Hindawi
Scientific Programming
Volume 2022, Article ID 6368709, 12 pages
https://doi.org/10.1155/2022/6368709

Research Article

Persian Named Entity Recognition by Gray Wolf Optimization Algorithm

Aynaz Forouzandeh, Mohammad-Reza Feizi-Derakhshi, and Pejman Gholami-Dastgerdi

Computerized Intelligence Systems Laboratory, Department of Computer Engineering, University of Tabriz, Tabriz, Iran

Correspondence should be addressed to Mohammad-Reza Feizi-Derakhshi; mfeizi@tabrizu.ac.ir

Received 31 May 2022; Revised 20 September 2022; Accepted 3 November 2022; Published 10 December 2022

Academic Editor: Dongpo Xu

Named entity recognition (NER) is a subfield of natural language processing (NLP). It is able to identify proper nouns, such as person names, locations, and organizations, and has been widely used in various tasks. NER can be practical in extracting information from social media data. However, the unstructured and noisy nature of social media (such as grammatical errors and typos) causes new challenges for NER, especially for low-resource languages such as Persian, and existing NER methods mainly focus on formal texts and English social media. To overcome this challenge, we consider Persian NER as an optimization problem and use the binary Gray Wolf Optimization (GWO) algorithm to segment posts into small possible phrases of named entities. Later, named entities are recognized based on their score. Also, we prove that even human opinion can differ in the NER task and compare our method with other systems with the Sep_TD_Tel01 dataset and the results show that our proposed system obtains a higher *F1* score in comparison with other methods.

1. Introduction

Named entity recognition (NER) is commonly accepted as one of the core tasks in natural language processing (NLP). The key purpose of NER is to identify and classify certain proper nouns, such as person names, locations, and organizations [1] which can be used in numerous text processing applications, such as information extraction, machine translation, information retrieval, and text summarization.

Moreover, with the pervasiveness of social networks and social media such as Telegram and Instagram, a very large pile of information is being produced and published daily that carries useful data. This has led to fundamental changes in so many tasks carried out by service providers, business owners, factories, brands, or even governments. Thus, NER can be practical in extracting information from unstructured texts such as emails, social media, and text messages [2]. Nevertheless, social media NER is challenging due to its unstructured and noisy short texts which can contain spelling inconsistencies, improper grammatical structures, and numerous informal abbreviations. Also, in Persian, unlike in English, fewer kinds of research have been conducted due to the lack of corpus and not being widespread of this language which poses another challenge. Therefore, previous methods may be very effective in their language but not applicable in the Persian language due to differences in the structure of languages.

In this work, to overcome the above-mentioned challenges, we use metaheuristic optimization algorithms which have become popular in solving many problems. We view NER as a binary optimization problem to partition Telegram posts into small possible phrases of named entities which we call them segments. To be specific, it can be a possible segmentation boundary between each pair of words which is defined by a binary value. Then, we apply the binary Gray Wolf Optimization (GWO) algorithm to obtain optimum values. Finally, we rank these segments to recognize segments that are true named entities. Our main contributions are as follows:

(i) We propose a NER system for the Persian language that is based on segmentation using the GWO algorithm. Background knowledge extracted from two corpora (Persian Wikipedia and *N*-Gram) is used as global contexts in the segmentation process.

(ii) We developed *N*-Gram from the Hamshahri corpus and Telegram large dataset

(iii) We compare our method with three other methods on the Sep_TD_Tel01 dataset

In Section 2, we review some related works for NER and its different categories of proposed methods. Section 3 describes our method. In Section 4, we talk about the evaluation method, dataset, parameter setting, and evaluation results. Finally, Section 5 presents some concluding notes.

2. Related Work

This paper describes NER systems based on the GWO algorithm for Persian data. Thus, in this section, some related works on NER will be discussed.

In the past few years, NER has been researched widely. However, due to the lack of a Persian data corpus, fewer researches have been carried out in this language. In the literature, different methods of NER have been proposed, such as machine learning methods, linguistic rule-based methods, statistical methods, and hybrid methods.

Early NER research studies were mostly allocated to rule-based systems. In linguistic rule-based systems, named entities are recognized based on structural and grammatical rules. However, rule-based systems suffer from poor generalization as including all linguistic rules is practically impossible [2]. Riaz in [3] proposes a hand-crafted rule-based Urdu NER algorithm. Khormuji and Bazrafkan in [1] present an approach based on local filters for Persian NER. This approach uses multiple dictionaries, which are available on the web. References [4–6] are some other rule-based NER systems that have been carried out in different languages.

To date, various machine learning methods have been used to recognize named entities. The most common examples are hidden Markov models (HMMs), decision trees, maximum entropy, support vector machines (SVMs), conditional random fields (CRFs), and bidirectional LSTM [7, 8]. All these methods are based on supervised learning, which is the most common learning method in NER. There have also been semisupervised and unsupervised systems, but the results of these systems are mostly worse [9]. Konkol et al. in [9] use an unsupervised machine learning method based on semantic features, which can be used in a variety of languages. Chiu and Nichols in [8] extract named entities using bidirectional long short-term memory (BiLSTM) and convolutional neural network (CNN) architecture on an English dataset. In this model, word- and character-level features are extracted with a hybrid BiLSTM and CNN architecture. Furthermore, as lexicon is important in NER, a new lexicon encoding scheme and the matching algorithm have been proposed, making use of partial matches. Poostchi et al. in [7] propose a deep learning method based on BiLSTM-CRF for Persian NER, which is a combination of BiLSTM and a CRF. The BiLSTM processes each sentence token-by-token and produces an intermediate representation. Then, CRF uses this representation as input to predict all tokens' labels. Before the BiLSTM process, embedding methods, which is part of the training stage, have been used to convert each token into a high-dimensional numerical vector. As well, Momtazi and Totrabi in [10] present NER for Persian text in which for entity recognition BiLSTM is used, and for the word representation, word2vec, and fastText vector representation are used and compared. Taher et al. in [11] proposed Beheshti-NER which utilizes BERT for Persian NER. In this method, they have trained CRF on the top of pretrained bidirectional transformer BERT. Also, Asgari-Chenaghlu et al. in [12] propose two approaches for NER by deep learning and transformers. In both approaches, image features have been used. In [13], Dogan et al. use deep learning models (ELMos) and an extensive knowledge base for fine-grained NER. Zali and Firoozbakht in [14] present an approach for Persian NER based on neural networks benefiting from the vector representation of words. Tafreshi and Soltanzadeh in [15] implement Persian NER based on CRF in which various linguistic features have been designed to extract appropriate features for the learning phase. Farahani et al. in [16] present ParsBERT which is a transformer-based model for Persian and evaluate the model on sentiment analysis, text classification, and NER task.

