J Med Syst (2016) 40: 240
DOI 10.1007/s10916-016-0603-5

A Firefly Algorithm-based Approach for Pseudo-Relevance Feedback: Application to Medical Database

Ilyes Khennak[1] · Habiba Drias[1]

Received: 29 February 2016 / Accepted: 8 September 2016 / Published online: 27 September 2016

Abstract The difficulty of disambiguating the sense of the incomplete and imprecise keywords that are extensively used in the search queries has caused the failure of search systems to retrieve the desired information. One of the most powerful and promising method to overcome this shortcoming and improve the performance of search engines is Query Expansion, whereby the user's original query is augmented by new keywords that best characterize the user's information needs and produce more useful query. In this paper, a new Firefly Algorithm-based approach is proposed to enhance the retrieval effectiveness of query expansion while maintaining low computational complexity. In contrast to the existing literature, the proposed approach uses a Firefly Algorithm to find the best expanded query among a set of expanded query candidates. Moreover, this new approach allows the determination of the length of the expanded query empirically. Experimental results on MEDLINE, the online medical information database, show that our proposed approach is more effective and efficient compared to the state-of-the-art.

This article is part of the Topical Collection on *Systems-Level Quality Improvement*

✉ Ilyes Khennak
ikhennak@usthb.dz

Habiba Drias
hdrias@usthb.dz

1 Laboratory for Research in Artificial Intelligence, Computer Science Department, USTHB, BP 32 El Alia, 16111, Bab Ezzouar, Algiers, Algeria

Keywords Medical data management · Web intelligence · Query expansion · Pseudo-relevance feedback · Firefly Algorithm · Swarm intelligence · MEDLINE

Introduction

Nowadays, the volume of information available on the World Wide Web is constantly increasing and the number of published websites is continuously growing. For instance, the total number of websites had risen from 200 million in 2010 to 900 million in 2014 and this number is expected to exceed 1 Billion in 2016. In addition, the amount of user-generated content posted on websites is exponentially expanding, especially on social networking sites. Every minute, Facebook and Twitter users share nearly 2.5 million pieces of content and 300 thousand tweets, respectively. The number of search queries has also considerably grown. In 2012, Google received 2 million queries per minute and this number had doubled in 2014. Besides the rapidly growing amount of data, the Internet traffic has dramatically increased. According to the latest Cisco study, the Internet data traffic will reach 1 ZB by 2016, and it is expected to double in 2019. This is mainly due to the large number of devices connected to the net, such as: PCs, smartphones, tablets, and machine-to-machine (M2M). The unprecedented explosion of information that the Web is experiencing has led to the following results:

– New words are continuously being introduced in the World Wide Web. According to Williams and Zobel [41], there is one new word in every two hundred words. Studies by [16, 39, 41] showed that this invasion is mainly owing to: neologisms, first occurrences of rare personal names/place names, abbreviations, acronyms, emoticons, URLs and typographical errors.

– The Web users are constantly exploiting these new words in their search queries. Chen et al. [12] indicated in their study that more than 17 % of query keywords are out of vocabulary, 45 % of them are E-speak (lol), 18 % are companies and products such as new medicament or viruses names, 16 % are proper names, 15 % are misspellings and foreign words [1, 38].

Undoubtedly, the difficulty of exploring the meanings of these new ambiguous and imprecise words will certainly result in a failure of the web search engines and cause retrieval of irrelevant information. One of the most natural and successful method to overcome this limitation is Query Expansion (QE), whereby the user's original query is augmented by new keywords that best capture the actual user intent, or that simply produce a more useful query, i.e., a query that is more likely to retrieve relevant information [11]. Currently, QE is considered an extremely promising technique to enhance the retrieval effectiveness and it is widely used in various applications such as question answering [3], multimedia information retrieval [43], medical and health [29], information filtering [26], e-commerce [23], and mobile search [17]. Recently, a huge number of new QE techniques have been proposed using a variety of approaches that leverage on several data sources and employ sophisticated methods for finding new features correlated with the query keywords. Nonetheless, the retrieval effectiveness of these techniques was not substantially different compared to previous works and state-of-the-art. This is mainly due to the fact that these techniques are basically derived from traditional methods which are looking for the best expansion keywords and not really the appropriate expanded query. In contrast, extracting the best expanded query instead of the best expansion keywords is a serious issue as the number of possible expanded query candidates is tremendous.

One way to tackle this issue is to consider the problem of selecting the best expanded query as an optimization problem, which aims at searching the best solution from the set of all potential ones [18]. Due to the practical importance of optimization, numerous efforts have been made to develop diverse kinds of algorithms to cope with complex problems. There are two categories of methods: complete and approximate [7]. Complete algorithms are guaranteed to find for every finite size instance of a problem an optimal solution in bounded time. Nevertheless, for NP-hard problems, no polynomial time algorithm exists. Therefore, complete methods might need exponential computation time in the worst-case. This often leads to computation times too high for practical purposes. Thus, the use of approximate methods to solve combinatorial optimization problems has received more and more attention. In approximate methods, we sacrifice the guarantee of finding optimal solutions for the sake of getting good solutions in a significantly reduced amount of time. In the last two decades, a new class of approximate methods has emerged. These methods are often surprisingly efficient in practice in solving difficult optimization problems. We are interested in this paper in swarm algorithms that are nature-inspired, mimicking some successful characteristics of animal swarms [46]. These evolutionary algorithms include strategies that efficiently explore the search space in order to find (near) best solutions. Good examples of swarm algorithms are Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) and Firefly Algorithm (FA) [47]. Selecting a robust and strong evolutionary algorithm in order to solve the complex problem yields effective and efficient outcomes. One of the recent swarm algorithms that has a great potential success is Firefly Algorithm (FA) developed by Yang in 2008 [44, 45]. FA is based on the flashing behavior of fireflies and the phenomenon of bioluminescent communication. The interest exhibited by the scientists for this algorithm, still increases day by day in the literature because of its flexible structure and clear and simple modelling. Moreover, it includes important features brought from previous evolutionary approaches such as a random walk strategy, a simulated annealing convergence strategy and movement rules of Particles Swarm Optimization (PSO), in order to increase effectiveness, efficiency and to avoid premature convergence.

In this paper, an original approach based on Firefly Algorithm is proposed to enhance the retrieval effectiveness of query expansion while maintaining low computational cost. Unlike the existing state-of-the-art techniques, a Firefly Algorithm is adapted for query expansion issue to extract the best query candidates. The overall procedure consists in, first retrieving the pseudo-relevant documents to the query, extract their terms to constitute potential solutions and finally launch the Firefly Algorithm to determine the best way to build the expanded query. Among the strengths of our proposal is a balance between effectiveness and efficiency but further, the determination through experiments, of the number of pseudo-relevant documents to consider for the expansion process and the length of the expanded query.

We evaluate our approach using MEDLINE dataset, the on-line medical database, and we use the two well-known query expansion techniques: Rocchio's method, and Robertson/Sparck Jones' term-ranking function as the baseline for comparison. The reminder of this paper is organized as follows. In the next section, we review briefly the related work. In "Background", we survey some definitions and concepts, covering the expansion query, swarm intelligence and Firefly Algorithm. In "Firefly algorithm for pseudo-relevance feedback", we introduce a Firefly Algorithm for generating the best possible query expansion. Experimental results are presented in "Experiments" before we conclude in "Conclusion".

