# Choosing Document Structure Weights

ANDREW TROTMAN
*Department of Computer Science, University of Otago,*
*PO Box 56, Dunedin, New Zealand*
*andrew@cs.otago.ac.nz*

## Abstract

Existing ranking schemes assume all term occurrences in a given document are of equal influence. Intuitively, terms occurring in some places should have a greater influence than those elsewhere. An occurrence in an abstract may be more important than an occurrence in the body text. Although this observation is not new, there remains the issue of finding good weights for each structure.

Vector space, probability, and Okapi BM25 ranking are extended to include structure weighting. Weights are then selected for the TREC WSJ collection using a genetic algorithm. The learned weights are then tested on an evaluation set of queries. Structure weighted vector space inner product and structure weighted probabilistic retrieval show an about 5% improvement in mean average precision over their unstructured counterparts. Structure weighted BM25 shows nearly no improvement. Analysis suggests BM25 cannot be improved using structure weighting.

Keywords: Structured Information Retrieval, Genetic Algorithms, Vector Space Model, Probability Model.

## 1. Introduction

Not all parts of a document are created equal. For academic papers, authors are asked to write a few words that concisely describe their work. This is the title. They are asked to write a few paragraphs that outline their work, the abstract. They are asked to write a few pages that precisely describe the work, the body text. Finally they are asked to summarize the work with a conclusion. These are very different and unequal parts of the same document.

Vector space (Salton, Wong, & Yang, 1975) and probabilistic (Robertson & Sparck-Jones, 1976) IR systems rank documents without regard to term location. A term found in an abstract is of equal importance to the same term in the body text of the same document. The document structure is ignored even though authors write documents with structure. A document may even be originated with explicit structure in a mark-up language like XML (Bray, Paoli, & Sperberg-McQueen, 1988), structure discarded when indexing. There is a mismatch: documents have structure, yet the IR system ignores it.

Document structure should be utilized in ranking. For example, knowledge in an abstract is denser than elsewhere, fulfilling the principle of summarization. This principle can be applied to ranking. Terms should receive a weighting based on where in the document they occur. In other words, if a term occurs in an abstract it should be weighted as such.

Fuller *et al.* (1993) first suggested structure weighting in 1993. Since then the probability model has been extended to include structure weighting (Wolff, Flörke, & Cremers, 2000), IR query languages have been extended to allow the user the option of choosing the weights (Fuhr & Großjohann, 2001), and index structures have been proposed (Schlieder & Meuss, 2002).

One remaining problem is the choice of the weights. If users don't specify the weights themselves, the IR system should default to a good set of cross corpus weights. But how can those weights be selected? In this investigation genetic algorithms (GA) are used. Each document structure is represented in a chromosome in a GA learning simulation. Selective pressure is then applied to maximise mean average precision.

Experiments with the TREC (Harman, 1993) Wall Street Journal (WSJ) collection using structured variants of inner product, probability, and Okapi BM25 (Robertson, Walker, Beaulieu, Gatford, & Payne, 1995) are conducted. Results demonstrate an about 5% improvement in mean average precision for vector space and probability, while no improvement is shown for BM25.

## 2. Approach

Just as someone encountering this document will glance at the title and abstract to determine its relevance, a computer can score a document by examining its structures.

First, the document structures are identified. Conveniently, for XML these structures are already explicit. This mark-up is not the DTD, but the document tree. The tree is used as it is also unreasonable to assume all occurrences of a tag are equal. If a <title> tag occurs in <bibliography> it is perhaps of less interest than when occurring as the document's title.

Second, structured ranking is defined. The score of a document with respect to a term usually uses term frequency, *tf*, as an integral component. It assumes all occurrences of a term within a single document are of equal importance. Term frequency is replaced by a structure weighted term frequency.

Third, weights are selected. Many approaches have been suggested including using humans with domain knowledge (Rapela, 2001), trial and error (Wilkinson, 1994), and simulated annealing (Boyan, Freitag, & Joachims, 1996). In this investigation genetic algorithms are used. The ubiquity of the TREC Wall Street Journal collection makes this dataset ideal for testing the genetic algorithm approach. The INEX collection (Fuhr, Gövert, Kazai, & Lalmas, 2002) was not used as it is not distributed with judgements.

Finally, the learned weights are evaluated for effectiveness against a set of queries not used in training. A *t*-test is applied to the results to provide evidence of significance.

### 2.1. Related Work

#### *2.1.1. Genetic Algorithms*

Machine learning and genetic algorithms are not new to information retrieval (Chen, 1995; Savoy & Vrajitoru, 1996). Gordon (1988) suggests representing a posting as a chromosome and using genetic algorithms to select good indexes. Yang *et al.* (1992) suggest using GAs with user feedback to choose weights for search terms in a query. Morgan and Kilgour (1996) suggest an intermediary between the user and IR system employing GAs to choose search terms from a thesaurus and dictionary. Vrajitoru (1998), Boughanem *et al.* (2002), and Horng and Yeh (2000) examine GAs for information retrieval and suggest new crossover and mutation operators. Vrajitoru (2000) examines the effect of population size on learning ability, concluding that a large population size is important.

Despite the successes, little use has been made of genetic algorithms for *ad hoc* queries.

Harman (1993) observes different IR systems returning substantially different results, yet maintaining approximately equal performance. Building on this, Bartell *et al.* (1994) suggest combining the output of different ranking functions to improve performance. Pathak *et al.* (2000) use a genetic algorithm to choose weights for such a combination.

#### *2.1.2. Identification of the Problem*

Research into structured document indexing (Meuss & Strohmaier, 1999; Shin, Jang, & Jin, 1998; Thom, Zobel, & Grima, 1995), querying (Chinenyanga & Kushmerick, 2001; Fuhr & Großjohann, 2001) and ranking (Kotsakis, 2002; Schlieder & Meuss, 2000, 2002) has suggested using document structures for document-centric ranking, but not how to choose good weights.

Schlieder and Meuss (2002) suggest structure weights should be specified in the query. This approach is embraced in the query language XIRQL (Fuhr & Großjohann, 2000, 2001), and the graphical query language of Baeza-Yates *et al.* (1998). Although users should be given the option of choosing weights, choosing the weights has proven difficult. Rapela's (2001) investigation shows a decrease in precision at almost all points of recall when a human subject is asked to choose structure weights. This task may prove less difficult if a good set of weights can be presented for adjustment.

Boyan *et al.* (1996) examine tag weighting for HTML. A hand selected set of tags and tag weights are chosen alongside other configurable parameters. Users present queries, receive results, and view documents. A log of documents viewed against each query is generated. Weights are then successfully learned using simulated annealing. However, Boyan *et al.* outline presentation bias as a problem to be overcome; users select documents with high ranks regardless of how badly they fit their information need. A good set of starting weights is needed.

*2.1.3. Choosing Weights*
Wilkinson (1994) uses trial and error to choose structure weights. All structures except one are given a weight of 1 while the one's weight is adjusted to 2. The performance is measured as the one varies from structure to structure. Finally, one of three possible weights (0.5, 1, or 2) is assigned to each structure. Wilkinson demonstrates a performance improvement.

