Vague Content and Structure (VCAS) Retrieval for Document-Centric XML Collections

Shaorong Liu, Wesley W. Chu and Ruzan Shahinian
UCLA Computer Science Department, Los Angeles, CA, USA 90095
{sliu, wwc, ruzan}@cs.ucla.edu

ABSTRACT
Querying document-centric XML collections with structure conditions improves retrieval precisions. The structures of such XML collections, however, are often too complex for users to fully grasp. Thus, for queries regarding such collections, it is more appropriate to retrieve answers that approximately match the structure and content conditions in these queries, a process also known as vague content and structure (VCAS) retrieval. Most existing XML engines, however, only support content-only (CO) retrieval and/or strict content and structure (SCAS) retrieval. To remedy these shortcomings, we propose an approach for VCAS retrieval using existing XML engines. Our approach first decomposes a VCAS query into a SCAS sub-query and a CO sub-query, then uses existing XML engines to retrieve SCAS results and CO results for the decomposed sub-queries, and finally combines results from both retrievals to produce approximate results for the original query. Further, to improve retrieval precision, we propose two similarity metrics to adjust the scores of CO retrieval results by their relevancies to the path condition for the original query target. We evaluate our VCAS retrieval approach through extensive experiments with the INEX 04 XML collection and VCAS query sets. The experimental results demonstrate the effectiveness of our VCAS retrieval approach.

1. INTRODUCTION
The increasing use of the eXtensible Markup Language (XML) in scientific data repositories, digital libraries and web applications has increased the need for effective retrieving of information from these XML repositories. The INEX Initiative for the Evaluation of XML retrieval (INEX) [1], for example, was established in April 2002 and has prompted researchers worldwide to promote the evaluation of effective XML retrieval.

XML information can be retrieved by means of either content-only (CO) or content-and-structure (CAS) queries. CO queries, similar to keyword searches in text retrieval, contain only content related conditions. CAS queries contain both content and structure conditions, in which users specify not only what a result should be about (via content conditions) but also what that result is (via structural constraints). Thus, CAS queries are more expressive and have better retrieval precision as demonstrated in past research [10, 11, 13]. Specifying exact structural constraints in queries for document-centric XML collections, however, is not an easy task. Such collections are usually marked up with a large variety of tags. For example, there are about 170 different tags in the INEX document collection. Thus, it is often difficult for users to completely grasp the structure properties of such collections and specify the exact structural constraints in queries. Therefore, for queries regarding such collections, it is more appropriate to retrieve answers that approximately match the structure and content conditions in these queries, a process also known as vague content and structure (VCAS) retrieval. For example, suppose a user is looking for article sections about “internet security.” The VCAS retrieval may return article paragraphs about “internet security” to the user, even though they do not strictly satisfy the query’s structural constraint (i.e., article sections).

Most existing XML engines, however, only support content only retrieval and/or strict content and structure (SCAS) retrieval. In SCAS retrieval, a query’s content conditions can be loosely interpreted, but the query target’s structural constraint must be processed strictly. A query target is a special node in the query’s structure conditions, whose matching elements in XML collections are returned as results. For example, suppose a user is interested in article sections about “internet security.” Thus, compared to the SCAS retrieval, the new feature in the VCAS retrieval is the approximate processing of a query target’s structural constraint. This introduces two challenges to VCAS retrieval: 1) how to extend existing XML engines to derive results that approximately satisfy a query target’s structure condition; and 2) how to measure the relevancy of a result to a query target’s structural constraint.

Many existing approaches to XML VCAS retrieval can be classified into two categories: 1) content-only approaches (e.g., [12]); and 2) relaxation-based approaches [1, 2]. The former approaches transform a VCAS query into a CO query by ignoring structural constraints; and such approaches are simple because XML engines can be directly used for the VCAS retrieval without any extensions. Such approaches, however, lose the precision benefits that can be derived from XML structures. The latter approaches relax a query’s structural constraints and then retrieve the SCAS results for the relaxed queries, which are approximate answers to the original query. Such approaches are systematic and efficient, but they may miss relevant answers due to its strict structural relaxation semantics.

To remedy these problems, in this paper, we propose a general approach that extends existing XML engines for CO and SCAS retrieval to support effective VCAS retrieval. Our approach combines the simplicity advantage provided by CO retrieval and the precision advantage rendered by SCAS retrieval. Our retrieval process consists of three steps:

- Decomposition. We decompose a VCAS query into a CO sub-query and a SCAS sub-query such that both sub-queries can be processed by existing XML engines.
• **Retrieval.** We use existing XML engines to retrieve CO and SCAS results for the two sub-queries.

• **Combination.** Results from the SCAS retrieval are answer to one part of the original query and results from the CO retrieval are approximate answers to the remaining part of the original query. Thus, results from both retrievals can be combined to produce approximate answers to the original query.

To improve retrieval precision, we adjust the score of a CO sub-query result by the relevancy of the result to the path condition for the query target, which is measured by target path similarity. We propose two metrics to compute the target path similarity.

To empirically evaluate the effectiveness of the proposed VCAS retrieval approach, we conduct extensive experiments on the INEX 04 document collection with all the 33 queries in the VCAS task. We use the INEX 04 VCAS relevance assessments as the “gold standard” to evaluate our experimental results.

The rest of the paper is organized as follows. Section 2 introduces the XML data model, query language and VCAS retrieval task. In Section 3, we present our XML VCAS retrieval approach and the similarity metrics. We describe our experimental studies in Section 4. Section 5 overviews related works and Section 6 concludes the paper.