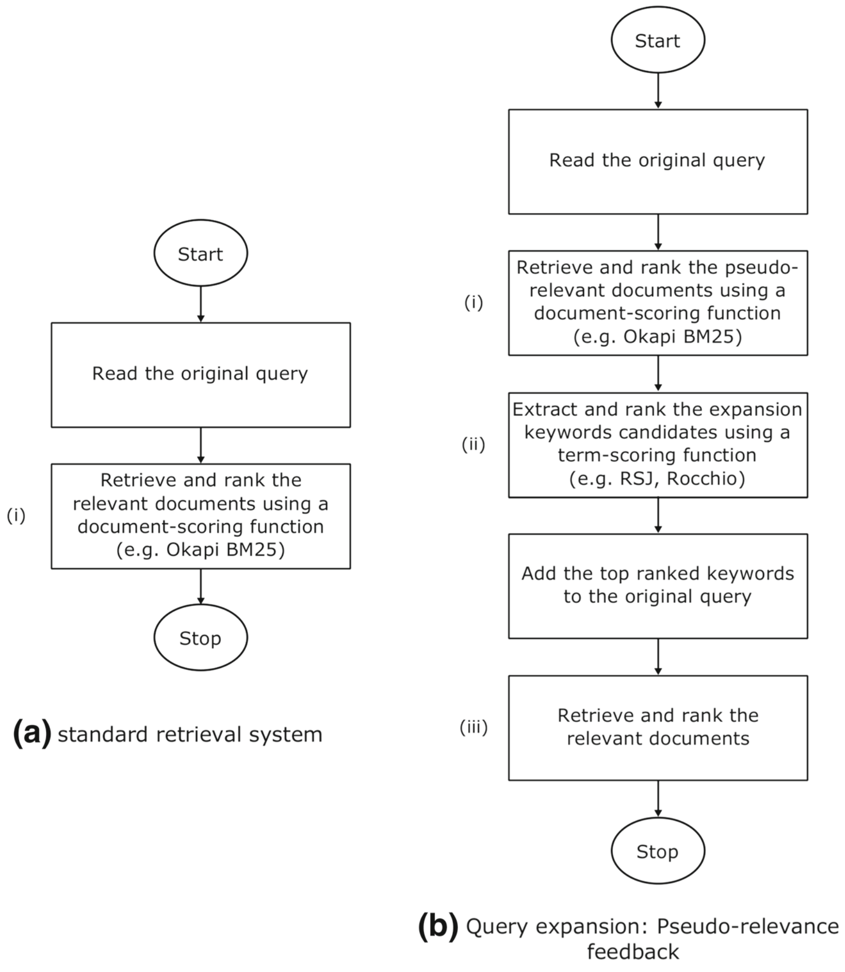
Related work

The massive influx of new features on the web such as first occurrences of proper names, abbreviations, misspelling words, etc., as well as the use of these ambiguous and imprecise features to describe the user need have resulted in a failure to retrieve the relevant information. Furthermore, the Vocabulary problem (or the term mismatch problem) still remains one of the most obstacles presently facing the retrieval effectiveness. To cope with these critical issues, several methods have been suggested including interactive query refinement (Google suggest), word sense disambiguation [40], search results clustering [5], boolean term decomposition [42], spreading activation networks [13], concept lattice-based IR [10], random indexing [37], and contextual document ranking modeled as basis vectors [30]. However, expanding the original query with additional keywords is one of the most successful and promising techniques to enhance the retrieval effectiveness of document ranking [11].

Pseudo-relevance feedback for query expansion

As mentioned above, one of the most natural and successful techniques to improve the retrieval effectiveness of document ranking is to enrich the original query with additional keywords that best capture the actual user intent. Many approaches have been proposed to select and extract those additional keywords. One well-known query expansion (QE) approach is Pseudo-Relevance Feedback (PRF) called also Blind Feedback or Retrieval Feedback. It uses the pseudo-relevant documents, i.e. the first documents retrieved in response to the initial query, to select the most important words to be used as expansion keywords [11]. Figure 1 illustrates the integration of the PRF technique in a classic retrieval system. In its simplest version, the pseudo-relevance feedback starts by: (i) performing an

Fig. 1 Standard system vs. system using a PRF technique


Start
Read the original query
(i)
Retrieve and rank the relevant documents using a document-scoring function (e.g. Okapi BM25)
Stop
(a) standard retrieval system
Start
Read the original query
(i)
Retrieve and rank the pseudo-relevant documents using a document-scoring function (e.g. Okapi BM25)
(ii)
Extract and rank the expansion keywords candidates using a term-scoring function (e.g. RSJ, Rocchio)
Add the top ranked keywords to the original query
(iii)
Retrieve and rank the relevant documents
Stop
(b) Query expansion: Pseudo-relevance feedback

initial search on the original query. For each query term, the inverted index is retrieved and processed. An inverted index specifies which documents that particular term occurs in. For each document in the list, a score is calculated and added to the ranked list of scores. This score is calculated using a document-scoring function to retrieve and rank the documents. One widely popular document-scoring function is introduced within the Probabilistic Relevance Framework: Okapi BM25 (see "Background"). The top ranked documents are considered as pseudo-relevant documents. (ii) Then, extracting the expansion keywords from the pseudo-relevant documents and ranking them using a term-scoring function. One of the best term-scoring functions are Robertson/Sparck Jones term-ranking function (RSJ) and Rocchio weight (see "Background"). (iii) Finally, expanding the original query by adding the top ranked keywords and retrieving the relevant documents to the expanded query using a document-scoring function.

Ineffectiveness of pseudo-relevance feedback

Query expansion has made a big leap, forward by inventing effective techniques which overcome the difficulty of providing more precise description of user request. Furthermore, it has yielded significant improvements in retrieval effectiveness over the traditional information retrieval systems. Nevertheless, the retrieval effectiveness of latest pseudo-relevance feedback techniques was not substantially different compared to previous works and state-of-the-art. This is predominantly due to the fact that all of these techniques are basically derived from traditional methods which are mainly looking for the best expansion keywords and not really the suitable expanded query.

For instance, the work of Miao et al. [31] proposed a proximity-based classical Rocchio's Model for pseudo-relevance feedback. This proposed approach incorporates proximity information into the Rocchio's model, and proposes a proximity-based Rocchio's model, called PRoc, with three kinds of proximity measures. The traditional statistics of expansion terms and the proximity relationship between expansion terms and the query terms were also taken into consideration in order to enhance the effectiveness of PRF.

Cao et al. [9] attempted also to enhance PRF using a supervised learning method for term selection in order to separate good expansion terms from the others directly according to their potential impact on the retrieval effectiveness. This work considers the term selection as a term classification and uses the classifier C-SVM with term distribution and term proximity criteria to select the best expansion keywords.

Another related work is the one presented by Lee and Croft [24]. It employs a cluster-based resampling method to select novel pseudo-relevant documents based on Lavrenko's relevance model approach. Its main idea is to use overlapping clusters to find dominant documents for the initial retrieval set, and to repeatedly use these documents to emphasize the core topics of a query. This proposed resampling method can skip some documents in the initial high-ranked documents and deterministically construct overlapping clusters as sampling units.

A boosting approach to improve the performance of pseudo-relevance feedback has been proposed in [28]. It uses a learning algorithm, called FeedbackBoost, based on the boosting framework to improve pseudo-relevance feedback through optimizing the combination of a set of basis feedback algorithms using a loss function defined to measure both robustness and effectiveness. The FeedbackBoost algorithm accommodates different document weighting strategies in the feedback model. Still in the same direction, Dillon and Collins-Thompson [15] developed new pseudo-relevance feedback algorithm, called CCCP, that unifies two complementary PRF algorithm: (i) The CT algorithm, which finds a set of feedback terms by combining term covariance information with a set of task-specific constraints that prune out bad expansion models, and (ii) The TZ algorithm, which based on the language modeling approach that jointly solves for both term and document weights. A simple term frequency transformation model [48] is another stream of work which based on traditional methods to improve the performance of pseudo-relevance feedback but did not provide significant improvements over baseline. This work proposes a simple and heuristic model, in which three term frequency transformation techniques are integrated to capture the saliency of a candidate term associated with the original query terms in the feedback documents.

Particle swarm optimization algorithm was also introduced to improve pseudo relevance feedback through choosing the appropriate combination of weights of terms in query vector and uses fitness function as a measure of the proximity between the re-weighted query vector and top ranked documents retrieved from original query vector [6]. This work is the one that is the closest to ours. Using PSO for query expansion is a fine idea, as it is very popular because of its simplicity of implementation, its convergence and effectiveness. What we find as weaknesses to this work is the way the solution is modelled and the achieved results are almost similar to those of the state-of-the-art. In our study, we propose a more realistic modelling for the query expansion and second, an algorithm based on firefly, which is more general than PSO. The algorithm has many interesting features such as integrating a random walk strategy, a simulated annealing-like strategy [22] for premature convergence avoidance. All these components allow Firefly Algorithm, to balance between effectiveness and efficiency.

Background

In this section, we present some background needed in the rest of this paper. We first introduce the well-known document-scoring function of probabilistic relevance framework (Okapi BM25) and the best-known functions for term-scoring. Next, we survey the flashing behavior of fireflies and the standard formulation of FA.