Rapela (2001) successfully used a gradient-based optimisation function to learn weights for HTML tag and non-tag heuristics.

Kim *et al.* (Kim & Zhang, 2000, 2001; Kim, Kim, Eom, & Zhang, 2000) suggest HTML documents can be represented by just the contents of a few tags. New documents are derived from old by stopping all term occurrences outside those tags. Ranking is not based on the existence of a term in a document, but the existence of a term in a tag in document. The rank for a given document is the product of two scores, one for the term and one for the tag. Using Kim's technique, terms outside the chosen tags cannot be found. For those in the chosen tags, a document containing a search term twice in <TITLE> and once in <B> has an identical score to twice in <B> and once in <TITLE>.

*2.1.4. XML and SGML*
Constructions for HTML cannot necessarily be applied to XML and SGML. HTML has no hierarchical structuring whereas XML does. Should an XML <title> tag occur in <bibliography> it is unreasonable to assume weighting equal to the same tag used as the title of the document, something perhaps reasonable for HTML <title> tags. When structure-weighing XML, the hierarchical structure is more important than the tag names. Further, in XML the tag names may not be known in advance.

*2.1.5. Document Fragmentation*
Dividing a document into parts using tag boundaries can be likened to document fragmentation and passage retrieval. These have been examined in the literature extensively (Callan, 1994; Kaszkiel & Zobel, 1997, 2001; Wilkinson & Zobel, 1994). Documents are divided into fragments and each indexed independently. The weight of a document with respect to a query is taken as the weight of the highest scoring document fragment with respect to that query. Fragmentation does not preclude document structure weighting; the two could be used in conjunction. Structured weighted fragmentation is not examined herein.

## 3. Methods

### 3.1. Structured Information Retrieval

An inverted file index of a corpus consists of two parts, the dictionary and the postings. The dictionary stores a list of all the unique terms in the corpus and a pointer to a posting. A posting is a vector $\{<d_1, f_1>, <d_2, f_2>, \ldots <d_n, f_n>\}$ where $d$ is a document number and $f$ is the number of occurrences of the term in $d$. Postings are sorted by increasing $d$ as a consequence of sequential indexing.

The hierarchical structure of an XML document represents a tree. Thom *et al.* (1995) suggest giving each unique tag an ordinal identifier and storing document paths directly in the indexes as paths. Meuss and Strohmaier (1999) also store paths directly in the indexes, however encoded as a bitstring. Kotsakis (2002) and Trotman (2003) prefer to build a corpus tree, but differ in how they store the postings.

The corpus tree is constructed during single pass indexing. Each time a previously unseen path through a document is encountered, it is added to the tree and given a unique ordinal identifier. Eventually, every path through every document is represented. The resulting tree is unlikely to be the exact structure of any given document, but the complete structure of every document is represented.

Figure 1 presents three XML documents, and the corpus tree. The tree contains all paths through all documents, but does not match any single document. Each node in the corpus tree has a unique identifier allotted in order of node creation. Nodes are not analogous to DTD elements, but to an instance of an element from a DTD. A term occurring in <name> in <place>, will be treated differently from <name> in <person>. Should a term occur in <name> in <place> it will not be considered to lie in <name>, or <place>, but rather in <name> in <place>.

```
<doc>
  <docid>1</docid>
  <place>
    <name>University of Otago</name>
  </place>
  <cntry>New Zealand</cntry>
</doc>

<doc>
  <docid>2</docid>
  <place>
    <name>Dunedin</name>
    <rank>expert</rank>
  </place>
</doc>

<doc>
  <docid>3</docid>
  <person>
    <name>Smith</name>
    <rank>Intermediate</rank>
  </person>
</doc>
```

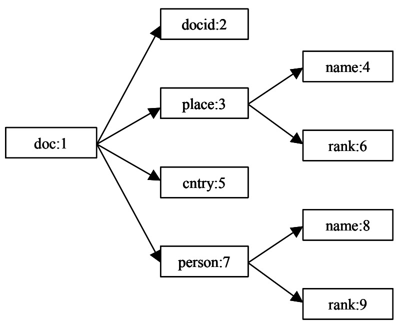


**Figure 1: Three XML documents and the corpus tree. The name of each node is an XML tag name. The path to each node is a path through a document. Numbers at each node are the position identifiers allotted at creation time.**

The postings are now modified to include pointers into the corpus tree. Each pair <*d, f*> becomes one or more triples <*d, p, f*> where *d* is the document number, *p* is the position in the corpus tree, and *f* is the number of occurrences of the given term in the given position of the given document.

These structured postings can be converted back into an unstructured posting by collecting the occurrence counts for each document regardless of position. Term weighting and ranking could be applied as if no change to the index had occurred, however this investigation takes advantage of structured indexes by altering the ranking equations to use the position member of the triple.

**3.2. Structured Ranking**

Fuller *et al.* (1993) suggest applying simple scalar weighting to each DTD element, but go no further. Wolff *et al.* (Wolff, Flörke, & Cremers, 1999; Wolff *et al.*, 2000) build on this suggestion and extend the probability model to include structure weighting. In this investigation weights are applied to each node in the corpus tree, not to each DTD element, or to the query vector.

*3.2.1. Vector Space Model*

Vector space similarity is often calculated using the inner product

$$R_{dq} = \sum_{i=1}^{t} w_{id} \times w_{iq} \tag{1}$$

where $R_{dq}$ is the relevance of document *d* with respect to query *q*, $w_{iq}$ is the weight of term *i* in *q*,

$$w_{iq} = tf_{iq} \times IDF_i \tag{2}$$

and $w_{id}$ is the weight of term *i* in document *d*.

$$w_{id} = tf_{id} \times IDF_i \tag{3}$$

where $tf_{id}$ is the term frequency of term *i* in document *d*, and likewise for $tf_{iq}$ in the query. $IDF_i$ is the inverse document frequency of term *i*

$$IDF_i = \log_2 \frac{N+1}{n_i} \tag{4}$$

where $N$ is the number of documents in the corpus and $n_i$ is the number of documents containing term $i$ in the corpus.

To extend the vector space model to support structured ranking, occurrences within each document structure must be included so

$$tf_{id} = \sum_{p=1}^{n} tf_{ipd} \tag{5}$$

where $tf_{ipd}$ is the number of occurrences of term $t$ in position $p$ of document $d$. A simple scale factor $C_p$ for each document structure can now be applied

$$tf'_{id} = \sum_{p=1}^{n} \left(C_p \times tf_{ipd}\right) \tag{6}$$

$tf'_{id}$ is then substituted for $tf_{id}$ in equation (3) giving

$$w_{id} = tf'_{id} \times IDF_i \tag{7}$$

*3.2.2. Probability Model*

$R_{dq}$, the weight of a document with respect to a query is given by substitution of $tf'_{id}$ for $tf_{id}$

$$R_{dq} = \sum_{i \in q} \left(C + IDF_i\right) \times \left( L + (1 - L) \times \frac{tf'_{id}}{m_d} \right) \tag{8}$$

where $C = 1.0$, $L = 0.3$ and $m_d$ is the term frequency of the most frequent term in document $d$.

