The Accessibility Dimension for Structured Document Retrieval

Thomas Roelleke, **Mounia Lalmas** and **Gabriella Kazai**
Queen Mary, University of London
{thor,mounia,gabs}@dcs.qmul.ac.uk

Ian Ruthven
University of Strathclyde
Ian.Ruthven@cis.strath.ac.uk

Stefan Quicker
University of Dortmund
quicker@ls6.cs.uni-dortmund.de

Abstract

Structured document retrieval aims at retrieving the document components that best satisfy a query, instead of merely retrieving pre-defined document units. This paper reports on an investigation of a *tf-idf-acc* approach, where *tf* and *idf* are the classical term frequency and inverse document frequency, and *acc*, a new parameter called *accessibility*, that captures the structure of documents.

The *tf-idf-acc* approach is defined using a probabilistic relational algebra. To investigate the retrieval quality and estimate the *acc* values, we developed a method that automatically constructs diverse test collections of structured documents from a standard test collection, with which experiments were carried out. The analysis of the experiments provides estimates of the *acc* values.

1 Introduction

In traditional information retrieval (IR) systems [18], retrievable units are fixed. For example, the whole document, or, sometimes, pre-defined parts such as paragraphs constitute the retrievable units. The *logical structure* of documents (chapter, section, table, formula, author information, bibliographic item, etc) is therefore "flattened" and not exploited. Classical retrieval methods lack the possibility to interactively determine the size and the type of retrievable units, that best suit an actual retrieval task or user preferences.

Current research is aiming at developing retrieval models that dynamically return document components of varying complexity. A retrieval result may then consist of several entry points to a same document, whereby each entry point is weighted according to how it satisfies the query. Authors such as [17, 6, 13, 15, 8] have developed and implemented such approaches. Their models exploit the content

and the logical structure of documents to estimate the relevance of document components to queries, based on the aggregation of the estimated relevance of their related components. These models have been based on various formal theories (e.g. fuzzy logic [3], Dempster-Shafer's theory of evidence [13], probabilistic logic [17, 2], and Bayesian inference [15]). What these models have in common is that the basic components of their retrieval function are variants of the two standard term weighting schemes, term frequency (tf) and inverse document frequency (idf). Evidence associated with the logical structure of documents is often encoded into one or both of these dimensions.

In this paper, we make this evidence explicit by introducing the "accessibility" dimension, denoted by acc. This dimension measures the strength of the structural relationship between document components: the stronger the relationship, the more impact has the content of a component in describing the content of its related components (e.g. [17, 6]). We refer to the approach as *tf-idf-acc*. We are interested in investigating how the acc dimension affects the retrieval of document components of varying complexity.

To carry out this investigation, we require a framework that explicitly captures the content and the structure of documents in terms of our *tf-idf-acc* approach. We use a probabilistic relational algebra [17] for this purpose (Section 2). Our investigation also requires test collections of structured documents. Since relevance assessments for structured documents are difficult to obtain and manual assessment is expensive and task specific, we developed an automatic approach for creating structured documents and generating relevance assessments from a flat test collection (Section 3). Finally, we perform retrieval runs for different settings of the accessibility dimension. The analysis provides us with methods for estimating the appropriate setting of the accessibility dimension for structured document retrieval (Section 4).

2 The *tf-idf-acc* Approach

We view a structured document as a tree whose nodes, called *contexts*, are the components of the document (e.g., chapters, sections, etc.) and whose edges represent the composition relationship (e.g., a chapter contains several sections). The *root* context of the tree, which is unique for each document, embodies the whole document. *Atomic* contexts are document components that correspond to the last elements of the composition chains. All other nodes are referred to as *inner* contexts.

Retrieval on structured documents takes both the logical structure and the content of documents into account and returns, in response to a user query, document components of varying complexity (root, inner or atomic contexts). This retrieval methodology combines querying - finding which atomic contexts match the query - and browsing along the documents' structure - finding the level of complexity that best match the query [5]. Take, for example, an article with five sections. We can define a retrieval strategy that is to retrieve the whole article if more than three of its sections are relevant to the query. Dynamically returning document components of varying complexity can, however, lead to user disorientation and cognitive overload [6, 7, 10]. This is because the presentation of document components in the retrieval result does not take into account their structural proximity within the documents. In the above example,

depending on the assigned relevance status values, the article and its sections could be displayed at different positions in the ranking.

To reduce user disorientation and cognitive overload, a retrieval strategy would be to retrieve (and therefore display to the user) *super-contexts* composed of many relevant *sub-contexts* before - or instead of - retrieving the sub-contexts themselves. This strategy would prioritise the retrieval of larger super-contexts from where the sub-contexts could be accessed through direct browsing [6, 5]. Other strategies could favour smaller, more specific contexts, by assigning smaller relevance status values to large contexts.

To allow the implementation of any of the above retrieval strategies, we follow the aggregation-based approach to structured document retrieval (e.g. [17, 11, 2, 15, 6, 8]). We base the estimation of relevance, or in other words, we compute the retrieval status value (RSV) of a context based on a content description that is derived from its content and the content of its sub-contexts. For this purpose, we *augment* the content of a super-context with that of its sub-contexts. The augmentation process is applied to the whole document tree structure, starting with the atomic contexts, where no augmentation is performed, and ending with the root context.

In this framework, the browsing element of the retrieval strategy is then implemented within the method of augmentation. By controlling the extent to which the sub-contexts of a super-context contribute to its representation we can directly influence the derived RSVs.

Via the augmentation process we can also influence the extent to which the individual sub-contexts contribute to the representation of the super-context. This way certain sub-contexts, such as titles, abstracts and conclusions, etc. could be emphasised while others could be de-emphasised.

