Applied Mathematical Modelling 38 (2014) 3193–3211

Contents lists available at ScienceDirect

Applied Mathematical Modelling

journal homepage: www.elsevier.com/locate/apm

Hybrid PoS-tagging: A cooperation of evolutionary and statistical approaches

Rana Forsati [*,1], Mehrnoush Shamsfard

Natural Language Processing (NLP) Research Lab., Faculty of Electrical and Computer Engineering, Shahid Beheshti University, G. C., Tehran, Iran

ARTICLE INFO

Article history:
Received 6 September 2012
Received in revised form 13 November 2013
Accepted 26 November 2013
Available online 20 December 2013

Keywords:
Part-of-Speech tagging
Bee colony optimization
Natural language processing
Evolutionary algorithms
Optimization

ABSTRACT

The assigning of syntactic categories to words in a sentence, which is referred to as part-of-speech (PoS) tagging problem, plays an essential role in many natural language processing and information retrieval applications. Despite the vast scope of methods, PoS-tagging brings an array of challenges that require novel solutions. To address these challenges in a principled way, one solution would be to formulate the tagging problem as an optimization problem with well-specified objectives and then apply the evolutionary methods to solve the optimization problem. This paper discusses the relative advantages of different evolutionary approaches to handle Part-of-Speech tagging problem and aims at presenting novel language-independent evolutionary algorithms to solve the PoS tagging problem. We show that by exploiting statistical measures to evaluate the solutions in tagging process, the proposed algorithms are able to generate more accurate solution in a reasonable amount of time. The experiments we have conducted on few well known corpus reveal that the proposed algorithms achieve better average accuracy in comparison to other evolutionary-based and classical Part-of-Speech tagging methods.

1. Introduction

One of the main building blocks in many applications such as machine translation, text-to-speech conversion, question answering, speech recognition, word sense disambiguation and information retrieval is to extract basic syntactic features of individual words in the sentences [1,2]. This problem which is known as Part-of-Speech tagging (PoS) has been the main focus of many researches and different models and methods have been proposed to tackle it, but still developing fast and high-quality tagging systems is a challenging issue.

PoS-tagging aims at assigning the syntactic categories to every word in a sentence according to its context [2] – i.e., relationship with adjacent and related words in a phrase, sentence, or paragraph. This task is not trivial since many words could have a large number of candidate tags associated with them. To gain a sense of the complexity of the problem we note that according to [3] over 40% of the words appearing in the hand-tagged Brown corpus [4] are syntactically ambiguous. Thus, ambiguity resolution is the key challenge in tagging. Typically, there are different kinds of approaches for automatic PoS-tagging which differs based on the amount of training, their internal model and processing of information required.[2] In recent years, there has been a growing interest in data-driven machine learning disambiguation methods, which can be used in many situations such as tagging [6]. We can find several approaches to PoS-tagging based on rule-based approaches [6,7],

* Corresponding author. Tel.: +98 2129904111.
E-mail addresses: r_forsati@sbu.ac.ir (R. Forsati), m_shams@sbu.ac.ir (M. Shamsfard).
[1] Part of this work was done when RF was a visiting scholar at University of Minnesota.
[2] For a state-of-the-art list of methods one can refer to [5].

0307-904X/$ - see front matter © 2014 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.apm.2013.11.047

statistical and machine learning techniques, including among many others: Hidden Markov Models (HMM) [8–10] and its variants [11,12], Maximum Entropy taggers [13,14], Transformation based learning [7], and its variants [15], Memory based learning [16], Decision Trees [17], Conditional Random Field [18], Support Vector Machines [19], and averaged perceptron [20]. Among stochastic models, HMMs are quite popular.

The classical techniques, i.e., rule-based approaches [6,7], define how each word is assigned to its corresponding PoS-tag employing rules, given an input word sequence. Rule-based approaches apply language rules to improve the accuracy of tagging [21] and consist of a two-stage architecture for designing PoS-tagging. The first stage of these systems is to apply dictionary and to assign all possible PoS-tags to every word. The second stage employs a number of handcrafted disambiguation rules to find out the most appropriate tag for each word [22]. This stage is labor intensive and time consuming because of the need to write rules or templates for each language. In addition when the format changes it is not robust and may fail.

Statistical taggers generally solve the tagging problem by using a tagged corpus, i.e., a set of texts linguistically labeled [23] to compute the probability of a given word having a specified tag in a given context. In statistical methods the most probable tag sequence is identified based on the statistical occurrence of the tag N-gram and word-tag frequencies. In particular, speaking, statistical methods amounts to maximize a global measure of the probability of the set of contexts (a tag and its neighboring tags) corresponding to a given tagging of the sentence [24]. These methods have been utilizied in PoS-tagging. Thanks to simplicity, language independence [25] and increased availability of large tagged corpora [6], research in this direction has grown extensively in recent years. In particular, few studies showed that statistical methods yield better performance than the other tagging approaches [10,26,27].

Features such as ambiguity make PoS-tagging as a complex search process, which require specifically designed algorithms to be efficiently solved. The use of heuristic methods can help to deal with this problem. When one deals with optimization problems with an extremely large space of solutions, evolutionary algorithms (EAs) are among the favorable methods to be employed. The efficiency of EAs algorithm in exploring the large search spaces is mostly due to their stochastic nature. This characteristic results in the rapid location of good candidate solutions and makes evolutionary based methods appealing for many applications where we want to optimize global function over complex search spaces.

In the evolutionary-based tagging algorithms, each candidate solution is represented as a sequence of tags assigned to the words in a sentence. This is in contrast to statistical methods where there is a disambiguation in the assigned tags by coupling each tag with a probability. The probability of a tag indicates how likely this tag would be the correct assignment for a given word in the sentence. Usually this probability depends on the neighboring tags on both sides of the word [21]. The experiments conducted in [24] revealed that the evolutionary based tagging algorithms are able to attain better accuracy compared to statistical approaches. Surprisingly, evolutionary based methods also generate more robust to the inherent ambiguity.

In the recent years some works have been done using machine-learning approaches and, a small number of studies have applied the evolutionary algorithms to PoS-tagging task to improve the quality of solutions. Among the different evolutionary algorithms, genetic algorithm (GA) [21,24] and simulated annealing (SA) [21] have been previously applied to solve PoS-tagging problem.

Recently, researchers have also proposed hybrid EAs based algorithms for PoS tagging problem which are aimed at the cooperation of EA based methods with classical tagging algorithms (i.e, rule-based and statistical methods). Losee [28] describes an implementation called LUST (Linguistics Using Sexual Techniques) that also uses GAs to generate an entire grammar for a test corpus. In [29] an algorithm proposed which interpolates between evolutionary computing and inductive logic programming (ILP). The main idea is that instead of considering the entire research space, a small subset can be examined by using ILP rules. Curran and Wong [30] have actually suggested the use of evolved transformations in the Brill Tagger.

In this paper, we show that by casting PoS-tagging problem as an optimization problem which intends to locate a solution in the solution space which optimize a predefined global objective function, we are able to utilize the harmony search (HS) and bee colony optimization (BCO) algorithms in the scope of tagging problem. In particular, we are interested in the cooperation between evolutionary algorithms and statistical PoS-tagging to take advantages of both methods. The main intuition behind the proposed hybrid methods stems from the observation that the statistical measurements extracted by the stochastic approaches to PoS-tagging provide an appropriate fitness function for evolutionary algorithms in a natural way [23]. The used statistical model amounts to maximize a global measure of the probability of the set of contexts (a tag and its neighboring tags) corresponding to a given tagging of the sentence.

Harmony search (HS) [31,32], is a new optimization method imitating the music improvisation process, where musicians improvise the pitch of their instruments searching for a perfect state of harmony. Since its inception, HS has been vigorously applied to a wide variety of practical optimization problems [33–35]. The Bee Colony Optimization (BCO) is also another evolutionary algorithm from the family of stochastic meta-heuristic optimization methods. In BCO analogous to bees, special types of artificial bees are created which collaboratively solve complex combinatorial optimization problems. BCO has proven to be an efficient algorithm in many domains [36,37]. We note that statistical measurements, which are extracted by the statistical taggers, are appropriate alternatives for use as the function to compare the solutions which are extracted by the EAs. Accordingly, in this paper a suitable framework for application of evolutionary techniques such as HS and BCO to statistical PoS-tagging is presented.

One of the advantages of using an evolutionary algorithms as the search algorithm for PoS-tagging is that these algorithms can be applied to any statistical model, even if they do not rely on the Markov assumption (i.e., the tag of a word only depends on the previous words), as it is required in classic search algorithms for tagging, such as the widely used Viterbi

[38,24]. In this way, they can also be applied to other models in which the context of a word is composed of both, the tag of the preceding words and also the tag of the following words [24].

The structure of the individual in the proposed algorithms is simply a tag with different probabilities of associated contexts attached to it. Individuals structure amounts to a tag assignment mechanism that is statistical. The fitness function for the implementation is rather complex. By using the statistical fitness function, we attempt to search for the most probable tag for a word expressed as a component of the solutions.

In comparison to the existing algorithms, and in particular a preliminary version of this paper [39], the main contributions of the present work can be summarized as:

- We introduce a general framework to utilize the evolutionary based optimization methods in combination with statistical methods to tackle the PoS-tagging problem. We note that [39] only considers bee colony method for this purpose.
- We elaborate different fitness measures proposed in the literature and discuss the weak and strong points of each measure. We also propose a novel fitness measure which intelligently balances the exploration and exploitation during the optimization process.
- The proposed algorithms can be applied to any training text of any language, without making any modification to the model or having a post-processing phase.
- We empirically evaluate how different parameters affect the performance of the proposed algorithms and provide few hints on deciding the value of these parameters.
- We conduct experimental results on standard corpus data sets with different characteristics to demonstrate the merits of the proposed framework in terms of accuracy and convergence rate in comparison to both evolutionary based and classic algorithms. We also evaluate the different fitness measures proposed in the literature. The results show that the proposed methods in combination with the fitness function proposed in this paper get more accurate tagging results with faster convergence than other methods.

Outline. The remainder of this paper is organized as follows: Section 2 provides a formal definition of the probabilistic PoS-tagging problem. Section 3 provides a general overview of the harmony search algorithm and bee colony optimization technique. Section 4 provides a detailed description of evolutionary-based Pos-tagging algorithms. Section 5 presents the test bed of our experiments, empirical study of evolutionary parameters on convergence behavior of the proposed algorithm, and the performance evaluation of the proposed algorithms compared to other algorithms. Finally, Section 6 summarizes the conclusions of this work.

