these efforts did not account for the effect of word embeddings vectors initialization and quality, domain-specific sentiment features and dataset balance.

## 3 PROPOSED MCE-CNN MODEL

In this section, the proposed MCE-CNN model for Arabic sentiment classification is described. Starting from the data preparation module, input sequences are preprocessed by normalizing Arabic letters and cleaning noisy data such as stop words, diacritics, elongated words, punctuations, numbers, symbols, and Non-Arabic as shown in Figure 1. A vocabulary reduction operation is performed to reduce the number of OOV words and avoiding poor or random initialization of WEVs of these words. At the embedding channels module, preprocessed input sequences are mapped to different PWEs to generate WEVs as the input of the embedding layer in each embedding block. The embedding channels module is implemented to fine-tune WEVs trained on general and domain-specific data. Furthermore, the CNN block is implemented to extract sentiment features using

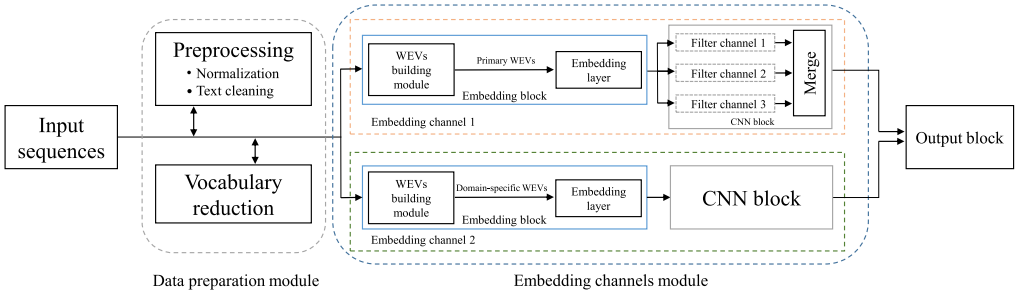



Fig. 1. MCE-CNN framework.

various filter channels. In the end, an output block is used to perform classification and produce the final results.

MCE-CNN consists of five types of layers: embedding, convolution, max-pooling, merge, and fully connected (FC). Also, there are two types of channels: embedding channels and convolution filters channels. Firstly, the preprocessed input sequences are fed to two parallel embedding channels separately. Secondly, a series of WEVs mapped to different PWEs using the WEVs building module are used to initialize the embedding layer on each embedding block where they will be fine-tuned during the training. The CNN block consists of three layers: convolution, max-pooling, and merge. Each convolution filter channel consists of a convolution layer with a unique filter size to extract word $n$-grams sentiment features and a max-pooling layer. The max-pooling layer extracts the max-over convolution layer results, which are intended to be the most important sentiment features. A merge layer at the end of convolution filters channels with a concatenation operation will merge all extracted features. Then, merge, and FC layers act as output block to produce the final classification result. Details of each block in MCE-CNN will be given in the following sections.

### 3.1 Embedding Block

Embedding block consists of the WEVs building module and an embedding layer. PWEs used in our experiments are classified into three classes: primary PWE, domain-specific PWE, and D-PWE to cover general and domain-specific sentiment features. PWEs classes were selected after conducting an experiment, which is described in Section 4.2, where dataset coverage rates, OOV words coverage rates, and PWEs text domains were taken as PWEs classification conditions. MCE-CNN takes a combination of WEVs after mapping the preprocessed input sequences to a primary and domain-specific PWEs as input into the embedding layer in the embedding block. Moreover, OOV words from the primary and domain-specific PWEs are mapped to D-PWE.

Figure 2 shows the process of building D-PWE and generating dataset WEVs used for the embedding layer initialization on each embedding block. First, D-PWE will be built by training FastText on the preprocessed input sequences of a dataset to capture character $n$-grams levels features. On the other hand, each word in the dataset vocabulary is mapped directly to a 300-dimension word vector in primary PWE and domain-specific PWE to generate primary and domain-specific WEVs. Secondly, vectors of words not present in primary and domain-specific PWEs are tagged as OOV WEVs. Each out-of-vocabulary WEV will be mapped to a WEV from D-PWE rather than performing random initialization or ignoring these OOV WEVs. The size of each feature matrix will be $N \times d$, where $N$ is the vocabulary size of each dataset after vocabulary reduction, and $d$ is the word vector dimension. With such a variation in WEVs, MCE-CNN can learn rich sentiment features from both text domains, word, and character $n$-grams levels, respectively.

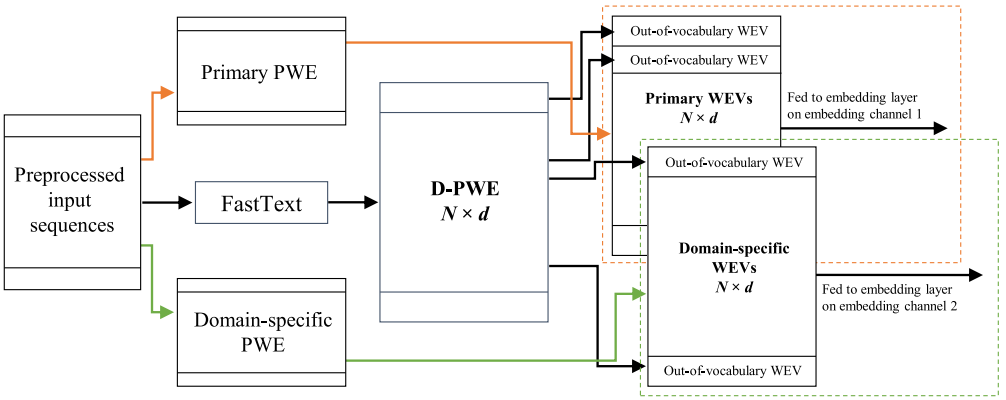



Fig. 2. WEVs building module.

### 3.2 CNN Block

CNN block is a concatenation of convolution and max-pooling layers followed by a merge layer. Three different convolution filter channels are used in the CNN block where each channel consists of convolution and max-pooling operations. Convolution operation on each filter channel is applied using a unique filter size. Convolution operation is followed by a nonlinear activation function using the ReLU (Rectified Linear Unit) function (Nair and Hinton 2010), which enables the extraction of more complex features. Zero padding is applied to limit sequence length to a maximum length before convolution operation. By applying multiple convolving filters, active local word *n*-grams features are extracted in different scales. After that, a max-over-time pooling layer is placed after the convolution layer, which used to perform dimensionality reduction of the output and extract the most salient sentiment features from each feature map. Then, extracted features on each convolution filter channel are fed into a merge layer to generate one single feature vector with a fixed-length.

### 3.3 Output Block

Output block is used to produce the final classification result, which consists of two layers: merge and fully connected. The merge layer is used to join the outputs from the two embedding channels. The output of merge layer is fed into a non-linear fully connected layer with ReLU activation function. The generated vectors from the previous layer are finally inputted to a fully connected layer using sigmoid as an activation function to produce the polarity of a sequence.

Moreover, a regularization operation to prevent overfitting is applied for all evaluated CNNs after the max-over-time pooling layer and the non-linear fully connected layer. Following the work presented in Hinton et al. (2012), values in the feature vectors are randomly set to zero using Dropout. The architecture of MCE-CNN with layers order is given in Figure 3.

## 4 EXPERIMENTAL SETUP

In this section, the performance of MCE-CNN is evaluated on binary classification task by using a set of different datasets, which are described in Section 4.1. PWEs setup and classification is described in Section 4.2. Seven varieties of CNN and their setups including MCE-CNN are introduced in Section 4.3. Performance measures, as well as experimental results and discussions are presented in Sections 4.4 and 4.5, respectively.