We model this impact of the sub-contexts on the super-context by expanding the two dimensions, term frequency, tf, and inverse document frequency, idf, standard to IR, with a third dimension, the **accessibility dimension**, acc[1]. We refer to the influencing power of a sub-context over its super-context as the sub-context's importance. **What the *acc* dimension represents, then, is the degree to which the sub-context is important to the super-context.**

In the remainder of this section we show, by means of probabilistic relational algebra defined in [17, 9], how this dimension can be used to incorporate this qualitative notion of context importance for structured document retrieval.

2.1 Probabilistic Relational Algebra

Probabilistic Relational Algebra (PRA) is a language model that incorporates probability theory with the well known relational paradigm. The algebra allows the modelling of document and query representations as relations consisting of probabilistic tuples, and it defines operators, with similar semantics to SQL but

[1] The term "accessibility" is taken from the framework of possible worlds and accessibility relations of a Kripke structure [4]. In [12], a semantics of the *tf-idf-acc* is defined, where contexts are modelled as possible worlds and their structure is modelled through the use of accessibility relations.

with probabilistic interpretations, which allow the description of retrieval strategies.

2.1.1 Document and query modelling

A PRA program describing the representation of a structured document collection uses relations such as *term*, *termspace*, and *acc*. The *term* relation represents the *tf* dimension and consists of probabilistic tuples in the form of *tf_weight term(index_term,context)*, which assign *tf_weight* values to each (*index_term*, *context*) pair in the collection (e.g. index terms that occur in contexts). The value of *tf_weight* $\in$ [0,1] is a probabilistic interpretation of the term frequency. The *termspace* relation models the *idf* dimension by assigning *idf* values to the index terms in the collection. This is stored in tuples in the form of *idf_weight termspace(index_term)*, where *idf_weight* $\in$ [0,1]. The *acc* relation describes the document structure and consists of tuples *acc_weight acc(context_p,context_c)*, where *context_c* is "accessible" from *context_p* with a probability *acc_weight*[2].

A query representation in PRA is described by the *qterm* relation which is in the form of *q_weight qterm(query_term)*, where the *q_weight* describes the importance of the query term, and *query_term* are the terms composing the query.

2.1.2 Relational operators in PRA

Similarly to SQL, PRA supports a number of relational operators. Those used in this paper are SELECT, PROJECT, JOIN, and UNITE. Their syntax and functionalities are described next.

- **SELECT[*criteria*](*relation*)** returns those probabilistic tuples of *relation* that match the specified *criteria*, where the format of *criteria* is $\$column =^3 a\_value$. For example, SELECT[$1=sailing](*termspace*) will return all those tuples from the *termspace* relation that have the term "sailing" in column one. To store the resulting tuple in a relation we use the following syntax: *new_relation* = SELECT[*criteria*](*relation*). The arity of the *new_relation* equals to the arity of *relation*.

- **JOIN[*$column1=$column2*](*relation1,relation2*)** joins (matches) two relations and returns tuples that contain matching data in their respective columns, where *column1* specifies a column of *relation1* and *column2* relates to *relation2*. The arity of the returned tuples is the sum of the arity of the tuples in *relation1* and *relation2*. For example, *new_term*=JOIN[$1=$1](*term,termspace*) will populate *new_term* with tuples that have the same value in column one. The format of the resulting tuples is *new_weight new_term(index_term,context,index_term)*, where the value of *new_weight* is

[2] On a conceptual level there is no restriction on which contexts can access which other contexts, so this formalism can be adopted to describe networked architectures. In this study, however, we only deal with tree type structures where *context_p* and *context_c* form a parent-child (super-context and sub-context) relationship.

[3] Also $<, >, \leq$, etc.

derived from the values of *tf_weight* and *idf_weight* (e.g. based on probability theory, or fuzzy theory).

- **PROJECT[*columns*](*relation*)** returns tuples that contain only the specified columns of *relation*, where the format of *columns* is $*column1*,$*column2*, etc. For example, *new_term*=PROJECT[$1,$3](JOIN[$2=$2](*term,acc*)) returns all *(index_term,context_p)* pairs where the *index_term* occurs in *context_p*'s sub-context. This is because the JOIN operator returns the tuples *new_weight(index_term,context_c,context_p,context_c)*, where *context_p* is the super-context of *context_c*. Projecting column one and three into *new_term* results in *new_weight new_term(index_term,context_p)*.

- **UNITE(*relation1*,*relation2*)** returns the union of the tuples stored in *relation1* and *relation2*. For example, *new_term*=UNITE(*term*,PROJECT[$1,$3](JOIN[$2=$2](*term,acc*))) will produce a relation that includes tuples of the *term* relation and the resulting tuples of the PROJECT operation. The probabilities of the tuples in *new_term* are calculated according to probability theory or fuzzy theory.

Based on the relations describing the document and query space and with the use of the PRA operators we can implement a retrieval strategy that takes into account both the structure and the content of the documents by augmenting the content of sub-contexts into the super-context, where the augmentation can be controlled by the *acc_weight* values.

2.1.3 Retrieval strategies

Let us first model the classical *tf_idf* retrieval function. For this, we use the JOIN and PROJECT operations of PRA.

tfidf_index = PROJECT[$1,$2](JOIN[$1=$1](*term,termspace*))
retrieve_tfidf = PROJECT[$3](JOIN[$1=$1](*qterm,tfidf_index*))

Here, the first function computes the *tf_idf* indexing. It produces tuples in the form of *tf_idf_weight tfidf_index(index_term,context)*. The *tf_idf_weight* is calculated using probability theory and the independence assumption as *tf_weight* × *idf_weight*. The second function joins (matches) the query terms with the *tfidf_index* terms and produces the retrieval results in the form of *rsv retrieve_tfidf(context)*. The RSVs given in *rsv* are calculated according to probability theory assuming disjointness with respect to the term space (*termspace*) as follows:

$$rsv(context) = \sum_{query\_term_i} q\_weight_i \times tf\_weight_i(context) \times idf\_weight_i$$

