CLUSTERING FOOD RECIPES USING A FLOWER POLLINATION-BASED METHOD

Viorica R. CHIFU, Cristina Bianca POP, Ion SALOMIE, Andreea ACZEL
Department of Computer Science, Technical University of Cluj-Napoca
Cluj-Napoca, Romania
{viorica.chifu, cristina.pop, ioan.salomie}@cs.utcluj.ro

Abstract: **This paper models the clustering of documents containing food recipes as an optimization problem which is solved using a Flower Pollination-based method. The first step of this method consists in preprocessing the documents describing food recipes using natural language techniques, while the second step consists of representing the preprocessed documents using vector space models. The vector space model representations of the documents containing food recipes are further submitted to the Flower Pollination-based clustering algorithm which clusters the documents based on their content similarity. The content similarity between two documents is evaluated using the cosine similarity measure. The proposed method has been evaluated on a set of food recipes documents available in the RecipeML format.**

Keywords: *food recipes clustering, Flower Pollination Algorithm, tf-idf, cosine similarity.*

I. INTRODUCTION

Nowadays, people are becoming more and more interested in cooking. As the market offers us a large variety of ingredients, the temptation of trying several types of recipes is high. However, sometimes due to the lack of inspiration, people search for recipes online on the Web finding a large number of relevant and irrelevant recipes making the selection of the most appropriate one difficult. In this context, an organization of the available recipes in clusters, based on their similarity would be useful and appropriate because it would reduce the search space thus providing only the recipes that are relevant to the user. The problem of clustering food recipes can be seen as an optimization problem as the aim is to find the optimal or near-optimal configuration of recipes clusters by searching in a large search space of food recipes. Such optimization problems can be solved using bio-inspired meta-heuristics which are capable of providing an optimal or a near-optimal solution in a relatively short time by using search strategies inspired from the intelligence of insects, animals and even plants, intelligence which has enabled them to survive in nature.

In this paper we propose a new method for clustering documents describing food recipes based on their content. The proposed method is based on the Flower Pollination Algorithm and consists of three main phases: *the preprocessing phase*, *the feature selection and document representation model phase*, and *the Flower Pollination-based clustering phase*. The *preprocessing phase* consists of applying natural language techniques such as stop-words elimination, part of speech tagging and stemming on the considered documents in order to eliminate irrelevant words, to associate a part of speech to each word, and to reduce the words to their base form. The *feature selection and document representation phase* refers to extracting the nouns and the noun phrases from the preprocessed documents and then representing each document as a vector space model. By using vector space models, each document is represented as a vector of *m* elements where *m* represents the number of the distinct terms (i.e. nouns or noun phrases) from the documents collection, and the elements of the vector reflect the relevance of each specific term for the considered document, the relevance being computed using the *tf-idf* measure. The *Flower Pollination-based clustering phase* consists of adapting the state of the art Flower Pollination Algorithm [1] in the context of clustering documents containing food recipes based on their similarity. In our approach, the cosine similarity measure has been used to evaluate the similarity between documents. The proposed method has been evaluated on a set of documents containing recipes described in the RecipeML format, an XML standard for sharing recipes provided by [13].

The paper is structured as follows. Section II reviews bio-inspired state of the art approaches in the field of document clustering. Section III addresses the preprocessing and the feature selection and document representation phases, while Section IV details the Flower Pollination-based clustering phase. Experimental results are presented in Section V. The paper ends with conclusions.

II. RELATED WORK

In the past years, nature-inspired meta-heuristics have been considered for solving the data clustering problem, as alternatives or complements to the classical clustering algorithms such as k-means.

For example, in [5], authors combine the Cuckoo Search [7] and Bat [8] nature-inspired algorithms with k-means to overcome the k-means algorithm's problem of getting stuck in a local optimum solution due to the random generation of the initial partitioning. In both approaches, an agent is mapped on a clustering solution, i.e. a configuration of cluster centroids, which is evolved using the operations

Manuscript received November 5, 2015; revised December 10, 2015

specific to each nature-inspired algorithm (e.g. Levy Flight in the case of the Cuckoo Search algorithm, and position/velocity update in the case of the Bat algorithm). The quality of a solution is evaluated in both cases using a fitness function that computes the intra cluster and the inter cluster similarities. Experimental results have shown that the use of Cuckoo Search and Bat algorithms brings a performance improvement.