Moreover, there have been some research studies that have proposed statistical systems. These studies are generally focused on language-independent machine-learning techniques that aim to learn statistical models from data [17]. Bikel et al. in [18] have built a NER system which is called "Nymble." This system uses a slightly modified HMM in which observations are the words, and hidden states represent the labels. Nymble has been implanted in English and Spanish. Chenliang et al. in [19] propose a novel 2-step unsupervised NER system called "TwiNER" on which our proposed system is based. TwiNER focuses on a targeted Twitter stream that is used to collect users' opinions about organizations. This system uses two global contexts, Microsoft Web *N*-Gram and Wikipedia, to partition tweets into valid segments. This process has been carried out with a dynamic programming algorithm. Then, named entities are extracted using both global and local contexts.

Furthermore, some studies make use of hybrid methods. In hybrid approaches, ruled-based and machine-learning methods are integrated. Moradi et al. in [20] present a hybrid system for Persian NER. Because of HMM innate weaknesses, they combine it with ruled-based methods. This system uses HMM and Viterbi algorithms in its machine learning section and lexical resources and pattern bases in its rule-based section. Dashtipour et al. in [2] propose a hybrid Persian NER by combining dictionaries of Persian NER, grammar rules, and SVM. Persian-named entity dictionaries of three entities (name, location, and date) were collected manually in this system. Then, different grammar rules were added. Finally, a classifier was trained with bigram features. Kung et al. [21] present a Hybrid Mandarin NER using transfer learning for disaster management. This approach, which has three modules, combines established rules and learned sentence patterns.

3. Model Description

In this paper, named entities of the Telegram posts have been extracted by the TwiNER [19] method and gray wolf optimization [22]. As these kinds of posts are considered informal, the process of extracting named entities in this context is challenging, and the common methods are not useful. Therefore, in this paper, a segmentation-based system is presented. The proposed system architecture can be divided into two steps: 1-segmentation and 2-named entity extraction. The architecture of the proposed system is shown in Figure 1.

In the first step, the posts are segmented using a stickiness function. Two global contexts, Wikipedia and *N*-Gram, were used to calculate the stickiness. In fact, in the segmentation phase, each Telegram post is broken down into smaller terms, each of which is a candidate for named entities in the next step. Nevertheless, due to the fact that the state space for the segmentation of each post will increase exponentially, an optimization algorithm is needed to find the optimum state. In this paper, the binary GWO [23] is used to find the optimal segmentation. This step is fully explained in Section 3.1.

In the second step, after segmenting the posts, these segments must be scored, considering that not all of the segments are true named entities. To do this, a graph is created whose nodes will be the extracted segments, and the weight of its edge will be calculated using the Jaccard. The Random walk model then scores graph nodes using Wikipedia's linguistic knowledge. The outputs of this section are scored segments. Finally, high-scored segments are considered as named entities. The full description of this step is given in Section 3.2.

3.1. Segmentation

3.1. Segmentation. In the first step, posts need to be segmented based on a stickiness criterion. In this paper, segmentation means the partition of a post into smaller phrases. However, the number of segmentation states grows exponentially with respect to the length of the post, and examining all these states is impossible. If we assume that a sentence has m words, then we will have $2^{(m-1)}$ state for segmenting. For instance, in order to segment the post, which can be seen in Figure 2 has 11 tokens, will have $2^{10} = 1,024$ states. Therefore, an optimization algorithm is needed to do the segmentation. This paper uses GWO for segmentation. In Figure 2, some examples of possible segments are represented. According to the results, the second segmentation has the better output since it considers the named entity of "Ayatollah Hashemi Rafsanjani" as a segment.

3.1.1. GWO in Segmentation. As mentioned in Section 3.1, search space size exceeds exponentially. Hence, because of the huge search space, attribute reduction is a challenging task. Various search techniques have been employed to solve the attribute reduction problem, for instance, greedy search based on sequential forward selection and sequential backward selection. Nonetheless, these approaches suffer from different issues, such as increasing computational cost and stagnation in local optima [24]. Metaheuristic optimization algorithms have become more popular in various applications due to their simplicity in implementation, not requiring gradient information and bypassing local optima [25]. Metaheuristics are fundamentally based on trajectory such as simulated annealing (SA) or based on population, such as genetic algorithm (GA), ant colony algorithm (ACO), particle swarm optimization (PSO), and GWO [26].

GWO is a recently developed algorithm introduced by Mirjalili et al., which mimics the leadership hierarchy (alpha, beta, omega, and delta) and haunting mechanism of gray wolfs. In this algorithm, alpha is considered the best solution. Beta and omega are, respectively, the second and third-best solutions. For more information about GWO, please refer to [22]. This algorithm is based on swarm intelligence (SI). SI algorithms have fewer parameters and are easy to implement, and GWO is one of the rapidly progressing SI algorithms drawing researchers' attention [26].

This section describes how to use the GWO to segment posts. As mentioned, GWO is based on SI, which processes the solution of the problem in vector form. In this paper, each solution is called a gray wolf vector. The segmentation problem is defined as a binary problem; nevertheless, GWO is not able to produce binary vectors. Consequently, the values of this algorithm need to be in binary form. In order to do this, the method described in [23] has been utilized in which two approaches are proposed. This article uses the second approach. In this approach, only the updated values of the gray wolf vector are converted to binary values. To be more specific, if the sigmoid value of the gray wolf vector exceeds the threshold (the threshold is generated randomly), it returns 1, and otherwise 0.

To start optimization, the dimension of gray wolf vectors must be determined. If a post has n words, then the dimension of the gray wolf vector will be $n-1$. Because to segment, each post, between each word can be considered as a segmentation boundary. Segment boundary is the boundary of a word or some selected words as a segment in the segmentation phase. For instance, in the post "Urgent: Ayatollah Hashemi Rafsanjani was hospitalized." the dimension of gray wolf vectors is equal to 7.

Each gray wolf vector can be generated by three values (0, 1, and −1). A value of 0 means that this point is not considered as a segmentation boundary (pink cells in Figure 3), and values 1 and −1 indicate the segmentation boundary (green and blue cells in Figure 3). The difference between 1 and −1 is that −1 has been used for definite segmentation boundaries, so for each post, it remains constant in the process of optimization. Definite segmentation boundaries are stop words, punctuations, and emojis, which are not used as named entity or part of named entity in the Persian language. The points that are before and after these tokens are marked with −1 in the gray wolf vector (green cells in Figure 3). These points, to some extent, prevent the overproduction of segments that are not named entities. Therefore, it will reduce the number of states. According to this explanation, our goal is to find the optimum value for nondefinite boundaries. Algorithm 1 outlines the segment extraction algorithm.

