Expert Systems With Applications 94 (2018) 126–136

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Bees swarm optimization guided by data mining techniques for document information retrieval

Youcef Djenouri[a,*], Asma Belhadi[b], Riadh Belkebir[c]

[a] Ulsan National Institute of Science and Technology, Ulsan, South Korea
[b] RIMA Lab, USTHB, Algiers, Algeria
[c] LRIA Lab, USTHB, Algiers, Algeria

ARTICLE INFO

Article history:
Received 27 July 2017
Revised 15 October 2017
Accepted 16 October 2017
Available online 5 November 2017

Keywords:
Information retrieval
Data mining
Big data
BSO algorithm
Bio-inspired methods

ABSTRACT

This paper explores advances in the data mining field to solve the fundamental Document Information Retrieval problem. In the proposed approach, useful knowledge is first discovered by using data mining techniques, then swarms use this knowledge to explore the whole space of documents intelligently. We have investigated two data mining techniques in the preprocessing step. The first one aims to split the collection of documents into similar clusters by using the K-means algorithm, while the second one extracts the most closed frequent terms on each cluster already created using the DCI_Closed algorithm. For the solving step, BSO (Bees Swarm Optimization) is used to explore the cluster of documents deeply. The proposed approach has been evaluated on well-known collections such as CACM (Collection of ACM), TREC (Text REtrieval Conference), Webdocs, and Wikilinks, and it has been compared to state-of-the-art data mining, bio-inspired and other documents information retrieval based approaches. The results show that the proposed approach improves the quality of returned documents considerably, with a competitive computational time compared to state-of-the-art approaches.

1. Introduction

Let $D = \{d_1, d_2 \dots d_m\}$ be the set of m documents and let $T = \{t_1, t_2 \dots t_n\}$ be the set of n terms. Each document is composed of the subset of terms included in T. The user's request Req is represented by the set of terms. The Document Information Retrieval (DIR) is the process of finding relevant documents, according to the user's request Req, from a collection of documents D (Salton & McGill, 1986). The classical approach for DIR problem scans all documents and computes the score between each one and the given user's request. The ranking function is then used to keep only the relevant documents (Salton & McGill, 1986). This process has a polynomial complexity, however, when dealing with a large number of documents, the runtime will be extremely high.

In the last decade, many data mining based approaches have been used as preprocessing step to solve the IR problem by discovering knowledge from collections of documents. The extracted knowledge is then used for the solving step. Indeed, these approaches split the collection of documents into many groups. After that, the search process is performed on each group using high-performance computing tools. Two main approaches have been proposed. The works reported in Steinbach, Karypis, Kumar et al. (2000), Cai, He, and Han (2005), Hammouda and Kamel (2004), Hatamlou, Abdullah, and Nezamabadi-Pour (2012), Mahdavi and Abolhassani (2009) use the K-means algorithm (MacQueen, 1965) to group the clusters into k disjoints clusters, where each group contains similar documents. Whereas, the works reported in Beil, Ester, and Xu (2002), Fung, Wang, and Ester (2003), Yu, Searsmith, Li, and Han (2004), Menezes et al. (2010) use Frequent Patterns Mining (FPM) (Zaki & Hsiao, 2002) to discover the frequent terms in the collection. Afterward, the top k frequent patterns are used to create the groups of documents. These approaches decompose the initial problem into many subproblems, where each of which could be solved independently. However, the runtime of the DIR problem is still prohibitive, especially when dealing with massive number of documents existing in the World Wide Web (WWW).

More recently, some bio-inspired approaches have been developed to handle the DIR problem. The evolutionary-based approaches (Benetos, Dixon, Giannoulis, Kirchhoff, & Klapuri, 2013; Blei, 2012; Fan, Gordon, & Pathak, 2004; Lin, Chen, & Wu, 2014) and the swarm intelligence approaches (Buscher, Dengel, Biedert, & Elst, 2012; Collett, Graham, Harris, & Hempel-de Ibarra, 2006; Picard, Revel, & Cord, 2012; Zhang, Mei, Liu, Tao, & Zhou, 2011) proposed in the literature transform the DIR problem into an op-

* Corresponding author.
E-mail address: ydjenouri@unist.ac.kr (A. Belhadi).

https://doi.org/10.1016/j.eswa.2017.10.042
0957-4174/

timization problem and consider the collection of documents as a space of solutions. The intensification and diversification strategies of these approaches allow finding an approximate subset of documents in a reasonable computational time. Moreover, the communication between the swarms permits to find high-quality solutions compared to the evolutionary ones. Nevertheless, the process is still stochastic, and when the space of solutions is large, the swarms are disoriented, which degrades the quality of the final subset of documents returned by the swarms. Motivated by the success of BSO in dealing with many optimization problems, this paper aims to develop and investigate a new BSO algorithm for solving the DIR problem. The proposed algorithm exploits the information extracted in the pre-processing step by using both clustering and closed frequent itemset mining techniques to guide the swarms in the exploration of the space of documents.

The main contributions of this paper are the following:

- The proposed approach improves the preprocessing step of existing information retrieval approaches by applying both clustering and closed frequent itemset mining to extract knowledge from a collection of documents. Indeed, the K-means algorithm is first applied to generate k clusters; then FPM is performed on each cluster to extract the frequent patterns between highly correlated documents. To reduce the dimensionality of the frequent patterns, the closed algorithm (Lucchese, Orlando, & Perego, 2006) is used to extract only the most frequent patterns.
- Two bees swarm optimization algorithms are proposed by developing different heuristics which allow the swarms to well explore the search space. During the search process, the bees are guided by the knowledge extracted in the preprocessing step.
- To validate the runtime performance and the quality of returned documents, extensive experiments have been done on medium, large and big collections. The results show that our approach outperforms the data mining based approach when using large collections. Moreover, it outperforms the bio-inspired and other document information retrieval based approaches in terms of solution's quality using big collections and it is very competitive to them regarding the runtime process.

The remainder of the paper is organized as follows. Section 2 discusses the most used data mining and bio-inspired-based approaches for the document information retrieval problem. Section 3 presents our main proposition regarding the information retrieval problem followed by a detailed description of each components of our framework in Sections 3.1.1 and 3.2. In Section 4, the performance evaluation of the proposed approach is presented. Finally, Section 5, concludes the work by giving some remarks and future perspectives.

2. Related work

The DIR problem has been tackled in the literature using several approaches (Blei, Ng, & Jordan, 2003; Croft & Harper, 1979; Lafferty & Zhai, 2017; Ponte & Croft, 1998; Wei & Croft, 2006). In this work, we try to investigate the efficiency of using only data-mining and bio-inspired approaches for the DIR problem since the aim of these approaches is to provide approximate solutions in a reasonable time complexity. In the following, we present some data mining and bio-inspired approaches for the DIR problem.

2.1. Data mining-based approaches for DIR

In recent literature, some data mining-based approaches have been proposed to improve the information retrieval process. In the following, we discuss the most used approaches for DIR problem.

Beil et al. (2002) developed the first ARM algorithm for information retrieval called HFTC (Hierarchical Frequent Term-based Clustering). The algorithm starts by extracting frequent itemsets using Apriori algorithm. The itemsets are modeled by the terms of the collection. Then, the most frequent itemsets are considered as clusters where each frequent itemset is one cluster containing the documents that verify it.

Fung et al. (2003) proposed a new approach called FIHC (Frequent Itemset-based Hierarchical Clustering). The frequent itemsets are used to construct the hierarchical tree representing the collection. Using the frequents itemsets in the classification of the documents, the experiments reveal that the execution time of the user's requests has been reduced.

Yu et al. (2004) presented a new mining rules algorithm called TDC (Topic Document Clustering) to improve the quality of the classification of documents. It generates dynamically the different topics of the document's base using only the closet frequent itemsets. This can reduce the execution time comparing with the FIHC algorithm. TDC uses an intelligent structure that allows constructing the different links between each itemset of size k with the itemsets of size k-1 hierarchically. This approach gives high precision, but it could cause an overlap between clusters when the terms of the documents are highly linked.

In Babashzadeh, Daoud, and Huang (2013), the authors proposed a new algorithm for text processing called ARMIR (Association Rule Mining for Information Retrieval). In this approach, a given request is modeled by a set of concepts where the relations between concepts of the same request are determined by an association rules mining process. In Veloso, Almeida, Gonçalves, and Meira Jr (2008), a ranking function is proposed to sort the documents. A rules mining process is applied for training documents. The consequent part of each rule represents the scores of documents containing the terms of the antecedent part of this rule.

