# TSP을 이용한 효율적인 군집화 기법

리마진°, 정혜진, 김용성
전북대학교 공과대학 전자정보공학부
limajian@hotmail.com, {arayes,yskim}@chonbuk.ac.kr

# A Solution Technique Method Effective Clustering with Characteristic of TSP

Ma-Jian Li°, Hye-jin Jeong, Yong-Sung Kim
Department of Computer Science, 1Chonbuk National University,Korean
limajian@hotmail.com, {arayes,yskim}@chonbuk.ac.kr

요 약

원하는 정보를 보다 빠르게 찾기 위해서 활용하는 방법 중에 하나가 군집화이다. 군집화를 보다 효과적으로 할 수 있다면, 군집화내에서 원하는 정보를 보다 쉽게 얻을 수가 있다. 따라서, 본 논문에서는 군집화하기 위한 여러 가지 방법 중에서 TSP(Traveling Salesman Problem)를 이용해서 문서를 보다 정교하게 군집화하는 알고리즘을 제안하고, 제한된 알고리즘을 온톨로지 기반으로 실험하여 그 효율성을 입증하였다.

## 1. Introduction

Follow the rapid prevalence of internet, applications needs to be organized efficiently. Data search and management became more and more useful everyday; some webs like Google and Vivisimo are great helpful to human work. They have brought a new interest in the development of more intelligent and efficient real time web clustering algorithms. As one of the hotspots and key techniques in information retrieval (IR) and web knowledge management, the text clustering method has been used for semantic web and text ontology fields.

A clustering[1, 2, 3] method refers to the algorithm used to form clusters. Some of the clustering methods are single linkage, complete linkage; Wards method, Centroid clustering, median clustering etc. Each method has its own peculiarities. Some algorithm employed for clustering like k-means, ant system (AS), GAs and minimal spanning tree (MST). K-means algorithm [4] is one of the most widely used, attempts to solve the clustering problem. It put n objects in m dimensional space. Then partition them into a fixed number of clusters K known in advance by distance to each center [5]. Among the many bio-inspired techniques, ant-based clustering algorithms [6, 7] have received special attention from the community over the past few years for two main reasons. Firstly, they are particularly suitable to perform exploratory data analysis and, secondly, they still require much investigation to improve performance, stability, convergence, and other key features that would make such algorithms mature tools for diverse applications. GAs are adaptive heuristic search algorithm premised on the evolutionary ideas of natural selection and genetic. Genetic and other evolutionary algorithms have been earlier used for pattern classification [8]. Genetic also adapts to solve TSP problem well as a very fast process [9, 10]. The basic concept of GAs is designed to simulate processes in natural system necessary for evolution, specifically those that follow the principles first laid down by Charles Darwin of survival of the fittest. As such they represent an intelligent exploitation of a random search within a defined search space to solve a problem.

## 2. Related Work

In clustering text document, one of characteristic of cluster should like a ball in space. More similar documents will more closes. If there is a shortest line could go through all the points, the line has to pass clusters one by one for shortest. What we need to do, is to part the line for categorization. Just like searching the breakpoint.

The traveling salesman problem (TSP) asks for the s hortest route to visit a collection of cities and return to the starting point. Just as what we need.

Suppose that we put documents into m dimensions space by Euclidean distance, which have similar topic documents will closer together than other topics. When we find feasible solutions of TSP, the route must pass similar topic documents first which required shorter distance. Below fig. 1 and fig. 2 show the result in 2-dimensions.

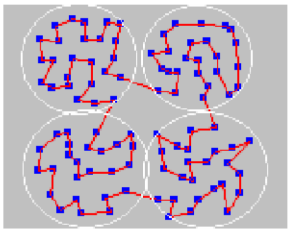
Fig. 1 no across status

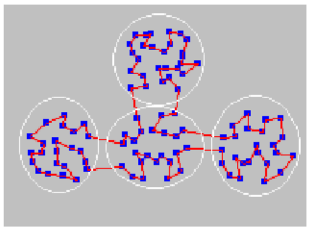
Fig. 2 across status

For aim at high-quality result, we compute weight between documents onto ontology. The weight value between two documents is more large more similar. We use the value of 1 minus weight to replace Euclidean distance between documents. Then documents replace of cities in TSP program.