2. Probabilistic PoS-tagging modeling

Given an input sentence $S = w_1 w_2 \cdots w_n$ consisting of n words to be tagged, the goal of a tagging algorithm is to determine the best sequence of tags for the sentence. More specifically, the goal is to find the tag sequence T^* that has optimal adequacy with respect to all other feasible candidate solutions in $\mathcal{T} = \{T^1, T^2, \ldots, T^{N(n)}\}$ where each $T^i = \{t_1^i, t_2^i, \ldots, t_n^i\}$ represents a candidate tagging, $N(n) = \prod_{j=1}^{n} k_j$ is the number of all feasible tag sequences to tag the sentence where k_i is the number of valid tags for ith word in the sentence, i.e., w_i.

The Bayesian interpretation of this task starts by considering all possible sequences of tags. Out of this universe of tag sequence, we want to choose the tag sequence which is most probable given the observation sequence of n words [14]. On the other hand, from the statistical point of view, PoS tagging is defined as a task of choosing a tag-sequence with the maximum probability. Moreover, in statistical methods such as HMM tagger [8–10], we need to make two simplifying assumptions too [40]. The first assumption is that the probability of a word appearing is dependent only on its own part-of-speech tag; that it is independent of other words around it, and of the other tags around it. The second assumption is the tag of a word only depends on the K previous tags (transition probability) [14]. We search for a solution in $\mathcal{T}$ in a way that maximizes the following objective function:

$$T^* = \arg\max_{T \in \mathcal{T}} \Pr(T|S) = \arg\max_{T \in \mathcal{T}} \left[\prod_{i=1}^{n} \Pr(w_i|t_i) \times \Pr(t_i|t_{i-1}) \right] \tag{1}$$

where T^* is the best sequence of tags that maximizes (1), $\Pr(w_i|t_i)$ denotes the emission probability, and $\Pr(t_i|t_{i-1})$ is a bigram transition probability.

Having computed the transition and emission probabilities and assigning all possible tag sequences to all the input words, now we need an algorithm that can search the tagging sequences and maximize the product of transition and emission probabilities.

The complexity of this problem increases with the length of the sentence to be tagged. Having computed the transition and emission probabilities and assigning all possible tag sequences to all the input words, now we need an algorithm that can search among all the tagging sequences $\mathcal{T}$ and find the tagging which maximizes the product of transition and emission probabilities. For this purpose, by modeling PoS-tagging as an optimization problem, we propose HS and BCO based tagging algorithms that usually provide more accurate results in a reasonable computation time.

In the next sections, firstly HS and BCO algorithms are described and the details of the each proposed algorithm based on HS and BCO to handle PoS-tagging are discussed. Table 1 contains a list of the basic notations we used throughout the paper.

3. Basic algorithms

3.1. The harmony search algorithm

One of the popular search methods which mimic the music improvisation process is harmony search (HS) [41]. Recent years have witnessed a flurry of research on HS to solving different optimization problems [33–35,42,43]. The main steps of the algorithm are as follows:

1. Initialize the problem and algorithm parameters.
2. Initialize the harmony memory.
3. Improvise a new harmony.
4. Update the harmony memory.
5. Check the stopping criterion.

These steps are described in the next five subsections.

3.1.1. Initialize the problem and algorithm parameters

In Step 1, the optimization problem is specified as follows:

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} \quad f(x) \\ &\qquad\quad x_k \in [l_k, u_k] \quad k = 1, 2, \ldots, n, \end{aligned}$$

where $f(x)$ is the objective function and n is the number of decision variables. The lower and upper bounds for each decision variable are l_k and u_k, respectively. The HS parameters are also specified in this step. These are the harmony memory size (HMS), or the number of solution vectors in the harmony memory, the probability of memory considering (HMCR), the probability of pitch adjusting (PAR), and the number of improvisations (NI), or stopping criterion. The harmony memory (HM) is a memory location where all the solution vectors (sets of decision variables) are stored. This HM is similar to the genetic pool in the GA. The HMCR, which varies between 0 and 1, is the rate of choosing one value from the historical values stored in the HM, while 1 − HMCR is the rate of randomly selecting one value from the possible range of values.

3.1.2. Initialize the harmony memory

In this step, the HM matrix is filled with as many randomly generated solution vectors as the HMS:

$$HM = \left[\begin{array}{ccccc|c} x_1^1 & x_2^1 & \ldots & x_{n-1}^1 & x_n^1 & f(x^1) \\ x_1^2 & x_2^2 & \ldots & x_{n-1}^2 & x_n^2 & f(x^2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1^{\mathrm{HMS}-1} & x_2^{\mathrm{HMS}-1} & \ldots & x_{n-1}^{\mathrm{HMS}-1} & x_n^{\mathrm{HMS}-1} & f(x^{\mathrm{HMS}-1}) \\ x_1^{\mathrm{HMS}} & x_2^{\mathrm{HMS}} & \ldots & x_{n-1}^{\mathrm{HMS}} & x_n^{\mathrm{HMS}} & f(x^{\mathrm{HMS}}) \end{array}\right]$$

The initial harmony memory is generated from a uniform distribution in the ranges $[l_i, u_i]$, where $1 \leqslant i \leqslant n$. This is done as follows:

$$x_i^j = l_i + r \times (u_i - l_i), \quad j = 1, 2, \ldots, \mathrm{HMS}$$

where $r \sim U(0, 1)$ and U is a uniform random number generator.

Table 1
Summary of notations consistently used in the paper and their meaning.

Symbol	Meaning
$S = (w_1, w_2, \cdots, w_n)$	The input sentence with n words to be tagged
$\mathcal{T}$	The set of all valid taggings for a sentence under consideration
$N(n)$	The size of T for a sentence with n words
$T = (t_1, t_2, \cdots, t_n)$	An instance tagging of sentence S with n words
k_i	The number of candidate tags for word w_i
HM	Harmony memory
HMS	Harmony memory size
HMCR	Harmony memory consideration rate
PAR	Pitch adjusting rate
NHV	A solution resulted from improvisation step of HS algorithm
B	The number of bees

3.1.3. Improvise a new harmony

Generating a new harmony is called improvisation. A new harmony vector (NHV), $x' = (x'_1, x'_2, \ldots, x'_n)$, is generated based on three rules: (i) memory consideration, (ii) pitch adjustment, (iii) random selection. In the memory consideration, the value for a decision variable is randomly chosen from the historical values stored in the HM with the probability of HMCR. Every component obtained by the memory consideration is examined to determine whether it should be pitch-adjusted. This operation uses the PAR parameter, which is the probability of pitch adjustment. If it happens that the decision variable x'_i to be pitch adjusted, its value becomes $x'_i \leftarrow \pm r \times b$ where b is an arbitrary distance bandwidth and $r \sim U(0,1)$. Variables which are not selected for memory consideration will be randomly chosen from the entire possible range with a probability equal to $1 - \text{HMCR}$.

3.1.4. Update harmony memory

If the new harmony vector, $x' = (x'_1, x'_2, \ldots, x'_n)$, has better fitness function than the worst harmony in the HM, the new harmony is included in the HM and the existing worst harmony is excluded from the HM.

3.1.5. Check stopping criterion

The HS is terminated when the stopping criterion (e.g., maximum number of improvisations) has been met. Otherwise, Steps 3 and 4 are repeated.

3.2. Bee colony optimization

The bee colony optimization (BCO) which uses the basic principles of collective bee intelligence in solving combinatorial optimization problems is the youngest Swarm Intelligence technique which has been proposed by [44]. The algorithm simulates the intelligent behavior of bee swarms. It is based on the concept of cooperation, which increases the efficiency of artificial bees and allows achievement of goals that could not be reached individually. The basic idea is to create a multi-agent system (a colony of artificial bees) capable of successfully solving difficult combinatorial optimization problems. A population of artificial bees collaboratively searches for the optimal solution with every artificial bee generating one solution to the problem [37].

The algorithm consists of two alternating phases constituting single step in the BCO algorithm: a forward pass and a backward pass. In each forward pass, every artificial bee visits NC components (number of constructive moves), creates partial solution, and then returns to the hive and start the second phase, the so-called backward pass [37]. The number of solution components NC to be visited within one forward pass is prescribed by the analyst at the beginning of the search process. During the backward pass, all bees share information about their solutions.

During this pass, based on the quality of the partial solutions generated, every bee decides with a certain probability whether it will advertise its solution or not. The bees with better solutions have more chance to advertise their solutions. The remaining bees have to decide whether to continue to explore their own solution in the next forward pass, or to start exploring the neighborhood of one of the solutions being advertised. Similarly, this decision is taken with a probability, so that better solutions have a higher probability of being chosen for exploration. Depending on the quality of the partial solutions generated, every bee possesses certain level of loyalty to the path leading to the previously discovered partial solution. Artificial bees that are loyal to their partial solutions also act as recruiters, i.e., their solutions would be advertised. Once the solution is abandoned by a bee, the bee becomes uncommitted and has to select one of the advertised solutions. This decision is taken with a probability too, so that better advertised solutions have bigger opportunity to be chosen for further exploration.

The two phases of the search algorithm, forward and backward passes, are alternating in order to generate all required feasible solutions (one for each bee). The first iteration is finished when bees for the first time create one or more feasible solutions by visiting all the nodes. When all solutions are completed the best discovered solution during the first iteration is determined. It is used to update global best solution and an iteration of the BCO is accomplished. The two phases of the BCO are carried out iteration by iteration, until a stopping condition is satisfied.

4. Cooperative evolutionary and statistical PoS-taggers

In this section, we introduce the algorithms based on cooperation of evolutionary and statistical techniques to handle PoS-tagging. The proposed algorithms largely rely on the success of statistical method to PoS-tagging. In these algorithms, the assessment of the established solutions by the evolutionary techniques (harmony search and bee colony optimization) is based on the statistical measurements extracted by stochastic techniques. Statistical methods have used an acceptable benchmark for fine-tuning the parameters of the algorithms and evaluating the fitness function. Therefore statistical measurements which are extracted from an annotated corpus or training text are required.

