

Information Retrieval in Network Administration

Ashley George, Adetokunbo Makanju
A. Nur Zincir-Heywood, Evangelos E. Milios
Faculty of Computer Science
Dalhousie University
Halifax, Nova Scotia
B3H 1W5
Canada
{ageorge,makanju,zincir,eem}@cs.dal.ca

Abstract

Network administration is a task that requires experience in relating symptoms of network problems with possible causes and corrective actions. We describe the design of a system and more specifically its information retrieval component, which aims to retrieve articles relevant to a given problem case from a collection of articles describing previously solved problems and their associated solutions. An article is described by a term vector. We present a methodology for defining the vocabulary and preliminary results for assessing the quality of expert-proposed modifications to the vocabulary. We obtain vocabulary-derived document classes from a self-organising map and assess vocabulary quality using the quality of classification into these classes.

1. Introduction

The web hosting and off-site data storage market is one that is fast growing and highly competitive. As the web hosting market has grown, so has the complexity of the systems which hosting providers build and maintain. When downtime occurs, failures must be promptly corrected to preserve high quality service. It is also observed that hosting provider trouble cases tend to recur and follow a typical sequence of process steps. The following list outlines the typical steps in problem resolution.

1. Downtime Occurs
2. Service Alerts Observed
3. Trouble Ticket Raised
4. Ticket Assigned

5. Root Cause of Downtime is Identified
6. Resolution is Planned
7. Resolution is Implemented
8. Notification/Clean Up

Steps 5 and 6 account for the majority of the troubleshooting effort. This results from the time-consuming and knowledge-intensive process of identifying the root cause of a failure. If these two steps could be made faster and easier to perform, there would be a large savings in time and effort for system administrators. The system we describe in this paper is part of an effort to address these steps in the resolution cycle. We further describe specific results pertaining to quantifying proposed expansions to the vocabulary of a taxonomy which is used internally by our system.

We discuss this in the following sections. Section 2 gives an overview of techniques and technology related to web service provision, experience management and information retrieval. Section 3 describes our methodology for evaluating our system taxonomy and recategorising a document collection. Section 4 outlines the experiments in greater detail and describes the results. Section 5 presents discussion and conclusions are given in Section 6.

2. Managing System Administrative Experience

System administrators draw on a variety of resources to acquire information regarding a downtime issue. These resources may include co-workers, trouble tickets, log entries or offsite information sources. For the system administrator, this acquired information can be said

to accumulate as experience knowledge [1]. Given the typically recurrent nature of downtime issues, this experience knowledge, gained over time by system administrators, becomes critical in saving the effort required to execute steps 5 and 6 in the described problem resolution process. Achieving this savings would require a system which associates system events with information relating to the system failure, including corrective administrator actions and internal or off-site information sources. This stored entity would be described as a problem case. New problem cases would either be triggered automatically by system events or manually by an administrator. In the event of a new case, the system would retrieve similar cases in the knowledge base, aiming to present potential corrective actions to the system administrator for reuse or inspiration to improve troubleshooting and decrease downtime costs. The problem in part reduces to an information retrieval (IR) task in which, given a document vector, one must retrieve 'similar' documents from a database. However, this retrieval task presents numerous challenges in the form of indexing, retrieving and ranking heterogeneous data structures.

2.1. Web Hosting Control Panel Applications (WHCPA)

Web hosting requires management of interactions between a variety of actors. This includes system administrators, managers, resellers and customers, resulting in a complex business process. To deal with this complexity, some companies use Web Hosting Control Panel Applications to mediate the business process. A Web Hosting Control Panel Application (WHCPA) is a web-based server administration and management tool designed specifically for the web-hosting environment. The control panel can typically be configured to provide differing levels of access privileges for system administrators, managers and users. The web control panel serves as a unified solution for web service management. From the point of view of a system administrator, a WHCPA must not only assist her with her routine tasks in maintaining her hosting environment and web pages, it should also enable her to resolve downtime incidents within the shortest possible time. Several commercial open source and commercial WHCPA are in use today. In developing the interface for our system, we considered a variety of WHCPAs with regard to the features they offer system administrators for maintenance and recovery tasks. The WHCPAs we considered include cPanel, Ensim, HSphere, Plesk, VHCS and ZPanel [2, 3, 4, 5, 6, 7]. Apart from Plesk, cPanel and ZPanel, none of the WHCPAs provide any features for acquisition, transfer and re-

trieval of knowledge about the problem cases over time, their causes and their possible solutions. Moreover, the solutions provided by Plesk, cPanel and ZPanel are specific to their applications and not intended for organisational knowledge acquisition. Thus we attempt to fill this gap with a combination of techniques, some of which are described in the following section.