## 2.1 Ontology

The word "ontology" seems to generate a lot of controversy about artificial intelligence [11]. Some webs like Google and Vivisimo are used widely everyday for human work. They have brought a new interest in the development of more intelligent and efficient real time web clustering algorithms.

It has a long history in philosophy, in which it refers to the subject of existence. Ontology consist of Concepts, relations, concept hierarchy and function construct ontology with connection method of among the concepts and axioms. It is also often confused with epistemology, which is about knowledge and knowing. "Ontology" term is explicit formal specifications of the terms in the domain and relations among them.

We used the term ontology to mean a specification of a conceptualization. That is, ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents. This definition is consistent with the usage of ontology as set-of-concept-definitions, but more general. And is certainly a different sense of the word than its use in philosophy. Ontology are often equated with taxonomic hierarchies of classes, but class definitions, and the subsumption relation, but ontology need not be limited to these forms. Ontology are also not limited to conservative definitions, that is, definitions in the traditional logic sense that only introduce terminology and do not add any knowledge about the world. To specify a conceptualization one needs to state axioms that do constrain the possible interpretations for the defined terms.

## 2.2 Introduce TSP

TSP is a problem in discrete or combinatorial optimization. It is a prominent illustration of a class of problems in computational complexity theory which are classified as NP-hard. Basically, the proposed algorithm either ant colony optimization (ACO) or GAs and can explore and exploit search spaces [14]. It has both the advantage of ACO, the ability to find feasible solutions and to avoid premature convergence, and that of GA, the ability to avoid being trapped in local optima. In this process, we use GA to find feasible solutions of TSP, because we just need a mainly outline for reduce time losing.

According to the fig 1 and fig 2, principium of our method is based on more similar documents will more closely. If we could find breakpoint in each two different clusters exactly, we could get more precise results and make the clustering more efficient. In the real time text clustering, across status happen frequently, how we could find breakpoint exactly is one of keystone in this paper. (We will discuss in section 3)

### 2.2.1 Computational Generation of TSP

In order to measure the performance of TSP, we employed GA to test the subsets from the Reuter-21578 text collection.

Following more amount of documents, the generation increase faster. The best result of TSP is not allowed by time. For proving this has worth value to clustering, we choose mainly feasible solution to experiment.

## 3. Design the New Categorization

After we use TSP onto ontology, we should know how TSP result works. After we get feasible solutions of TSP, how could we divide them into itself cluster? How to find breakpoint exactly in first? The breakpoint belongs to which segment. Even if in one cluster, each two points also maybe have long distance. Obvious, a good fitness function is very important at first.

## 3.1. Fitness Function

In fig 3, it is ant clustering algorithm [7], every ant is in a domain, there are 8 grids along each ant.

| | | |
|---|---|---|
| | O | |
| | O | O |
| O | | |

Fig. 3 Ant clustering algorithm

Some grids have other ants. We call these ant is neighbors. $f$ value represents if this ant similar with neighbors. More amount of similar topic documents (closer) $f$ value will bigger. Various topic documents (farer) will reduce $f$ value. So we modify this fitness function to adapt our process.

$$f(i) = \sum_j (1 - \frac{d(i,j)^2}{a_{ij}}) \quad \text{......(1)}$$

$$a_{ij} = (\frac{1}{n}\sum_{k=1}^{n} d(i,k))(\frac{1}{n}\sum_{k=1}^{n} d(j,k)) \quad \text{......(2)}$$

$d(x,y)$ means distance of document $x$ and $y$, it will work out at preprocess stage.

We already know solutions of TSP. set $x_k(k=1,2,3,,,99)$ domains. Put front N and behind N documents to be neighbors of $x_k$ into $x_k$ domain. Compute all the documents by sequence of TSP. In addition, the longest edge is no mean. We could delete it. And then, set one side of this edge as beginning. The N value relates result should like below.

$$N = docs \,\%\, 100 + 1 \cdots\cdots(3)$$

**Example:** We choose 100 texts from Reuter 21578 data set, which belonging to four topics (ship, sugar, trade, earn). We show it in Table 1.