*3.2.3. Okapi BM25*

For Okapi BM25 ranking, $tf'$ can be substituted directly into the ranking equation giving

$$R_{dq} = \sum_{i=1}^{t} w_i \times \frac{(k_1 + 1) \times tf'_{id}}{K + tf'_{id}} \times \frac{(k_3 + 1) \times tf_{iq}}{k_3 + tf_{iq}} \tag{9}$$

where

$$w_i = \log \frac{N - n_i + 0.5}{n_i + 0.5} \tag{10}$$

and

$$K = k_1 \times \left( (1 - b) + \frac{b \times T_d}{T_{av}} \right) \tag{11}$$

$k_1 = 1.2$, $k_3 = 7$, $b = 0.75$ and $T_{av}$ is the average document length measured in terms (the same unit of measure as $T_d$). $tf_q$ is the number of times term $i$ occurs in query $q$.

*3.2.4. Discussion*

The weighted $tf'$ can be substituted into any ranking function that ordinarily uses $tf$. These new equations allow weighting for term occurrences at different nodes in the corpus tree. If any $C_p$ is 0, node $p$ will have no ranking influence. If $C_p$ are all 1, unstructured ranking results. The relative importance of document structures is reflected in the $C_p$ weights.

To mark terms in titles, abstracts, or figure captions as "of more interest", suitable $C_p$ values are selected for these nodes. XIRQL, and other query languages, give the user the power to choose weights; however if weights are not chosen defaults must be assigned. A genetic algorithm can be used to determine good default weights. Equally, by algorithmically determining good weights it is possible to identify the most interesting document structures.

## 3.3. Genetic Algorithms

It is impractical to iteratively test all possible $C_p$ weight combinations, however, the search for good weights can be directed using a genetic algorithm.

The TREC collection is an ideal training set. The collection is distributed with a series of queries called topics and a series of judgments. A judgment is a binary decision as to whether or not a topic is relevant to a given document.

Genetic algorithms (Holland, 1975) direct a search towards a problem solution using the Darwinian notion of survival of the fittest. The robustness of genetic algorithms has been demonstrated in many domains (Goldberg, 1989) including information retrieval (Gordon, 1988; Vrajitoru, 1998). Section 2.1 discusses the already diverse use of genetic algorithms in information retrieval.

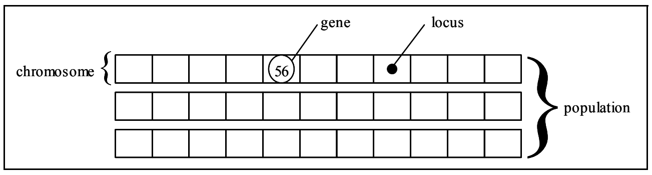


**Figure 2: Chromosomes, loci, genes and populations are the constituent parts of the genetic algorithm. They are, in essence, an array, a member, a value and a list.**

Using genetic algorithm terminology, a population is a collection of individuals. An individual is a chromosomal encoding of a potential problem solution. A chromosome is an array of fixed length; each position in a chromosome is a locus. Each locus takes a value, that value being a gene. Figure 2 graphically depicts this representation.

An initial population is created with completely randomised genes. The fitness of each individual is calculated algorithmically and selective pressure applied. The weak perish and the strong survive. A new population, the next generation, is created through reproduction, mutation, and crossover. The process is then applied iteratively.

To use genetic algorithms to choose structure weights it must be possible to represent potential solutions as individuals, to calculate fitness, and to reproduce, mutate, and crossover.

Document structure weights are numeric. Each node in the corpus tree has a unique identifier. So a chromosomal encoding is chosen with one locus for each node in the corpus tree, the gene at locus $p$ being the weight $C_p$.

Mean average precision is chosen as the fitness $f(n)$ of individual $n$. Average precision for a single query is the sum of precisions for each found and relevant document, divided by the number of relevant documents. Mean average precision is the mean of such scores over a set of queries.

Although individual fitness is a quantitative measure of an individual's adaptation to the environment, it says nothing about the fitness of an individual with respect to the population, that is, the individual's fitness proportion $fp(n)$.

$$fp(n) = \frac{f(n) - F + \varepsilon}{\sum_{m=1}^{G}(f(m) - F + \varepsilon)} \tag{12}$$

where $G$ is the number of individuals in the generation, $F$ is the minimum observed $f()$ in the generation, and ε is included to prevent division by zero. This linear dynamic scaled fitness proportion (Grefenstette, 1986) is preferred over the unscaled counterpart as it is general purpose (Khuri, Bäck, & Heitkötter, 1994).

The sum of fitness proportions for a single generation is 1. A generation can therefore be represented as a number line in the range [0…1]. Each individual takes a part of the line equal to its fitness proportion. In the process of fitness proportionate selection, a random number between 0 and 1 is chosen and the individual at that point on the line is selected.

Individuals reproduce with probability equal to their fitness proportion. An individual is chosen from the current generation using fitness proportionate selection, then carried over into the next generation.

For mutation, an individual is selected using fitness proportionate selection. A locus is chosen at random and the gene there is replaced with a random gene. The mutated individual is carried over into the next generation.

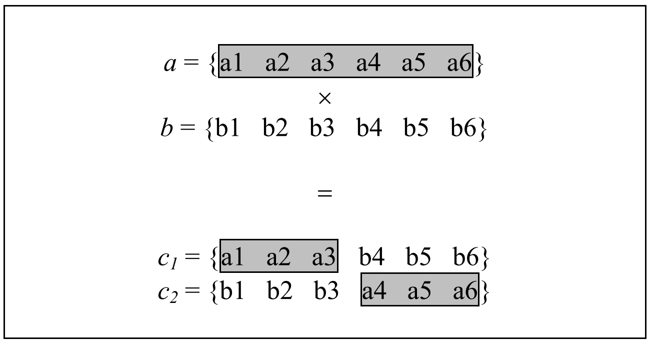


**Figure 3: Crossover between chromosome *a* and *b* at locus (g=3) results in two children $c_1$ and $c_2$.**

Using fitness proportionate selection, two individuals (*a* and *b*) are chosen for single point crossover. A locus *g* is chosen at random and two new individuals are created. One with the genes from *a* up to *g* and *b* thereafter, the other from *b* to *g* and *a* thereafter. Figure 3 illustrates this process. Both new individuals are carried over into the new generation.

Reproduction, mutation, and crossover occur with configurable probabilities (which sum to one). Once a new generation has been created, the genetic process is repeated iteratively until a problem solution is found.

## 4. Experimental Methods

Experiments to choose good document structure weights were conducted using a genetic algorithm. Mean average precision was used as fitness. The fitness of the found weights was then evaluated against ranking without structures.

Average precision was computed as the sum of precisions for each found and relevant document, divided by the number of relevant documents. Mean average precision as the mean of such scores over a set of queries.

### 4.1. Test Collection

The TREC Wall Street Journal files (1987-1992) on the TREC collection disks 1 and 2 were indexed using the structured information retrieval system described above. The corpus tree was generated during indexing.

TREC topics 151-200 were used for training. Topics 101-150 were used for evaluation. Queries were built by extracting the description field then stopping commonly used words. Topics 121, 175, 178, and 181 were discarded, each having fewer than 5 judgments. Terms were not stemmed.