The measurement of the essential elements of evolutionary algorithms, such as the fitness function and the adjustment of parameters, are done by using probabilistic models whose values are determined by statistical techniques.

In order to design cooperative evolutionary and statistical based taggers, individuals consisting of sequences of tags assigned to each word in the sentence to be tagged plus some additional information useful in the evaluation (such as counts of different contexts for this tag according to the training table) are required. The structure of the individual in the proposed algorithms is simply a tag with probabilities of various associated contexts attached to it. Individuals structure amounts to a tag assignment mechanism that is statistical. Moreover, the probabilistic model, which assigns a probability to a sequence of tags according to the context of each word in the sentence, provides the fitness function. The evolution process is run for each sentence in the text to be tagged.

Evolution aims to maximize the probability of the tagging of the sentences in the test corpus. So we have used the most common statistical models for tagging which are based on assigning a probability to a given tag according to its neighboring tags (context). Then, as in HMM, the tagger tries to maximize the total probability of the tagging of each sentence. The process finishes either when the evolutionary process has been running for a maximum number of iterations or when the fitness deviation lies below a threshold value (convergence).

In the next section preparing the statistical measurement which is commonly used in the proposed algorithms is discussed. Then two PoS-tagging algorithms based on harmony search and bee colony optimization are proposed. The details of the each proposed algorithms have been discussed.

4.1. Preliminary data setup

In the proposed algorithms, the evaluation of solutions and fine-tuning the parameters are based on probabilistic model, therefore the tagging process requires to construct both word-PoS frequencies (lexicon) and the frequencies of the context of the words (N-gram). We have extracted and computed this information from the Brown corpus [4].

Word-PoS frequencies are computed and recorded in a table called PoS lexicon. To be consistent with the POS tagging literature, in order to generate the PoS lexicon, the frequency of each word in the training corpus, with an especial tag, is computed. Finally, words and their PoS candidates are written in the lexicon format. In the next sections, we show that this is one of the most important features of PoS-tagging. N-gram frequencies are also computed and recorded in a table called contextual table. This table can be computed by going through the training texts of corpus and recording the different contexts (neighboring words) and the number of occurrences of each of them for every tag in the training text. The set of attributes consists of a window covering tags to the left and right. The size of this window, which is the number of tags it covers around each word, has been determined on an empirical basis. All the trigrams ($N = 3$) are also applied to the candidate tags during the unknown-word PoS guessing and smoothing portion of the evolutionary based tagging process.

Table 2 shows some real examples from the contextual table where the tags are extracted from the Brown corpus. In this table, tags correspond to the tag set defined in the Brown corpus: "AT" stands for article, "NN" for common noun, "IN" for preposition and "JJ" for adjective.

4.2. Harmony search based tagger algorithm

In this section, harmony search based tagger algorithm namely HSTAGger is proposed. HSTAGger uses random selection and refines the solutions at each iteration. In order to tag the sentences using harmony search algorithm, we must first model PoS-tagging as an optimization problem that locates the optimal sequence of tags, with tagging quality as the objective, and use a suitable general purpose optimization method to find a good tag sequence. When a general purpose evolutionary algorithms is used for tagging problem, a number of important design decisions have to be made. Predominantly, these are the problem representation and the objective function. Both of these can have an important effect on optimization performance and thus on the quality of tagger.

In all of the proposed algorithms in the rest of the paper, each word represents one component of a multi-component sentence and each possible solution is a vector of tag values. This model offers us a chance to apply HS optimization algorithm to tag the sentence. In this section we describe HSTAGger algorithm and discuss its initialization, improvisation steps and an evaluation and termination criteria to continue or stop the process.

4.2.1. Representation of solutions

The first decision in solving sentence tagging problem by HS is how to represent solutions to apply harmony operations. Let us consider a sentence S formed by n words $\{w_i, i = 1, 2, ,n\}$ in which each w_i (ith word) has k_i possible tags ($1 \leqslant i \leqslant n$). To represent this situation we assign a vector consisting of n integers to each row of HM, each element of the vector denoting

Table 2
Sample of contextual table extracted from the Brown corpus [4].

Tag_{-3}	Tag_{-2}	Tag_{-1}	$<$ tag $>$	Tag_{+1}	Tag_{+2}	Tag_{+3}	# Of occurrence
NN	IN	AT	NN	IN	AT	NN	47
AT	NN	IN	AT	NN	IN	AT	31
AT	NN	IN	AT	JJR	NN	IN	48

the tag label of a word in the sentence. In other words, each position in the vector corresponds to a word in the sentence and its value shows its PoS-tag chosen among k_i possible tags for that word.

Number of rows denotes the number of different tag sequences among which we are looking for the best.

4.2.2. Initialization

For initialization, HM is filled with as many randomly generated solution vectors as the size of the HM (HMS). Each row of harmony memory corresponds to a specific sequence of tags in which, the value of the ith element in each row is randomly selected from the uniform distribution over the set $\{1, 2, \ldots, k_i\}$ and indicates the tag index number of ith word (we recall that k_i is the number of valid tags for ith word).

4.2.3. Improvisation step

In improvising step, we need a technique to generate a new harmony vector, namely NHV, from all the HMS solution vectors that are in HM. The new generated harmony vector must inherit as much information as possible from the solution vectors that are in HM. If the generated vector, which is corresponding to a new tag sequence, consists mostly or entirely of assignments found in the vectors in HM, then it provides good heritability.

In this algorithm, each decision variable corresponds to a word and its value indicating the tag label of that word. The selection of a value for the tag label of a word is as follows: the tag number of each word in the new solution vector is selected with probability of HMCR from the harmony memory and with probability of $(1 - \text{HMCR})$ is randomly selected from the set $\{1, 2, \ldots, k_i\}$. After generating the new solution, the PAR process is applied. PAR is originally the rate of allocating a different tag index to a word. In the original HS algorithm, PAR is the rate of moving from current selected tag to a neighboring tag. We have modified PAR process for our computation. In this computation, PAR is defined as the rate of moving to the most probable tag which has the highest frequency in the lexicon among the other tags for that word. PAR in the HS algorithm is a very important parameter for fine-tuning the optimized solution vectors and can be potentially useful in adjusting convergence rate of algorithm to optimal solution.

For each word w_i that the label of its tag is selected from HM, the current tag of w_i is replaced with the new valid tag chosen randomly from the following distribution for that word, with probability of PAR:

$$\Pr(j) = \Pr(\text{tag j being selected as new tag for ith word}) = \Pr(t_j|w_i) = \frac{count(w_i, t_j)}{\sum count(w_i, t_k)}$$

4.2.4. Evaluation of solutions

Designing a good fitness function is a key issue in harmony search. Each row in HM corresponds to a possible sequence of tags for input sentence which is comprised of n words. Let $T = (t_1, t_2, \ldots, t_n)$ be the set of n tags for a row in HM. Our objective function is to discover the proper sequence of tags for a sentence S.

Although few works [21,23,24,45], use only transition-based fitness, experiments show that the lack of word-tag frequencies is a significant weakness in evaluation of solutions and exploiting N-gram frequencies in the fitness metric is not enough. Word-tag frequencies strongly correlate with the quality of the tagger and this feature is thus extremely important in automatic taggers. HSTAGger uses and emphasizes modified emission in addition to transition.

Fitness value of each row, which corresponds to one potential solution, is computed by trigram transition probabilities, i.e., $\Pr(t_i|t_{i-1}, t_{i+1})$, and word-tag frequencies, i.e., $\Pr(w_i|t_i)$, according to the data extracted from the training table. The transition probabilities are extensively appropriate since they are composed of a certain number of tags on the left and another on the right of the word at the considered position. Viterbi [38], a classical method used for stochastic tagging cannot be applied in this case because this algorithm is designed to search the data sequence which maximizes the observed data according to a Markov model, i.e., a model in which the current state only depends on the previous one [24]. If we consider tags on the right of the word being tagged, our model is not a Markov process anymore and Viterbi cannot be applied [24]. This alone is a strong reason to research with evolutionary based taggers.

Having computed the occurrence probability of the tag trigram and the probability of occurrence of the word-tag and assigning all possible tag sequences to all the input words, the solution is obtained when we can maximize the product value of them as stated in (2).

$$\text{fitness}(S) = \prod_{i=1}^{n} \Pr(w_i|t_j) \times \Pr(t_i|t_{i-1}, t_{i+1}) \tag{2}$$

A particular sequence $t_{i-1}t_it_{i+1}$ may not be listed in the training table, either because its probability is approximately zero (if the sequence of tags is forbidden for some reasons) or, most likely, because there are insufficient statistics, unless a very large training text is used. Since the HSTAGger suffers from sparse data, and the probability estimates of low frequency events lead to inaccurate estimations and to avoid null probability for those transitions a compensation function using unigram and bigrams is employed to calculate the probabilities of transitions as shown below:

$$\text{fitness}(S) = \sum_{i=1}^{n} \log(f(w_i))$$

$$f(w_i) = (\Psi(\text{transition}(w_i)))^{\alpha} \times \left(\frac{\text{emission}(w_i)}{\Pr(t_i)}\right)^{\beta}$$

$$\Psi(z) = \begin{cases} \gamma & z < \text{trans}_{\min} \\ \text{trans}(w_i) & \textit{otherwise} \end{cases}$$

$$\text{transition}\,(w_i) = \Pr(t_i|t_{i-1}, t_{i+1}) = \frac{\text{freq}(t_{i-1}, t_i, t_{i+1})}{\sum_{t' \in T} \text{freq}(t_{i-1}, t', t_{i+1})}$$

$$\text{emission}\,(w_i) = \Pr(w_i|t_j) = \frac{\text{freq}(w_i, t_j)}{\sum \text{freq}(t_i)}, \quad \forall j = 1, 2, \ldots, k$$

$$\text{modified} - \text{emission} = \frac{\text{emission}\,(w_i)}{\Pr(t_i)}$$

$$\gamma = \lambda_1 \Pr(t_i|t_{i+1}) + \lambda_2 \Pr(t_i|t_{i-1}) + \lambda_3 \Pr(t_i),$$

where $\Pr(t_i|t_{i+1})$ is the transition probability, $P(w_i|t_i)$ is the emission probability of the ith word, count (t_{i-1}, t_i, t_{i+1}) is the number of occurrence of the list of tags (t_{i-1}, t_i, t_{i+1}) in the training table, T is the set of all possible tags in this list, k_i is the number of valid tags for w_i, γ represents the linear interpolation, $0 < \lambda_1, \lambda_2, \lambda_3 < 0, n$ is the number of words in the sentence and $\text{trans}_{\min}$ is a threshold value as a constraint to compensate sequences that are listed less than $\text{trans}_{\min}$ times in all the training table. The parameters α and β are weights and are set at 0.6 and 0.4, respectively from the earlier experiments, which means that transition probability is more important than modified emission probability.