In [6], authors use the Firefly algorithm [9] to solve the data clustering problem, where the position of a firefly agent (i.e. solution) is mapped on a configuration of cluster centroids. A solution is evaluated using a fitness function that computes the sum of the distances between the data items and their associated centroids, the aim being to minimize this function. This clustering approach uses the same operations for updating the firefly positions as the ones defined in the state of the art Firefly algorithm.

Nature-inspired meta-heuristics have been also applied for text clustering.

For example, in [2] authors model the problem of text clustering as an optimization problem solved with genetic algorithms. A chromosome is modeled as a set of centroids structures, where each structure contains a list of documents. To evolve the population of chromosomes authors use selection, classical single-point crossover and the Gaussian mutation. The algorithm's iterative stage lasts until the algorithm has stagnated on the same optimal solution for a number of iterations. To encode a document, authors use latent semantic indexing. A chromosome is evaluated using a fitness function that is based on the Davies Bouldin index.

In [3], authors propose two methods, one discrete and one continuous, for clustering short texts using Particle Swarm Optimization. These methods rely on representing a document using Vector Space Models. In the first method, a particle (i.e. solution of the optimization problem) is mapped on a structure containing the identifier of the cluster to which each document belongs to, while in the second method, a particle is mapped on a structure containing centroids, each centroid having a set of terms associated. The quality of a particle's position is evaluated using Expected Density Measure and the Global Silhouette coefficient.

In [4], authors propose a method for clustering short text documents using the state of the art AntTree-based clustering algorithm [10] to which they introduce the Silhouette Coefficient and the attraction of a cluster. The AntTree-based clustering algorithm is inspired by the behavior ants exhibit while building their nests. In this algorithm, the ants, which are mapped on the data submitted to clustering, build a tree structure by taking into account the similarity of the associated data. Authors of [4] introduce the Silhouette Coefficient in the initial step of the AntTree-based clustering algorithm to obtain an initial ordering of the ants by processing the clusters resulted using a clustering algorithm. The attraction of a cluster is introduced to help a moving ant decide to which ant already stored in the tree structure to link by taking into account not only the similarity to the ant selected from the tree but also the similarity to the other ants connected to this one. This attraction can be computed using Internal Clusters Validity Measures.

III. REPRESENTING FOOD RECIPES USING A VECTOR SPACE MODEL APPROACH

In our approach we use a set of food recipes provided by [13]. Such a food recipe is stored in an XML format and contains the following information: (*i*) the name of the food recipe, (*ii*) optionally, the categories to which the food recipe belongs (e.g. snacks, meat), (*iii*) the ingredients, where for each ingredient the quantity and the unit of measure is specified, (*iv*) the recipe text, and (*v*) the nutritional values of the food recipe. The XML text in Figure 1 illustrates a fragment of such a food recipe referring to the "1-2-3-4 Cake with Caramel Icing" [13].

```
<recipeml version="0.5">
 <recipe>
  <head>
   <title>1-2-3-4 Cake with Caramel Icing</title>
   <categories>
    <cat>None</cat></categories>
   <yield>12</yield>
  </head>
  <ingredients>
   <ing-div>
    <title/>
    <ing>
     <amt>
      <qty>3</qty>
      <unit>cups</unit></amt>
     <item>All-purpose flour</item>
    </ing>
    <ing>
     <amt>
      <qty>3</qty>
      <unit>teaspoons</unit></amt>
     <item>Baking powder</item>
    </ing>
    …
  </ingredients>
  <directions>
   <step> 1. Preheat oven to 350-F Grease three 9-inch-round cake pans.
Line bottoms with waxed paper. Grease paper, and flour insides of pans.
Combine flour and baking powder in bowl. 2. Beat butter and sugar in
another bowl at medium speed until smooth. Add eggs, beating until
smooth. At low speed, alternately beat in flour mixture and milk. Beat in
vanilla. Scrape into prepared pans, dividing evenly. 3. Bake in 350-F oven
20 to 25 minutes or until wooden pick comes out clean. Cool in pans on
wire racks 10 minutes. Remove cakes from pans and cool completely on
racks. …
Nutrient Value Per Serving: 449 calories, 5 g protein, 19 g fat, 65 g
carbohydrate, 123 mg sodium, 102 mg cholesterol. Exchanges: 1
starch/bread, 1/4 meat, 1/10 milk, 3 fruit, 4 3/4 fat.
   </step>
  </directions>
 </recipe>
</recipeml>
```

