IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Hershey PA 17033-1117, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

#ITP4355

N³ Query-By-Structure Approach

(Neural Network Net Query-by-Structure Approach)

Michael Johnson, Farshad Fotouhi, and Sorin Draghici
Department of Computer Science, Wayne State University, Detroit, MI 48202
(313) 577 - 3107 • {mlj, fotouhi, sod}@cs.wayne.edu

ABSTRACT

This paper describes our continuing research into a query-by-structure approach. The proposed approach extends existing work, which has focused on querying the Web by content, by allowing the user to request documents containing not only specific content information, but also to specify that documents be of a certain type. We have developed two prototype systems that utilize a supervised Hamming Neural Network and an unsupervised Competitive Neural Network, respectively. Both systems are designed to capture and utilize structure information as well as content during a distributed query on the Web. The primary objective was to test the feasibility of utilizing neural networks to improve document query-by-structure performance. Initial testing has shown promising result when comparing to straight keyword searches.

1. INTRODUCTION

The task of accessing and processing information in today's business world is a necessity. However, as more information is recorded, being able to sift through it all to capture what is relevant is becoming increasingly more difficult. To add to the burden, the process of information retrieval now has to deal with the distributed nature of today's information repositories. For example, consider the task of searching for information on the World Wide Web. A typical keyword search using a standard search engine typically results in thousands of purportedly *relevant* web documents. Savvy search engines allow users to perform quasi database queries by combining keywords with logical boolean operators to pair down the number of *matched* web pages. Unfortunately, results from these searches can still yield a significant number of pages, many of which will likely be irrelevant. Clearly, a means to improve these types of results would be of great benefit.

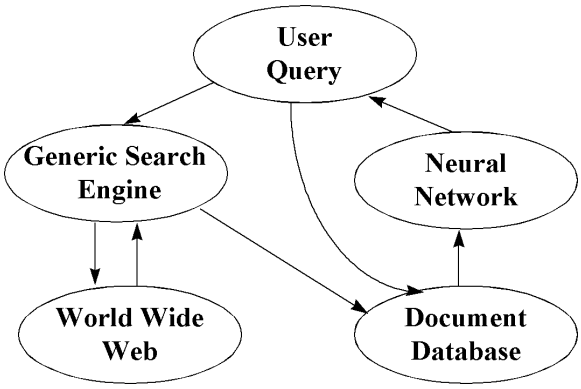
Information retrieval from homogeneous data sources typically involves querying based on content. However, the query process utilizes very little, if any, information about the underlying structure of the data. Yet, the way in which information is structured typically plays a significant role in the importance of that information. Considering our web search example again, most search engines flag a document as relevant if it simply contains the keywords specified in a query. But, the context in which the keywords are used is usually not considered. However, a web page is a semi-structured document, and web designers utilize structure to emphasize particularly important aspects of their documents. Hence, by creating a query process that captures how information is structured, and subsequently, performing queries based on some structure criteria, query performance can be greatly enhanced.

The objective of this paper is to describe our continuing research into the query-by-structure approach. We have constructed two prototype systems designed to capture and utilize structure information as well as content during a distributed query. Both systems employ neural networks to organize information based on relevancy of both content and structure. Our primary goal at this point is to test the feasibility of using neural networks to improve the query process. It should be noted that the ultimate objective of our research is not to create another Web search engine. However, due to the vast number of easily accessible semi-structured documents available, we have utilized the web as a resource for constructing our prototype systems.

2. N³ QUERY-BY-STRUCTURE

We studied the concept of query by content, link and structure with our CLaP system[4], which provided promising results when comparing keyword searches to searches utilizing structure information. However, the query process was quite cumbersome to the user, as they were required to specify details about where keywords could appear. Obviously, to become a viable querying alternative, this issue needed to be resolved. Hence, a key design consideration during the development of the N³ query-by-structure prototypes was to eliminate these complex user queries.