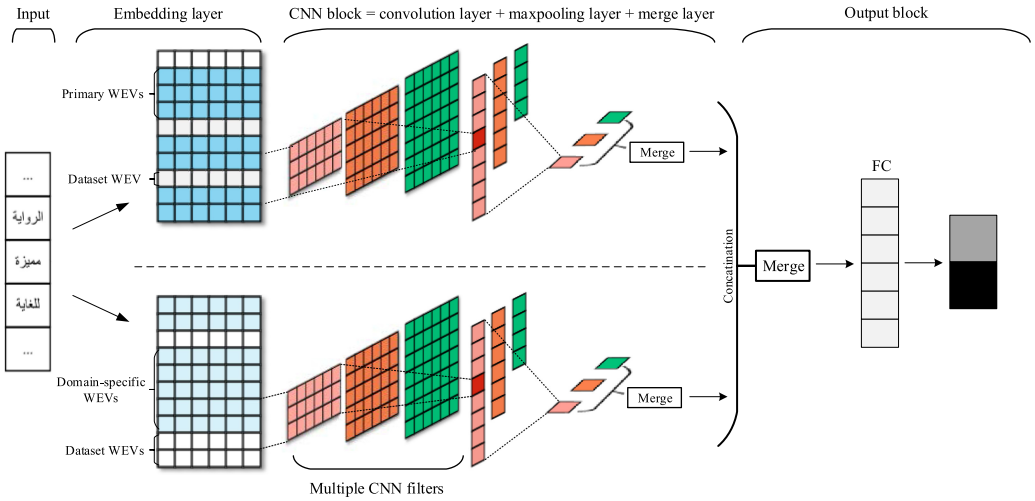



Fig. 3. MCE-CNN architecture.

### 4.1 Sentiment Classification Datasets

In our experiments, two types of corpora based on their text domain are selected: tweets and reviews. To reflect real-world using cases, where most of the datasets are not balanced, we conducted our experiments on two shapes of datasets, balanced and imbalanced. A maximum threshold of training samples is set to extract balanced datasets from original ones, where the maximum threshold is equal to the number of reviews or tweets in the minority class of the dataset. On the other hand, the imbalanced datasets are kept in their original shapes.

Several review datasets, including books, hotels, restaurants, movies, and products reviews and their descriptions are listed in Table 2. Large-scale Arabic Book Review dataset (LABR) book reviews (LABR-R) is a balanced dataset used by Altowayan and Tao (2016) where it has been extracted from the original Arabic book reviews dataset (LABR) (Aly and Atiya 2013). LABR consists of over 63,000 reviews downloaded from Goodreads[3] in 2013. Another dataset has been used in our experiment collected by ElSahar and El-Beltagy (2015) that covers four domains: hotel reviews[4] (HTL), restaurant reviews[5] (RES), movie reviews[6] (MOV), and product reviews[7] (PROD). The dataset includes reviews from Egypt, Saudi Arabia, and the United Arab Emirates and consist of a total of 33K annotated reviews.

The Twitter datasets covering tweets from modern standard Arabic (MSA) and different Arabic dialects were used to build Twitter datasets such as Arabic sentiment tweets dataset (ASTD), ArTwitter, QRCI, STD and Arabic Jordanian general tweets (AJGT). ASTD (Nabil et al. 2015) contains more than 10,000 Arabic tweets for subjectivity and sentiment analysis. A combination of Twitter datasets ASTD, ArTwitter, and QRCI used to create a balanced dataset for binary sentiment classification by Altowayan and Tao (2016) is used in our experiments and we refer to it as AAQ. Syrian tweets Arabic sentiment analysis dataset (STD) (Mohammad et al. 2016) is a manually annotated dataset collected in May 2014 consisting of 2,000 tweets from Syria. AJGT corpus (Alomari et al. 2017) is a manually annotated corpus collected in May 2016. AJGT consists of 1,800

[3]www.goodreads.com.
[4]Source: TripAdvisor.com.
[5]Source: Qaym.com, TripAdvisor.
[6]Source: Elcinemas.com.
[7]Source: Souq.com.

Table 2. Reviews Datasets

| Dataset Name | Size | Sentiments | | | Balance |
|---|---|---|---|---|---|
| | | Positive | Negative | Mixed | |
| LABR-R | 16,448 | 8,224 | 8,224 | - | Balanced |
| Hotel reviews (HTL) | 15,572 | 10,775 | 2,647 | 2,150 | Imbalanced |
| Restaurant reviews (RES) | 10,970 | 8,030 | 2,675 | 265 | Imbalanced |
| Movie reviews (MOV) | 1,524 | 969 | 384 | 171 | Imbalanced |
| Product reviews (PROD) | 4,272 | 3,101 | 863 | 308 | Imbalanced |

Table 3. Twitter Datasets

| Dataset Name | Size | Sentiments | | | | Balance |
|---|---|---|---|---|---|---|
| | | Positive | Negative | Neutral | Objective | |
| ArTwitter | 1,951 | 993 | 958 | - | - | Balanced |
| AAQ | 4,296 | 2,148 | 2,148 | - | - | Balanced |
| AJGT | 1,800 | 900 | 900 | - | - | Balanced |
| ASTD | 10,006 | 799 | 1,684 | 832 | 6,691 | Imbalanced |
| STD | 2,000 | 448 | 1,350 | 202 | - | Imbalanced |

Table 4. Dataset Preprocessing Actions

| Text | Action |
|---|---|
| Stop words, diacritics | Removed |
| Elongated words | Normalized to base form |
| Punctuation, numbers | Replaced by : PUNK, NUM |
| Symbols, Non-Arabic | Replaced by : UNK |
| Letter (Alef) إ, أ, آ | Normalized to ا |
| Letter (Teh Marbuta) ة | Normalized to ه |
| Letter (ALEF Maksura) ى | Normalized to ي |

tweets written in Jordanian dialect and MSA. The Twitter dataset for Arabic sentiment analysis (ArTwitter) consisting of 2,000 labeled tweets was collected by Abdulla et al. (2013). ArTwitter version evaluated by Altowayan and Tao (2016) is used in our experiments. Table 3 lists information of all mentioned Twitter datasets.

Table 4 shows preprocessing steps to reduce noise and sparsity in the data. A Stop words[8] list containing 750 words is used to filter, normalize, and accommodate the known tokens in the PWE. Diacritics are infrequently used in Arabic text on social media platforms. They are considered noise and are removed. Elongated words having the Tatweel symbol are simply normalized to base form by removing the Tatweel symbol. Punctuations and numbers are replaced with PUNK and NUM, respectively. Symbols and Non-Arabic words are replaced with UNK. Letter normalization is performed on several Arabic letters that are often misspelled using variants (Habash 2010).

Table 5 shows vocabulary size used during the training on each dataset after performing vocabulary reduction during the data preparation phase to reduce the number of OOV words and to avoid poor or random initialization of their corresponding WEVs.

[8]https://github.com/mohataher/arabic-stop-words.