### 4.2. Genetic Parameters

Each chromosome had 20 loci, 1 locus for each node in the corpus tree. Genes took a value in the range [0..1]. An initial population of 50 individuals was chosen at random. Mutation and crossover rates were set at 0.2. Reproduction rate was 0.6. Alternative population sizes, mutation rates, and crossover rates were not investigated, but could affect the evolution rate.

Mean average precision was used for fitness. Learning continued for 25 generations. The initial population was seeded with an individual with all weights equal to 1; the equivalent of unweighted ranking.

Purely by chance, the fittest individual in a generation might not be selected to carry over into the next generation. Should this happen the fittest-so-far would perish. To prevent this, the fittest individual in each generation automatically reproduces; learning was elitist (De Jong, 1975).

**4.3. Experimental Process**
Experiments were run to find weights for weighted vector space (W-VSM), weighted probability (W-PM), and weighted BM25 (W-BM25).

An initial random generation was created. Then for each generation, each individual was presented to the retrieval system one at a time. Values for $C_p$ were taken from the chromosome. Mean average precision over the training set was calculated and stored with each individual.

At the end of each generation, with the fittest individual in that generation: (i) mean average precision was recorded; (ii) average precision was calculated and recorded for each query in the evaluation set; (iii) mean average precision for the evaluation set was calculated and recorded. Subsequently, a new generation was created from the old using just results from the training set. Each experiment was conducted 180 times to eliminate any chance of error.

Average precision scores for each query using unweighted vector space (VSM), probability (PM) and BM25 (BM25) were calculated, recorded and compared to their structure weighted counterparts. A one-tailed *t*-test was applied to provide evidence of a significant improvement.

Finally, the training and evaluation set were swapped and the experiment rerun for validation.

## 5. Results

### 5.1. Learning Results
Figure 4 shows how the mean average precision changed over time. Thick lines represent the training set whereas thin lines represent the evaluation set. The MAP scores plotted are the mean over 180 runs. Changes in the training set are reflected as changes in the evaluation set. Dips in the evaluation MAP during learning are a consequence of over-fitting – seen in BM25 learning.

The training samples used in these experiments are characterised by having "correct" target documents, but not "correct" average precisions. Fitness can be computed, but error cannot (without knowing the target, deviation from the target cannot be computed). Traditional cross-validation (Moody, 1994; Weiss & Kulikowski, 1991) techniques therefore cannot be used. A validation estimate comes through an estimation of best performance. To make this estimate, the GA is assumed to find a reasonable approximation to the optimal solution, the training and evaluation sets are swapped, and the experiment re-run.

Table 1 presents three sets of results – those from unweighted retrieval, weighted retrieval, and from the validation experiment. For weighted and validation, the results presented are those from the best learning run – the run with the highest MAP in the queries used in training.

Examining probabilistic retrieval: a set of weights is learned using the training set. The MAP of the evaluation set using these weights is measured as 0.1787. The validation experiment is run. The optimal MAP for the evaluation set is estimated as 0.1823. Finally the error is computed as the distance from the measured to the optimal, 0.0036.

Figure 5 presents precision / recall graphs showing using mean average precision at 11 points of recall in the evaluation set. Improvements can be seen for vector space and probability models, but not for BM25.

### 5.2. Information Retrieval Results
In an online system, structure weights would be selected through training before the system comes online. Evaluation would occur continuously as users present queries to the system. The training and

evaluation sets used in the experiments herein mirror this. A set of weights was learned using the training set and evaluated using the evaluation set.

A comparison of how such a system might perform is presented in figure 6 and figure 7. In figure 6 the average precision (computed at each document) for each evaluation query is compared. In figure 7 the change in average precision is presented. For users queries a significant improvement is seen.

Weighted vector space model shows an improvement in average precision for 61% of the queries in the evaluation set. Overall a 4.72% improvement is observed in mean average precision. A one tailed *t*-test gave a P value of 0.0014; the weighted vector space model improvement is significant at the 1% level. The best improvement seen in the evaluation set was 4.8% (59.2% of queries), however this was not in the best run.

Weighted probability model shows an improvement in average precision for 75.5% of the queries in the evaluation set. Overall a 6.67% improvement is observed in mean average precision. A one tailed *t*-test gave a P value of 0.0033; the weighted probability model improvement is significant at the 1% level. The best improvement seen in the evaluation set was 8.4% (77.6% of queries), however this was not in the best run.

Weighted BM25 shows an improvement in average precision for 37.8% of the queries in the evaluation set. Overall a 0.33% degradation is observed in mean average precision. A one tailed *t*-test gave a P value of 0.0522; the weighted BM25 improvement is not significant at the 5% level. The best improvement seen in the evaluation set was 0.7% (65.3% of queries), however this was not in the best run.

On unseen queries, vector space and probability models show significant improvement when weighting is used. The weights used for these comparisons is shown in table 2.

| **Model** | **Vector Space** | | **Probabilistic** | | **BM25** | |
|---|---|---|---|---|---|---|
| | **Training** | **Evaluation** | **Training** | **Evaluation** | **Training** | **Evaluation** |
| **Unweighted** | 0.1508 | 0.1657 | 0.1890 | 0.1675 | 0.2553 | 0.2289 |
| | | | | | | |
| **Weighted** | 0.1565 | 0.1735 | 0.1986 | 0.1787 | 0.2561 | 0.2281 |
| **Improvement** | 3.80% | 4.72% | 5.09% | 6.67% | 0.34% | -0.33% |
| | | | | | | |
| **Validation** | 0.1554 | 0.1746 | 0.1974 | 0.1823 | 0.2539 | 0.2323 |
| **Improvement** | 3.07% | 5.36% | 4.42% | 8.82% | -0.51% | 1.52% |
| | | | | | | |
| **Mean** | 0.1559 | 0.1740 | 0.1980 | 0.1805 | 0.2550 | 0.2302 |
| **Mean Improvement** | 3.44% | 5.04% | 4.76% | 7.74% | -0.09% | 0.60% |
| | | | | | | |
| **Error** | 0.70% | 0.62% | 0.64% | 1.98% | 0.85% | 1.82% |

**Table 1: Comparison before and after learning of each ranking algorithms using structure weighted and unweighted retrieval. Training on the evaluation set gives an approximation to the optimal performance of the evaluation set.**