Table 1 Reuter 21578 data set

| docs | ship | sugar | trade | earn | generation |
|---|---|---|---|---|---|
| 100 | 25 | 25 | 25 | 25 | 10000 |

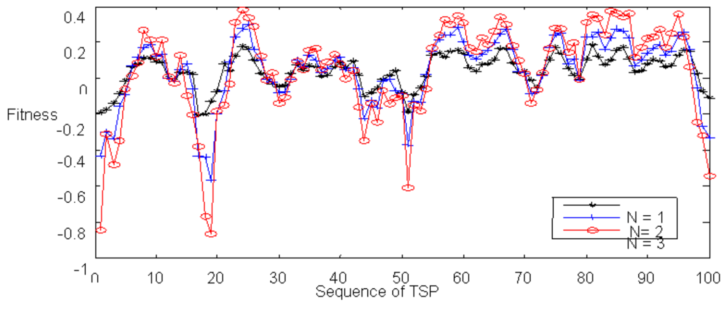

Fig. 4 Documents fitness against the sequence of TSP with various value of n.

In the Fig 4, each point is a document. It is 100 docs, there are about 15 to 25 bad-points (introduce in section 3.3). When N equals 1, the breakpoint is too many and inaccurate. On the opposite, when N equals 3, the breakpoint is too less and more exactly.

Obviously, for text clustering, a lot of compact segment are cut by the solution of TSP. when N equals 2, the result is better than others. We also do experiment for the 50, 150, 200, 250 or 300 docs. The result shows us function about N is better.

## 3.2. Breakpoint

Define: we connect all the documents with a shortest path. So some documents only similar with one side of document, we call these document breakpoints.

Because the solution of TSP is from a compact segment to another one, so when two or several different clusters are put into a domain, the f value will reduce, nevertheless, oppositions will increase. The trough of fig. 4 is formed by this reason. From define we know, breakpoints should be happened in the wave-trough most time or around wave-trough sometimes in the wave.

After we find breakpoint, we should decide which segment is fit to together with each breakpoint. The process below expresses how to deal with breakpoint.

Algorithm. 1 finding the best segment for breakpoint

```
(1) Suppose there is m segments total.
(2) if( $f_{front}$ >0 || $f_{behind}$ >0)
        Which one is bigger, put breakpoint with it.
    else
        Put each segment into $k_{breakpoint}$ domain
    endif
(3) $f_{avg} = f / length_{segment}(segment = 1,2....m)$
//Length means the amount documents of segment.
//Which $f_{avg}$ value is bigger, put breakpoint with it.
```

## 3.3. Self Repair and Bad-point

An idealizing result of TSP should only form 4 segments. But in the real world, some documents have the keywords about two or three topics all together. It is hard to decide which topic is the most similar before us clustering obvious feature. We call this documents are bad-point. As we know, the segments are not the last result of text clustering. Some segments have similar topic, we need incorporate them. But in the segments, it is hard to avoid wrong documents in each segment. If we want a high quality clustering, we should repair each segment first.

In text clustering, always exist some documents has inconspicuous character with all other documents. We define these points to be Bad-point. Bad-point always is a difficult problem of text clustering. We use below function to move documents to the most similar segment. It will find a biggest value of R.

$$\begin{cases} R = f / (\text{length-1})^2, & \text{same row} \\ R = f / \text{length}^2, & \text{different row} \end{cases} \quad \text{...........(4)}$$

If every $f$ is bigger than 0, the more similar documents will get together.

If all the $f$ are smaller than 0, the Bad-points will be together, because Bad-points always have a negative $f$ in each segment.

When we incorporate two segments once, we use self repair function once. At last, which documents have available feature will stay together. The bad-points will stay together. Then use each obvious feature to decide bad-point's cluster.