Another algorithm called LATRE (Lazy Associative Tag REcommender) has been proposed in Menezes et al. (2010). It extracts the association rules from the training set of documents. The obtained rules provide the keywords of a given object. This algorithm handles the pretreatment phase of information retrieval process. Furthermore, the set of relevant tags is associated with each document. This operation reduces the response time of different requests efficiently.

In Zhong, Li, and Wu (2012), the authors proposed PTM (Pattern Taxonomy Mining) algorithm to improve the comprehension of the user's request using a patterns mining algorithm. The pattern taxonomy of terms is discovered by applying the closed algorithm in the training set of documents. This technique reduces the noise between the user's request and the set of terms in the collection of documents.

In Joachims (2002) a classifier-based approach called SVMIR (Support Vector Machine for Information Retrieval) has been proposed. A Support Vector Machine learning algorithm is developed using clickthrough existing data. The results reveal that this algorithm creates groups of users according to their preferences which enhances the performance of the given information retrieval process automatically.

In Lan, Tan, Su, and Lu (2009), a new supervised term weighting approach called KNNIR (K-Nearest Neighbors for Information Retrieval) is proposed. It combines the support vector model representation and KNN (K-Nearest Neighbors) algorithm to compute the weight of each term in the given documents. The weights of the training terms are first computed, then the KNN classifier is launched to compute the score between each test term and the training terms.

2.2. Bio-inspired approaches for DIR problem

Several bio-inspired approaches have been developed to reduce the computational time of the DIR problem. For the evolutionary-based approaches we cite Fan et al. (2004), Blei (2012), Benetos et al. (2013), Lin et al. (2014), Jung (2012) and for the swarm intelligence approaches we cite Picard et al. (2012), Zhang et al. (2011), Buscher et al. (2012), Collett et al. (2006), Diaz-Aviles, Nejdl, and Schmidt-Thieme (2009), Khennak and Drias (2016).

In Fan et al. (2004), an automatic ranking strategy called GP-IR (Genetic Programming for Information Retrieval) has been suggested. Indeed, a GP-based method is investigated in context-specific ranking function aiming to discover the weights of terms automatically. The ranking function is first modeled by a tree structure that represents one individual in the solution space. Afterward, a GP approach is established for ranking function discovery by proposing two new crossover and mutation operators. In Lin et al. (2014), a GAFS (Genetic Algorithm for Feature Selection) approach is proposed, it combines both feature selection and stochastic genetic algorithm to deal with the information retrieval problem. The feature selection is performed to select the best terms from the original collection using the genetic algorithm. Each chromosome represents a set of applicable features. Afterward, a classical GA operators are used such as crossover, mutation, and evolution. Jung (2012) developed EQS (Evolutionary Query Sampling) algorithm to retrieve large documents in multiple information sources. It is an evolutionary approach for automatically splitting the user's query according to the given sources. Evolution operators used in this work allow giving better matching between the user's request and the information sources.

BSO has been successfully used in many applications and domains such as numerical function optimization (Akbari, Mohammadi, & Ziarati, 2010), dynamic economic dispatch (Niknam & Golestaneh, 2013), association rule mining (Djenouri, Drias, & Chemchem, 2013), text categorization (Belkebir & Guessoum, 2013). For the DIR problem, Collett et al. (2006) proposed BSO-IR (Bees Swarm Optimization for Information Retrieval). In this algorithm, each bee of the colony tries to explore its region by applying the intensification strategy. After that, the communication between bees is done to extract the best documents according to the given user's request. In Diaz-Aviles et al. (2009), a SwarmRank algorithm is suggested for optimizing the retrieval quality of ranking function. The particle swarm optimization algorithm is used to learn with Mean Average Precision (MAP), a widely used evaluation measure in DIR problem. In Khennak and Drias (2016), a firefly metaheuristic is applied named FLYIR (fireFLY for Information Retrieval) to deal with DIR problem in the medical context. Each firefly tries to improve its position locally by using its light intensity. At each pass of the algorithm, the communication between the fireflies is done by the bioluminescent concept to follow the best firefly having the best current documents. Experimental results show the effectiveness of this approach on MEDLINE instance compared to the existing approaches for medical information retrieval.

3. System design

In this section, we start by presenting the general BSO-IR approach then we explain the system design of our approach.

Swarm intelligence is based on the collective behavior of decentralized, self-organized systems. More and more researchers are interested in this new existing way of achieving a form of artificial intelligence. Modeling the behavior of social insects, such as ants and bees, and using these models for search and problem-solving are the context of the emerging area of swarm intelligence. Bees are among the well-studied social insects (Seeley, Camazine, & Sneyd, 1991). The experience reported in Seeley et al. (1991) showed that when several food sources are reached by the bees colony, the latter exploit the source having highest food concentration. This mechanism is made by dance to allow other bees follow the direction of the source with greater concentration. BSO (Karaboga, 2005) (Bees Swarm Optimization) is inspired by the foraging bees behavior. It is based on a swarm of artificial bees co-operating to solve a problem. Several extensions of BSO have been proposed to solve complex and real problems, (data mining Djenouri, Bendjoudi, Mehdi, Nouali-Taboudjemat, and Habbas, 2015, Zhang, Ouyang, and Ning, 2010, constraint programming Djenouri, Habbas, and Aggoune-Mtalaa, 2016 and information retrieval Collett et al., 2006). In this paper, we are interested in document information problem. In the sequel, we discuss the main idea of BSO-IR.

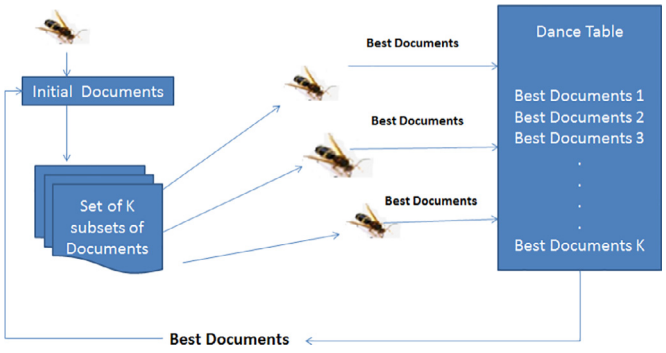

Fig. 1. Bees swarm optimization for documents information retrieval.

A bee called *BeeInit* represents a potential set of documents. A space of documents is then determined from *BeeInit* using a well-defined diversification strategy. It consists in determining equidistant points in a large documents space; these points are obtained by modifying recursively the solution of the *BeeInit* using the flip parameter. This allows to explore the documents space. Each bee is then assigned to one point and performs the intensification strategy. This strategy allows to explore deeply one region of the documents space. It is established by modifying recursively only one bit of solution of each bee. After the bees have accomplished their search, they communicate to their congeners their best results through a table called *Dance*. The best solution stored in this table is taken as the *BeeInit* for the next iteration. The process is iterated until the number of iterations reaches a certain number of iterations or the number of relevant documents is reached by the bees. The framework of BSO-IR is shown in Fig. 1.

Now we give an overview of our proposed approach which performs on two steps. A more detailed discussion of each step will be provided in Sections 3.1–3.3 .

1. **Preprocessing** : The aim here is to exploit the power of data mining techniques for extracting relevant knowledge, which will be used later by the swarms. It performs on two main stages. The first stage is to split the collection of documents into several clusters, where each cluster can be viewed as a subset of documents of the whole collection of documents. The set of terms shared by two clusters is called a separator set. An interesting decomposition approach is to minimize the size of separator sets while putting into the same cluster documents that are connected, i.e., documents that share the maximum number of terms. In this paper, we propose a partitioning-based strategy by adapting K-means algorithm for decomposing the whole collection of documents which will be presented in Section 3.1.1. As a result of this stage, the clusters are created, each of which contains a subset of documents highly con-

nected. Afterward, a Frequent Patterns Mining (FPM) approach is applied to each cluster of documents. In this context, closed frequent patterns are discovered on each cluster. We also propose the adaptation of DCI_CLOSED algorithm to discover the closed frequent patterns of each group of documents highly connected which will be presented in Section 3.1.2.

2. **Search Process**: At the arrival of a new user's request, the swarms use the closed frequent patterns already extracted on each cluster to explore the solution's space. In another term, the swarms are guided by the closed frequent patterns discovered during the preprocessing step when exploring the solution's space. Two strategies are developed to show the usefulness of the discovered patterns for guiding the search of swarms. It is worth to mention that several swarm-based approaches can be investigated here. However, this investigation is out of the scope of this work. A simple Bees Swarm Optimization method is used for the searching process which is already presented in Section 3.