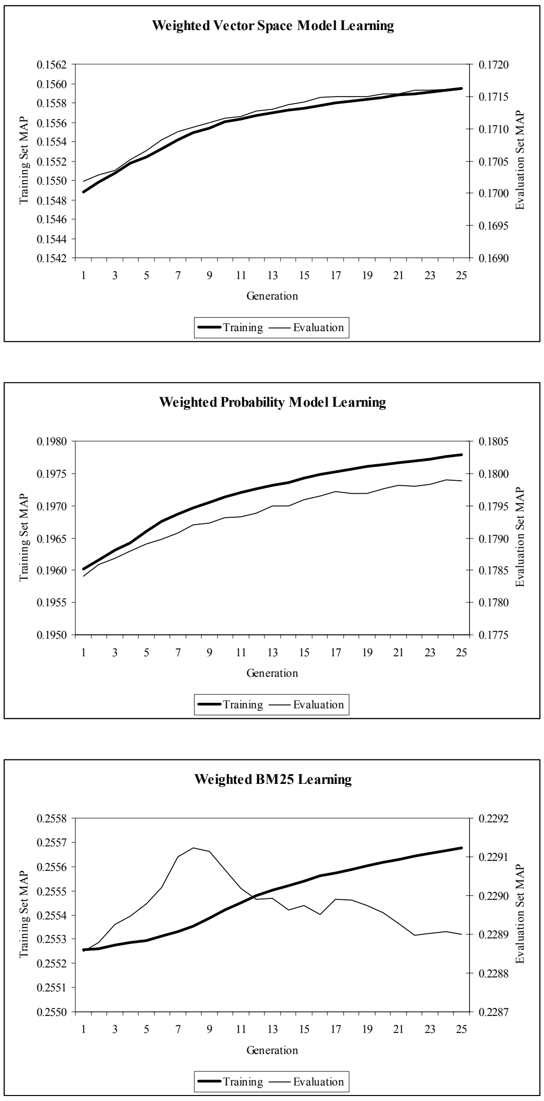


**Figure 4: Mean average precision for the training set and the evaluation set during training (shown as the mean over 180 runs).**

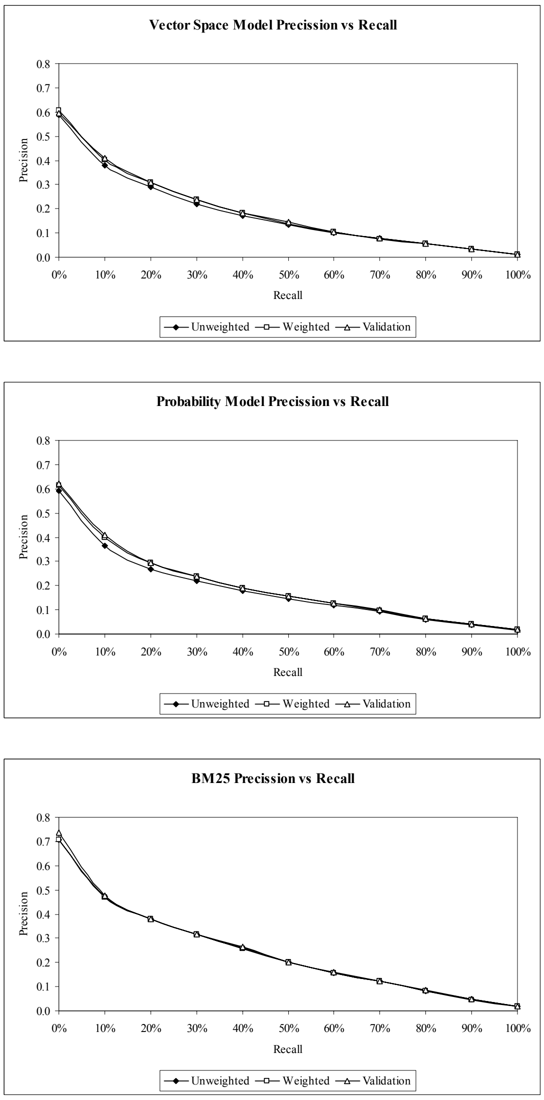
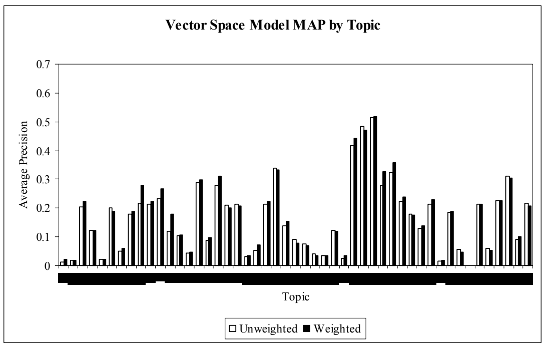
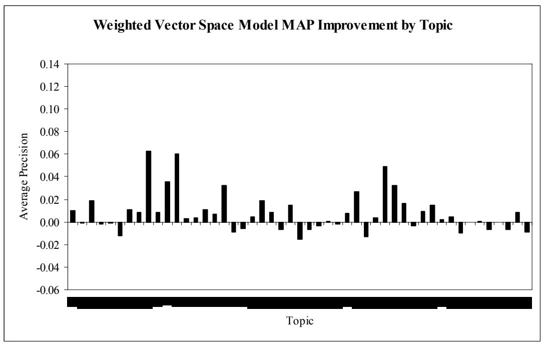
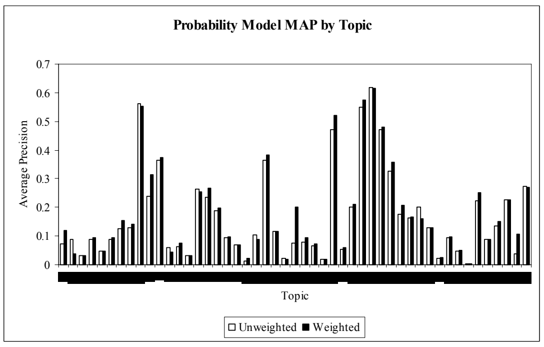
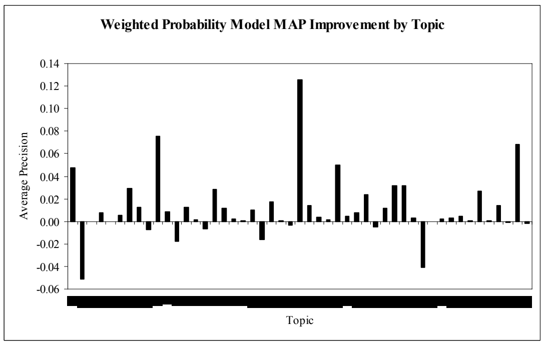


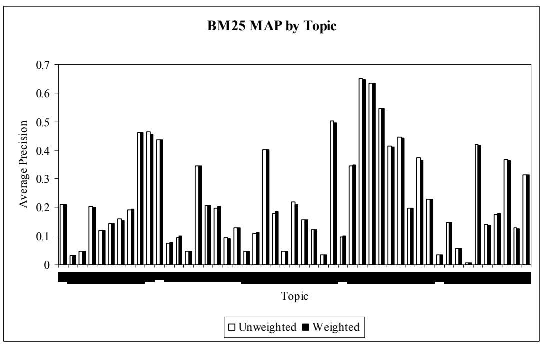
**Figure 5: Precision / Recall graphs comparing structure weighted retrieval with unweighted retrieval for the evaluation set.**











