ORIGINAL ARTICLE

Swarm intelligence for handling out-of-vocabulary in Arabic Dialect Identification with different representations

Mahmoud Sobhy[1] · Ahmed H. AbuElAtta[1] · Ahmed A. El-Sawy[1,2] · Hamada Nayel[1,3]

Received: 30 September 2024 / Accepted: 9 July 2025 / Published online: 14 August 2025

Abstract
With the rise in popularity of social networks and programs that let users connect instantaneously, communication has become more dynamic. So, regularly occurring new words affect the quality of representation models and make spelling errors. As the natural language processing applications depend on vector representations of texts, out-of-vocabulary (OOV) terms are unfamiliar to the models and must be handled with degrading their quality. For this, we present an OOV handling approach based on four swarm intelligence techniques, ant colony optimization, chicken swarm optimization, gray wolf optimization, and particle swarm optimization. In this study, three word embedding models have been used to obtain the representation of words. The performance of the proposed methods is evaluated on three tasks, dialect identification, sentiment analysis, sarcasm detection, and the results show that the suggested methods are promising for handling OOV and demonstrated high performance in all experiments. GWO-OOV-SVM achieved a 53.43% F1-score for dialect identification, while CSO-OOV-SVM achieved 75.66% and 57.68% F1-scores for sentiment analysis and sarcasm detection respectively, exceeding other models.

Keywords Arabic Dialect Identification · Word embedding · Gray Wolf Optimizer · Chicken Swarm Optimization · Text classification

JEL Classification D8 · H51

Mathematics Subject Classification 35A01 · 65L10 · 65L12 · 65L20 · 65L70

1 Introduction

The volume of unstructured data available online has grown to such an extent that handling it with human resources has become intolerable as the world has become more digitally connected. Tasks in natural language processing (NLP) were created to automatically handle it by utilizing computer power. These duties include, but are not limited to, opinion mining, audio transcription, translation, test grading, and assessment of written summaries [1]. Spoken dialects are becoming more and more prevalent in textual social media and communication channels in this day of hyperconnectivity. Since there are no standards for writing in dialects, informal language, processing a dialect is a significant challenge to current NLP technology in general [1].

When people switch between the standard and the dialect inside sentences, it exacerbates the issue since informal language can be considered a dialect of the language that is very different and divergent from the language standard. While Arabic vernaculars differ significantly from it to form dialects known as Dialectal Arabic (DA), which are frequently used in informal settings like the web and social media, Modern Standard Arabic (MSA), as the name suggests, is the official standard for the Arabic language usually used in formal structure [2].

MSA possesses a significantly greater abundance of NLP tools and resources in comparison with a clear deficit in similar resources for DA. The combination of MSA and DA in utterances constitutes a serious problem of Out-of-Vocabulary (OOV) in NLP applications. The OOV problem presents two challenges: entirely unseen terms that do not appear in the training data, and homograph OOVs, where a word is present in the training data but carries a different meaning or connotation. [3].

When OOV arise, the majority of predictive models for NLP tasks perform worse because of information loss brought on by the model representation's inability to handle OOV terms correctly; therefore, this highlights the necessity for additional DA research. One possible solution is to assign a unique random vector to each OOV word or to use a single random vector to represent all OOV words. [4]. However, these straightforward strategies provide extremely few details about OOV to learning models in downstream tasks. Ignoring words during the procedure of composition gives no information regarding a word that is missing from the phrase. Consequently, current research proposes various methods to infer the embeddings of OOV words by analyzing their structure and the context in which they appear [5, 6], or to substitute OOV with words from the lexicon [7].

Regarding models of distributed representation, handling OOV words can be obtained with simple methods. As OOV terms have no representations, they can be disregarded but, this makes the prediction model fit data without being aware of a word's absence. So, one random vector can be used to represent all OOV words

Identification of Arabic dialects has been done at the provincial (Cairo, Al-Madinah) [8], national (Egypt, Saudi Arabia) [9], and regional (Levant, Gulf) [10]. The problem of distinguishing Arabic regional dialects for users of social media is the focus of this work. This work introduces new methods for identifying various regional Arabic dialects using a big vocabulary constructed from four different WE models while the representation of OOV words is determined using swarm intelligence techniques.

DA is frequently used on social media sites. By collecting data from these sources, computational linguists may create enormous datasets applicable to contexts of statistical methods. Since the character set and substantial vocabulary are consistent among all Arabic dialects, it is difficult to distinguish and separate them from one another. Along with MSA, which is generally not spoken as a first language, there are six significant Arabic regional dialects:

1. Egypt: The most well recognized and comprehended dialect, as a result of Egypt's robust movie and television sectors. In addition to its noteworthy impact over a substantial segment of the 20th century [11].
2. Levant: A collection of Aramaic-related dialects that, while having somewhat varied intonations and sounds, are essentially similar when written [12].
3. Gulf: The Gulf-region Arabic dialect gave rise to the regional dialect that is most similar to MSA as it exists now. Despite the differences, the Gulf dialect has kept more of MSA's verb conjugation than other variations [13].
4. Iraqi: Despite having unique characteristics of its own regarding inflection of verbs, sound, and prepositions, it is sometimes categorized among the Gulf dialects [13].
5. Maghreb: This dialect was greatly influenced by French and Berber. Oral in oral form, speakers from other Middle Eastern nations may find it difficult to understand the westernmost dialects [14].

Because there are few datasets available, the morphology of the Arabic language is complex, and the majority of the datasets that are available are skewed. In its early stages, Arabic research received little attention, particularly when it came to dialect identification. The elevated level of dialect resemblance presents a number of difficulties, especially when uttering brief phrases, like:

- In the Arabic language, there is a similarity between Levantine Arabic, Syria, Lebanon, Jordan, Palestine, and Iraqi Arabic. These dialects share many common words and grammatical structures, which can result in confusion. For example, the word بيتى (pronounced "bayti") means "my house" in both dialects. However, differences in pronunciation and certain expressions can lead to misunderstandings. In Iraqi Arabic, شلونك (pronounced "shlonak") means "how are you," while in Levantine Arabic, the phrase is كيفك (pronounced "keefak"). Such resemblances make it difficult for speakers to distinguish between the dialects, complicate linguistic research, and pose challenges for learners trying to grasp the unique nuances of each dialect.
- There are several brief expressions with identical meanings in the same tongue. For example, in Egyptian dialect, the words هايل (pronounced "hayel"), روعه (pronounced "rawaa"), تحفه (pronounced "tohfa"), يجنن (pronounced "yganin"), رايق (pronounced "rayek") means "amazing" or "wonderful".

In conclusion, we provide the subsequent contributions:

1. We construct a big vocabulary to get the representation of words in the dataset using Word2vec and fastText pre-trained models, and a fined-tuned model trained on training data.
2. We apply the data augmentation as a preprocessing step to elevate the training data quality.
3. We assemble a new corpus from diverse Arabic datasets for the development of our CBOW and fastText models.
4. We implement four swarm intelligence techniques, ACO, CSO, GWO, and PSO, to get the best representation of OOV words.

The rest of the paper is structured as follows: The related work is outlined in Sect. 2. Section 3 describes the dataset in Sect. 3. Our proposed system is explained in Sect. 4. The results on standard data set and discussion are presented in Sect. 5. Finally, the conclusion with some further observations in Sect. 6.

2 Related work

Various significant work has been carried out for Arabic Dialect Identification (ADI). In [9], NADI 2020, the initial Nuanced Arabic Dialect Identification Shared Task, has been introduced. Subtask 1 of this collaborative project is to identify Arabic countries, while subtask 2 is to identify Arabic provinces. The NADI 2020 dataset,[1] which was collected from Twitter, includes 100 provinces and 21 Arab countries. The Google mBERT model, which served as the baseline, was improved with eight batches and a maximum sequence length of 50 tokens. Many techniques, including machine learning (ML) and deep learning (DL), have been used for NADI 2020. The top-performing model obtained 26.78% F1-score for subtask1 and 6.39% F1-score for subtask2. The subsequent shared task, NADI 2021, sought to determine the linguistic variety of short texts in Arabic dialects based on limited geographic locations of origin [8]. An unlabeled corpus of 10 M tweets has been uploaded to NADI 2021 and is available for optional use.[2] ML and DL techniques were combined with the baseline model utilized in NADI 2020 to refine it further. The F1-scores for county-level-MSA and province-level-MSA were 22.38% and 32.26% respectively, where for country-level-dialect Arabic and province-level-dialect Arabic were 6.43% and 8.60%, respectively. A different NADI shared objective in 2022[3] was to identify Arabic country-level dialects [9]. Together with certain BERT-based pre-trained models, three baselines, XLMR, mBERT, and MARBERT,

[1] NADI 2020 dataset.
[2] NADI 2021 dataset.
[3] NADI 2022 dataset.

were refined in NADI 2022. For test sets, Test-A and Test-B, the top systems reported F1-scores of 36.48% and 18.95%, respectively.

The Fourth NADI (NADI 2023) [15] focused on Arabic dialects and machine translation.[4] Teams utilized various techniques such as N-gram, TF-IDF, neural networks, and pre-trained language models to tackle three subtasks: dialect identification, dialect-to-MSA machine translation, and open track dialect-to-MSA machine translation. Out of 58 team registrations, 76 valid entries were submitted by 18 teams. The best performing models achieved F1-score 87.27%, F1-scores of 14.76%, and F1-score of 21.10% for Subtask 1, Subtask 2 and Subtask 3 respectively. In [16], the Fifth NADI (NADI 2024) is discussed, which concentrated on identifying Arabic dialects,[5] Arabic level of dialectness estimation, and dialect-to-MSA machine translation. The competition included three subtasks: dialect identification, dialect scoring, and machine translation. Participants were allowed to submit up to five runs for each subtask, with only the highest scoring submission retained. Evaluation metrics included F1-score for dialect identification, RMSE for dialect scoring, and BLEU score for machine translation. The results indicated room for improvement in all tasks, with challenges such as limited dialect data and evaluation references. The best F1-score achieved in the competition was 50.57% F1-score for subtask1, and the best performance (RMSE) for subtask2 was 0.1403, and for subtask3, the best BLEU was 20.44.

The AraBERT model was presented by Antoun et al. [17]. It was trained on a sizable Arabic corpus and attained cutting-edge performance on a variety of tasks in Arabic NLP, such as named entity recognition, sentiment analysis, and question answering. Twelve encoder blocks, seventy-six hidden dimensions, 512 maximum sequence length, and 110 M total parameters made up the BERT-base configuration. The model performs better than other earlier methods including the multilingual BERT version. Abdul-Mageed et al. presented another transformer-based model [10] that was created for the comprehension of the Arabic language. With an emphasis on MSA and Arabic dialects, respectively, they introduced MARBERT and ARBERT, bidirectional transformers designed for Arabic language processing. To train MARBERT, 1 billion Arabic tweets were chosen at random, creating a dataset of roughly 6 billion tweets. Only tweets with more than two Arabic words were counted. With batch size of 256 and sequence length of 128, MARBERT was learned for 36 epochs and produced excellent results in a variety of Arabic dialect datasets.