Probabilistic relevance framework: Okapi BM25

The probabilistic relevance framework is a formal framework for document retrieval which led to the implementation of one of the most successful text retrieval algorithms: Okapi BM25 [35]. The classic version of Okapi BM25 term-weighting function, in which the weight w_i^{BM25} is attributed to a given term t_i in a document d, is obtained using the Formula (1):

$$w_i^{BM25} = \frac{tf}{k_1\left((1-b)+b\dfrac{dl}{avdl}\right)+tf} w_i^{RSJ} \tag{1}$$

Where: tf, is the frequency of the term t_i in a document d;
k_1, is a constant;
b, is a constant;
dl, is the document length;
$avdl$, is the average of document length;
w_i^{RSJ}, is the well-know Robertson/Sparck Jones weight [34]. It is calculated using Eq. 2:

$$w_i^{RSJ} = \log \frac{(r_i+0.5)(N-R-n_i+r_i+0.5)}{(n_i-r_i+0.5)(R-r_i+0.5)} \tag{2}$$

Where:
N, is the number of documents in the whole collection;
n_i, is the number of documents in the collection containing t_i;
R, is the number of documents judged relevant;
r_i, is the number of judged relevant documents containing t_i.

The RSJ weight can be used with or without relevance information. In the absence of relevance information, the weight is reduced to a form of classical IDF as shown in Eq. 3:

$$w_i^{IDF} = \log \frac{N-n_i+0.5}{n_i+0.5} \tag{3}$$

The final Okapi BM25 term-weighting function is therefore given by Formula (4):

$$w_i^{BM25} = \frac{tf}{k_1\left((1-b)+b\dfrac{dl}{avdl}\right)+tf} \log \frac{N-n_i+0.5}{n_i+0.5} \tag{4}$$

Regarding the internal parameters, a considerable number of experiments have been conducted and suggested that values such as $1.2 < k_1 < 2$ and $0.5 < b < 0.8$ are reasonably good in many cases. The study of Robertson and Zaragoza [33] indicated that published versions of Okapi BM25 are based on specific values assigned to k_1 and b: $k_1 = 2, b = 0.5$. As part of the indexing process, an inverted index is created containing the weight w_i^{BM25} of each term t_i in each document d. The similarity score between the document d and a given query Q is then computed using a document-scoring function, as shown in Eq. 5:

$$Score_{BM25}(d,Q) = \sum_{t_i \in Q} w_i^{BM25} \tag{5}$$

During the search process, the relevant documents are selected and ranked using this similarity score.

Term-scoring functions for pseudo-relevance feedback

As outlined previously, the retrieval pseudo-relevance feedback technique starts by performing an initial search on the original query using the previous Okapi BM25 term-weighting and document-scoring function, suppose the best ranked documents to be relevant, assign a score to each keyword in the top retrieved documents using a term-scoring function, and then sort them on the basis of their scores. One of the best-known functions for term-scoring is the Robertson/Sparck Jones term-ranking function, defined by Formula 2. Another well-known function used for term-scoring is the Rocchio weight [36] shown in Eq. 6:

$$w_i^{Rocchio} = \sum_{d \in R} w_i^{BM25} \tag{6}$$

Where:
R, is a set of pseudo-relevant documents.

The original query is then expanded by adding the top ranked terms and re-interrogated using the Okapi BM25 similarity score (Formula (5)), in order to get more relevant results.

Firefly algorithm

Swarm algorithms are powerful methods for solving many tough optimization problems [14, 20, 32]. Their strength lies in their capability in dealing with complex problems with no or little knowledge of the search space. For example, particle swarm optimization was developed based on the birds flocks and fishes schools behavior [20, 21]. New algorithms are also emerging recently, including Harmony Search (HS) and Bat Algorithm (BA). One of the recent swarm intelligence algorithm, which has great potential is Firefly Algorithm (FA) proposed by Yang in 2008 [44, 45]. FA is based on the flashing behavior of fireflies and the phenomenon of bioluminescent communication. Fireflies, also referred to as lightening bugs, are luminous beetles from the family Lampyridae. Most of fireflies are characterized by their use of bioluminescence to communicate, attract mate, hunt for prey, and to warn predators of their bitter taste. There are up to 2000 different species of fireflies, found in temperate and tropical environments around the world. All of them fall under five main subfamilies: Photurinae, Luciolinae, Cyphonocerinae, Lampyrinae, and Otetrinae.

Standard firefly algorithm

FA has been applied to various domains, including combinatorial optimization [4], neural networks [2], clustering [25], data mining [49], image processing [8], fuzzy logic [27], and scheduling [19]. The FA procedure is based on interactions between individual fireflies. The level of interaction is modeled by the strength of this event. Each firefly has its attractiveness, which is used to attract other fireflies to it. Attractiveness depends on the light intensity, so each firefly is attracted to the neighbor that glows brighter. Relying on the behavior of the flashing characteristics of fireflies, the Firefly Algorithm was designed with the following three rules:

- All fireflies are unisex so that one firefly is attracted to other fireflies regardless of their sex.
- The attractiveness of a firefly is proportional to its light intensity which decreases as the distance from the other firefly increases. If there is not a more attractive firefly than a particular one, it will move randomly.
- The light intensity of a firefly is affected or determined by the landscape of the objective function. For maximization problems, the light intensity is proportional to the value of the fitness function.

Firefly movement

The FA is a population-based algorithm, where each population member (i.e., firefly) represents a candidate solution of the problem to be solved and thus denotes a point in the search space.

In simulation, at iteration t, each firefly i in the population is characterized by a position x_i^t. The position of each solution is evaluated by calculating its fitness value $f(x_i)$. The overall best solution x_* is selected according to the light intensity I_i, which is expressed by the objective function $f(x_i)$. The position x_i^{t+1} of each member in the swarm is adjusted by moving firefly i to another more attractive firefly j according to Eq. 7:

$$x_i^{t+1} = x_i^t + \beta(x_j^t - x_i^t) + \alpha\left(rand - \frac{1}{2}\right) \tag{7}$$

Equation 7 consists of three terms. The first term determines the current position of the ith firefly. The second term refers to a social component of moving the firefly i towards the more attractive firefly j, while the third term is connected with the randomized move of the ith firefly within the search space.

In the second term of Eq. 7, parameter β is the attractiveness of firefly j. It is described by monotonically decreasing function of the distance r_{ij} between the fireflies i and j, as shown in Eq. 8:

$$\beta = \beta_0 e^{-\gamma r_{ij}^2} \tag{8}$$

Where:

β_0, is the maximum attractiveness value (i.e. at $r = 0$);

γ, is the light absorption coefficient, which controls the decrease of the light intensity;

r_{ij}, is the distance between the fireflies i and j at the positions x_i and x_j. It is obtained by the Cartesian distance as defined in Formula (9):

$$r_{ij} = \| x_i - x_j \| = \sqrt{\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2} \tag{9}$$

Where:

$x_{i,k}$, is the kth element of the ith firefly position within the search space;

d, the dimensionality of a problem.

In the third term of Eq. 7, α is a randomization parameter and $rand$ is a random number generator uniformly distributed in the range [0, 1].

Controlling randomization

To prevent the algorithm from the premature convergence, the randomization parameter α can be varied and gradually decreased using Formula (10):

$$\alpha = \alpha_0 \theta^t \tag{10}$$

Where:

α_0, is the initial randomization parameter;

θ, is the randomness reduction constant;

$t \in [0, T]$, is the pseudo time for simulations and T is the maximum number of generations.

The original firefly algorithm

The main steps of the Firefly Algorithm are presented in Algorithm 1 and its description can be summarized as follows:

Algorithm 1 pseudo code of the Firefly Algorithm (FA)

```
1: Fitness function $f(x)$
2: Define light absorption coefficient $\gamma$, Initial attractiveness $\beta_0$, parameter $\alpha$
3: Light intensity $I_i$ at $x_i$ is determined by $f(x_i)$
4: Generate an initial population of $N$ fireflies $x_i$ $(i = 1, 2, ..., N)$
5: while ($t$ < Max number of iterations) do
6:     for $i \leftarrow 1$ to $N$ (all $N$ fireflies) do
7:         for $j \leftarrow 1$ to $N$ (all $N$ fireflies) do
8:             if $I_i < I_j$ then
9:                 Move firefly $i$ towards $j$ (Equation 7);
10:            end if;
11:            Vary attractiveness;
12:            Evaluate new solutions and update light intensity;
13:        end for
14:    end for
15:    Rank the fireflies and find the current global best $x_*$;
16: end while
```

The algorithm starts by setting the initial values of the maximum attractiveness β_0, the absorption coefficient γ, and randomization parameter α. The initial population of N solutions $(i = 1, 2, ..., N)$ is generated randomly, where each solution is initialized with a location x_i^0. The light intensity I_i^0 of each solution in the initial population is determined by the fitness function value $f(x_i^0)$. The best solution x_* is then initialized by the best solutions of all the fireflies. The FA iteratively moves from the initial set of solutions to the best solution by adjusting the position x_i of firefly i to the position x_j of another more attractive firefly j according to Eq. 7. Solutions are continuously updated based on the continuous flying iterations. This is repeated till the termination criteria are satisfied. Finally, when all criteria successfully met the best so far solution is reported.