We use next the *acc* relation to take the structure into consideration. We *augment* the content of the super-context by that of its sub-contexts. This is modelled by the following PRA equation.

tfidfacc_index = PROJECT[$1,$3](JOIN[$2=$2](*tfidf_index,acc*))

The augmented relation consists of tuples *tfidfacc_weight tfidfacc_index(index_term,context_p)* where *context_p* is the super-context of *context_c* which is indexed by *index_term*. The value of *tfidfacc_weight* is calculated as $tf\_idf\_weight \times acc\_weight$ (assuming probabilistic independence). Based on the *tfidfacc_index* we can now define our retrieval strategy for structured documents.

tfidfacc_index = PROJECT[$1,$3](JOIN[$2=$2](*tfidf_index,acc*))
retrieve_tfidfacc = PROJECT[$3](JOIN[$1=$1](*qterm,tfidfacc_index*))
retrieve = UNITE(*retrieve_tfidf,retrieve_tfidfacc*)

The RSVs of the retrieval result are calculated (by the UNITE operator) according to probability theory and assuming independence:

$$\begin{aligned} rsv(context) &= P(retrieve\_tfidf(context)\ OR\ retrieve\_tfidfacc(context)) \\ &= P(retrieve\_tfidf(context)) + P(retrieve\_tfidfacc(context)) - \\ &\quad P(retrieve\_tfidf(context)\ AND\ retrieve\_tfidfacc(context)) \\ &= P(retrieve\_tfidf(context)) + P(retrieve\_tfidfacc(context)) - \\ &\quad P(retrieve\_tfidf(context)) \times P(retrieve\_tfidfacc(context)) \end{aligned}$$

Since the weight of *tfidfacc_index* is directly influenced by the weight associated with *acc*, the resulting RSVs is also dependent on *acc*.

2.2 Example

Consider the following collection of one document doc1 composed of two sections, sec1 and sec2. Terms such as sailing, boats, etc. occur in the collection:

0.1 *term*(sailing, doc1)
0.8 *term*(boats, doc1)
0.7 *term*(sailing, sec1)
0.8 *term*(greece, sec2)

0.4 *termspace*(sailing)
0.3 *termspace*(boats)
0.2 *termspace*(greece)
0.1 *termspace*(santorini)

0.8 *acc*(doc1, sec1)

0.6 *acc*(doc1, sec2)

Let us take the query "sailing boats", represented by the following PRA program.

qterm(sailing)
qterm(boats)

Given this query and applying the classical *tf_idf* retrieval function (as described in the previous section) to our document collection we retrieve the following document components.

0.28 *retrieved_tfidf*(doc1)
0.28 *retrieved_tfidf*(sec1)

Both retrieved contexts have the relevance status value of 0.28. The above retrieval strategy, however, does not take into account the structure of the document, e.g. that sec1 which is about sailing is part of doc1. From a user's point of view, it might be better to retrieve first - or only - doc1 since sec1 can be accessed from doc1 by browsing down from doc1 to sec1. Let us now apply our *tf_idf_acc* retrieval strategy. We obtain:

0.441 *retrieve*(doc1)
0.28 *retrieve*(sec1)

This shows that the RSV of doc1 increases when we take into account the fact that doc1 is composed of sec1, which is also indexed by the term "sailing". This is done using the structural knowledge stored in *acc*. This demonstrates that by using our third dimension, *acc*, we obtain a ranking that exploits the structure of the document to determine which document components should be retrieved higher in the ranking.

In designing applications for structured document retrieval, we are faced with the problem of determining the probabilities (weights) of the *acc* relation. In our retrieval applications so far, constant *acc* values such as 0.5 and 0.6 were used. However we want to establish methods to derive estimates of the *acc* values. To achieve this, we require test collections with controlled parameters to allow us to derive appropriate estimations of the *acc* values with respect to these parameters. In the following section we present a method for creating simulated test collections of structured documents that allow such an investigation.

3 Automatic Construction of Structured Document Test Collections

Although many test collections are composed of documents that contain some internal structure [19, 1], relevance judgements are usually made at the document level (root contexts) or at the atomic context level. This means that they cannot be used for the evaluation of structured document retrieval systems, which would require relevance judgements at the root, atomic *and* inner levels.

Our investigation requires several test collections of structured documents with different characteristics (e.g. depth and width of document tree structure). These will enable us to investigate the *acc* dimension under different conditions. Since relevance assessments for structured documents are difficult to obtain and manual assessment is expensive and task specific, it was imperative to find a way to automatically build such test collections. We developed a methodology that allowed us to create diverse collections of structured documents and automatically generate relevance assessments. Our methodology exploits existing standard test collections with their existing queries and relevance judgements so that no human resources are necessary. In addition our methodology allows the creation of all test collections deemed necessary to carry out our investigation regarding the effect of the *acc* dimension for structured document retrieval.

In Section 3.1 we discuss how we created the structured documents, in Section 3.2 we discuss how we decided on the relevance of the document components, and finally, in Section 3.3 we show the results of the methodology using the CACM collection[4].

3.1 Construction of the documents

Our basic methodology is to *combine* documents from a test collection to form simulated structured documents. That is to treat a number of original documents from the collection as components of a structured document. A simplified version of this strategy was used in [13]. In the remainder of this section we shall present a more sophisticated version, and deal with some of the issues arising from the construction of simulated structured documents. To illustrate our methodology, we used a well known small standard test collection, the CACM test collection.

Using the methodology of combining documents, it is possible to create two types of test collections: *homogeneous* collections in which the documents have the same logical structure and *heterogeneous* collections in which the documents have varying logical structure. In our experiments, Section 4, we use these collections to see how the values of *acc* compare for the two types of collections.