2.2. System Framework

Our solution is intended not to perform automatic diagnosis and correction but to assist the administrator in identifying similar faults that have previously occurred and to retrieve information about the steps previously taken to correct the problem. The system's main components include the fault diagnosis engine, the case knowledge base, the problem/solution matching engine, the presentation module, the feedback module and knowledge base editor. The first three components form the core of the system and require information retrieval techniques. All six components are explained in greater detail below.

- *Fault diagnosis engine* Transforms the system alerts generated by the fault detection systems into a format which can be used to represent a new case and retrieve from the knowledge base similar cases which have been resolved in the past.
- *Case knowledge base* At the core of our proposed system is the case knowledge base which represents stored experience derived from previously resolved cases. A case is a heterogeneous structure composed of its problem definition, relevant system alerts, administrative actions taken to resolve the issue, and offsite resources which describe the problem in more detail.
- *Problem/solution matching engine* The problem/solution matching engine retrieves past cases using the new case description provided by the fault diagnosis engine. We generate separate indices for text-based and time-based querying. The similarity of two cases is a composite score based on the distance between their shared subcomponents.
- *Presentation module* The interface through which the administrator can interact with the system.
- *Feedback module* The feedback module helps in acquiring knowledge relevant to the new case from the administrator. The administrator can enter any relevant notes or information for the case which are not directly captured by the system in the course of normal operation.

- *Knowledge base editor* This module can be used by the administrator or knowledge engineer for manual maintenance of the case knowledge base.

2.3. Information Retrieval

Information Retrieval (IR) is typically framed as searching for documents containing desired information within a body of documents. It is also concerned with the representation, storage and organisation of information items [8]. Owing to the enormous volumes of documents and media available online, today IR techniques have become central in finding relevant information quickly and effectively.

The information retrieval process begins when a user formulates a query. In web-based practice the canonical full-text query is approximated by a set of query terms capturing the user’s search intuition. The IR system compares the terms in the user query with the terms characteristic of each document in the knowledge base. Documents whose characteristic terms match most nearly to the user’s query are selected and ranked according to their relevancy for the query. The most relevant documents are then presented to the user in an accessible fashion, normally a ranked list sometimes with a relevance indicator.

2.3.1. Term Frequency and Inverse Document Frequency (TFIDF)

The Term Frequency-Inverse Document Frequency (tf-idf) metric [8] is one of the key metrics used to determine the relevance of a word for a given document. We are interested in a metric for relevance so that we can define a distance between a user query and any document in our database. Let T be the set of all terms occurring in the documents in a database. The term frequency (tf_{ij}) records n_{ij} , the number of times a term t_i appears in a document j , normalised by the number of terms in the document, $|D_j|$, as in Equation (1). As document lengths may vary considerably, it is important to capture the relative frequency of the term within its document. The second component of tf-idf is the inverse document frequency (idf), as in Equation (2), where d_j is the j ’th document, t_i the i ’th term in T , and $|D|$ is the total number of documents. The inverse document frequency characterises the frequency of the term within the entire document set and decreases rapidly for terms which appear in many documents. (idf tends to zero as the ratio in Equation (2) approaches one.) The tf-idf metric is obtained via the product of term frequency and inverse document frequency, as in Equation (3). The captured intuition is that terms which

appear in all or most documents do not serve to distinguish among those documents - therefore, their tf score is reduced proportionately.