Table 5. Datasets Vocabulary Size

| Dataset | LABR-R | HTL | RES | MOV | PROD | ASTD | ArTwitter | AAQ | STD | AJGT |
|---|---|---|---|---|---|---|---|---|---|---|
| Vocabulary size | 30,000 | 30,000 | 30,000 | 15,000 | 5,000 | 10,000 | 5,000 | 15,000 | 5,000 | 5,000 |

Table 6. Arabic PWEs Statistics

| Name | Dimension | Corpus | # Tokens | Vocabulary size | Method | Language |
|---|---|---|---|---|---|---|
| C-PWE (CBOW58) | 300 | Web-crawled | 3.3B | 2.2M | Word2vec (CBOW) | Arabic |
| T-PWE (Twt-CBOW) | 300 | Twitter | 1090M | 164K | Word2vec (CBOW) | Arabic |
| W-PWE (WWW-CBOW) | 300 | Common Crawl | 2225.3M | 146K | Word2vec (CBOW) | Arabic |
| D-PWE | 300 | Training dataset | Variable | Variable | FastText | Arabic |

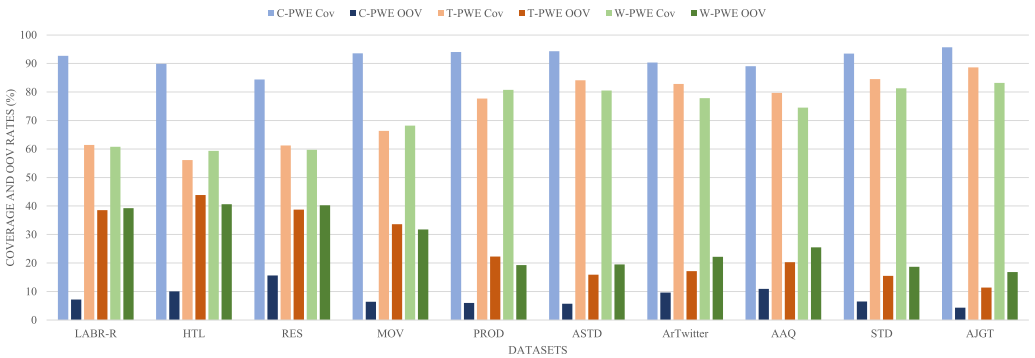



Fig. 4. Vocabulary coverage rates of various PWEs on all datasets.

### 4.2 PWEs Setup

To assign the proper PWE for each text domain and each embedding channel from the three Arabic PWEs, we conducted an experiment to evaluate the Arabic PWEs listed in Table 6 based on datasets coverage rates, OOV words coverage rates, and PWE text domain. CBOW58 is a PWE trained on a massive amount of web-crawled data built from our previous work (Dahou et al. 2016), where we followed Arts et al. (2014) to crawl such massive data. Twt-CBOW and WWW-CBOW from AraVec are two external PWEs, where Twt-CBOW and WWW-CBOW were built on top of two text domains: Tweets and World Wide Web pages using CBOW architecture from Word2vec framework. In this article, we refer to CBOW58, Twt-CBOW, and WWW-CBOW as C-PWE, T-PWE, and W-PWE, respectively. The reason for choosing CBOW to train PWEs among other architectures such as SG and Glove is based on our previous work (Dahou et al. 2016), and the work done by Zahran et al. (2015), Elrazzaz et al. (2017), and Soliman et al. (2017), where these architectures have been investigated and the quality of the generated Arabic word embeddings has been evaluated. The results show that CBOW shows the best performance results among various intrinsic and extrinsic evaluation tasks.

Experiment results are shown in Figure 4 where the coverage rates are calculated using Equations (1) and (2).

$$Coverage(Cov) = \frac{W_{we}}{Vocab} \times 100 \tag{1}$$

$$OOV = \frac{OOV_{we}}{Vocab} \times 100 \tag{2}$$

Table 7. FastText Training Parameters

| Parameter | Model | Word ngrams | Min character $n$-grams | Max character $n$-grams | Alpha | Window | Negative |
|---|---|---|---|---|---|---|---|
| Value | Skip-gram | 1 | 3 | 6 | 0.025 | 5 | 5 |

Where $W_{we}$ is the number of words from the dataset existing in PWE vocabulary, and $Vocab$ is the number of unique words in a dataset. OOV coverage rate is calculated using Equation (2) where $OOV_{we}$ is the number of OOV words from PWE vocabulary. Results show that C-PWE covers more words from all used datasets, with an average coverage rate of 92.03% of all unique words across all datasets, and 7.94% OOV average coverage rate. T-PWE and W-PWE have less coverage rates of words across all datasets. An average coverage rate of 72.35% and an OOV average coverage rate of 27.63% for T-PWE. W-PWE results show an average coverage rate of 71.57% and an OOV average coverage rate of 27.63%. In some datasets, the average coverage rate of OOV words can reach more than 30% across review datasets when using T-PWE and W-PWE. In most of the initialization scenarios before the training of a sentiment classification model, these OOV words will be mapped to random vectors. Random WEVs are low-quality vectors that do not hold any Semantic-Syntactic features that are usually captured from training data. Thus, randomly generated WEVs will affect the classification performance.

In the following experiments, C-PWE is set as a primary PWE in WEVs building module assigned to embedding channel 1 to cover general sentiment features. T-PWE and W-PWE are set as domain-specific PWEs assigned to embedding channel 2 to cover Twitter and reviews data domains, respectively. For single embedding channel CNNs (SCNN), which are described in Section 4.3, W-PWE and T-PWE are set as primary PWE for reviews and Twitter datasets, respectively.

FastText is used to build the corresponding D-PWE for each training dataset. It is derived from the skip-gram model, which ignores the internal structure of words. Therefore, character $n$-grams information is integrated into FastText to take into account the language morphology (Bojanowski et al. 2016). FastText represents a word $w$ as a bag of character $n$-gram including the word itself. We consider a dictionary of $n$-grams of size $G$, and $G_w$ is a set of $n$-grams appearing in $w$. The symbol $c$ is the context word surrounding the word $w$. A vector representation $\mathbf{z}_g$ will be assigned to each $n$-gram $g$ where $w$ will be represented as the sum of the vector representations of its $n$-grams. Equation (3) defines $s$ as a scoring function that maps pairs of (word, context) to scores in $\mathbb{R}$.

$$s(w, c) = \sum_{g \in G_w} \mathbf{z}_g^T \mathbf{v}_c \tag{3}$$

Table 7 shows FastText training parameters used to build D-PWE for each dataset where the minimum character $n$-grams to learn word embeddings is set to three and the maximum character $n$-grams is set to six. The number of tokens and the vocabulary size used to build D-PWE is variable and relies on dataset size. For each dataset, we build the corresponding D-PWE to analyze the effect of the generated word embeddings by mapping them to OOV words vectors rather than randomly initializing them. In addition, building D-PWE using FastText allows learning meaningful representation for rare words.

### 4.3 CNNs Setup

Across all experiments, seven varieties of CNN with different setups are implemented. CNN is used to assess the impact of several factors such as word embeddings initialization, data balance, and the number of embedding channels. Table 8 shows different setup options for each implemented CNN.

Table 8. CNNs Setup

| Network name | Embedding layer initialization | D-PWE | Primary PWE | Domain-specific PWE | Embedding channels | CNN block channels |
|---|---|---|---|---|---|---|
| SCNN-1 | Random | Not used | — | — | 1 | 3 |
| SCNN-2 | Domain-specific WEVs | Not used | — | T-PWE or W-PWE | 1 | 3 |
| SCNN-3 | Domain-specific WEVs | Used | — | T-PWE or W-PWE | 1 | 3 |
| SCNN-4 | Primary WEVs | Not used | C-PWE | — | 1 | 3 |
| SCNN-5 | Primary WEVs | Used | C-PWE | — | 1 | 3 |
| SCNN-6 | MCE-CNN trained features | Used | — | T-PWE or W-PWE | 1 | 3 |
| MCE-CNN | Primary and domain-specific PWE | Used | C-PWE | T-PWE or W-PWE | 2 | 3 |

Table 9. CNNs Training Hyper-parameters

| Dataset | WEVs dimension | Filter sizes | Feature maps | Pool length | Dropout | FC layer units | Batch size |
|---|---|---|---|---|---|---|---|
| Twitter | 300 | 3, 5, 7 | 100 | 2 | 0.7 | 150 | 32 |
| Reviews | 300 | 3, 5, 7 | 100 | 2 | 0.7 | 150 | 128 |

The main differences between the proposed varieties of CNN are the WEVs fed to the embedding channel and the number of embedding channels.