3.1. Preprocessing step

This step includes two stages (document decomposition and closed frequent patterns determination) which are given as follows:

3.1.1. Documents decomposition

In this first stage, K-means is used to decompose a given collection of documents without loss of generality. It is one of the simplest unsupervised learning algorithms for the clustering problems. It defines a simple and easy procedure to divide a given data set into a certain number of clusters, say k clusters, fixed a priori. The main idea is to define k centroids, one for every cluster. The centroids should be placed in a cunning way since the clustering result depends on their location in the clusters. To optimize the outcomes, it is judicious to place them as far as possible from each other. The next step is to take each point belonging to a given data set and associate it with the nearest centroid. When no point is pending, the first step is completed, and an early grouping is performed. At this stage, k new centroids are needed to be re-calculated for the new clusters that result from the previous step, and the process should be iterated. The latter stops when no more changes of the clusters are observed, i.e., when centroids stop moving. In the literature, several adaptations of K-means of grouping documents have been proposed. The most cited works are Steinbach et al. (2000), Cai et al. (2005), Hammouda and Kamel (2004), Hatamlou et al. (2012), and Mahdavi and Abolhassani (2009). In this work, we used the adaptation explored in Mahdavi and Abolhassani (2009) for its simplicity and efficiency.

The general algorithm of k-means could be represented as follows.

$$J = \sum_{j=1}^{K} \sum_{n \in S_j} |x_n - \mu_j|^2 \quad (1)$$

Where:

- x_n is a vector that represents the nth data point
- μ_j is the centroid of the data points in S_j.

The K-means algorithm aims at partitioning (or clustering) N data points into K disjoint subsets S_j. Each subset S_j contains N_j data points. The aim then is to minimize the sum-of-squares criterion.

First, the data points are assigned randomly to the K clusters and the centroid is computed for each cluster. Then, every point is assigned to the cluster whose centroid is the closest to that point. These two steps are repeated until there is no further assignment of the data points to the clusters. In the following, we present the adaptation of K-means to our problem.

- **Documents Representation** The documents are represented using the vector space model. Each document d is represented by a vector $\{w_1, w_2 \ldots w_n\}$, where w_i represents the weight of the term t_i. The term weight value represents the significance of this term in a document which is computed using the well-known $TF - IDF$ (Term Frequency with Inverse Document Frequency) formula Blei et al. (2003) as follows:

  $$w_{ij} = tf_{ji} \times idf_{ji} \quad (2)$$

  Where w_{ij} represents the weight of the term i in the document j.
  tf_{ji} is the number of occurrences of term i in the document j.
  $idf_{ji} = log_2(m/df_{ji})$ such df_{ji} indicates the term frequency in the collections of m documents.
- **Similarity Computation** The similarity measure between two documents d_i and d_j is computed using the cosine correlation measure Tata and Patel (2007) given by:

  $$\cos(d_i, d_j) = \frac{d_i^t d_j}{|d_i||d_j|} \quad (3)$$

  Where
  $d_i^t d_j$ denotes the dot-product of the two document vectors d_i and d_j.
  $|d_i|$ indicates the length of the vector d_i, i.e., the number of terms having weights non null in the document d_i.
- **Centroids Updating** The centroid updating is calculated as:

  $$g_i = \frac{1}{|C_i|} \sum_{j=1}^{|C_i|} d_j \quad (4)$$

  Where:
  g_i is the new center of the cluster C_i.

  Moreover, Algorithm 1 presents the use of K-means algorithm for the decomposition of a collection of documents.

Algorithm 1 K-means for clustering documents.

```
Input:
D = {d_1, d_2 ... d_m}: The collection of the documents.
T = {t_1, t_2 ... t_n}: The set of n terms.
g_i : gravity center of group i
Output:
C = {C_1, C_2 ..., C_k}: The set of k clusters.
Choosek initial centers g_1, g_2 ..., g_k.
DocumentRepresentation(D) /*using EQ. 2*/.
for eachdocument d_i do
    min ← cos(d_i, g_1).
    r ← 1.
    for j=2 to k do
        distance← cos(d_i, g_j) /*using EQ. 3*/.
        if distance < min then
            min ← distance
            r← j
        end if
    end for
    Affect d_i to C_r
end for
if any document changes the state of a group then
    Stop and exit
else
    Calculate the new center gravity g_ifor all C_i using EQ. 4.
    Go to 5.
end if
return C
```

3.1.2. Closed frequent patterns mining

Frequent Patterns Mining (FPM) aims to discover interesting patterns from the given input data. Traditional FPM usually generates a high number of patterns, especially when the minimum support is low. Other approaches allow reducing the dimensionality of the resulted frequent patterns significantly. Among them, we mention the closed frequent patterns mining which extracts only the closed frequent patterns. A frequent pattern is called closed if no superset of this pattern with the same support exists.

The following definitions are needed to understand the closed frequent pattern problem.

Definition 3.1 (Pattern Definition)**.** Let $\mathcal{I} = \{1, \ldots, r\}$ be the set of items. A transaction database on $\mathcal{I}$ is a set $\mathcal{T} = \{t_1, \ldots, t_s\}$ such that each t_i is included in $\mathcal{I}$. Each t_i is called a transaction. We denote by $\| \mathcal{T} \|$ the sum of sizes of all transaction in $\mathcal{T}$, that is, the size of database $\mathcal{T}$. A set $P \subseteq \mathcal{I}$ is called pattern.

Definition 3.2 (Frequent Pattern)**.** For pattern P, a transaction including P is called an occurrence of P. The denotation of P, denoted by $\mathcal{T}(P)$ is the set of occurrences of P. $| \mathcal{T}(P) |$ is called the frequency of P, and denoted by $frq(P)$. For given constant $MinSup$, called a minimum support, pattern P is frequent if $frq(P) \geq MinSup$.

Definition 3.3 (Frequent Closed Patterns)**.** A pattern Y is a frequent closed pattern if it is frequent and there exists no proper superset $Y' \supset Y$ such that $sup(Y') = sup(Y)$.

Several algorithms for solving Closed FPM problem have been explored in the literature. Among them, we can cite (CHARM Zaki and Hsiao (2002), CLOSET Pei, Han, Mao et al. (2000), CLOSET+ Wang, Han, and Pei (2003) and DCI_CLOSED Lucchese et al. (2006)). The latter show its effectiveness compared to the existing Closed FPM approaches. It adopts several optimizations to save both space and time in enumerating the closed frequent patterns. In the following, this algorithm is adapted to discover closed frequent patterns from each cluster of documents. Indeed, the relative minimum support of each cluster of documents is computed based on the absolute minimum support of the collection and the ratio between the size of this cluster and the size of the whole collection. It is given by $MinSup_i = MinSup \times \frac{|C_i|}{m}$, where $MinSup_i$ is the relative minimum support of the cluster C_i, $MinSup$ is the absolute minimum support of the collection and m is the number of all documents in the collection. After calculating the minimum support of each cluster, the DCI_CLOSED algorithm is applied on each cluster of documents with the given relative minimum support. The closed frequent patterns of each document are stored in the vector called F_i. Moreover, Algorithm 2 presents the adaptation of DCI_Closed for mining the clusters of documents.

Algorithm 2 DCI_Closed for mining clusters of documents.

1: **Input**: $MinSup$: absolute minimum support.
   C: The set of clusters of documents.
   m: The number of all documents.
   k: The number of clusters.
2: **Output**: F: The setof all closed frequent terms.
3: **for** each cluster C_i **do**
4: $MinSup_i \leftarrow MinSup \times \frac{|C_i|}{m}$
5: $F_i \leftarrow$ DCI_CLOSED(C_i, $MinSup_i$)
6: **end for**
7: F$\leftarrow \bigcup_{i=0}^{k} F_i$
8: **return** F

Algorithm 2has as input the set of clusters C, the absolute minimum support value $MinSup$, the number of all documents m and the number of all clusters k. The algorithm returns the set of all closed frequent terms F. First, the relative minimum $MinSup_i$ of each cluster C_i is calculated (see the Line 4). The set of the frequent terms of each cluster C_i is then discovered by applying the DCI_CLOSED algorithm (see the Line 5). Finally, the union of all F_i is affected to the set of all frequent terms of clusters F (see the Line 7).