$$tf_{ij} = \frac{n_{ij}}{|D_j|} \quad (1)$$

$$idf_i = \log \left(\frac{|D|}{|\{d_j : t_i \in d_j\}|} \right) \quad (2)$$

$$tfidf_{ij} = tf_{ij} \cdot idf_i \quad (3)$$

3. Evaluating Taxonomic Coverage for a Case Base

In our system, the elements of a newly constructed case, which originate from system events and user input, tend to have limited keywords. To mitigate this, we decided to use a taxonomy to define a document representation space. When cases are automatically (or user-) generated from system alerts, the taxonomy provides a source of keywords for the initial case description. Consequently, we are interested in a method for assessing the quality of candidate terms for inclusion in said taxonomy, in conjunction with a corpus of case descriptions. Such a method should support the expert in his or her taxonomy design. The taxonomy itself is a collection of trees representing different conceptual relations within the problem domain of web hosting. Nearer to the root, we have generic concepts; near the leaves, we have specific software or hardware components. The taxonomy helps to relate concepts, products, files and log data sources in our network.

To construct the initial case base with which to exercise our system, we relied on bootstrapping from a repository (kb.swsoft.com) of collected problems and solutions for the Plesk web hosting control panel application contributed for our use by SWSOft.com. As initial cases, they are useful in that they contain a number of keywords which can serve as a basis for information retrieval and are related to the problem domain which we are currently exploring.

3.1. Initialisation and the SWSOft Knowledge Base

We retrieved a total of 1380 articles in HTML format from the SWSOft knowledge base. In the construction of our case base we needed to consider how to perform case matching. Previous cases which in some sense match with the active case must be retrieved and presented to the administrator. We tackle the text aspect of this problem by indexing the keywords from all case fields to the greatest extent possible and comparing the cases

using a vector space model. We constructed an inverted keyword index using tf-idf scores to support keyword-based retrieval and ranking of cases from the knowledge base.

The SWSOft Knowledge Base articles have been manually filed into categories. The categories available from the KB website are mainly conceived as a product tree. The original knowledge base categories are titled to reflect the products offered by the company. These include “Virtuozzo”, “Plesk”, “SiteBuilder”, “Confixx”, “Ensim Pro”, “H-Sphere” and “SiteStudio”. For each of these top-level categories there are a number of sub-categories which are distinguished by product version and operating system. The articles themselves deal with system administration and web hosting topics harvested from the company’s online forum.

3.2. Populating and Assessing the Taxonomy

Our taxonomy is manually organised into a hierarchy of concepts and terms relating to our problem domain: web hosting administration. Initially, we manually constructed a taxonomy consisting of 62 terms relating to web hosting and system administration. We subsequently expanded this taxonomy to 678 terms through a combination of automatic selection and manual refinement. To gain an idea of the representation of our documents in the space of taxonomy terms, we plotted histograms depicting how many articles contain a given number of terms from each taxonomy. The term-document distribution for the small taxonomy can be seen in Figure 1 while the distribution for the large taxonomy is shown in Figure 2. We observe that the small hand-selected taxonomy provides a weak representation of our documents - the majority of documents contain less than 5 terms from the taxonomy. The larger taxonomy provides increased coverage. As the taxonomy grows and as we add new sets of candidate terms, a measure of performance is desired to compare different versions of the taxonomy.

3.3. Classification as a Measure of Taxonomic Quality

As such a system develops, its taxonomy will need to be expanded and updated. Given a candidate set of terms, we would like to be able to automatically assess whether these terms are a useful representation of the documents in our database. We propose using classification performance as an indirect assessment of the quality of the underlying taxonomy as a document representation.

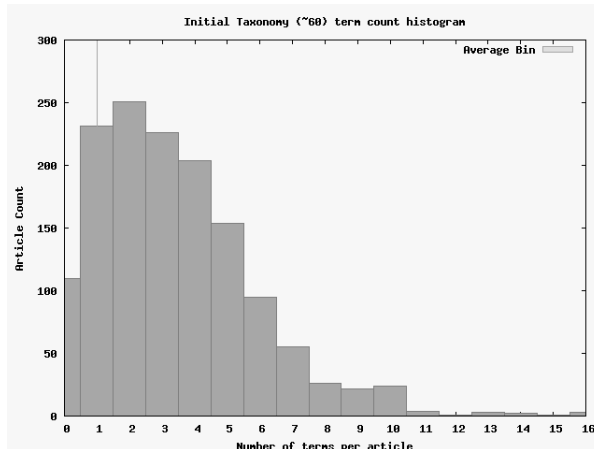


Figure 1: Histogram showing the number of articles containing a given number of terms from the small taxonomy. Most articles only contain a few terms from the small taxonomy.