Fig 2.1 N³ Query-by-Structure Process



Although similar, the systems developed utilize different types of neural networks: a *supervised* Hamming Network and an *unsupervised* Competitive Network. The query process for both systems involves five steps (as shown in Fig. 2.1). First, the user initiates a keyword web search. A generic search engine then identifies relevant documents based on the provided keywords. However, instead of returning the results directly to the user, the results of the query become the input to the neural network. Both systems then employ their respective neural network to organize the information based on the relevancy of both the content and structure of the document. Lastly, the neural network outputs the results of the user query. The following sections breakdown each of these steps, highlighting the differences between the two systems.

2.1 User Query

The user initiates a query by entering a standard keyword search and the maximum number of URLs they wish to receive in their results. Currently the systems support a maximum of 50 URLs. This is due to time constraints since each URL represents a web document that must be retrieved and parsed by the system. Addi-

This paper appears in the book, *Managing Information Technology in a Global Economy*, the proceedings of the Inforrmation Resour es Management Association International Conference. Edited by Mehdi Khosrow-Pour.

tionally, the supervised Hamming network system requires that the user select a document *type*.

The type field is used by the neural network to categorize document structure. This will be discussed in greater detail later, but the basic idea is that documents of similar type should have a similar structure. For example, most research papers start with a *centered* title, contain headings for an abstract, introduction, conclusion, references, etc.. Likewise, documents of different types should be *structured* differently. Hence, an on-line news article should look significantly different than a research paper. The unsupervised Competitive network also utilizes the notion of document type, but does not require the user to provide this information. For our initial testing, we selected four document types: resumes, conference papers, news articles, and e-commerce web documents. Obviously there could be an infinite number of different document types, along with variations within those types, but our objective was not to classify as many different types of documents as possible, but rather to simply test the feasibility of using neural networks to improve query-by-structure performance.

2.2 Generic Search Engine Query

Upon issuance of the query, the search is initially performed using the keywords provided by the user. Since any commercial search engine can perform this operation, the Alta Vista[1] search engine was selected. This choice was made because Alta Vista indexes more documents that most other search engines. The results of the search are the URLs and descriptions of the documents deemed to be *relevant* by the search engine. The remaining modules of the system utilize these results, along with the document type specified by the user.

2.3 World Wide Web Query

Although the results obtained from the generic search engine are actual documents on the web, the only web query performed in the previous section was to contact the search engine, which in turn simply performed a query on a local database. As a result, in order to capture document structure information, it is necessary to actually retrieve each of the web documents specified in the results. Hence, this module takes a list of URLs, and one at a time, retrieves the source code of each of the web documents. These documents are then parsed in order to analyze the document structure.

In order to test the feasibility of the neural network approach, a simplified approach to analyzing structure was adopted. For each web document, a feature vector was created based on whether specific words appeared between specific HTML tags in the document source code. The words selected were words deemed relevant to one of the document types specified in the user query. The following is a partial list of words used in the initial testing: abstract, activities, bibliography, business, by, buy, cart, conclusion, education, experience, introduction, mall, mastercard, news, objective, price, purchase, related, references, resume, shop, sports, store, visa, vitae, work. Some of the HTML tags utilized were: <b>, , <emp>, <u>, <h#>, <font> and <i>. Hence, the feature vector contained an entry for each word specified above, and if that word appeared between one of the specified HTML tags, a 1 was placed in the feature vector for that word, and a 0 otherwise. This feature vector was then used as the input vector to the neural networks.

2.4 Document Database

During the initial design of the system, an existing database/search engine package called ht://dig[6] was installed. However, it became apparent that even when utilizing this application, it was still necessary to perform an additional retrieval of each of the web documents in order to capture document structure. As a result, each web document was being retrieved twice during each user query. Due to query time constraints, it was necessary to temporarily remove this feature from the system.

2.5 Neural Networks

A significant portion of the work involved with this project was in the design of the neural networks and their inputs. Although other types of neural networks will be considered in the future, we chose to construct the two systems using the supervised Hamming network and the unsupervised Competitive network because of their straightforward and contrasting approaches. The implementation of the neural networks was done using the Perl programming language. Although Perl is not the most efficient language, it is well suited for web document retrieval and parsing. The implementation and performance details are discussed in the following sections.