Using a BERT architecture for NADI, Talafha et al. [18] were able to attain a 26.78% F1-score. Additionally, an Arabic-BERT model along with ensemble techniques and data augmentation for NADI was provided by Gaanoun and Benelallam [19]. The train data was split into three segments and mixed for each nation in order to undertake data augmentation after the training data supplied was used to train the Arabic-BERT model. Several models, including the "Mix" model, were trained using the enriched data. This model performed well, obtaining a F1-score of 5.75% for province-level data and 23.26% for country-level data. In [20], N-GRAM features combined with BERT were shown. The challenge of identifying dialects at the country levels and provincial levels was examined by the writers. They presented an ensemble model with encouraging outcomes. M-NGRAM makes use of a stochastic gradient descent (SGD) and TF-IDF with word n-grams and character n-gram. For province-level identification and country-level identification, the ensemble approach produced F1-scores of 6.39% and 25.99%, respectively.

AbuElAtta et al. [21] presented an author profiling model based on ML for Arabic users, focusing on regional dialect identification. The study utilized the ArSarcasm dataset and implemented various classification algorithms. Data augmentation techniques were applied to enhance training data quality, leading to improved model performance. The methodology comprised data augmentation as a text preprocessing step. The best f-score 50.52% was achieved by CBOW and SVM with rbf kernel. The SVM algorithm demonstrated high performance, exploring the application of different classification algorithms or more advanced language representation models like BERT could be avenues for future research to further enhance the model's capabilities and robustness.

[4] NADI 2023 dataset.

[5] NADI 2024 dataset.

For OOV handling issue, considerable work has been done. A word is considered OOV by a representation model when it appears in a sample. A sentence or document contains an OOV word, it is often ignored, and no information related to it is incorporated into the resulting vector if the representation model cannot handle it correctly while creating a vector to represent it. With an increase in the amount of OOV terms per sample, this information deficit tends to deteriorate the predictive models' performance.

Simple replacement techniques, such as replacing each OOV word with either unique random vector or a new random vector, are simple methods for handling OOVs. Another option is the zero-vector representation substitution, which is appropriate for preventing that particular OOV word's neural network from activating [22]. Even while more complex techniques for managing OOVs were unable to pick up new representations, they nevertheless produced positive results. As an example, Khodak et al. [23] suggested averaging the embeddings of samples that are closest to an OOV word. Although their proposed strategy performed better than disregarding the OOV terms, it was difficult to capture complicated semantic relationships.

Hu et al. [24] demonstrated that the representation techniques that are capable of learning new embeddings to OOV handling typically use the context in which the OOV is placed as a source of information [24, 25]. FastText [24] a well-liked distributed representation approach that links the words' vector representation to their morphological structure, or subword. With the help of this function, FastText can use its capabilities of morphological learning to handle OOV or uncommon words. Each word representation in this embedding model is made up of a bag of character n-grams with the actual word. The fastText n-gram encoding of the word "model" with n = 3 is<mo, mod, ode, del, el>. To separate n-grams from words, boundaries such as< and> are applied. Mimick [25] utilizes the word structure to express an OOV, just like FastText. It is a DL model that creates a vector representation by utilizing each character in the OOV word. The word's characters are supplied into a neural network, and the goal is a trained embedding model that contains a known representation of the word. The neural network's goal function is to reduce the gap between a word's letters and its recognized vector in the representation model. The main limitation of morphologically based approaches is their inability to handle words that are OOV and have variable meanings depending on the context. No matter what context an OOV word occurs in, its representation will always be the same if morphological information is the only source used to create it.

Although the majority of conventional language models are statistical, more recent methods use DL architectures to achieve improved accuracy when performing neural network language model tasks. RoBERTa [26] is a novel model that leverages bigger mini batches and learning rates to enhance performance on a variety of NLP tasks while eliminating the previous BERT's next sentence pretraining target. The new dataset, which uses public news articles, is several times larger than the one used for BERT training, and it is related to the efficacy of this new model. DistillBERT is a more compact general-purpose language representation model that can be optimized for particular NLP domain requirements, as explained by Sanh et al. [27]. According to the authors' research, knowledge distillation during pretraining can cut the size of the BERT by 40% while maintaining 97% of language comprehension skills and generating a model 60% faster.

3 Data

To achieve the generalization and robustness of this study, the experiments were applied on two different datasets. The first dataset is ArSarcasm [28], a set of sentiment analysis datasets for Arabic collected from SemEval 2017 [29] and ASTD [30] is the dataset that was used in this study. There are 10,547 tweets in the dataset that have had dialect labels added. The second dataset is ArSarcasm-V2 [31], which provided by WANLP 2021 Shared Task as an extension of ArSarcasm [28]. ArSarcasm-V2 has 15,548 tweets labeled for dialect, sentiment, and sarcasm. Figure 1 displays the distribution of the test and train sets across all classes.

The datasets include the following features:

- Tweet: the source text.

https://doi.org/10.1007/s00521-025-11501-1

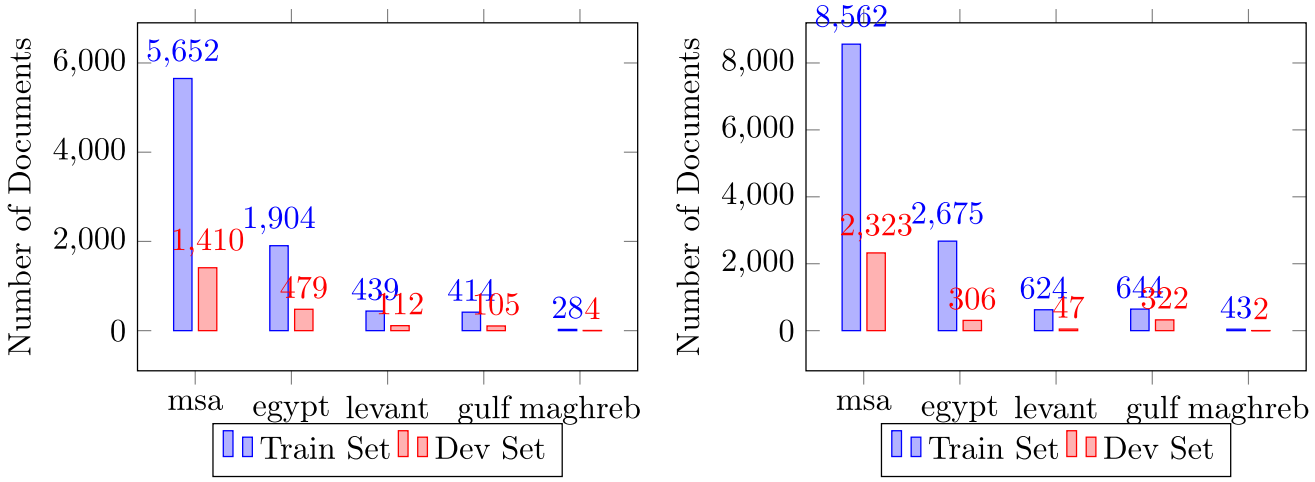
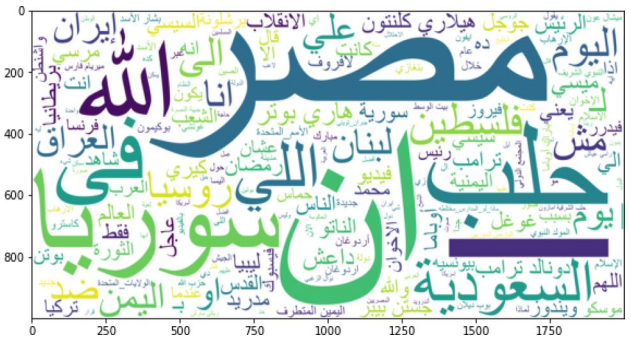


Fig. 1 Distribution of Train set and Dev set documents in two datasets

Fig. 2 WordCloud of Arsarcasm dataset



- Sarcasm: indicates if the tweet contains sarcasm.
- Sentiment: the sentiment classification (positive, neutral, or negative).
- Source: the origin of source tweet.
- Dialect: the region dialect used in tweet: egypt, levant, gulf, maghreb, or msa.

As a beneficial for text analysis, a word cloud [32] is a visual tool that displays the most frequently occurring words in a dataset. This type of visualization enables quick identification of the main themes and key terms in the data. For instance, in an Arabic language dataset, a word cloud can reveal the most common words and phrases, offering insights into the dataset's central themes and content. Figure 2 shows the word cloud of words in ArSarcasm dataset [28]. In Fig. 2, the prominent words include مصر (Egypt), سوريا (Syria), الله (God), العراق (Iraq), السعودية (Saudi Arabia), and فلسطين (Palestine), indicating that these terms are highly significant within the text. The word cloud is densely packed, with words oriented in multiple directions, creating a visually engaging and informative representation of the key terms in the dataset. For the training set, an empty vocabulary, referred to BIG_VOCAB, is initially created.

4 Methodology

Overall structure of the proposed system is given in Fig. 3. The process begins with the preprocessing step for Train set after it has augmented and a development (Dev) set. For Train set, an empty vocabulary, referred to BIG_VOCAB, is initially created.