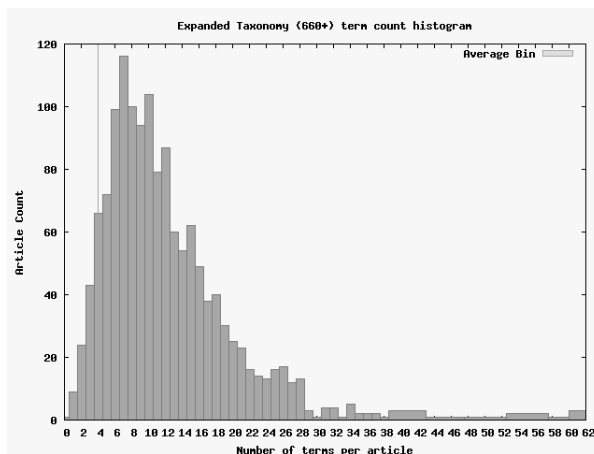


Figure 2: Histogram showing the number of articles containing a given number of terms from the large taxonomy. The larger taxonomy offers a stronger cross-section of the term space.

In our experiment, we use the J48 Java implementation of C4.5 from the WEKA machine learning algorithm collection [9] with a pruning threshold of 0.25. We tried several pruning thresholds (0.25, 0.05, 0.005) across the various datasets we produced but found the difference in performance to be negligible. We characterise our results in the next section using precision, recall and the F-measure, as described in [10]. Let C be the set of unique category labels in our document set. We build a confusion matrix, A , with dimension $|C|$ where A_{ij} records the number of documents in category C_i which were classified into category C_j . Precision, recall and the F-measure for a particular C_i are then defined in terms of A .

$$precision(C_i) = \frac{A_{ii}}{\sum_j A_{ij}} \quad (4)$$

$$recall(C_i) = \frac{A_{ii}}{\sum_j A_{ji}} \quad (5)$$

$$F(C_i) = \frac{2 \cdot precision(C_i) \cdot recall(C_i)}{precision(C_i) + recall(C_i)} \quad (6)$$

The document categories assigned to the SWSOft knowledge base articles largely serve as product and version filters and provide little insight regarding document topics. In our work, we are interested in a categorisation which relates more closely to the categories of problems discussed in the document set. To achieve this, we use self-organising maps for labeling the document set in order to investigate our approach’s performance.

3.4. Self-Organising Maps for Analysing our Corpus

The self-organising map (SOM) [11] is a method for producing a low-dimensional mapping of distance relationships among data points originating in a high-dimensional space. The SOM algorithm is a vector quantization algorithm. Its efficient update scheme and the ability to express topological relationships makes it very convenient for expressing relationships between different groups of documents. In this work, we used a combination of the SOM_PAK package and the SOM Toolbox to perform our SOM-based experiments [12, 13].

Here we intend to use the SOM to analyse our document set. In this work, we select the documents that relate to the Plesk product and search for a plausible clustering based on our knowledge of the domain and visual inspection and grouping of the nodes represented by the most prominent terms in their vectors.

