ArabiziVec: A Set of Arabizi Word Embeddings for Informal Arabic Sentiment Analysis

ASRITA VENKATA MANDALAM, BITS Pilani, Pilani Campus, India
OUMAIMA OUESLATI, SenticNet, Singapore
ERIK CAMBRIA, Nanyang Technological University, Singapore
YASHVARDHAN SHARMA, BITS Pilani, Pilani Campus, India

The current circumstances of the Arab world have provided bloggers and commenters with various subjects to discuss. Therefore, Arabic-generated content in social media is ramping up continuously. An informal written form of spoken Arabic called Arabizi has recently emerged as a commonly used language in the Arabic space, attracting great interest for sentiment analysis tasks. However, only a few sentiment resources exist, and state-of-the-art language models such as BERT and FastText do not consider Arabizi yet. This paper presents the first version of ArabiziVec, a set of pre-trained distributed word representations. ArabiziVec provides six different word embedding models to deal with Arabizi sentiment analysis challenges. The presented work surpasses all of the baseline sets for each experiment, regardless of whether the test set is from a previously published dataset or an extracted one. To the best of our knowledge, this is one of the first few resources that deals with Arabizi content and semantics in the context of sentiment analysis.

CCS Concepts: • **Information systems → Sentiment analysis**; • **Computing methodologies → Lexical semantics**; **Language resources**.

Additional Key Words and Phrases: Sentiment analysis, Arabizi, Semantic models, Deep learning

ACM Reference Format:
Asrita Venkata Mandalam, Oumaima Oueslati, Erik Cambria, and Yashvardhan Sharma. 2023. ArabiziVec: A Set of Arabizi Word Embeddings for Informal Arabic Sentiment Analysis. 1, 1 (December 2023), 23 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The rise of Arabic speakers in social media has bound an upsurge of Arabic-generated content. The importance of such content was further amplified due to the Arab spring [15, 28, 40]. The Arab world has become a key player in both economics and politics. Therefore, there is a greater need to understand, analyze, and measure Arabic social media content. Sentiment analysis of Arabic content figures among the newest trends and applications in social media analytics. However, this field is still in its infancy compared to the English language. A recent review by Oueslati et al. [42] studied significant research in the context of Arabic sentiment analysis in social media. They argued that, to develop this field further, informal content processing is required. Arab people in social media tend to express their opinion using their local dialects. They also use roman letters and numerals and bring words from other languages such as English, French, and Italian to convey

Authors' addresses: Asrita Venkata Mandalam, BITS Pilani, Pilani Campus, Vidya Vihar, Pilani, India, f20171179@pilani.bits-pilani.ac.in; Oumaima Oueslati, SenticNet, Singapore, oummaymma@gmail.com; Erik Cambria, Nanyang Technological University, Singapore, cambria@ntu.edu.sg; Yashvardhan Sharma, BITS Pilani, Pilani Campus, Vidya Vihar, Pilani, India, yash@pilani.bits-pilani.ac.in.

XXXX-XXXX/2023/12-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

information. The use of multiple languages in addition to Arabic, written using roman letters and numerals, is called Arabizi [3, 59].

The usage of Roman characters in Arabizi leads to a variety of spellings for a single word. For example, 'jamel', 'jameeel' and 'jamil' all mean 'nice' in English and are variations in spelling of the same word. Consolidating such variants in spelling is an important task. The difficulty of this task arises due to the inability to know every variant of spelling for each word in Arabizi. Due to this, standardizing the translation from Arabizi to Arabic is tough. Sometimes, English and French words are used along with Arabic in Arabizi as most Arabic speakers are bilingual or multilingual. While speaking informal Arabic, terms from other local languages are used as well. This phenomenon is known as code-switching and is prevalent among multilingual speakers. If one considers posts on Twitter, a popular social media platform, one can see Arabic words written in Latin characters intertwined with mentions of 'happy birthday' or 'ça va'. Another characteristic of Arabizi is the use of numbers instead of certain Arabic letters. For instance, 'tamatim' can also be written as '6ama6im' as the shape of some Arabic letters are similar to numbers, and it means 'tomato'.

Lo et al. [34] reviewed existing approaches used for multilingual sentiment analysis. Although they reviewed formal languages extensively, they also took into account informal and under-resourced languages. They noted that a hybrid framework might be useful for developing sentiment analysis tools for languages with limited resources. One of the most significant recent developments in Natural Language Processing (NLP), particularly semantic models for sentiment analysis, is the use of word embeddings [41]. Here, words are represented as vectors in a continuous space, capturing many syntactic and semantic relations among them. The benefits of using these distributed word representations have been highlighted in several NLP tasks, including multilingual sentiment analysis in social media [10, 49]. However, these benefits are mostly associated with the English language through the availability of several open-source word representation models in English [37, 50, 60, 61]. Arabic sentiment analysis systems fail in the context of social media due to the scarcity of such resources. Arab space in social media is well-known to generate informal content. People prefer using their local dialects and writing them using Latin letters and numbers. This informal way of expressing their opinion is called Arabizi.

Although there has been research in related topics [7, 9, 17, 55–58], there has not been extensive work in this field. As it is an under-resourced language, creating semantic models for sentiment analysis would aid in processing Arabizi content. The present paper proposes ArabiziVec, a distributed word representation, and an open-source project. ArabiziVec aims to provide the research community interested in Arabic NLP with free-to-use and qualitative word embedding models. To the best of our knowledge, there does not exist such a resource to deal with Arabizi content.

2 RELATED WORK

Sentiment analysis and sentiment polarity detection are crucial tasks that produce useful results used to control online bullying, understand product reviews, and identify the opinion of the public about political activities [27, 43, 53]. Over the past couple of years, great strides have been made in this field. Minaee et al. [39] provided a review of over 150 deep learning models and more than 40 popular datasets used in the past for text classification. They also analyzed the performance of these models on the prevailing baselines. To extract both past and future contexts while classifying a sentence according to the sentiment, Basiri et al. [11] proposed an Attention-based Bidirectional CNN-RNN Deep Model that performed well on both long and short data. Li et al. [31] used information about the sentiment lexicon to propose a sentiment padding method that improved upon the proportion of useful sentiment information in each of the user reviews taken as an input. This lexicon integrated two-channel CNN–LSTM model outperformed multiple existing baselines. The requirement for sarcasm detection has been on the rise due to its increased usage in online

forums and social media. Majumder et al. [35] proved that sentiment classification and sarcasm detection are related tasks and presented a multitask learning-based method. They used a Gated Recurrent Unit (GRU) based model to outperform existing state-of-the-art methods.

One of the most useful developments in sentiment analysis is the use of word embeddings [5, 13, 32, 54]. Rezaeinia et al. [48] proposed a new method, known as Improved Word Vectors, to improve the efficiency of word embeddings that have been pre-trained on large corpora for sentiment analysis. Word embeddings improve the performance of a sentiment classification model significantly. Using label information from sentiment lexicons, Li et al. [32] proved that word embeddings learned by incorporating document-level sentiment ratio on the target word significantly improved the performance of a sentiment analysis system. Rezaeinia et al. [47] created an improved version of word embeddings that increased the accuracy of pre-trained word embeddings in sentiment classification. The work of Elrazzaz et al. [21] justifies that language-specific word embeddings help achieve better performance, especially with Arabic text. Zhou et al. [64] realized that although bilingual word embeddings took words of two languages into account, it failed to account for the sentiment of words from both of these languages. Their word embeddings captured the unaccounted sentiment, and experiments proved that their embeddings outperformed the rest in all of the tested datasets. We believe that Arabizi textual content, which consists of Arabic, English, and words from other regional languages, can be approached with the same mindset and analysis.

Soliman et al. [51] provided a set of pre-trained word embeddings for Arabic. Using K-means clustering and the t-Distributed Stochastic Neighbor Embedding (t-SNE) tool by Ulyanov, they managed to capture the sentiment of the text. Even though they did not use any extensive feature engineering or complex machine learning models, their Skip-Gram and Continuous Bag of Words (CBOW) models gave results comparable to the final scores of SemEval-2017 Semantic Textual Similarity Task 1. Although theirs was done for Arabic text, Tobaili et al. [57] proved that word embeddings could be applied successfully to Arabizi as well. Their main goal was to create an Arabizi sentiment lexicon, also known as SenZi, for the Lebanese dialect. They attempted to improve upon it by using word embeddings for each of the sentiment words in Senzi. Word2Vec [36] has already proved that building word representations in vector space has worked for tasks that require English word embeddings. For Arabic word embeddings, Elrazzaz et al. [21] reviewed various techniques that could be used to generate Arabic word embeddings that include the CBOW model, the Skip-Gram model, GloVe [44], and the Arabic part of the Polyglot word embeddings. From their intrinsic and extrinsic results, it was shown that CBOW and the Skip-Gram model outperformed Polyglot.