SCNN-1 is a CNN with a single embedding channel, where the WEVs fed to the embedding layer are randomly initialized and fine-tuned during the training. SCNN-2, SCNN-3, SCNN-4, and SCNN-5 are CNN models with only a single embedding channel initialized using domain-specific (T-PWE or W-PWE) or primary WEVs (C-PWE) as Table 8 shows. SCNN-6 is a variation of MCE-CNN and SCNN-3, where the two channels of word embeddings from MCE-CNN are concatenated as pre-trained word vectors as inputs to train SCNN-3. OOV vectors are initialized in two ways, which are random such as in SCNN-2 and SCNN-4, or mapped to WEVs from D-PWE such as in SCNN-3 and SCNN-5. MCE-CNN is a two channel embedding CNN model, where primary WEVs are used to initialize the embedding layer on embedding channel 1, while a domain-specific WEV is used to initialize the embedding layer on embedding channel 2. D-PWE vectors are encoded into both channel WEVs to cover OOV WEVs.

All seven varieties of CNN are trained for 50 epochs, and Adam (Kingma and Ba 2014) is employed to optimize the training process with learning rate 0.001. A uniform distribution is used to initialize CNNs parameters. Several parameter exploration experiments are executed to find the most suitable hyper-parameters for the two text domains datasets. Table 9 lists the default training hyper-parameters for CNNs.

### 4.4 Performance Measures

A set of measures are used to assess the performance of each CNN. These measures are precision (Prc), recall (Rec), accuracy (Acc), and F-score (F1). These measures are defined as follows:

$$Precision = \frac{TP}{TP + FP} \times 100$$

$$Recall = \frac{TP}{TP + FN} \times 100$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100$$

$$F\text{-}score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Where TP, TN, FP, and FN denote true positive, true negative, false positive, and false negative, respectively. The 10-fold cross-validation (CV) method is used to determine the training and testing sets, where the dataset is split into 10 different groups with 10 times running. At each run, nine groups are used for the training and the rest of the group is used for testing. The final result is computed as the average of all 10 runs results. Experiments in this study were conducted on a server with GeForce GTX 1080Ti GPU and 126Gb of RAM.

### 4.5 Results and Discussion

In this section, we present the experimental results and discussion to assess the impact of (1) random WEVs initialization for both Twitter and the reviews datasets, (2) WEVs and OOV WEVs initialization using PWEs, (3) CNN varieties that rely on single or multiple embedding channels, and (4) the performance of MCE-CNN on a 3-class classification task. SCNN-1 is considered as our baseline that uses randomly initialized WEVs.

*Impact of Random Initialization of WEVs.* The effect of random initialization of WEVs was analyzed to measure the performance of CNNs trained on top of random embeddings features. SCNN-1 uses a single embedding channel to fine-tune random WEVs fed into the embedding layer and to extract sentiment features using the CNN block. Tables 10 and 11 show classification results for balanced and imbalanced datasets. Results indicate that SCNN-1 shows poor performance compared to other SCNNs, where a domain-specific PWE is used to initialize datasets WEVs. The performance gap in terms of average accuracy between SCNN-1 and SCNN-2 is 4.79% balanced datasets.

We can also find that SCNN-1 did not perform well on Twitter datasets when using random initialization as shown in Table 11. The performance gap in terms of average accuracy between SCNN-1 and SCNN-2 is 3.43% across imbalanced datasets. These results show that the difference between data from each domain and WEVs initialization have a significant impact on CNNs performance. Factors such as sequence length and the number of training samples will also affect the classification accuracy.

Results indicate that initializing datasets WEVs randomly can achieve similar performances to the case of using PWE when training on large datasets having long word sequences. However, SCNN-1 could not capture most of the information presented in Arabic text when fine-tuning the randomly initialized WEVs of small datasets with short word sequences such as Twitter.

*Impact of WEVs and OOV WEVs Initialization Using PWEs.* The most important phase of optimizing a deep learning model can be the extraction of most relevant features. Hence, initializing the training starting with high-quality WEVs can be decisive to gain an accurate model. Good initialization helps to perform training successfully and leads to an accurate model when dealing with datasets having small amounts of training samples or short text sequences. In Section 4.5, SCNN-2 with domain-specific WEVs outperforms SCNN-1 significantly. Tables 10 and 11 show that mapping OOV words to D-PWE word embeddings vectors leads to slightly low performance in term of accuracy when evaluating on review datasets. On the other hand, SCNN-3 shows a small accuracy improvement on the LABR-R dataset. The accuracy difference between SCNN-2 and SCNN-3 is less than 0.9%.

The combination of different WEVs in the Twitter dataset can lead to performance improvement as shown in Tables 10 and 11. Using FastText to build D-PWE increases the kind of sentiment

Table 10. Results of Balanced Review and Twitter Datasets (Time in hours)