4.3. BEETAGger: bee colony based tagging algorithm

In this section we introduce an algorithm based on the BCO method to address the PoS-tagging problem which is referred to as BEETAGger. The algorithm consists of an initialization step followed by forward and backward passes as done in the BCO algorithm. However, these steps are adopted carefully to the tagging process. In the following subsections we begin by describing the representation we have used, and then we will discuss adaptations of operations and the evaluation metrics used to guide the bees through the optimization process.

4.3.1. Representation of solutions

We now turn to describing the representation of solutions to effectively apply the BCO operations. Recall that the input given to the algorithm is a sentence of n words in which ith word can take k_i different tags as its syntactic category. We decompose the PoS-tagging problem into the n stages where ith stage represent the i word in the sentence. The nodes at each stage correspond to the valid tags taken from a dictionary. That is, the number of nodes at each stage is equal to the number of tags for the word assigned to that stage. In order to handle words that are neither in the lexicon nor in the training data, we take all the possible PoS-tags as candidate tags. At each stage, every bee must elect one node out of all nodes as the tags of the word assigned to that stage.

In this model the value of each node is represented by an integer, which is the index of the related tag. For instance, consider the sentence in Fig. 1, extracted from the Brown corpus. Tags for the words in a sentence extracted from the Brown corpus.

An example of representation of solutions on a sample sentence of Fig. 1 is reported in Fig. 2. Fig. 2 shows the set of stages and also the nodes visited in each stage.

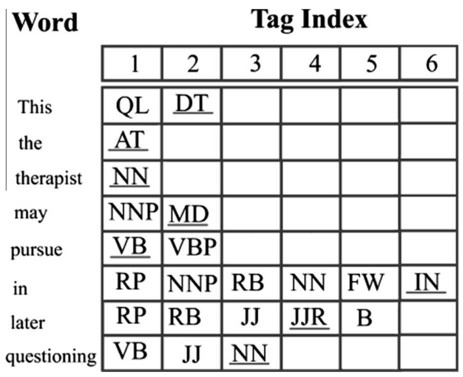
4.3.2. Constructive moves in forward pass

At every stage, each bee first visits just one node according to fragrance (weight) of nodes. We assume in this paper that the weight of nodes is mutually unequal and proportional to their frequencies which makes nodes to be not equally interesting for bees. The weight denotes the Fragrance value of stage i at node j for the bee, which is computed follows:

$$\text{fragrance}_i^j = \frac{\sum \text{freq}(s_i, n_j)}{\sum \text{freq}(s_i)} \tag{3}$$

where $\text{freq}(s_i, n_j)$ denotes the number of times the ith word (stage i) occurs with its jth tag (node j). After sticking to a node, each bee generates the partial solution based on the node it chooses and collects the nectar according to the weight of the selected node. Finally, each bee returns into the hive.

We note that following the idea of TBL algorithm [46], it is more efficient to choose the most probable tag, which has also the highest frequency for that word in the lexicon. In this paper during each stage, the node selection step is modified. In our method the jth node in the ith stage is chosen randomly by:



Word	Tag Index 1	2	3	4	5	6
This	QL	DT				
the	AT					
therapist	NN					
may	NNP	MD				
pursue	VB	VBP				
in	RP	NNP	RB	NN	FW	IN
later	RP	RB	JJ	JJR	B	
questioning	VB	JJ	NN			

Fig. 1. Different applicable tag for the words in a sentence "This the therapist may pursue in later questioning" [4]. Underlined tags are the correct ones, according to the Brown corpus [4].

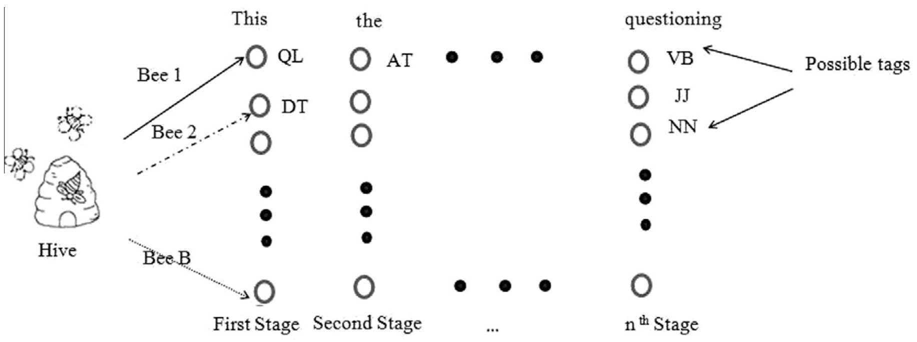


Fig. 2. Tags for the words in a sentence extracted from the Brown corpus [4].

$$p_i^j = \Pr(\text{node j being selected as a tag for the word in ith stage}) = \frac{\text{fragrance}_i^j}{\sum_{m=1}^{k_i}\text{fragrance}_i^m} \tag{4}$$

where p_i^j is the probability of selecting node j at stage i by bees and k_i is the maximum number of nodes in stage i.

According to Eq. (3) the most probable tag has the better chance to be selected. Furthermore, by increasing the weight of a particular node, the probability of the preferred solution increases proportionally. This leads to speeding -up the convergence of the algorithm in contrast to selecting tags randomly.

Let us also assume that each constructive move from one node to the next node in each forward pass has the certain weight for the bees while flying along it. The quality of each move is proportional to the partial function, called *pollen* of that move. The quality of gathered pollen along each move is computed as follow:

$$\text{pollen}(j, j+1) = \frac{\text{freq}(j, j+1)}{\sum_{j' \in T}\text{freq}(j, j\prime)} \tag{5}$$

where freq$(j, j+1)$ is the number of occurrences of the list of nodes $j, j+1$ in the training table and T is the set of all possible nodes in this list.

The parameters Fragrance and Pollen are two strong indictors to be used in validating tags. Hence, flexible merging of these parameters seems to be sufficient to asses nectar quantity which can be utilized in automatic tagging. These features motivate us to combine these parameters in a single weight score. In particular, we use the harmonic mean of these parameters as a single score and call it *convenience* of node. As it will be discusses later, the experiments we have conducted have demonstrated that the accuracy of tagging algorithms always receives noticeably more value when the assigned weight to Pollen is higher than Fragrance. Therefore, we treat these parameters with unequal weights when computing the convenience. In our combination, Fragrance and Pollen of nodes contribute differently to better capture the importance of the parameters. This novel combination departs from what has been used in the literature as summarized in Table 3 and significantly improves the performance of the proposed algorithm as we will show in our experiments. We have formulated the nectar quantity gathered by kth bee by choosing node j, located in stage i as follows:

$$\text{convinience}(s_{ij}) = \frac{\alpha_{frag} + \alpha_{poll}}{\frac{\alpha_{frag}}{\text{fragrance}(s_{ij})} + \frac{\alpha_{frag}}{\text{pollen}_k^j}} \tag{6}$$

where α_{Frag} is the weight of Fragrance measure, and α_{Poll} is the weight of Pollen measure. In this algorithm α_{Frag} and α_{Poll} are set to four and one, respectively, based on previous experiments. This means that Fragrances probability is more important than pollen probability given the relatively free word order of English.

4.3.3. Bees partial solutions comparison mechanism

After all bees generate their own partial solution, they return to the hive to participate in a collective decision making process. In particular, each be compares all the generated solutions and needs to make a decision to either abandon the partial solution or expand it in the next forward pass. To this end, we need to equip bees with a mechanism to able to evaluate the partial solution of other bees. The following measure, which is simply the sum of the nectar quantity of the partial solution, has been introduced to be used by bees for evaluations:

$$\mathrm{Nect}_b(u,z) = \sum_{m=1}^{u} \log(\mathrm{convinience}(s_m)), \quad b = 1,2,\ldots,B \tag{7}$$

where $\mathrm{Nect}_b(u,z)$ is the sum of nectar generated by the ith bee, b is the bees number, u is the stage number and z is in iteration number. Depending on the quantity of the gathered nectar, every bee possesses certain level of loyalty to the tag sequence previously discovered.

Bees use approximate reasoning, and compare their discovered partial solutions with the best as well as the worst discovered partial solution from the start of the search process. In this way, "historical facts" discovered by all members of the bee colony have significant influence on the future search directions. The probability that, bth bee (at the beginning of a new forward pass at stage $u+1$ at iteration z) is loyal to the previously discovered partial solution is calculated in the following way:

$$p_b(u+1,z) = \exp\left(-\frac{\mathrm{Norm}-\mathrm{Nect}_{\max}(u,z) - \mathrm{Norm}-\mathrm{Nect}_b(u,z)}{uz}\right), \quad b = 1,2,\ldots,B \tag{8}$$

where u is the ordinary number of the forward pass, $u = 1,2,\ldots,U$ and $\mathrm{Norm}-\mathrm{Nect}(\cdot,\cdot)$ represents normalized value of the quantity nectar value of the partial solution discovered by the bth bee.

$$\mathrm{Norm}-\mathrm{Nect}_b(u,z) = \frac{\mathrm{Nect}_{\max}(u,z) - \mathrm{Nect}_b(u,z)}{\mathrm{Nect}_{\max}(u,z) - \mathrm{Nect}_{\min}(u,z)}, \quad b = 1,2,\ldots,B \tag{9}$$

where $\mathrm{Nect}_b(\cdot,\cdot)$ shows the nectar values of the partial solution discovered by the bth bee, and $\mathrm{Nect}_{\max}$ and $\mathrm{Nect}_{\min}$ are the nectar values of the best and the worst discovered partial solution from the beginning of the search process, respectively.

Using Eq. (8) and a random number generator, every artificial bee decides to become an uncommitted follower, or to continue flight along already known path. The better generated partial solution (higher $\mathrm{Nect}_b(\cdot,\cdot)$ value) the higher the probability that the bee will be loyal to the previously discovered partial solution. The greater the ordinary number of the forward pass, the higher the influence of the already discovered partial solution. This is expressed by the term u in the nominator of the exponent.