There are not many papers that cater to dialectal Arabic (which is used frequently on social media) in terms of word embeddings [6, 16]. For example, Dahou et al. [16] created a huge web-crawled corpus to create word embeddings for Modern Standard Arabic (MSA) and dialectal Arabic text. After cleaning and normalizing the data, they used the Word2Vec tool to build their CBOW and Skip Gram models. Using the created word embeddings, they made a sentiment classification model to classify reviews and tweets. Results show that using a Convolutional Neural Network (CNN) model and initializing word vectors using pre-trained word embeddings helped the performance of their model. Although Altowayan and Tao [6] derived their corpus from different sources (Arabic newspapers and the holy Quran), they used the Word2Vec model as well. They did not attempt to tune their sentiment classifiers for better results. They noted that despite that, their Support Vector Machine (SVM) classifier and Logistic Regression classifier performed better than others in most of their baseline results.

Tobaili et al. [57] created an Arabizi Identification dataset and a Sentiment Analysis dataset. They used a lexicon-based approach to evaluate their Arabizi sentiment lexicon, SenZi. Using the approach proposed by Al-Twairesh et al. [1], SenZi attained a Recall of 79%, Precision of 66%,

and an F1-score of 72%. Their F1-score was 15% more than the baseline and showed that word forms and variants were important in sentiment classification. Duwairi et al. [20] used supervised learning to assign sentiment labels to tweets written in Arabizi. They decided to convert Arabizi tweets (Latin text) to Arabic text. They applied a Naïve Bayes and an SVM classifier to classify the tweets. As they realized that both classifiers misclassified neutral tweets, they changed their pipeline to identify objective from subjective tweets. As a result, the SVM model fared better than the Naïve Bayes model in terms of Precision (54.9% compared to 50.4%) and Recall (58.4% compared to 53.7%). They also noted that filtering, such as removing stopwords and mapping emoticons to their corresponding Arabic words, did not affect the Arabizi data significantly. Guellil et al. [25] followed a similar approach by transliterating the input message from Arabizi to Arabic by using a corpus extracted from Facebook. They used five different classifiers: SVM, Naïve Bayes, Logistic Regression, Decision Tree, and Random Forest. For them, their Naïve Bayes classifier worked the best with an F1-score of 78% and 76% for manual and automatic transliterated data, respectively. A comparison between their results and the results obtained by Duwairi et al. [20] could not be made as the data used by the latter was not available publicly. Baert et al. [9] created two datasets- one for Arabizi tweets and the other for Arabizi sentiment analysis. For the task of classification, they developed a BERT-based architecture for the same.

Tobaili [55] examined Twitter data across Lebanon and Egypt. He noted that 4.9% to 5.7% of the Twitter data of those countries is in Arabizi. This large amount of data might carry valuable information, and it is vital to study and develop a system to identify and classify sentiment. In contrast, as Arabizi has its meaning conveyed by Arabic and is written using Latin text, most sentiment classification models fail to classify Arabizi text. To the best of our knowledge, a set of word embeddings specific to Arabizi sentiment analysis has not been created yet. Thus, in this paper, we propose ArabiziVec, a distributed word representation, to provide the research community interested in informal Arabic NLP tasks with a set of qualitative word embedding models.

3 ARABIZIVEC DATA COLLECTION AND PREPARATION

3.1 Data Collection

(1) SenZi
The SenZi dataset Tobaili et al. [57] mentioned in the previous section was used as a test set in half of the experiments to test the effectiveness of the created word vectors on an externally created dataset.

(2) Dataset Extracted From the Top *n*-grams of the Senzi SA dataset
As the previously described dataset consisted of only 1,602 tweets, a new dataset was extracted from Twitter using the Python library - tweepy[1]. The first part of the dataset was extracted according to the top unigrams, bigrams, and trigrams of the SA dataset provided by SenZi. The top 100 tweets were extracted per keyword. Based on the Twitter users that posted the most frequently in Arabizi, the next part of the dataset was extracted. The data was refined by classifying those tweets according to the top n-grams for each sentiment of the SA dataset. We noted that this dataset covered a limited vocabulary. This dataset was used to identify which type of embeddings would prove to be the most useful.
For example, the bigram '*ya rab*' was one of the top bigrams of the SenZi SA dataset. The tweets extracted using it as a keyword can be seen in the table 1.

(3) Dataset Extracted from Lexicon of SenZi
Tobaili et al. [57] created an Arabizi lexicon consisting of around 13k negative and 11k positive words. Taking these words as the set of keywords, tweets were extracted for each word.

[1]https://www.tweepy.org/

Table 1. Example for the tweets extracted from the top *n*-grams of the SenZi SA dataset.

Original *n*-gram	Extracted Tweet
ya rab (If God wants)	happy birthday y a7la 7aga fe el donia de kolhaa we 32balll 1oooo sana ya rab
	ya rab ne istiyorsan benden onu diliyorum senden
	walikum salam gsubah bakhr have a great day and stay safe and healthyameen ya rabjazaak allah blessing
	thanks abby ameen ya rab

A majority of the extracted dataset was obtained from this lexicon and was collected over different weeks to overcome the previously observed limitation. There are 43,374 negative and 30,955 positive tweets in this dataset. It was used as the train set for half of the experiments. For the rest of them, it was split into a train and test set.
Table 2 shows a couple of the tweets extracted from the word '*saye2*'. This word was present in the SenZi lexicon that contained negative words.

Table 2. Example for the tweets extracted from the SenZi lexicon.

Original Word	Extracted Tweet
saye2 (Very bad)	El nas el betklmny te2oly enta feen we ba3d keda te2oly tab asr3 aw engez aw e5les lama ab2a ana el saye2 el taxi
	Edilo 110 3shan yom saye2 gedan gedan
	ana asfa kan ekhtyar saye2 meny
	Howa begad saye2 lel daraga

3.2 Preprocessing

Preprocessing the data prepares it for classification. This is done by cleaning the data and removing noise and other features that do not contribute to the overall sentiment of the sentence. It has been seen that preprocessing the input improves the classification accuracy significantly [8, 19, 26].

As some words were present in both the positive and negative lists, those words were removed before extracting according to the lexicon present in the list of each sentiment. The data was cleaned by removing retweet symbols, Twitter mentions (@ user_name), non-Arabizi characters, newline characters, tabs, and extra spaces. Finally, the extracted datasets were combined while ensuring that none of the tweets were repeated. The distribution of the same can be found in Table 3.

Table 3. Distribution of data across each class in each Dataset.

Source	Positive	Negative	Total
SenZi	801	801	1,602
Extracted From the top *n*-grams of the SenZi SA dataset	1,360	1,712	3,072
Extracted from the Lexicon of SenZi	30,955	43,374	74,329

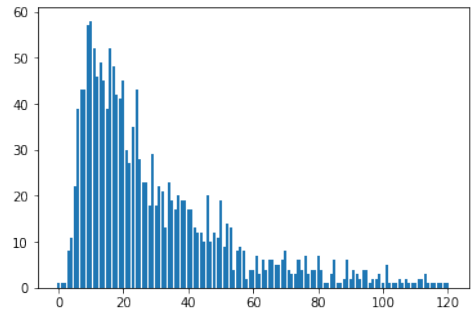

Fig. 1. Text length distribution for the Senzi Dataset.

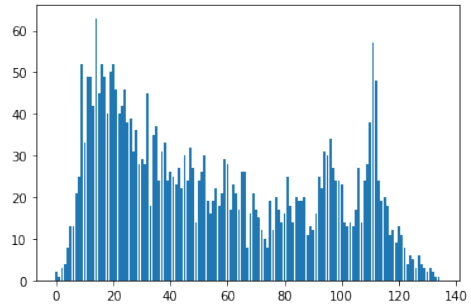

Fig. 2. Text length distribution for the dataset extracted from the top *n*-grams of the Senzi SA dataset.

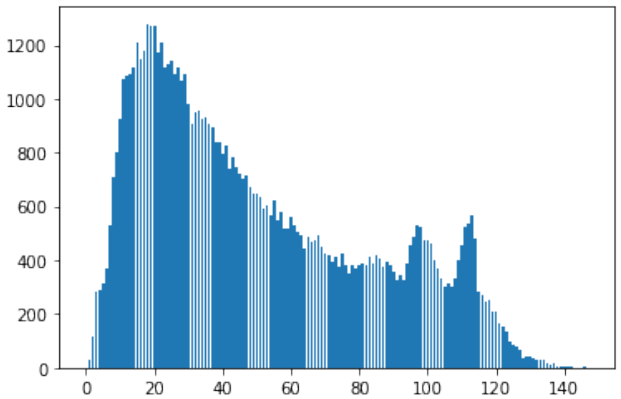

Fig. 3. Text length distribution for the dataset extracted from the Lexicon of SenZi

3.3 Statistical Description

In Table 4, we present a statistical description of our dataset. The average length of the SenZi SA dataset is the least, with a length of 33.1.