| Model | Measures | LABR-R | HTL | RES | MOV | PROD | ASTD | ArTwitter | AAQ | STD | AJGT | **Average** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Acc | 82.69 | 89.76 | 80.66 | 71.38 | 75.28 | 71.46 | 84.60 | 75.14 | 79.4 | 83.5 | 79.39 |
| | Prc | 82.46 | 90.42 | 80.08 | 69.2 | 73.95 | 71.66 | 81.77 | 72.38 | 77.84 | 79.85 | 77.96 |
| SCNN-1 | Rec | 83.33 | 88.96 | 81.67 | 77.92 | 78.17 | 71.21 | 90 | 81.59 | 82.31 | 89.66 | 82.48 |
| | F1 | 82.85 | 89.67 | 80.86 | 72.99 | 75.9 | 71.33 | 85.67 | 76.68 | 79.96 | 84.43 | 80.03 |
| | Time | 5.65 | 1.61 | 1.62 | 0.38 | 0.48 | 0.13 | 0.29 | 0.71 | 0.14 | 0.33 | 1.13 |
| | Acc | 84.96 | 93.03 | 82.22 | 72.55 | 78.99 | 80.47 | 90.95 | 84.69 | 83.77 | 90.16 | 84.18 |
| | Prc | 84.67 | 93.16 | 82.53 | 71.79 | 79.25 | 81.08 | 89.76 | 83.62 | 83.66 | 89.62 | 83.91 |
| SCNN-2 | Rec | 85.56 | 92.89 | 81.82 | 75.29 | 78.64 | 79.72 | 93.03 | 86.48 | 84.34 | 91 | 84.88 |
| | F1 | 85.1 | 93 | 82.14 | 72.92 | 78.78 | 80.27 | 91.32 | 85 | 83.85 | 90.24 | 84.26 |
| | Time | 6.48 | 1.60 | 2.59 | 0.47 | 0.79 | 0.23 | 0.28 | 0.73 | 0.21 | 0.34 | 1.37 |
| | Acc | 85.15 | 92.7 | 82.61 | 72.02 | 79.86 | 81.6 | 91.01 | 85.88 | 83.65 | 90.77 | 84.53 |
| | Prc | 84.83 | 92.79 | 82.99 | 72.76 | 79.56 | 82.12 | 90.41 | 84.68 | 84.90 | 91.51 | 84.66 |
| SCNN-3 | Rec | 85.56 | 92.63 | 82.05 | 73.21 | 80.50 | 81.22 | 92.33 | 87.74 | 82.31 | 90 | 84.76 |
| | F1 | 85.3 | 92.7 | 82.49 | 72.26 | 79.93 | 81.59 | 91.32 | 86.17 | 82.42 | 90.69 | 84.49 |
| | Time | 4.91 | 2.69 | 2.69 | 0.47 | 0.64 | 0.16 | 0.48 | 0.45 | 0.15 | 0.25 | 1.29 |
| | Acc | 87.44 | 93.35 | 84.98 | 70.32 | 79.81 | 81.17 | 91.84 | 85.03 | 84 | 89.33 | 84.73 |
| | Prc | 85.24 | 89.37 | 82.09 | 68.46 | 75.81 | 78.24 | 87.87 | 79.65 | 79.84 | 84.09 | 81.07 |
| SCNN-4 | Rec | 90.72 | 98.48 | 89.58 | 81.09 | 87.81 | 86.73 | 97.67 | 94.45 | 91.48 | 97.11 | 91.51 |
| | F1 | 87.87 | 93.69 | 85.64 | 72.74 | 81.29 | 82.04 | 92.48 | 86.37 | 85.12 | 90.1 | 85.73 |
| | Time | 6.66 | 1.98 | 1.11 | 0.62 | 0.33 | 0.18 | 0.31 | 0.51 | 0.12 | 0.20 | 1.20 |
| | Acc | 87.48 | 93.94 | 85.33 | 70.32 | 79.92 | 82.36 | 91.73 | 84.51 | 83.1 | 89.5 | 84.82 |
| | Prc | 85.19 | 90.46 | 83.09 | 68.45 | 75.73 | 79.84 | 88.29 | 78.89 | 78.33 | 84.52 | 81.28 |
| SCNN-5 | Rec | 90.85 | 98.3 | 88.79 | 81.6 | 88.15 | 86.86 | 96.87 | 94.45 | 91.7 | 96.88 | 91.45 |
| | F1 | 87.92 | 94.2 | 85.82 | 73.17 | 81.43 | 83.08 | 92.32 | 85.96 | 84.41 | 90.24 | 85.86 |
| | Time | 6.30 | 0.99 | 1.74 | 0.50 | 0.35 | 0.18 | 0.31 | 0.52 | 0.12 | 0.20 | 1.12 |
| | Acc | 84.75 | 93.48 | 82.95 | 75.83 | 78.59 | 81.1 | 92.1 | 85.98 | 84.57 | 91.89 | 85.12 |
| | Prc | 83.73 | 94.62 | 81.73 | 81.33 | 79.08 | 81.71 | 90.83 | 84.55 | 84.64 | 90.35 | 85.26 |
| SCNN-6 | Rec | 87.36 | 92.33 | 85.76 | 68.70 | 78.75 | 80.47 | 94.15 | 88.40 | 84.78 | 93.89 | 85.46 |
| | F1 | 85.24 | 93.34 | 83.43 | 73.43 | 78.45 | 80.95 | 92.41 | 86.37 | 84.59 | 92.05 | 85.03 |
| | Time | 4.73 | 1.21 | 2.06 | 0.27 | 0.39 | 0.19 | 0.26 | 0.90 | 0.24 | 0.24 | 1.05 |
| | Acc | 86.3 | 93.86 | 84.08 | 70.99 | 79.87 | 82.79 | 92.66 | 87.33 | 84.65 | 92.55 | **85.51** |
| | Prc | 86.58 | 93.75 | 84.02 | 76.08 | 79.46 | 82.88 | 91.76 | 86.74 | 85.37 | 91.99 | 85.86 |
| MCE-CNN | Rec | 86.06 | 94.03 | 84.3 | 64.49 | 80.72 | 82.97 | 94.24 | 88.29 | 84.08 | 93.22 | 85.24 |
| | F1 | 86.31 | 93.86 | 84.11 | 68.04 | 79.91 | 82.78 | 92.93 | 87.45 | 84.59 | 92.59 | 85.26 |
| | Time | 8.62 | 2.27 | 2.55 | 0.32 | 0.14 | 0.27 | 0.33 | 0.93 | 0.21 | 0.50 | 1.61 |

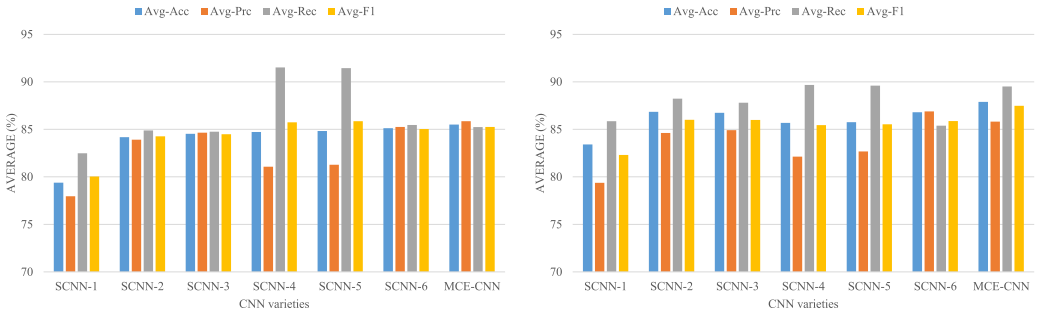
features extracted from the dataset itself to cover OOV words by paying more attention to characters *n*-grams level. SCNN-3 performs better than SCNN-2 in terms of accuracy by 1.13% and 1.19% on ASTD and AQQ balanced datasets, respectively. SCNN-3 has lower accuracy on the STD balanced dataset compared to SCNN-2. SCNN-3 performs better on Twitter datasets when using imbalance shape.

*Impact of CNN Varieties.* CNNs are trained to extract and learn more sentiment features from the initial WEVs. Random initializing of WEVs for review datasets can boost these WEVs to perform

Table 11. Results of Imbalanced Review and Twitter Datasets (Time in Hours)

| Model | Measures | HTL | RES | MOV | PROD | ASTD | STD | **Average** |
|---|---|---|---|---|---|---|---|---|
| SCNN-1 | Acc | 93.74 | 85.28 | 78.26 | 83.7 | 76.71 | 82.79 | 83.41 |
| | Prc | 94.93 | 89.15 | 77.35 | 86.45 | 62.61 | 65.7 | 79.37 |
| | Rec | 97.4 | 91.52 | 98.42 | 93.89 | 69.08 | 64.86 | 85.86 |
| | F1 | 96.15 | 90.32 | 86.62 | 90.01 | 65.57 | 65.17 | 82.31 |
| | Time | 2.27 | 2.14 | 0.61 | 1.18 | 0.13 | 0.31 | 1.11 |
| SCNN-2 | Acc | 94.93 | 87.76 | 79.76 | 86.84 | 84.53 | 87.24 | 86.84 |
| | Prc | 95.64 | 89.15 | 79.22 | 88.39 | 75.06 | 80.26 | 84,62 |
| | Rec | 98.16 | 95.3 | 97.26 | 95.77 | 78.1 | 64.82 | 88.24 |
| | F1 | 96.88 | 92.11 | 87.3 | 91.92 | 76.46 | 71.39 | 86.01 |
| | Time | 2.18 | 4.82 | 0.77 | 1.38 | 0.26 | 0.31 | 1.62 |
| SCNN-3 | Acc | 94.71 | 87.56 | 78.93 | 86.81 | 84.85 | 87.58 | 86.74 |
| | Prc | 95.49 | 88.98 | 78.45 | 88.23 | 78.66 | 79.63 | 84.91 |
| | Rec | 98.05 | 95.21 | 97.27 | 95.96 | 72.83 | 67.52 | 87.81 |
| | F1 | 96.75 | 91.99 | 86.84 | 91.92 | 75.51 | 72.92 | 85.99 |
| | Time | 3.75 | 3.34 | 0.51 | 0.62 | 0.22 | 0.25 | 1.45 |
| SCNN-4 | Acc | 94.21 | 86.42 | 77.36 | 87.4 | 81.92 | 86.75 | 85.68 |
| | Prc | 93.55 | 85.69 | 84.81 | 87.22 | 66.59 | 74.92 | 82.13 |
| | Rec | 99.66 | 98.31 | 83.2 | 98.32 | 88.11 | 70.44 | 89.67 |
| | F1 | 96.5 | 91.57 | 83.88 | 92.43 | 75.8 | 72.51 | 85.45 |
| | Time | 5.83 | 2.78 | 0.56 | 0.38 | 0.28 | 0.16 | 1.66 |
| SCNN-5 | Acc | 94.66 | 86.45 | 76.62 | 86.84 | 82.8 | 87.19 | 85.76 |
| | Prc | 94.1 | 85.87 | 84.97 | 86.78 | 67.6 | 76.7 | 82.67 |
| | Rec | 99.61 | 98.08 | 82.37 | 98.15 | 89.6 | 69.75 | 89.59 |
| | F1 | 96.77 | 91.57 | 82.77 | 92.11 | 77.02 | 72.91 | 85.53 |
| | Time | 5.80 | 2.54 | 1.13 | 0.38 | 0.28 | 0.16 | 1.71 |
| SCNN-6 | Acc | 95.84 | 88.27 | 78.49 | 86.21 | 84.53 | 87.42 | 86.79 |
| | Prc | 97.77 | 92.53 | 89.18 | 90.89 | 74.02 | 76.91 | 86.88 |
| | Rec | 97.04 | 91.80 | 79.85 | 91.61 | 80.60 | 71.38 | 85.38 |
| | F1 | 97.40 | 92.14 | 83.67 | 91.21 | 77.06 | 73.75 | 85.87 |
| | Time | 3.25 | 2.46 | 0.41 | 0.68 | 0.22 | 0.37 | 1.23 |
| MCE-CNN | Acc | 95.69 | 88.62 | 80.36 | 87.9 | 86.02 | 88.75 | **87.89** |
| | Prc | 96.43 | 90.36 | 81.78 | 89.69 | 77.13 | 79.51 | 85.82 |
| | Rec | 98.26 | 94.96 | 93.71 | 95.54 | 80.6 | 74.02 | 89.52 |
| | F1 | 97.33 | 92.6 | 87.22 | 92.51 | 78.76 | 76.43 | 87.48 |
| | Time | 5.77 | 2.06 | 0.56 | 0.40 | 0.29 | 0.27 | 1.56 |