One coincidence, when we get result of TSP, we delete the longest edge, and set one side of this edge as beginning. But the longest edge always connects by bad-points, because this point has a longest distance to other point. So the first segment will collects all the bad-points after self repair step. Experiment proves this point.

### 3.4. Incorporate

Incorporate is the last function of our process. After self repair step, we think we already have enough feature of each segment. We random choose two segments. Put these documents into each document domain. If $\rho$ percent document's f value bigger than before, we incorporate the two segments to a new segment. ($\rho$ is a constant, It always set above 90 %.) Then, use self repair function to check all segments.

At last, there are a bad-point group and other clusters with available feature. Put each clusters into each bad-point domain. Choose the biggest $f$ value for the bad-point.

## 4. Apply TSP onto Ontology

The using TSP onto ontology problem can thus be viewed as the problem of identifying an appropriate similarity relation on given data. Documents are more similar more closer.

This section describes information indexing and categorization. We show the algorithm as follows

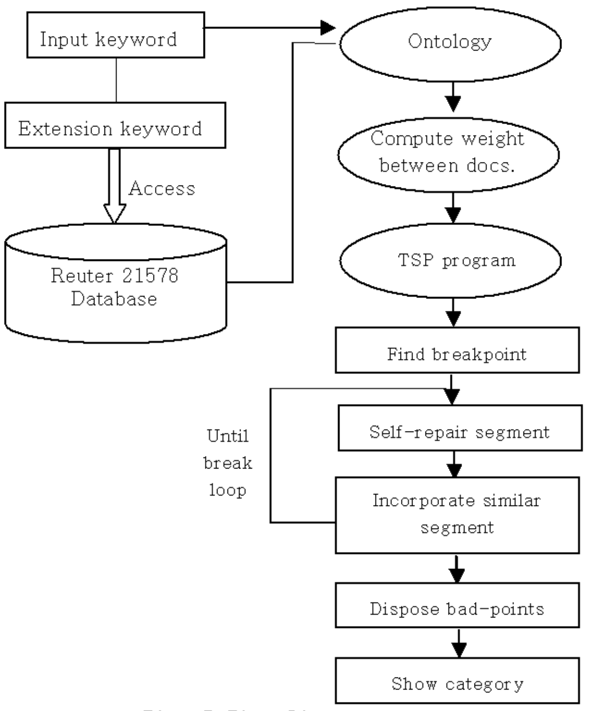

Fig. 5 Flow Diagram

### 4.1 Preprocessing step

(1) First, input Reuter 21578 data set into database. Then input a sentence of language. We need to know keyword occurrence location information and frequency calculation of Reuter 21578.
(2) Select domain data set in database.
(3) Calculate weight by keyword occurrence location and important degree between documents.
(4) Depend on weight to decide distance value. Put these values into TSP process. It is operation by GAs algorithm to find sequence of documents.

### 4.2 Categorization

We give our PC process to show how categorization working in Algorithm 2. The computer language is C++.