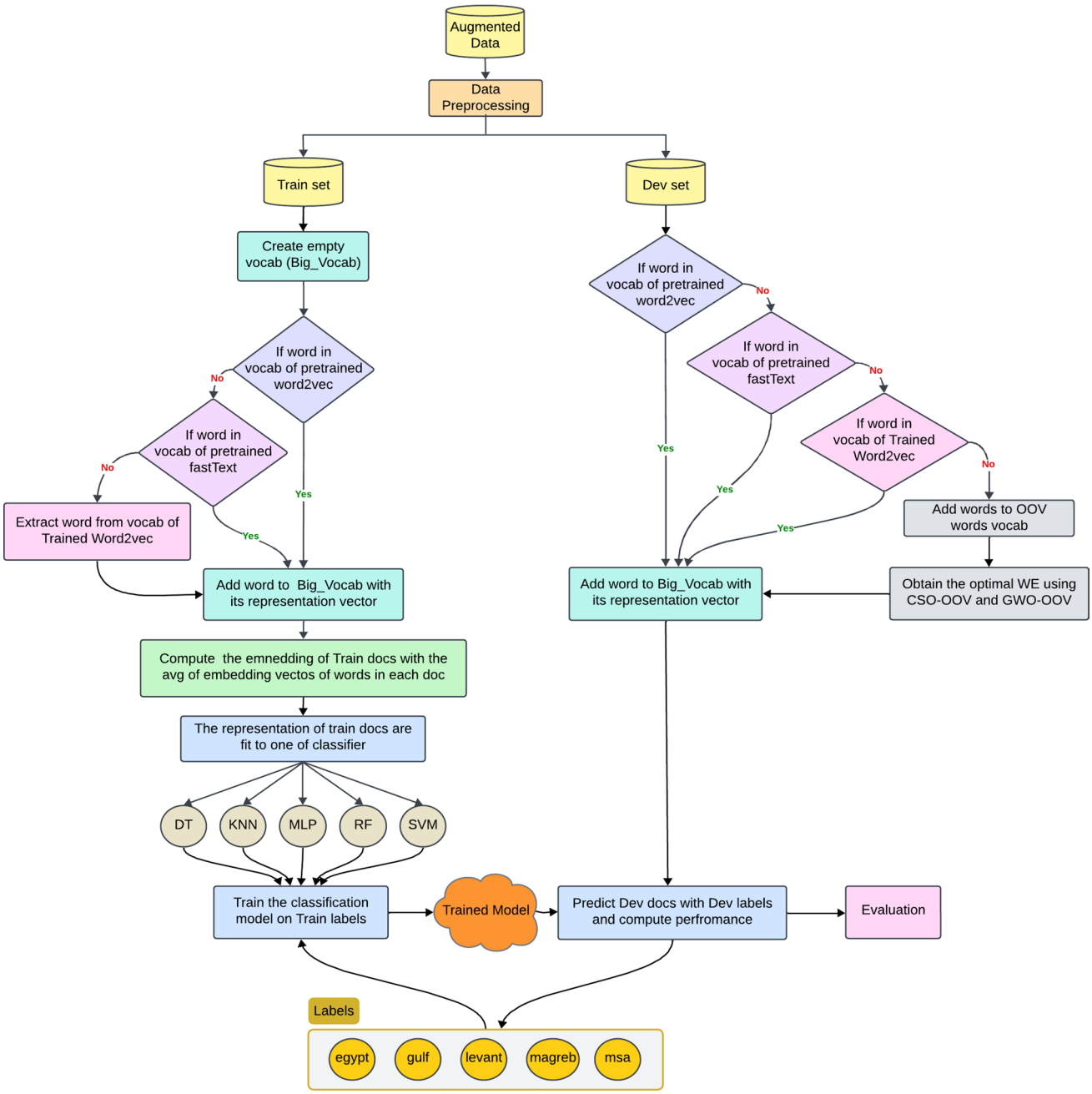


Fig. 3 Overall structure of the proposed system

Words from the training data are then checked against the vocabularies of various pre-trained models, including Word2Vec, fastText, and a specifically trained Word2Vec model which is trained on Train set. If a word exists in any of these vocabularies, it is added to BIG_VOCAB along with its corresponding representation vector. This ensures that the vocabulary includes words with established embeddings, enhancing the richness and comprehensiveness of the vocabulary.

Once the vocabulary is constructed, the embeddings of training documents are computed by averaging the embedding vectors of the words in each document. These averaged vectors represent the documents, which are then used as input features for various classifiers. The classifiers considered include K-nearest neighbors (KNN), decision trees (DT), multilayer perceptron (MLP), support vector machines (SVM), and random forest (RF). The

https://doi.org/10.1007/s00521-025-11501-1

Fig. 4 WordCloud of OOV Terms in Test set

classification models are trained on these document representations with the associated training labels. Simultaneously, the Dev set undergoes a similar process. Words in the Dev set are checked against the vocabularies of the pre-trained models.

If a word is not found, it is added to an OOV list, which is the most challenged step in this study. The word cloud in Fig. 4 of has been used to recognize the nature of the OOV terms and their impact on the model performance. This figure helped to reveal and offer insights into the OOV terms of Test set. In Fig. 4, the most prominent words are written in incorrect way, like بالنفطمصر which are two Arabic words written without space among, , where the correct words should be بالنفط مصر (means with oil, Egypt) and مغربعشاءفجر which are three Arabic words written without spaces too, where the correct words should be مغرب عشاء فجر (means Maghrib, Isha, Fajr prayers). Also, the words فعلابدليل which must written as فعلا بدليل (means actually, as evidence). These OOV terms indicate the insignificancy within the text in Train set and Test set and so this affects the performance of the model. To handle this issue in this study, the optimal word embeddings (WEs) for these OOV words were obtained using the proposed algorithms, ACO-OOV, CSO-OOV, GWO-OOV, and PSO-OOV ensuring that all words in the Dev set have representation vectors. These vectors are added to Big_Vocab, just as with the Train set.

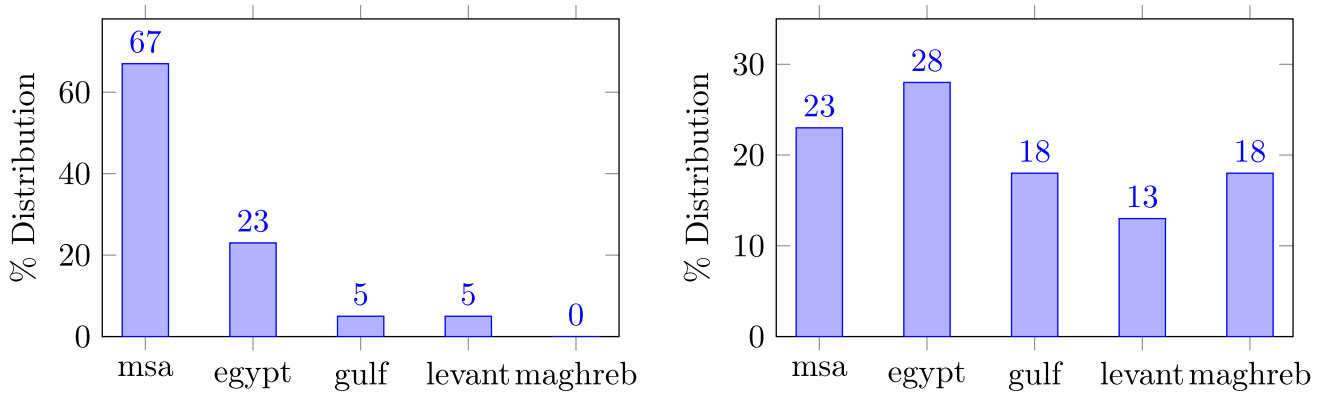
The trained model, once ready, is used to predict the labels of the documents in the Dev set. The performance of the model is then evaluated based on these predictions. This methodology ensures a robust and comprehensive approach to training NLP models, leveraging both pre-trained embeddings and optimal embeddings for OOV words, thereby enhancing the model's accuracy and effectiveness in handling diverse and complex datasets.

At the final stage, the methodology accounts for various regional dialects, Egyptian, Gulf, Levant, Maghreb, and MSA. This inclusion of dialectal variations underscores the approach's adaptability and precision in processing and analyzing Arabic language data across different regions. All steps of this methodology are explained in the following subsections.

4.1 Data augmentation

By generating new versions of training samples from existing data, data augmentation essentially allows for the artificial expansion of the training set [33]. It entails employing deep learning techniques to create new data points or slightly alter the dataset. Data augmentation serves multiple functions, including: (1) preventing overfitting; (2) handling imbalanced datasets; (3) improving model accuracy; and (4) handling insufficient datasets. Table 1 illustrates how the training data is uneven since the portions of class "maghreb" are too little and "egypt" and "msa" were significantly greater than the other classes. Several text augmentation techniques have been employed in this work, including:

- Rearranging phrases or words at random.
- Using synonyms to replace words.



(a) Before Augmentation (b) After Augmentation

Fig. 5 Bar chart of distribution of Train set before and after augmentation process

- Rewording sentences while maintaining their meanings.
- Random word additions or deletions.

Additionally, the new augmented data[6] additional datasets from the same domain have been incorporated into the original training data to enhance the overall value of the dataset. In this phase, we chose a few datasets that shared texts in the Arabic dialect with our dataset. We specifically chose records from our dataset that had the same regions. The dataset utilized in NADI 2022 is the initial dataset [9], which spans 18 dialects and focuses on sophisticated Arabic dialect recognition for Arabic tweets at the national level (about 20K tweets). DA and MSA are covered NADI 2021 dataset, which is the second one [8]. The third dataset, Habibi, is the most ancient Arabic lyrics corpus [34]. The corpus contains song lyrics in six Arabic dialects and over 30,000 songs in Arabic sung by singers from 18 Arabic-speaking nations. The lyrics consist of around 3.5 million words and over 500,000 sentences [34]. This corpus's advantage is that every word is written in DA rather than MSA. The dataset was fairly balanced after the data augmentation step, and data preprocessing was the following stage. The allocation of training data across each class, analyzed before and after the implementation of data augmentation process, is displayed in Fig. 5.

4.2 Text preprocessing

Light preprocessing has been applied to the data to prepare it for training while maintaining an precise depiction of the text that naturally appears. Latin characters, mentions, URLs, digits, emojis, and non-Arabic letters were eliminated due to the chaotic and extremely loud nature of Arabic writings, especially those on social media. Furthermore, the subsequent actions have been carried out.

- Transform the Arabic letters into their respective forms, such as ة (spoken as Haa) and ه to be ه.
- Eliminating superfluous Arabic forms like, ال (spoken as "al") and it acts as a determiner.
- Eliminating punctuation marks such as '?'; '.'; '!' to prevent the introduction of unnecessary features that could expand the dimensionality of the feature space.

[6] Our Augmented Dataset

https://doi.org/10.1007/s00521-025-11501-1

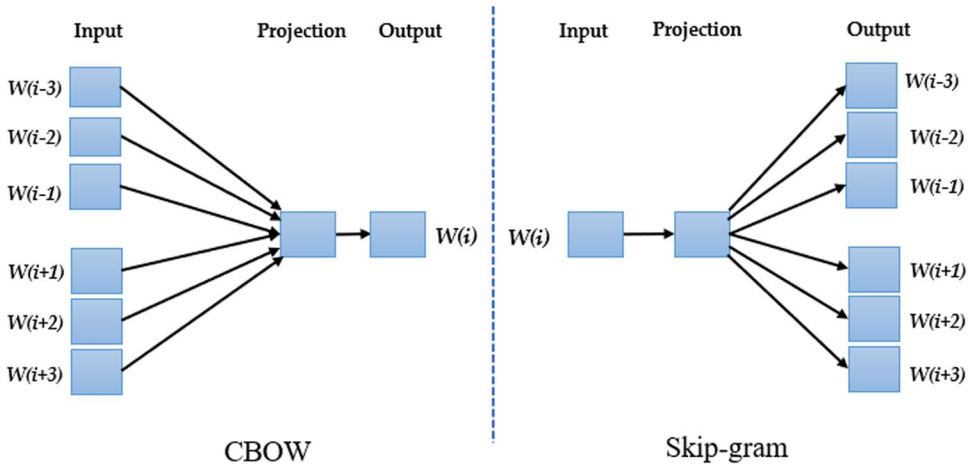


Fig. 6 Architecture of CBOW and Skip-gram models [21]

- Lowering the letter redundancy because tweets in Arabic are typically less formal. Reducing feature space can be achieved by removing the unnecessary tokens from the letters. In this study, we regarded the letter that appears three times or more as superfluous. For example, the Arabic phrase باااااارع (spoken as "Baarea") which means "brilliant" will be decreased to بارع, and also the Arabic phrase بدييييع (spoken as "Badeea") which means "awesome" will be minimized to بديع.

