Web Page Classification with an Ant Colony Algorithm

Nicholas Holden and Alex A. Freitas

Computing Laboratory, University of Kent
Canterbury, CT2 7NF, UK
{nph4, A.A.Freitas}@kent.ac.uk

Abstract. This paper utilizes Ant-Miner – the first Ant Colony algorithm for discovering classification rules – in the field of web content mining, and shows that it is more effective than C5.0 in two sets of BBC and Yahoo web pages used in our experiments. It also investigates the benefits and dangers of several linguistics-based text preprocessing techniques to reduce the large numbers of attributes associated with web content mining.

1. Introduction

The amount of information available on the web is huge and growing each year. At present Google searches more than 4.2 billion pages. As the web has grown, the ability to mine for specific information has become almost important as the web itself. Data mining consists of a set of techniques used to find useful patterns within a set of data and to express these patterns in a way which can be used for intelligent decision making [1], [2]. In this project the knowledge is represented as classification rules. A rule consists of an antecedent (a set of attribute values) and a consequent (class):

```
IF <attrib = value> AND ... AND <attrib = value> THEN <class>.
```

The class part of the rule (consequent) is the class predicted by the rule for the records where the predictor attributes hold. An example rule might be IF <Salary = high> AND <Mortgate = No> THEN <Good Credit>. This kind of knowledge representation has the advantage of being intuitively comprehensible to the user. This is important, because the general goal of data mining is to discover knowledge that is not only accurate, but also comprehensible to the user [2], [1]. In the classification task, the goal is to discover rules from a set of training data and apply those rules to a set of test data (unseen during training), and hopefully predict the correct class in the test set.

In this project, the goal is to discover a good set of classification rules to classify web pages based on their subject. The main classification algorithm to be used in this paper is Ant-Miner [3], the first Ant Colony Optimisation (ACO) algorithm for discovering classification rules. Investigating the use of Ant-Miner in web mining is an important research direction, as follows. First, an empirical comparison between Ant-Miner and two very popular rule induction algorithms (C4.5 and CN2), across six data sets, has shown that Ant-Miner is not only competitive with respect to predictive accuracy, but also tends to discover much simpler rules [3], [4]. However, that comparison

involved only “conventional” data mining – i.e., mining structured data sets. Web mining is more challenging, because it involves unstructured or semi-structured text found in web pages. In addition, there are a potentially very large number of attributes (words) associated with web pages, and a theoretical analysis of Ant-Miner (under very pessimistic assumptions) shows that its computational time is quite sensitive to the number of attributes [3]. Hence, it is important to understand how scalable Ant-Miner is to data sets with a large number of attributes in practice, in a challenge real-world domain such as web mining. Finally, it is important to investigate the influence of different text preprocessing techniques (which reduce the number of attributes) in the performance of Ant-Miner. This is also addressed in this paper.

2 Web Mining and Linguistic Text Preprocessing

Web mining can be split into three main categories: content mining, usage mining, and structure mining [5]. Content mining involves the automatic analysis of the text stored in the files (i.e. HTML and email), images and any other media available on the web. Usage mining [6] analyses access logs from web servers to discover the patterns that users make when they use the web. Structure mining analyses how web pages link to each other through hyperlinks, for example.

This project focuses on web-content mining. Also, like most web-content mining projects, we mine only text – not images and other media. Web-content mining is a challenging task, as follows. Firstly the amount of attributes (words) is unusually high in comparison to simpler data mining applications. The number of possible classification rules is exponential on the number of words, so that the search space quickly becomes very large. Secondly the English language (all languages in general) is very complicated. There is no program at the moment that can fully understand the meaning of a given web page. We can only hope for a relatively simple interpretation.

There is some hope, however, that html code gives us clues to help us cut down the number of attributes [7]. The authors of web sites leave summaries or descriptions of the web page in <meta> tags: in <meta NAME="keywords"> the content field gives us a list of keywords the author thinks is suitable for the page; and there is also <meta NAME="description"> which gives us hopefully a good overview of the page’s content. Going further, it is possible to apply linguistics-based text preprocessing techniques to select the most relevant words from the text.