Algorithm 2. PC Algorithm work on C++

```
(1) Compute fitness of each doc. Prepare to find
    breakpoint.
      doc_id = 1;
      Begin
         do
         Fitness(doc_id);
         doc_id := doc_id + 1;
         while (doc_id := Maxdoc_id;)
      end;
(2) Apply the fitness result to find breakpoint
      doc_id := 1;
      Begin
```

```
          do
          If (Fitness(doc_id)<both side Fitness(doc))
array_Breakpoint{} := doc_id;
          until doc_id := Maxdoc_id;
       end;
(3) After find breakpoints. Then part docs to segments.
    breakpoint_id := 1
    segment_id := 1
       Begin
          do
          For(int i; breakpoint-id - 1 < i <breakpoint_id+1 )
          Segment_id{} := doc_id;
              endfor
              breakpoint_id := breakpoint_id + 1;
              until breakpoint_id := masbreakpoint_id;
       end;
(4) Now, have segments and data set of breakpoint.
have to judge breakpoint belong to which segment best;
   breakpoint_id := 1;
   segment_id := 1;
   begin
      do
      begin
         do
         segment_id   := segment_id + 1;
         if(f(breakpoint_id)>temp)
             temp := f(breakpoint_id)
             temp_id := segment_id;
         endif
         until segment_id := maxsegment_id;
     end;
     move breakpoint to segment_id := temp_id;
     breakpoint_id := breakpoint_id + 1;
     until breakpoint_id := maxbreakpoint_id;
   end;
(5) The experiment already forms some small clusters.
They need self-repair themselves.
   doc_id := 1;
   begin
      do
      Self-repair ( );
      doc_id := doc_id + 1;
      until self-repair function finish;
   end;
(6) After last step, incorporate small clusters to reduce
number of segment. Then repeat last step until numbers of
segment is no change.
   random choose 2 segment compute fitness()
   if   p% doc of segment's f() increase
// P is a variable, follow incorporate speed to reduce.
 Its area is bigger than 90%   and smaller than 100%.
        incorporate them
   else
       choose again; until 500times can't find incorporate
objects.
   Endif
(7) Last step, all the bad-points is centralized in first
segment. But in this time, all other segment already have
enough character. We could find right segment more
accurate for bad-point.
   bad-points_id := 1;
   segment_id := 1;
   begin
      do
      begin
         do
         if(f(bad-point_id)>temp)
             temp := f(bad-point_id)
             temp_id := segment_id;
         endif
         segment_id := segment_id + 1;
         until segment_id := maxsegment_id;
      end;
      move bad-point_id to segment_id := temp_id;
      bad-point_id := bad-point_id + 1;
      until bad-point_id := maxbad-point_id;
   end;
(8) At last, show the categorization result.
```

## 5. Experiment and result

I proposes methods with analyzed by experiment with Reuter 21578 data set and compare with existing works by Gas, $k$ -means. Data_set has 100 texts belonging to four topics (ship, sugar, trade, earn). After being processed by word extraction, stop words removal, and stemming, there are 2,246 terms in the vocabulary respectively. Table 2 shows the Data_set we used in our experiments.

Table 2 The number of texts, topics in data_set

| Name | Texts | Topic | Data set |
|---|---|---|---|
| Data_set | 200 | 4 | Ship(0-49)sugar(50-99) trade (100-149)earn(150-199) |

We use Fig.6 to show the result of TSP modify method. Compare with GA and K-means by Fig.7 and Fig.8.