2.5.1 Hamming Neural Network

A Hamming neural network is a supervised learning network that consists of two layers. The first layer (Feedforward Layer) performs a correlation between the input vector and the prototype vectors. The second layer (Recurrent Layer) performs a competition to determine which of the prototype vectors is closest to the input vector[5].

The network is considered supervised because the prototype vectors are determined in advance and are designed to match the expected input vectors.

2.5.1.1 Layer 1 Implementation

The first step was to initialize the prototype vectors. Four prototype vectors were utilized, with each one representing a different document type. Each vector contains Boolean values which represent whether each of the words specified earlier is expected to appear within a given structure context inside that specific document type. After initialization, the Feedforward Layer performs the correlation between the prototype vectors and each of the input vectors provided to the neural network. The number of input vectors is equivalent to the number of URLs requested by the user in the initial web query.

2.5.1.2 Layer 2 Implementation

The second layer of the neural network performs the competition between the vectors produced as output from layer 1. This layer utilizes a weight matrix that has diagonal elements having a value of 1, and off-diagonal elements having a small negative value. For our implementation, a value of -.25 (-1/# of prototype vectors) was selected. The output from the second layer of the neural network is a set of vectors representing which prototype vector most closely matches each input vector. However, for this system, the objective was also to determine which of the input vectors most closely matches the selected document type prototype vector. Fortunately, the output vector with the appropriate largest value matches the prototype vector better than any other vector. As a result, sorting the values in each of the output vectors, from highest to lowest, provids us with a new ranking of the web documents. After sorting, the system simply outputs the URLs and descriptions captured in the generic search engine query but in the order specified by the neural network.

2.5.2 Unsupervised Competitive Neural Network

The unsupervised competitive neural network replaces the recurrent layer in the Hamming network with a competition transfer function. Like the Hamming network, this competition layer still identifies the largest value in the vectors produced by the *feedforward* network. However, instead of applying the recurrence equation on the output vectors, the competition transfer function simply produces a vector of all 0s except for the neuron that contains the largest value, which it sets to 1. In addition, the objective of the unsupervised network is significantly different than that of a supervised network. The goal of the supervised network is to match the input vectors to the predetermined prototype vectors. However, in the unsupervised network, the prototype vectors are not used in the training. Instead, the input vectors are used to configure the weight matrix using the Kohonen learning rule[5].

2.5.2.1 Competitive Network Implementation

The implementation of the competitive network was fairly straightforward. First, a weight matrix for the prototype vectors is randomly generated, and an input vector for one of the web documents is randomly selected. Then, the competitive layer in the neural network finds the largest value in the vectors produced by the feedforward layer. After the winner is selected, the weight matrix is update using the Kohonen rule. This process is repeated for a fixed number of iterations.

3. EXPERIMENTAL PERFORMANCE AND RESULTS

The N^3 query-by-structure systems have been fully implemented and are available for testing on the web. The system that utilizes the Hamming network can be accessed at the following URL: http://cca.munet.edu/~mjohnson/nn/hamming.html. The system that utilizes the Competitive network can be accessed at: http://cca.munet.edu/~mjohnson/nn/compete.html.

3.1 Hamming Network Performance

Preliminary tests of the Hamming neural network system were promising. As an example, consider a query that retrieves only 10 URLs of a *resume* document type using the keywords: "Computer Science". At the time this document was created, the Alta Vista results for this query placed a career services web site first and a Computer Science program's student resume *repository* second. After converting these results to input vectors and filtering them through the neural network, the career services web site ended up ranked 7th and the resume repository ended up last at 10th. Additionally, between these two documents were a very poorly structured resume and a web page that contained a note stating that if you were interested in that individual's resume, you could contact them via e-mail. At the top of the rankings were web documents containing actual resumes.

Other experiments were performed with similar results to the example above. Table 3.1 compares the results of the system to the Alta Vista search results for these tests. A brute force approach of viewing each document to determine if it was of the desired type was necessary. Occasionally, the browser *timed out* when the query took too long. Although this is purely a browser related issue, future enhancements to our system need to account for this problem.