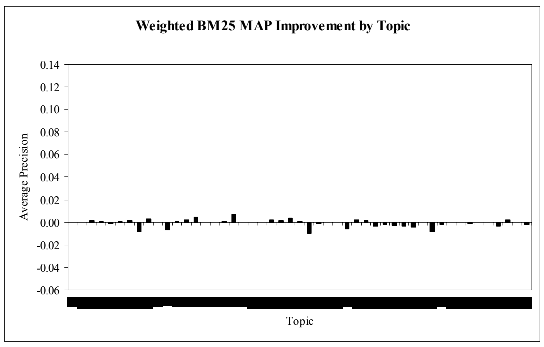


**Figure 6: Average precision of each topic using structure weighed and unweighted retrieval**

**Figure 7: Improvement in average precision for structure weighted retrieval over unweighted retrieval.**

| Field | Structure | W- VSM Weights | W-PM Weight s | W-M25 Weight s | Trainin g Set | Evaluatio n Set |
|---|---|---|---|---|---|---|
| Document | DOC | - | - | - | - | |
| Document Number | DOCNO | - | - | - | - | - |
| **Headline** | **HL** | **0.4873** | **0.5827** | **1.0000** | **47** | **49** |
| Story date | DD | - | - | - | - | - |
| Story source | SO | - | - | - | - | - |
| Category terms | IN | 0.4503 | 0.0536 | 0.8492 | 41 | 40 |
| Location | DATELIN E | 0.1519 | 0.1770 | 0.0685 | 46 | 48 |
| **Text** | **TEXT** | **0.1659** | **0.9953** | **0.9597** | **47** | **49** |
| Companies | CO | 0.7749 | 0.0558 | 0.2692 | 21 | 16 |
| Author | AUTHOR | 0.5545 | 0.2453 | 0.2200 | 3 | 5 |
| Government agencies | G | - | - | - | 1 | - |
| Identification code | AN | - | - | - | - | - |
| Government agencies | GV | 0.9733 | 0.0494 | 1.0000 | 15 | 25 |
| Document ID | DOCID | - | - | - | - | - |
| Date | DATE | - | - | - | - | - |
| **Leading paragraph** | **LP** | **0.3001** | **0.6589** | **1.0000** | **47** | **49** |
| Indexing codes | MS | 0.1945 | 0.5507 | 0.2936 | 1 | 8 |
| Region code | RE | 0.1206 | 0.7067 | 1.0000 | 35 | 40 |
| Indexing codes | NS | 0.3274 | 0.6123 | 1.0000 | 31 | 29 |
| Indexing codes | ST | 0.7019 | 0.7900 | 1.0000 | 7 | 4 |

**Table 2: Weights learned at each node of the corpus tree. Weights are only shown for structures influential in the evaluation set. Other structures are not present in any query in the evaluation set. The number of queries utilising the given structure is shown in the last two columns.**

## 6. Discussion

A method of indexing and searching structured data allowing structure weighting is presented and a genetic algorithm is used to learn weights. Weighted ranking using vector space and probabilistic retrieval showed significant improvements over unweighted retrieval.

### 6.1. Ranking

#### *6.1.1. Vector Space Model*

The vector space model used for these experiments computes the document weight as the inner product of two vectors, the query vector and the document vector. Structure weighting does not influence the query vector; it only affects the document vector. The influence due to the document vector, $w_{id}$, is given by

$$w_{id} = tf'_{id} \times IDF_i \tag{13}$$

where $tf'_{id}$ directly scales $IDF_i$ to give the term influence. The structure scalar values $C_p$ directly affect the influence of each term and as such reflect the relative importance of each document structure. Structure weights in the range [0..1] were chosen for consistency with the other methods.

#### *6.1.2. Probability Model*

The ranking equation used with the probability model scales term frequencies by $m_d$, the term frequency of the most frequently occurring term in the document. This term is often a stop-word such as “and” or “the”. Even if such terms are removed from the indexes, the remaining most frequent terms are likely to be noise by Zipf’s law. Terms relevant to the document / query pair are the so-called middle terms.

$R_{dq}$, the weight of a document, $d$, with respect to query $q$ is given by

$$R_{dq} = \sum_{i \in q} d_i \times e_i \tag{14}$$

where

$$d_i = C + IDF_i \tag{15}$$

and

$$e_i = L + (1 - L) \times \frac{tf'_{id}}{m_d} \tag{16}$$

When ranking without structure weighting, $tf'_{id}$ is at most $m_d$, so $e_i$ is always in the range [0..1]. Term influence for term $i$ in probability ranking adds to the document / query weight some linear proportion of $d_i$, weighted by $L$. As $tf'_{id}$ becomes very large, the influence of $L$ becomes very small and $e_i$ tends to $tf'_{id}$. To this end, to preserve the influence of $L$, it is necessary to keep structure weights strictly in the range [0..1].

*6.1.3. BM25 Ranking*

BM25 showed no significant improvement when using structure weighting. Examining the BM25 ranking equation suggests why. $R_{dq}$, the weight of a document, $d$, with respect to query, $q$, is given by

$$R_{dq} = \sum_{i=1}^{t} a_i \times b_i \times c_i \tag{17}$$

where

$$a_i = \log \frac{N - n_i + 0.5}{n_i + 0.5} \tag{18}$$

the log odds of the term occurrence in the collection,

$$b_i = \frac{(k_1 + 1) \times tf'_{id}}{K + tf'_{id}} \tag{19}$$

the influence of the term with respect to the document, and

$$c_i = \frac{(k_3 + 1) \times tf_{iq}}{k_3 + tf_{iq}} \tag{20}$$

the influence of the term with respect to the query.

Weighted ranking does not affect the occurrence of the term in the collection, or in the query. It only affects term occurrences with respect to the document. Neither $a_i$ nor $c_i$ will be effected. Only $b_i$ is affected by structure weighted ranking.

Expanding $b_i$ using the constants given above,

$$b_i = \frac{2.2 \times tf'_{id}}{1.2 \times \left(0.25 + \frac{0.75 \times T_d}{T_{av}}\right) + tf'_{id}} \tag{21}$$

assuming all documents lengths, $T_d$, are constant, and therefore equal to $T_{av}$, the average document length,

$$b_i = \frac{2.2 \times tf'_{id}}{1.2 + tf'_{id}} \tag{22}$$

which tends from 1 when $tf'_{id}$ equals 1, to 2.2 as $tf'_{id}$ tends to infinity as shown in figure 8.

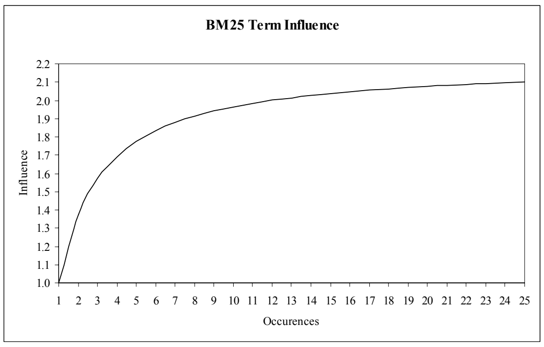


**Figure 8: Ranking influence of terms using BM25.**

The term influence expression, $b_i$, in the BM25 ranking equation is highly discriminatory when term occurrences are small, but not so when term occurrences are large. Structure weights greater than 1 create virtual term occurrences increasing the number of terms in a given document. These additional occurrences add an ever-decreasing influence. Eventually, the additional influence becomes smaller than the gap between document weights and document ordering is preserved regardless of the term count. Structure weights between 0 and 1 were used to push discrimination into the steeper parts of figure 8.