almost similar to a CNN trained using WEVs mapped from a PWE. On the other hand, SCNNs could not boost randomly initialized WEVs trained on Twitter datasets to perform better due to the short length of input tweet. This phenomenon indicates the ability of our CNN varieties to learn better sentiment features from WEVs when they are trained on a large amount of samples with long word sequences. MCE-CNN outperforms SCNNs that rely only on a single channel since it benefits from the second channel domain-specific sentiment features. Furthermore, MCE-CNN considerably boosts the accuracy and gets the best accuracy among SCNNs, even SCNN-6. Furthermore, SCNN-6 shows the best performance in terms of accuracy on the MOV dataset compared




(a) Balanced (b) Imbalanced

Fig. 5. Average results on balanced and imbalanced datasets.

to all evaluated models due to its ability of fine-tuning the features inputted from MCE-CNN. Our comparisons with the experimented CNN varieties that use distinct PWEs show that an additional embedding channel with fine-tuning during the training can significantly boost the performance. If a fair amount of data such as reviews is inputted to the network, it can have a significant capacity for generalization. Moreover, the network will be able to learn proper sentiment features and classify tweets that were not shown during training. The reason for the success of MCE-CNN is its superior ability to capture relevant sentiment features encoded on the word and character *n*-grams level, as well as domain-specific sentiment features. Concerning WEVs and out-of-vocabulary WEVs, the CNN block in MCE-CNN with multiple filter sizes can benefit more from the encoded WEVs of each dataset on both channels. In this case, CNN block acts as a separate feature extraction module in both embedding channels focusing on general and domain-specific sentiment features. Table 10 shows classification results of SCNNs and MCE-CNN on the balanced shape of each dataset. Compared to the evaluated SCNNs, MCE-CNN outperforms all of them in terms of performance accuracy. The same observations can be noticed on the imbalanced shape of each dataset as shown in Table 11. For highly imbalanced datasets, we split them into balanced and imbalanced to evaluate our CNN varieties and to reflect real-world using cases, where most of the datasets are not balanced. Moreover, CNN varieties show aptitude to handle the class imbalance problem, while it is significant in Twitter datasets. In some cases, SCNN-2 performs better than SCNN-3 due to the usage of D-PWE. In SCNN-3, D-PWE is trained on the training dataset itself using FastText. In most cases, these datasets are small, which generates word embeddings with poor quality.

Figure 5(a) and (b) shows a comparison between different CNN varieties across balanced and imbalanced datasets. From the charts, a performance improvement on different CNN varieties can be noticed . Furthermore, MCE-CNN benefits from a second embedding channel since it can refine WEVs from primary PWE and domain-specific PWE at the same time. A second embedding channel in MCE-CNN leads to an increase in the performance accuracy that outperforms evaluated SCNNs.

MCE-CNN outperforms the evaluated SCNNs in our experiments on all datasets in terms of average accuracy, especially on imbalanced datasets. The results of imbalanced datasets are mostly better than balanced datasets since more training samples are present in the imbalanced datasets; whereas, we limited the number of samples in balanced datasets to make the classes equal. In addition, CNN varieties perform well even with fewer per-class examples such as in Twitter datasets and scaling better than other existing methods for Arabic sentiment classification. We can also find that reducing the vocabulary size on both text domains datasets by selecting only the most

Table 12. Results of Imbalanced Review and Twitter Datasets Using MCE-CNN for 3-Class Classification

| Dataset | HTL | | | | RES | | | | MOV | | | | PROD | | | | ASTD | | | | STD | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Measures | Acc | Prc | Rec | F1 | Acc | Prc | Rec | F1 | Acc | Prc | Rec | F1 | Acc | Prc | Rec | F1 | Acc | Prc | Rec | F1 | Acc | Prc | Rec | F1 |
| MCE-CNN (5-CV) | 81.31 | 82.97 | 81.31 | 81.89 | 86.54 | 85.16 | 86.53 | 85.29 | 64.14 | 48.87 | 64.13 | 51.25 | 80.45 | 76.9 | 80.45 | 77.61 | 64.83 | 64.72 | 64.82 | 64.61 | 81.73 | 80.79 | 81.73 | 80.86 |
| MCE-CNN (10-CV) | 81.64 | 82.55 | 81.64 | 81.87 | 86.10 | 84.53 | 86.10 | 84.75 | 66.60 | 52.87 | 66.59 | 56.37 | 80.27 | 76.33 | 80.26 | 77.18 | 65.40 | 65.53 | 65.39 | 65.16 | 81.99 | 81.44 | 81.98 | 81.37 |
| Lexicon-based | 67.1 | - | - | - | 58.9 | - | - | - | 52.6 | - | - | - | 53.7 | - | - | - | - | - | - | - | - | - | - | - |

Table 13. Computation Time (Hours) of Review and Twitter Datasets Using MCE-CNN for 3-Class Classification

| Datasets | HTL | RES | MOV | PROD | ASTD | STD |
|---|---|---|---|---|---|---|
| MCE-CNN (5-CV) | 2.96 | 2.00 | 0.42 | 0.34 | 0.12 | 0.08 |
| MCE-CNN (10-CV) | 4.75 | 3.22 | 0.68 | 0.52 | 0.25 | 0.13 |
| **Average** | 3.86 | 2.61 | 0.55 | 0.43 | 0.19 | 0.11 |

frequent words can increase CNN varieties performance, accuracy, and reduce the number of OOV words.

The time complexity in terms of running time is reported in Table 10. Usually, MCE-CNN required more time since it has additional computations due to the use of two embedding channels. However, the average running time, which indicates the additional computations, is not time consuming. The results show that MCE-CNN achieves higher accuracy in a reasonable time.