Table 3.1 Hamming Network vs. Alta Vista Result Comparison

Keywords	Document Type	Pages Retrieved	Hamming Network	Alta Vista
Biology	Resume	50	9 out of top 10 were resumes	5 out of top 10 were resumes
Secretary	Resume	30	4 out of top 6 were resumes	2 out of top 6 were resumes
Political Science	Research Paper	30	4 out of top 6 were research papers	0 out of top 6 were research papers
War	Research Paper	30	3 out of top 6 were research papers	1 out of top 6 were research papers
Microsoft	News Article	30	3 out of top 6 contained news articles	1 out of top 6 contained news articles
College Football	News Article	40	5 out of top 8 contained news articles	2 out of top 8 contained news articles
Textbooks	E-Commerce Site	40	7 out of top 8 are e-commerce sites	3 out of top 8 are e-commerce sites
Flowers	E-Commerce Site	30	5 out of top 6 are e-commerce sites	2 out of top 6 are e-commerce sites

3.2 Competitive Network Performance

The performance of the unsupervised competitive neural network was not as consistent as that of the supervised Hamming network. Preliminary testing resulted in most documents being placed in only one or two classes (currently the system can only cluster to four classes). One modification that was recently made was to use a variable learning rate that decreases in value during the iterative process. Although it is too early to make a judgement, early tests look promising.

There are several contributing factors to the sporadic results. First, unlike the supervised neural network that creates the weight matrix based on the prototype vectors, this network randomly generates its weight matrix. As a result, each test yields different results. Secondly, the number of times the network iterates also plays a role in determining the results. This value can only be determined through extensive testing. Currently, the number of iterations performed is equal to three times the number of input vectors. Another variable factor in this network is the choice of the learning rate. This value ranges between 0 and 1, with higher values resulting in fast but unstable learning, and lower values resulting in slow but stable learning. Once again, the optimal value for the learning rate can only be determined through extensive testing. Finally, a third issue is with the creation of the input vectors. As discussed earlier, these vectors are generated based on document structure. Hence, creating input vectors that properly distinguish between different types of documents is paramount to the performance of the network.

4. RELATED WORK

The focus of the research presented in this paper has been twofold. The first goal was to design a process that captures structure information as well as content during a distributed query. Secondly, neural networks were utilized to organize the information based on relevancy of not only the content but the structure of the document as well. Below we describe related research in these two areas.

4.1 Web Querying

We have identified a number of research projects focusing on web querying. However, it is our opinion that most focus very little attention to document structure. For instance, in [7], the authors present a query language called WebSQL designed specifically for querying the web. The system designed to apply WebSQL navigates the web starting from a known URL, potentially travers-

ing multiple child links. However, other than the text contained within the anchor tags, no structure information is utilized in the queries. WebLog[9] is another language designed to directly query the web. The WebLog system allows the user to incorporate some forms of presentation knowledge into the query through a structure called a rel-infon. However, utilizing rel-infons requires the user to have extensive knowledge of declarative logic concepts in order to write even the simplest of queries.

The creators of the Google[3] search engine have shown that the use of some aspects of document structure during the web page ranking process proved to be beneficial. We have identified two systems that do use a significant amount of document structure information. The first is the ClaP system we designed, which incorporates content, link and structure information to directly query the web. Another system is W3QL[10]. This system maintains the results of user queries in a database. In addition to document content, specific node and link information is cataloged. Additional presentation information such as HTML *form* elements is also maintained. However, a significant drawback to these types of systems is that they typically require the user to have knowledge about document location and structure.