We would like to elaborate Eq. (8) a little more. As we can see, when a bee finds a tag sequence which is a high fitness value, it sticks to that solution with probability one. In particular, the smaller the fitness value, the smaller the probability that the bee flies along the same tag would be. This feature gives the bees more freedom in early stages to explore the solution space, while more forward passes they make, the less courage to explore the solution space (i.e., the bess tend to exploit the solution found so far instead of exploring the solution space). Therefore, Eq. (8) balances the exploration and exploitation based on the stage the bees are as desired.

4.3.4. Recruiting process

We now turn to explaining the recruiting process. As mentioned in the previous step, each bee makes decision to either abandon the already known solution or expand it. If the decision of bee at the beginning of a new stage is to expand the solution obtained so far, it will go into the dancing area to follow another beee. The dancing area in intended to facilitate

Table 3
Different convenience measures used in the literature to evaluate the tagging.

References	Convenience ($s_{i,j}$)
[21,23,24,45]	$\Pr(t_i \vert t_{i-1}, t_{i+1})$
[47,48]	$\Pr(w_i \vert t_i) \times \Pr(t_i \vert t_{i-1}, t_{i+1})$
[49]	$\Pr(w_i \vert t_i)^{\alpha} \times \Pr(t_i \vert t_{i-1}, t_{i+1})^{\beta}$
[3,48]	$\Pr(t_i \vert w_i) \times \Pr(t_i \vert t_{i-1}, t_{i+1})$
[49]	$\Pr(t_i \vert w_i)^{\alpha} \times \Pr(t_i \vert t_{i-1}, t_{i+1})^{\beta}$
[49]	$\frac{\Pr(t_i \vert w_i)}{\Pr(t_i)} \times \Pr(t_i \vert t_{i-1} t_{i+1})$
The present work	$\dfrac{\alpha_{frag} + \alpha_{poll}}{\frac{\alpha_{frag}}{\Pr(t_i \vert t_{i-1}, t_{i+1})} + \frac{\alpha_{poll}}{\Pr(t_i \vert w_i)}}$

the cooperation between the bees in the colony. The bee-dancers (recruiters) try to *advertise* different partial solutions. To this end, we need to equip uncommitted bees (i.e., who decide to choose the new partial solution) with a mechanism to choose a advertiser bee and utilize its solution. The probability of selecting a particular advertiser bee is computed by:

$$p_b = \frac{\text{Nect}_b}{\sum_{k=1}^{RC} \text{Nect}_k}, \quad b = 1, 2, \ldots, RC \tag{10}$$

where Nect_k is the nectar quantity value of the kth advertised solution and RC is the number of recruiters. Using Eq. (10) and a random number generator, every uncommitted follower joins one bee dancer (recruiter). At the end of this path all bees are free to independently search the solution space and generate the next constructive iteration moves.

5. Experiments, analyses and comparison

5.1. Used corpus

To demonstrate the merits of proposed algorithm, we have selected two data sets with different sizes for English language to conduct our experiments. Both data sets are sampled from one of the most popular corpora named Brown tagged corpus. Brown corpus contains one million words of written American-English texts published in the US in 1961. Approximately, 40% of the words are ambitious which makes the automatic tagging a challenging problem.

The first data set (DS1), which has a medium size, consists of 75 texts, including 38 texts from books and 37 texts from periodicals. The second data set (DS2), which has a large size, contains different categories such as natural, social and behavioral sciences. Table 4 shows the data statistics of these data sets. We have reduced the tag set of Brown corpus into a tag set with 47 different tags. Table 6 and Fig. 3 show some of the tags and their corresponding frequencies in the training corpus of DS1. Studying Table 5 carefully reveals that the tag NN (noun) is the most frequent tag in the training corpus of DS1. On the other hand, the tag WQL (WH-qualifier) with only 37 occurrences is the least frequent tag.After recreating the corpus with 47 distinct tags, the data sets have been split into 90% “training” and 10% “test” sets. We use 10-fold tests in which 10 different training and test sets with the size of 90% and 10 % of corpus, respectively, are selected to test the taggers. The results of each tagger is compared to tagged sentences in the test corpus. Each tagger is allowed to assign exactly one tag to each word in the sentence. The measure we have used to asses the performance of tagging algorithms is the accuracy, which measures ratio of correct tags (i.e., the proportion of correctly assigned tags to the total number of tokens in the processed corpus) as defined below:

$$\text{Accuracy} = \frac{\#\text{ of correctly tagged words}}{\#\text{ of overall words}} \tag{11}$$

5.2. Results and comparison

The proposed algorithms are compared with previously proposed algorithms in the literature including Genetic algorithm-based (GA) [21,23,24,45], Simulated Annealing-based (SA) [24] and CHC taggers [24]. The results reported in [24] have shown that GA base taggers obtain the same accuracy as Viterbi [38] which has been used for stochastic tagging.

To be fair in comparing different algorithms, we need a mechanism to terminate each algorithm appropriately. In particular, the termination rule we used in our experiments is as follows. We record the current optimal solution and its iteration for the algorithm. Then, we count the number iterations proceeds from the iteration number of current optimal solution without any improvement. If the ratio of these iterations to the total number of passed iterations exceeds the given upper bound ratio, we terminate the algorithm. This means that contenting the algorithms will not result in significant improvement of the optimal solution. Obviously, there is a maximum number of iterations that guarantees the algorithm will not proceed forever.

We would like to emphasize at this point that the results shown in the rest of the paper are the average over 10 runs of the algorithms, in order to avoid accidental results and to make a fair comparison to be used for drawing conclusions. In addition, to lubricate the comparisons, the algorithms are iterated 500 times in each run since the 500 generations are enough to make converge all of the algorithms. The performance comparison between our proposed (i.e, BEETAGger and HSTAGger) taggers with other state-of-the-art evolutionary based taggers (i.e., GA-based, CHC, and SA-based) is demon-

Table 4
Data statistics.

Data set	DS1	DS2
# Of documents	75	155
# Of sentences	2346	6850
# Of tokens	165489	685900

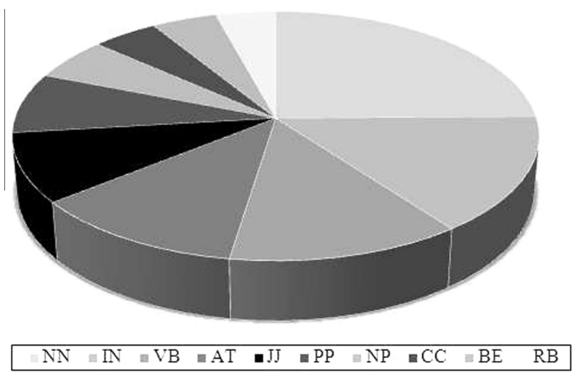


Fig. 3. The percentage of different tags extracted from Brown corpus.

Table 5
Sample of tag set for DS1 data set extracted from Brown corpus.

Tag	Description	# Of occurrence
NN	Noun, singular, common	31464
IN	Preposition	19080
VB	Verb	16031
AT	Article	14921
JJ	Adjective	11367
pp	Determiner	9968
NP	Noun, singular, proper	7052
CC	Conjunction	5896
BE	Verb “to be”	5808
RB	Adverb	5380

Table 6
Result of difference word context of BEETAGger.

Word context	Accuracy (%)
3 word before (1-1-1-w-0)	94.49
2 word before (1-1-w-0)	94.16
1 word before (1-w-0)	95.64
1 word after (0-w-1)	94.7
2 word after (0-w-1-1)	94.25
3 word after (0-w-1-1-1)	94.0
1 word before and 1 word after (1-w-1)	96.68

strated in Fig. 4. It can be inferred from the results of Fig. 4 that BEETAGer algorithm outperforms the other evolutionary based taggers significantly in all data sets.

The obtained results were very competitive when compared with the results of SA [24]. As shown in Fig. 4, SA provides worse results than any of the evolutionary algorithms. This proves the advantage of the evolutionary approach. The results obtained for SA are very poor, indicating that SA is not able to solve Pos-tagging problem adequately.

The performance of taggers depends on not only the specific parameters of each tagger but also some general factors such as the language, the used corpus (for training and testing), the size of tagset and also the size of training and testing sets [50]. Different corpora and even different subsets of a unique corpus may generate different results due to features such as word patterns they contain or the percentage of unknown words or the word ambiguity ratio they have. For example experiments show that the results of testing taggers on Wall Street Journal corpus is usually better than testing the same tagger on Brown corpus [4]. So although there are interesting results announced on Wall Street Journal (WSJ) such as SVM (97.2%) citeref17, the traditional Brill (97.0%) and GA (96.63%) [24], we compare our results with those who have reported their work on Brown corpus or those whose works are re-generated by us in our test environment. In other words, for comparing the performance of different taggers we should test them in the same environment or rely on the results achieved in situations close to ours.

Even though a great number of researchers have dedicated their efforts to developing or improving POS taggers in the last few years, the state-of-the-art performance of a single POS tagger (96–97%) still leaves a little room for improvement [51].