Figure 1. 1-2-3-4 Cake with Caramel Icing Recipe [13]

Before representing the food recipes using vector space models a pre-processing step is performed using stop-words elimination, part of speech tagging and stemming which are specific to natural language processing. Stop-words elimination implies eliminating stop-words such as prepositions, articles and pronouns which are not relevant in the context of a recipe. Part of speech tagging refers to tagging each word with its associated part of speech. Stemming refers to reducing the words to its base form (i.e. stem). In order to perform stop-words elimination and stemming the WEKA framework [12] has been used, while for part of speech tagging the Stanford Log-linear Part-Of-Speech Tagger has been used [11]. The document resulted after performing stop-words elimination, part of speech tagging and stemming is further represented using Vector Space Models.

Using the Vector Space Model approach, a document is represented as a matrix with *n* rows and *m* columns, where *n* is the total number of distinct words (i.e. nouns or noun phrases) part of the set of recipes, and *m* is the total number of documents containing food recipes. An element in this matrix corresponding to the i^{th} and j^{th} column contains a number reflecting the relevance of the i^{th} word in the j^{th} document. The relevance is computed using the *tf-idf* measure as follows:

$$tfidf(word, doc) = tf(word, doc) * idf(word, Docs) \quad (1)$$

where *tf* evaluates the frequency of the word *word* in the document *doc*, and *idf* evaluates the inverse document frequency referring to how relevant is *word* in the set of documents *Docs*.

IV.THE FLOWER POLLINATION-BASED CLUSTERING

A. The Flower Pollination Algorithm

The Flower Pollination Algorithm [1] proposed for solving global optimization problems has been inspired by the flower pollination process taking place in nature. Flower pollination is the process by which the pollen is transmitted between flowers. Pollination can be classified according to the type of the entity performing pollination in biotic and abiotic, and according to the type of pollen in cross-pollination and self-pollination. Biotic pollination is performed by pollinators such as insects or birds, while abiotic pollination is performed as a result of wind blowing. In the case of self – pollination, the pollination occurs from the pollen of the same flower or from the pollen of a different flower part of the same species, while in the case of cross pollination, the pollination occurs from the pollen of a flower of a another species.

In the context of the Flower Pollination Algorithm, the flower and the pollen are mapped on a solution of the optimization problem, abiotic pollination and self-pollination are mapped on a global pollination strategy, while biotic pollination and cross-pollination are mapped on a local pollination strategy.

The global pollination strategy implies modifying a given solution according to the global best solution identified so far using the following formula [1]:

$$sol_i^{'} = sol_i + L * (sol_{opt} - sol_i) \quad (2)$$

where: (*i*) sol_i is the current solution submitted to global pollination; (*ii*) $sol_i^{'}$ is the updated solution, (*iii*) sol_{opt} is the current global best solution, and (*iv*) *L* is the pollination strength.

The local pollination strategy implies modifying a given solution based on two other solutions selected from its neighborhood according to the following formula [1]:

$$sol_i^{'} = sol_i + \varepsilon * (sol_j - sol_k) \quad (3)$$

where: (*i*) sol_i is the current solution submitted to local pollination; (*ii*) $sol_i^{'}$ is the updated solution, (*iii*) sol_j and sol_k are two solutions selected from the neighborhood of sol_i, and (*iv*) ε is a number drawn from a uniform distribution in [0, 1].