4.2 Neural Network Techniques

There has been a significant amount of research in an attempt to improve web searches using machine learning techniques. WebWatcher[2] interactively assists users in locating desired information on a specific site. The system tracks the user actions and utilizes machine learning methods to acquire knowledge about user goals, web pages users visit, and success or failure of the search. Rankings are based solely on keywords. Syskill & Webert[8] is a system that was tested with six different machine learning algorithms including Perceptron and Backprop neural networks. This system also requires user interaction by requesting that the user rate pages on a three-point scale. Individual user profiles are then created by analyzing information on each page. Based on the user profile, the system suggests other links that might be of interest to the user. The drawbacks to these systems are that they require users to actively participate in the ranking process.

5. CONCLUSION

As the amount of information being collected increases, new mechanisms need to be developed to sift through this information to capture what is relevant. The CLaP system led us to believe that utilizing structure as well as content can significantly improve query performance. However, CLaP required that the user specify document structure as part of their query. As a result, when performing even the simplest of queries, an inordinate amount of information was required from the user. To eliminate these complex queries, we chose to test the feasibility of a different approach that utilized neural networks within the query process.

Although the N^3 query-by-structure systems are in the early developmental stages, initial experimental results look promising. Clearly, by utilizing a neural network, the burden of identifying structure within the document can become the responsibility of the system rather than the user. However, much more work must be done to make the systems usable.

6. FUTURE WORK

We are continuing to look at ways to improve the current system as well as incorporating some of the features previously mentioned that have not yet been implemented. One of our first priorities is to construct the database module to enable us to capture the web documents for future user queries. Including this module will not only make this a more viable system but will also make it significantly easier to test our results due to the rapid response time. We are also looking into ways to alleviate the burden from the user of selecting a document type altogether. Currently, the system matches documents based on a predetermined notion of what a specific document should look like. However, users of the system may have entirely different opinions about how a specific type of document should be structured. It is likely that a better way for a user to specify a document type is *by example*. In other words, instead of specifying a document type, the user can provide a limited number of web document URLs that they feel resemble the types of documents they are seeking. These corresponding web documents can then be parsed by the system and applied to a neural network to determine their common features. It is from these features that the prototype vectors can be formed and then subsequently utilized by the currently implemented neural networks. Implementing the system in this fashion will also solve the problem of there being an almost infinite number of different types of web documents.

8. REFERENCES

[1] Alta Vista. *http://www.altavista.com*

[2] R. Armstrong, D. Freitag, T. Joachims, T. Mitchell, "WebWatcher: A learning apprentice for the World Wide Web", *Workshop on Information Gathering for Heterogeneous Distributed Environments*, AAAI Spring Symposium Series, Stanford, CA, March 1995,http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-6/web-agent/www/ijcai97.ps

[3] S. Brin, L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine", Proceedings of the 7th International World Wide Web Conference, Brisbane, Australia, April 1998 http://www7.scu.edu.au/programme/fullpapers/1921/com1921.htm

[4] F. Fotouhi, W. Grosky, M. Johnson, "CLaP: A System to Query the Web using Content, Link, and Presentation Information", Proceedings of the Fourteenth International Symposium on Computer and Information Sciences, Kusadasi, Turkey, October 1999

[5] M. Hagan, H. Demuth, M. Beale, "Neural Network Design", *PWS Publishing Company*, 1st Edition, 1996

[6] The ht://dig Group. *http://www.htdig.org*

[7] A.O. Mendelzon, G.A. Mihaila, T. Milo, "Querying the World Wide Web," *Proceedings of the International Conference on Parallel and Distributed Information Systems (PDIS'96)*, Miami, Florida, 1996.

[8] M. Pazzani, J. Muramatsu, D. Billsus, "*Syskill & Webert: Identifying interesting web sites*", Proceedings of the 13th National Conference on Artificial Intelligence, 1996, http://www.ics.uci.edu/~pazzani/RTF/AAAI.html

[9] L.V.S. Lakshmanan, F. Sadri, I.N. Subramanian, "A Declarative Language for Querying and Restructuring the Web", *Proceedings of the Sixth International Workshop on Reasearch Issues in Data Engineering*, 1996.

[10] D.Konopnicki, O.Shmueli, "W3QS: A Query System for the World Wide Web", *Proceedings of the 21th International Conference on Very Large Data Bases*, Zurich, Switzerland, 1995.