Hidden Markov Model (HMM) [8] and Maximum Entropy (MXPOST) [52] taggers are the well-known statistical tagging methods and Brills model taggers is a conventional transformation-based learning (TBL) method [23]. We also define a

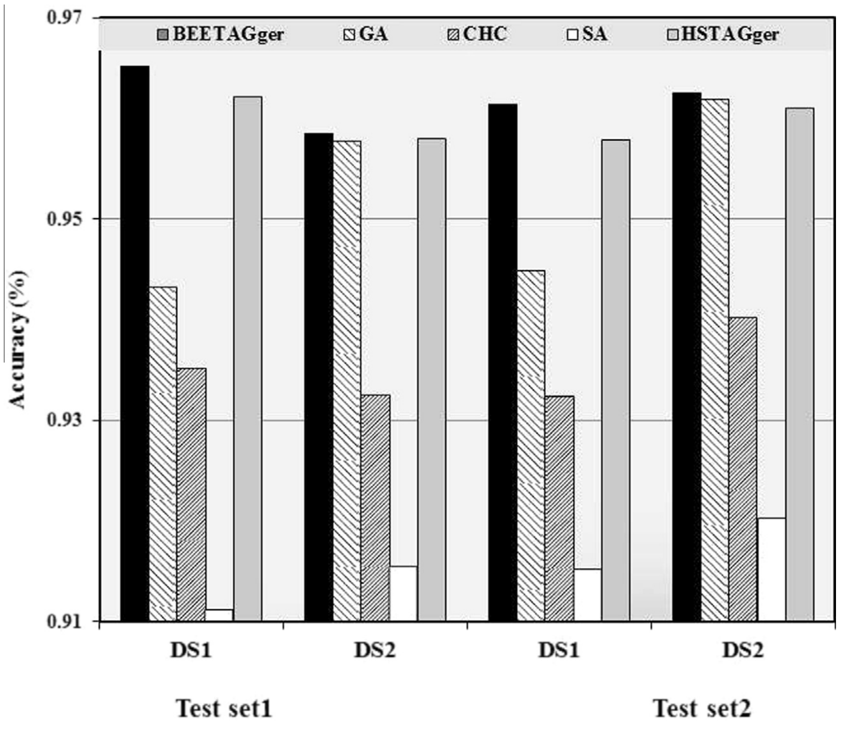


Fig. 4. Comparison of the performance of the evolutionary based taggers.

random and the most frequent tag (MFT) baseline on the Brown corpus. The random baseline is calculated by completely randomly picking one of the tags of each word and it also represents the amount of ambiguity in the corpus. The MFT baseline simply selects the most frequent PoS-tag of each word from the used corpus. If the target word does not exist in the training set, the MFT baseline randomly picks one of the possible tags of the unknown word.

Fig. 5 presents a comparison between HMM, MXPOST, TBL and our algorithms as well as two standard baselines. As shown in Fig. 5, we have achieved some improvements in terms of accuracy. In this way, the heuristic nature of harmony search is shown to be useful for tagging where traditional algorithms have low accuracy. As there are very few parameters to be tuned in the proposed algorithms, they are very easy to use. In addition, the behavior of them is very flexible, allowing the size of the context including word and PoS N-gram, to be defined as bigrams or trigram or even higher N-gram. Some classical approaches cannot be applied with two more complex contexts because they are designed to search the data sequence which maximizes the observed data according to a Markov model, i.e., a model in which the current state only depends on the previous one. If we consider tags on the right hand of the word being tagged, our model is not a Markov process anymore. This is a strong reason to further researches with evolutionary based approaches [24].

Another key innovation of the proposed algorithms is the ability of tuning free parameters within the metric in order to optimize them for various languages. It has been successfully applied to English without a priori knowledge other than a tagged corpus. Compared to state-of-the-art PoS taggers reported up to date in the Brown corpus, it exhibits a very interesting accuracy (over 96. 65% for English on Brown corpus) in the reasonable time. The final advantage is that the greedy search used by the other algorithms can be avoided in the proposed taggers. In each generation of the proposed algorithms a number of possible solutions to the tagging problem are tried. There is the possibility of finding a solution with less effort than a greedy search.

5.3. Comparative analysis of our proposed algorithms

In this section we present a number of experiments to compare the performance of the proposed algorithms. The experiments in this section focus on the comparison of the convergence behavior of the proposed algorithms.

5.3.1. The size of the training set

Increasing the size of the corpus being used affects the performance of the proposed taggers in some aspects. Small sized corpora cannot completely represent the language they are presenting. Therefore, increasing the size of corpus improves the quality of statistical data in the lexicon which is used in evaluation mechanism and consequently improves the performance of the tagger. On the other side, by using a larger corpus size, the number of unknown words found in the test set is reduced.

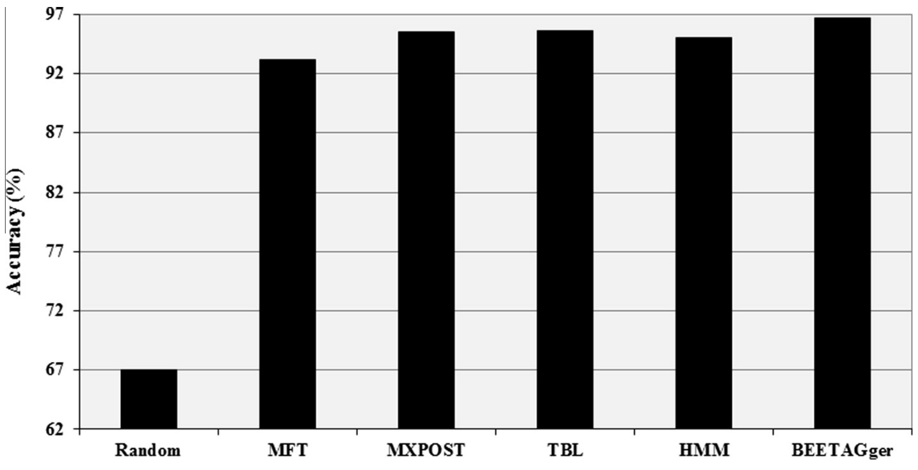


Fig. 5. Results of different methods.

This allows us to find more examples of infrequent words. Fig. 6 shows the occurrence of unknown words versus the size of the training set. As the figure shows, the percentage of these words decreases almost exponentially as the size of the training data grows. To show the effect of increasing the size of the used corpus on the performance of the proposed taggers, we have trained proposed taggers on increasing amounts of training data from DS2 of Brown corpus. Fig. 7 shows the learning curve of the proposed taggers, i.e., the accuracy depending on the size of the training data.

The taggers outputs have comparable accuracy growth curves, although the initial growth is much higher for BEETAGger. The curves for BEETAGger and HSTAGger appear to be leveling out towards the right end of the graph. However, it would seem that the accuracy level at the optimal size for that context (around 225*K* words), is a good approximation of the eventual ceiling and it will be saturated further than this size. It is significant that the accuracy of the proposed taggers is very high even with small amount of training data. As we will see, the increase in training set size indeed results in better performance. This phenomenon becomes stronger in the BEETAGger where the accuracy of the estimation (which is used in both selection and comparison mechanism of the algorithm) is proportional to the word and the tag frequency of occurrence in the training text. Thus, similar to the all statistical taggers a larger training text can be used in order to minimize the error rate of taggers. It can be inferred from this diagram that by increasing the size of training corpus the difference between accuracy values becomes smaller. In this situation, the accuracy of them may be even reduced. Because it must be taken into account that when tagging a sentence that each of its words requires one of its frequent tags, appearing in a frequent context, accuracy will be increased by increasing the size of training data. However, when tagging a sentence including words that adopt some of their most rare tags, increasing the size of training data can degrade the results.

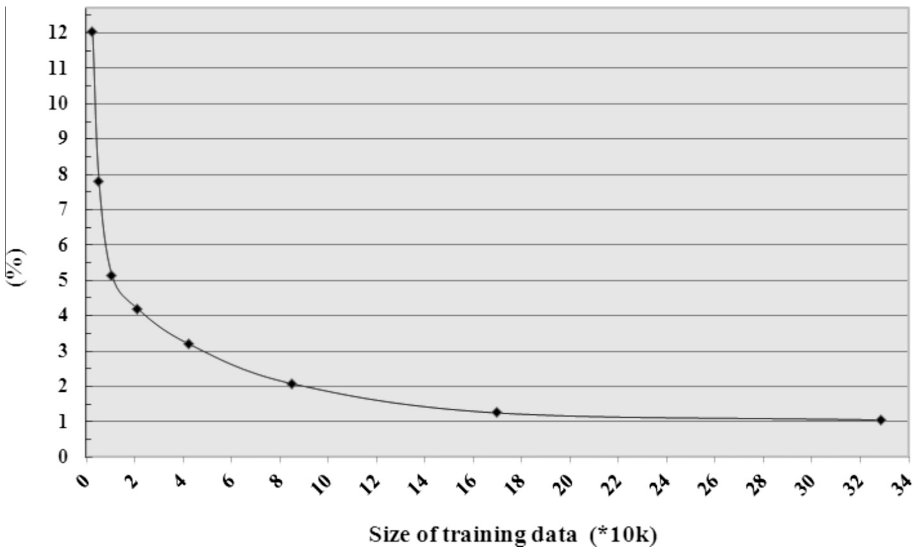


Fig. 6. Percentage of unknown words for various sizes of the training data.

Though, it may intuitively seem apparent that the larger the training corpus, the better, we must take into account that only a significant increase of the accuracy can justify the increase of the training corpus. In addition, it should be considered that when the maximum accuracy for that context is reached, further increasing of the size does not improve the algorithm any more. On the other hand, the increased amount of data also increases time and space requirements.

5.3.2. Convergence behavior

Proposed taggers run with a number of generations, and in each generation a number of possible solutions to the tagging problem are tried. Our objective is to identify the correct PoS-tag for each word. However, an important question here is how to determine the number of iterations that give the best result. The convergence behavior of the proposed taggers is demonstrated in Fig. 8. Fig. 8 shows clearly that the value of accuracy to be commensurate with number of iteration. Each of the algorithms reaches a near optimal solution but the result that is obtained by the BEETAGger is better than the results of the HSTAGger algorithm. With the same features, BEETAGger algorithm is better than HSTAGger, but the difference is not much, less than 1% in accuracy. The performed experiments reveal that BEETAGger can obtain the optimal solutions faster than HSTAGger and it converts to the good solution faster. In other words, the BEETAGger was able to produce optimal solutions in a reasonable computation time. The preliminary results obtained are considered to be very good.

Our experimental results show that if a constant number of iteration is specified for all sentences, the optimum solution may be found at a relatively early generation; nevertheless, many redundant computations will be performed. However, according to the experimental results, the convergence of proposed algorithms to an optimal solution depends on the length of solution (sentence), which is the number of its decision variables. Due to that we introduce a new termination criterion. When the difference between the best and the worst solution in each run is lower than a pre-specified convergence criterion, the algorithms are said to have reached to the expected convergence.

Furthermore, because the best tagging in BEETAGger algorithms can be accomplished with a few iteration steps, the best tagging can be reached with small populations and just a few generations. Experiments show that although HSTAGger works better than most of the other systems in our study but BEETAGger has better results yet. In the rest of this section we study the impact of various parameters on the performance of BEETAGger.