By controlling the number of documents combined, and the way documents are combined, it is also possible to generate different types of structured documents. We used two main criteria to generate structured documents. The first criterion is *width*. This corresponds to the number of documents that are combined at each level, i.e. how many contexts in a document, and how many sub-contexts per context. The second criterion is *depth*. This corresponds to how many levels are in the tree structure. For example a document with no sub-contexts (all the text is at one level) has depth of 1, a document with sub-contexts has depth 2, a document with sub-sub-contexts has depth 3, and so on. Using these criteria it is possible to automatically generate test collections of structured documents that vary in width and depth.

[4]The collection has 3204 documents and 64 queries. See www.dcs.gla.ac.uk/idom/ir_resources/tests_collections/ for details of the collection.

For the experiments we describe in Section 4 we constructed eight homogeneous collections of structured documents. The types of logical structure are shown in Figure 1. Pair (EE), Triple (EEE), Quad (EEEE), Sext (EEEEEE) and Oct (EEEEEEEE) are composed of root and atomic contexts only. These test collections will be useful in estimating the *acc* values based on the width criterion. The other collections Pair-E ((EE)E), Pair-2 ((EE)(EE)) and Triple-3 ((EEE)(EEE)(EEE)) have root, inner and atomic contexts. These collections are useful for estimating the *acc* values based on the depth criterion. Collections of each type were built from the CACM test collection where documents of the test collection, referred to as "original documents", were used to form the atomic contexts.

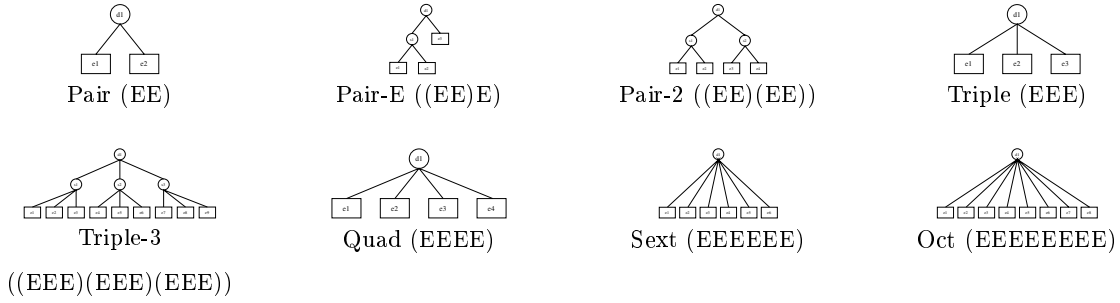


Figure 1: Types of logical structure

We also created one *heterogeneous* test collection, referred to as Mix, which is composed of a mixture of Pair (EE) and Triple (EEE) documents.

We should note here that we are aware that the structured documents created in this manner often will not have a meaningful content and may not reflect term distributions in real structured documents. Nevertheless the use of simulated documents does allow for extensive investigation, Section 4, to provide initial estimates for the *acc* values. We are currently using these estimates in our current work on a real test collection of structured documents (XML-based documents). We are not, therefore, suggesting that we can use the artificially created test collections as substitutes for real documents and relevance assessments. Rather we use them as a test-bed to obtain estimates for parameters that will be used in more realistic evaluations. As mentioned before, the necessity of using artificial test collections comes from the lack of real test collections.

One of the advantages of our approach is that we can automatically create collections diverse in type and size. However we must take steps to ensure that the created collections are realistic and of manageable size to allow experimentation.

With a straight combinatoric approach, we can derive from a collection of N original documents the possible number of structured documents would be N^2 over 2 for the Pair type of collection (that is about 5 million documents for the 3204 documents of the CACM collection). Therefore we require methods to cut down the number of actual documents combined. In the particular experiments we carried out, we used two strategies to accomplish this: discarding "noisy" documents and minimising "dependent" documents. These strategies are based on the assumption that a document which has not been explicitly

marked relevant to a query is considered not-relevant. Both strategies are based on an analysis of the atomic contexts of the structured documents, i.e. the original documents from the CACM collection.

(1) discarding "noisy" documents: If a document's sub-contexts are a mixture of relevant and non-relevant contexts for all queries in the collection then the document is considered to be noisy.

That is, there is no query in the collection for which all sub-contexts are relevant or all sub-contexts are non-relevant. We discard all noisy documents from the collection. This does not mean that we are only considering structured documents where all sub-contexts are in agreement; we simply insist that they are in agreement for at least one query in the collection[5].

(2) minimising "dependent" documents: With a straight combination approach we also have the problem of multiple occurrences of the same atomic concepts (the original documents in the test collection appearing many times). This could mean that our simulated structured documents may be very similar - or *dependent* - due to the overlap between the sub-contexts.

Our second approach to cutting down the number of created documents is therefore to minimise the number of dependent documents.

We do not, however, want to eradicate multiple occurrence completely. First, multiple occurrence mimic real-world situations where similar document parts are used in several documents (e.g. web, hypertext, digital libraries). Second, exclusive usage of an atomic context requires a procedure to determine which atomic context leads to the "best" structured document, which is difficult, if not impossible to assess.

The way we reduce the number of multiple occurrences is to reduce the repeated use of atomic contexts that are *relevant* to the same query. That is, we do not want to create many structured documents that contain the same set of relevant contexts.

Our basic procedure is to reduce the number of documents whose atomic contexts are all relevant to the same query, i.e. composed of components that are all relevant to the query. The reason we concentrate on relevant contexts is that these are the ones we use to decide whether the whole structured document is relevant or not, (see Section 3.2).

For each atomic context, e_i, which is relevant to a query, q_j, we only allow e_i to appear in one document whose other atomic contexts are all relevant to q_j. This reduces multiple occurrences of e_i in documents composed entirely of relevant atomic contexts. As there may be many structured documents containing e_i whose atomic contexts are relevant, we need a method to choose which of these documents to use in the collection. We do this by choosing the document with the lowest noise value. This means that we prefer documents that are relevant to multiple queries over documents that are only relevant to one query. If more than one such documents exists we choose one randomly.