WordNet is an electronic lexicon that contains several relationships between words [8]. It is an attempt to map the human understanding of words and their relationships into an electronic database. In this project we have used three linguistic resources from WordNet to preprocess the data, as follows.

Firstly, we used the morphological processor of WordNet to perform stemming (i.e., removing the word suffixes). This is useful as instead of having, e.g., the words *borrow*, *borrowing* and *borrowed* we would like to just have the word *borrow* added to our list of attributes. This cuts down the number of attributes and allows us to find patterns more easily. We may not be able to find a pattern with these separate words, but when they are amalgamated together into one attribute, a pattern may emerge.

Secondly, we used WordNet to identify all the nouns in the text. As the amount of words is so high in web mining, it may be useful to only use nouns as attributes, as they are usually the subject of a sentence. Hence we trade off the completeness of the information against the ability to find more useful patterns within a given time.

Thirdly, we use WordNet to capture the idea of a given word in a more generic form and use that instead of the word itself. If different pages have the same idea behind what they contain, then this should allow us to find more trends in the data. For example, if one page contains the words: window, roof, and door, and another web page contains the words chimney, room and brick then we should be able to use WordNet to find the relationship or root of the tree, the word house. As you can see this would reduce the number of attributes from six to just one. Although this is potentially the most rewarding technique discussed, it is also the most risky. If WordNet finds the wrong relationship between the words we may end up with the wrong root word. To perform this kind of word generalization, we use the hypernym/hyponym ("is a") relationship of WordNet, where words are arranged in a tree-like structure.

```
OriginalWordsList = [Words From current web page];
GeneralisedWordsList = [];
RelationshipMaxLength = 2;
WHILE (OriginalWordsList.Size > 2)
  BestRelationShip = NULL;
  CurrentWord = remove first word from OriginalWordsList
  FOR (i = 0; i < OriginalWordsList.Size)
    Get all relationships between all senses of CurrentWord
        and all senses of OriginalWordsList element i, and
        for each relationship compute the number of edges
        in the WordNet taxonomy between CurrentWord and
        OriginalWordsList element i
    Get the relationship with the shortest number of edges,
        out of all relationships identified in previous step
    IF (number of edges in the shortest relationship ≤
        RelationshipMaxLength)
        Save shortest relationship as BestRelationship:
        BestParent = the parent (generalized) word
        BestSecondWord = OriginalWordsList element i
  END FOR
  IF (BestRelationship ≠ NULL)
    Add BestParent to GeneralisedWordList
    Remove BestSecondWord from the OriginalWordsList
  ELSE
    Add CurrentWord to GeneralisedWordsList
END WHILE
```

Alg. 1. Finding the best word-generalisation relationships

We have developed an algorithm (implemented using the JWNL library) to search for the hypernyms (generalizations) of a pair of words and return the best hypernym. The pseudo-code of the algorithm is shown in Algorithm 1. For each word in the current web page, the algorithm finds the "best hypernym" that is generalizing both that word and another word in the page. The best hypernym is the one associated with the smallest number of edges in the path linking the two generalized words via the hypernym. The best hypernym for each possible pair of words is then added to the "Gen-

eralisedWordList". At the end of the algorithm this list contains the generalized words that will replace their corresponding base words in the representation of the web page.

3 The Ant-Miner Algorithm

In nature ants are seen creating "highways" to and from their food, often using the shortest route. Each ant lays down an amount of pheromone and the other ants are attracted to the strongest scent. As a result, ants tend to converge to the shortest path. This is because a shorter path is faster to transverse, so if an equal amount of ants follow the long path and the short path, the ants that follow the short path will make more trips to the food and back to the colony. If the ants make more trips when following the shorter path, then they will deposit more pheromone over a given distance when compared to the longer path. This is a type of positive feedback and the ants following the longer path will be more likely to change to follow the shorter path, where scent from the pheromone is stronger [9], [10].