5.4. Empirical study of the impact of different parameters on the performance of BEETAGger

In this section we present a number of comprehensive experiments to evaluate the performance of BEETAGger in many aspects. First of all, we analyze the effect of the size and shape of the context used for filling the contextual table. Then the effect of changing the BEETAGger parameters and the parameter of its evaluation function are studied. By studying the effect of the main factors of BEETAGger on its performance the optimal setting of used parameters is determined.

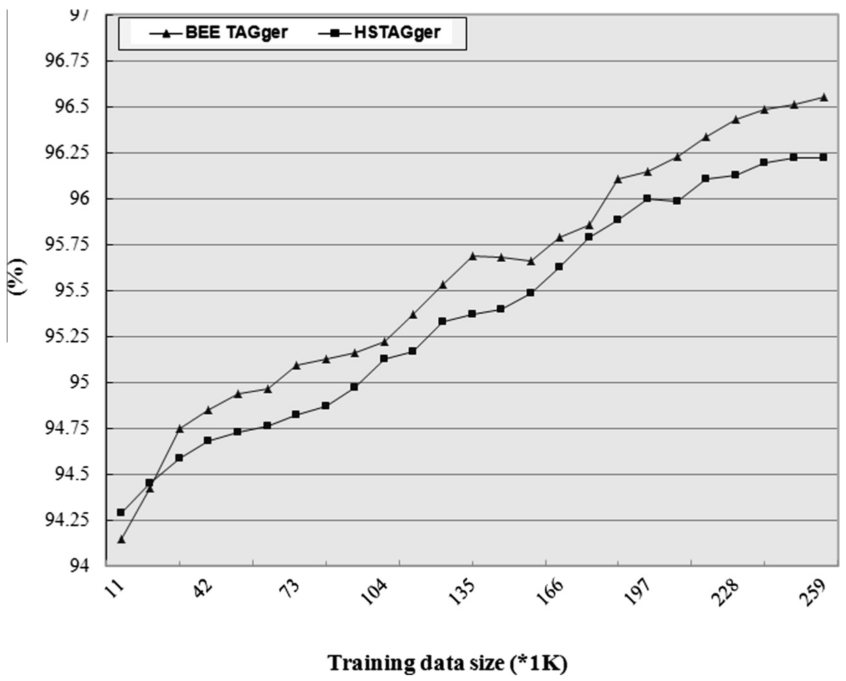


Fig. 7. The accuracy of proposed taggers on DS2 as a function of the number of tokens of training material.

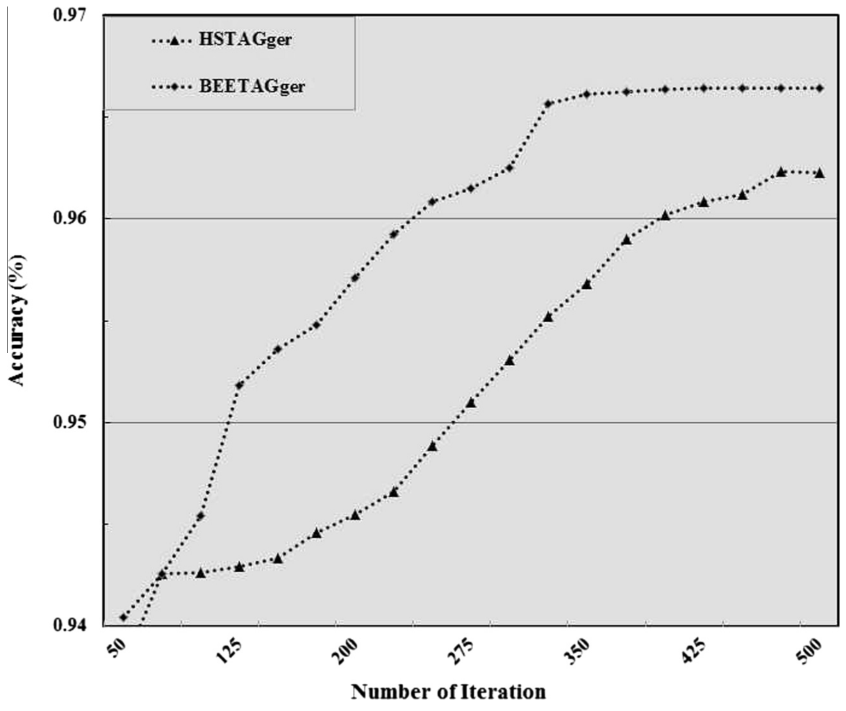


Fig. 8. The comparison of the proposed algorithms.

5.4.1. The size of word context sliding window

In our definition, we used the notion of N-gram by putting a sliding window on words of the sentences to determine the number of constructive moves in each forward pass. From the hypothesis better use of context will improve the accuracy, we set up an experiment to find out which context is best suitable for BEETAGger. To consider the impact of the length of contexts on the performance of BEETAGger, we have conducted an experiment in which the window size is modified from one up to six and computed the accuracy of BEETAGger independently for each N-gram. In this experiment, we only use the word context surrounding the current word. Results are depicted in Table 6.

It can be inferred from this table, that a window of size one, which considers only the tag before or after the current word does not hold enough information to make a good prediction. Another outstanding feature of the results is that considering the left hand side contexts such as (1-w-0) leads to a slightly better performance in comparison with considering the right hand sides. This confirms that the models, which pay more attention to the previous word, may represent the structure of language better. On this dataset, we found that if we take one word before and one word after the current word, the result is higher than taking more preceding and succeeding words.

Although it may be considered that larger context window provide more information to BEETAGger, it causes the number of occurrences for many tag sequences, appearing in the contextual table, to become less frequent. Therefore, there will not be enough statistical data for those tag sequences which result in predicting the tag of words based on weaker evidences.

5.4.2. Empirical study of BEETAGger parameters

In this subsection, the accuracy of the BEETAGger is investigated under different number of artificial bees. The results of the performed tests to demonstrate the effect of changing the number of bees on the accuracy of the BEETAGger on DS2 is shown in Table 7. Table 5 shows that a small number of bees is adequate to achieve high value of accuracy. This is due to the fact that in this problem the sentences are tagged one by one and therefore a small number of bees is usually enough to depict the variety of possible tags. On the other hand, smaller number of bees leads to quicker algorithm with lower space requirements. As larger numbers of bees are more time consuming to find the optimal solution in BEETAGger, they are ignored.

Another observation is the correlation between the number of the bees and the complexity of the sentence to be tagged. When the sentence is more complicated, the number of its words that has more than one possible tag increases. As a result, a larger number of bees is required to quickly search for the correct tagging. Therefore, it seems reasonable to define the number of bees as a linear function of the length of the sentence to be tagged. Our results reveal that setting the number of bees as three times the length of the sentence ($3 \times |S|$) leads to higher accuracy in comparison with others.

Table 7
Impacts of the number of bees on the performance of BEETAGger.

Number of artificial bees	Accuracy (%)
5	94.8
20	95.74
30	95.74
$\lvert s\rvert$	95.15
$2 \times \lvert s\rvert$	95.95
$3 \times \lvert s\rvert$	96.65
8	94.92
10	94.9
12	95.25

Table 8
Accuracy of BEETAGger for different fitness functions applied on DS1.

References	Functions to be tested	Accuracy (%)
[21,23,24,45]	$F(s(n)) = \sum_{i=1}^{n} \log(\Pr(t_i \mid t_{i-1}, t_{i+1}))$	91.98
[47,48]	$F(s(n)) = \sum_{i=1}^{n} \log(\Pr(w_i \mid t_i) \times \Pr(t_i \mid t_{i-1}, t_{i+1}))$	92.12
[49]	$F(s(n)) = \sum_{i=1}^{n} \log\left(\Pr(w_i \mid t_i)^{\alpha} \times \Pr(t_i \mid t_{i-1}, t_{i+1})^{\beta}\right)$	95.18
[3,48]	$F(s(n)) = \sum_{i=1}^{n} \log(\Pr(t_i \mid w_i) \times \Pr(t_i \mid t_{i-1}, t_{i+1}))$	95.28
[49]	$F(s(n)) = \sum_{i=1}^{n} \log\left(\Pr(t_i \mid w_i)^{\alpha} \times \Pr(t_i \mid t_{i-1}, t_{i+1})^{\beta}\right)$	95.89
[49]	$F(s(n)) = \sum_{i=1}^{n} \log\left(\frac{\Pr(t_i \mid w_i)}{\Pr(t_i)} \times \Pr(t_i \mid t_{i-1} t_{i+1})\right)$	96.20
The proposed method	$F(s(n)) = \sum_{i=1}^{n} \log\left(\frac{\alpha_{frag} + \alpha_{poll}}{\frac{\alpha_{frag}}{\Pr(t_i \mid t_{i-1}, t_{i+1})} + \frac{\alpha_{poll}}{\Pr(t_i \mid w_i)}}\right)$	96.65

5.4.3. The impact of different functions to compare the partial solution in the BEETAGger

In this subsection various functions are investigated. We have tested these functions and evaluated the results which are shown in Table 8. The functions used in ([23]) has not considered the lexical probability, which means that the dependency of the word on its tag has been ignored. As our experiments show using this equation has not led to desired accuracy in BEETAGger, we believe that the lack of lexical probability within evaluation function is a significant drawback. In addition, by comparing the function of [3,47], it has been obtained that the equation of [3] led to higher performance. This is due to the fact that function of [47] gives more priority to common tags, while a data set may contain both common and unusual tags with any probability. On the other side, because ti is very larger than w_i, $\Pr(t_i)$ in equation of [47] becomes extremely small and difficult to be estimated. As Equation of [49] is equivalent to including the factor $\Pr(t_i)$ in the equation of [3] and no significant improvement of tagging performance was obtained when equation of [49] was used, we prefer to propose evolutionary tagging model based on equation of [3]. The proposed evolutionary tagging model is a modification of the standard equation of [3] which improves the lexical and trigram tag transition probabilities. As we can see in Table 8, by using the proposed function, the performance of the BEETAGger increases by 4%–5%.

It has already been demonstrated by several evaluations of metrics that trigram tag transition strongly correlates with the quality of taggers, and it is thus an extremely important feature component in BEETAGger. On the other hand, our results demonstrate that to some extent the lexical probability is an important metric in comparing the extracted solutions. Hence, we assign different weights to lexicon and trigram transition parameters, corresponding to their importance. The effect of this modification on lexicon parameter is that the probabilities of common word-tags are decreased, by means of that making it easier for BEETAGger to choose lexically less probable tags. Therefore, proposed function was finally selected after extensive experiments using the described seven equations.