Both these steps reduce the number of structured documents to a manageable size.

[5] This approach can be extended to define the degree of noise we allow in the collection.

3.2 Constructing the relevance assessments

We have so far described how we created the structured documents and how we cut down the potential number of documents created. What we have now to consider are the queries and relevance assessments. The queries and relevance assessments come from the standard test collections that are used to build the simulated structured documents. However, given that a structured document may be composed of a mixture of relevant and non-relevant documents, we have to decide when to classify a root context (structured document), or an inner context, as relevant or non-relevant.

Our approach defines the relevance of non-atomic contexts as the aggregation of the relevance of their sub-contexts.

Let a non-atomic context d be composed of k sub-contexts $e1, ..., ek$. For a given query, we have three cases: all the sub-contexts $e1$ to ek are relevant; all the sub-contexts are not relevant; and neither of the previous two cases holds. In the latter, we say that we have "contradictory" relevance assessments. For the first two cases, it is reasonable to assess that d is relevant and d is not relevant to the query, respectively. In the third case, an aggregation strategy is required to decide the relevance of d to a query. We apply the following two strategies:

- *optimistic relevance*: d is assessed relevant to the query if at least one of its sub-contexts is assessed relevant to the query; d is assessed non-relevant if all its sub-contexts are assessed non-relevant to the query.
- *pessimistic relevance*: d is assessed relevant to the query if all its sub-contexts are assessed relevant to the query; in all other cases, d is assessed non-relevant to the query.

Variants of the above could be used; e.g., d is considered relevant if $2/3$ of its sub-contexts are relevant [11]. We are currently carrying out research to devise strategies that may be closer to user's views of relevance with respect to structured document retrieval[6].

The point of using different aggregation strategies is that it allows us to investigate the performance of the *acc* dimension when using different relevance criteria. For example, the optimistic strategy corresponds to a loose definition of relevance (where a document is relevant if it contains any relevant component) and the pessimistic strategy corresponds to a strict definition of relevance (where all components must be relevant before the structured document is relevant).

3.3 Example

In the previous sections we have shown how we can use existing test collections to create collections of structured documents. These collections can be of varying width and depth, be based on differing notions

[6] For instance, in an experiment related to passage retrieval, some relevant documents contained no parts that were individually assessed relevant by (expert) users [20]. See [14] for a survey on the notion of relevance in IR.

of relevance and be of identical or varying structure (homogeneous or heterogeneous). The flexibility of this methodology is that it allows the creation of diverse collection types from a single original test collection.

The collections we created for the experiments reported in this paper were based on the CACM collection. We have described the collection types, we shall now examine the collections in more detail to show the differences between them.

Table 1 shows the number of root, inner and atomic contexts for the collections. As it can be seen, the homogeneous collections display a relationship between the atomic contexts and root and inner context. For instance, the Pair collection has twice as many atomic contexts as root contexts, the Triple collection has three times as many atomic contexts as root contexts, etc. This does not hold, however, for the heterogeneous Mix collection, which is combined of a mixture of document types.

Coll.	Num. Root	Num. Inner	Num. Atomic	Total Num.
Pair	383	0	766	1149
Pair-E	247	247	741	1235
Triple	247	0	741	988
Quad	180	0	720	900
Pair-2	180	360	720	1260
Triple-3	66	198	594	858
Sext	109	0	654	763
Oct	80	0	640	720
Mix	280	0	700	980

Table 1: Number of contexts

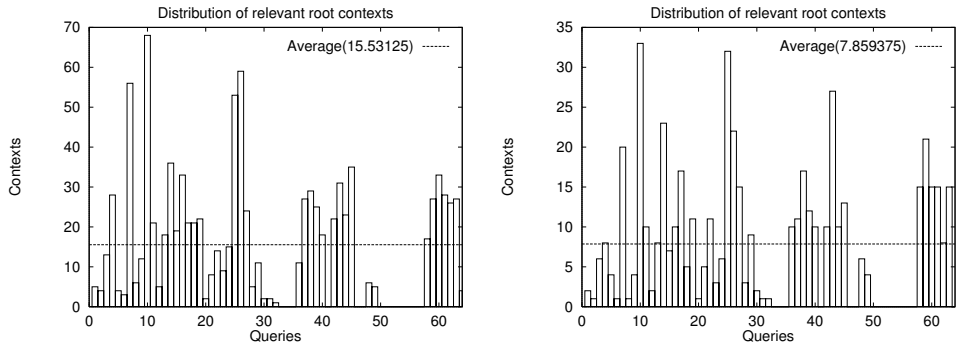
One of the ways we cut down the number of created structured documents was to reduce the number of multiple occurrences of atomic contexts. As shown in Table 2, we do not exclude all multiple occurrences, however such occurrences are rare. For, example, in the Triple-3 collection, 20 of the original 3204 documents are used three times among the 594 atomic contexts.

The above two measures are independent of how we decide on the relevance of a context, i.e. whether we use the optimistic or pessimistic aggregation strategy. The choice of aggregation strategy will affect the number of relevant contexts. As an example we show in, Figure 2, the number of relevant root contexts when using the Pair collection, (full figures can be found in [16]). As it can be seen, for the optimistic aggregation strategy we have almost twice as many relevant root contexts (average 15.53 per query) as for the pessimistic aggregation strategy (average 7.85 per query). This demonstrates that the aggregation strategy can be used to create collections with different characteristics.

In this section we described the creation of a number of collections based on the CACM collection. In the following section we investigate the *acc* dimension using these collections.