4. Experimental Results

4.1. SOM Clustering

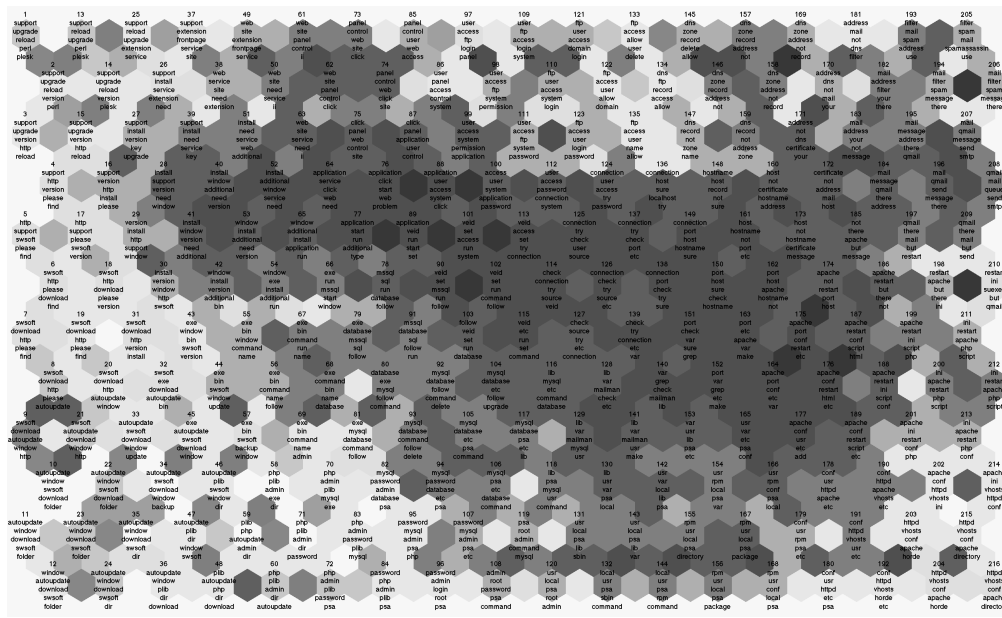
Performing clustering using a self-organising map requires exploration of parameters. The primary parameters include the map dimensions, the number of iterations, the learning rate and the neighbourhood function. Additionally, we require a metric for selecting the vocabulary we would use to represent documents in the SOM. We decided to compute the term frequency variance (as in [14]) across the entire document set. We define $V(t_i)$ using the quantities in Section 2.3. Thus, $|D|$ is the number of documents, and n_{ij} is the frequency of term t_i in document d_j . When we have calculated the variance for each term, we sort the terms by their variance and take a threshold capturing the upper quartile of the terms.

$$V(t_i) = \frac{\sum_{j=1}^{|D|} n_{ij}^2}{|D|} - \frac{1}{|D|} \left(\sum_{j=1}^{|D|} n_{ij} \right)^2 \quad (7)$$

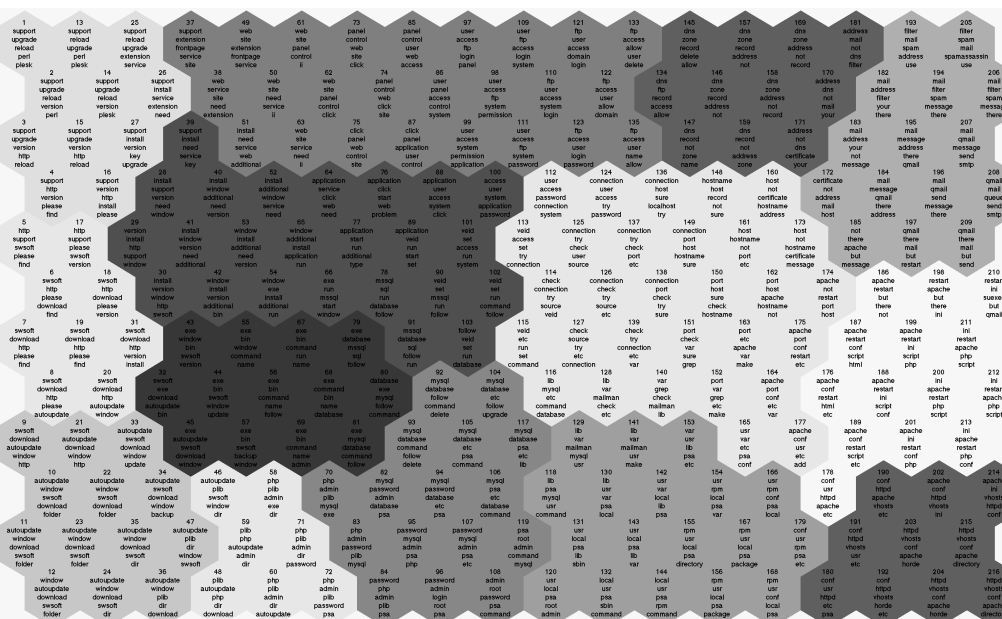
For our experiments, we varied the map dimensions as well as the number of iterations to find a simple solution with a modest error rate. In our initial tests, we varied the map size while holding the number of iterations fixed. We tried maps of sizes 12 by 8, 18 by 12 and 24 by 16. We iterated each for 2.2 million iterations. With random initialisation, the run time varies from tens of minutes (for the 12 by 8 map) to over an hour (for the 24 by 16 map). Visual inspection suggested that from 9 to 12 clusters persisted across runs. The configuration of the map was reproduced at each run, with an occasional inversion or flipping of the mapping due to symmetry. Additionally, it was observed that the map structure tended to stabilise between 500k and 1M iterations. To narrow this value down, we ran further trials at all map sizes for 500k, 1M, 2M and 5M iterations. The results are summarised in Table 1. The underlined entries indicate our choice of the best compromise between map size, overall error and required number of iterations. At 500k iterations, the run time for this map is on the order of minutes.