*Impact of Increasing Dataset Classes.* In previous experiments, binary classification has been performed on all datasets by removing all the neutral, mixed, and objective samples from the datasets. Table 12 lists the results of the MCE-CNN performance on datasets having at least three classes to perform 3-class classification. Whereas, Table 13 lists the 3-class classification experiments' computation time. The performance results of MCE-CNN are reported using 5- and 10-fold cross-validation (5-CV and 10-CV) for a fair comparison with lexicon-based approach results proposed by ElSahar and El-Beltagy (2015).

*Comparison with Other Methods.* The results of MCE-CNN are compared against the state-of-the-art methods. It is worth mentioning that some datasets used in the comparison table are not compatible with our experimental setup. To evaluate a combined LSTM on an ASTD dataset, authors Altowayan and Tao (2016) used in their experiments an ASTD version from our previous work (Dahou et al. 2016) in its balanced and preprocessed shape. However, in this article, different preprocessing actions on all datasets are performed. Rabab'Ah et al. (2016) evaluated a tool named SentiStrength to tackle the Arabic SA task on several datasets including HTL, RES, MOV, and PROD. In their work, they only used the imbalanced shape of datasets and did not mention the used train and test split technique. Al-Azani and El-Alfy (2017b) used 80% of the STD dataset for training and 20% for testing.

Concerning review datasets, MCE-CNN has a very good performance in terms of accuracy on all evaluated datasets compared to existing methods, where Table 14 introduces accuracy improvement of 4.36% on the LABR-R dataset.

Table 15 illustrates the performance improvement achieved by MCE-CNN on the aforementioned Twitter datasets. MCE-CNN has accuracy improvements of 5.39%, 7.12%, 3.47%, 3.83%, and 1.16% on the ArTwitter, AAQ, STD, AGJT, and ASTD datasets, respectively. The reason for the performance accuracy improvement is that MCE-CNN can pay special attention to domain-specific sentiment features and learn much better quality features by fine-tuning WEVs from both embedding channels at the same time.

## 5 CONCLUSIONS

In this article, an MCE-CNN was proposed for Arabic sentiment classification and thoroughly evaluated. Moreover, the impact of different WEVs initialization techniques, deep learning approaches used in NLMs, and sentiment classification was analyzed, as well as dataset text domain, balance, and effectiveness of vocabulary reduction. In addition, MCE-CNN and SCNNs were implemented

Table 14. Comparison Against State-of-the-Art Methods on Review Datasets

| Datasets | LABR-R Balanced | | | | HTL Imbalanced | | | | RES Imbalanced | | | | MOV Imbalanced | | | | PROD Imbalanced | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Measure | Acc | Prc | Rec | F1 | Acc | Prc | Rec | F1 | Acc | Prc | Rec | F1 | Acc | Prc | Rec | F1 | Acc | Prc | Rec | F1 |
| Logistic regression (Altowayan and Tao 2016) | 81.88 | 80.59 | 82.60 | 81.58 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| SentiStrength (Rabab'Ah et al. 2016) | - | - | - | - | 77.1 | 91.1 | 79.2 | 84.7 | 67.6 | 81 | 74.2 | 77.5 | 44.6 | 75.4 | 33.5 | 46.4 | 62.5 | 81.9 | 66.8 | 73.6 |
| MCE-CNN | 86.3 | 86.58 | 86.06 | 86.31 | 95.69 | 96.43 | 98.26 | 97.33 | 88.62 | 90.36 | 94.96 | 92.6 | 80.36 | 81.78 | 93.71 | 87.22 | 87.9 | 89.69 | 95.54 | 92.51 |

Table 15. Comparison Against State-of-the-Art Methods on Twitter Datasets

| Datasets | ArTwitter Balanced | | | | AAQ Balanced | | | | STD Imbalanced | | | | AGJT Balanced | | | | ASTD Balanced | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Measure | Acc | Prc | Rec | F1 | Acc | Prc | Rec | F1 | Acc | Prc | Rec | F1 | Acc | Prc | Rec | F1 | Acc | Prc | Rec | F1 |
| Combined-LSTM (Al-Azani and El-Alfy 2017a) | 87.27 | 87.36 | 87.27 | 87.28 | - | - | - | - | - | - | - | - | - | - | - | - | 81.63 | 82.32 | 81.63 | 81.64 |
| NuSVC (Altowayan and Tao 2016) | - | - | - | - | 80.21 | 83.00 | 76.50 | 79.62 | - | - | - | - | - | - | - | - | - | - | - | - |
| Stacking (eclf14) (Al-Azani and El-Alfy 2017b) | - | - | - | - | - | - | - | - | 85.28 | 61.04 | 67.14 | 63.95 | - | - | - | - | - | - | - | - |
| SVM (bigrams) (Alomari et al. 2017) | - | - | - | - | - | - | - | - | - | - | - | - | 88.72 | 92.08 | 84.89 | 88.27 | - | - | - | - |
| MCE-CNN | 92.66 | 91.76 | 94.21 | 92.93 | 87.33 | 86.74 | 88.29 | 87.45 | 88.75 | 79.51 | 74.02 | 76.43 | 92.55 | 91.99 | 93.22 | 92.59 | 82.79 | 82.88 | 82.97 | 82.78 |

with a single embedding channel to assess the performance of different CNN varieties, the impact of WEVs initialization techniques, and the number of embedding channels. MCE-CNN initializes the embedding layer on each embedding channel with combinations of PWEs vectors trained on common web domains data, domain-specific data, as well as on the training dataset itself. These PWEs vectors are trained on different word and character *n*-grams levels. MCE-CNN improved the Arabic sentiment classification performance by several essential operations such as (1) mapping high-quality WEVs as an initial phase to start the training, and fine-tuning WEVs passed from completely unsupervised neural language models. (2) Encoding different levels of word and character *n*-grams sentiment features into WEVs before being fed to the embedding layer. (3) Avoiding poor or random initialization of WEVs and especially out-of-vocabulary WEVs using D-PWE and vocabulary reduction. (4) Learning various sentiment features from domain-specific WEVs on the second embedding channel and from the convolutional filter channels in the CNN block.

Finally, the performance of MCE-CNN was evaluated in a set of experimental series on balanced and highly imbalanced datasets. Based on the experimental results, it can be concluded that MCE-CNN provided better results than evaluated SCNNs and state-of-the-art methods, especially in the case of imbalanced datasets. Moreover, according to the promising results and findings, this study could provide useful insights to help the advancement of sentiment classification research in the Arabic language.

## APPENDIX

## A ENGLISH DATASET CLASSIFICATION USING MCE-CNN

Table 16 presents results of MCE-CNN on the English movie reviews dataset, where the movie reviews dataset[9] (Pang and Lee 2005) contains 10,662 samples divided into positive and negative samples. The maximum sentence length is 56 tokens with 18,688 tokens as the maximum number of tokens presented in this dataset. We evaluated the model using 5- and 10-fold cross-validation. English word embeddings trained on 42 billion tokens from Common Crawl using Glove[10] is set as C-PWE. In addition, a pre-trained English word embeddings trained using Word2vec[11] on Google News dataset (about 100 billion words) is set as domain-specific PWE for MCE-CNN.