The Ant-Miner algorithm takes the ideas from the Ant Colony paradigm and applies them to the field of data mining. Instead of foraging for food the ants in the Ant-Miner algorithm forage for classification rules, and the path they take correspond to a conjunction of attribute-value pairs (terms). A high-level pseudocode of Ant-Miner is shown in Algorithm 2. A detailed description of the algorithm can be found in [3].

Ant-Miner starts by initializing the training set to the set of all training cases (web pages, in this project), and initializing the discovered rule list to an empty list. Then it performs an outer Repeat-Until loop. Each iteration of this loop discovers one classification rule. This first step of this loop is to initialize all trails with the same amount of pheromone, which means that all terms have the same probability of being chosen (by the current ant) to incrementally construct the current classification rule.

```
TrainSet = {all training cases};
DiscoveredRuleList = [];   /* initialized with empty list */
REPEAT
   Initialize all trails with the same amount of pheromone;
   REPEAT
      An ant incrementally constructs a classification rule;
      Prune the just-constructed rule;
      Update the pheromone of all trails;
   UNTIL (stopping criteria)
   Choose best rule out of all rules constructed by all ants;
   Add the best rule to DiscoveredRuleList;
   TrainSet = TrainSet – {cases correctly covered by best rule};
UNTIL (stopping criteria)
```

Alg. 2. High-level pseudocode of Ant-Miner

The construction of an individual rule is performed by the inner Repeat-Until loop, consisting of three steps. First, an ant starts with an empty rule and incrementally constructs a classification rule by adding one term at a time to the current rule. In this step a $term_{ij}$ – representing a triple $<Attribute_i = Value_j>$ – is chosen to be added to the current rule with probability proportional to the product of $\eta_{ij} \times \tau_{ij}(t)$, where η_{ij} is the

value of a problem-dependent heuristic function for $term_{ij}$ and $\tau_{ij}(t)$ is the amount of pheromone associated with $term_{ij}$ at iteration (time index) t. More precisely, η_{ij} is essentially the information gain associated with $term_{ij}$ – see e.g. [1] for a discussion on information gain. The higher the value of η_{ij} the more relevant for classification $term_{ij}$ is and so the higher its probability of being chosen. $\tau_{ij}(t)$ corresponds to the amount of pheromone currently available in the position i,j of the trail being followed by the current ant. The better the quality of the rule constructed by an ant, the higher the amount of pheromone added to the trail positions ("terms") visited ("used") by the ant. (Rule quality is measured by *Sensitivity* × *Specificity* [3].) Therefore, as time goes by, the best trail positions to be followed – i.e., the best terms to be added to a rule – will have greater and greater amounts of pheromone, increasing their probability of being chosen to construct a rule.

The second step of the inner loop consists of pruning the just-constructed rule, i.e., removing irrelevant terms – terms that do not improve the predictive accuracy of the rule. In essence, a term is removed from a rule if this operation does not decrease the quality of the rule – as assessed by the same rule-quality measure used to update the pheromones of the trails. The third step of the inner loop consists of updating the pheromone of all trails by increasing the pheromone in the trail followed by the ant, proportionally to the rule's quality. In other words, the higher the quality of the rule, the higher the increase in the pheromone of the terms occurring in the rule antecedent.

The inner loop is performed until some stopping criterion(a) is(are) satisfied, e.g., until a maximum number of candidate rules has been constructed. Once the inner loop is over, the algorithm chooses the highest-quality rule out of all the rules constructed by all the ants in the inner loop, and it adds the chosen rule to the discovered rule list. Next, the algorithm removes from the training set all cases correctly covered by the rule, i.e., all cases that satisfy the rule antecedent and have the same class as predicted by the rule consequent. Hence, the next iteration of the outer loop starts with a smaller training set, consisting only of cases which have not been correctly covered by any rule discovered in previous iterations. The outer loop is performed until some stopping criterion(a) is(are) satisfied, e.g., until the number of uncovered cases is smaller than a user-specified threshold. The output of Ant-Miner is the discovered rule list.