Occurrence frequency	1	2	3	4	5	6
Pair	398	86	33	14	7	1
Pair-E	392	82	31	14	6	1
Triple	392	82	31	14	6	1
Quad	388	82	28	12	6	1
Pair-2	391	84	32	11	3	1
Triple-3	336	83	20	8	0	0
Sext	362	74	22	12	6	0
Oct	352	75	21	11	5	1
Mix	366	85	27	12	7	0

Table 2: Multiple occurrence of contexts



Pair, opt, root

Pair, pess, root

Figure 2: Distribution of relevant contexts

4 Experiments

Using the different types of collections, and their associated properties, we carried out a number of experiments to investigate the accessibility dimension for the retrieval of structured documents. With our set of test collections of structured documents and their various and controlled characteristics, we studied the effect of different *acc* values on the retrieval quality.

We targeted the following questions:

1. Is there an optimal setting of the *acc* parameter for a context with n sub-contexts? (Section 4.1). An optimal setting is one which gives the best average precision.
2. With high *acc* values, we expect large contexts to be retrieved with a higher RSV than small contexts. What is the "break-even point", i.e. which setting of *acc* will retrieve large and small contexts with the same preference? (Section 4.2)

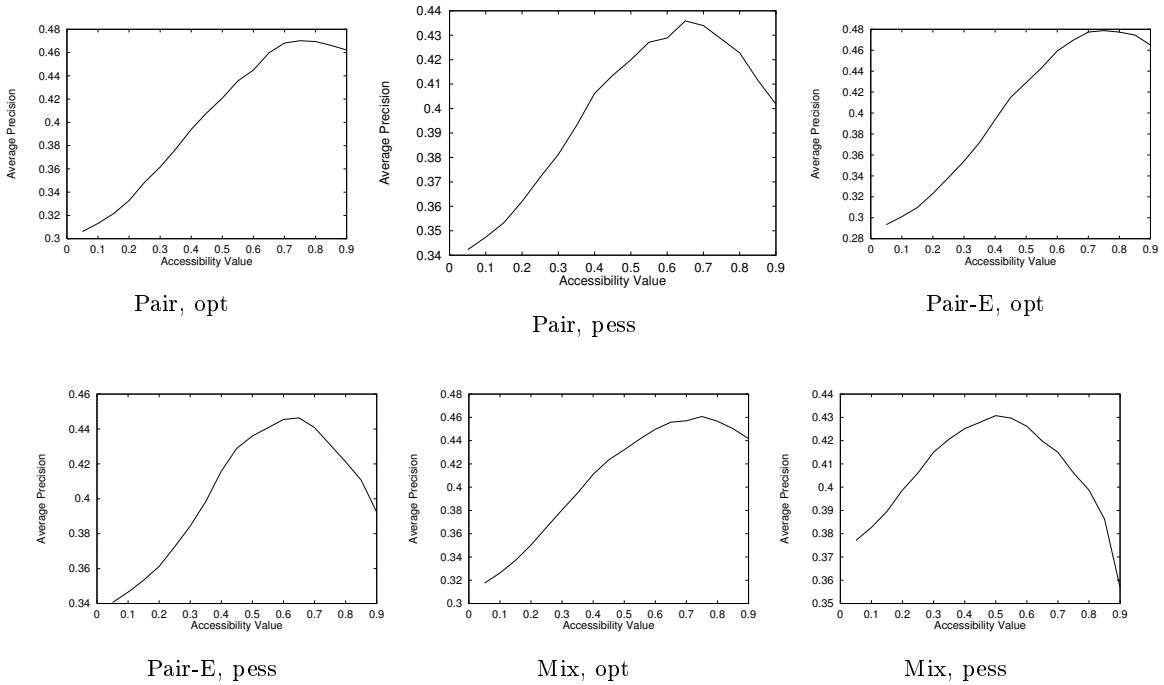
4.1 Optimal values of the accessibility dimension

For all our constructed collections, for increasing values of *acc* (ranging from 0.1 to 0.9), we computed the RSV of each context, using the augmentation process described in Section 2. With the obtained rankings (of root, inner and atomic contexts) and our relevance assessments (optimistic or pessimistic), we calculated precision/recall values and then the average precision values. The graphs in Figure 3 show for each accessibility value the corresponding average precision. We show the graphs for Pair, Pair-E and Mix only. All graphs show a "bell shape". The optimal accessibility values and their corresponding maximal precision values are given in Table 3.

	Optimistic relevance		Pessimistic relevance	
collection	max. av. precision	*acc*	max. av. precision	*acc*
Pair	0.4702	0.75	0.4359	0.65
Triple	0.4719	0.6	0.4479	0.45
Quad	0.455	0.55	0.4474	0.35
Sext	0.4431	0.45	0.4507	0.25
Oct	0.4277	0.35	0.4404	0.2
Pair-2	0.4722	0.8	0.4556	0.6
Pair-E	0.4787	0.75	0.4464	0.65
Triple-3	0.4566	0.65	0.4694	0.4
Mix	0.4608	0.75	0.4307	0.5

Table 3: Optimal accessibility values and corresponding maximum precision

Looking at Pair, Triple, Quad, Sext and Oct, we can see that the optimal accessibility values decrease



Pair, opt | Pair, pess | Pair-E, opt

Pair-E, pess | Mix, opt | Mix, pess

Figure 3: Accessibility values and corresponding average precision values

with the number of sub-contexts. This holds for both relevance aggregation strategies. The *acc* value can be approximated with the function

$$acc = a \cdot \frac{1}{\sqrt{n}}$$

where n is the number of sub-contexts. The parameter a depends on the relevance aggregation strategy. With the method of least square polynomials (see Appendix), values of a are 1.068 and 0.78 for optimistic and pessimistic relevance assessments, respectively.

The optimal *acc* values obtained for Pair and Triple are close to those of Pair-2 and Pair-E, and Triple-3, respectively. This indicates that for depth-two collections (Pair-2, Pair-E, Triple-3) we can apply the above estimates for *acc* independently of the depth of the collection, indicating that approximations based on the number of sub-contexts seem appropriate.