The total number of words and unique words are also the least out of the three datasets. The dataset extracted from the top *n*-grams of the SenZi SA dataset had the largest average length of the lot. The dataset extracted from the SenZi lexicon had the highest number of documents. Therefore, it had the highest number of words and unique words for both the balanced and unbalanced datasets. Figures 1, 2 and 3 show the distribution of the text length of the three datasets, specifically, the unbalanced datasets.

4 WORD EMBEDDINGS

Word embeddings play an important role in sentiment analysis due to their ability to capture the similarity, or lack of it, between different words regarding their contexts. Capturing this information and details about its appearance in a positive or negative connotation helps the classifier mark the sentiment of sentences it has not seen previously. Some of the state-of-the-art word embedding models are Word2Vec [36], GloVe [44], FastText [12], BERT [18] and ELMo [45].

Table 4. Statistical Description of the used Datasets.

Dataset	Number of Documents	Number of Unique Tokens	Total Number of Words	Average Length	Ratio
Senzi Dataset	1,602	4,783	9,628	33.1	Balanced
Extracted from the top *n*-grams of the Senzi SA dataset	2,720	9,658	30,903	56.55	Balanced
Extracted from the top *n*-grams of the Senzi SA dataset	3,072	10,874	35,700	57.85	Un-Balanced
Extracted from the Lexicon of SenZi	61,910	116,809	605,456	50.97	Balanced
Extracted from the Lexicon of SenZi	74,329	132,858	732,001	51.26	Un-Balanced

4.1 Techniques

Word2Vec was used as multiple works focusing on embedding-creation for Arabic have used the same [21, 51]. As most previous work on Arabic word embeddings explored CBOW, the presented work decided to explore CBOW as well, along with Skip-Gram, GloVe, and FastText. Embeddings such as GloVe capture global context, something that Word2Vec does not include. Although it is known that Word2Vec architectures along with negative sampling are more efficient in the case of Arabic word embeddings [52], this paper tests the same for Arabizi. FastText was used as it incorporates the use of sub-word information. This feature is useful as there are cases when a word in Arabizi is written differently in different tweets but may contain certain common sub-words with its other written forms.

WORD2VEC was created by Mikolov et al. [36] in 2013. It is used to identify and detect similarities between words in terms of their meaning and usage in text. Each word in the trained corpus is identified by a set of vectors and is then used in the form of embeddings for various Natural Language Tasks such as sentiment analysis and recommendation systems. There are two architectures used to create Word2Vec word embeddings - CBOW and Skip-Gram.

The CBOW model takes into account the context of the surrounding words before making a prediction. For example, in the sentence 'the dog jumped over the pond', 'jumped' can be predicted by looking at the rest of the words of the sentence. To decide how many context words need to be used, a window size is set. Suppose the window size is two, the words 'dog' and 'over' are used together as context words to determine the target word 'jumped'. Figure 4 gives a representation of how CBOW works. As the number of context words is set to 2, two 1xN input vectors will be used in the input layer. 'N' represents the size of the sentence. It is then passed to a hidden layer which multiplies the vectors with an NxM matrix. 'M' is the dimensionality of the output vectors. It produces output vectors with a size of 1xM. They are summed up to produce the output.

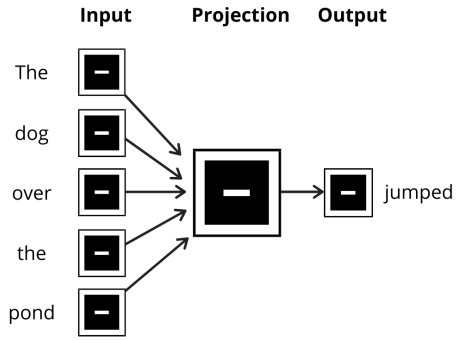

Fig. 4. CBOW Model.

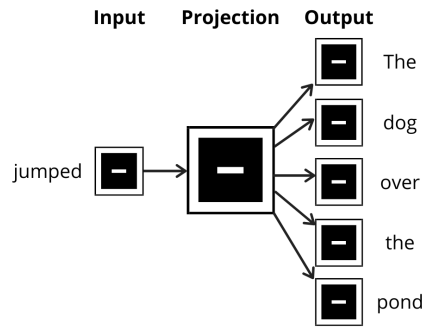

Fig. 5. Skip-Gram Model.

The Skip-Gram architecture predicts the context words using the target word. Taking the same example as above, the target word 'jumped' helps predict context words such as 'dog' and 'over'. Depending on the window size, the context word 'pond' can also be predicted from 'jumped'. The working of this model can be seen in Figure 5. Here, the input is one target word and the output of this model is W (1xM) vectors representing the predicted context words. W is the window size. Skip-Gram is known to work better than the CBOW method as it can capture multiple meanings for the same word. For example, the word 'chair' can mean a seat or the person in charge of a meeting. The same word will have two different vector representations.

Word2Vec works well as it captures the context while assigning vectors to each word in the vocabulary. This model can be used for any dataset, regardless of size. Unfortunately, this model does not capture rare or out of vocabulary (OOV) words. Another disadvantage of both Skip-Gram and CBOW is the difficulty of finding optimal values for their parameters.

GLOVE embeddings [44] capture global context. Similar to Word2Vec, it also captures local context. It is a count-based model and uses a co-occurrence matrix of size NxN, where N is the number of words in the given corpus. It uses the ratio between the probabilities of the co-occurrence of two different words to detect word similarity. GloVe is advantageous as it is faster than Word2Vec embeddings. It also assigns a lower weight for stopwords such as 'the' and 'a', thus increasing the accuracy of the model. Similar to the Word2Vec model, it cannot handle OOV words during its implementation.

FASTTEXT [12] is a library that aids in word representation and text data classification. It was developed by Bojanowski et al. and unlike the previously discussed embeddings, it takes into account rare and OOV words. Using a bag of character n-grams to represent words in the given corpus, it creates a vector representation for each word. Sometimes, rare words have common subwords with those existing in the input corpus. For OOV words, FastText takes care of these with their character level n-gram model. For example, if the words 'Subset' and 'Categorical' exist in the train set but the word 'Subcategory' exists only in the test set, FastText will connect the meanings of the words in the train set to the new word in the test set. This characteristic of the architecture is useful in processing Arabizi as some words that have the same meaning and spelling in Arabic may be written in various ways when using the Roman script. For example, '*jamel*', '*jamil*' and '*jameel*' have the same meaning. If the first two lie in the training data and the latter in the test data, FastText embeddings will be able to identify that they have a similar meaning due to the sub-word '*jam*' and character '*l*' at the end of the words.

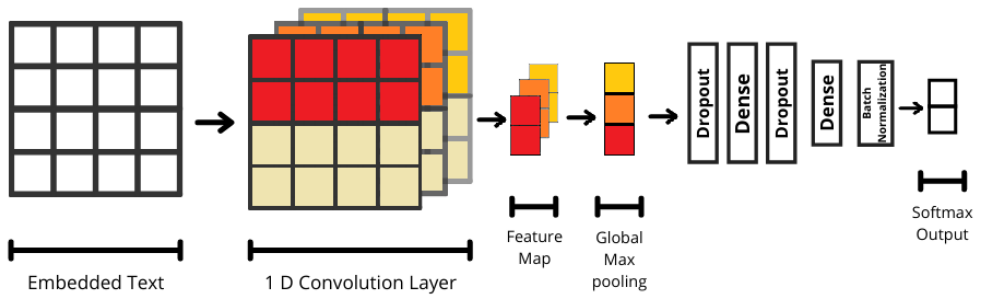

Fig. 6. Diagram of the CNN model used to test the embeddings.

4.2 Building the models

Word2Vec embeddings [36] have proven to be useful in Arabic sentiment analysis. Many have used CBOW and Skip-Gram for their experiments [21, 51]. After obtaining the vocabulary of the train set, the Word2Vec vectors for CBOW and Skip-Gram are created. After setting the dimensionality as 100 for both the CBOW and Skip-Gram vectors, the model is trained for 30 epochs. A negative sampling size of 2 was used. Negative sampling helps minimize the similarity score of words that do not frequently occur together [22, 38]. Words that had a frequency equal to or greater than two were considered while creating the embeddings. A window size of 3 is used. This is done to ensure that the distance between the predicted and current word is not more than 3. As Tweets can have a maximum length of 140 characters, keeping the window size higher would consider unrelated words and reduce the effectiveness of the word vectors generated. The initial and final learning rates are 0.1 and 0.001 respectively. The learning rate drops linearly as the model is being trained. The vectors of the CBOW and Skip-Gram models are concatenated. This is then used to create an embedding matrix and the input data before passing it to the neural networks.

GloVe [44] was trained differently as compared to the Word2Vec embeddings. The trained GloVe embeddings were converted to the Word2Vec format using a function from the Gensim library to observe the vectors of different words. Then, the embeddings generated from the data were appended to it. The classifiers then used these embeddings.