4 Computational Results

4.1 Experimental Setup

A set of 127 web pages in three different classes (Education, Technology and Sport) were harvested from the BBC web site. This site was chosen for analysis because it is arranged in a rigid standard way, and all pages have standard tags which can be used for mining. The standard of writing is also high, making it possible to draw relationships between the content, the information in the Meta fields, and the class (subject) of the page. Some pages published by the BBC are released in more than one class, so a page that appears in, say, the Technology section may also appear in the Education section. In these cases the page in question is removed from the collected set.

We extracted, from each web page, a set of binary attributes (words). Each attribute represents whether or not the corresponding word occurs in a given web page. Since using all words occurring in any web page would produce a huge and impractical number of attributes, we used WordNet to perform the three kinds of linguistics-based text preprocessing discussed in section 2. We also performed controlled experiments to evaluate the effect of each of these preprocessing techniques, as follows.

First, we performed experiments with and without stemming. Second, we performed experiments using only nouns as attributes and using all kinds of words as attributes. In both cases, words that were not recognized by WordNet were presumed to be proper nouns. These proper nouns were left in, as they usually contain important and relevant names. Third, we performed experiments with and without the generalisation of words based on the hypernym relationship of WordNet (using Algorithm 1).

We also performed a basic text preprocessing that is often used in text mining, where stop words, as well as punctuation, were removed. Stop words are words that convey little or no useful information in terms of text mining – e.g. "*the*, *and*, *they*".

To gauge the accuracy of the discovered rules, a conventional five-fold cross-validation procedure was used [1]. Reported results are the average predictive accuracy in the test set over the five iterations of the cross-validation procedure. The following standard Ant-Miner settings [3] (except (d)) were used in all the experiments:

(a) No_of_Ants (number of ants, i.e. maximum number of rules evaluated) = 3000
(b) Min_cases_per_rule (minimum number of cases per rule) = 10
(c) Max_uncovered_cases (maximum number of uncovered cases) = 10
(d) No_rules_converg (number of identical consecutive rules required for indicating convergence) = 20. This parameter was increased from 10 (default value of Ant-Miner) to 20, to try and stop premature convergence to worse rules.

4.2 Results On the Influence of Linguistics-based Text Preprocessing Techniques

The experiments reported in this section evaluate the influence of different linguistics-based text processing techniques in the performance of Ant-Miner. Tables 1 and 2 report, for each setup, the number of attributes (after text preprocessing) and the average cross-validation accuracy with the standard deviation shown after the "±" symbol. In these figures, WN-generalization denotes WordNet generalization based on the hypernym relation. Title is where the words are harvested from the title field in the documents, Description is where the words are taken from the description field and Union is the union of the two sets of words (Title + Description).

Table 1: Ant-Miner Results in BBC web site – using only nouns

Test Setup	No. of attrib.	Accuracy
WN-generalisation, Title	41	77.34 ± 2.27
WN-generalisation, Description	125	68.01 ± 2.37
WN-generalisation, Union	188	70.42 ± 5.27
Stemming, Title	46	69.09 ± 5.92
Stemming, Description	159	71.00 ± 1.71
Stemming, Union	293	74.79 ± 2.86

Table 1 shows the accuracies from the different setups when using only nouns (rather than all kinds of words) to create attributes. There are two different ways to analyse this table. First, one can analyse the effect of using nouns from the web page Title only, nouns from the web page Description only, and nouns from both (Union) in the performance of Ant-Miner. There is no clear pattern associated with Title versus Description or Union. However, both when using WordNet generalization and when using Stemming, nouns from Union produced better results than nouns from Description only. Second, it is interesting to analyse the use of WordNet generalization versus the use of stemming as a heuristic to reduce the number of attributes. The use of WordNet was beneficial when the attributes contained only words in the Title. Indeed, WordNet generalization with Title produced the best result (77.34% of accuracy). However, the use of WordNet produced worse results than stemming when the attributes contained words in Description or in Union.