3.2. BSOGDM-IR: Bees swarm optimization guided by data mining for documents information retrieval

In this section, two approaches that use the knowledge extracted in the previous section are proposed. They guide BSO for solving the document information retrieval problem.

3.2.1. BSOGDM1: first BSO algorithm guided by data mining

BSOGDM1 is similar to BSO, except that in BSOGDM1, each bee explores its region by taking into account the structural knowledge coming from the preprocessing step (the clusters and the closed frequent patterns). BSOGDM1 is described in Algorithm 3.

Algorithm 3 Algorithm BSOGDM1.

Input: The set of Clusters of Documents $C = \{C_1, C_2 \ldots C_k\}$
The set of Closed Frequent Patterns $F = \{F_1, F_2 \ldots F_k\}$
The User's Request *Req*.
Begin
1: **for** each cluster C_i **do**
2: $n_i \rightarrow$Matching(F_i, *Req*)
3: $P_i \rightarrow$AssignmentProbabilities(n_i, C_i, n).
4: **end for**
5: *BeeInit* $\leftarrow$ Initial_Solution
6: **while** non stop **do**
7: FindSearchRegion1(BeeInit, p, R_1, R_2, ..., R_p)
8: **for** each bee i **do**
9: $BestSol_i \rightarrow$UpdatingPositions(R_i, P_1, P_2, ..., P_k, C)
10: DanceTable $\leftarrow$ DanceTable $\cup$ $BestSol_i$
11: **end for**
12: *BeeInit* $\leftarrow$ DancingStep(DanceTable)
13: **end while**
14: **End**

According to this algorithm, the matching procedure is first applied to each cluster of documents and the user's request. This procedure returns the number of common terms between the closed frequent patterns of each cluster and the user's request. The probability of selecting the documents of each cluster is then assigned using the matching function and the number of all terms n. As in BSO, the initial bee *BeeInit* creates the initial solution; determines the set of regions of each bee using FindSearchRegion1; then each bee explores its region by considering the probabilities of all documents already computed by *AssignmentProbabilities* function. The best solution found by each bee is saved on the *dance* table. At the end of each pass of the algorithm, the best solution will be *BeeInit* for the next iteration. This process should be repeated until the maximum number of iterations is reached. In the sequel, we present the main components of the suggested algorithm.

1. **Matching and Probabilities Assignments** For the matching operator, it computes the value of n_i for each cluster C_i according to the given request *Req* as follows:

$$Matching(Req, C_i) = n_i = |\bigcup_{i=0}^{|F_i|} (Req \cap F_i^j)| \quad (5)$$

Where F_i^j is the j^{th} set in the closed frequent patterns F_i of the cluster C_i.

The probability of selecting the cluster C_i is given by the following formula:

$$P_i = \frac{n_i}{n} \quad (6)$$

For instance, let's consider a collection of 10 documents $D = \{d_1, d_2 \ldots d_{10}\}$, five terms $\{t_1, t_2 \ldots t_5\}$, and three clusters $\{C_1, C_2, C_3\}$ represented as follows:

- $C_1 = \{d_1, d_2, d_3\}$ with the closed frequent patterns: $F_1 = \{\{t_1, t_2\}, \{t_1, t_3\}\}$.
- $C_2 = \{d_4, d_5, d_6\}$ with the closed frequent patterns: $F_2 = \{\{t_2\}, \{t_4\}\}$.
- $C_3 = \{d_7, d_8, d_9, d_{10}\}$ with the closed frequent patterns: $F_3 = \{\{t_4, t_5\}, \{t_3, t_5\}\}$.

If we have the following request $Req = \{t_1, t_2, t_3\}$, the matching function between each cluster and Req is determined as follows:

- $n_1 = |(F_1^1 \cap Req) \cup (F_1^2 \cap Req)| = |\{t_1, t_2, t_3\}| = 3$.
- $n_2 = |(F_2^1 \cap Req) \cup (F_2^2 \cap Req)| = |\{t_2\}| = 1$.
- $n_3 = |(F_3^1 \cap Req) \cup (F_3^2 \cap Req)| = |\{t_3\}| = 1$.

Furthermore, the probability of each document in D is calculated as follows:

- $P_1 = 3/5 = 60\%$ (Probability of the cluster C_1 which is the probabilities of documents d_1, d_2 and d_3.)
- $P_2 = 20\%$ (Probability of the cluster C_2 which is the probabilities of documents d_4, d_5 and d_6.)
- $P_3 = 20\%$ (Probability of the cluster C_3 which is the probabilities of documents d_7, d_8, d_9 and d_{10}.)

2. **Encoding Solution** Each solution is a vector of l elements, where each element S_i is an integer and its value ranges from 1 to m. For instance, let's consider the whole collection of documents $\{d_1, d_2 \ldots . d_{100}\}$. Let's $l = 4$ be the size of solutions which is the number of relevant documents to the given user's request, and two solutions s_1 and s_2 given by:
   - $s_1 = \{1, 6, 75, 85\}$ is an admissible solution which represents the documents $\{d_1, d_6, d_{75}$ *and* $d_{85}\}$.
   - $s_2 = \{2, 84, 96, 101\}$ is an inadmissible solution because the document D_{101} does not belong to the collection.
3. **Fitness Computing** The fitness computing of a solution s according to the given request Req is determined by adapting the Jaccard coefficient Hamers et al. (1989) as follow:

$$Fitness_{max}(s) = \sum_{i=1}^{|s|} \frac{|Req \cap d_{s[i]}|}{|Req| \times |d_{s[i]}|} \quad (7)$$

4. **Updating Positions** Each bee updates its solution s using the probability vector P. Two heuristics are proposed.
   - **First heuristic** In this context, one document belonging to s is replaced by another one that does not belong to it. By scanning the bits of s, we select the bit having less probability in P, and replace it by document that does not belong to s and having high probability in P. Let's consider the probabilities vector of 10 documents $P = \{60\%, 60\%, 60\%, 20\%, 20\%, 20\%, 20\%, 20\%, 20\%, 20\%\}$, and the solution $s = \{1, 5, 8\}$. According to the bits of the solution, on the one hand, the documents d_5 and d_8 have less probability (20%) compared to the document d_1. On the other hand, d_2 and d_3 have high probabilities value compared to the remaining documents which do not belong to s. For this reason, we choose randomly one document among d_5 and d_8 to replace it with one document among d_2 and d_3. The new solution obtained by replacing d_8 by d_3 is {1, 5, 3}.
   - **Second heuristic** Regarding the second heuristic, we scan all bits of the solution s. For each selected bit, a random number μ is generated and its values are between [0:1]. If the probability of the current document d_i is less than μ, d_i is replaced by a document that does not belong to s chosen randomly. For instance, consider the previous example, we have:
     - For the first document d_1, and for μ_1 set to 20%, this document is kept on s.
     - For the second document d_5, and for μ_2 set to 25%, this document is replaced by one of the following documents $\{d_2, d_3, d_4, d_6, d_7, d_9, d_{10}\}$, let's consider the document d_3.
     - For the third document d_8, and for μ_3 set to 28%, this document is replaced by one of the following documents $\{d_2, d_4, d_6, d_7, d_9, d_{10}\}$, let's consider the document d_2.

     The current solution should be {1, 2, 3}.

The two heuristics use the probabilities vector of documents computed previously. However, the first heuristic is fast, and converge rapidly to the documents having high probability value. In another term, it converges to the local optima. Counterpart, the second heuristic avoid the local optima by considering all documents in the collection, however, it consumes more time compared to the first one. This claim will be validated later in the performance evaluation section (See Section 4 for more details).

3.3. BSOGDM2: Second BSO algorithm guided by data mining

Contrary to BSOGDM1, BSOGDM2 each bee has a local view to explores the structural knowledge extracted during the preprocessing step (the clusters and the closed frequent patterns). BSOGDM2 is described in algorithm 4

Algorithm 4 Algorithm BSOGDM2.