The **FastText** embedding creation followed a path similar to that of the creation of the CBOW and Skip-Gram embeddings. For this representation of the dataset, the vectors were created with a dimensionality of 100, a window size of 5. A window size of 3 was originally used, but the embeddings worked more effectively when the size was 5. It was trained for five epochs. A negative sampling size of 10 was used. The created FastText embeddings were an extension of the Skip-Gram model, except that subword information was used. Grave et al. [23] proved its effectiveness. They used FastText along with CBOW vectors to create word representation for 157 languages.

Although some of the previous work done in the field of word embeddings for Arabic mentioned that the Word2Vec model worked the best for them [2, 4, 21, 51], this work aimed to explore and compare the results of different word embeddings before choosing one to move forward with one or a combination of two embeddings. There were three word embedding models that were compared. The first consisted of a combination of CBOW and Skip-Gram vectors, the second were the GloVe embeddings, and the third was a combination of CBOW and FastText vectors. All three embeddings were trained from scratch and were tested on the CNN and LSTM models described below.

(1) CNN model

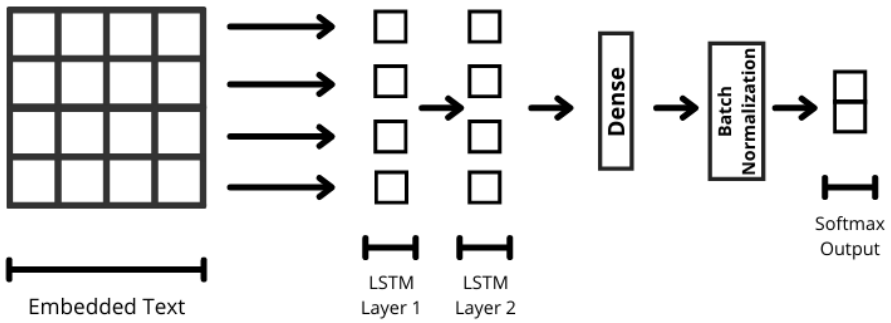

Fig. 7. Diagram of the LSTM model used to test the embeddings.

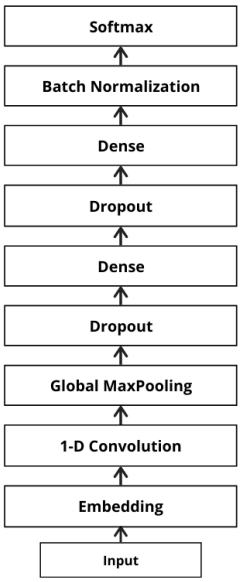

Fig. 8. The CNN model used to test the embeddings.

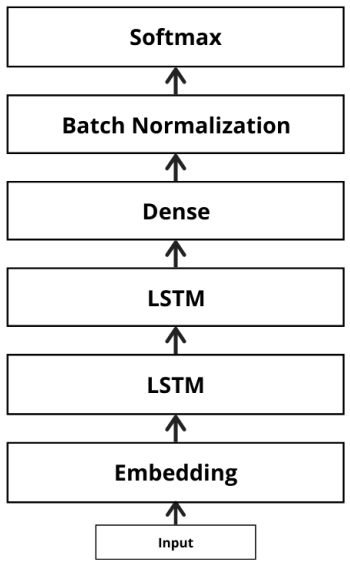

Fig. 9. The LSTM model used to test the embeddings.

The CNN model consisted of an embedding layer that took the embedding matrix created earlier as an input. After using a 1D convolutional layer with 32 output filters and a convolutional window size of 2, the model uses max pooling to get the top features. The dropout layers have a rate of 0.4 and batch normalization has been used to prevent overfitting. Adam optimizer has been used with a learning rate of 0.001. The structure of the model can be seen in Figures 6 and 8.

(2) LSTM model
The LSTM model, as shown in Figures 7 and 9, also had an embedding layer for the embedding matrix created by the previous step. It had two LSTM layers. The dimensionality of the output

Table 5. Classification report used to select the embeddings.

Word Embedding	Model	Precision	Recall	F1-Score	Accuracy
CBOW + Skip-Gram	CNN	**0.94**	**0.94**	**0.94**	**0.94**
	LSTM	0.77	0.61	0.56	0.60
GloVe	CNN	0.93	0.93	0.93	0.93
	LSTM	0.90	0.90	0.90	0.90
CBOW + FastText	CNN	0.72	0.63	0.56	0.63
	LSTM	**0.93**	**0.93**	**0.93**	**0.93**

Table 6. Details about each word embedding model.

Model Name	Number of Documents	Number of Unique Tokens	Minimum Word Frequency	Window Size	Technique
CBOW_ub	74329	132858	2	3	CBOW
SG_ub	74329	132858	2	3	Skip-Gram
FastText_ub	74329	132858	5	5	FastText
CBOW_b	61910	132858	2	3	CBOW
SG_b	61910	132858	2	3	Skip-Gram
FastText_b	61910	132858	5	5	FastText

space for the first was 64 and it had a dropout rate of 0.3. For the second layer, the output dimensionality was 32, with a dropout rate of 0.3 as well. Batch normalization was used.

Table 5 shows that the CBOW and Skip-Gram embeddings perform the best for the CNN model with an F1-score of 0.94. However, for the LSTM model, the CBOW and FastText word embeddings perform the best with an F1-score of 0.93. Thus, the CBOW and Skip-Gram embeddings were chosen, along with the FastText and CBOW embeddings. The dataset used here was the dataset that was extracted from the top *n*-grams of the Senzi dataset.

Table 5 showed that the CBOW and Skip-Gram model performed the best for the CNN model. For the LSTM model, the CBOW and FastText embeddings attained the highest score. As the GloVe embeddings performed poorer in both models, it was not used to test the rest of the datasets. Although GloVe does discount stopwords, the language of Arabizi does not contain them, thereby eliminating the need for one of the advantages of using GloVe embeddings. As mentioned in section 2, CBOW and Skip-Gram embeddings were a popular choice for creating Arabic word embeddings. It is also not surprising that the CBOW and FastText embeddings perform well as a similar methodology with a few tweaks was used to create word embeddings for 157 languages [23]. For the rest of the experiments, the dataset that was extracted from the top *n*-grams of the Senzi dataset was not used as it contained a limited vocabulary. This was due to the fact that it was extracted based on the top *n*-grams of a small dataset, the SenZi dataset. As a result, the word embeddings were very specific and Table 5 had a set of high values.

The details about the created and trained CBOW, Skip-Gram, and FastText word embeddings can be seen in Table 6. The CBOW_ub, SG_ub, and FastText_ub models have been trained on an unbalanced dataset, while the rest have been trained on a balanced dataset.

5 DEEP LEARNING CLASSIFIERS

Deep learning models for text classification using Keras [14] were used to compare our models with other models. It would help us gain a fair idea about how our embeddings model fared with a simple neural network (in terms of the number of layers). Two experiments were performed. The first round did not include Word2Vec embeddings, while the second round did. The models, along with a short description of their components, are given below.

TextCNN

This model was proposed by Kim [29] and implemented as shown in Figure 10. After using an

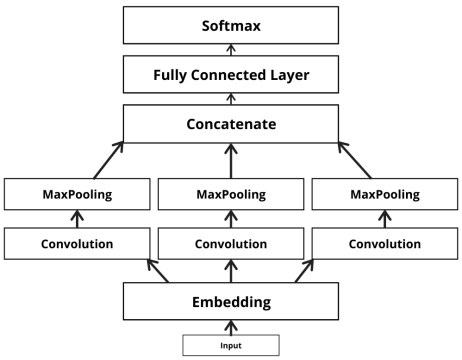

Fig. 10. TextCNN.

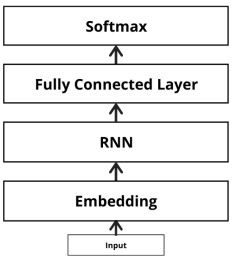

Fig. 11. TextRNN

embedding layer that took the embedding matrix as an input, it used a convolutional layer with multiple filter sizes. The model proposed in this paper used sizes of 3, 4, and 5. Max-pooling was then used to capture the most important features from the feature maps. Finally, a fully connected softmax layer was used. The output was a distribution of the probabilities of each class.

TextRNN

Liu et al. [33] proposed this model. It uses a recurrent neural network (RNN) layer to classify text. The tested model used an output dimensionality of 128. The components of this model can be seen in Figure 11.

TextBiRNN

This model is an improved version of the TextRNN model. All of the components are similar to the TextRNN model, except that it uses a bidirectional RNN layer instead of the RNN layer. The output dimensionality of the bidirectional RNN layer is 128 as well. The model can be seen in Figure 12.