Table 2: Ant-Miner Results in BBC web site – using all words

Test Setup	No. of Attrib.	Accuracy
WN-generalisation, Title	47	81.00 ± 2.93
WN-generalisation, Description	163	68.69 ± 2.90
WN-generalisation, Union	226	67.81 ± 2.62
Stemming, Title	52	71.28 ± 6.04
Stemming, Description	188	74.29 ± 4.90
Stemming, Union	339	70.97 ± 4.04

Table 2 shows the accuracies from the different setups when using all kinds of words (except, of course, stop words) to create attributes. Again, there are two kinds of analyses to be made. First, one can analyse the effect of using Title only, Description only, and the Union of Title and Description in the performance of Ant-Miner. Unlike the results in Table 1, Table 2 shows that – both when using WordNet generalization and when using stemming – Union produces the worst results. Hence, it seems that when all kinds of words are used, Union leads to a degradation of accuracy because the search space becomes very large, i.e., the large number of attributes (which tends to have many irrelevant attributes) degrades Ant-Miner's performance.

Second, one can analyse the use of WordNet generalization vs. stemming. Similarly to Table 1, Table 2 shows that: (a) the use of WordNet was beneficial when the attributes contained only words in the Title –WordNet generalization with Title produced the best result (81.0% of accuracy); (b) the use of WordNet produced worse results than stemming when the attributes contained words in Description or in Union.

Hence, both Table 1 and Table 2 are evidence that WordNet generalization is a very effective heuristic when the attributes contain words from the Title only, which are the scenarios with the smallest sets of attributes used in our experiments. When the attributes contain words from Description and Union, the larger number of attributes seems to be a serious problem for WordNet generalization, leading to worse results than stemming. Indeed, the title of a web page tends to be a very compact description of its contents in only one sentence, possibly leading to fewer WordNet confusions between different senses of a word.

4.3 Results Comparing Ant-Miner and C5.0

The results with all kinds of words (Table 2) were better than the results with nouns (Table 1) in 4 out of 6 cases. Hence, we decided to focus on the results with all words and do an additional experiment, comparing Ant-Miner with the well-known C5.0 algorithm, implemented in Clementine (an industrial-strength data mining tool). The results of this experiment are reported in Table 3. C5.0 was run with the default settings for its parameters. To make the comparison as fair as possible, both algorithms used exactly the same training and test set partitions in each of the iterations of the cross-validation procedure. Table 3 reports the average cross-validation results with respect to both accuracy and simplicity – number of discovered rules and total number of terms (conditions) in all discovered rules. The reported rule count does not include the default rule for Ant-Miner or C5.0.

For each setup in Table 3, the best result is shown in bold. With respect to accuracy, Ant-Miner obtained the best result in three setups, and C5.0 obtained the best result in the other three setups. In 4 out of the 6 setups the difference between the two algorithms is not significant, since the accuracy rate intervals (taking into account the standard deviations) overlap. There were just two setups in which the difference in accuracy was significant (i.e. the accuracy rate intervals do not overlap), namely the first setup (WordNet generalization, Title, All words), where Ant-Miner significantly outperformed C5.0, and the last setup (Stemming, Union, All words), where C5.0 significantly outperformed Ant-Miner.

With respect to the simplicity of the discovered rule set, Ant-Miner discovered a significantly smaller number of rules in all setups. The total number of terms discovered by Ant-Miner was also significantly smaller than the number of terms discovered by C5.0 in all setups. This means that Ant-Miner has performed very well in terms of knowledge comprehensibility in comparison to C5.0. I.e., a user would find it much easier to interpret and possibly use the knowledge discovered by Ant-Miner.