Input: The set of Clusters of Documents $C = \{C_1, C_2 \ldots C_k\}$
The set of Closed Frequent Patterns $F = \{F_1, F_2 \ldots F_k\}$
The User's Request Req.
Begin
1: **for** each cluster C_i **do**
2: $RT_i \rightarrow$RelevantTerms(F_i, Req)
3: **end for**
4: $BeeInit \leftarrow$ Initial_Solution
5: **while** non stop **do**
6: FindSearchRegion2(BeeInit, p, C, $R_1, R_2, \ldots, R_p$)
7: **for** each bee i **do**
8: $BestSol_i \rightarrow$UpdatingPositions2(R_i, RT_i, C_i)
9: DanceTable$\leftarrow$ $DanceTable \cup BestSol_i$
10: **end for**
11: $BeeInit \leftarrow$ Merging(DanceTable)
12: **end while**
13: **End**

According to this algorithm, the RelevantTerms procedure is first applied to each cluster of documents and the user's request. This procedure returns the set of common terms between the closed frequent patterns of each cluster and the user's request. The result obtained from each cluster C_i is saved on the list of terms called RT_i. As in BSO, the initial bee *BeeInit* creates the initial solution. Afterward, the set of regions is then determined using FindSearchRegion2. Each bee is assigned to explore deeply one cluster of documents. It has a local view of the solution space by taking into account only the documents of the cluster assigned to it. Each bee explores its region by considering the relevant terms of the cluster assigned to it. The best partial solution found by each bee is saved on the dance table. At the end of each pass of the algorithm, the merging procedure is established to find the best global of the current iteration becoming the *BeeInit* for the next iteration. This process should be repeated until the maximum number of iterations is reached. In the following, we provide the main compo-

nents of this algorithm. Note that the encoding of solution and the fitness computing are the same as for the previous strategies:

1. **Relevant Terms** For the RelevantTerms function, it consists to determine the set RT_i for each cluster C_i according to the given request Req as follows:

$$RelevantTerms(Req, C_i) = RT_i = \bigcup_{i=0}^{|F_i|} (Req \cap F_i^j) \quad (8)$$

Where F_i^j is the j^{th} set in the closed frequent patterns F_i of the cluster C_i.
For instance, let's consider the collection of 6 documents $\{D_1, D_2 \dots D_6\}$, four terms $\{t_1, t_2 \dots t_4\}$, and two clusters $\{C_1, C_2\}$ represented as follows:
- $C_1 = \{D_1, D_2, D_3\}$ with the closed frequent terms: $F_1 = \{\{t_1\}, \{t_1, t_2\}\}$.
- $C_2 = \{D_4, D_5, D_6\}$ with the closed frequent terms: $F_2 = \{\{t_2, t_4\}\}$.

If we have the following request $Req = \{t_1, t_2, t_3\}$, the relevant terms of each cluster related to Req are determined as follows:
- $RT_1 = (F_1^1 \cap Req) \cup (F_1^2 \cap Req) = \{t_1\} \cup \{t_1, t_2\} = \{t_1, t_2\}$
- $RT_2 = (F_2^1 \cap Req) = (\{t_2, t_4\} \cap Req) = \{t_2\}$

2. **FindSearchRegion2** Each bee has only one view of the solution space. Indeed, the i^{th} bee takes from *BeeInit* only the documents that belong to the i^{th} cluster and constructs the partial solution from it. As a result, the size of bees may be different according to the size of each cluster and the documents that belong to *BeeInit*. To deal with this issue, in the initialization of *BeeInit*, an equal number of documents should be selected from each cluster. For instance, if we have k clusters and the size of solutions is l, we select $\frac{l}{k}$ documents from each cluster for constructing the initial bee *BeeInit*. Note that the number of bees should be equal to the number of clusters, i.e., p is set to k. To illustrate this strategy, let's consider the set of 100 documents $D = \{d_1, d_2 \dots d_{100}\}$, three cluster $C_1 = \{d_1 \dots d_{30}\}$, $C_2 = \{d_{31} \dots d_{60}\}$ and $C_3 = \{d_{61} \dots d_{100}\}$. For $l = 6$, we have $BeeInit = \{d_1, d_{26}, d_{45}, d_{60}, d_{91}, d_{100}\}$ and the regions of bees are given by:
- $bee_1 = \{d_1, d_{26}\}$
- $bee_2 = \{d_{45}, d_{60}\}$
- $bee_3 = \{d_{91}, d_100\}$

3. **UpdatingPositions2** For each bee i, we scan all bits of the solution s assigned to it. For each selected bit j, the probability of the document d_{s_j} noted $P(d_{s_j})$ is calculated as $\frac{|RT_i \cap d_{s_j}|}{|RT_i|}$. Afterward, a random number μ is generated. Its values are between [0:1]. If $P(d_{s_j})$ is less than μ then, d_{s_j} should be replaced by one document from the cluster C_i that does not belong to s chosen randomly. For instance, in the previous example, we have:
- For the first document d_1 of the first bee, if $P(d_1) = 30\%$ and μ_1 is set to 20%, this document is kept on s.
- Otherwise, for the second document d_{26} of the first bee, if $P(d_1) = 20\%$ and μ_2 is set to 25%, this document is replaced by one of the remaining documents of the first cluster. Let's consider the document d_5.

The current solution of the first bee should be {1, 5}.

4. **Merging** Each bee improves in *BeeInit* only the documents of its region and returns in *Dance* table the best partial solution of the whole collection, i.e., the best partial solution of the cluster assigned to it. The next solution of *BeeInit* is determined as follows:

$$BeeInit \leftarrow BestSol_[bee_1] \odot BestSol_[bee_2] \dots\dots BestSol_[bee_k] \quad (9)$$

Where, $BestSol_[bee_i]$ is the partial solution found by a bee i and $\odot$ is a simple concatenation of the partial solutions. For instance, if we consider 3 bees with the following best partial solutions $bee_1 = \{1, 5\}$, $bee_2 = \{50, 52\}$ and $bee_3 = \{89, 95\}$, the *BeeInit* of the next iteration will be {1, 5, 50, 52, 89, 95}.

4. Performance evaluation

A number of experiments have been carried out to demonstrate the performance of the proposed approach in the DIR problem. We first present the collections of documents used in the experiments, followed by a discussion of how the parameters in the suggested approach have been fixed. Then, we compare the proposed approach with existing bio-inspired approaches for dealing with DIR problem in terms of runtime performance and the quality of returned documents. Regarding the evaluation measure, we have used F-measure. It is based on (Recall and Precision) and it is the well-known measure for the DIR problem:

Recall is the ratio of the number of relevant documents retrieved to the total number of all relevant documents.

$$Recall = \frac{|RDR|}{|ARD|} \quad (10)$$

Where:
RDR: The set of the Relevant Documents Retrieved.
ARD: The set of All Relevant Documents.

Precision is the ratio of the number of relevant documents retrieved to the total number of returned documents.

$$Precision = \frac{|RDR|}{|RD|} \quad (11)$$

Where:
RD: The set of all returned documents.

F-measure This measure allows to combine precision and recall measures, which is defined as follows:

$$F - measure = \frac{2 \times Recall \times Precision}{Recall + Precision} \quad (12)$$

4.1. Collection of documents

The collections considered in our evaluation differ by size, i.e., we consider medium, large and big collections. The first collection used is CACM which is a collection of abstracts of articles published in CACM journal between the years 1958 and 1979. It contains 3204 documents and 6468 terms. It is considered as a medium collection. The second set of data instances that we consider is a large collection retrieved from the renowned TREC repositories.[1] It was first created in 1992 by the U.S. National Institute of Standards TREC and Technology (NIST). Within this framework, there have been many tracks over a range of different topics including Ad Hoc, Medical, Weblogs, and Others. Moreover, two real big collections have been used (Webdocs and Wikilinks).

1. Webdocs[2]: It is built from a collection of HTML document available on the Web. The collection contains about 1.7 millions of documents, mainly written in English, and its size is about 5GB. The transactions are generated from the documents, whereby terms in a document represent items in the database. The documents are first scanned and filtered using NLP (Natural Language Processing)techniques implemented in the NLTK (Natural Language ToolKit) package in Python (Lucchese, Orlando, Perego, & Silvestri, 2004).
2. Wikilinks[3]: It is a collection of documents representing a subset of Wikipedia pages. It contains 40 millions of documents over 3 million entities. As in the case of the Webdocs dataset, we have

[1] http://trec.nist.gov/data.html.
[2] available at http://fimi.ua.ac.be/data/webdocs.
[3] available at: http://www.iesl.cs.umass.edu/data/wiki-links.