Once we selected a set of map dimensions and iterations that we considered optimal for our problem and resources, we used the mapping to produce a clustering for the SOM output nodes through a combination of visualising the term feature distribution over the SOM nodes, k-means partitioning and human expert intuition. Figure 3 depicts the similarity map and the per-node top terms which we used to find the strongest clusters as well as the k-means partitioning of the SOM nodes.

Finally, having selected clusters from the SOM mapping, we employed these clusters in labelling our data



(a) U-matrix



(b) k-means over BMUs

Figure 3: a) The U-matrix produced for the 18x12 SOM after 500k iterations presenting all the documents of the Plesk category. Several distinct clusters present along the edges of the map. b) A k-means clustering of the same SOM nodes, used as a baseline for cluster selection. The top 5 terms are printed at each map node.

Table 1: SOM runs and error for all map sizes and number of iterations. The red entries (18x12 for 500k) are not strictly the lowest error values but the 18x12 map has the potential to be much less resource-intensive than its 24x16 counterpart

RunLength	500k	1M	2M	5M	500k	1M	2M	5M	500k	1M	2M	5M
Dimensions	12x8				18x12				24x16			
Quantisation Error	0.2185	0.2184	0.2180	0.2182	0.2122	0.2110	0.2099	0.2092	0.2088	0.2061	0.2043	0.2022
Topological Error	0.0100	0.0112	0.0125	0.0175	0.0075	0.0112	0.0138	0.0112	0.0063	0.0088	0.0088	0.0088

set. We took the k-means clustering as a baseline. Then, based on our visual inspection of the top terms at each SOM node, we adjusted the borders for some clusters. Finally, we labelled each data vector according to the class label for its best matching unit (BMU) in the map.

4.2. Classification

Having labelled our dataset using the results of the SOM clustering, we evaluated the performance of the taxonomic term space using our proposed classification method on the new categories.

We produced three datasets of TF-IDF vectors from the original document data. In order of decreasing number of terms, they are the 'full term space', 'taxonomy' and 'infogain' dataset representations. The 'full term space' representation represents all the documents using those terms which we obtained using the term variance procedure. The 'taxonomy' representation represents all the documents using only those terms which occur in the taxonomy. Finally, the 'infogain' representation was produced by using information gain to rank the feature set with respect to the document categories. We use the top 118 terms (based on a threshold of the ranked terms) to build the document representation. In each representation, the j 'th document d_j is represented as a vector of TF-IDF scores, $d_j = \{tfidf_1, tfidf_2, tfidf_3, \dots, tfidf_n\}$, where n is the number of terms used in that representation.

We finally classify each dataset in a tenfold cross-validation with the "J48" implementation of the C4.5 algorithm from the Java-based WEKA machine learning library. The classification results, as well as the dataset parameters, are shown in Table 2.