Firefly algorithm for pseudo-relevance feedback

In this section, we present a Firefly Algorithm-based approach to improve the retrieval effectiveness of pseudo-relevance feedback while maintaining high efficiency. The overall procedure consists in, first retrieving the pseudo-relevant documents to the query, extract their terms to constitute potential solutions and finally launch the Firefly Algorithm to determine the best way to build the expanded query. Figure 2 describes the proposed Firefly Algorithm for PRF.

In the following, we describe our proper modelling of the query expansion problem with Firefly Algorithm. The encoding of the solution, the search space or solutions space, the fitness function and the parameters setting are presented in particular.

Representation of solutions and the initialization of the population

The first step in designing a Firefly Algorithm is to devise a suitable representation scheme for candidate solutions. Each candidate solution is encapsulated in a firefly, the search space is then considered as a heart of artificial fireflies. A firefly for our problem is an expanded query composed of two parts: the first includes the keywords of the initial query and the second is formed by other terms belonging to R, the set of the pseudo-relevant documents retrieved in response

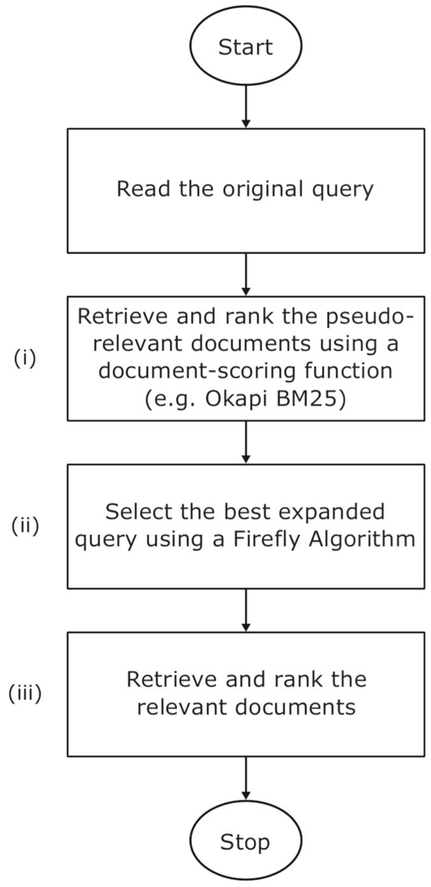

Start
Read the original query
(i)
Retrieve and rank the pseudo-relevant documents using a document-scoring function (e.g. Okapi BM25)
(ii)
Select the best expanded query using a Firefly Algorithm
(iii)
Retrieve and rank the relevant documents
Stop

Fig. 2 The proposed Firefly Algorithm for PRF

to the initial query. But for optimization concern, only the terms to be added to the query are considered because the first part is fixed. It is represented by a vector of keywords, as shown in Eq. 11:

$$\grave{Q} =< \grave{q}_1, \grave{q}_2, \grave{q}_3, ..., \grave{q}_{|\grave{Q}|} > \tag{11}$$

Where:

$\grave{Q}$, is a vector containing the terms to be added to the initial query. The overall expanded query is represented as follows:

$$EQ = Q + \grave{Q} =< q_1, q_2, ..., q_{|Q|}, \grave{q}_1, \grave{q}_2, ..., \grave{q}_{|\grave{Q}|} > \tag{12}$$

Where:

q_i, is a term of the initial query Q, $i = 1, ..., |Q|$;

$\grave{q}_i$, is a term to be added to the initial query and belonging to $\grave{V}$

$\grave{V} \subset V$, is the set of terms of all pseudo-relevant documents;

V, is the vocabulary (dictionary) of the total collection of documents;

$|Q|$, is the number of initial query keywords;

$|\grave{Q}|$, is the number of expansion keywords, which is a parameter to be empirically determined. The number of pseudo-relevant documents is also another parameter to be experimentally estimated. These features are considered for the first time for query expansion approaches, and provide more strength to our proposal.

The search space includes all the potential expanded parts without the fixed part of the expanded query, which is the initial user query. The search or solution space includes all possible combinations of terms belonging to $\grave{V}$ having the same length equal to $|\grave{Q}|$. The size of the search space is equal to the binomial coefficient, as stated in Eq. 13:

$$c(\grave{Q}, \grave{V}) = \frac{|\grave{V}|!}{|\grave{Q}|!(|\grave{V}| - |\grave{Q}|)!} \tag{13}$$

Where:

$c(\grave{Q}, \grave{V})$, is the number of combinations of $|\grave{Q}|$ terms among the total $|\grave{V}|$ terms.

This huge number expresses the numerous possibilities to build a query expansion. Considering the issue as an optimization problem, and addressing it with a proper approach is the optimal way to achieve effectiveness as well as efficiency. Regarding the creation of the initial swarm of fireflies, a fictive firefly is generated merely by selecting $|\grave{Q}|$ terms from $\grave{V}$. N fireflies with this schema, are then drawn at random to constitute a population.

Fitness function

The objective function has the aim to evaluate the quality of the solution. As a solution in our case is an expanded query involving terms, the obvious way to assess its performance is to take into consideration the inverted indexes of its elements and then compute the scores of each document belonging to R with respect to the expanded query EQ. The best score is then considered as the fitness value of the expanded query. The proposed fitness function is described by Formula (14) and detailed in Algorithm 2:

$$f(EQ) = max\,(Score_{BM25}(d_1, EQ), ..., Score_{BM25}(d_{|R|}, EQ)) \tag{14}$$

Algorithm 2 pseudo code of the fitness function

```
1:  Fitness function $f(EQ)$
2:  Inverted index $In(\grave{q}_i)$
3:  Initialize the vector of scores $S \leftarrow []$
4:  for each keyword $\grave{q}_i$ in $\grave{Q}$ do
5:      for each document $d$ in $R$ do
6:          if $d \in In(\grave{q}_i)$ and $S[d] \neq 0$ then
7:              $S[d] \leftarrow Score_{BM25}(d, Q) + Score_{BM25}(d, \grave{Q})$;
8:          end if;
9:      end for
10: end for
11: $f(EQ) \leftarrow max(S)$;
```

Algorithm 2 starts by recovering the inverted index In of each expansion term $\grave{q}_i \in \grave{Q}$, and iteratively computes the score of each document d included in both R and In with respect to the expanded query EQ. In order to avoid calculating the same score several times, a check is performed to ensure that no score has yet been assigned to document d in the vector of scores S. In addition, as the score of d to Q is calculated in the first step (i), only the score of d to $\grave{Q}$ is computed at this stage. Once all the scores are calculated, the maximum value of S is returned as the fitness value of the expanded query EQ.

When the last generation of the proposed Firefly Algorithm is achieved, the vector S of the best solution is sorted and its elements are considered as relevant documents to the user's original query.

From Algorithm 2, it can be observed that the query expansion issue is formulated as a maximization problem. Thus, the light intensity is expressed by the value of the fitness function.

Update locations of fireflies

The searched solution changes all along the firefly process in an increasing order of its fitness function. The current best solution of all the previous iterations is denoted by $\grave{Q}_*$. At each iteration, the algorithm computes the best firefly of the generation, which becomes the global best one if it is better than $\grave{Q}_*$. To adapt an effective Firefly Algorithm for pseudo-relevance feedback issue, some modifications on the basic FA are made. Hence, the movement of firefly $\grave{Q}_i$ towards another more attractive $\grave{Q}_j$ is performed in two steps. The first step, which employs the second term of Eq. 7, is expressed by Formula (15):

$$\beta = \frac{1}{1 + \gamma r_{ij}} \tag{15}$$

Where:
$\beta \in]0, 1]$, is the attractiveness of firefly j;
γ, is the light absorption coefficient;
r_{ij}, is the distance between $\grave{Q}_i$ and $\grave{Q}_j$. It is calculated using the well-known Hamming distance. The Hamming distance between two fireflies is determined by the number of non-corresponding elements in the sequence. For example, given two solutions $\grave{Q}_1$ and $\grave{Q}_2$ containing 5 keywords:

$$\grave{Q}_1 = < \grave{q}_4, \grave{q}_2, \boldsymbol{\grave{q}_6}, \boldsymbol{\grave{q}_3}, \boldsymbol{\grave{q}_7} >$$
$$\grave{Q}_2 = < \boldsymbol{\grave{q}_1}, \grave{q}_4, \boldsymbol{\grave{q}_8}, \boldsymbol{\grave{q}_5}, \grave{q}_2 >$$

The Hamming Distance between $\grave{Q}_1$ and $\grave{Q}_1$ would be 3.