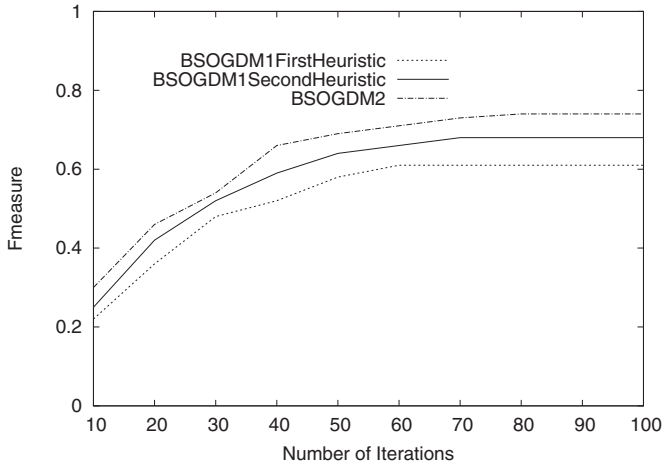

Fig. 2. Quality of returned documents by our approaches using CACM collection with different number of iterations.

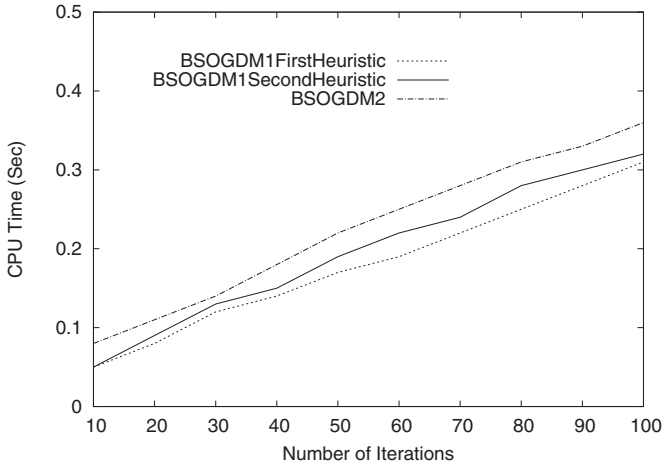

Fig. 3. Runtime (Sec) of our approaches using CACM collection with different number of iterations.

used NLP techniques implemented in the NLTK Python package to retrieve the Wikilinks documents.

Note that all algorithms have been implemented in C++ and experiments have been run on a desktop machine equipped with Intel *I3* processor and 4GB memory.

4.2. Parameter's settings of the proposed framework

Figs. 2 and 3 show the quality of returned documents (F-measure) and the runtime (in Sec) of our approaches (BSOGDM1 with the first heuristic, BSOGDM2 with second heuristic and BSOGDM2) using CACM collection. By varying the number of iterations from 10 to 100, the results show that BSOGDM2 outperforms the two other approaches regarding the quality of the returned documents. However, the runtime of BSOGDM2 is a slightly higher compared to the two other approaches. This can be explained by the fact that in BSOGMD2, each bee explores deeply the cluster of documents assigned to it, which improves the quality and reduces the execution time. According to these results, the BSOGDM2 is chosen for the remaining of experiments. Furthermore, Fig. 2 reveals that the quality starts increasing then stabilizes after 70 iterations. Thus, the number of iterations is set to 70 for the remaining of these experiments.

The aim of the next experiments is to fix the parameters of BSOGDM2 in the preprocessing step. We fix the number of clusters

Table 1
Quality of returned documents and Runtime (Sec) of BSOGDM2 using CACM collection with different number of clusters.

Number of clusters	Fmeasure	Runtime (Sec)
5	0.42	0.16
10	0.54	0.25
20	0.74	0.31
30	0.74	0.51
40	0.74	0.91
50	0.74	1.11

Table 2
Quality of returned documents and Runtime (Sec) of BSOGDM2 using CACM collection with different number of minimum support .

Minimum supports	Fmeasure	Runtime (Sec)
10	0.74	1.59
20	0.74	1.12
30	0.74	0.96
40	0.74	0.84
50	0.74	0.31
60	0.71	0.29
70	0.66	0.24
80	0.61	0.17
90	0.51	0.12
100	0.42	0.09

of K-means algorithm and the minimum support of DCI_Closed algorithm. Tables 1 and 2 show both the quality of returned documents (F-measure) and the runtime (in Sec) of BSOGDM2 using CACM collection with different number of clusters and different number of minimum support. By increasing the number of clusters from 5 to 50, the quality of returned documents improves until it reaches the iteration number 20, where it stabilizes at 0.74 with slight difference in runtime. As a result, the number of clusters is set to 20 for the remaining of experiments.

Again, by increasing the minimum support from 10% to 100%, the quality of returned documents stabilizes at 0.74 until the iteration number 50%. After that, the quality start decreasing from 60% to 100%. Nevertheless, the runtime is reduced while the minimum support is decreased. Thus, the minimum support is set to 50% for the remaining of experiments.

4.3. BSOGDM2 vs. Data mining approaches

This experiment aims to compare our approach with the existing data mining-based approaches using large TREC collections. Table. 3 presents the quality of returned documents by BSOGDM2 and the other data mining-based approaches (PTM (Zhong et al., 2012), SVMIR (Joachims, 2002), KNNIR (Lan et al., 2009) and ARMIR (Babashzadeh et al., 2013)) using the TREC collection. By varying the number of documents from 1 to 35 thousands documents, the quality is reduced whatever the approach used. However, BSOGDM2 outperforms the other approaches regarding the quality of documents. These results are obtained thanks to the efficient preprocessing step used in our approach that combines both clustering and frequent itemsets mining to extract useful knowledge in the solving step.

Fig. 4presents the runtime (Sec) of BSOGDM2 and the other data mining-based approaches (PTM, SVMIR, KNNIR and ARMIR) using 35, 000 documents of TREC collection. By varying the number of user's request from 1 to 50 thousand requests, the runtime is enhanced whatever the approach used. Furthermore, our approach is competitive to SVMIR and KNNIR and outperforms the PTM approach. These results are obtained due to the preprocess-

Table 3
Quality of returned documents by BSOGDM2 and the other data mining-based approaches using TREC collection with different number of documents (in thousands) .

Number of documents (in thousands)	BSOGDM2	PTM	SVMIR	KNNIR	ARMIR
1	0.75	0.74	0.73	0.70	0.65
5	0.71	0.70	0.69	0.67	0.60
10	0.65	0.64	0.61	0.60	0.55
20	0.50	0.48	0.45	0.42	0.35
30	0.42	0.38	0.35	0.35	0.31
35	0.39	0.36	0.35	0.33	0.25

Table 4
Quality of returned documents by BSOGDM2 and the other bio-inspired approaches using WebDocs collection with different number of documents (in thousands).

Number of documents (in thousands)	BSOGDM2	FlyIR	SwarmRank	EQS
100	0.35	0.34	0.30	0.25
200	0.32	0.31	0.29	0.24
500	0.28	0.27	0.23	0.18
800	0.24	0.23	0.20	0.15
1200	0.22	0.21	0.18	0.14
1500	0.21	0.19	0.16	0.14
1700	0.20	0.18	0.15	0.12

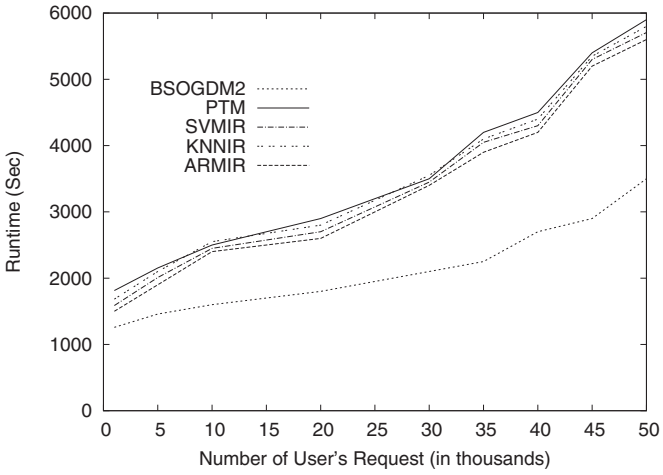

Fig. 4. Runtime (Sec) of BSOGDM2 and the other data mining-based approaches using TREC collection (containing 35000 Documents) with different number of user's request (in thousands).