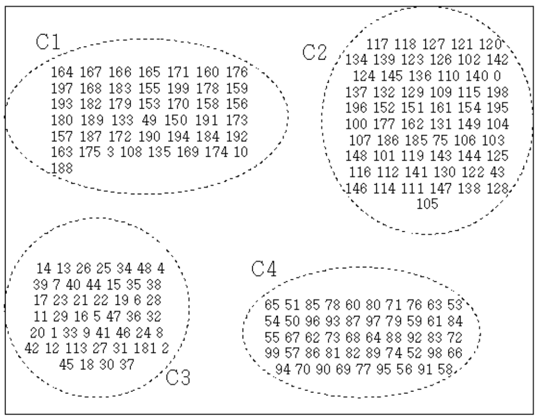

Fig. 6 TSP is applied to Data_set and obtains 4 clusters in the end

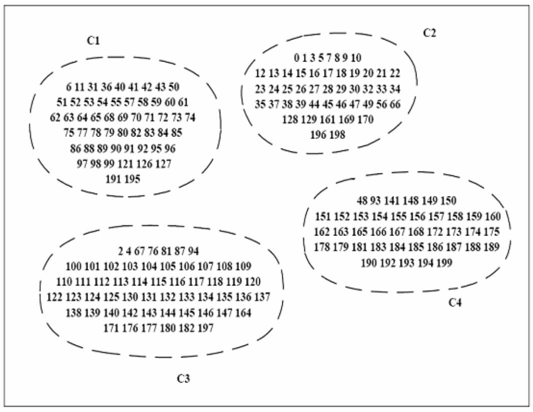

Fig. 7 GA is applied to Data_set and obtains 4 clusters in the end

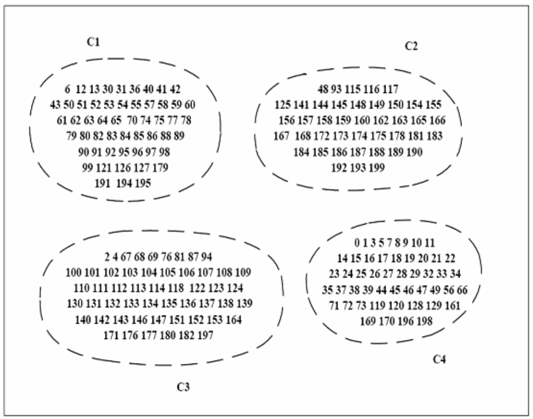

Fig. 8 K-Means is applied to Data_set (set 4 clusters in advance)

Table 3 Relate result

| | Precision (%) | Recall(%) | F1 (%) |
|---|---|---|---|
| TSP method | 90.53 | 90 | 90.26 |
| GA | 86.53 | 87.22 | 86.87 |
| $k$ -means | 78.65 | 80.12 | 79.38 |

From Table 3, we could see our result is better than GA and $k$ -means. This example result proves our method could be working well in Reuter 21578 data set. We will do more examples in other dataset in the future.

## 6. Conclusion and Future Work

This paper has presented a novel approach to clustering with the characteristic of TSP. This work differs from existing approaches in that it automatically forms cluster by itself functions. I compare its performance with the other feature selection methods in text categorization. The experiments show that our program has a better performance and simpler computation than the other feature selection methods. There are two benefits to this method.

The use of part-to-part algorithm that converts the complexity program into clustering allows computation to be availed. The other is judging the bad-point by formed feature to increase quality of result. This paper is first step to try a novel method to solve text categorization with TSP characteristic. Many detail need to optimize. I will test other data set in the future work.

## 7. References

[1] Ju-in Youn, He-Jue Eun, Yong-Sung Kim, "Fuzzy Clustering for Documents based on Optimization of Classifier Using the Genetic Algorithm," ICCSA 2005, LNCS 3481

[2] Ju-in Youn, He-Jue Eun, Yong-sung Kim, "A adoptive Document classification system base on Ontology Constructed by Fuzzy Function and Fuzzy Relations," CW2004, pp.182-187,2004.

[3] M. A. Hearst and J. O. Pedersen. "Reexamining the cluster hypothesis," In Proceedings of SIGIR '96, pp.76—84, 1996.

[4] S. Z. Selim and M. A Ismail, "K-means-type Algorithm: Generalized Convergence Theorem and Characterization of Local Optimality," IEEE Trans on pattern Anal. Intell. 6, pp. 81-87, 1984.

[5] K. Fukunaga, "Introduction to Statistical Pattern Recognition," Academic Press, New York, 1990.

[6] André L. Vizine1,2, Leandro N. de Castro1,2, Eduardo R. Hruschka1, Ricardo R. Gudwin2 "Towards Improving Clustering Ants: An Adaptive Ant Clustering Algorithm," 2004

[7] XU Xiao.Hua , CHEN Lin, "An Adaptive Ant Clustering Algorithm," ISSN 1000-9825, CODEN RuXUEW Journal of Software, Vo1.1 7, No.9, September 2006PP1884—1889 DOI : 10. 1360~osl71884 2006

[8] Ujjwal Mauilk and Sanghamitra Bandyopadhyay, "Performance Evaluation of Some Clustering Algorithms and Validity Indices," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no.12, 2002.

[9] HU Xiao—bingl, HUANG Xi—yue "Solving TSP with Characteristic of Clustering by Ant Colony Algorithm," JOURNAL OF SYSTEM SIM ULATION VbL16 NO. 12 2004

[10] Hitachi, Ltd. Ikuo YOSHIHARA "A fast TSP solver using a genetic algorithm," 1997.

[11] R. Baeza-ates, B. Ribeiro-Neto, "Modern Information Retrieval," p.230-255, 1998.

[12] Zne-Jung Lee*, "A Novel algorithm of Genetic Ant Colony Optimization (GACO) for Traveling Salesman Problema," 2004