The importance of this step lies in bringing the solution $\grave{Q}_i$ closer to another more attractive solution $\grave{Q}_j$. In other words, after performing this step, the distance between $\grave{Q}_i$ and $\grave{Q}_j$ is decreased while the attractiveness value is increased. Moreover, the number of keywords of $\grave{Q}_j$ in $\grave{Q}_i$ at iteration $t + 1$ is greater than or equal to this number at iteration t. Hence, the next position of $\grave{Q}_i$ is updated by moving each of its elements $\grave{Q}_{ik}$ (keyword), which does not belong to $\grave{Q}_j$, as shown in Eq. 16:

$$\grave{Q}_{ik}^{t+1} = \begin{cases} \grave{Q}_{ik}^t, & \text{if } rand > \beta \\ \grave{Q}_{jl}^t, & \text{otherwise} \end{cases} \tag{16}$$

Where:
$\grave{Q}_{ik}^t \notin \grave{Q}_j^t$, is a keyword belonging to $\grave{Q}_i^t$;
$\grave{Q}_{jl}^t \notin \grave{Q}_i^t$, is a keyword selected randomly from $\grave{Q}_j^t$.

Once the first step is completed, the second stage is then conducted using the third term of Eq. 7. For our problem, the third term is determined by the value of α. The randomization parameter α is very important in a Firefly Algorithm as it avoids the solutions being trapped at local optima. Its value is gradually decreased to prevent the algorithm from the premature convergence. Based on the value of α, the position of $\grave{Q}_i$ is updated by adjusting randomly each of its elements $\grave{Q}_{ik}$, which is not included in $\grave{Q}_j$, as given by Formula (17):

$$\grave{Q}_{ik}^{t+1} = \begin{cases} \grave{Q}_{ik}^t, & \text{if } rand > \alpha \\ \grave{q}_l, & \text{otherwise} \end{cases} \tag{17}$$

Where:
$\grave{q}_l$, is a keyword selected randomly from the set of terms $\grave{V}$.

Exploitation and exploration are two main features of any swarm algorithms. From the above two steps, it can be observed that those features are achieved. In the first step, the Exploitation is attained by focusing on the search in a local region by exploiting the information that a more attractive firefly is found in this region. In the second stage, the exploration is realized by exploring the search space and generating diverse solutions via the randomization. The good combination of these two major components will usually ensure that the global optimality is achievable.

Pseudo code of the proposed firefly algorithm for PRF

The main steps of the proposed Firefly Algorithm for pseudo-relevance feedback are summarized in Algorithm 3. The algorithm starts by setting the initial values of the absorption coefficient γ and the randomization parameter α. The initial population of N solutions ($i = 1, 2, ..., N$) is then created randomly. Each solution in the initial swarm is generated by selecting $|\grave{Q}|$ terms from $\grave{V}$ and initialized with a location $\grave{Q}_i^0$. The light intensity I_i^0 of each solution in the initial population is determined by the fitness function value $f(\grave{Q}_i^0)$. The best solution $\grave{Q}_*$ is then initialized by the best solutions of all the fireflies. The FA iteratively moves from the initial set of solutions to the best solution by moving the firefly $\grave{Q}_i$ towards another more attractive $\grave{Q}_j$ using Eqs. 16 and 17. Solutions are continuously updated based on the continuous flying iterations. This is repeated till the best solution does not change for a pre-specified interval of generations or the maximum number of iterations is reached. Finally, when the termination criteria are successfully met, the best so far expanded query is reported.

Algorithm 3 pseudo code of the proposed Firefly Algorithm for PRF

```
1:  Fitness function $f(\grave{Q})$
2:  Define light absorption coefficient $\gamma$, parameter $\alpha$
3:  Light intensity $I_i$ at $\grave{Q}_i$ is determined by $f(\grave{Q}_i)$
4:  Generate an initial population of $N$ fireflies $\grave{Q}_i$ $(i = 1, 2, ..., N)$
5:  while ($t$ < Max number of iterations and the best expanded query still changes) do
6:      for $i \leftarrow 1$ to $N$ (all $N$ fireflies) do
7:          for $j \leftarrow 1$ to $N$ (all $N$ fireflies) do
8:              if $f(\grave{Q}_i) < f(\grave{Q}_j)$ then
9:                  Move query firefly $\grave{Q}_i$ towards $\grave{Q}_j$ (Eqs. 16 and 17);
10:             end if;
11:             Vary attractiveness;
12:             Evaluate new solutions;
13:         end for
14:     end for
15:     Rank the fireflies and find the current best expanded query $\grave{Q}_*$;
16: end while
```

Finally, as indicated earlier, the vector S of the best solution is sorted and its elements are considered as relevant documents to the user's original query.

Experiments

In order to evaluate the performance of the proposed approach, we conduct extensive experiments on MEDLINE dataset and compare the results to those obtained by RSJ and Rocchio techniques. We first describe the dataset used, the software, the effectiveness and efficiency measures, and then present the experimental results.

Dataset

Extensive experiments are performed on MEDLINE collection, the on-line medical information database. This collection includes 348 566 references, consisting of titles and/or abstracts from 270 medical journals over a five-year period (1987-1991). The available fields are title, abstract, MeSH indexing terms, author, source, and publication type. In addition, the collection contains a set of queries, and relevance judgments (a list of which documents are relevant to each query). Regarding the queries, the MEDLINE collection includes 106 queries. Each query is accompanied by a set of relevance judgments chosen from the whole collection of documents. Table 1 summarizes the characteristics of the MEDLINE collection.

Table 1 Summary of MEDLINE collection and queries used in our experiments

Number of document in the collection	348566
Average document length in terms of words ($avdl$)	122
Number of words in the dictionary	308083
Number of queries	106
Average length query in terms of words	5.25

During the indexing phase, all non-informative words such as prepositions, conjunctions, pronouns and very common verbs are disregarded using a stop-word list. Subsequently, the most common morphological and inflectional suffixes were removed by adopting a standard stemming algorithm. Finally, the weights of the words are calculated for each document using to the well-known Okapi BM25 term weighting presented in "Background".

Software, effectiveness and efficiency measures

The proposed FA for pseudo-relevance feedback (FA-PRF) is implemented in Python. All the experiments are performed on a Sony-Vaio workstation having an Intel i3-2330M/2.20GHz processor, 4GB RAM and running Ubuntu GNU/Linux 12.04. The precision (P) and the mean average precision (MAP) measures are used to evaluate the effectiveness of the proposed approach in comparison with RSJ and Rocchio techniques. The precision is the ration of relevant documents retrieved over the total number of documents retrieved, as shown in Eq. 18:

$$P = \frac{\text{Number of relevant documents retrieved}}{\text{Number of documents retrieved}} \tag{18}$$

The mean average precision is the mean of the average precision scores over all queries. It is computed using Eq. 19:

$$MAP = \frac{1}{NbQ} \sum_{i=1}^{NbQ} \frac{1}{m_i} \sum_{j=1}^{m_i} P(R_{ij}) \tag{19}$$

Where:

NbQ, is the number of queries;

m_i, is the number of relevant documents for the ith query;

R_{ij}, is the set of ranked retrieval results from the top result until the document d_j is achieved.

The computational efficiency of the proposed approach is measured in terms of the computational complexity.