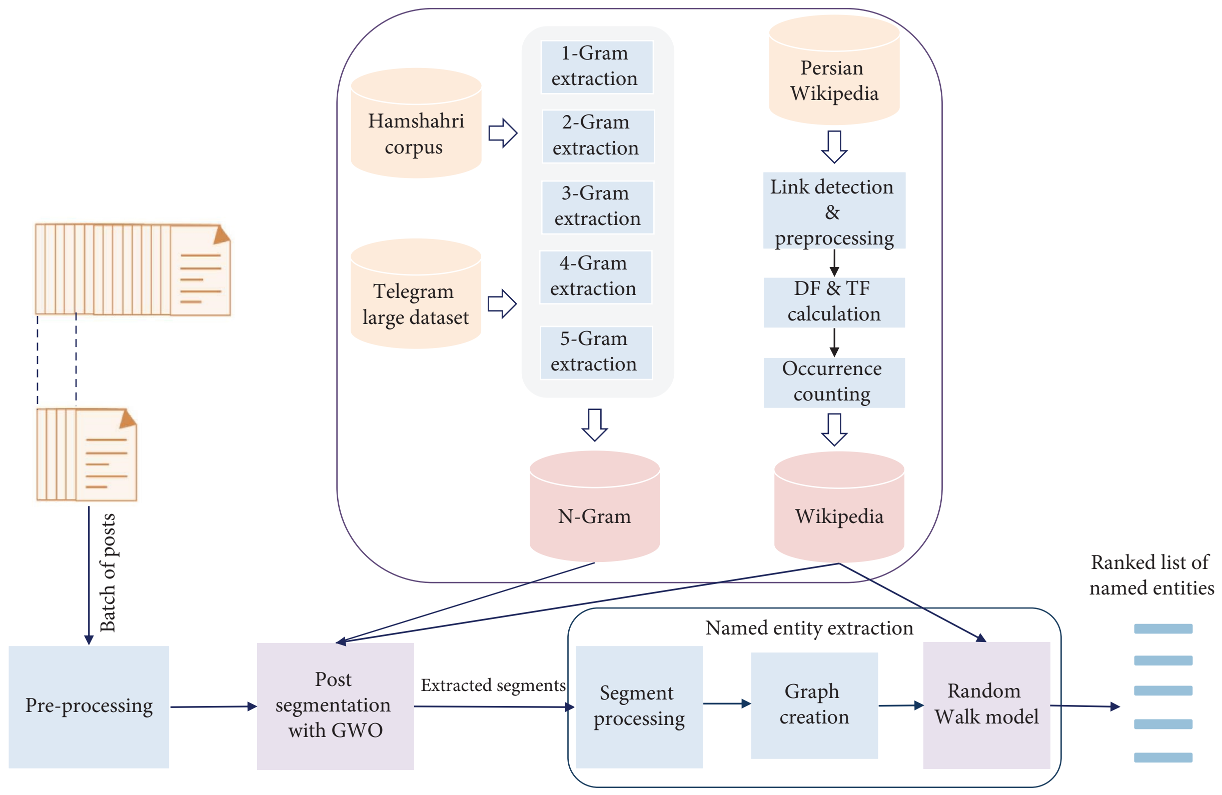
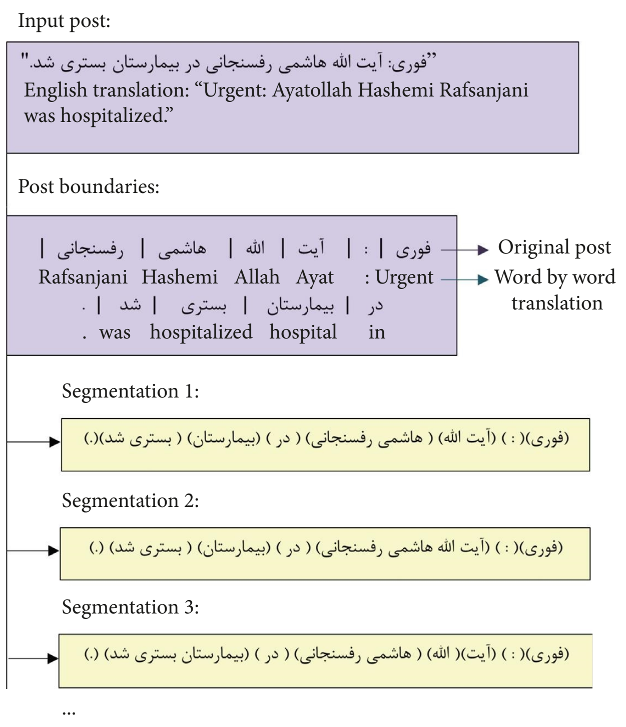


Figure 1: Proposed system architecture.



Input post:
"فوری: آیت الله هاشمی رفسنجانی در بیمارستان بستری شد."
English translation: "Urgent: Ayatollah Hashemi Rafsanjani was hospitalized."
Post boundaries:
فوری | : | آیت | الله | هاشمی | رفسنجانی |
Urgent : Ayat Allah Hashemi Rafsanjani
در | بیمارستان | بستری | شد | .
in hospital hospitalized was .
Original post
Word by word translation
Segmentation 1:
(فوری)(:) (آیت الله) (هاشمی رفسنجانی) (در) (بیمارستان) (بستری شد)(.)
Segmentation 2:
(فوری)(:) (آیت الله هاشمی رفسنجانی) (در) (بیمارستان) (بستری شد) (.)
Segmentation 3:
(فوری)(:) (آیت)(الله) (هاشمی رفسنجانی) (در) (بیمارستان بستری شد) (.)
...

Figure 2: Example of a post and its boundaries and some possible segmentation of this post.

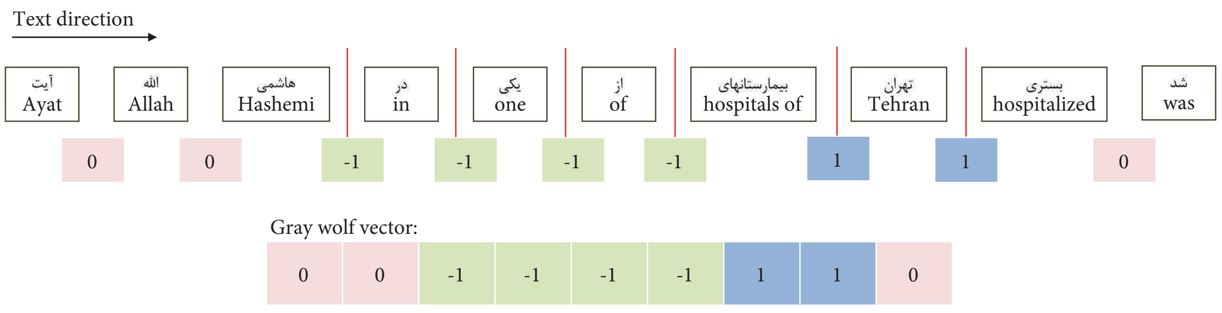


FIGURE 3: An example of a gray wolf vector. The pink cells with the value of 0 are not segmentation boundaries, the value of 1 in the blue cells indicates segmentation boundaries, and the green cells are definite segmentation boundaries for stop words, punctuations, and emojis.

```
input: p: A Telegram post $p = w_1 w_2 \ldots w_n$
where $w_i$ is a word in the post
GWV: gray wolf vector
output: S: set of segments for a post extracted from GWV
(1) NewSegment ⟵ True
(2) DetectSegment ⟵ False
(3) $S \longleftarrow []$
(4) for $i = 1{:}len(GWV)$ do
(5) if $GWV[i]\ != -1$ then
(6) if NewSegment or $GWV[i-1] == -1$ then
(7) start ⟵ $i$
(8) NewSegment ⟵ False
(9) if $GWV[i] == 1$ or $i == len(GWV)$ then
(10) end ⟵ $i$
(11) NewSegment ⟵ True
(12) DetectSegment ⟵ True
(13) else if $GWV[i+1] == -1$ then
(14) $i \longleftarrow i + 1$
(15) end ⟵ $i$
(16) NewSegment ⟵ True
(17) DetectSegment ⟵ True
(18) else
(19) NewSegment ⟵ False
(20) DetectSegment ⟵ False
(21) if DetectSegment then
(22) currentSegment ⟵ $w_{\text{start}} \ldots w_{\text{end}}$
(23) add currentSegment to $S$
(24) DetectSegment ⟵ False
(25) NewSegment ⟵ True
(26) return $S$
```

ALGORITHM 1: Segment extraction.