Even by keeping structure weights low, it has not proven possible to improve BM25 with structure ranking. This is likely to be because BM25 is already tuned for this data. The tuning parameter $k_1$ is already very good and has a similar effect to structure weighting.

*6.1.4. Comparison*

For structures HL, LP and TEXT, the weights differ between weighted vector space model and weighted probability model (Table 2). Using W-VSM the weights are ordered as expected, HL, LP, TEXT. Using W-PM they are ordered in the reverse order.

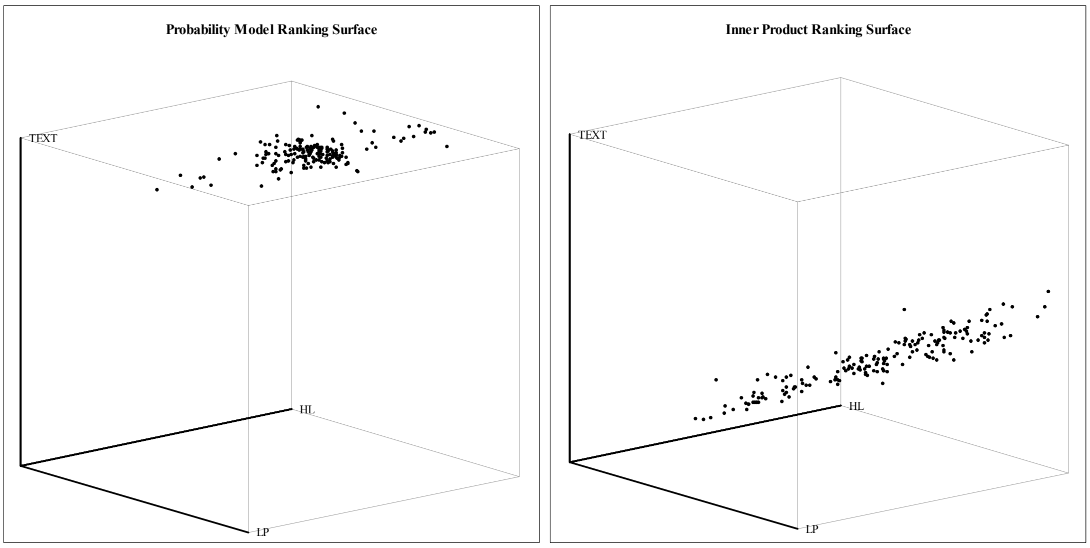


**Figure 9: Scatter plot for the three structures TEXT, LP and HL. Each point represents a single 25-generation run. The learned values for these three structures appear to lie in a plane. Different planes are seen for probabilistic and vector space models.**

Examining only these three structures, each run can be considered to return a coordinate in three-dimensional space; each structure representing an axis. Plotted in figure 9 are such points for each experimental run. From the figure it is clear the points lie in a plane, but a different plane depending on ranking function. This plane represents a three-dimensional projection of the multi-dimensional ranking surface. The genetic algorithms can be considered as a search for this surface.

To determine if weights are interchangeable between ranking function, the probability weights were tried with vector space ranking and vice versa. This experiment resulted in a decrease in mean average precision of 3.9% in vector space ranking and 7.4% in probability ranking. The structure weights are dependant on ranking function.

Each ranking function behaves differently, and subsequently different structure weights are learned. This dependence of structure weights on ranking function may contribute to why human subjects are unable to choose good values. Although the user should be given the option of choosing weights, default weights tuned to the particular ranking function should be provided. Default weights should be derived from the collection and ranking function, not chosen *ad hoc*.

In some topics a performance decrease was observed. In these cases, search terms influential in the meaning of the query only occurred in low-weighted structures. When this happens, less important query terms become more influential in ranking. Should the most important query terms be found only in structures with low weights, the *meaning* of the query can be lost.

### 6.2. Efficiency
The experiments necessary to learn good structure weights need only be run once for each document collection. The weights will not change if the experiment is run a second time. The cost in CPU cycles, and real time, is therefore not an important issue. However, the cost remains low.

The training set contained 47 queries. The initial population contained 50 individuals. In the first generation 2,350 searches were necessary. After the first generation, searches were necessary only for new individuals. Mean average precision is already known for individuals that reproduce (60%) so a search is unnecessary. A search is only necessary for those individuals that come about through crossover or mutation. In all the number of queries required for 25 generations is about 24,910. On a 1.6GHz Pentium 4, these 25 generations can be completed in less than an hour. Over a weekend, 50 such runs can easily be completed on a single CPU.

## 7. Conclusions
Some parts of a document are more interesting than others. When flipping through a journal a reader will stop on seeing something interesting. These interesting structures should be weighted as such during ranking.

Experiments were conducted with the TREC WSJ collection. Indexing was with the presented structured information retrieval system. A single tree representing the tagging structure of the collection was built. Structure weights, assigned to each node in the tree, were found using a genetic algorithm. Finally, the results were evaluated demonstrating significant improvements when using vector space model or probability model. No significant improvement was observed for BM25.

Improvements gained through learning are mirrored in evaluation. Not only was there a gain in mean average precision, but also in most queries. The one-off task of determining document weights using a genetic algorithm has resulted in significant improvements when using vector space and probability models.

## 8. Acknowledgements
This work was supported by University of Otago Research Grant (UORG) funding.

## 9. References
Baeza-Yates, R., Navarro, G., & Vegas, J. (1998). A model and a visual query language for structured text. In *Proceedings of the String Processing and Information Retrieval: A South American Symposium*, (pp. 7-13).

Bartell, B. T., Cottrell, G. W., & Belew, R. K. (1994). Automatic combination of multiple ranked retrieval systems. In *Proceedings of the 17th ACM SIGIR Conference on Information Retrieval*, (pp. 173-181).

Boughanem, M., Chrisment, C., & Tamine, L. (2002). On using genetic algorithms for multimodal relevance optimization in information retrieval. *Journal of the American Society for Information Science and Technology,* 53(11), 934-942.

Boyan, J., Freitag, D., & Joachims, T. (1996). A machine learning architecture for optimizing web search engines. In *Proceedings of the AAAI Workshop on Internet-Based Information Systems*.

Bray, T., Paoli, J., & Sperberg-McQueen, C. (1988). Extensible markup language (XML) 1.0, W3C recommendation. Available: http://www.w3.org/TR/REC-xml.

Callan, J. P. (1994). Passage-level evidence in document retrieval. In *Proceedings of the 17th ACM SIGIR Conference on Information Retrieval*, (pp. 302-310).

Chen, H. (1995). Machine learning for information retrieval: Neural networks, symbolic learning, and genetic algorithms. *Journal of the American Society for Information Science,* 46(3), 194-216.

Chinenyanga, T. T., & Kushmerick, N. (2001). Expressive retrieval from XML documents. In *Proceedings of the 24th ACM SIGIR Conference on Information Retrieval*, (pp. 163-171).

De Jong, K. A. (1975). *An analysis of the behavior of a class of genetic adaptive systems.* Unpublished Ph.D., University of Michigan.