B. Mapping the Flower Pollination Algorithm Concepts to Food Recipes Clustering Concepts

In our approach we have mapped the concepts from the state of the art Flower Pollination Algorithm to the concepts of the problem of clustering food recipes as illustrated in Table 1.

Table 1. Concepts mapping

Flower Pollination Algorithm Concepts	**Food Recipes Clustering Concepts**
Solution	Set of clusters of food recipes documents (the documents are represented as Vector Space Models)
Fitness	Value reflecting the similarity between the food recipes documents from each cluster
Global pollination strategy	Crossover-based strategy between a solution and the global optimal solution
Local pollination strategy	Crossover-based strategy between a solution and two of its neighborhood solutions

In our approach, a solution is formally represented as follows:

$$sol = \{cls_1, cls_2, ..., cls_{noClusters}\} \quad (4)$$

where *noClusters* is the number of considered clusters, and cls_i is the cluster *i* formally represented as:

$$cls_i = \{(c_i, \{r_{i1,...,}r_{ij}\})\} \quad (5)$$

where c_i is the cluster centroid (i.e. a recipe that initially is randomly selected), $\{r_{i1,...,}r_{ij}\}$ is the set of recipes part of the cluster cls_i, and *j* is the number of recipes part of the cluster cls_i.

The quality of a solution (i.e. a set of clusters configuration) is evaluated using the following fitness function:

$$Fitness(sol) = \frac{\sum_{i=1}^{noClusters} OverallSimilarity(c_i)}{noClusters} \quad (6)$$

where *OverallSimilarity* is a function that evaluates the similarity between the documents of a cluster c_i as follows:

$$OverallSimilarity(c_i) = \frac{\sum_{j=1}^{noClusterDocs-1}(1 - cosSim(cent_i, doc_{ij}))}{noClusterDocs - 1} \quad (7)$$

where $noClusterDocs_i$ represents the number of documents part of cluster *i*, doc_{ij} represents the j^{th} document from cluster *i*, and *cosSim* is a function that evaluates the cosine similarity between the cluster's centroid and a document part of the same cluster.

In our approach, the global and local pollination strategies are implemented as crossover-based strategies.

The global pollination strategy modifies the current solution sol_i according to the current global solution sol_g as follows: (1) a vector *D* containing all the documents to be clustered is considered; (2) a vector *V* of values randomly selected from the (0, 1] interval is generated, where the number of the vector's elements is equal with the number of documents to be clustered; (3) for each document *D*[*i*] from the vector *D* and its corresponding element *V*[*i*] from the vector *V*:

- If the value of *V*[*i*] is greater than a global pollination threshold $\tau \in (0,1]$, then the document *D*[*i*] will be put in the new solution sol_i' in the same cluster as in the solution sol_g.
- Else, the document *D*[*i*] will be put in the new solution sol_i' in the same cluster as in the initial solution sol_i.

The local pollination strategy implies modifying the current solution sol_i according to other two solutions sol_j and sol_k selected from its neighborhood. We consider that a solution sol_j is in the neighborhood of a solution sol_i if their structures are similar, i.e. they have similar clusters of recipes. The steps of the local pollination strategy are the following: (1) a vector *D* containing all the documents to be clustered is generated; (2) a vector *V* of values randomly selected from the (0, 1] interval is generated, where the number of the vector's elements is equal with the number of documents to be clustered; (3) for each document *D*[*i*] from the vector *D* and its corresponding element *V*[*i*] from the vector *V*:

- If the value of *V*[*i*] is greater than a local pollination threshold $\lambda \in (0,1]$, and if sol_j is more similar to sol_i than sol_k, then the document *D*[*i*] will be put in the new solution sol_i' in the same cluster as in the sol_j solution. Otherwise, the document *D*[*i*] will be put in the new solution sol_i' in the same cluster as in the sol_k solution.