5. Discussion

Ideally we would prefer the taxonomic performance on the classification to be as close to the full term space performance as possible. Yet, we observe some degradation in classification power as the term space is reduced to the size of the taxonomy. This implies that the taxonomy may not yet contain the terms it needs

to be a truly discriminative structure for this dataset. But, thanks to the upper limit in performance represented by the vocabulary selected by information gain, we know the taxonomy can become more representative as additional judiciously chosen terms are incorporated into it.

6. Conclusion and Future Work

In this paper, we propose the use of a taxonomy for reducing the term space required to represent documents in a collection and for populating the term vectors for new cases. We show that the taxonomy can be comparable in informative coverage to the full set of terms, particularly as the overlap between informative terms (with respect to the categories) and taxonomic terms increases. We also show that classification can play a dual role as a means for assessing the relevance or the term space coverage of a particular set of terms.

In future work, we will consider means for automatically extending the taxonomy through a combination of user feedback and query parsing. Additionally, we are preparing to evaluate the system in a testbed scenario with expert users in differing contexts, particularly system administrators and help desk employees who hold system administration responsibilities.

Acknowledgements

This research is funded by the PRECARN Small Company Program fund. The authors would also like to acknowledge the staff of Palomino System Innovations Inc., based in Toronto, Ontario and Telecoms Applications Research Alliance (TARA), based in Halifax, Nova Scotia for their support in completing this work.

This work is conducted as part of the Dalhousie NIMS Lab at <http://www.cs.dal.ca/projectx/>.

References

- [1] R. Bergmann, *Experience Management: Foundations, Development Methodology, and Internet-Based Applications*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2002.

Table 2: Classification results w.r.t. the categories generated from the SOM clustering. $|D|$ is the size of the document set; $|F|$ is the number of features for that dataset.

Microaverages						Macroaverages					
	$ D $	$ F $	Precision	Recall	F Measure		$ D $	$ F $	Precision	Recall	F Measure
full term space	800	2242	59%	59%	59%	full term space	800	2242	59%	59%	59%
taxonomy	798	496	52.8%	52.8%	52.8%	taxonomy	798	496	53.8%	53.7%	53.3%
infogain	765	118	59.8%	59.8%	59.8%	infogain	765	118	59.7%	59.4%	58.8%

- [2] cPanel Inc., “The leading control panel - <http://www.cpanel.net/>.” Retrieved from the web., November 2007.
- [3] E. Corporation, “Ensim - the leading management software for unified communications infrastructure - <http://www.ensim.com/>.” Retrieved from the web., November 2007.
- [4] P. S. Corporation, “H-sphere - a multi-server web hosting solution - <http://www.psoft.net/hsphere-overview.html>.” Retrieved from the web., November 2007.
- [5] SWSOft, “Plesk - control panel software for hosting <http://www.swsoft.com/plesk/>.” Retrieved from the web., November 2007.
- [6] V. Team, “Virtual hosting control system - <http://www.vhcs.net/>.” Retrieved from the web., November 2007.
- [7] Z. Project, “Zpanel project - <http://www.thezpanel.com/>.” Retrieved from the web., November 2007.
- [8] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. Addison Wesley Inc., 1999.
- [9] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*. San Francisco: Morgan Kaufmann, 2005.
- [10] F. Sebastiani, “Machine learning in automated text categorization,” *ACM Computing Surveys*, vol. 34, no. 1, pp. 1–47, 2002.
- [11] T. Kohonen, *Self-Organizing Maps*, vol. 30 of *Springer Series in Information Sciences*. Berlin, Heidelberg: Springer, 1995. (Second Extended Edition 1997).
- [12] Kohonen, Hynninen, Kangas, and Laaksonen, “SOM_PAK The Self-Organizing Map Program Package,” tech. rep., Helsinki University of Technology, 1995.
- [13] J. Vesanto, J. Himberg, E. Alhoniemi, and J. Parhankangas, “SOM Toolbox for Matlab,” tech. rep., Helsinki University of Technology, 2000.
- [14] J. Kogan, C. Nicholas, and V. Volkovich, “Text mining with information-theoretic clustering,” *Computing in Science and Engg.*, vol. 5, no. 6, pp. 52–59, 2003.