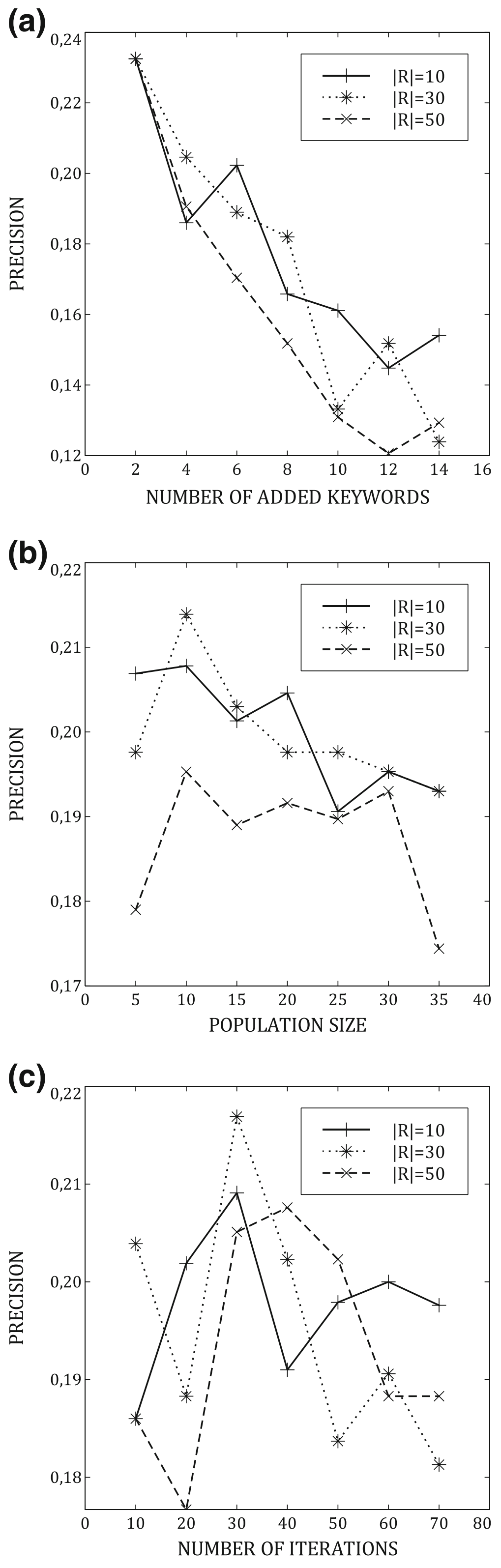


Fig. 3 The precision values achieved when fixing the FA parameters

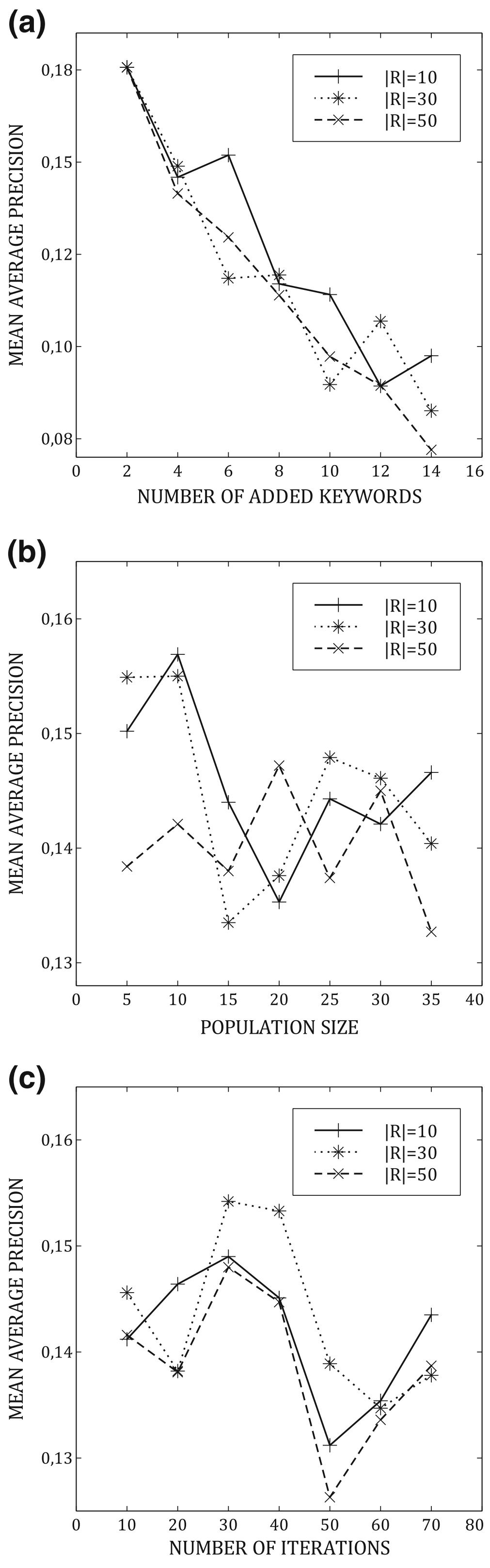


Fig. 4 The MAP values obtained when fixing the FA parameters

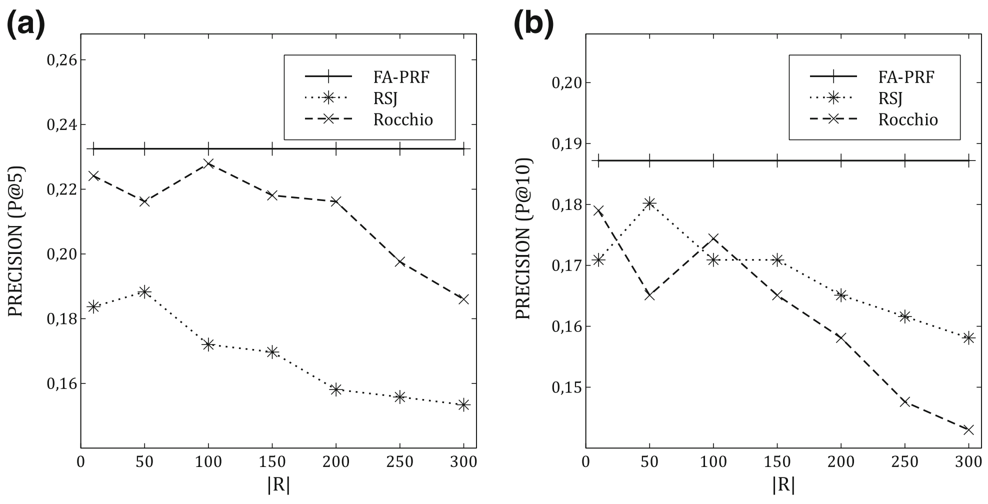


Fig. 5 Comparing the effectiveness of FA-PRF, RSJ and Rocchio in terms of precision

Results

Before proceeding to compare the performance of the proposed approach with RSJ and Rocchio methods, we first fix the following FA parameters: $|\grave{Q}|$, N, and T, which respectively represent the solution length, the population size, and the maximum number of generations. In this preliminary experiment, each of these parameters is varied individually while keeping the remaining parameters constant. The number of pseudo-relevant documents $|R|$ is tuned at 10, 30 and 50. The parameters γ, θ, and α_0 are set to 1, 0.95 and 1, respectively; following the guidelines proposed in several studies of the literature [47]. We use both the precision values after retrieving 5 documents (P@5) and the mean average precision metric to determine the best value for each parameter. Figure 3a, b and c present the precision values obtained when fixing the parameters.

In Fig. 3a, we set N and T to 5 and 10, and systematically varied $|\grave{Q}|$ in terms of the number of keywords from 2 to 14 in increments of 2. As illustrated in 3a, the appropriate value for $|\grave{Q}|$, which brings the best performance, is 2. In Fig. 3b, we fix $|\grave{Q}|$ and T to 4 and 10, and varied the size of population N from 5 to 35 in increments of 5. In this experiment, we set $|\grave{Q}|$ to 4 rather than 2 because the results obtained with $|\grave{Q}| = 2$ are identical. According to Fig. 3b, the best value for N is 10. In Fig. 3c, $|\grave{Q}|$ and N are respectively tuned to 4 and 10 while the maximum number of iterations T are varied from 10 to 70 in increments of 10. It can be seen from Fig. 3c that the appropriate value for T is 30. The MAPs results are shown in Fig. 4. Similar to the above results, the best mean average precision results are reached when $|\grave{Q}| = 2$, $N = 10$ and $T = 30$.

In the next set of experiments, we evaluate and compare the results obtained by the FA-PRF with those obtained by RSJ and Rocchio methods. In these tests, the length of $\grave{Q}$ is set to 2, while the number of pseudo-relevant documents is varied from 10 to 300 in increments of 50. The parameters N and T of the Firefly Algorithm are set to 10 and 30, respectively. Figure 5 shows the precision values of each method after retrieving 5 and 10 documents.