TextAttBiRNN

This model is based on the implementation in the paper by Raffel et al. [46]. The implemented model uses an Attention layer that helps different states (in terms of time) of the same model for direct long-term dependencies. A tensor with the shape (samples, steps, features) is taken as an input, and a tensor with the shape (samples, features) is given as an output by this layer. It is based on the Feed-Forward Attention mechanism. After this, a fully connected softmax layer was used. The final output was a distribution of the probabilities of each class.

HAN

The Hierarchical Attention Networks (HAN) model for document classification was proposed by Yang et al. [62]. There are four stages in this architecture. The first part is the word encoder. A

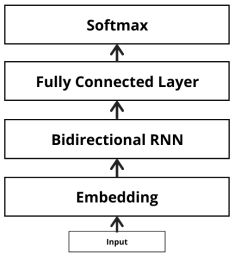

Fig. 12. TextBiRNN.

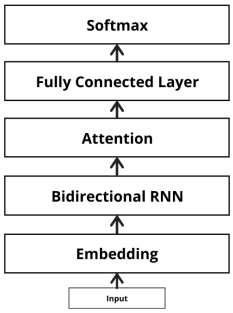

Fig. 13. TextAttBirnn

sentence is taken as an input and each word is embedded to vectors using the embedding matrix created previously. Annotations of these words are obtained using a bidirectional GRU. This helps summarize information from both directions of a sequence for a specific word. The implemented model had an output dimensionality of 128. The word attention layer was used to extract words that contributed to the meaning of a sentence. A summation of the most important words created a sentence vector. The attention layer used here was based on the Feed-Forward Attention mechanism, similar to the TextAttBiRNN model. Every step till here was done on the same time step, and it was achieved using a timeDistributed layer. The methodology used in the first stage was used again to encode the previously calculated sentence vectors. The Attention layer was used once again to measure the importance of each sentence and summarize the importance of the overall document. Finally, a fully connected layer was used to classify the text.

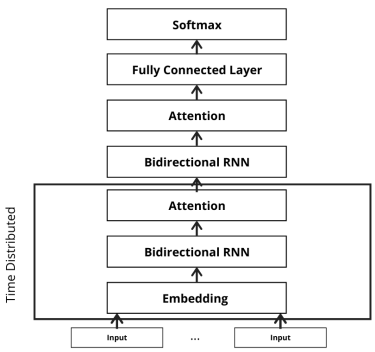

Fig. 14. HAN.

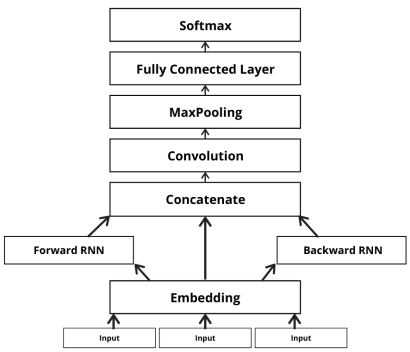

Fig. 15. RCNN.

RCNN

The Recurrent Convolutional Neural Network (RCNN) model was proposed by Lai et al. [30]. The first stage of this model uses a bidirectional recurrent neural network to capture the left and right contexts of each word. The left context is captured during the forward feed and the right context is obtained during the backward scan of the input text. Each of the RNN layers in the model proposed by this paper used an output dimensionality of 128. A linear transformation along with the tanh activation function is used to produce the input to the next stage - text representation learning.

The number of output filters of the 1-D convolutional layer is 64. The convolutional window had a size of 1. In the text representation learning stage, a max-pooling layer is used to extract essential information in the document. A fully connected softmax layer was used to output a distribution of the probabilities of each class.

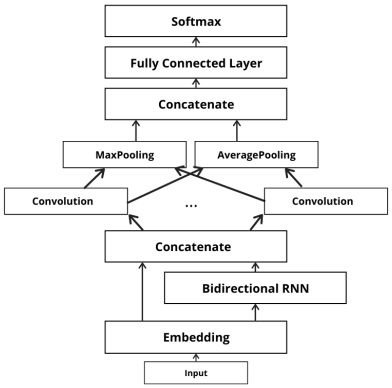

Fig. 16. RCNNVariant.

RCNNVariant

This model is an improved version of the RCNN model. The previous model used three inputs as it had the current, right, and left contexts. This one uses a single input. Instead, a bidirectional LSTM is used to encode the context. Multiple layers of CNNs are used, with different kernel sizes ranging from 1 to 5 and an output dimensionality of 128. Instead of using the tanh activation function, the rectified linear activation function (ReLU) is used. Both average and max pooling have been used and concatenated. It has been seen that a combination of both performs better than both each pooling method by itself. It also reduces overfitting [63] [24].

6 EVALUATION

In Figure 17, we illustrate the general workflow of the proposed work.

6.1 Similarity scores of Sentiment words

For the language of English, there are certain benchmark vectors to identify whether the given word embeddings have been able to represent the vectors of words from the data correctly. As Arabizi is an under-resourced language, a set of words were taken to identify and compare their similarity scores. Table 7 represents the words taken for the benchmark and their similarity scores.

The words '*jamel*', '*helow*' and '*hassan*' translate to 'good' or 'nice' in English while '*wehesh*', '*khara*' and '*saye2*' mean 'bad'. A comparison of words with opposite meanings was made and their similarity scores have been written in bold in the table. Some of the opposite words receive a negative score in some cases, indicating that their vector values are not similar. Except for the Skip-Gram embeddings that were trained on a skewed dataset, the rest of the embeddings had a higher similarity score for words with similar meaning, compared to the words with opposite meanings. In sarcastic social media comments and tweets, positive words are used negatively, and similarly, negative terms with a positive connotation. This leads to a higher similarity score between positive and negative words than what was expected.

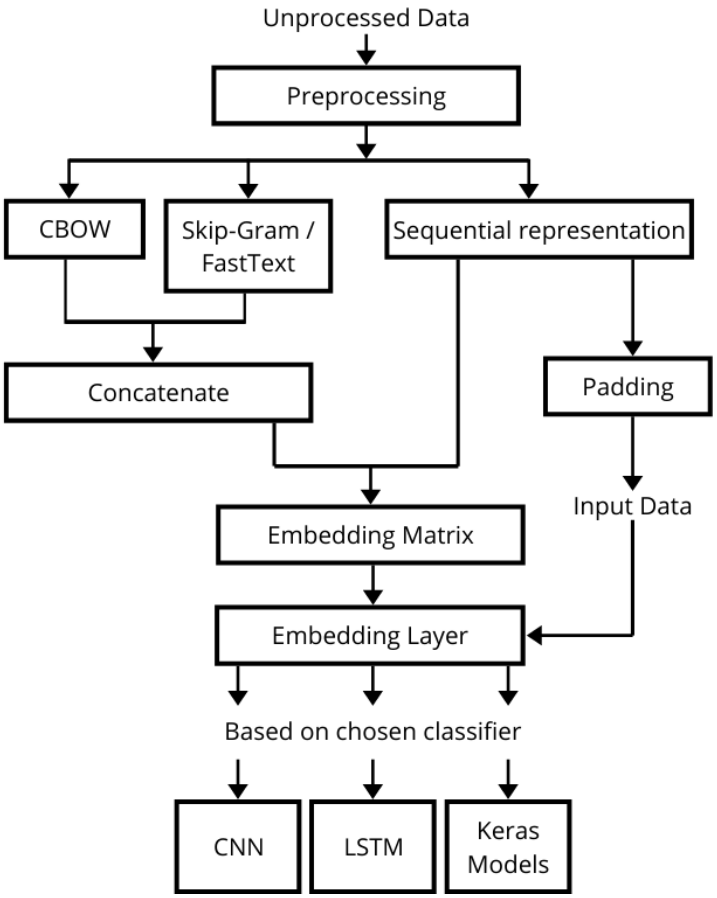

Fig. 17. General diagram of the proposed model.

6.2 Baseline

(1) Extracted Tweets as Train Set, Senzi Dataset as Test Set
This experiment has a baseline value of 60%. As the proposed models use the embeddings that were trained on the extracted tweets to test an unrelated dataset, this paper aims to achieve an F1-score of at least 60% for each of the Keras models.
(2) Extracted Tweets as Train and Test Set
This experiment has a baseline value of 70%. In this experiment, the extracted dataset was split with 80% of it being used to train embeddings and the other 20% used to test the trained embeddings. The proposed methodology aims to achieve an F1-score of at least 70% for each of the Keras models.

6.3 ArabiziVec Sentiment Classification

Table 8 shows the results obtained while training the model on the extracted dataset and testing it on the Senzi SA dataset.

As the extracted dataset had more negative than positive tweets, the negative tweets were shuffled and reduced to the same count as that of the positive tweets. Here, the train set did not

Table 7. Similarity scores of words with similar and opposite meanings.