The *acc* value for Mix used the same fixed accessibility values for all documents, whether they were Pair or Triple documents. This could be considered as "unfair", since, as discussed above, the setting of the *acc* for a context depends on the number of its sub-contexts. Therefore, we performed an additional experiment, where the *acc* values were set to 0.75 and 0.6, respectively, for contexts with two and three sub-contexts in the optimistic relevance case, and 0.65 and 0.45 for the pessimistic case. These are the optimal accessibility values obtained for Pair and Triple (see Table 3). The average precision values are 0.4615 and 0.4301 for optimistic and pessimistic relevance assessments, respectively. Compared to the values obtained with fixed accessibility values (0.4608 and 0.4307, respectively), there is no significant change. An experimental setting with a more heterogeneous collection would be more appropriate for

comparing fixed and variable settings of the *acc* value.

From the results on the homogeneous collections, we conclude that we can set the *acc* parameter for a context according to the function $a \cdot \frac{1}{\sqrt{n}}$ where a can be viewed as the parameter reflecting the relevance aggregation strategy.

4.2 Large or small contexts

One major role of the accessibility dimension is to emphasise the retrieval preference of large vs small contexts. For example, contexts deeper in the structure (small contexts) should be retrieved before contexts upper (large contexts) in the structure when specific contexts are preferred to more exhaustive contexts [6]. The *acc* value gives powerful control regarding exhaustiveness and specificity of the retrieval. With small *acc* values, small contexts "overtake" large contexts, whereas with high *acc* values large contexts dominate the upper ranks.

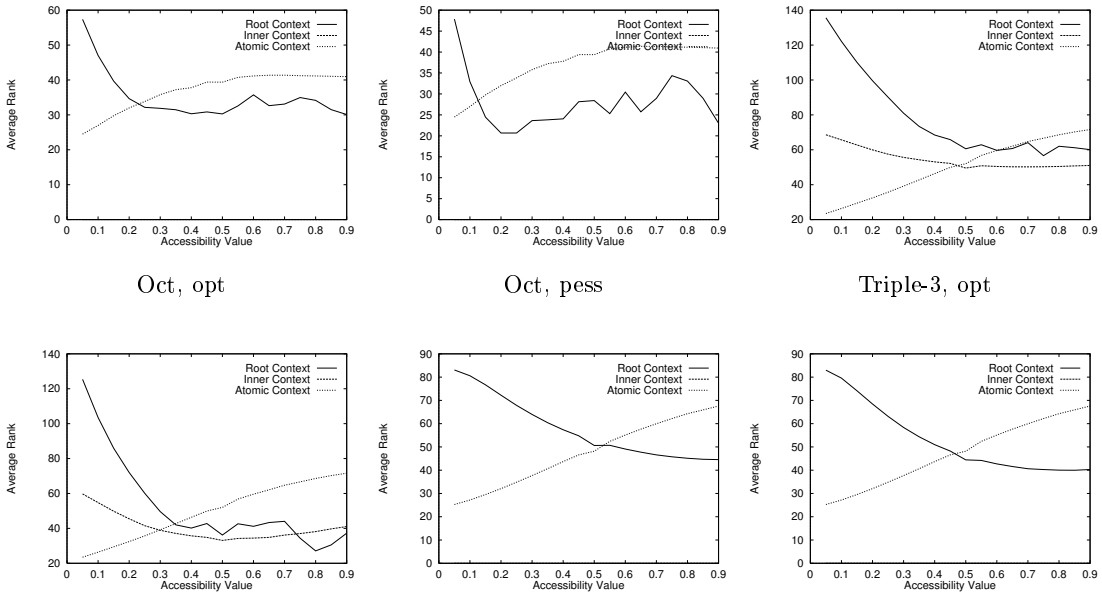
For demonstrating and investigating this effect, we produced for each collection with our *tf-idf-acc* method defined in Section 2 a ranked list of contexts for different *acc* values, ranging again from 0.1 to 0.9. For each type of contexts (atomic, inner and root), we calculate its average rank over all retrieval results for a collection. These average values are then plotted into a graph in relation to the accessibility values. Figure 4 shows the obtained graphs for Oct, Triple-3 and Mix. In all graphs, the root context curve starts in the upper left corner, whereas the atomic context curve starts in the lower left corner. For instance, we see that for the Oct collection, the "break-even point" is around 0.1 and 0.2 for pessimistic relevance assessment.

With the Triple-3 collection we obtain three break-even points for root-inner, root-atomic, and inner-atomic. Whereas the average rank of inner nodes does not vary greatly with varying *acc* values, the effect on root and atomic contexts is similar to the effect observed with the Oct collection, but with different break-even points values (e.g. around $0.4 - 0.5$ for optimistic relevance assessment). For the Mix collection the break-even-point locates around 0.5, a higher value than that for the Oct collection.

Whereas as in Section 4.1, the maximum average precision leads to a setting of *acc*, the experiments regarding small and large contexts provide us with a second source for setting the *acc* value, one that controls the retrieval of exhaustive vs specific document entry points.

5 Conclusion

In this work we investigated how to explicitly incorporate the notion of structure into structured document retrieval. This is in contrast to other research, (e.g. [3, 15, 8, 20]), where the structure of a document is only implicitly captured within the retrieval model. The advantage of our approach is that we can investigate the effect of differing document structures upon the success of structured document retrieval.



Oct, opt | Oct, pess | Triple-3, opt

Triple-3, pess | Mix, opt | Mix, pess

Figure 4: Effect of the accessibility on the types of retrieved contexts

Our approach to structured documents ranks document contexts (document components of varying granularity) based on a description of their individual content augmented with that of their sub-contexts. Therefore a document's context encapsulates the content of all its sub-contexts taking into account their importance. This was implemented using PRA, a probabilistic relational algebra. In our model, and implementation, we quantitatively incorporated the degree to which a sub-context contributes to the content of a super-context using the *acc* dimension, i.e. higher *acc* values mean that the sub-context contributes more to the description of a super-context.