As mentioned, a stickiness criterion is required in segmentation. As a result, a fitness function must be defined to determine the optimal value. The following section describes this function in detail.

3.1.2. Fitness Function.

3.1.2. Fitness Function. In the previous section, the process of creating gray wolf vectors and their values were described. This section describes how to calculate fitness function.

In this step, a criterion that determines the amount of stickiness between the words is used to assess the optimal segmentation. The high amount of stickiness means that the cut-off point between two words should not be equal to 1 because it separates the correct expression. Thus, for each post $p \epsilon P_i$, the goal is to divide p into m optimal segment $p = s_1, s_2, \ldots, s_m$. Therefore, the following fitness function is used to obtain an optimal state:

$$F(p) = \max \sum_{i=1}^{m} C(s_i).e^{\text{wiki}(s_i)}.l(s_i). \quad (1)$$

This equation consists of three parameters. C is a function that measures the stickiness of the words, $e^{\text{wiki}(s)}$ calculates the Wikipedia score, and $l(s)$ normalizes the length of the segments. In the rest of this section, each parameter will be described in more detail.

As stated in [19], two stickiness functions point mutual information (PMI) and symmetric conditional probability(SCP) are used to calculate C.

(1). PMI-Based Stickiness. PMI measures the probability of two words occurring together more often than by chance. PMI for bigram w_1w_2 is defined as follows:

$$\mathrm{PMI}(w_1w_2) = \log\frac{\Pr(w_1|w_2)}{\Pr(w_1)} = \log\frac{\Pr(w_1w_2)}{\Pr(w_1)\Pr(w_2)}. \tag{2}$$

For a segments $s = w_1 \ldots w_n$, PMI is extended by averaging all binary partitions as follows:

$$\mathrm{PMI}(s) = \log\frac{\Pr(s)}{1/n-1\sum_{i=1}^{n-1}\Pr(w_1\ldots w_i)\Pr(w_{i+1}\ldots w_n)}, \tag{3}$$

where $\Pr(s)$ represents the N-Gram probability of the s segment. If the segment has only one word, we have $\mathrm{PMI}(s) = \log\Pr(w)$. Finally, in order to map PMI in the range (0, 1), the stickiness of segment s is as follows:

$$C(s) = \frac{1}{1+e^{-\mathrm{PMI}(s)}}. \tag{4}$$

(2). SCP-Based Stickiness. SCP measures the cohesiveness of bigram w_1w_2 by considering both conditional probabilities for the bigram as follows:

$$\begin{aligned}\mathrm{SCP}(w_1w_2) &= \Pr(w_1w_2|w_1)\Pr(w_1w_2|w_2)\\ &= \log\frac{\Pr(w_1w_2)^2}{\Pr(w_1)\Pr(w_2)}.\end{aligned} \tag{5}$$

For a segment with n words, SCP is as follows:

$$\mathrm{SCP}(s) = \frac{\Pr(s)^2}{1/n-1\sum_{i=1}^{n-1}\Pr(w_1\ldots w_i)\Pr(w_{i+1}\ldots w_n)}. \tag{6}$$

To smooth SCP value logarithm calculation is used as follows:

$$\mathrm{SCP}(s) = \log\frac{\Pr(s)^2}{1/n-1\sum_{i=1}^{n-1}\Pr(w_1\ldots w_i)\Pr(w_{i+1}\ldots w_n)}. \tag{7}$$

Similarly, for a single word segment, we have $\mathrm{SCP}(s) = 2\log\Pr(w)$. Eventually, the sigmoid function is applied as follows:

$$C(s) = \frac{2}{1+e^{-\mathrm{SCP}(s)}}. \tag{8}$$

According to and equations (3) and (7), a big corpus is necessary to calculate $\Pr(s)$ more accurately. In this paper, Hamshahri corpus and Telegram data have been used to calculate N-Gram.

Obtained $\Pr(s)$ from Hamshahri corpus and Telegram data is only based on frequent patterns. Therefore, another parameter is required that increases the probability of partitioning a post into segments which are named entities. To do this, Wikipedia has been used in a global context. Generally, in Wikipedia, links title are named entities, and with the emergence of new named entities, a web page in Wikipedia is dedicated to them, from which a large dictionary is produced. This dictionary can be used in segmentation. Hence, if $\mathrm{Wiki}(s)$ be the probability of occurring segment s in Wikipedia links title, the stickiness function will be as follows:

$$C'(s) = C(s).e^{\mathrm{wiki}(s)}, \tag{9}$$

$$\mathrm{wiki}(s) = \frac{N\_\mathrm{anchor}(s)}{N\_\mathrm{total}(s)}, \tag{10}$$

where $N\_\mathrm{anchor}(s)$ is the occurrence number of segment s in Wikipedia links, and $N\_\mathrm{total}(s)$ indicates the total occurrence number of segment s.

So far, segments of different lengths have been treated the same. However, observations show that segments with longer lengths are more likely to be named entities. Therefore, the length of the segments must be normalized. To do this, $l(s)$ is added to equation (9).

$$C'(s) = C(s).e^{\mathrm{wiki}(s)}.l(s), \tag{11}$$

where $l(s)$ is defined as follows:

$$f(x) = \begin{cases}\dfrac{|s|-1}{|s|}, & \mathrm{if}|s| > 1,\\[2ex] \dfrac{1}{3}, & \mathrm{if}|s| = 1.\end{cases} \tag{12}$$

On the other hand, since, in most cases, the length of named entities does not exceed some amount, a threshold value (u) should be considered for the maximum number of words for each segment, which in this paper is 5. If the number of words exceeds this threshold, the fitness function will get a penalty which is 15 in this paper. Thus, the optimization algorithm eliminates vectors that have segments of more than 5 words. In fact, it prevents the existence of more than 4 zeroes consecutively. Algorithm 2 shows this process where it gets extracted segments from Algorithm 1 as input and calculates fitness.

Also, to increase the speed of the system, when more than two −1 occur in a row (boundaries of stop words, punctuations, and emojis), the cells in between are not included in the fitness function because they add a constant value to fitness function and do not affect the result.

3.2. Named Entity Extraction.

3.2. Named Entity Extraction. At this stage, after the segmentation was performed, the segments are scored to identify named entities because not all the segments obtained from the previous stage are named entities. At first, before scoring the segments, those segments that are stop words, punctuations, hashtags, and emojis have been removed. Furthermore, some of the conversational words and abbreviations also have been removed from the segments.