4.3 Feature extraction

In this step, tweets are represented as feature vectors, with individual embeddings assigned to each word within the tweet. Several techniques were applied in order to extract vectors of WE from the surrounding context words of words found. Here, two pre-trained models and a fine-tuned model were employed. In addition, two swarm intelligence algorithm, CSO and GWO, are used to handle the OOV representations.

4.3.1 Pre-trained Word2vec

The first pre-trained model is Word2vec, which is a neural network model developed by Google [35] for analyzing the input text. Word2Vec is a NN architecture-based model including three layers, one input layer, one output layer, and one hidden layer. The hidden layer operates without an activation function. Moreover, there is a match between the hidden layer neurons number and the WEs vector size. In order to accurately represent the semantic and syntactic form of the words, Word2Vec model uses enormous datasets during training [35]. This enables the efficiency measurement of word similarity. Word2Vec comes with two learning models: Skip-gram and Continuous Bag of Words (CBOW). As seen in Fig. 6, CBOW obtains the word from its surrounding context, whereas Skip-gram obtains the surrounding context words given the word itself. The two approaches share two hyperparameters, vocabulary size and window size. The amount of context words is indicated by the window size.

The CBOW technique uses a log-linear classifier to classify the projected middle word from the historical and future terms. The context has the same size of the sliding window or number of words; for simplicity, if the sliding window has four words, the context contains three words. The context of the middle word's next three words and its preceding three words must also be taken into account when predicting a term. Utilizing UNLABELED-10 M, a dataset consisting of ten million unlabeled Arabic tweets provided by NADI, formatted as tweet IDs, the first Word2Vec model was constructed. Word vectors of size 300 have been produced using gensim [36], a pre-built implementation of the Word2Vec using CBOW.

4.3.2 Pre-trained fastText

The fastText [37] is a pre-trained model, developed by Facebook's AI Research (FAIR) laboratory, that significantly enhances the processing and understanding of the Arabic language for various NLP tasks. Unlike traditional word embedding models that view words as separate entities, fastText treats each word as a collection of character n-grams. This approach allows the model to approximate the vector of a word not present in the training data using the vectors of its character n-grams, making it particularly effective at managing out-of-vocabulary words. This feature is especially valuable for Arabic because of its rich morphology, where words often have multiple forms due to the addition of prefixes, suffixes, and infixes. The fastText model adeptly captures these morphological variations, producing more precise word embeddings even for complex and rarely encountered words [37].

Trained on an extensive corpus of Arabic text that includes a wide variety of dialects and styles, the pre-trained fastText model captures the language's nuances and subtleties, ensuring its embeddings are comprehensive and contextually accurate. Consequently, the fastText model can accommodate the diverse linguistic characteristics of Arabic, ranging from MSA used in formal contexts to the various dialects spoken across different regions. A notable advantage of the fastText is that it is able to handle subword information, essential for Arabic due to the frequent use of root words and patterns. By decomposing words into less substantial units, fastText generates embeddings that reflect the internal organization of the language, providing a more detailed understanding of word meanings and relationships. The fastText pre-trained model for Arabic is a versatile and robust tool that utilizes character n-grams to create highly accurate word embeddings [37].

4.3.3 Fine-tuned Word2vec

However, not all of the words in the dataset could be covered by the embedding vectors for the terms in the pre-trained models. Thus, we developed a fine-tuned Word2Vec model. Since CBOW has a faster processing speed than Skip-gram [38], we used it to create the word vectors. The whole training dataset and a few external datasets, the Habibi corpus [34] and the NADI 2021 shared task dataset [8], were used to construct the vocabulary of this model.

Although the three WE models cover all words in Train set, the problem lies in the Dev set words as there are a lot of OOV words that are not unseen in the Big_Vocab of the three WE models. Before, the embedding vectors of OOV words were set at random vectors, so the proposed models in this study solved this issue by using two swarm intelligence algorithms, CSO and GWO.

4.3.4 Swarm intelligence for handling OOV

4.3.4.1 Chicken Swarm Optimization (CSO) Meng, X.B. et al. [39] introduced a bio-inspired metaheuristic optimization technique called chicken swarm optimization. The program emulates the individual chickens' behaviors as well as the hierarchical structure of a swarm of chickens. A chicken swarm's hierarchical structure is separated into multiple groups, each of which consists of a rooster and a large number of hens and chicks. Every variety of chickens adheres to different laws of motion. The social lives of chickens are heavily influenced by a hierarchical structure. In a flock, the dominant hens will subjugate the weaker ones. Both the most subservient hens and roosters who hang out at the group's edge and the most dominant hens who stick close to the head roosters exist. The CSO algorithm is explained as follows in [39]:

(1) There are many groups within the swarm of chickens. There is a dominant rooster in each group, tailing several hens and chicks.
(2) An assessment is made of the hens' fitness level. The roosters, who are all expected to be leaders of their groups, are the ones that are physically fittest; the chicks, conversely, are the ones who are least fit. It would be the hens among them.

(3) The swarm is made up of virtual chickens N, which are separated into the following numbers: Rn, Hn, Mn, and Cn, which stand for the number of hens, roosters, mother hens, and chicks, respectively. Each person is represented by their places in a D-dimensional space as follows:

$$X_{i,j} = (i \in [1; \vdots\vdots; N]; j \in [1; \vdots\vdots; D])$$

4.3.4.2 Movement of Roosters As shown in Eqs. (1) and (2), Roosters with higher fitness values are able to search for food in a bigger variety of locations than those with lower fitness values.

$$X_{i,j}^{t+1} = X_{i,j}^{t} \cdot (1 + \text{Randn}(0, \sigma^2)) \tag{1}$$

$$\sigma^2 = \begin{cases} 1, & \text{if } f_i \le f_k \\ \exp\left(\frac{f_k - f_i}{|f_i| + \epsilon}\right), & \text{otherwise} \end{cases} \quad k \in [1, N],\ k \ne i \tag{2}$$

where $\text{Randn}(0, \sigma^2)$ is a Gaussian distribution with mean 0 and standard deviation σ^2, ϵ is the smallest constant in the computer used to prevent zero-division, $X_{i,j}$ is the picked rooster with index i, k is the rooster's index, which is a randomly selected rooster from the group, and f_i is the matching rooster's fitness value.

4.3.4.3 Movement of Hens In order to find food, hens follow the roosters in their group. In addition, they would haphazardly pilfer tasty food that the other chickens had located; nevertheless, the other chickens would put an end to them. In a competition for food, the more assertive chickens would have an advantage over the more timid ones.

$$X_{i,j}^{t+1} = X_{i,j}^{t} + S1 \cdot \text{Rand} \cdot \left(x_{r_1,j}^{t} - x_{i,j}^{t}\right) + S2 \cdot \text{Rand} \cdot \left(x_{r_2,j}^{t} - x_{i,j}^{t}\right) \tag{3}$$

$$S1 = \exp\left(\frac{f_i - f_{r_1}}{\text{abs}(f_i) + \epsilon}\right) \tag{4}$$

$$S2 = \exp(f_{r_2} - f_i) \tag{5}$$

where Rand is a uniform random number between [0, 1], $r_1 \in [1, \ldots, \text{N}]$ is an index of a rooster, which is the i_{th} hen's group-mate, while $r_2 \in [1, \ldots, \text{N}]$ is randomly chosen index of a chicken (hen or rooster) from the swarm.

4.3.4.4 Movement of Chicks The chicks are limited to moving about their mother in order to get nourishment. This is expressed as equation 6

$$x_{i,j}^{t+1} = x_{i,j}^{t} + \text{FL} \cdot x_{m,j}^{t} - x_{i,j}^{t} \tag{6}$$

Whereas FL is a parameter that indicates the speed at which a chick would follow its mother, $\text{m} \in [1, \text{N}]$ is the position of the chick's mother. To account for variations between each chick, FL is randomly selected within the range of [0, 2].

4.4 CSO Convergence

The convergence of the CSO is facilitated by the collaborative behavior and social hierarchy within the swarm. The population is divided into roosters, hens, and chicks, each influencing movement and position updates differently. Roosters independently explore based on fitness, hens adjust positions by following roosters, and chicks follow their mother hens. This hierarchical structure guides the population toward optimal solutions. The iterative process

of position updating, fitness evaluation, and hierarchical adjustment ensures effective optimization by exploiting promising areas while maintaining diversity and preventing premature convergence.

4.4.1 Gray Wolf Optimization (GWO)

The GWO is an optimization algorithm inspired by nature, specifically designed to emulate the leadership hierarchy and hunting habit of gray wolves. Introduced by S. Mirjalili et al. [40] in 2014, GWO addresses complex optimization problems by replicating the social hierarchy and cooperative interactions observed in gray wolf packs.

Gray wolves (*Canis lupus*) are recognized for their intricate social organization and collaborative hunting techniques. Their pack hierarchy is rigorously maintained, consisting of four primary levels: alpha (α), beta (β), delta (δ), and omega (ω). The alpha wolves act as the primary leaders and decision-makers, the beta wolves support the alpha wolves, the delta wolves hold subordinate roles to both alpha and beta wolves, and the omega wolves occupy the lowest rank, often serving as the scapegoats.

Within the algorithm of GWO, the most optimal solution is designated as the alpha, the second solution is beta, while the third best solution is identified delta, and all other solutions are classified as omega. The algorithm directs the search agents toward the optimal solution by simulating the hunting strategies of gray wolves, which encompass tracking, encircling, and attacking prey.

4.4.2 Encircling Prey

Gray wolves encircle their prey during the hunt. This behavior is mathematically modeled as follows:

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot |\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)| \tag{7}$$

where

- $\vec{X}(t)$ is the position vector of a gray wolf at iteration t.
- $\vec{X}_p(t)$ is the position vector of the prey.
- $\vec{A}$ and $\vec{C}$ are coefficient vectors calculated as:

$$\vec{A} = 2 \cdot \vec{a} \cdot \vec{r}_1 - \vec{a} \tag{8}$$

$$\vec{C} = 2 \cdot \vec{r}_2 \tag{9}$$

where

- $\vec{a}$ decreases linearly from 2 to 0 over the course of iterations.
- $\vec{r}_1$ and $\vec{r}_2$ are random vectors in the range [0,1].