We carried out extensive experiments on collections of documents with varying structure to provide estimates for *acc*. This investigation is necessary to allow the setting of *acc* to values that will facilitate the retrieval of document components of varying granularity.

The experiments required the development of test collections of structured documents. We developed a methodology for the automatic construction of test collections of structured documents using standard test collections with their set of documents, queries and corresponding relevance assessments. The methodology makes it possible to generate test collections of structured documents with varying width and depth, based on differing notions of relevance and with identical or varying structure.

The analysis of the retrieval results allowed us to derive a general recommendation for appropriate settings of the *acc* value for structured document retrieval. The *acc* values depend on the number of sub-contexts of a contexts, and the relevance assessment aggregation strategies. They also depend on the required exhaustiveness and specificity of the retrieval. These results are being used as the basis for an evaluation on a real structured document collection.

References

[1] Baeza-Yates, R., and Ribeiro-Neto, B. *Modern Information Retrieval.* Addison Wesley, 1999.

[2] Baumgarten, C. A probabilistic model for distributed information retrieval. In *Proceedings of ACM-SIGIR Conference on Research and Development in Information Retrieval* (Philadelphia, USA, 1997), pp. 258–266.

[3] Bordogna, G., and Pasi, G. Flexible querying of structured documents. In *Proceedings of Flexible Query Answering Systems (FQAS)* (Warsaw, Poland, 2000), pp. 350–361.

[4] Chellas, B. *Modal Logic.* Cambridge University Press, 1980.

[5] Chiaramella, Y. Browsing and querying: two complementary approaches for multimedia information retrieval. In *Proceedings Hypermedia - Information Retrieval - Multimedia* (Dortmund, Germany, 1997).

[6] Chiaramella, Y., Mulhem, P., and Fourel, F. A model for multimedia information retrieval. Tech. Rep. Fermi ESPRIT BRA 8134, University of Glasgow, 1996.

[7] Edwards, D., and Hardman, L. Lost in hyperspace: Cognitive navigation in a hypertext environment. In *Proceedings of Hypertext I* (1989), pp. 105–125.

[8] Frisse, M. Searching for information in a hypertext medical handbook. *Communications of the ACM 31*, 7 (1988), 880–886.

[9] Fuhr, N., and Roelleke, T. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Transactions on Information Systems 14*, 1 (1997).

[10] Iweha, C. *Visualisation of Structured Documents: An Investigation into the Role of Visualising Structure for Information Retrieval Interfaces and Human Computer Interaction.* PhD thesis, Queen Marty & Westfield College, 1999.

[11] Lalmas, M., and Moutogianni, E. A Dempster-Shafer indexing for the focussed retrieval of hierarchically structured documents: Implememtation and experiments on a web museum collection. In *RIAO* (2000).

[12] Lalmas, M., and Roelleke, T. Four-valued knowledge augmentation for structured document retrieval. Submitted for Publication.

[13] Lalmas, M., and Ruthven, I. Representing and retrieving structured documents with Dempster-Shafer's theory of evidence: Modelling and evaluation. *Journal of Documentation 54*, 5 (1998), 529–565.

[14] Mizzaro, S. Relevance: The whole story. *Journal of the America Society for Information Science 48*, 9 (1997), 810–832.

[15] Myaeng, S., Jang, D. H., Kim, M. S., and Zhoo, Z. C. A flexible model for retrieval of SGML documents. In *Proceedings of ACM-SIGIR Conference on Research and Development in Information Retrieval* (Melbourne, Australia, 1998), pp. 138–145.

[16] Quicker, S. Relevanzuntersuchung fur das Retrieval von strukturierten Dokumenten. Master's thesis, University of Dortmund, 1998.

[17] Roelleke, T. *POOL: Probabilistic Object-Oriented Logical Representation and Retrieval of Complex Objects - A Model for Hypermedia Retrieva.* PhD thesis, University of Dortmund, Germany, 1999.

[18] van Rijsbergen, C. J. *Information Retrieval*, 2 ed. Butterworths, London, 1979.

[19] Voorhees, E., and Harman, D. Overview of the Fifth Text REtrieval Conference (TREC-5). In *Proceedings of the 5th Text Retrieval Conference* (Gaitherburg, 1996), pp. 1–29.

[20] Wilkinson, R. Effective retrieval of structured documents. In *Proceedings of ACM-SIGIR Conference on Research and Development in Information Retrieval* (Dublin, Ireland, 1994), pp. 311–317.

Least square polynomials

Consider the experimental values of *acc* in relation with the square root of the number of sub-contexts. Assuming $a \cdot \frac{1}{\sqrt{n_i}}$ where n_i ranges in the set $\{2, 3, 4, 6, 8\}$ is the function for estimating the optimal accessibility values, we apply least square polynomials as follows for calculating a.

$$\begin{aligned}
err(a) &= \sum_i \left(y_i - a \cdot \frac{1}{\sqrt{n_i}} \right)^2 \\
\frac{err(a)}{\delta a} &= 2 \cdot \sum_i \left(y_i - a \cdot \frac{1}{\sqrt{n_i}} \right) \cdot \frac{1}{\sqrt{n_i}} \\
0 &= \sum_i \left(y_i \cdot \frac{1}{\sqrt{n_i}} - a \cdot \frac{1}{n_i} \right) \\
a \cdot \sum_i \frac{1}{x_i} &= \sum_i y_i \cdot \frac{1}{\sqrt{n_i}} \\
a &= \frac{\sum_i y_i \cdot \frac{1}{\sqrt{n_i}}}{\sum_i \frac{1}{n_i}}
\end{aligned}$$