Model	Word 1	Word 2	Similarity Score
CBOW	jamel	helow	0.06
	jamel	hassan	0.23
	helow	hassan	0.11
	wehesh	khara	0.24
	wehesh	saye2	0.29
	khara	saye2	0.06
	wehesh	**helow**	**-0.05**
Skip-Gram	jamel	helow	0.19
	jamel	hassan	0.31
	helow	hassan	0.05
	wehesh	khara	0.36
	wehesh	saye2	0.38
	khara	saye2	0.25
	saye2	**helow**	**0.08**
FastText	jamel	helow	0.56
	jamel	hassan	0.75
	helow	hassan	0.50
	wehesh	khara	0.55
	wehesh	saye2	0.89
	khara	saye2	0.53
	khara	**helow**	**0.46**

contain any tweets that were common with those in the Senzi SA dataset and the tweets extracted from its top *n*-grams. The baseline for this experiment is 0.60 for the F1-score. From the table, it can be seen that all of the models that used the CBOW and Skip-Gram embeddings surpassed this baseline. For the Keras models, the TextRNN model performed the best with an F1-score of 0.71. The lowest F1-score achieved is at 0.66 and was obtained by the RCNNVariant model. For the CBOW and FastText embeddings, the highest value obtained by the Keras models was 0.71 and was obtained by the HAN model. The lowest scoring Keras model was the TextRNN model with an F1-score of 0.67. However, for the simple neural network models, the CNN model received a low F1-score of 0.33. The LSTM model scored a value of 0.71 for the F1-score.

To check if training a skewed dataset would attain better results, the models were trained on the extracted dataset and tested on the Senzi SA dataset. The train set did not contain any tweets that were common with those in the Senzi SA dataset and the tweets extracted from its top *n*-grams. There were more negative than positive tweets. Table 9 gives the results obtained for the same.

The baseline to beat was 0.60 for this experiment as well. For the CBOW and Skip-Gram embeddings, all of the tested classifiers surpassed the baseline. The highest scoring model was TextRNN, with an F1-score of 0.73. The lowest-scoring one was the RCNNVariant model, with a score of 0.66. The highest scoring Keras model for the CBOW and FastText embeddings was the HAN model, with a value of 0.72. The lowest-scoring models were the TextCNN and TextAttBiRNN models with an F1-score of 0.69. Among the tested simple neural network models, the LSTM model scored 0.71, and the CNN model obtained an F1-score of 0.36.

Table 10 gives the classification report obtained by training and testing the models on the extracted dataset. The number of positive and negative tweets was equal. 80% of the dataset was

Table 8. Classification report of the extracted dataset as train set, the Senzi SA dataset as test set. Same ratio of positive to negative tweets.

			Precision	Recall	F1-Score	Accuracy
CBOW + Skip-Gram Embeddings	Simple NN Models	CNN	0.68	0.68	0.68	0.68
		LSTM	0.71	0.70	0.70	0.70
	Keras Models	TextCNN	0.67	0.67	0.67	0.67
		TextRNN	0.71	0.71	0.71	0.71
		TextBiRNN	0.70	0.70	0.70	0.70
		TextAttBiRNN	0.70	0.69	0.69	0.69
		HAN	0.69	0.69	0.69	0.69
		RCNN	0.69	0.68	0.68	0.68
		RCNNVariant	0.68	0.67	0.66	0.67
CBOW + FastText Embeddings	Simple NN Models	CNN	0.25	0.50	**0.33**	**0.50**
		LSTM	0.71	0.71	0.71	0.71
	Keras Models	TextCNN	0.68	0.68	0.68	0.68
		TextRNN	0.68	0.68	0.67	0.68
		TextBiRNN	0.68	0.68	0.68	0.68
		TextAttBiRNN	0.70	0.70	0.70	0.70
		HAN	0.71	0.71	0.71	0.71
		RCNN	0.68	0.68	0.68	0.68
		RCNNVariant	0.69	0.69	0.69	0.69

used for training and 20% for testing the classifiers. As the dataset was extracted and not obtained from any previous works, the baseline is 0.70. For the CBOW and Skip-Gram embeddings, the simple neural networks performed well with F1-scores of 0.82 and 0.88 for the CNN and LSTM models respectively. Among the tested Keras models, the HAN and TextAttBiRNN models scored the highest with an F1-score of 0.84. The lowest scoring model was the TextCNN model, with a value of 0.72. For the CBOW and FastText embeddings, the highest-scoring Keras model is the HAN model, with a score of 0.87. The lowest scoring Keras model was the TextCNN model with a score of 0.71. For the simple neural network models, the CNN and LSTM models attained F1-scores of 0.81 and 0.89 respectively.

When more negative than positive tweets are used in the dataset, the results can be seen in table 11. The extracted dataset has been used as the train and test set and has been split in the ratio 80:20. Among the results for the Keras models that used the CBOW and Skip-Gram embeddings, the TextCNN model scored the lowest with a value of 0.74. The TextAttBiRNN and HAN models scored the highest with an F1-score of 0.88. For the simple neural network models, the CNN and LSTM scored 0.85 and 0.90 respectively. For the CBOW and FastText embeddings, the HAN model scored the highest with an F1-score of 0.88. All of the Keras models that used this set of embeddings surpassed the baseline F1-score. For the simple neural network models, the CNN scored a value of 0.82. The LSTM model obtained a value of 0.90.

In Tables 8, 9, 10 and 11, 'Simple NN Models' refer to the Simple Neural Network (NN) models that were discussed in section 4.2 which were used to select the set of embeddings to be used in these experiments.

A comparison of the performances of the various models and datasets can be seen in Figure 18. Here, Experiment 1, 2, 3 and 4 refer to tables 8, 9, 10 and 11 respectively.

Table 9. Classification report of the extracted dataset as train set, the Senzi SA dataset as test set. Skewed ratio of positive to negative tweets.

			Precision	Recall	F1-Score	Accuracy
CBOW + Skip-Gram Embeddings	Simple NN Models	CNN	0.71	0.69	0.69	0.69
		LSTM	0.72	0.71	0.71	0.71
	Keras Models	TextCNN	0.68	0.67	0.67	0.67
		TextRNN	0.74	0.73	0.73	0.73
		TextBiRNN	0.69	0.69	0.69	0.69
		TextAttBiRNN	0.70	0.70	0.70	0.70
		HAN	0.71	0.71	0.71	0.71
		RCNN	0.70	0.70	0.70	0.70
		RCNNVariant	0.66	0.66	0.66	0.66
CBOW + FastText Embeddings	Simple NN Models	CNN	0.64	0.51	**0.36**	**0.51**
		LSTM	0.71	0.71	0.71	0.71
	Keras Models	TextCNN	0.69	0.69	0.69	0.69
		TextRNN	0.71	0.71	0.71	0.71
		TextBiRNN	0.70	0.70	0.70	0.70
		TextAttBiRNN	0.73	0.70	0.69	0.70
		HAN	0.74	0.72	0.72	0.72
		RCNN	0.70	0.70	0.70	0.70
		RCNNVariant	0.72	0.70	0.70	0.70

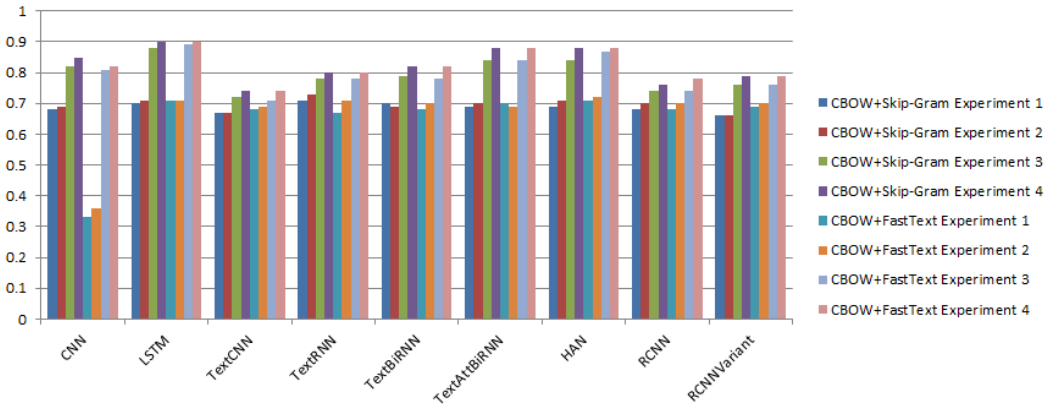

Fig. 18. A comparison of the F1-scores obtained by the four experiments.

7 DISCUSSION

The first set of experiments on the 'Extracted from Lexicon of SenZi' dataset use the SenZi Dataset as the test set and the 'Extracted from Lexicon of SenZi' dataset as the train set. The results are illustrated in Table 8 and Table 9. The train set did not contain any tweets that were common with the SenZi Dataset or Dataset Extracted From the Top *n*-grams of the Senzi SA dataset to avoid any bias towards the vocabulary of the test set. For the 'Extracted from Lexicon of SenZi' dataset, we found that all of the Keras models cleared the baseline of 0.60 for both the equal and skewed ratio

Table 10. Classification report of the extracted dataset as train and test set. Same ratio of positive to negative tweets.