C.The Flower Pollination-based Clustering Algorithm

The proposed Flower Pollination-based clustering algorithm (see Algorithm 1) clusters food recipes documents based on their similarity. The algorithm takes as input the following parameters: (*i*) *noCl* – the number of clusters, (*ii*) *popSize* – population size, (*iii*) *noIt* – the number of algorithm iterations, (*iv*) *p* – switch probability value, (*v*) *pc* – percent of the worst solutions in the population that will be replaced with randomly generated solutions, (*vi*) τ – global pollination threshold, (*vii*) λ - local pollination threshold, and (*viii*) *Recipes* – the repository of recipe documents. The algorithm returns the optimal or a near-optimal configuration of clusters of recipe documents.

In the algorithm's initialization, the initial population of solutions is randomly generated and the global best solution is identified. To randomly generate the solutions part of the initial population, the following steps are performed: (1) a number of *noCl* recipe documents are randomly selected and set as the centroids of *noCl* clusters in all solutions (i.e. each solution part of the initial population will have the same cluster centroids), (2) for each generated centroid, and for each solution, the list of recipe documents to be associated to the centroid is generated by randomly selecting a number of recipe documents from the repository. The strategy used for assigning a list of recipe documents to a centroid ensures that there will be no overlapping clusters within the same solution.

In the algorithm's iterative stage, which lasts until the maximum number of iterations *noIt* is reached, the following steps are performed:

ALGORITHM: Flower Pollination-based Clustering

```
Input: noCl, popSize, noIt, p, pc, $\tau$, $\lambda$, Recipes
Output: $sol_{opt}$
Begin
  Population = Randomly_Generate_Population(noCl, popSize,

  Recipes)
  $sol_{opt}$ = Get_Highest_Fitness_Solution(Population)
  it = 0
  while(it < noIt) do
     foreach sol in Population do
       if (RandomNumber() < p) then
          sol' = Global_Pollination(sol, $sol_{opt}$, $\tau$)
       else
            $sol_j$ = Get_Solution_from_Neighborhood(sol)
            $sol_k$ =
                 Get_Solution_from_Neighborhood(sol)
          sol' = Local_Pollination(sol, $sol_j$, $sol_k$, $\lambda$)
          sol' = Global_Pollination(sol', sol, $\tau$)
       end if
       Population = Population ∪ sol'
     end foreach
     Population = Update_Centroids(Population)
     Population =
            Elliminate_Worst_Solutions(Population)
     $sol_{opt}$ = Get_Highest_Fitness_Solution(Population)
     if (Stagnation($sol_{opt}$)) then
       Population =  Random_Replace(Population, pc)
       noIt = Increase(noIt, pc)
     else it++
     end if
   end while
   return $sol_{opt}$
End
```

---

1. For each solution sol_i part of the population of solutions:
   1.1 If a randomly generated number is less than the given switch probability value, then the global pollination strategy described in sub-section IV.B is applied between the current solution sol_i and the global best solution sol_g.
   1.2 Else, the local pollination strategy described in sub-section IV.B is applied between the current solution sol_i and two solutions sol_j and sol_k selected from the neighborhood of sol_i, followed by applying a global pollination-based strategy between the new generated solution and the initial one *sol*.
   1.3 The newly generated solution sol_i*'* is added to the population of solutions, *Population*.
2. The centroids of all the solutions part of the population of solutions are updated according to the following strategy:
   2.1 First, for each cluster index *i*, we identify the solution which has the best centroid by evaluating the overall similarity of the centroid using Formula 7. After the best centroid has been identified it will replace all the adjacent centroids from the other solutions.
   2.2 Second, for each solution sol_i and for each cluster from the solution sol_i, the cluster's fitness, fit_c, is evaluated using Formula 7. The obtained value fit_c is used to set the radius of a circle C_1 whose center is the cluster's centroid. Since the fitness is the average of the distances from the cluster center to each document, then most of the documents will be situated in the white and orange areas. Then, two new circles C_2 and C_3 are considered, both having the center in the cluster's centroid, with a radius $fit_c \pm \Delta$ (see Figure 2). Let us consider the documents that are part of C_3 and are not in C_2; from these documents we randomly select a document which will be further set as the new centroid of the current cluster if this leads to an increase in the value of the cluster's fitness function. If the selected centroid does not improve the value of the cluster's fitness function, the search is repeated for a predefined number of times. If no appropriate centroid is found then the old centroid remains the centroid of the cluster.