4.4.3 Hunting

The wolves in alpha, beta, and delta are more knowledgeable about possible prey locations. The positions of the omega wolves are updated based on the positions of these three leading wolves. This process is mathematically modeled as follows:

$$\vec{D}_\alpha = |\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}|$$
$$\vec{D}_\beta = |\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}|$$
$$\vec{D}_\delta = |\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}|$$
$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot \vec{D}_\alpha$$
$$\vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot \vec{D}_\beta$$
$$\vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \cdot \vec{D}_\delta$$

The final position update for a gray wolf is calculated as:

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \tag{10}$$

where

- $\vec{X}_\alpha$, $\vec{X}_\beta$, and $\vec{X}_\delta$ are the positions of the alpha, beta, and delta wolves, respectively.
- $\vec{A}_1$, $\vec{A}_2$, and $\vec{A}_3$ are the $\vec{A}$ coefficients calculated for the alpha, beta, and delta wolves.
- $\vec{C}_1$, $\vec{C}_2$, and $\vec{C}_3$ are the $\vec{C}$ coefficients calculated for the alpha, beta, and delta wolves.

4.5 GWO Convergence

The algorithm iterates through the above steps, gradually converging toward the optimal solution. The control parameter $\vec{a}$ plays a crucial role in balancing exploration and exploitation by reducing linearly over iterations, thereby ensuring that the wolves initially explore the search space broadly and then exploit the most promising regions.

The GWO is an effective tool for a variety of optimization issues across multiple domains because it skillfully strikes a balance between exploration and exploitation. Its popularity in the optimization field can be attributed to its efficacy, versatility, and simplicity.

4.5.1 Ant Colony Optimization (ACO)

Ant Colony Optimization (ACO) is a bio-inspired metaheuristic algorithm motivated by the foraging behavior of ants [41]. It simulates how ants locate the shortest path between their nest and a food source by depositing pheromones along traveled paths, which guide other ants.

The ACO process, as outlined in [41], proceeds as follows:

Ants are randomly positioned in the search space and incrementally construct solutions by making sequential decisions.

Movement decisions depend on pheromone trails and heuristic information, determined by the following probability formula:

$$P_{ij}(t) = \frac{\tau_{ij}(t)^\alpha \cdot \eta_{ij}^\beta}{\sum_{k \in allowed} \tau_{ik}(t)^\alpha \cdot \eta_{ik}^\beta} \tag{11}$$

where $\tau_{ij}(t)$ represents the pheromone level on edge (i, j) at time t, η_{ij} denotes the heuristic value (e.g., inverse distance), and α and β control the influence of pheromone intensity and heuristic information, respectively.

After building solutions, ants reinforce pheromone trails by depositing pheromones based on the quality of their solutions. Pheromone levels are updated using:

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{12}$$

where ρ denotes the evaporation rate, and $\Delta\tau_{ij}^{k}$ is the pheromone contribution from ant k.

4.5.2 Convergence of ACO

ACO achieves convergence by emphasizing pheromone trails along optimal paths, thereby increasing their probability of selection. Evaporation prevents premature convergence by gradually reducing the influence of suboptimal paths. This iterative framework facilitates a balance between exploration and exploitation, enabling robust search and optimization performance.

4.5.3 Particle Swarm Optimization (PSO)

Particle swarm optimization (PSO) is a stochastic, population-based optimization technique introduced by Kennedy and Eberhart [42]. It draws inspiration from the collective movement patterns observed in bird flocks and fish schools. Each particle in the algorithm represents a candidate solution within the search space, adjusting its position by leveraging its own past experience and the experiences of neighboring particles.

The PSO process, as described in [42], unfolds as follows:

Every particle is characterized by a position and a velocity. The position indicates a potential solution, while the velocity dictates the direction and magnitude of its movement.

Each particle evaluates its fitness based on an objective function and records both its personal best position and the global best position achieved by the swarm.

Positions are updated iteratively using velocity updates influenced by both personal and global best values. The governing equations are given below:

$$v_{i,j}^{t+1} = \omega \cdot v_{i,j}^{t} + c_1 \cdot r_1 \cdot (p_{i,j}^{t} - x_{i,j}^{t}) + c_2 \cdot r_2 \cdot (g_j^{t} - x_{i,j}^{t}) \tag{13}$$

$$x_{i,j}^{t+1} = x_{i,j}^{t} + v_{i,j}^{t+1} \tag{14}$$

where $v_{i,j}$ represents the velocity of particle i in dimension j, $x_{i,j}$ denotes its position, $p_{i,j}$ is its personal best, and g_j is the global best. Parameters ω, c_1, and c_2 correspond to inertia weight, cognitive coefficient, and social coefficient, respectively, while r_1 and r_2 are random values between 0 and 1.

4.5.4 Convergence of PSO

The convergence behavior of PSO is governed by balancing exploration and exploitation. The inertia weight ω manages this balance, where larger values encourage exploration and smaller values enhance exploitation. The exchange of information through personal and global best values enables effective optimization. PSO refines solutions iteratively, taking advantage of swarm intelligence to achieve optimal results.

4.5.5 Chicken Swarm Optimization for Handling OOV (CSO-OOV)

Algorithm 1 explains the steps of CSO-OOV. A little number of chosen characteristics are chosen in order to obtain the OOV representations that maximize the F1-score which represents the fitness function in CSO algorithm. Initially, the OOV vectors are randomly initialized. These vectors represents the positions of chickens in CSO. So, an intelligent search strategy is needed to identify the ideal location in the search space that optimizes the specified fitness function. Given the vocab of OOV, the CSO's fitness function is to maximize macro-averaged F1-score to increase the classification performance. For most NLP tasks, the official statistic is the macro-averaged F1-score, a fundamental aggregation metric for F1-score. The ratio of correctly classified tweets to all tweets classified is known as precision. Another parameter called recall divides all tweets that were correctly classified by the total number of tweets that were correctly classified. The macro-averaged F1-score represents the average of F1-scores calculated for each class, without giving any weight to the number of samples in each class. [43] states that the following is the macro F1-score formula:

$$F1 - score = 2 * \frac{R * P}{R + P} \tag{15}$$

4.5.6 Gray Wolf Optimizer for Handling OOV (GWO-OOV)

Similar to CSO-OOV, the goal is to obtain the OOV representations that maximize the F1-score fitness function. The OOV vectors (positions of gray wolves) are randomly initialized. The fitness function is calculated using Eq. 11. All steps of GWO-OOV algorithms are described in Algotihm 2.

4.5.7 Ant Colony Optimization for Handling OOV (ACO-OOV)

ACO-OOV aims to get the OOV that maximize the same fitness function. Here, the OOV vectors are the positions of ants that are randomly initialized. So, the fitness function is computed using equation 11. Steps of ACO-OOV algorithm are described in Algotihm 3.

4.5.8 Particla Swarm Optimization for Handling OOV (PSO-OOV)

In PSO-OOV, the goal is to get the best OOV representations to maximize the F1-score function. The vectors of OOV are the positions of particles and they are randomly initialized and the fitness function is calculated using Eq. 11. All steps of PSO-OOV algorithms are described in Algorithm 4.

4.6 Classification

In this work, five classification methods were used: DT, KNN, MLP, RF, and SVM. These classifiers' hyperparameters were used to enhance performance.

A linear classifier, the SVM makes advantage of training data at class boundaries [44]. In this work, radial basis function (RBF) kernels were utilized by the SVM model, along with other kernel functions such as sigmoid and linear, to classify nonlinear data. The K-nearest neighbors (KNN) algorithm, on the other hand, assigns a new sample to the category most similar to existing categories by assuming a similarity between the new sample and the existing ones [45]. Using DT , according to observations about the words in the branches of the tree and deductions about the goal value of the words represented in the leaves of the tree, the DT classifier [46] makes predictions. By applying multiple decision tree classifiers to different dataset sub-samples, the RF is an ensemble meta-estimator that enhances predictive accuracy and reduces overfitting by averaging predictions across multiple decision trees. [47]. A feed-forward artificial neural network (ANN) that is fully linked is the MLP [48]. Single

input layer, single hidden layer, and single output layer collectively construct three layers or less of nodes in a conventional MLP. A nonlinear activation function is used by each node, with the exception of the input nodes.

4.7 Evaluation

The ArSarcasm dataset was used to test every algorithm that was discussed in section 4.4. Accuracy (Acc), Precision (P), Recall (R), and F1-score are the evaluation metrics that have been applied [43]. The ratio of accurately classified tweets to all tweets is used to calculate accuracy. Another indicator called precision is the number of tweets accurately classified divided by the total number of tweets classified. Recall is a metric that calculates the proportion of correctly identified tweets out of the total number of relevant tweets. Since the macro-averaged F1-score is the primary aggregate metric for the F1-score, it is widely used as the standard statistic for most NLP tasks. The macro-averaged F1-scores is calculated for each class using formula in Eq. 11.

Algorithm 1 Chicken Swarm Optimization for Handling OOV (CSO-OOV)

Input the vocabulary of OOV words (OOV-vocab).
$OOV\text{-}WE$ = [] # Word Embedding of each OOV word.
Initialize matrix of positions X_i for chickens randomly ($\dim(X_i)$ = max length of $OOV\text{-}WE$))
Add $OOV\text{-}WE$ to BIG_VOCAB of all WE models.
Calculate the embedding of all tweets of Dev by averaging the WEs in each tweet.
Input the embedding vectors to classifiers and calculate the fitness (f1-score) in equation 11.
Initialize number of Roosters (Rn), Hens (Hn), Chicks (Cn), Mother-Hens (Mn) in the swarm, and G
Initialize maximum number of iterations M
while M is not reached **do**
    if T%G == 0 **then**
        Rank the fitness of all chickens and establish a hierarchical order in the swarm;
        Divide the swarm into different groups, and
        Determine the relationship between the chicks and mother hens in a group;
    end if
    for each chicken X_i in the swarm **do**
        if X_i is a rooster **then**
            Update X_i's location using equation 1;
        else if X_i is a hen **then**
            Update X_i's location using equation 3;
        else if X_i is a chick **then**
            Update X_i's location using equation 6;
        end if
        Evaluate the new solution using equation 15;
        if the new representation is better than its previous one **then**
            Update it;
        end if
    end for
end while

 https://doi.org/10.1007/s00521-025-11501-1

Algorithm 2 Gray Wolf Optimizer for Handling OOV (GWO-OOV)