			Precision	Recall	F1-Score	Accuracy
CBOW + Skip-Gram Embeddings	Simple NN Models	CNN	0.82	0.82	0.82	0.82
		LSTM	0.88	0.88	0.88	0.88
	Keras Models	TextCNN	0.73	0.73	0.72	0.73
		TextRNN	0.79	0.78	0.78	0.78
		TextBiRNN	0.79	0.79	0.79	0.79
		TextAttBiRNN	0.85	0.85	0.84	0.85
		HAN	0.86	0.84	0.84	0.84
		RCNN	0.74	0.74	0.74	0.74
		RCNNVariant	0.76	0.76	0.76	0.76
CBOW + FastText Embeddings	Simple NN Models	CNN	0.83	0.82	0.81	0.82
		LSTM	0.89	0.89	0.89	0.89
	Keras Models	TextCNN	0.71	0.71	0.71	0.71
		TextRNN	0.79	0.79	0.78	0.79
		TextBiRNN	0.78	0.78	0.78	0.78
		TextAttBiRNN	0.86	0.84	0.84	0.84
		HAN	0.87	0.87	0.87	0.87
		RCNN	0.74	0.74	0.74	0.74
		RCNNVariant	0.76	0.76	0.76	0.76

of tweets. Although the simple CNN model that used the CBOW and FastText embeddings did not clear the baseline F1-score (for both the same and skewed ratio of positive to negative tweets), the results of the Keras models were focused on as they are the standard set of Keras classifiers. The highest F1-scores achieved were the same for both sets of embeddings.

Tables 10 and 11 report the results of the second set of experiments. They used 80% of the 'Extracted from Lexicon of SenZi' dataset as the train set and the remaining as the test set. All the Keras models tested on the CBOW and Skip-Gram embeddings cleared this baseline. The CBOW and FastText embeddings were tested on the same dataset and set of Keras models and cleared the baseline as well. When the same set of embeddings (CBOW and FastText) were tested on a skewed ratio of negative to positive tweets, all of the Keras models surpassed the baseline F1-score. For the skewed dataset, all of the models cleared the baseline as well.

We noted that the CBOW and FastText embeddings performed slightly better than the CBOW and Skip-Gram embeddings in the case of a skewed dataset. Comparing the scores attained by the various Keras models, the embeddings had similar scores except for the RCNN model. For this model, the CBOW and FastText embeddings performed a bit better. Regardless of the balance of the dataset, the embeddings performed well, and some of the classifiers managed to score an F1-score 18% over the baseline of 70%. According to the Frequently Asked Questions (FAQ) on the FastText website[2], the FastText embeddings perform well on unbalanced data due to the hierarchical softmax loss. For the language of Arabizi, the classification of positive tweets are easier than negative tweets as their vocabulary is more limited than that of the latter. As FastText embeddings can produce vectors for unknown words using substrings of known words, the higher number of negative tweets aid in the same.

[2]https://fasttext.cc/docs/en/faqs.html

Table 11. Classification report of the extracted dataset as train and test set. Skewed ratio of positive to negative tweets.

			Precision	Recall	F1-Score	Accuracy
CBOW + Skip-Gram Embeddings	Simple NN Models	CNN	0.86	0.85	0.85	0.85
		LSTM	0.90	0.90	0.90	0.90
	Keras Models	TextCNN	0.74	0.74	0.74	0.74
		TextRNN	0.81	0.81	0.80	0.81
		TextBiRNN	0.82	0.82	0.82	0.82
		TextAttBiRNN	0.88	0.88	0.88	0.88
		HAN	0.88	0.88	0.88	0.88
		RCNN	0.76	0.76	0.76	0.76
		RCNNVariant	0.79	0.79	0.79	0.79
CBOW + FastText Embeddings	Simple NN Models	CNN	0.85	0.83	0.82	0.83
		LSTM	0.91	0.91	0.90	0.91
	Keras Models	TextCNN	0.74	0.74	0.74	0.74
		TextRNN	0.82	0.81	0.80	0.81
		TextBiRNN	0.82	0.82	0.82	0.82
		TextAttBiRNN	0.89	0.88	0.88	0.88
		HAN	0.88	0.88	0.88	0.88
		RCNN	0.78	0.78	0.78	0.78
		RCNNVariant	0.79	0.79	0.79	0.79

Overall, the proposed word embeddings models cleared the baseline of 0.60 when the SenZi Dataset was taken as the test set. When the 'Extracted from Lexicon of SenZi' dataset was used as the train and test set, the word embedding models managed to surpass the baseline F1-score of 0.70. For both sets of experiments, some of the Keras models managed to score more than 0.10 above the baseline (more than 0.70 and 0.80 respectively), proving that the proposed word embeddings are useful and can potentially be used for a variety of tasks.

8 CONCLUSION

This paper proposed ArabiziVec, a new distributed word representation for Arabizi, and an open-source project. It provides the field of Arabic NLP with free-to-use and qualitative word embedding models. Two types of word embeddings were presented in this paper: the first type consists of CBOW and Skip-Gram embeddings, the second is created using CBOW and FastText.

The proposed word embedding models surpass common baselines. Regarding the word embeddings models, CBOW and Skip-Gram embeddings performed the best in the case of a balanced dataset. For the case where there were more negative than positive tweets, the CBOW and FastText embeddings performed better. An error analysis was presented to understand the reason for different embeddings working better in different scenarios.

Although the proposed baseline for the experiments that used an 80:20 split for the dataset was set at an F1-score of 0.70, six out of nine models have surpassed that score and achieved 0.80 and beyond for both, the CBOW and Skip-Gram embeddings as well as the CBOW and FastText embeddings. This proves that ArabiziVec enhances the accuracy of sentiment analysis tasks. In the future, we plan to extend ArabiziVec to other tasks, e.g., question answering and dialogue systems.

ACKNOWLEDGMENTS

The authors would like to convey their sincere thanks to the Department of Science and Technology (ICPS Division), New Delhi, India, for providing financial assistance under the Data Science (DS) Research of Interdisciplinary Cyber Physical Systems (ICPS) Programme [DST/ICPS/CLUSTER/Data Science/2018/Proposal-16:(T-856)] at the department of computer science, Birla Institute of Technology and Science, Pilani, India.

Authors are also thankful to the authorities of Birla Institute of Technology and Science, Pilani, to provide basic infrastructure facilities during the preparation of the paper.

REFERENCES

[1] Nora Al-Twairesh, Hend Al-Khalifa, and AbdulMalik Al-Salman. 2016. Arasenti: large-scale twitter-specific Arabic sentiment lexicons. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 697–705.

[2] Abdulaziz M Alayba, Vasile Palade, Matthew England, and Rahat Iqbal. 2018. Improving sentiment analysis in Arabic using word representation. In *2018 IEEE 2nd International Workshop on Arabic and Derived Script Analysis and Recognition (ASAR)*. IEEE, 13–18.

[3] Wid H Allehaiby. 2013. Arabizi: An Analysis of the Romanization of the Arabic Script from a Sociolinguistic Perspective. *Arab World English Journal* 4, 3 (2013).

[4] Maram Almaghrabi and Girija Chetty. 2019. Investigating Deep Learning Word2vec Model for Sentiment Analysis in Arabic and English languages for User's reviews. In *2019 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*. IEEE, 1–6.

[5] Felipe Almeida and Geraldo Xexéo. 2019. Word embeddings: A survey. *arXiv preprint arXiv:1901.09069* (2019).

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

[7] Tahani H Alwaneen, Aqil M Azmi, Hatim A Aboalsamh, Erik Cambria, and Amir Hussain. 2021. Arabic Question Answering System: A Survey. *Artificial Intelligence Review, 1-47* (2021).

[8] Giulio Angiani, Laura Ferrari, Tomaso Fontanini, Paolo Fornacciari, Eleonora Iotti, Federico Magliani, and Stefano Manicardi. 2016. A Comparison between Preprocessing Techniques for Sentiment Analysis in Twitter.. In *KDWeb*.

[9] Gaétan Baert, Souhir Gahbiche, Guillaume Gadek, and Alexandre Pauchet. 2020. Arabizi Language Models for Sentiment Analysis. In *Proceedings of the 28th International Conference on Computational Linguistics*. 592–603.

[10] Amira Barhoumi, Nathalie Camelin, Chafik Aloulou, Yannick Estève, and Lamia Hadrich Belguith. 2020. Toward Qualitative Evaluation of Embeddings for Arabic Sentiment Analysis. In *Proceedings of The 12th Language Resources and Evaluation Conference*. 4955–4963.

[11] Mohammad Ehsan Basiri, Shahla Nemati, Moloud Abdar, Erik Cambria, and U Rajendra Acharya. 2021. ABCDM: An attention-based bidirectional CNN-RNN deep model for sentiment analysis. *Future Generation Computer Systems* 115 (2021), 279–294.