Table 3: Comparison between Ant-Miner and C5.0 in BBC news web site, all words

Test Setup	Algorithm	Accuracy	No. of rules	Total No. of Terms
WordNet generalization, Title, All words	Ant-Miner	**81.00±2.93**	**3.0±0.00**	**9.40±1.91**
	C5.0	73.19±4.77	12.00±1.44	24.80±1.71
WordNet generalization, Description, All words	Ant-Miner	**68.69±2.90**	**3.0±0.00**	**12.40±2.58**
	C5.0	67.78±1.43	12.40±0.50	27.20±1.46
WordNet generalization, Union, All words	Ant-Miner	67.81±2.62	**3.0±0.00**	**11.60±2.40**
	C5.0	**71.83±2.08**	11.60±0.40	23.40±0.87
Stemming, Title, All words	Ant-Miner	71.28±6.04	**3.0±0.00**	**12.13±1.70**
	C5.0	**77.08±4.48**	14.00±0.54	26.4±0.74
Stemming, Description, All words	Ant-Miner	**74.29±4.90**	**3.0±0.00**	**11.66±2.56**
	C5.0	71.03±4.41	11.00±0.54	22.25±1.79
Stemming, Union, All words	Ant-Miner	70.97±4.04	**3.0±0.00**	**10.06±2.16**
	C5.0	**76.39±1.01**	13.80±0.73	27.60±1.63

We also did experiments with 429 web pages from the Yahoo web site. Each web page belonged to one of the following three classes: business, tech and entertainment. The results are reported in Table 4.

With respect to accuracy, Ant-Miner obtained the best result in four setups, and C5.0 obtained the best result in the other two setups. However, the differences in accuracy were not significant in any setup, since the accuracy rate intervals overlap. With respect to the simplicity of the discovered rule set, again Ant-Miner discovered a significantly smaller rule set than the rule set discovered by C5.0 in all setups.

Table 4: Comparison between Ant-Miner and C5.0 in Yahoo news web site, all words

Test Setup	Algorithm	Accuracy	No. of rules	Total No. of Terms
WordNet generalization, Title, All words	Ant-Miner	88.00±2.16	**3.6±0.24**	**12.83±2.32**
	C5.0	**89.87±1.88**	18.6±1.20	42.20±6.80
WordNet generalization, Description, All words	Ant-Miner	**86.50±1.99**	**3.0±0.00**	**14.53±2.93**
	C5.0	86.48±1.25	15.8±1.01	34.60±2.54
WordNet generalization, Union, All words	Ant-Miner	**88.15±1.96**	**3.0±0.00**	**13.53±2.62**
	C5.0	86.46±1.24	16.6±0.74	39.80±2.41
Stemming, Title, All words	Ant-Miner	83.54±2.52	**3.4±0.24**	**12.88±2.48**
	C5.0	**86.70±1.10**	16.8±0.66	30.40±1.80
Stemming, Description, All words	Ant-Miner	**87.91±1.75**	**3.4±0.24**	**11.05±2.19**
	C5.0	83.14±3.63	17.4±1.07	29.00±1.22
Stemming, Union, All words	Ant-Miner	**90.01±2.62**	**3.0±0.00**	**12.00±2.33**
	C5.0	89.29±2.09	11.2±0.19	21.40±0.87

5 Discussion and Future Research

This project was the first attempt to apply Ant-Miner to the challenging problem of web page classification, which is plagued by a large number of attributes and the very complex nature of relationships between words. To the best of our knowledge there are just two other projects on using Ant Colony algorithms in web mining, namely the projects described in [6] and [11]. However, our work is very different from those two projects, since our project addresses the classification task, whereas those projects addressed the clustering task (which is very different from classification [2]).

This paper has the following contributions. First, it showed that: (a) Ant-Miner produces accuracies that are at worst comparable to the more established C5.0 algorithm; and (b) Ant-Miner discovers knowledge in a much more compact form than C5.0, facilitating the interpretation of the knowledge by the user. These results agree entirely with previous results comparing Ant-Miner with C4.5 and CN2 in "conventional" data mining (rather than the more challenging text mining scenario), where Ant-Miner also found much simpler rule sets than those algorithms [3], [4].