Input the vocabulary of OOV words (OOV-vocab).
$OOV\text{-}WE$ = [ ] # Word Embedding of each OOV word.
Initialize matrix of positions X_i for wolves randomly ($\dim(X_i)$ = max length of $OOV\text{-}WE$)
Add $OOV\text{-}WE$ to BIG_VOCAB of all WE models.
Calculate the embedding of all tweets of Dev by averaging the WE in each tweet.
Input the embedding vectors to classifiers and calculate the fitness (f1-score) in equation 11.
Identify $\vec{X_\alpha}$, $\vec{X_\beta}$, and $\vec{X_\delta}$.
Initialize maximum number of iterations M.
while M is not reached **do**
    Calculate the control parameter $\vec{a}$.
    Calculate values of $\vec{A}$ and $\vec{C}$ using equations 8 and 9.
    for each wolf X_i in the population **do**
        Update the position of X_i using using equation 10;
        Evaluate the new solution using equation 15;
        if the new representation is better than its previous one **then**
            Update it;
        end if
    end for
    Identify new $\vec{X_\alpha}$, $\vec{X_\beta}$, and $\vec{X_\delta}$.
end while

Algorithm 3 Ant Colony Optimization for Handling OOV (ACO-OOV)

Input the vocabulary of OOV words (OOV-vocab).
$OOV\text{-}WE$ = [] # Word Embedding of each OOV word.
Initialize pheromone levels τ_{ij} and heuristic values η_{ij}.
Add $OOV\text{-}WE$ to BIG_VOCAB of all WE models.
Calculate the embedding of all tweets of Dev by averaging the WEs in each tweet.
Input the embedding vectors to classifiers and calculate the fitness (f1-score) in equation 15.
Set parameters α, β, ρ, and the number of ants.
Initialize maximum number of iterations M.
while M is not reached **do**
    for each ant in the colony **do**
        Construct a solution based on pheromone and heuristic values.
        Evaluate the new solution using equation 11.
        if the new representation is better than its previous one **then**
            Update pheromone levels using equation 12.
        end if
    end for
    Evaporate pheromones to avoid convergence on suboptimal paths.
end while

Table 1 Performance of using BIG_VOCAB of WE without swarm intelligence on ArSarcasm

Algorithm	Parameters	P (%)	R (%)	f-score (%)	Acc (%)
SVM	Linear kernel	49.07	51.46	48.25	76.73
	Sigmoid kernel	39.01	41.26	39.56	69.81
	RBF kernel	**51.81**	**52.32**	**50.26**	**77.96**
KNN		45.85	48.55	44.19	73.89
DT		32.35	38.96	32.37	58.86
RF		43.59	43.23	42.71	75.78
MLP	h=5	49.68	51.91	48.59	77.35
MLP	h=10	45.65	46.88	45.91	75.17
MLP	h=20	42.52	49.13	43.369	71.61

Algorithm 4 Particle Swarm Optimization for Handling OOV (PSO-OOV)

Input the vocabulary of OOV words (OOV-vocab).
$OOV\text{-}WE$ = [] # Word Embedding of each OOV word.
Initialize positions X_i and velocities V_i randomly for particles.
Add $OOV\text{-}WE$ to BIG_VOCAB of all WE models.
Calculate the embedding of all tweets of Dev by averaging the WEs in each tweet.
Input the embedding vectors to classifiers and calculate the fitness (f1-score) in equation 15.
Set parameters ω, c_1, c_2, and maximum iterations M.
while M is not reached **do**
    for each particle i in the swarm **do**
        Update velocity using equation 13.
        Update position using equation 14.
        Evaluate the new solution using equation 15.
        if the new representation is better than its previous one **then**
            Update personal best (p_i).
            if new fitness is better than global best (g) **then**
                Update global best (g).
            end if
        end if
    end for
end while

5 Experimental results and discussion

5.1 Arabic Dialect Identification

The proposed models were evaluated using variations of WE models. The outcomes of the proposed models using BIG_VOCAB of pre-trained models, UNLABELED-10 M and fastText-arabic, and the fine-tuned word2vec model which was trained on Train set are shown in Table 1. Table 1 provides a comprehensive overview of how different machine learning algorithms perform on a dataset using BIG_VOCAB of word embeddings without the application of swarm intelligence for handling OOV words.

In Table 1, SVM algorithm is evaluated with three different kernels, linear, sigmoid, and RBF. In contrast, the RBF kernel delivers the best results among the SVM configurations, achieving a precision of 51.81%, a recall of 52.32%, an f-score of 50.26%, and an accuracy of 77.96% and outperformed all other classifiers. The KNN algorithm exhibits a balanced performance and a moderate capability of KNN in handling the dataset. The DT algorithm shows the lowest performance among the evaluated algorithms. This suggests that DT may not be well-suited for this particular dataset or problem. Lastly, the MLP algorithm is tested with different numbers of

https://doi.org/10.1007/s00521-025-11501-1

Table 2 F1-score of all classifiers with CSO-OOV on ArSarcasm

Algorithm	F1-score (%)
SVM (Linear)	49.47
SVM (Sigmoid)	41.01
SVM (RBF)	**52.64**
KNN	44.23
DT	33.33
RF	43.24
MLP (5 H)	48.31
MLP (10 H)	44.28
MLP (20 H)	44.54

Table 3 F1-score of all classifiers with GWO-OOV on ArSarcasm

Algorithm	F1-score (%)
SVM (Linear)	50.43
SVM (Sigmoid)	41.02
SVM (RBF)	**53.34**
KNN	45.22
DT	34.23
RF	44.42
MLP (5 H)	49.63
MLP (10 H)	45.1
MLP (20 H)	45.25

hidden layers (h). With h=5, the MLP achieves a precision of 49.68%, a recall of 51.91%, an f-score of 48.59%, and an accuracy of 77.35%.

Table 2 provides a detailed comparison of all classifiers evaluated based on their macro-averaged F1-scores when using the proposed CSO-OOV method. The F1-score is a crucial metric that considers both precision and recall, offering a balanced measure of the classifier's performance. Among the classifiers, the RBF-based SVM demonstrates the best performance, achieving an F1-score of 52.64%. This indicates that the RBF kernel effectively manages the classification tasks when optimized with the CSO-OOV method, likely due to its ability to handle nonlinear relationships within the data. The SVM with a linear kernel follows, with an F1-score of 49.47%. This result shows that while the linear kernel is simpler and computationally less intensive than the RBF kernel, it still performs robustly with the CSO-OOV method. On the other hand, the SVM with a sigmoid kernel shows a lower F1-score of 41.01%, suggesting that the sigmoid kernel may not be as effective in capturing the data patterns as the other SVM kernels. The KNN classifier achieves an F1-score of 44.23%, reflecting its moderate ability to classify data points accurately using the CSO-OOV method. The KNN's performance is likely influenced by its reliance on local neighborhood information, which can be both a strength and a limitation based on the distribution of data . The DT classifier shows a comparatively lower F1-score of 33.33%. This result suggests that DT may struggle with the complexity of the data when optimized with CSO-OOV. RF performs better than the DT, with an F1-score of 43.24%. The ensemble nature of RF, which combines multiple decision trees, likely contributes to its improved performance by reducing overfitting and increasing generalization compared to a unique DT. The MLP classifier is evaluated with various numbers of hidden layers (H). The MLP with five hidden layers achieves an F1-score of 48.31%, indicating a strong performance and suggesting that a moderate network depth can capture the underlying patterns effectively. However, increasing the number of hidden layers to ten results in a slight decrease in F1-score.

Table 3 showcases the performance of various classifiers based on their F1-scores using GWO-OOV. The SVM with an RBF kernel achieves the highest F1-score of 53.34%, indicating its effectiveness in handling complex,

Table 4 F1-score of proposed systems on ArSarcasm before and after data augmentation

Algorithm	Before augmentation (%)	After Augmentation (%)
CSO-OOV-SVM	44.85	**52.64**
GWO-OOV-SVM	45.53	**53.43**

Table 5 F1-score of proposed systems and previous models on ArSarcasm

Algorithm	F1-score (%)
mBERT [10]	43.81
XLM-RB [10]	41.71
XLM-RL [10]	41.8
AraBERT [10]	47.54
MARBERT [10]	51.27
Big_Vocab_SVM	50.26
ACO-OOV-SVM	**51.30**
CSO-OOV-SVM	**52.64**
GWO-OOV-SVM	**53.43**
PSO-OOV-SVM	50.68

nonlinear relationships inside the data when optimized with the GWO-OOV method. This highlights the RBF kernel's ability to capture intricate patterns, demonstrating its superiority in this context. Following closely, the SVM with a linear kernel attains an F1-score of 50.43%, showing that even a simpler, computationally less intensive linear approach can perform robustly with the GWO-OOV method. This suggests that GWO can enhance the linear classifier performance. However, the SVM with a sigmoid kernel achieves a lower F1-score of 41.02%, indicating it may not be as effective in capturing data patterns as the other SVM kernels within the GWO-OOV framework. The KNN classifier achieves an F1-score of 45.22%, reflecting a moderate capability in accurately classifying data points using the GWO-OOV method. KNN's performance depends on local neighborhood information, which can be advantageous or limiting based on the data's distribution and characteristics. The DT classifier records a relatively lower F1-score of 34.23%, suggesting it may struggle with the data's complexity when optimized with GWO-OOV. The RF classifier, which aggregates multiple decision trees, shows better performance than a single decision tree, achieving an F1-score of 44.42%. This improvement is likely due to the ensemble nature of RF, which reduces overfitting and enhances generalization. Lastly, MLP with five hidden layers achieves an F1-score of 49.63%, indicating strong performance and suggesting that a moderate network depth can effectively capture underlying patterns in the data.

Table 4 provides a performance comparison of the proposed algorithms, CSO-OOV-SVM and GWO-OOV-SVM, before and after applying data augmentation techniques. The results shows that CSO-OOV-SVM achieved an F1-score of 44.85%, while GWO-OOV-SVM performed slightly better with a score of 45.53% and highlights that the F1-scores after data augmentation are improved, where CSO-OOV-SVM improved to 52.64%, and GWO-OOV-SVM achieved 53.43%. These results demonstrate a noticeable improvement in performance for both algorithms after the application of data augmentation. Specifically, CSO-OOV-SVM shows an increase of 7.79%, while GWO-OOV-SVM exhibits a slightly higher improvement of 7.90%. This indicates that data augmentation effectively enhanced the diversity and quality of the training data, enabling the classifiers to generalize better and handle OOV more efficiently. The table underscores the importance of data augmentation in improving model performance, particularly in scenarios where handling OOV terms is critical.

Table 5 presents a comparative analysis of the F1-scores of our proposed methods, Big_Vocab-SVM without swarm, CSO-OOV and GWO-OOV methods, against previously established models. This table highlights the effectiveness of proposed methods in handling OOV words. The mBERT model achieved an F1-score of 43.81%. mBERT, known for its multilingual capabilities, demonstrates a solid performance, yet it falls short compared with some other models in this evaluation. The XLM-R_B model records a slightly lower F1-score of 41.71%,

https://doi.org/10.1007/s00521-025-11501-1

Table 6 F1-score of proposed algorithms for Dialect Identification on ArSarcasm-V2

Algorithm	F1-score (%)
CSO-OOV-SVM	35.67
GWO-OOV-SVM	**36.04**
ACO-OOV-SVM	35.56
PSO-OOV-SVM	35.98

indicating that while it is competitive, it does not perform as well as mBERT in this specific context. Similarly, the XLM-R_L model achieved an F1-score of 41.83%, showing a marginal improvement over XLM-RB but still lagging behind other models. AraBERT, which is specifically fine-tuned for Arabic language, achieved F1-score of 47.54%. MARBERT, another model fine-tuned for Arabic, shows an even higher performance with an F1-score of 51.27%. This indicates that MARBERT is particularly well-suited for the task, benefiting from its specialized training on Arabic data.