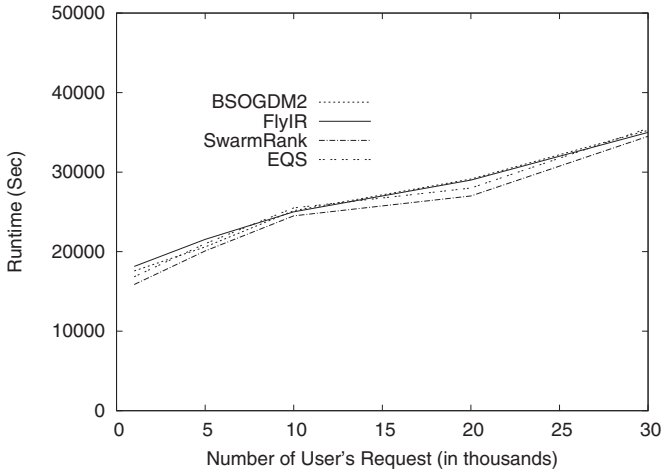

Fig. 5. Runtime (Sec) of BSOGDM2 and the other bio-inspired approaches using WebDocs collection (containing 170000 Documents) with different number of user's request (in thousands).

Table 5
Quality of returned documents by BSOGDM2 and the other bio-inspired approaches using Wikilinks collection with different number of documents (in million) .

Number of documents (in million)	BSOGDM2	FlyIR	SwarmRank	EQS
10	0.35	0.34	0.30	0.25
20	0.32	0.31	0.29	0.24
30	0.28	0.27	0.23	0.18
40	0.24	0.23	0.20	0.15

ing step of BSOGDM2 which is performed only one time for any number of user's request.

4.4. BSOGDM2 vs. Bio-inspired approaches

The next experiments aim to compare our approach with the recent bio-inspired approaches using two big collections of documents (Webdocs and Wikilinks) described above. Tables 4 and 5 present the quality of documents of our approach (BSOGDM2) and the other bio-inspired approaches (FLYIR (Khennak & Drias, 2016), SwarmRank (Diaz-Aviles et al., 2009) and EQS Jung (2012)) using Webdocs and Wikilinks collections respectively. By varying the number of documents of Webdocs (in thousand) and the number of documents of Wikilinks (in million), BSOGDM2 outperforms the other bio-inspired approaches in terms of the quality of returned documents. These results are obtained thanks to our methodology which uses the knowledge extracted on the preprocessing step to guide the swarms in exploring the whole space of collections intelligently.

Figs. 5 and 6 present the runtime (in Sec) of BSOGDM2 and (FLYIR, SwarmRank and EQS) using respectively Webdocs and Wikilinks collections. By varying the number of user's request (from 1 to 30 thousands of Webdocs) and (from 1 to 10 thousands of Wikilinks), the results reveal that our approach outperforms FLYIR approach, however, it needs few more computational time compared to SwarmRank and EQS. These results have been obtained due to the fact that each bee explores the cluster assigned to it deeply. To improve the results the process requires more computational time.

4.5. BSOGDM2 vs. State-of-the-art DIR approaches

The last experiments aim to compare our approach with recent state-of-the-art approaches using the Wikilinks instance. Table 6

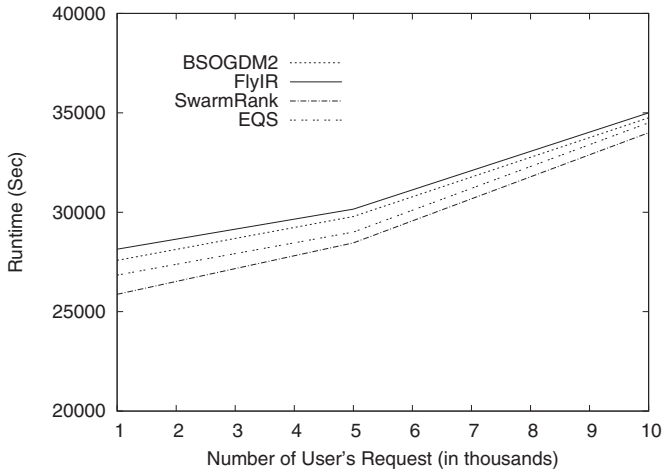

Fig. 6. Runtime (Sec) of BSOGDM2 and the other bio-inspired approaches using Wikilinks collection (containing 40 million Documents) with different number of user's request (in thousands).

Table 6
Quality of returned documents by BSOGDM2 and the state-of-the-art DIR approaches using Wikilinks collection with different number of documents (in million) .

Number of documents (in million)	BSOGDM2	NSPR	PLSA
10	0.35	0.32	0.31
20	0.32	0.28	0.26
30	0.28	0.20	0.18
40	0.24	0.21	0.21

presents the quality of documents of our approach (BSOGDM2) and two recent DIR approaches NSPR (Neural Semantic Personalized Analysis) Ebesu and Fang (2017) and PLSA (Probability Latent Semantic Analysis) (Zhai, 2017). By varying the number of documents of Wikilinks (in million), BSOGDM2 outperforms the two DIR approaches in terms of the quality of returned documents. These results confirms again the usefulness of the data mining techniques to guide the swarms in exploring the whole space of documents.

4.6. Discussion

For the sake of conciseness, in the remainder of this section we discuss the main findings from the application of our approach to real collections of documents .

- The first finding of our study is that using data mining techniques is not sufficient to provide the exact solution to the user. The solution found by our approach is still approximative compared to the exact-based approaches. However, our approach can deal with big data collections, which the exact ones are bluntly blocked for such dimensions.
- Being based on a bio-inspired techniques, our approach has a typically inductive and predictive character. In the context of information retrieval, we argue that current tools enable finding bad solutions in fast time. Our approach, on the contrary, gives approximate solutions in a reasonable computational time. In this context, we argue that our approach benefits form the knowledge extracted from the pre-processing step that shifts the intelligence required for identifying relevant patterns from the whole collection of documents.
- From a data mining research standpoint, our paper is an example of the application of a generic data mining technique to a specific context. The literature calls for this type of research, particularly in the times of Big Data, where increasingly large amounts of data are available in different domains. As in many other cases, porting a pure data mining technique into a specific application domain requires methodological refinement and adaptation. In our specific context, this adaptation is implemented in different phases, such as clustering the collection of documents and extracting closed frequent terms.

To the best of our knowledge the approach proposed in this paper is the first one that has used data mining techniques in exploring the solutions space for dealing with big collections of documents.

5. Conclusion

In this paper, we have proposed a new swarm intelligence approach for solving the DIR problem. The swarms explore the space of documents using knowledge discovery by two data mining techniques. The first one decomposes the whole collection of documents into similar and disjoint clusters using the K-means algorithm. Its main feature is the use of a distance measure between documents, and a technique to determine the centroid for the set of documents. The second one extracts the closed frequent terms from each cluster of documents using the DCI_Closed algorithm. The specificity of this approach is to use minimum supports which controls the set of frequent terms extracted. These two techniques (K-means and DCI_Closed) allow dividing a collection of documents into several sub-collections, each of which is featured by the set of closed frequent terms between the documents belonging to it. The swarms then use this knowledge to explore each cluster deeply for any user's request.

To demonstrate the performance of the suggested approach, several experiments have been carried out using well-known collections of documents. The results revealed that the swarms benefit from the relevant knowledge extracted in the preprocessing step and improve the quality of returned documents. The proposed approach has also been compared to state-of-the-art data mining, bio-inspired, and other document information retrieval based approaches using large TREC and big (Webdocs and Wikilinks) instances. The results indicate that our approach outperforms the other algorithms in terms of document's quality and has a very competitive run time. As perspectives, we plan to investigate the following issues:

1. Integrating other swarm and evolutionary based approaches to our framework.
2. Extending the suggested approach to other optimization problems such as the coloring problem, constraint satisfaction problem and graph problems.
3. Discovering other kind of knowledge such as maximal itemsets, rare itemsets, and high utility itemsets, to improve the accuracy of our approach.
4. Propose a parallel version that explores high performance computing to solve very big collections is also in our future agenda.

References

Akbari, R., Mohammadi, A., & Ziarati, K. (2010). A novel bee swarm optimization algorithm for numerical function optimization. *Communications in Nonlinear Science and Numerical Simulation, 15*(10), 3142–3155.

Babashzadeh, A., Daoud, M., & Huang, J. (2013). Using semantic-based association rule mining for improving clinical text retrieval. In *International conference on health information science* (pp. 186–197). Springer.

Beil, F., Ester, M., & Xu, X. (2002). Frequent term-based text clustering. In *Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 436–442). ACM.

Belkebir, R., & Guessoum, A. (2013). A hybrid bso-chi2-svm approach to arabic text categorization. In *Computer systems and applications (AICCSA), 2013 acs international conference on* (pp. 1–7). IEEE.