In Telegram posts, named entities related to a specific topic will likely occur together. For example, “Trump” and “America” are more likely to happen together, while

```
    input: S: set of segments for a postextracted from GWV
    u: The maximum length of a segment
    penalty: penalty for the segments with more than u words
    output: Fitness for each vector
(1) fitness ⟵ 0
(2) foreach segment in S do
(3) if len(segment) > u then
(4) fitness ⟵ fitness − penalty
(5) else
(6) segmentFitness ⟵ calculate fitness for the segment using equation (1)
(7) fitness ⟵ fitness + segmentFitness
(8) return fitness
```

Algorithm 2: Fitness calculation.

"Trump" and "message of condolence" are less likely to happen together. Consequently, this local context is used to form a graph of segments, in which the nodes are the segments extracted from a set of telegram posts, and the Jaccard criterion obtains the weight of its edges. If e_{ab} is the edge passing through s_a to s_b, then we will have the following:

$$w(e_{ab}) = \frac{|M(s_a) \cap M(s_b)|}{|M(s_a) \cup M(s_b)|}. \quad (13)$$

$M(s)$ is the set of posts in P_i that contains segment s. In this way, local context is obtained, which is based on the connection between segments.

Then, the random walk model is applied to the graph. Assuming that the graph is bidirectional, the probability of transiting from node a to node b is as follows:

$$P_{ab} = \frac{w(e_{ab})}{\sum_{c\epsilon V} w(e_{ab})}. \quad (14)$$

In this section, the title of Wikipedia links is used as local context. Therefore, the transition of e_s for segment s is defined as follows:

$$e_s = \frac{\text{wiki}'(s)}{\sum_{s_j \epsilon V} \text{wiki}'(s_j)}, \text{where } \text{wiki}'(s) = e^{\text{wiki}(s)}. \quad (15)$$

Then, the stationary eigenvector π^T of P is calculated iteratively as follows:

$$\pi = \left(\gamma P^T + (1-\gamma)e1^T\right)\pi, \quad (16)$$

where γ controls the probability of teleportation, finally, to create a balance between local and global context, equation (17) has been used.

$$y(s) = e_s.\pi(s). \quad (17)$$

Equation (17) is used to score segments, and finally, top k segments are selected as named entities.

4. Evaluation

To evaluate the performance of our proposed system, we have used the NER evaluation method introduced in [27]. This method measures precision, recall, and *F1*score based on Message Understanding Conference (MUC) [28] metrics. These metrics, which consider different categories of errors, are listed as follows:

(i) Correct (COR). The output of the system and gold-standard are the same.

(ii) Incorrect (INC). The output of the system and gold-standard do not match.

(iii) Partial (PAR). The system output and gold-standard are somewhat similar but not the same.

(iv) Missing (Mis). A gold-standard is not captured by the system.

(v) Spurious (SPU). The output of the system does not exist in gold-standard.

Precision is the percentage of correct named entities extracted by the NER system, and recall shows the percentage of gold-standard named entities that are found by the system. In order to compute precision, recall, and *F*1score, two different scenarios can be used.

(vi) Exact Match. The exact match is stricter and considers partial matches as (INC).

(vii) Partial Match. This match considers partial matches as (PAR) and is less strict.

For calculating precision and recall, two quantities need to be defined. (1) Possible (POS), which is the number of gold-standard annotations contributing to the final score, and (2) actual (ACT), which is the number of NER system annotations. These two quantities are calculated as follows:

$$\begin{aligned} \text{POS} &= \text{COR} + \text{INC} + \text{PAR} + \text{MIS} = \text{TP} + \text{FN}, \\ \text{ACT} &= \text{COR} + \text{INC} + \text{PAR} + \text{SPU} = \text{TP} + \text{FP}. \end{aligned} \quad (18)$$

The exact match is computed as follows:

$$\begin{aligned} \text{Precision} &= \frac{\text{COR}}{\text{ACT}} = \frac{\text{TP}}{\text{TP}+\text{FP}}, \\ \text{Recall} &= \frac{\text{COR}}{\text{POS}} = \frac{\text{TP}}{\text{TP}+\text{FN}}. \end{aligned} \quad (19)$$

For the partial match, precision and recall would be as follows:

$$\text{Precision} = \frac{\text{COR} + 0.5 \times \text{PAR}}{\text{ACT}} = \frac{\text{TP}}{\text{TP} + \text{FP}},$$
$$\text{Recall} = \frac{\text{COR} + 0.5 \times \text{PAR}}{\text{POS}} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \quad (20)$$

Finally, the $F1$score, which is the harmonic mean of precision and recall, can be calculated as follows:

$$F1 = \frac{2 \times \text{Prcision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \quad (21)$$

4.1. Dataset. Since the proposed method is for texts extracted from social media, the golden standards corpus must be created from such texts. Sep_TD_Tel01 dataset [29] includes 10285 Telegram posts of public groups and channels collected from January 5, 2017, to July 20, 2017. This dataset has 720 windows in which every 12 hours is considered as a single window and have been utilized in [30]. Considering the volume of unusable information being shared on social media and labeling manually taking much time and effort, 300 posts of this dataset related to two topics were collected to create gold-standard for evaluating our system. We asked three people who are familiar with NER to label these posts based on IOB (inside, outside, and beginning) format in three versions.

In addition, to implement the proposed method, calculating the N-gram and probability of Wikipedia for every segment is needed. For the same reason, a database is created containing tables storing 1-gram through 5-grams [31], and another table for link texts of Wikipedia [32] which is available in ComInSys Lab.

As mentioned in Section 3 the Hamshahri corpus and Telegram posts have been used to develop N-Gram. To start the process, after eliminating stop words and normalization, the number of words is counted. Then, N-grams, up to 5-grams, are calculated and stored in the corresponding tables in the SQL database. The available Telegram corpus is not broad enough to generate reliable N-grams, but it can resolve the problem of the Hamshahri corpus that is outdated. It has taken five days to execute the process on a computer. Table 1 shows the total number of N-Gram extracted from these two corpora for $N = 1, 2, 3, 4, 5$.

To fill the table of Wikipedia link texts, the corpus of Persian Wikipedia, until May 24, 2020, including more than 3 million articles is used. Processing this corpus has two steps. In the first step, every link is detected and the body of each, after removing the stop words and normalization, is stored in the database. Then, document frequency (DF) and term frequency (TF) is calculated. In the second step, the occurrence of every text extracted from the previous step (either a link or a mere text) is counted. In the end, 3,177,923 link texts has been extracted from the mentioned corpus. The first step has taken 30 days by a single computer. In the second step, 6 computers have executed the process part-by-part parallelly which has taken a total time of 60 days.

Table 1: The number of N-Gram extracted from Hamshahri and Telegram from $N = 1$ up to $N = 5$.