Through Fig. 5a, we can see obviously that the proposed approach produces the highest P@5 values and achieves significant improvement over RSJ and Rocchio methods, e.g. for $|R| = 10$, there is an improvement of +26.56 % over RSJ and +3.74 % over Rocchio. Similarly, precision at 10 retrieved documents improves from 0.1709 (+9.53 %) and 0.1790 (+4.58 %) to 0.1872 over RSJ and Rocchio, respectively. In terms of mean average precision, we observe that the FA-PRF reports the best results against RSJ and Rocchio methods (Fig. 6), e.g. for $|R| = 10$, the proposed approach outperforms RSJ and Rocchio around +15.82 % and +5.97 %, respectively.

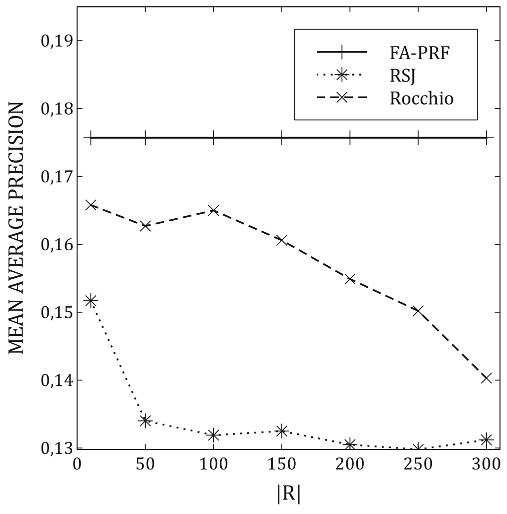


Fig. 6 Mean average precision results of the FA-PRF, the RSJ and Rocchio methods

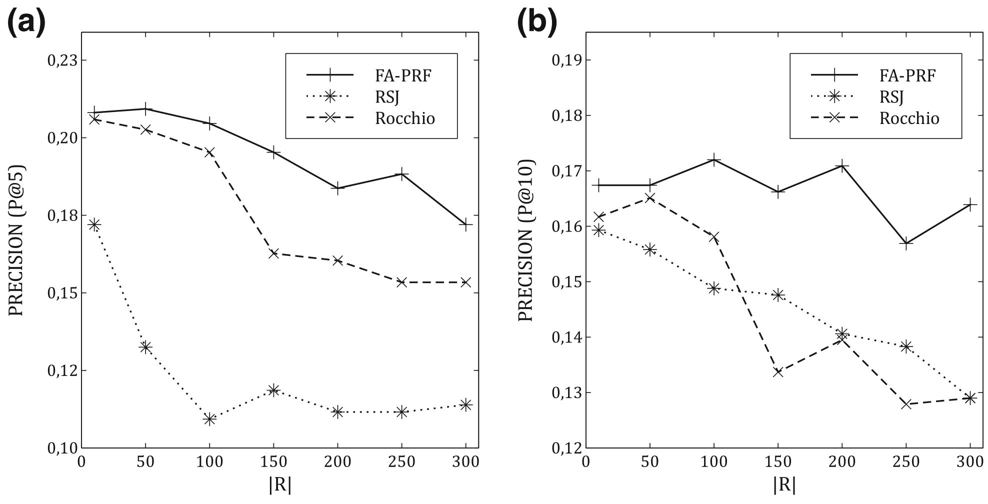


Fig. 7 Effectiveness comparison of the proposed approach to the RSJ and Rocchio methods in terms of precision

In the next phase of testing, the number of words in $|\grave{Q}|$ is tuned at 4, while the number of pseudo-relevant documents $|R|$ is varied from 10 to 300 in increments of 50. The Firefly Algorithm parameters N and T are set to 10 and 30, respectively. Figure 7 presents the precision values achieved by the proposed approach, RSJ and Rocchio after retrieving 5 and 10 documents.

From Fig. 7, we can observe that the results obtained with $|\grave{Q}| = 2$ are better than the results obtained with $|\grave{Q}| = 4$. Moreover, it is obviously seen from Fig. 7a that the proposed approach manages to enhance the search results against Rocchio and this enhancement is more significant against RSJ, e.g. for $|R| = 50$, the proposed approach shows improvement of +57.96 % over RSJ and +3.30 % over Rocchio after retrieving 5 documents. It can be also observed from Fig. 8 that the proposed approach achieves the highest values in terms of mean average precision, e.g. for $|R| = 50$, there is an improvement of +7.75 % over RSJ and +4.17 % over Rocchio.

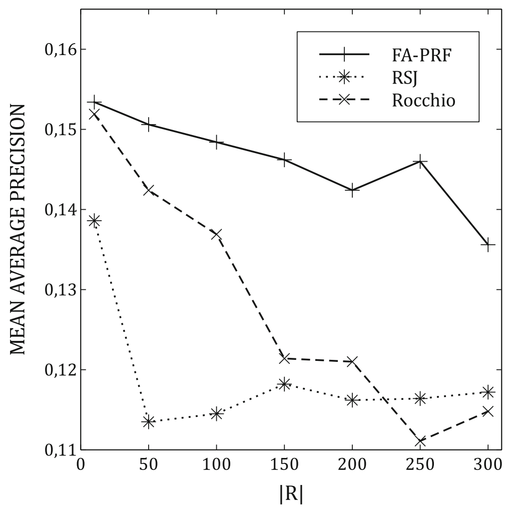


Fig. 8 MAP results of FA-PRF, RSJ and Rocchio

Computational complexity

The complexity of the proposed approach depends on both the complexity of the fitness function and the expected number of FA iterations. The computing complexity of the fitness function is in the order of $O(|R||\grave{Q}|N)$. Hence, the overall complexity of the algorithm is defined by $O(|R||\grave{Q}|NT)$. On the other hand, the complexity of both RSJ and Rocchio methods is determined by the number of pseudo-relevant documents and the number of terms in $|\grave{V}|$. The size of the set $|\grave{V}|$ can be expressed by the number of pseudo-relevant documents and the average of document length $avdl$. Thus, the total complexity of RSJ and Rocchio is given by $O(|R|^2 avdl)$. The computational complexity analysis of the proposed approach in comparison to RSJ and Rocchio is shown in Table 2. Following the above experiments, in this comparison, the size of R is varied from 10 to 300 and the length of $\grave{Q}$ is varied from 2 to 4. The remaining parameters N and T are set to 10 and 30, respectively.

From Table 2, we can observe that the FA-PRF achieves better results in all cases and gives great enhancement over RSJ and Rocchio. From the above experiments, we can conclude that the proposed FA for PRF succeeds to improve the ranking of relevant documents and make them in the first place. The precision values of the FA-PRF, after retrieving 5 documents, show a clear and significant superiority over RSJ and Rocchio methods. In terms of computational cost, the FA-PRF shows great improvement over RSJ and Rocchio.

Table 2 Efficiency comparison of FA, RSJ and Rocchio in terms of the computational complexity

$\|R\|$	10	50	100	150	200	250	300
FA-PRF	6K	30K	60K	90K	120K	150K	180K
$(\|\grave{Q}\| = 2)$	−51.7 %	−90.2 %	−95.1 %	−96.7 %	−97.5 %	−98.1 %	−98.4 %
FA-PRF	12K	60K	120K	180K	240K	300K	360K
$(\|\grave{Q}\| = 4)$	−1.6 %	−80.3 %	−90.2 %	−93.4 %	−95.1 %	−96.1 %	−96.7 %
Baseline	12.2K	305K	1.22M	2.76M	4.88M	7.63M	10.98M

Conclusion

In this paper, we propose a new approach based on Firefly Algorithm to enhance the retrieval effectiveness of pseudo-relevance feedback while maintaining low computational cost. A major contribution of our work is to use a Firefly Algorithm to find the best expanded query among a set of expanded query candidates.

We thoroughly evaluate our approach using MEDLINE dataset. The experimental results demonstrate that the proposed Firefly Algorithm for PRF succeeds to improve the ranking of the relevant documents and yields a substantial improvement over the baseline in terms of precision and mean average precision. From the point of view of computational complexity, the proposed approach is more efficient compared to RSJ and Rocchio methods.

In the future, we will try to design a parallel version of the Firefly Algorithm and to implement it in order to further improve both the retrieval effectiveness and efficiency. It is also interesting to test and experiment the proposed approach on other medical datasets. Another possible research direction is to extend our work to other areas of search applications, such as Twitter search.