Table 16. English Movie Reviews Dataset Classification Using MCE-CNN

| Measures | Acc | Prc | Rec | F1 |
|---|---|---|---|---|
| MCE-CNN (5-CV) | 77.59 | 77.08 | 78.61 | 77.82 |
| MCE-CNN (10-CV) | 78.06 | 77.23 | 79.68 | 78.41 |
| CharCNN (Wang et al. 2017) | 77.01 | - | - | - |
| Word CNN (Kim 2014) | 81.5 | - | - | - |
| WCCNN (Wang et al. 2017) | 83.77 | - | - | - |

The results show that MCE-CNN with its default setup achieved better results than CharCNN on the evaluated dataset. The gap between MCE-CNN and the state-of-the-art models is related to the used data preprocessing techniques for English text. Besides, the English word embedding models used during the initialization phase, and the parameters used to train the CNN model were

[9] https://www.cs.cornell.edu/people/pabo/movie-review-data/.
[10] https://nlp.stanford.edu/projects/glove/.
[11] https://code.google.com/archive/p/word2vec/.

not investigated or tuned in these experiments. A potential improvement is worth investigating to improve the model architecture for Arabic SC and other languages.

## REFERENCES


Nawaf A. Abdulla, Nizar A. Ahmed, Mohammad A. Shehab, and Mahmoud Al-Ayyoub. 2013. Arabic sentiment analysis: Lexicon-based and corpus-based. In *Proceedings of the 2013 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*. 1–6.

Sadam Al-Azani and El-Sayed M. El-Alfy. 2017a. Hybrid deep learning for sentiment polarity determination of Arabic microblogs. In *Proceedings of the International Conference on Neural Information Processing*. Springer, 491–500.

Sadam Al-Azani and El-Sayed M. El-Alfy. 2017b. Using word embedding and ensemble learning for highly imbalanced data sentiment analysis in short Arabic text. *Procedia Comput. Sci.* 109 (2017), 359–366.

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. *arXiv preprint arXiv:1307.1662* (2013).

Ahmad Al-Sallab, Ramy Baly, Hazem Hajj, Khaled Bashir Shaban, Wassim El-Hajj, and Gilbert Badaro. 2017. AROMA: A recursive deep learning model for opinion mining in Arabic as a low resource language. *ACM Trans. Asian Low-Resour. Lang. Inf. Process. (TALLIP)* 16, 4 (2017), 25.

Ahmad Al Sallab, Hazem Hajj, Gilbert Badaro, Ramy Baly, Wassim El Hajj, and Khaled Bashir Shaban. 2015. Deep learning models for sentiment analysis in Arabic. In *Proceedings of the 2nd Workshop on Arabic Natural Language Processing*. 9–17.

Mohammad Al-Smadi, Omar Qawasmeh, Mahmoud Al-Ayyoub, Yaser Jararweh, and Brij Gupta. 2018. Deep recurrent neural network vs. support vector machine for aspect-based sentiment analysis of Arabic hotels' reviews. *J. Comput. Sci.* 27 (2018), 386–393.

Abdulaziz M. Alayba, Vasile Palade, Matthew England, and Rahat Iqbal. 2017. Arabic language sentiment analysis on health services. In *Proceedings of the 1st International Workshop on Arabic Script Analysis and Recognition (ASAR)*. 114–118.

Khaled Mohammad Alomari, Hatem M. ElSherif, and Khaled Shaalan. 2017. Arabic tweets sentimental analysis using machine learning. In *Proceedings of the International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. Springer, 602–610.

A. Aziz Altowayan and Lixin Tao. 2016. Word embeddings for Arabic sentiment analysis. In *IEEE International Conference on Big Data*. 3820–3825.

Mohamed Aly and Amir Atiya. 2013. LABR: A large scale Arabic book reviews dataset. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, Vol. 2. 494–498.

Tressy Arts, Yonatan Belinkov, Nizar Habash, Adam Kilgarriff, and Vit Suchomel. 2014. arTenTen: Arabic corpus and word sketches. *J. King Saud Univ. Comput. Inf. Sci.* 26, 4 (2014), 357–371.

Ramy Baly, Hazem Hajj, Nizar Habash, Khaled Bashir Shaban, and Wassim El-Hajj. 2017. A sentiment treebank and morphologically enriched recursive deep models for effective sentiment analysis in Arabic. *ACM Trans. Asian Low-Resour. Lang. Inf. Process. (TALLIP)* 16, 4 (2017), 23.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606* (2016).

Emil St. Chifu, Tiberiu St. Letia, and Viorica R. Chifu. 2015. Unsupervised aspect level sentiment analysis using self-organizing maps. In *Proceedings of the 2015 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*. 468–475.

Abdelghani Dahou, Shengwu Xiong, Junwei Zhou, Mohamed Houcine Haddoud, and Pengfei Duan. 2016. Word embeddings and convolutional neural network for Arabic sentiment classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics*. 2418–2427.

Samhaa R. El-Beltagy, Mona El Kalamawy, and Abu Bakr Soliman. 2017. NileTMRG at SemEval-2017 task 4: Arabic sentiment analysis. *arXiv preprint arXiv:1710.08458* (2017).

Mohammed Elrazzaz, Shady Elbassuoni, Khaled Shaban, and Chadi Helwe. 2017. Methodical evaluation of Arabic word embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vol. 2. 454–458.

Hady ElSahar and Samhaa R. El-Beltagy. 2015. Building large Arabic multi-domain resources for sentiment analysis. In *Computational Linguistics and Intelligent Text Processing*. 23–34.

Nikos Engonopoulos, Angeliki Lazaridou, Georgios Paliouras, and Konstantinos Chandrinos. 2011. ELS: A word-level method for entity-level sentiment analysis. In *Proceedings of the International Conference on Web Intelligence, Mining and Semantics*. 1–9.

Nizar Y. Habash. 2010. Introduction to Arabic natural language processing. *Synth. Lect. Hum. Lang. Technol.* 3, 1 (2010), 1–187.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* (2012), 1–18.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014), 1–6.
Diederik Kingma and Jimmy Ba. 2014. ADAM: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013), 1–9.
Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*. 3111–3119.
AL-Smadi Mohammad, Omar Qawasmeh, Mahmoud Al-Ayyoub, Yaser Jararweh, and Brij Gupta. 2018. Deep recurrent neural network vs. Dupport vector machine for aspect-based sentiment analysis of Arabic hotels reviews. *J. Comput. Sci.* 27 (2018), 386–393.
Saif M. Mohammad, Mohammad Salameh, and Svetlana Kiritchenko. 2016. How translation alters sentiment. *J. Artif. Intell. Res.* 55 (2016), 95–130.
Mahmoud Nabil, Mohamed Aly, and Amir Atiya. 2015. ASTD: Arabic sentiment tweets dataset. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 2515–2519.
Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML'10)*. 807–814.
Alexis Amid Neme and Eric Laporte. 2013. Pattern-and-root inflectional morphology: The Arabic broken plural. *Lang. Sci.* 40 (2013), 221–250.
Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. ACL, 115–124.
Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543.
Abdullateef M. Rabab'Ah, Mahmoud Al-Ayyoub, Yaser Jararweh, and Mohammed N. Al-Kabi. 2016. Evaluating sentiStrength for Arabic sentiment analysis. In *International Conference on Computer Science and Information Technology*. 1–6.
Abu Bakr Soliman, Kareem Eissa, and Samhaa R. El-Beltagy. 2017. AraVec: A set of Arabic word embedding mdels for use in Arabic NLP. *Procedia Comput. Sci.* 117 (2017), 256–265.
Jin Wang, Zhongyuan Wang, Dawei Zhang, and Jun Yan. 2017. Combining knowledge with deep convolutional neural networks for short text classification. In *Proceedings of IJCAI*, Vol. 350.
Ainur Yessenalina, Yisong Yue, and Claire Cardie. 2010. Multi-level structured models for document-level sentiment classification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. ACL, 1046–1056.
Mohamed A. Zahran, Ahmed Magooda, Ashraf Y. Mahgoub, Hazem Raafat, Mohsen Rashwan, and Amir Atyia. 2015. Word representations in vector space and their applications for Arabic. In *Computational Linguistics and Intelligent Text Processing*. 430–443.

Received April 2018; revised December 2018; accepted February 2019