N-Gram	Number of N-Gram
1-Gram	99,603,396
2-Gram	94,305,722
3-Gram	89,158,246
4-Gram	84,110,988
5-Gram	79,176,683

4.2. Preprocessing. Before collecting posts for creating gold-standard, we preprocessed the dataset to remove some posts that are not suitable for our use. These posts are the posts that are only a link or are repeated more than once. Duplicate posts were removed using edit distance criteria. Then, the posts were tokenized, and their token types, including stop words, punctuations, and emojis and were saved for further use.

4.3. Parameter Setting. Our proposed method includes five parameters that need to be adjusted. These parameters can be divided into two categories: 1-TwiNER parameters and 2-GWO parameters. TwiNER parameters are the K value, Y value, and stickiness function (PMI and SCP). GWO parameters are the number of wolves (NW) and the number of iteration (NI).

In the first step, we adjust TwiNER parameters. Figure 4 shows the results of the proposed system for these parameters. In these results, GWO parameters are considered to be fixed (NW = 50 and NI = 200) and the obtained $F1$ amounts are the average of 10 runs of the system. As we can see, our system has a better performance for $60 < K < 80$ in PMI, $60 < k < 100$ in SCP, and $\gamma = 0$ or $\gamma = 0.3$. Getting more detailed information determines that overall PMI functions better than SCP. Furthermore, $\gamma = 0.3$ has the better result except for SCP in the partial match. Also, the results are preferable when k is around 70 except for the SCP exact match which is around 95. In the rest of the paper, we will set the TwiNER parameter to PMI, $\gamma = 0.3$ and $k = 70$.

In the second step, the parameters of GWO need adjustment. Figure 5 indicates the result of the exact $F1$ (average of 10 runs) for different NW and NI, respectively. In order to tun NW, TwiNER parameters are fixed to PMI, $K = 70$, $\gamma = 0.3$, and NI is set to be 200. As shown in Figure 5(a), NW = 20 has the best output. Similarly, for tunning NI, NW is considered 20 with the same TwiNER parameters, the best output is for NI = 200.

4.4. Comparison with Other Methods. In this section, we compare our proposed method with three other NER systems. The first system is TwiNER [19] which our proposed method is based on it, the second system is a hybrid method for Persian NER [20], and the third one is based on monolingual BERT for the Persian language (ParsBERT) [16]. For a fair comparison with our proposed method, our dataset has been used in these systems. Moreover, we have employed modified TwiNER for Persian data that utilizes the Hamshahri corpus and Telegram posts since TwiNER is suggested for English tweets.

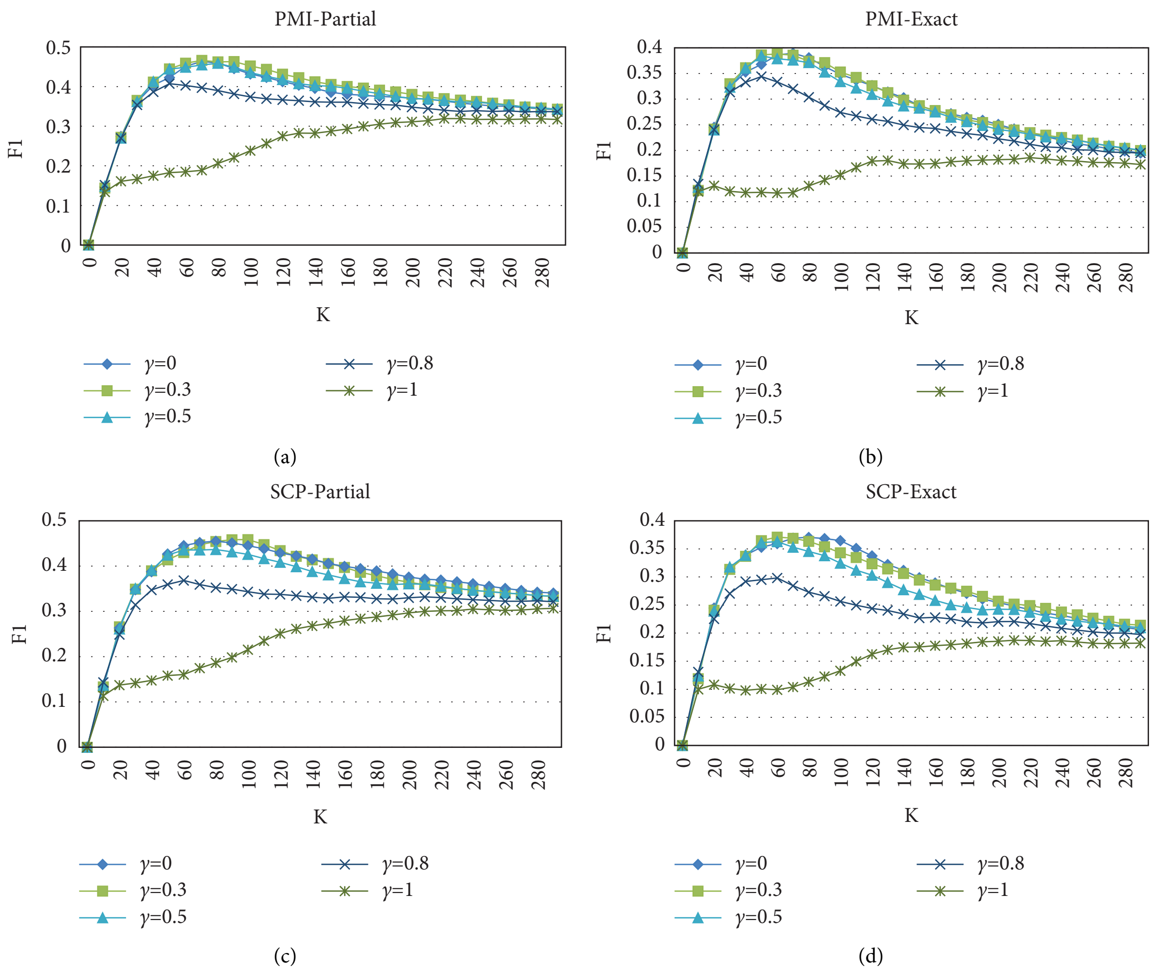


Figure 4: Results of proposed system for different values of K and Y. (a) The results of the partial match for PMI-based stickiness, (b) the results of an exact match with PMI stickiness function, (c) using partial match for the results of SCP-based stickiness, and (d) exact match results for SCP-based stickiness.