Concerning the results proposed methods, the Big_Vocab-SVM model, which uses a large vocabulary of word embeddings without handling OOV words, achieves an F1-score of 50.26%. This result sets a benchmark for comparing the effectiveness of incorporating optimization techniques. The second section of the table highlights the proposed systems that integrate swarm intelligence algorithms with SVM to handle OOV. These optimization-based approaches consistently outperformed the pre-trained transformer models and the baseline SVM approach. Among the proposed methods, ACO-OOV-SVM, based on ant colony optimization, achieved an F1-score of 51.30%, indicating that the algorithm effectively optimizes embeddings for OOV words through pheromone-based search mechanisms. CSO-OOV-SVM performed even better with an F1-score of 52.64%, demonstrating its strength in leveraging swarm intelligence techniques to enhance classification performance. The best performing algorithm in the table is GWO-OOV-SVM, which achieved an F1-score of 53.43%. This result highlights the effectiveness of GWO in balancing exploration and exploitation to optimize OOV representations. Its superior performance suggests that GWO is particularly well-suited for handling complex feature spaces, making it the most effective algorithm for sarcasm detection in this study. On the other hand, PSO-OOV-SVM, based on PSO, achieved a score of 50.68%, slightly outperforming the baseline but lagging behind other optimization methods. While PSO showed reasonable performance, its tendency for premature convergence might have limited its effectiveness compared to GWO and CSO.

The results ensures the importance of effectively addressing OOV challenges in Arabic regional dialect identification. Although pre-trained models such as MARBERT demonstrated strong performance, the proposed optimization-based systems consistently delivered better results. The improvement observed with swarm intelligence methods can be attributed to their ability to enhance feature representations through dynamic optimization, enabling more accurate classifications. Among these approaches, GWO-OOV-SVM emerged as the most promising solution, demonstrating superior performance over all other methods.

To achieve the generalization and robustness, the proposed algorithms are tested on another Arabic dialectal dataset called ArSarcasm-V2, mentioned in section 2, and the results were affirmative, as shown in Table 6.

The results in Table 6 demonstrate that GWO-OOV-SVM achieved the highest F1-score of 36.04%, outperforming the other algorithms in this task. This indicates that GWO effectively optimized feature representations, enabling better discrimination between dialects. The superior performance of GWO can be attributed to its strong balance between exploration and exploitation, which prevents premature convergence and ensures that the model explores the feature space more effectively. Following closely, PSO-OOV-SVM achieved an F1-score of 35.98%, demonstrating competitive performance. While slightly lower than GWO, PSO effectively handled the task by leveraging particle movement strategies influenced by personal and global best solutions. Its performance suggests that PSO can efficiently optimize feature representations for dialect identification, but it may face limitations due to the possibility of premature convergence, particularly in high-dimensional feature spaces, where maintaining diversity in particle positions is critical. The CSO-OOV-SVM model scored 35.67%, reflecting a slightly lower performance and The ACO-OOV-SVM algorithm attained the lowest F1-score of 35.56% among

Table 7 F1-score of proposed systems and previous models for sentiment analysis on ArSarcasm-V2

Algorithm	F1-score (%)
CS-UM6P Team [31]	74.80
DeepBlueAI Team [31]	73.92
rematchka Team [31]	73.21
Phonemer Team [31]	72.55
IDC Team [31]	71.90
CSO-OOV-SVM	**75.66**
GWO-OOV-SVM	**75.31**

the proposed methods. While ACO is known for its ability to mimic the pheromone-based communication system of ants, which is useful for path optimization, its performance in this context suggests that it may have struggled to effectively capture complex relationships in dialect features. This could be attributed to the discrete nature of its search mechanism, which might not align well with the continuous feature representations used in this task.

5.2 Arabic sentiment analysis

For sentiment analysis, the proposed models were tested on ArSarcasm-V2 dataset. The results demonstrated competitive performance, with an F1-score that outperformed all participants in WANLP 2021 Shared Task [31] for sentiment analysis in the test phase, as shown in Table 7. The success of the proposed models in this task highlights its ability to effectively capture sentiment-specific features, including nuances in expressions, emotions, and polarities, even in the presence of out-of-vocabulary (OOV) words. This performance further validates the model's capacity to generalize across tasks that require semantic and syntactic understanding.

Table 7 provides a comparative evaluation of the performance of the proposed optimization-based approaches with the top ranked developed models submitted in WANLP 2021 Shared Task [31] for sentiment analysis on the ArSarcasm-V2. The first section of the table highlights the performance of previous models submitted by five teams. Among them, the CS-UM6P Team achieved the highest F1-score of 74.80%, followed closely by the DeepBlueAI Team with 73.92%. The other teams, including rematchka Team, Phonemer Team, and IDC Team, scored 73.21%, 72.55%, and 71.90%, respectively. These results indicate that the pre-trained models used by these teams performed well in handling sentiment analysis on this dataset.

In comparison, the proposed systems demonstrated even better performance. CSO-OOV-SVM achieved the highest F1-score of 75.66%, surpassing all previously submitted models. This significant improvement highlights the effectiveness of CSO in optimizing feature representations, particularly for handling OOV words, which often pose difficulties in Arabic sentiment analysis due to dialectal diversity and informal language usage. Similarly, GWO-OOV-SVM achieved a competitive F1-score of 75.31%, ranking just below CSO-OOV-SVM. The strong performance of GWO can be attributed to its structured approach, which balances exploration and exploitation which allows GWO to optimize feature selection effectively, improving the classifier's ability to distinguish sentiment-related patterns in text. Although slightly lower than CSO, the marginal difference suggests that both algorithms are highly effective and adaptable for sentiment analysis tasks. The improved scores of CSO-OOV-SVM and GWO-OOV-SVM suggest that swarm intelligence techniques offer significant advantages for sentiment analysis, particularly when addressing challenges related to OOV words and complex linguistic structures in Arabic text.

In conclusion, the results in this table highlight the effectiveness of the proposed CSO-OOV-SVM and GWO-OOV-SVM systems, which demonstrated superior performance over previous models for sentiment analysis. Their ability to address the complexities of sentiment classification, particularly in the presence of OOV words and dialectal variations, underscores the potential of optimization techniques in advancing NLP tasks.

Table 8 F1-score of proposed systems and previous models for sarcasm detection on ArSarcasm-V2

Algorithm	F1-score (%)
IDC Team [31]	56.77
rematchka Team [31]	56.62
UBC Team [31]	54.68
SalamBERT Team [31]	53.48
Juha Team [31]	51.91
CSO-OOV-SVM	**57.68**
GWO-OOV-SVM	**52.43**

5.3 Arabic sarcasm detection

For widening the scope of the study, the proposed models were also tested for sarcasm detection on ArSarcasm-V2 dataset. Sarcasm detection focuses on recognizing expressions where the implied meaning differs from the literal interpretation, posing significant challenges in dialectal Arabic due to its cultural and linguistic nuances. This task provides an excellent platform for assessing the flexibility and effectiveness of the proposed optimization-based approaches. The results revealed that both models achieved competitive performance, outperforming several pre-trained transformer-based models and demonstrating their ability to handle contextual complexities, as shown in Table 8.

Table 8 provides a comparative evaluation of several algorithms applied to sarcasm detection using the ArSarcasm-V2 dataset. This task is particularly challenging in Arabic, as sarcasm often relies on implicit meanings, cultural nuances, and dialectal variations, making it a robust benchmark for evaluating the effectiveness of the model.

The first section of the table presents the performance of previously developed models submitted by five teams in WANLP 2021 Shared Task [31]. There were 27 participants to this subtask. The submissions in Table 8 were ranked from 9th to 13th. the IDC Team achieved the 9th ranked F1-score of 56.77%, followed closely by the Rematchka Team with 56.62%. Other submissions, including the UBC Team and SalamBERT Team, scored 54.68% and 53.48%, respectively. The Juha Team achieved the 13th score with an F1-score of 51.91%.

In comparison, the proposed approaches yielded competitive and, in some cases, superior results. CSO-OOV-SVM for feature representation and optimization, achieved the highest F1-score in the table at 57.68%. This result surpasses all other models, demonstrating the effectiveness of swarm intelligence algorithms in addressing sarcasm detection challenges. The superior performance of CSO can be attributed to its ability to model social behavior and hierarchical dynamics, enabling it to optimize embeddings dynamically, capturing subtle patterns in sarcastic expressions that are often missed by pre-trained models.

On the other hand, GWO-OOV-SVM achieved an F1-score of 52.43%. While this score is lower than some of the pre-trained models and the CSO approach, it still highlights the potential of GWO in optimizing feature representations. The slight underperformance of GWO in this task could be attributed to its exploration-exploitation trade-off, which might not have been as effective in handling the highly nuanced and context-dependent nature of sarcasm compared to CSO.

The proposed models leverage swarm intelligence to optimize embeddings and handle OOV words, leading to improved classification performance. The higher score achieved by CSO-OOV-SVM reflects its ability to capture complex linguistic structures and adapt to the variability inherent in sarcasm. The success of the optimization-based models highlights the flexibility and scalability of swarm intelligence techniques in addressing NLP challenges that involve semantic ambiguity and contextual dependency.

6 Conclusion

In conclusion, the dynamic nature of language on social networks introduces challenges for NLP applications due to OOV. To address this, we proposed a novel OOV handling approach using swarm intelligence techniques, GWO-OOV, CSO-OOV, ACO-OOV, and PSO-OOV. The proposed models were evaluated on ADI on two datasets, demonstrated high performance, GWO-OOV integrated with SVM achieving a 53.43% F1-score. The system architecture involves creating a comprehensive vocabulary from training data, computing document embeddings, and using these embeddings as features for various classifiers. OOV words in the Dev set are optimally embedded using the proposed algorithms, ensuring robust word representation. The proposed models also were tested for sentiment analysis and sarcasm detection tasks and CSO-OOV model has achieved the highest performance compared to other models and previous work. For future work, we aim to explore the integration of advanced deep learning models and additional swarm intelligence techniques to further improve OOV handling, as well as extend our methodology to other datasets to validate its generality and robustness.

Author Contributions MS thought of the concept of the presented idea. MS and HN created the theory and carried out the computations. AH-A and MS verified the analytical methods. AAE supervised this work. All authors discussed the results and contributed to the final manuscript.