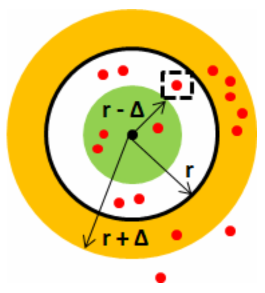

Figure 2. Centroid update

3. As a result of introducing new solutions generated through local and global pollination, the number of solutions from the population has increased significantly. To reduce the population, the worst solutions are eliminated.
4. The global best solution is updated if it is the case.
5. If the global best solution has not been modified for a number of iterations, then a randomization strategy is applied to introduce diversity. The randomization strategy implies replacing a percent *pc* of the worst solutions from the current population with other solutions randomly generated. If the randomization strategy is applied, then the number of iterations is increased with the percent *pc* to avoid situations in which the randomization strategy is triggered when there are few iterations left to run and the impact of randomization cannot be materialized.

V. EXPERIMENTAL RESULTS

This section presents the experimental results obtained by running the Flower Pollination-based clustering algorithm on the recipe documents available in the Squirrel's RecipeML repository [13]. To evaluate the clustering quality we have used the Davies Bouldin index [14] which needs to be minimized [15]. Throughout our experiments we aimed to identify the optimal values of the adjustable parameters associated to the Flower Pollination-based clustering algorithm such that the best clustering results are obtained. Thus we performed a series of experiments in which we have varied the values of the following adjustable parameters: *noCl* – the number of clusters, *popSize* – population size, *noIt* – the number of algorithm iterations, *p* – switch probability value, τ – global pollination threshold, λ - local pollination threshold, and *pc* – percent of the worst solutions in the population that will be replaced with randomly generated solutions. Tables 2 and 3 illustrate the best results obtained for 600 and 1000 recipe documents, where *t* refers to the clustering time measured in seconds, and *DBi* refers to the value of the Davies Bouldin index.

Table 2. Experimental results for 600 documents

noCl	***popSize***	***noIt***	***p***	***τ***	***λ***	***pc***	***t(s)***	***DBi***
20	20	20	0.8	0.6	0.7	30	136	3.265
20	15	20	0.8	0.6	0.8	30	81	3.656
20	20	20	0.6	0.6	0.7	30	135	3.692
20	15	20	0.65	0.6	0.7	30	90	3.883
20	15	30	0.65	0.6	0.7	30	148	3.071
15	15	30	0.75	0.6	0.7	30	154	3.893
15	15	30	0.75	0.7	0.6	30	163	3.589
10	20	30	0.75	0.7	0.6	30	181	3.66
20	20	30	0.8	0.7	0.5	20	140	3.478
20	25	30	0.8	0.7	0.5	20	192	3.487

By analyzing the experimental results we can notice that the best configuration of adjustable parameters that provides good results for the experiments performed on both 600 and

1000 documents is the one highlighted with grey in Tables 2 and 3.

For example, a cluster obtained for the best configuration of adjustable parameters from Table 2 contains the following recipes: Beer and Cheese Soup, Canadian Yellow Split-Pea Soup, Clear Tomato Consomme, Cold Peach Strawberry Soup, Green Tomato Soup, Lentil and Rice Soup, Minted Curried Fresh Pea Soup, Pizza Lovers Soup, Potato-Broccoli Soup, Scalloped Canned Soup, Tomato Cheese Soup.