Fuhr, N., Gövert, N., Kazai, G., & Lalmas, M. (2002). Inex: Initiative for the evaluation of XML retrieval. In *Proceedings of the ACM SIGIR 2000 Workshop on XML and Information Retrieval*.

Fuhr, N., & Großjohann, K. (2000). XIRQL an extension of XQL for information retrieval. In *Proceedings of the ACM SIGIR 2000 Workshop on XML and Information Retrieval.*

Fuhr, N., & Großjohann, K. (2001). XIRQL: A query language for information retrieval in XML documents. In *Proceedings of the 24th ACM SIGIR Conference on Information Retrieval*, (pp. 172-180).

Fuller, M., Mackie, E., Sacks-Davis, R., & Wilkinson, R. (1993). Structured answers for a large structured document collection. In *Proceedings of the 16th ACM SIGIR Conference on Information Retrieval*, (pp. 204-213).

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning.*: Addison-Wesley.

Gordon, M. (1988). Probabilistic and genetic algorithms in document retrieval. *Communications of the ACM,* 31(10), 1208-1218.

Grefenstette, J. J. (1986). Optimization of control parameters for genetic algorithms. *IEEE Transactions. on Systems, Man, and Cybernetics,* 16(1), 122-128.

Harman, D. (1993). Overview of the first TREC conference. In *Proceedings of the 16th ACM SIGIR Conference on Information Retrieval*, (pp. 36-47).

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: University of Michigan Press.

Horng, J.-T., & Yeh, C.-C. (2000). Applying genetic algorithms to query optimization in document retrieval. *Information Processing & Management,* 36(5), 737-759.

Kaszkiel, M., & Zobel, J. (1997). Passage retrieval revisited. In *Proceedings of the 20th ACM SIGIR Conference on Information Retrieval*, (pp. 178-185).

Kaszkiel, M., & Zobel, J. (2001). Effective ranking with arbitrary passages. *Journal of the American Society for Information Science and Technology,* 52(4), 344-364.

Khuri, S., Bäck, T., & Heitkötter, J. (1994). An evolutionary approach to combinatorial optimization problems. In *Proceedings of the 22nd annual ACM computer science conference*, (pp. 66-73).

Kim, S., & Zhang, B.-T. (2000). Web-document retrieval by genetic learning of importance factors for HTML tags. In *Proceedings of the PRICAI-2000 Workshop on Text and Web Mining*, (pp. 13-23).

Kim, S., & Zhang, B.-T. (2001). Evolutionary learning of web-document structure for information retrieval. In *Proceedings of the 2001 Congress on Evolutionary Computation (CEC2001)*, (pp. 1253-1260).

Kim, Y.-H., Kim, S., Eom, J.-H., & Zhang, B.-T. (2000). SCAI experiments on TREC-9. In *Proceedings of the 9th Text REtrieval Conference (TREC-9)*, (pp. 392-399).

Kotsakis, E. (2002). Structured information retrieval in XML documents. In *Proceedings of the ACM Symposium on Applied Computing*, (pp. 663-667).

Meuss, H., & Strohmaier, C. (1999). Improving index structures for structured document retrieval. In *Proceedings of the 21st Annual Colloquium on IR Research (IRSG'99).*

Moody, J. (1994). Prediction risk and architecture selection for neural networks. In V. Cherkassky & J. H. Friedman & H. Wechsler (Eds.), *Statistics to neural networks: Theory and pattern recognition applications.*: Springer-Verlag.

Morgan, J., & Kilgour, A. (1996). Personalising on-line information retrieval support with a genetic algorithm. In A. Moscardini & P. Smith (Eds.), *PolyModel 16: Applications of artificial intelligence* (pp. 142-149).

Pathak, P., Gordon, M. D., & Fan, W. (2000). Effective information retrieval using genetic algorithms based matching function adaptation. In *Proceedings of the The 33rd Hawaii International Conference on System Science*.

Rapela, J. (2001). Automatically combining ranking heuristics for HTML documents. In *Proceedings of the 3rd International Workshop on Web Information and Data Management*, (pp. 61-67).

Robertson, S. E., & Sparck-Jones, K. (1976). Relevance weighting of search terms. *Journal of the American Society for Information Science,* 27(3), 129-146.

Robertson, S. E., Walker, S., Beaulieu, M. M., Gatford, M., & Payne, A. (1995). Okapi at TREC-4. In *Proceedings of the 4th Text REtrieval Conference (TREC-4)*, (pp. 73-96).

Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. *Communications of the ACM,* 18(11), 613-620.

Savoy, J., & Vrajitoru, D. (1996). *Evaluation of learning schemes used in information retrieval* ( CR-I-95-02): Université de Neuchâtel, Faculté de droit et des Sciences Économiques.

Schlieder, T., & Meuss, H. (2000). Result ranking for structured queries against XML documents. In *Proceedings of the DELOS Workshop on Information Seeking, Searching and Querying in Digital Libraries*.

Schlieder, T., & Meuss, H. (2002). Querying and ranking XML documents. *Journal of the American Society for Information Science and Technology,* 53(6), 489-503.

Shin, D., Jang, H., & Jin, H. (1998). Bus: An effective indexing and retrieval scheme in structured documents. In *Proceedings of the 3rd ACM International Conference on Digital libraries*, (pp. 235-243).

Thom, J. A., Zobel, J., & Grima, B. (1995). *Design of indexes for structured documents* ( CITRI/TR-95- 8). Melbourne, Australia: Department of Computer Science, RMIT.

Trotman, A. (2003). Searching structured documents. *Information Processing & Management,* (to appear) doi:10.1016/S0306-4573(03)00041-4, available on ScienceDirect since 6 June 2003.

Vrajitoru, D. (1998). Crossover improvement for the genetic algorithm in information retrieval. *Information Processing & Management,* 34(4), 405-415.

Vrajitoru, D. (2000). Large population or many generations for genetic algorithms? Implications in information retrieval. In F. Crestani & G. Pasi (Eds.), *Soft computing in information retrieval. Techniques and applications* (pp. 199-222): Physica-Verlag.

Weiss, S. M., & Kulikowski, C. A. (1991). *Computer systems that learn: Classification and prediction methods from statistics, neural nets, machine learning, and expert systems.*: Morgan Kaufman.

Wilkinson, R. (1994). Effective retrieval of structured documents. In *Proceedings of the 17th ACM SIGIR Conference on Information Retrieval*, (pp. 311-317).

Wilkinson, R., & Zobel, J. (1994). Comparison of fragmentation schemes for document retrieval. In *Proceedings of the 3th Text REtrieval Conference (TREC-3)*, (pp. 81-84).

Wolff, J. E., Flörke, H., & Cremers, A. B. (1999). *Xpres: A ranking approach to retrieval on structured documents* ( IAI-TR-99-12). Bonn: University of Bonn.

Wolff, J. E., Flörke, H., & Cremers, A. B. (2000). Searching and browsing collections of structural information. In *Proceedings of the IEEE Advances in Digital Libraries*, (pp. 141-150).

Yang, J., Korfhage, R., & Rasmussen, E. (1992). Query improvement in information retrieval using genetic algorithms - a report on the experiments of the TREC project. In *Proceedings of the 1st Text REtrieval Conference (TREC-1)*, (pp. 31-58).