Secondly, we also investigated the relative effectiveness of different linguistics-based text preprocessing techniques – used as heuristics to reduce the number of attributes – in the performance of Ant-Miner. This is also, to the best of our knowledge,

the first time that an Ant Colony algorithm used WordNet. The results showed that a relatively simple use of WordNet, using the hypernym relationship to generalize words, is often beneficial. However, the errors and misinterpretations it produces when dealing with more complex and longer sentences can sometimes nullify the advantages described. In the scenarios investigated in this paper, WordNet generalisation is most beneficial when the words being generalized occur in a short sentence with a simple meaning, such as in the title field. It is possible that simply stemming the words would be more effective on the more complex sentences, if the number of attributes did not increase so much – overwhelming the Ant Miner algorithm.

Concerning future research, it has been shown that Ant Colony algorithms are good at problems involving continuous learning [12]. It hopefully would be relatively easy to adapt the Ant-Miner algorithm to continuous learning applications as the content available on the web is dynamic by nature. One possibility, for instance, would be to mine data represented in RSS (Really Simple Syndication), which is an XML based web content syndication format. By extending Ant-Miner to continuous learning, the algorithm could be easily used to cope with the dynamic nature of RSS. Another interesting research direction, which could help to achieve a much greater reduction in the number of attributes – while still preserving the most important words from the text – would be to use several other kinds of linguistic relationships available in WordNet.

References

1. I.H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools with Java Implementations*, Morgan Kaufmann Publications, 2000.
2. U.M. Fayyad, G. Piatetsky-Shapiro and P. Smyth. From data mining to knowledge discovery: an overview. In: U.M. Fayyad et al (Eds.) *Advances in Knowledge Discovery and Data Mining*, 1-34. AAAI/MIT, 1996.
3. R.S. Parpinelli, H.S. Lopes and A.A. Freitas. Data Mining with an Ant Colony Optimization Algorithm. *IEEE Trans. on Evolutionary Computation, special issue on Ant Colony algorithms*, 6(4), pp. 321-332, Aug. 2002.
4. R.S. Parpinelli, H.S. Lopes and A.A. Freitas. An Ant Colony Algorithm for Classification Rule Discovery. In: H.A. Abbass, R.A. Sarker, C.S. Newton. (Eds.) *Data Mining: a Heuristic Approach*, pp. 191-208. London: Idea Group Publishing, 2002.
5. S. Chakrabarti *Mining the web: discovering knowledge from hypertext data*. Morgan Kaufmann, 2003.
6. A. Abraham and V. Ramos. Web Usage Mining Using Artificial Ant Colony Clustering and Genetic Programming. *Proc. Congress on Evolut. Comp. (CEC-2003)*. IEEE Press, 2003.
7. M. Cutler, H. Deng, S. S. Maniccam and W. Meng, A New Study Using HTML Structures to Improve Retrieval. *Proc. 11th IEEE Int. Conf. on Tools with AI*, 406-409. IEEE, 1999.
8. C. Fellbaum (Ed.) *WordNet - an electronic lexical database*. MIT, 1998.
9. E. Bonabeau, M. Dorigo and G. Theraulaz. *Swarm Intelligence*: *from natural to artificial systems*. Oxford, 1999.
10. M. Dorigo and L.M. Gambardella, Ant colonies for the traveling salesman problem. *Biosystems* 43, 73-81. 1997.
11. K.M. Hoe, W.K. Lai, T.S.Y. Tai. Homogeneous ants for web document similarity modeling and categorization. *Ant algorithms, LNCS 2463*, 256-261. Springer, 2002.
12. R. Schoonderwoerd, O. Holland, J. Bruten, Ant-like agents for load balancing in telecommunications networks. *HP Labs Technical Report*, HPL-96-76, May 21, 1996.