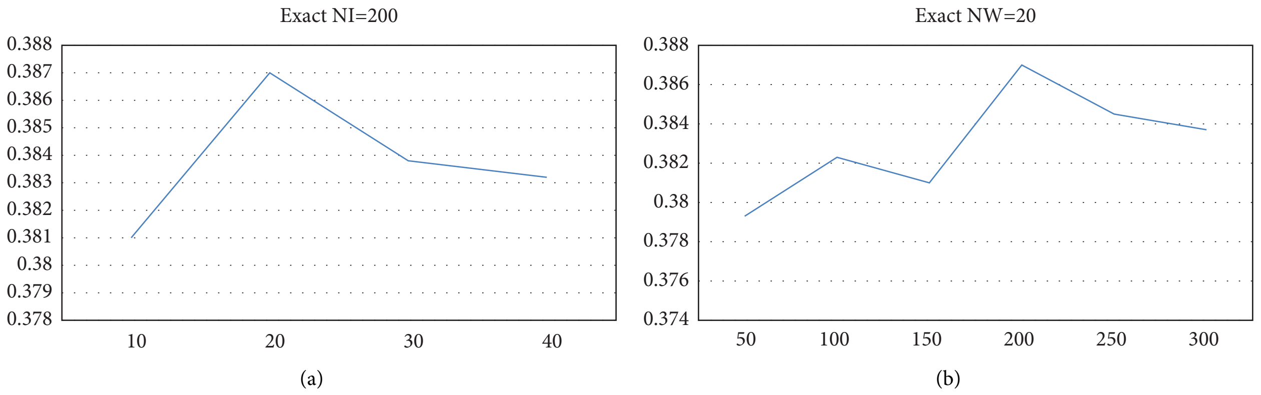


Figure 5: The result of the exact $F1$ for (a) different NW and (b) NI.

Before comparing our system with other methods, we compare our 3 gold-standards. The outcome shows that even human opinion can differ in recognizing named entities. The results of comparing these gold-standards are shown in Table 2. As indicated in this table, the average amount for $F1$ in all the partial matches is 0.5828, which is far from 1 (the highest possible amount). Therefore, we should not expect the system to have a high amount of precision and recall.

Table 3 shows the result of all these systems in terms of precision (Pr), recall (Re), and $F1$ for both partial (Par) and exact (Exc) match using GS2 as gold-standard. We take two

Table 2: Comparing three gold-standards (GS 1, GS 2, and GS 3) with each other.

Gold-standard		GS1		GS2		GS3		GS-average
Compared with		GS2	GS3	GS1	GS3	GS1	GS2	
Partial	Precision	0.584	0.450	0.735	0.541	0.630	0.633	0.595
	Recall	0.703	0.605	0.595	0.621	0.442	0.544	0.585
	F1	0.638	**0.516**	**0.658**	0.578	0.519	0.585	0.582
Exact	Precision	0.481	0.320	0.670	0.418	0.533	0.530	0.492
	Recall	0.579	0.430	0.543	0.480	0.373	0.455	0.477
	F1	0.525	**0.367**	**0.6**	0.447	0.439	0.489	0.478

The bold values show the highest and lowest results.

Table 3: Comparison of our proposed method (GWO + PMI and GWO + SCP) with other methods.

Method	ParPr	ParRe	ParF1	ExcPr	ExcRe	ExcF1
GWO + PMI	0.5273	0.4193	**0.4671**	0.4369	0.3474	**0.387**
GWO + SCP	0.4827	0.4293	0.4544	0.3837	0.3413	0.3612
TwiNER + PMI [19]	0.4595	0.3725	0.4115	0.3345	0.2712	0.2995
TwiNER + SCP [19]	0.4520	0.3879	0.4175	0.3289	0.2822	0.3038
ParsBERT [16]	0.54	0.367	0.437	0.427	0.29	0.345
Hybrid Persian NER [20]	0.3951	0.2237	0.2857	0.2096	0.1187	0.1516

The bold values show the best results.

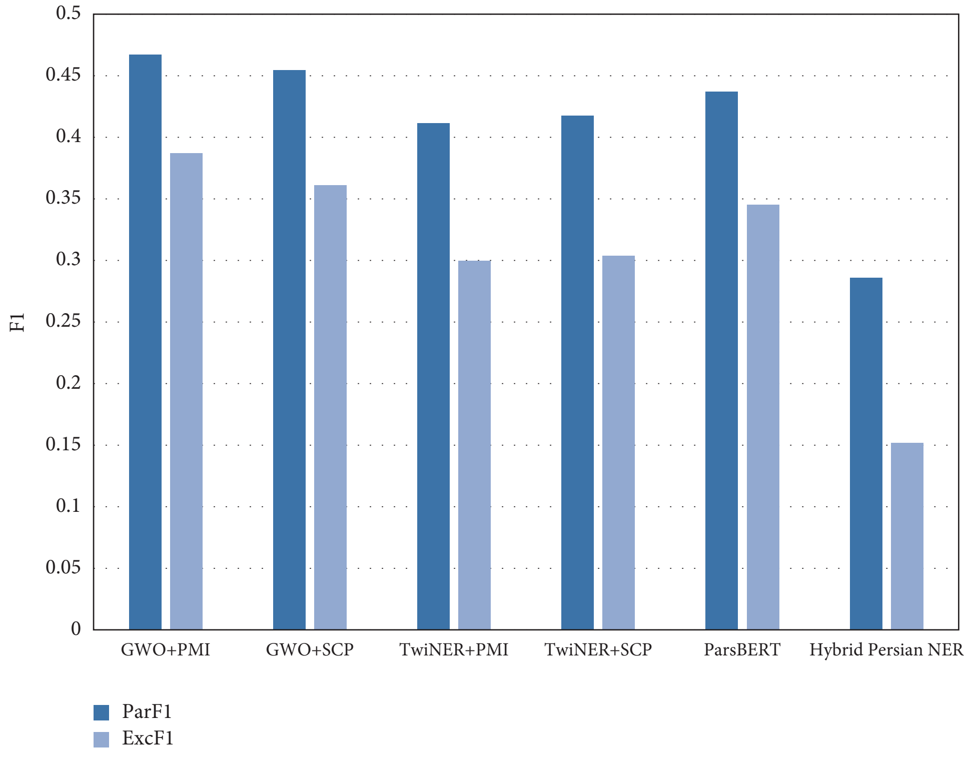


Figure 6: The result of partial and exact $F1$ for our proposed system (GWO + PMI and GWO + SCP) and other methods.

stickiness functions PMI and SCP into consideration in our comparison and as can be seen in Figure 6 our method outperform in both stickiness functions. Also, the results of our method are significantly better than hybrid Persian NER and superior to ParsBERT.

Based on the experiment conducted above, we see that our method shows promising performance in both stickiness functions by incorporating an optimization algorithm. However, in overall PMI outperformance SCP in our method, comparing our method with the average results of gold-standard also shows that our system is closer to human opinion results.

5. Conclusion

In this paper, we have proposed a statistical-based approach for Persian NER in which a binary GWO algorithm has been used. This system partition posts into segments of possible named entities. However, the number of segmentation states grows exponentially with respect to the length of the post. Therefore, to find the optimal segmentation, binary GWO has been employed. The fitness function for this problem is based on three criteria: (1) stickiness, (2) Wikipedia score, and (3) segment length. We have used the Hamshahri corpus and Telegram dataset to create N-Gram for calculating stickiness. Then, we score these segments with the random walk model to recognize true named entities among the segments. Additionally, to evaluate our method, the gold standard has been generated for Telegram posts by 3 individuals. The average partial $F1$ measure for these gold standards is 0.582 so even human opinion differs in NER. Therefore, we cannot expect NER systems to have a high amount in the $F1$ measure. Comparing our method with other methods with the same dataset shows that our proposed method outperforms these systems.