Benetos, E., Dixon, S., Giannoulis, D., Kirchhoff, H., & Klapuri, A. (2013). Automatic music transcription: Challenges and future directions. *Journal of Intelligent Information Systems, 41*(3), 407–434.

Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM, 55*(4), 77–84.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research, 3*(Jan), 993–1022.

Buscher, G., Dengel, A., Biedert, R., & Elst, L. V. (2012). Attentive documents: Eye tracking as implicit feedback for information retrieval and beyond. *ACM Transactions on Interactive Intelligent Systems (TiiS), 1*(2), 9.

Cai, D., He, X., & Han, J. (2005). Document clustering using locality preserving indexing. *IEEE Transactions on Knowledge and Data Engineering, 17*(12), 1624–1637.

Collett, T. S., Graham, P., Harris, R. A., & Hempel-de Ibarra, N. (2006). Navigational memories in ants and bees: Memory retrieval when selecting and following routes. *Advances in the Study of Behavior, 36*, 123–172.

Croft, W. B., & Harper, D. J. (1979). Using probabilistic models of document retrieval without relevance information. *Journal of Documentation, 35*(4), 285–295.

Diaz-Aviles, E., Nejdl, W., & Schmidt-Thieme, L. (2009). Swarming to rank for information retrieval. In *Proceedings of the 11th annual conference on genetic and evolutionary computation* (pp. 9–16). ACM.

Djenouri, Y., Bendjoudi, A., Mehdi, M., Nouali-Taboudjemat, N., & Habbas, Z. (2015). Gpu-based bees swarm optimization for association rules mining. *The Journal of Supercomputing, 71*(4), 1318–1344.

Djenouri, Y., Drias, H., & Chemchem, A. (2013). A hybrid bees swarm optimization and tabu search algorithm for association rule mining. In *Nature and biologically inspired computing (NaBIC), 2013 world congress on* (pp. 120–125). IEEE.

Djenouri, Y., Habbas, Z., & Aggoune-Mtalaa, W. (2016). Bees swarm optimization metaheuristic guided by decomposition for solving max-sat.. In *ICAART (2)* (pp. 472–479).

Ebesu, T., & Fang, Y. (2017). Neural semantic personalized ranking for item cold-start recommendation. *Information Retrieval Journal, 20*(2), 109–131.

Fan, W., Gordon, M. D., & Pathak, P. (2004). Discovery of context-specific ranking functions for effective information retrieval using genetic programming. *IEEE Transactions on Knowledge and Data Engineering, 16*(4), 523–527.

Fung, B. C., Wang, K., & Ester, M. (2003). Hierarchical document clustering using frequent itemsets. In *Proceedings of the 2003 SIAM international conference on data mining* (pp. 59–70). SIAM.

Hamers, L., Hemeryck, Y., Herweyers, G., Janssen, M., Keters, H., Rousseau, R., et al. (1989). Similarity measures in scientometric research: The jaccard index versus salton's cosine formula. *Information Processing & Management, 25*(3), 315–318.

Hammouda, K. M., & Kamel, M. S. (2004). Efficient phrase-based document indexing for web document clustering. *IEEE Transactions on Knowledge and Data Engineering, 16*(10), 1279–1296.

Hatamlou, A., Abdullah, S., & Nezamabadi-Pour, H. (2012). A combined approach for clustering based on k-means and gravitational search algorithms. *Swarm and Evolutionary Computation, 6*, 47–52.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 133–142). ACM.

Jung, J. J. (2012). Evolutionary approach for semantic-based query sampling in large-scale information sources. *Information Sciences, 182*(1), 30–39.

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. *Technical Report Technical report-tr06*. Erciyes university, engineering faculty, computer engineering department.

Khennak, I., & Drias, H. (2016). A firefly algorithm-based approach for pseudo-relevance feedback: Application to medical database. *Journal of Medical Systems, 40*(11), 240.

Lafferty, J., & Zhai, C. (2017). Document language models, query models, and risk minimization for information retrieval. In *ACM SIGIR forum: 51* (pp. 251–259). ACM.

Lan, M., Tan, C. L., Su, J., & Lu, Y. (2009). Supervised and traditional term weighting methods for automatic text categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 31*(4), 721–735.

Lin, C.-H., Chen, H.-Y., & Wu, Y.-S. (2014). Study of image retrieval and classification based on adaptive features using genetic algorithm feature selection. *Expert Systems with Applications, 41*(15), 6611–6621.

Lucchese, C., Orlando, S., & Perego, R. (2006). Fast and memory efficient mining of frequent closed itemsets. *IEEE Transactions on Knowledge and Data Engineering, 18*(1), 21–36.

Lucchese, C., Orlando, S., Perego, R., & Silvestri, F. (2004). Webdocs: A real-life huge transactional dataset.. *FIMI*: 126.

MacQueen, J. (1965). On convergence of k-means and partitions with minimum average variance. In *Annals of mathematical statistics: 36* (p. 1084). Inst Mathematical Statistics IMS Business Office-Suite 7, 3401 Investment Blvd, Hayward, CA 94545.

Mahdavi, M., & Abolhassani, H. (2009). Harmony k-means algorithm for document clustering. *Data Mining and Knowledge Discovery, 18*(3), 370–391.

Menezes, G., Almeida, J., Belém, F., Gonçalves, M., Lacerda, A., de Moura, E., et al. (2010). Demand-driven tag recommendation. *Machine Learning and Knowledge Discovery in Databases*, 402–417.

Niknam, T., & Golestaneh, F. (2013). Enhanced bee swarm optimization algorithm for dynamic economic dispatch. *IEEE Systems Journal, 7*(4), 754–762.

Pei, J., Han, J., Mao, R., et al. (2000). Closet: An efficient algorithm for mining frequent closed itemsets.. In *ACM SIGMOD workshop on research issues in data mining and knowledge discovery: 4* (pp. 21–30).

Picard, D., Revel, A., & Cord, M. (2012). An application of swarm intelligence to distributed image retrieval. *Information Sciences, 192*, 71–81.

Ponte, J. M., & Croft, W. B. (1998). A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on research and development in information retrieval* (pp. 275–281). ACM.

Salton, G., & McGill, M. J. (1986). Introduction to modern information retrieval,.

Seeley, T. D., Camazine, S., & Sneyd, J. (1991). Collective decision-making in honey bees: How colonies choose among nectar sources. *Behavioral Ecology and Sociobiology, 28*(4), 277–290.

Steinbach, M., Karypis, G., Kumar, V., et al. (2000). A comparison of document clustering techniques. In *KDD workshop on text mining: 400* (pp. 525–526). Boston.

Tata, S., & Patel, J. M. (2007). Estimating the selectivity of tf-idf based cosine similarity predicates. *ACM Sigmod Record, 36*(2), 7–12.

Veloso, A. A., Almeida, H. M., Gonçalves, M. A., & Meira Jr, W. (2008). Learning to rank at query-time using association rules. In *Proceedings of the 31st annual international ACM SIGIR conference on research and development in information retrieval* (pp. 267–274). ACM.

Wang, J., Han, J., & Pei, J. (2003). Closet+: Searching for the best strategies for mining frequent closed itemsets. In *Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 236–245). ACM.

Wei, X., & Croft, W. B. (2006). Lda-based document models for ad-hoc retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 178–185). ACM.

Yu, H., Searsmith, D., Li, X., & Han, J. (2004). Scalable construction of topic directory with nonparametric closed termset mining. In *Data mining, 2004. ICDM'04. fourth IEEE international conference on* (pp. 563–566). IEEE.

Zaki, M. J., & Hsiao, C.-J. (2002). Charm: An efficient algorithm for closed itemset mining. In *Proceedings of the 2002 SIAM international conference on data mining* (pp. 457–473). SIAM.

Zhai, C. (2017). Probabilistic topic models for text data retrieval and analysis. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval* (pp. 1399–1401). ACM.

Zhang, C., Ouyang, D., & Ning, J. (2010). An artificial bee colony approach for clustering. *Expert Systems with Applications, 37*(7), 4761–4767.

Zhang, L., Mei, T., Liu, Y., Tao, D., & Zhou, H.-Q. (2011). Visual search reranking via adaptive particle swarm optimization. *Pattern Recognition, 44*(8), 1811–1820.

Zhong, N., Li, Y., & Wu, S.-T. (2012). Effective pattern discovery for text mining. *IEEE transactions on knowledge and data engineering, 24*(1), 30–44.