References

1. Ahmad, F., and Kondrak, G., Learning a spelling error model from search query logs. In: Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing, pp. 955–962. Association for Computational Linguistics, 2005.
2. Alweshah, M., and Abdullah, S., Hybridizing firefly algorithms with a probabilistic neural network for solving classification problems. *Appl. Soft Comput.* 35:513–524, 2015.
3. Attardi, G., Atzori, L., and Simi, M., Index expansion for machine reading and question answering. In: CLEF 2012 Evaluation Labs and Workshop, Online Working Notes, 2012.
4. Baykasoğlu, A., and Ozsoydan, F. B., An improved firefly algorithm for solving dynamic multidimensional knapsack problems. *Expert Systems with Applications* 41(8):3712–3725, 2014.
5. Bernardini, A., Carpineto, C., and D'Amico, M., Full-subtopic retrieval with keyphrase-based search results clustering. In: Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, pp. 206–213. IEEE, 2009.
6. Bindal, A. K., and Sanyal, S., Query optimization in context of pseudo relevant documents. In: 3rd Italian Information Retrieval (IIR) workshop, 2012.
7. Blum, C., and Roli, A., Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.* 35(3):268–308, 2003.
8. Brajevic, I., and Tuba, M., Cuckoo search and firefly algorithm applied to multilevel image thresholding. In: Cuckoo Search and Firefly Algorithm, pp. 115–139: Springer, 2014.
9. Cao, G., Nie, J. Y., Gao, J., and Robertson, S., Selecting good expansion terms for pseudo-relevance feedback. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 243–250. ACM, 2008.
10. Carpineto, C., and Romano, G., Concept Data Analysis: Theory and Applications John Wiley & Sons, 2004.
11. Carpineto, C., and Romano, G., A survey of automatic query expansion in information retrieval. *ACM Comput. Surv.* 44(1):1–50, 2012.
12. Chen, Q., Li, M., and Zhou, M., Improving query spelling correction using web search results. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 181–189. Association for Computational Linguistics, 2007.
13. Crestani, F., Application of spreading activation techniques in information retrieval. *Artif. Intell. Rev.* 11(6):453–482, 1997.
14. Deep, K., and Bansal, J. C., Mean particle swarm optimisation for function optimisation. *International Journal of Computational Intelligence Studies* 1(1):72–92, 2009.
15. Dillon, J. V., and Collins-Thompson, K., A unified optimization framework for robust pseudo-relevance feedback algorithms. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, pp. 1069–1078. ACM, 2010.
16. Eisenstein, J., Xing, E. P., Smith, N. A., and O'Connor, B., Mapping the geographical diffusion of new words. Tech rep, 2012.
17. Gao, K., Zhang, Y., Zhang, D., and Lin, S., Accurate off-line query expansion for large-scale mobile visual search. *Signal Process.* 93(8):2305–2315, 2013.
18. Jouglet, A., and Carlier, J., Dominance rules in combinatorial optimization problems. *Eur. J. Oper. Res.* 212(3):433–444, 2011.
19. Karthikeyan, S., Asokan, P., and Nickolas, S., A hybrid discrete firefly algorithm for multi-objective flexible job shop scheduling problem with limited resource constraints. *Int. J. Adv. Manuf. Technol.* 72(9-12):1567–1579, 2014.
20. Kennedy, J., Particle swarm optimization. In: Encyclopedia of Machine Learning, pp. 760–766: Springer, 2011.
21. Kennedy, J., Kennedy, J. F., Eberhart, R. C., and Shi, Y., Swarm Intelligence Morgan Kaufmann, 2001.
22. Kirkpatrick, S., Optimization by simulated annealing: Quantitative studies. *J. Stat. Phys.* 34(5-6):975–986, 1984.

23. Lee, A., and Chau, M., The impact of query suggestion in e-commerce websites. In: E-Life: Web-Enabled Convergence of Commerce, Work, and Social Life 10th Workshop on E-Business, WEB 2011, pp. 248–254, 2011.
24. Lee, K. S., and Croft, W. B., A deterministic resampling method using overlapping document clusters for pseudo-relevance feedback. *Inf. Process. Manag.* 49(4):792–806, 2013.
25. Lei, X., Wang, F., Wu, F. X., Zhang, A., and Pedrycz, W., Protein complex identification through markov clustering with firefly algorithm on dynamic protein–protein interaction networks. *Inf. Sci.* 329:303–316, 2016.
26. Leturia, I., Gurrutxaga, A., Areta, N., Alegria, I., and Ezeiza, A., Morphological query expansion and language-filtering words for improving basque web retrieval. *Lang. Resour. Eval.* 47(2):425–448, 2013.
27. Long, N. C., and Meesad, P., An optimal design for type–2 fuzzy logic system using hybrid of chaos firefly algorithm and genetic algorithm and its application to sea level prediction. *J. Intell. Fuzzy Syst.* 27(3):1335–1346, 2014.
28. Lv, Y., Zhai, C., and Chen, W., A boosting approach to improving pseudo-relevance feedback. In: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 165–174. ACM, 2011.
29. Martinez, D., Otegi, A., Soroa, A., and Agirre, E., Improving search over electronic health records using umls-based query expansion through random walks. *J. Biomed. Inform.* 51:100–106, 2014.
30. Melucci, M., A basis for information retrieval in context. *ACM Trans. Inf. Syst.* 26(3):14:1–14:41, 2008.
31. Miao, J., Huang, J. X., and Ye, Z., Proximity-based rocchio's model for pseudo relevance. In: Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 535–544. ACM, 2012.
32. Mitchell, M., An Introduction to Genetic Algorithms MIT press, 1998.
33. Robertson, S., and Zaragoza, H., The Probabilistic Relevance Framework: BM25 and Beyond Now Publishers Inc, 2009.
34. Robertson, S. E., and Jones, K. S., Relevance weighting of search terms. *J. Am. Soc. Inf. Sci.* 27(3):129–146, 1976.
35. Robertson, S. E., Walker, S., Beaulieu, M., Gatford, M., and Payne, A., Okapi at trec-4. In: Proceedings of the 4th Text Retrieval Conference, pp. 73–97, 1995.
36. Rocchio, J. J., Relevance feedback in information retrieval. In: Salton, G. (Ed.) The SMART Retrieval System - Experiments in Automatic Document Processing, pp. 313–323, 1971.
37. Sahlgren, M., An introduction to random indexing. In: Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE, 2005.
38. Subramaniam, L. V., Roy, S., Faruquie, T. A., and Negi, S., A survey of types of text noise and techniques to handle noisy text. In: Proceedings of The 3rd Workshop on Analytics for Noisy Unstructured Text Data, pp. 115–122. ACM, 2009.
39. Sun, H. m., A study of the features of internet english from the linguistic perspective. *Tex. Stud. Lit. Lang.* 1(7):98, 2010.
40. Véronis, J., Hyperlex: Lexical cartography for information retrieval. *Comput. Speech Lang.* 18(3):223–252, 2004.
41. Williams, H. E., and Zobel, J., Searchable words on the web. *Int. J. Digit. Libr.* 5(2):99–105.
42. Wong, S. K., Ziarko, W., Raghavan, V. V., and Wong, P. C., On modeling of information retrieval concepts in vector spaces. *ACM Trans. Database Syst.* 12(2):299–321, 1987.
43. Xie, H., Zhang, Y., Tan, J., Guo, L., and Li, J., Contextual query expansion for image retrieval. *IEEE Trans. Multimedia* 16(4):1104–1114, 2014.
44. Yang, X. S., Nature-Inspired Metaheuristic Algorithms Luniver Press, 2008.
45. Yang, X. S., Firefly algorithms for multimodal optimization. In: Proceedings of the 5th International Conference on Stochastic Algorithms: Foundations and Applications, pp. 169–178, 2009.
46. Yang, X. S., Nature-Inspired Metaheuristic Algorithms: 2nd Edn Luniver Press, 2010.
47. Yang, X. S., Nature-inspired optimization algorithms Elsevier, 2014.
48. Ye, Z., and Huang, J. X., A simple term frequency transformation model for effective pseudo relevance feedback. In: Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 323–332. ACM, 2014.
49. Zeng, Y., Zhang, Z., and Kusiak, A., Predictive modeling and optimization of a multi-zone hvac system with data mining and firefly algorithms. *Energy* 86:393–402, 2015.