Data Availability

Previously reported "Sep_TD_Tel01," "Sep_Ngram_Tel-Ham01" and "Sep_Anchor-Title_Fawiki01" data were used to support this study and are available at https://doi.org/10.17632/372rnwf9pc, https://doi.org/10.17632/g4tnnf683m and https://doi.org/10.17632/tn22s9kvrt. These prior studies (and datasets) are cited at relevant places within the text as references [29–32].

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] M. K. Khormuji and M. Bazrafkan, "Persian named entity recognition based with local filters," *International Journal of Computer Application*, vol. 100, no. 4, pp. 1–6, 2014.

[2] K. Dashtipour, M. Gogate, A. Adeel, A. Algarafi, N. Howard, and A. Hussain, "Persian named entity recognition," in *Proceedings of the 2017 IEEE 16th International Conference on Cognitive Informatics and Cognitive Computing, ICCI*, pp. 79–83, Beijing China, October 2017.

[3] K. Riaz, "Rule-based named entity recognition in Urdu," in *Proceedings of the 2010 Named Entities Workshop*, NY China, July, 2010.

[4] J. H. Kim and P. C. Woodland, "A rule-based named entity recognition system for speech input," in *Proceedings of the 6th International Conference on Spoken Language Processing, ICSLP 2000*, Beijing, China, October 2000.

[5] D. Farmakiotou, V. Karkaletsis, J. Koutsias, S. George, C. D. Spyropoulos, and P. Stamatopoulos, "Rule-based named entity recognition for Greek financial texts," in *Proceedings of the Workshop on Computational lexicography and Multimedia Dictionaries (COMLEX 2000)*, Athens, Greece, May 2000.

[6] R. Alfred, L. C. Leong, C. K. On, and P. Anthony, "Malay named entity recognition based on rule-based approach," *International Journal of Machine Learning and Computing*, vol. 4, no. 3, pp. 300–306, 2014.

[7] H. Poostchi, E. Zare Borzeshi, and M. Piccardi, "BILSTM-CRF for Persian named-entity recognition armanpersonercorpus: the first entity-annotated Persian dataset," in *Proceedings of the LREC 2018 - 11th International Conference on Language Resources and Evaluation*, Paris, France, June 2019.

[8] J. P. C. Chiu and E. Nichols, "Named entity recognition with bidirectional LSTM-CNNs," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 357–370, 2016.

[9] M. Konkol, T. Brychcín, and M. Konopík, "Latent semantics in named entity recognition," *Expert Systems with Applications*, vol. 42, no. 7, pp. 3470–3479, 2015.

[10] S. Momtazi and F. Torabi, "Named entity recognition in Persian text using deep learning," *Signal and Data Processing*, vol. 16, no. 4, pp. 93–112, 2020.

[11] E. Taher, S. A. Hoseini, M. Shamsfard, and N. E. R. Beheshti, *Persian Named Entity Recognition Using BERT*, Netherlands, 2019.

[12] M. Asgari-Chenaghlu, M. R. Feizi-Derakhshi, L. Farzinvash, M. A. Balafar, and C. Motamed, "CWI: a multimodal deep learning approach for named entity recognition from social media using character, word and image features," *Neural Computing & Applications*, vol. 34, no. 3, pp. 1905–1922, 2022.

[13] C. Dogan, A. Dutra, G. Adam et al., *Fine-grained Named Entity Recognition Using ELMo and Wikidata*, Amsterdam, 2019.

[14] M. Zali and M. Firoozbakht, "Named entities recognition and classification system for Persian texts based on neural network," *Iranian Journal of Information Processing Management*, vol. 34, no. 1, 2018.

[15] L. J. Tafreshi and F. Soltanzadeh, "A novel approach to conditional random field-based named entity recognition using Persian specific features," vol. 8, no. 2, pp. 227–236, 2020.

[16] M. Farahani, M. Gharachorloo, M. Farahani, and M. Manthouri, "ParsBERT: Transformer - Based Model for Persian Language Understanding," *Neural Processing Letters*, vol. 53, 2020.

[17] H. Poostchi, E. Zare Borzeshi, M. Abdous, and M. Piccardi, "PersoNER: Persian named-entity recognition," in *Proceedings of the COLING 2016 - 26th International Conference on Computational Linguistics, Proceedings of COLING 2016*, Technical Papers, Seoul Korea, July 2016.

[18] D. Bikel, M. Scott, R. Schwartz, and R. Weischedel, "Nymble: A High-Performance Learning Name-Finder," *ANLP*, vol. 20, 1997.

[19] L. Chenliang, J. Weng, He Qi et al., "TwiNER: named entity recognition in targeted twitter stream," in *Proceedings of the SIGIR'12 - Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, NY, United States, August 2012.

[20] H. Moradi, F. Ahmadi, and M. R. Feizi-Derakhshi, "A hybrid approach for Persian named entity recognition," *Iranian Journal of Science and Technology Transaction A-Science*, vol. 41, no. 1, pp. 215–222, 2017.

[21] H.-K. Kung, C.-M. Hsieh, C.-Y. Ho, Y.-C. Tsai, H. Y. Chan, and M. H Tsai, "Data-augmented hybrid named entity

recognition for disaster management by transfer learning," *Applied Sciences*, vol. 10, no. 12, p. 4234, 2020.

[22] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.

[23] E. Emary, H. M. Zawbaa, and A. E. Hassanien, "Binary grey wolf optimization approaches for feature selection," *Neurocomputing*, vol. 172, pp. 371–381, 2016.

[24] E. Emary, W. Yamany, A. E. Hassanien, and V Snasel, "Multiobjective gray-wolf optimization for attribute reduction," *Procedia Computer Science*, vol. 65, pp. 623–632, 2015.

[25] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016.

[26] G. Negi, A. Kumar, S. Pant, and M. Ram, *A Review and Applications*, NY China, 2021.

[27] I. Segura-bedmar and P. Mart, "SemEval-2013 task 9: extraction of drug-drug interactions from biomedical texts," *DDIExtraction*, vol. 2, 2013.

[28] N. Chinchor and B. Sundheim, *MUC-5 Evaluation Metrics*, China, 1993.

[29] M. Ranjbar-Khadivi, M. R. Feizi Derakhshi, A. Forouzandeh, P. Gholami, A. R. Feizi-Derakhshi, and E. Zafarani-Moattar, "Sep_TD_Tel01," *Mendeley Data*, vol. V1, 2022.

[30] P. Gholami-Dastgerdi, M.-R. Feizi-Derakhshi, and A. Forouzandeh, "Named entities detection by beam search algorithm," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 27, 2022.

[31] P. Gholami-Dastgerdi, M. R. Feizi-Derakhshi, M. Ranjbar-Khadivi, E. Zafarani-Moattar, A. R. Feizi-Derakhshi, and A. Forouzandeh, *Sep_Ngram_Tel-Ham01"*, Mendeley Data, 2022.

[32] P. Gholami-Dastgerdi, M. R. Feizi-Derakhshi, and M. Ranjbar-Khadivi, *Sep_Anchor-Title_Fawiki01*, Mendeley Data, 2022.