Table 3. Experimental results for 1000 documents

noCl	*popSize*	*noIt*	*p*	*τ*	*λ*	*pc*	*t(s)*	*DBi*
20	20	20	0.8	0.6	0.7	30	272	3.638
20	15	20	0.8	0.6	0.8	30	193	3.406
20	20	20	0.6	0.6	0.7	30	245	3.927
20	15	20	0.65	0.6	0.7	30	198	3.745
20	15	30	0.65	0.6	0.7	30	309	3.61
15	15	30	0.75	0.6	0.7	30	281	3.648
15	15	30	0.75	0.7	0.6	30	258	3.602
10	20	30	0.75	0.7	0.6	30	323	3.11
20	20	30	0.8	0.7	0.5	20	416	3.478
20	25	30	0.8	0.7	0.5	20	411	3.63

VI. CONCLUSIONS

In this paper we have proposed a Flower Pollination-based algorithm for clustering documents based on their syntactic similarity. We have modeled the document clustering problem as an optimization problem in which a solution is represented by a set of clusters of documents and is evaluated using a fitness function based on the cosine similarity metric. The proposed clustering algorithm has been evaluated on a set of documents containing food recipes by using the Davies Bouldin index.

Acknowledgement. Living Joint Programme project DIET4Elders [www.diet4elders.eu] and was supported by a grant of the Romanian National Authority for Scientific Research, CCCDI – UEFISCDI, project number AAL16/2013. This document is a collaborative effort. The scientific contribution of all authors is the same.

REFERENCES

[1] X. S. Yang, “Flower Pollination Algorithm for Global Optimization”, *Unconventional Computation and Natural Computation* 2012, Lecture Notes in Computer Science, vol. 7445, pp. 240-249, 2012.

[2] W. Song, S. C. Park, “Genetic algorithm for text clustering based on latent semantic indexing”, *Computers & Mathematics with Applications*, vol. 57, issues 11–12, pp. 1901–1907, 2009.

[3] D. Ingaramo, M. Errecalde, L. Cagnina, P. Rosso, “Particle Swarm Optimization for Clustering Short-text Corpora”, *Proceedings of the 2009 Conference on Computational Intelligence and Bioengineering: Essays in Memory of Antonina Starita,* pp.3-19, 2009.

[4] M. L. Errecalde, D. A. Ingaramo, P. Rosso, “A new AntTree-based Algorithm for Clustering Short-text Corpora”, *Journal of Computer Science & Technology*, vol. 10, issue 1, 2010.

[5] S. Fong, S. Deb, X. S. Yang, Y Zhuang, “Towards Enhancement of Performance of K-Means Clustering Using Nature-Inspired Optimization Algorithms”, *The Scientific World Journal*, vol. 2014, pp. 1-16, 2014.

[6] J. Senthilnath, S.N. Omkar, V. Mani, “Clustering using firefly algorithm: Performance study”, *Swarm and Evolutionary Computation Journal*, vol. 1, issue 3, pp. 164–171, 2011.

[7] X. S. Yang, S. Deb, “Cuckoo search via Lévy flights”, *World Congress on Nature & Biologically Inspired Computing*, pp. 210–214, 2009.

[8] X. S. Yang, “A New Metaheuristic Bat-Inspired Algorithm, Nature Inspired Cooperative Strategies for Optimization”, Studies in Computational Intelligence, vol. 284, pp. 65-74, 2010.

[9] X. S. Yang, “Nature-Inspired Metaheuristic Algorithms”, Luniver Press, 2008.

[10] H. Azzag, N. Monmarche, M. Slimane, G. Venturini, and C. Guinot, “AntTree: A New Model for Clustering with Artificial Ants”, *Proceedings of the CEC 2003*, pp. 2642–2647, 2003.

[11] Stanford Log-linear Part-Of-Speech Tagger, Available: http://nlp.stanford.edu/software/tagger.shtml.

[12] WEKA, Available: http://www.cs.waikato.ac.nz/ml/index.html.

[13] Squirrel's RecipeML Archive, Available: http://dsquirrel.tripod.com/recipeml/indexrecipes2.html

[14] L. D. Davies, W. D Bouldin, “A Cluster Separation Measure”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, issue 2, pp. 224–227, 1979.

[15] F. Kovacs, C. Legany, A. Babos, “Cluster validity measurement”, *Proceedings of the* 6th *International Symposium of Hungarian Researchers on Computational Intelligence*, 2005.