6. Conclusions

This paper proposed schemes to design evolutionary algorithms devoted to tasks concerning statistical PoS-tagging. The statistical measurements extracted by the stochastic approaches of PoS-tagging provide an appropriate fitness function for the evolutionary algorithms in a natural way. In this paper the problem of assigning lexical categories to words is studied and novel algorithms based on harmony search and bee colony optimization named HSTAGger and BEETAGger, respectively, are proposed. The proposed algorithms are designed by modeling tagging problem as an optimization of an objective function. The evaluation of individuals is based on a training table comprised of contexts extracted from an annotated corpus. In situations in which there is no appropriate tagged corpus available for training of proposed taggers, by generalizing the tags for each word, the most suitable tag can be predicted, when sufficient information is gathered from the text to be tagged. An evolutionary algorithm can be used as the search algorithm for PoS-tagging and can be applied to any statistical model, even if they do not rely on the Markov assumption which is required in classic search algorithms for tagging, such as the widely used Viterbi [38]. Our experimental results on different data sets show that the proposed algorithms produce better solutions with high quality considering accuracy measures in comparison with other known algorithms.

References

[1] C.D. Manning, H. Schütze, Foundations of Statistical Natural Language Processing, MIT press, 1999.
[2] T. Gungor, Handbook of Natural Language Processing, vol. 2, Chapman and Hall/CRC, 2010. Ch. 10: Part-of-Speech Tagging.
[3] S.J. DeRose, Grammatical category disambiguation by statistical optimization, Comput. Linguist. 14 (1) (1988) 31–39.
[4] W.N. Francis, H. Kučera, Manual of information to accompany a standard corpus of presentday edited American English, for use with digital computers, Brown University, Department of Lingustics, 1979.
[5] Pos tagging (state of the art) (May 2013). URL http://aclweb.org/aclwiki/index.php?title=POS_Tagging%28State_of_the_art%29.
[6] S. Tasharofi, F. Raja, F. Oroumchian, M. Rahgozar, Evaluation of statistical part of speech tagging of persian text, in: 9th International Symposium on Signal Processing and Its Applications, 2007, ISSPA 2007, IEEE, 2007, pp. 1–4.
[7] E. Brill, Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging, Comput. Linguist. 21 (4) (1995) 543–565.
[8] E. Charniak, Statistical Language Learning, The MIT press, 1996.
[9] F. Al Shamsi, A. Guessoum, A hidden markov model-based pos tagger for arabic, in: Proceeding of the 8th International Conference on the Statistical Analysis of Textual Data, France, 2006, pp. 31–42.
[10] T. Brants, Tnt: a statistical part-of-speech tagger, in: Proceedings of the Sixth Conference on Applied Natural Language Processing, Association for Computational Linguistics, 2000, pp. 224–231.
[11] L. Araujo, Part-of-speech tagging with evolutionary algorithms, in: Computational Linguistics and Intelligent Text Processing, Springer, 2002, pp. 230–239.
[12] S.-Z. Lee, J.-I. Tsujii, H.-C. Rim, Part-of-speech tagging based on hidden markov model assuming joint independence, in: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics, Association for Computational Linguistics, 2000, pp. 263–269.
[13] R. Sarikaya, M. Afify, Y. Deng, H. Erdogan, Y. Gao, Joint morphological-lexical language modeling for processing morphologically rich languages with application to dialectal arabic, IEEE Trans. Audio Speech Lang. Process. 16 (7) (2008) 1330–1339.
[14] A. Ratnaparkhi, et al., A maximum entropy model for part-of-speech tagging, in: Proceedings of the conference on empirical methods in natural language processing, vol. 1, Philadelphia, PA, 1996, pp. 133–142.
[15] S. Carberry, K. Vijay-Shanker, A. Wilson, K. Samuel, Randomized rule selection in transformation-based learning: a comparative study, Nat. Lang. Eng. 7 (2) (2001) 99–116.
[16] W. Daelemans, J. Zavrel, P. Berck, S. Gillis, Mbt: a memory-based part of speech tagger generator, in: Proceedings of the Fourth Workshop on Very Large Corpora, 1996, pp. 14–27.
[17] L. Màrquez, H. Rodríguez, C.J. Girona, Automatically acquiring a language model for pos tagging using decision trees.
[18] T. Kudo, K. Yamamoto, Y. Matsumoto, Applying conditional random fields to japanese morphological analysis, in: Proceedings of EMNLP, 2004, pp. 230–237.
[19] J. Giménez, L. Marquez, Svmtool: a general pos tagger generator based on support vector machines, in: Proceedings of the 4th International Conference on Language Resources and Evaluation, 2004.
[20] M. Collins, Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms, Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing, vol. 10, Association for Computational Linguistics, 2002, pp. 1–8.
[21] L. Araujo, Studying the advantages of a messy evolutionary algorithm for natural language tagging, in: Genetic and Evolutionary Computation, Springer, 2003, pp. 1951–1962.
[22] D. Jurafsky, H. James, Speech and language processing an introduction to natural language processing, computational linguistics, and speech.
[23] L. Araujo, Symbiosis of evolutionary techniques and statistical natural language processing, IEEE Trans. Evol. Comput. 8 (1) (2004) 14–27.
[24] E. Alba, G. Luque, L. Araujo, Natural language tagging with genetic algorithms, Inf. Process. Lett. 100 (5) (2006) 173–182.
[25] A. Ekbal, S. Bandyopadhyay, Part of speech tagging in bengali using support vector machine, in: International Conference on Information Technology, 2008, ICIT'08, IEEE, 2008, pp. 106–111.
[26] H. Van Halteren, J. Zavrel, W. Daelemans, Improving data driven wordclass tagging by system combination, Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, vol. 1, Association for Computational Linguistics, 1998, pp. 491–497.
[27] M. Volk, G. Schneider, Comparing a statistical and a rule-based tagger for german, arXiv, preprint cs/9811016.
[28] R.M. Losee, Learning syntactic rules and tags with genetic algorithms for information retrieval and filtering: an empirical basis for grammatical rules, Inf. Process. Manage. 32 (2) (1996) 185–197.
[29] P.G. Reiser, P.J. Riddle, Evolution of logic programs: part-of-speech tagging, Proceedings of the 1999 Congress on Evolutionary Computation, 1999, CEC 99, vol. 2, IEEE, 1999.
[30] J.R. Curran, R.K. Wong, Formalisation of transformation-based learning, in: 23rd Australasian Computer Science Conference, 2000, ACSC 2000, IEEE, 2000, pp. 51–57.
[31] Z.W. Geem, C.-L. Tseng, Y. Park, Harmony search for generalized orienteering problem: best touring in china, in: Advances in Natural Computation, Springer, 2005, pp. 741–750.
[32] Z.W. Geem, Novel derivative of harmony search algorithm for discrete design variables, Appl. Math. Comput. 199 (1) (2008) 223–230.
[33] R. Forsati, A. Haghighat, M. Mahdavi, Harmony search based algorithms for bandwidth-delay-constrained least-cost multicast routing, Comput. Commun. 31 (10) (2008) 2505–2519.
[34] M. Mahdavi, M.H. Chehreghani, H. Abolhassani, R. Forsati, Novel meta-heuristic algorithms for clustering web documents, Appl. Math. Comput. 201 (1) (2008) 441–451.
[35] R. Forsati, M. Mahdavi, M. Shamsfard, M. Reza Meybodi, Efficient stochastic algorithms for document clustering, Inf. Sci. 220 (2013) 269–291.
[36] M. Šelmić, D. Teodorović, K. Vukadinović, Locating inspection facilities in traffic networks: an artificial intelligence approach, Transp. Planning Technol. 33 (6) (2010) 481–493.
[37] D. Teodorovic, T. Davidovic, M. Selmic, Bee colony optimization: the applications survey, ACM Trans. Comput. Logic 1529 (2011) 3785.
[38] G.D. Forney Jr, The viterbi algorithm, Proc. IEEE 61 (3) (1973) 268–278.
[39] R. Forsati, M. Shamsfard, Cooperation of evolutionary and statistical pos-tagging, in: 2012 16th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP), IEEE, 2012, pp. 446–451.
[40] D. Jurafsky, H. James, Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech, Pearson Education, 2010.
[41] K.S. Lee, Z.W. Geem, A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice, Comput. methods appl. mech. Eng. 194 (36) (2005) 3902–3933.
[42] M. Mirkhani, R. Forsati, M. Shahri, A. Moayedikia, A novel efficient algorithm for mobile robot localization, Robotics and Autonomous Systems.
[43] R. Forsati, M. Mahdavi, Web text mining using harmony search, in: Recent Advances in Harmony Search Algorithm, Springer, 2010, pp. 51–64.

[44] P. Lucic, D. Teodorovic, Bee system: modeling combinatorial optimization transportation engineering problems by swarm intelligence, in: Preprints of the TRISTAN IV triennial symposium on transportation analysis, 2001, pp. 441–445.
[45] K. Lua, Part of speech tagging of chinese sentences using genetic algorithm, in: Proceedings of ICCC96, National University of Singapore, 1996, pp. 45–49.
[46] C. Aone, K. Hausman, Unsupervised learning of a rule-based spanish part of speech tagger, in: Proceedings of COLING, vol. 96, 1996, pp. 53–58.
[47] F. Jelinek, Markov source modeling of text generation, in: The Impact of Processing Techniques on Communications, Springer, 1985, pp. 569–591.
[48] J. Carlberger, V. Kann, Implementing an efficient part-of-speech tagger, Softw. Pract. Experience 29 (9) (1999) 815–832.
[49] G.G. Lee, J. Cha, J.-H. Lee, Syllable-pattern-based unknown-morpheme segmentation and estimation for hybrid part-of-speech tagging of korean, Comput. Linguist. 28 (1) (2002) 53–70.
[50] L.H. Smith, T. Rindflesch, W. Wilbur, The importance of the lexicon in tagging biological text, Natural Lang. Eng. 12 (4) (2006) 335–351.
[51] F. Pla, A. Molina, Improving part-of-speech tagging using lexicalized hmms, Natural Lang. Eng. 10 (02) (2004) 167–189.
[52] ftp://ftp.cis.upenn.edu/pub/adwait/jmx/.