Funding No funding has been received in this research.

Data Availability The datasets used during this study are included in this article and are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The author declares that there is no conflict of interest.

Consent for publication Not applicable.

References

1. Cho K, Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Moschitti, A., Pang, B., Daelemans, W. (eds.) Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp. 1724–1734. Association for Computational Linguistics, Doha, Qatar. https://doi.org/10.3115/v1/D14-1179
2. Lochter JV, Pires PR, Bossolani C, Yamakami A, Almeida TA (2018) Evaluating the impact of corpora used to train distributed text representation models for noisy and short texts. In: 2018 International joint conference on neural networks (IJCNN), 1–8
3. Lochter JV, Silva RM, Almeida TA (2020) Deep learning models for representing out-of-vocabulary words. In: Cerri R, Prati RC (eds) Intelligent system. Springer, Cham, pp 418–434
4. Lochter JV, Silva RM, Almeida TA (2022) Multi-level out-of-vocabulary words handling approach. Knowl-Based Syst 251:108911. https://doi.org/10.1016/j.knosys.2022.108911
5. Garneau N, Leboeuf J-S, Lamontagne L (2018) Predicting and interpreting embeddings for out of vocabulary words in downstream tasks. In: Linzen, T., Chrupała, G., Alishahi, A. (eds.) Proceedings of the 2018 EMNLP workshop BlackboxNLP: analyzing and interpreting neural networks for NLP, pp. 331–333. Association for Computational Linguistics, Brussels, Belgium. https://doi.org/10.18653/v1/W18-5439
6. Bojanowski P, Grave E, Joulin A, Mikolov T (2017) Enriching word vectors with subword information. Trans Assoc Comput Linguist 5:135–146. https://doi.org/10.1162/tacl_a_00051
7. Sundermeyer M, Schlüter R, Ney H (2012) LSTM neural networks for language modeling. In Proc. Interspeech. https://doi.org/10.21437/Interspeech.2012-65
8. Abdul-Mageed M, Zhang C, Elmadany A, Bouamor H, Habash N (2021) Nadi 2021: the second nuanced arabic dialect identification shared task.

9. Abdul-Mageed M, Zhang C, Elmadany A, Bouamor H, Habash N (2022) Nadi 2022: the third nuanced arabic dialect identification shared task
10. Abdul-Mageed M, Elmadany A, Nagoudi EMB (2020) Arbert & marbert: deep bidirectional transformers for arabic
11. Boussofara-Omar N (2007) Niloofar haeri, sacred language, ordinary people: dilemmas of culture and politics in Egypt. Lang Soc 36(3):466–469. https://doi.org/10.1017/S0047404507070315
12. Aliwy A, Taher H, AboAltaheen Z (2020) Arabic dialects identification for all Arabic countries. In: Proceedings of the fifth Arabic natural language processing workshop, 302–307
13. Bassiouney R (2020) Arabic sociolinguistics: topics in diglossia, gender, identity, and politics. Georgetown University Press, Washington
14. Tilmatine M (1999) Substrat et convergences: le berbère et l' arabe nord-africain. EDNA Estudios de dialectología norteafricana y andalusí 4:99–119
15. Abdul-Mageed M, Elmadany A, Zhang C, Nagoudi EMB, Bouamor H, Habash N (2023) NADI 2023: the fourth nuanced Arabic dialect identification shared task
16. Abdul-Mageed M, Keleg A, Elmadany A, Zhang C, Hamed I, Magdy W, Bouamor H, Habash N (2024) NADI 2024: the fifth nuanced Arabic dialect identification shared task
17. Antoun W, Baly F, Hajj H (2020) Arabert: transformer-based model for Arabic language understanding
18. Talafha B, Ali M, Za'ter ME, Seelawi H, Tuffaha I, Samir M, Farhan W, Al-Natsheh HT (2020) Multi-dialect Arabic bert for country-level dialect identification
19. Gaanoun K, Benelallam I (2020) Arabic dialect identification: An arabic-bert model with data augmentation and ensembling strategy. In: Proceedings of the fifth Arabic natural language processing workshop, pp. 275–281
20. Abbassi A, Mechti S, Belguith LH, Faiz R (2017) Author profiling for arabic tweets based on n-grams. In: LPKM
21. AbuElAtta AH, Sobhy M, El-Sawy AA, Nayel H (2023) Arabic regional dialect identification (ardi) using pair of continuous bag-of-words and data augmentation. Int J Adv Comput Sci Appl 14(11) https://doi.org/10.14569/IJACSA.2023.0141125
22. Yang X, Macdonald C, Ounis I (2016) Using word embeddings in twitter election classification. Inf Retriev J 21:183–207
23. Khodak M, Saunshi N, Liang Y, Ma T, Stewart B, Arora S (2018) A la carte embedding: Cheap but effective induction of semantic feature vectors. In: Gurevych, I., Miyao, Y. (eds.) Proceedings of the 56th annual meeting of the association for computational linguistics (Volume 1: Long Papers), pp. 12–22. Association for Computational Linguistics, Melbourne, Australia. https://doi.org/10.18653/v1/P18-1002
24. Hu Z, Chen T, Chang K-W, Sun Y (2019) Few-shot representation learning for out-of-vocabulary words. In: Korhonen, A., Traum, D., Màrquez, L. (eds.) Proceedings of the 57th annual meeting of the association for computational linguistics, pp. 4102–4112. Association for Computational Linguistics, Florence, Italy. https://doi.org/10.18653/v1/P19-1402
25. Garneau N, Leboeuf J-S, Lamontagne L (2019) Contextual generation of word embeddings for out of vocabulary words in downstream tasks. In: Canadian Conference on AI
26. Zhuang L, Wayne L, Ya S, Jun Z (2021) A robustly optimized BERT pre-training approach with post-training. In: Li, S., Sun, M., Liu, Y., Wu, H., Liu, K., Che, W., He, S., Rao, G. (eds.) Proceedings of the 20th Chinese national conference on computational linguistics, pp. 1218–1227. Chinese Information Processing Society of China, Huhhot, China
27. Sanh V, Debut L, Chaumond J, Wolf T (2020) DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter
28. Abu Farha I, Magdy W (2020) From Arabic sentiment analysis to sarcasm detection: The ArSarcasm dataset. In: Al-Khalifa, H., Magdy, W., Darwish, K., Elsayed, T., Mubarak, H. (eds.) Proceedings of the 4th workshop on open-source Arabic corpora and processing tools, with a shared task on offensive language detection, pp. 32–39. European Language Resource Association, Marseille, France
29. Rosenthal S, Farra N, Nakov P (2019) Semeval-2017 task 4: sentiment analysis in twitter
30. Nabil M, Aly M, Atiya A (2015) Astd: Arabic sentiment tweets dataset. In: Proceedings of the 2015 conference on empirical methods in natural language processing, pp. 2515–2519
31. Abu Farha I, Zaghouani W, Magdy W (2021) Overview of the WANLP 2021 shared task on sarcasm and sentiment detection in Arabic. In: Habash, N., Bouamor, H., Hajj, H., Magdy, W., Zaghouani, W., Bougares, F., Tomeh, N., Abu Farha, I., Touileb, S. (eds.) Proceedings of the sixth Arabic natural language processing workshop, pp. 296–305. Association for Computational Linguistics, Kyiv, Ukraine (Virtual). https://aclanthology.org/2021.wanlp-1.36/
32. Depaolo CA, Wilkinson K (2014) Get your head into the clouds: using word clouds for analyzing qualitative assessment data. TechTrends 58(3):38–44 (**Copyright - Association for Educational Communications and Technology 2014**)
33. Shorten C, Khoshgoftaar TM, Furht B (2021) Text data augmentation for deep learning. J Big Data 8:1–34
34. El-Haj M (2020) Habibi-a multi dialect multi national Arabic song lyrics corpus
35. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space
36. Rehurek R, Sojka P (2011) Gensim – statistical semantics in python

37. Bojanowski P, Grave E, Joulin A, Mikolov T (2017) Enriching word vectors with subword information. Trans Assoc Comput Linguist 5:135–146. https://doi.org/10.1162/tacl_a_00051
38. Suleiman D, Awajan A (2018) Comparative study of word embeddings models and their usage in arabic language applications. In: 2018 International Arab Conference on Information Technology (ACIT), pp. 1–7. IEEE
39. Meng X, Liu Y, Gao X, Zhang H (2014) A new bio-inspired algorithm: Chicken swarm optimization. In: Tan Y, Shi Y, Coello CAC (eds) Adv Swarm Intell. Springer, Cham, pp 86–94
40. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007
41. Salami NA (2009) Ant colony optimization algorithm. https://api.semanticscholar.org/CorpusID:11754556
42. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of ICNN'95 - international conference on neural networks, 4:1942–19484. https://doi.org/10.1109/ICNN.1995.488968
43. Dalianis H, Dalianis H (2018) Evaluation metrics and evaluation. Clinical Text Mining: secondary use of electronic patient records, 45–53
44. Sarkar A, Chatterjee S, Das W, Datta D (2015) Text classification using support vector machine. Int J Eng Sci Invent 4(11):33–37
45. Jiang S, Pang G, Wu M, Kuang L (2012) An improved k-nearest-neighbor algorithm for text categorization. Expert Syst Appl 39(1):1503–1509
46. Harrag F, El-Qawasmeh E, Pichappan P (2009) Improving arabic text categorization using decision trees. In: 2009 First international conference on networked digital technologies, pp. 110–115. IEEE
47. Jalal N, Mehmood A, Choi GS, Ashraf I (2022) A novel improved random forest for text classification using feature ranking and optimal number of trees. J King Saud Univ-Comput Inf Sci 34(6):2733–2742
48. Alla H, Moumoun L, Balouki Y (2021) A multilayer perceptron neural network with selective-data training for flight arrival delay prediction. Sci Progr 2021:1–12

Authors and Affiliations

Mahmoud Sobhy[1] · Ahmed H. AbuElAtta[1] · Ahmed A. El-Sawy[1,2] · Hamada Nayel[1,3]

✉ Mahmoud Sobhy
mahmoud.hassan@fci.bu.edu.eg

Ahmed H. AbuElAtta
ahmed.aboalatah@fci.bu.edu.eg

Ahmed A. El-Sawy
ahmed.el_sawy@fci.bu.edu.eg

Hamada Nayel
hamada.ali@fci.bu.edu.eg

[1] Department of Computer Science, Faculty of Computers and Artificial Intelligence, Benha University, Benha, Egypt

[2] Information Technology Department, Faculty of Technological Industry and Energy, Delta Technological University, Mansoura, Egypt

[3] Department of Computer Engineering and Information, College of Engineering, Wadi Ad Dwaser, Prince Sattam Bin Abdulaziz University, Al-Kharj 16273, Saudi Arabia