[12] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146.

[13] Yanqing Chen, Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2013. The expressive power of word embeddings. *arXiv preprint arXiv:1301.3226* (2013).

[14] François Chollet et al. 2015. Keras. https://keras.io.

[15] Francesca Comunello and Giuseppe Anzera. 2012. Will the revolution be tweeted? A conceptual framework for understanding the social media and the Arab Spring. *Islam and Christian–Muslim Relations* 23, 4 (2012), 453–470.

[16] 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: Technical papers*. 2418–2427.

[17] Kia Dashtipour, Mandar Gogate, Erik Cambria, and Amir Hussain. 2021. A Novel Context-Aware Multimodal Framework for Persian Sentiment Analysis. *Neurocomputing* 457 (2021), 377–387.

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[19] Rehab Duwairi and Mahmoud El-Orfali. 2014. A study of the effects of preprocessing strategies on sentiment analysis for Arabic text. *Journal of Information Science* 40, 4 (2014), 501–513.

[20] Rehab M Duwairi, Mosab Alfaqeh, Mohammad Wardat, and Areen Alrabadi. 2016. Sentiment analysis for Arabizi text. In *2016 7th International Conference on Information and Communication Systems (ICICS)*. IEEE, 127–132.

[21] 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 (Volume 2: Short Papers)*. 454–458.

[22] Yoav Goldberg and Omer Levy. 2014. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722* (2014).

[23] Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. *arXiv preprint arXiv:1802.06893* (2018).

[24] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. 2018. Recent advances in convolutional neural networks. *Pattern Recognition* 77 (2018), 354–377.

[25] Imane Guellil, Ahsan Adeel, Faical Azouaou, Fodil Benali, Ala-eddine Hachani, and Amir Hussain. 2018. Arabizi sentiment analysis based on transliteration and automatic corpus annotation. In *Proceedings of the 9th workshop on computational approaches to subjectivity, sentiment and social media Analysis*. 335–341.

[26] Emma Haddi, Xiaohui Liu, and Yong Shi. 2013. The role of text pre-processing in sentiment analysis. *Procedia Computer Science* 17 (2013), 26–32.

[27] Moez Ben Hajhmida and Oumayma Oueslati. 2020. Predicting mobile application breakout using sentiment analysis of facebook posts. *Journal of Information Science* (2020), 0165551520917099.

[28] Philip N Howard, Aiden Duffy, Deen Freelon, Muzammil M Hussain, Will Mari, and Marwa Maziad. 2011. Opening closed regimes: what was the role of social media during the Arab Spring? *Available at SSRN 2595096* (2011).

[29] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).

[30] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*.

[31] Wei Li, Luyao Zhu, Yong Shi, Kun Guo, and Erik Cambria. 2020. User reviews: Sentiment analysis using lexicon integrated two-channel CNN–LSTM family models. *Applied Soft Computing* 94 (2020), 106435.

[32] Yang Li, Quan Pan, Tao Yang, Suhang Wang, Jiliang Tang, and Erik Cambria. 2017. Learning word representations for sentiment analysis. *Cognitive Computation* 9, 6 (2017), 843–851.

[33] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101* (2016).

[34] Siaw Ling Lo, Erik Cambria, Raymond Chiong, and David Cornforth. 2017. Multilingual sentiment analysis: from formal to informal and scarce resource languages. *Artificial Intelligence Review* 48, 4 (2017), 499–527.

[35] Navonil Majumder, Soujanya Poria, Haiyun Peng, Niyati Chhaya, Erik Cambria, and Alexander Gelbukh. 2019. Sentiment and sarcasm classification with multitask learning. *IEEE Intelligent Systems* 34, 3 (2019), 38–43.

[36] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[37] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in Pre-Training Distributed Word Representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

[38] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546* (2013).

[39] Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. 2021. Deep Learning based Text Classification: A Comprehensive Review. *Comput. Surveys* 54, 3 (2021), Article 62.

[40] Dhekra Najar and Slim Mesfar. 2017. Opinion mining and sentiment analysis for Arabic on-line texts: application on the political domain. *International Journal of Speech Technology* 20, 3 (2017), 575–585.

[41] Hien T Nguyen, Phuc H Duong, and Erik Cambria. 2019. Learning short-text semantic similarity with word embeddings and external knowledge sources. *Knowledge-Based Systems* 182 (2019), 104842.

[42] Oumaima Oueslati, Erik Cambria, Moez Ben HajHmida, and Habib Ounelli. 2020. A review of sentiment analysis research in Arabic language. *Future Generation Computer Systems* 112 (2020), 408–430.

[43] Oumayma Oueslati, Moez Ben Hajhmida, Habib Ounelli, and Erik Cambria. 2019. Sentiment Analysis of Influential Messages for Political Election Forecasting. CICLing.

[44] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[45] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

[46] Colin Raffel and Daniel PW Ellis. 2015. Feed-forward networks with attention can solve some long-term memory problems. *arXiv preprint arXiv:1512.08756* (2015).

[47] Seyed Mahdi Rezaeinia, Ali Ghodsi, and Rouhollah Rahmani. 2017. Improving the accuracy of pre-trained word embeddings for sentiment analysis. *arXiv preprint arXiv:1711.08609* (2017).

[48] Seyed Mahdi Rezaeinia, Rouhollah Rahmani, Ali Ghodsi, and Hadi Veisi. 2019. Sentiment analysis based on improved pre-trained word embeddings. *Expert Systems with Applications* 117 (2019), 139–147.
[49] Caroline Sabty, Mohamed Islam, and Slim Abdennadher. 2020. Contextual Embeddings for Arabic-English Code-Switched Data. In *Proceedings of the Fifth Arabic Natural Language Processing Workshop*. 215–225.
[50] Ehsan Sherkat and Evangelos E Milios. 2017. Vector embedding of wikipedia concepts and entities. In *International conference on applications of natural language to information systems*. Springer, 418–428.
[51] Abu Bakr Soliman, Kareem Eissa, and Samhaa R El-Beltagy. 2017. Aravec: A set of arabic word embedding models for use in arabic nlp. *Procedia Computer Science* 117 (2017), 256–265.
[52] Dima Suleiman and Arafat Awajan. 2018. Comparative study of word embeddings models and their usage in Arabic language applications. In *2018 International Arab Conference on Information Technology (ACIT)*. IEEE, 1–7.
[53] Yosephine Susanto, Erik Cambria, Bee Chin Ng, and Amir Hussain. 2021. Ten Years of Sentic Computing. *Cognitive Computation* (2021), 1–19.
[54] Duyu Tang, Furu Wei, Bing Qin, Nan Yang, Ting Liu, and Ming Zhou. 2015. Sentiment embeddings with applications to sentiment analysis. *IEEE transactions on knowledge and data Engineering* 28, 2 (2015), 496–509.
[55] Taha Tobaili. 2016. Arabizi identification in twitter data. In *Proceedings of the ACL 2016 Student Research Workshop*. 51–57.
[56] Taha Tobaili. 2020. *Sentiment Analysis for the Low-Resourced Latinised Arabic" Arabizi"*. Ph.D. Dissertation. The Open University.
[57] Taha Tobaili, Miriam Fernandez, Harith Alani, Sanaa Sharafeddine, Hazem Hajj, and Goran Glavaš. 2019. SenZi: A Sentiment Analysis Lexicon for the Latinised Arabic (Arabizi). In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*. 1203–1211.
[58] David Vilares, Haiyun Peng, Ranjan Satapathy, and Erik Cambria. 2018. BabelSenticNet: A Commonsense Reasoning Framework for Multilingual Sentiment Analysis. In *IEEE SSCI*. 1292–1298.
[59] Mohammad Ali Yaghan. 2008. "Arabizi": A contemporary style of Arabic Slang. *Design issues* 24, 2 (2008), 39–52.
[60] Ikuya Yamada, Akari Asai, Jin Sakuma, Hiroyuki Shindo, Hideaki Takeda, Yoshiyasu Takefuji, and Yuji Matsumoto. 2020. Wikipedia2Vec: An Efficient Toolkit for Learning and Visualizing the Embeddings of Words and Entities from Wikipedia. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 23–30.
[61] Xiao Yang, Craig Macdonald, and Iadh Ounis. 2018. Using word embeddings in twitter election classification. *Information Retrieval Journal* 21, 2 (2018), 183–207.
[62] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*. 1480–1489.
[63] Dingjun Yu, Hanli Wang, Peiqiu Chen, and Zhihua Wei. 2014. Mixed pooling for convolutional neural networks. In *International conference on rough sets and knowledge technology*. Springer, 364–375.
[64] Huiwei Zhou, Long Chen, Fulin Shi, and Degen Huang. 2015. Learning bilingual sentiment word embeddings for cross-language sentiment classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 430–440.