### 2. BACKGROUND

#### 2.1 XML Data Model

We model an XML document as an ordered, labeled tree where each element (attribute) is represented as a node and each element-to-sub-element (element-to-attribute) relationship is represented as an edge between the corresponding nodes. We represent each node as a triple $(id, label, <text>)$, where $id$ uniquely identifies the node, $label$ is the name of the corresponding element or attribute, and $text$ is the corresponding element’s textual content or attribute’s value. Text is optional because not every element contains textual content. We consider an attribute as a special sub-element of an element and a reference IDREF as a special type of value.

For example, Figure 1 shows a tree representation of a sample XML document collection. Each circle represents a node with the node $id$ inside the circle and $label$ beside the circle. To distinguish text nodes from element (attribute) nodes, the $text$ of a node is linked to the node with a dotted line.

We now introduce the definition for label path, which is useful for describing the group representation of an XML tree in Section 3. A label path for a node $v$ in an XML tree is a sequence of slash-separated labels of the nodes on the path from the root node to $v$. For example, node 6 in Figure 1 can be reached from the root node through the path: node $0 \rightarrow 1 \rightarrow 5 \rightarrow 6$. Thus, the label path for node 6 is: `/articles/article/body/section`.

#### 2.2 Query Language

We use a content-oriented XPath-like query language called Narrowed Extended XPath I (NEXI) [14], which is introduced by INEX. NEXI is based on a subset of XPath path expressions [1] with an extension of about functions. The syntax of NEXI is:

$$\text{path}_1[\text{abouts}_1]//...//\text{path}_n[\text{abouts}_n]$$

where each path is a sequence of nodes connected by either parent-to-child (“/”) or ancestor-to-descendant (“//”) axes; each about is a Boolean combination of about functions.

An about function, in the format of about(path, cont), requires that a certain context (i.e., path) should be relevant to a specific content description (i.e., cont). Given an about function $\alpha$, we use $\alpha.path$ and $\alpha.cont$ to represent its path and cont parameters respectively. About functions have non-Boolean semantics and thus they are the bases for result ranking.

With the introduction of the NEXI query format, now let us look at a sample query in the NEXI format. For example, suppose a user is searching for information on ‘route planning’ in articles that give an overview of intelligent transportation systems. Since ‘route planning’ is only one aspect of an intelligent transportation system, the user limits the search on ‘route planning’ to document components, such as section. Thus she formulates her information needs in the following NEXI query $Q$:

$$Q: \text{//article}[\text{about(.//title, overview) and about(., \text{intelligent transportation system})}] //\text{body}[/section][\text{about(., route planning)}]$$

With the description of the NEXI query format, we now introduce some notations and terminologies, which are useful for describing our VCAS retrieval methodology in Section 3.

Given a NEXI query $Q$ in the format of path$_1[\text{abouts}_1]//...//\text{path}_n[\text{abouts}_n]$, we call the last node on path$_n$ whose matches are returned as results, the query target. For example, in $Q$, node section is the query target. Further, we define target content condition, denoted as $C(Q)$, to be the union of the content descriptions in about$_n$. For instance, ‘route planning’ is $Q$’s target content condition. Finally, we call path$_n$, ..., path$_1$, the support paths and path$_n$ the target path. We represent the target path in a query $Q$ as $P_t(Q)$. Support paths and the target path provide different structural hints to a search engine: support paths indicate where to search and a target path suggests what to return. For example, in $Q$ //body//section is the support path and //body//section is the target path.
2.3 VCAS Retrieval
Specifying structure conditions in a CAS query is not an easy task, in particular for document-centric XML collections with a large variety of tag names. When users specify structural constraints in queries, they often have only a limited knowledge of the structure properties of such collections. In such cases, if we process a query’s structure conditions strictly, we may miss results that are not in rigid conformance with the structural constraints, but are highly relevant to users’ information needs. Thus, XML vague content and structure (VCAS) retrieval is introduced. The goal of VCAS retrieval is to help users with limited structural knowledge make the maximum utilization of XML structures for more precise retrieval. In VCAS retrieval, both the structure and content conditions can be processed approximately. Thus, the relevancy of a result is judged based on whether it satisfies a user’s information needs, but not on whether it strictly conforms to the structural constraints of the query. For example, for the sample query \( Q \), a user may judge XML nodes 4, 9 and 18 in Figure 1 to be relevant, although these nodes do not exactly match the structure conditions in \( Q \).

3. PROCESSING VCAS QUERIES
In this section, we present a general approach to XML VCAS retrieval, which consists of three steps: decomposition, retrieval and combinations.

3.1 Decomposition
Given a CAS query \( Q \), in principle, both its support paths and target path can be approximately processed. In this paper, we assume that users are strict in their search contexts but flexible in returning answers. Therefore, we process support paths strictly and the target path approximately.

With this assumption, our decomposition strategy is to decompose a VCAS query \( Q \) into two sub-queries: a CO sub-query, \( Q^{co} \), consisting of the target content condition and a SCAS sub-query, \( Q^{scas} \), consisting of support paths and all the \( about \) functions associated with these paths. Thus, we can use existing XML engines to perform CO and SCAS retrievals on the decomposed sub-queries respectively to collect XML nodes that approximately satisfy the target path and that strictly conform to the support paths. The following illustrates our decomposition process:

\( Q: \) path\([abouts\_1//\ldots//path\_n\[about\_s\_1\]]\)\)

\( Q^{co}: \) //\*[about\_(//title, overview)]\), where \( C(Q) \) is the target content condition in \( Q \).

\( Q^{scas}: \) path\_1\[about\_s\_1\]//\ldots// path\_n\[about\_s\_n\]//

For example, the sample query \( Q \) in Section 2.2 is decomposed into the following two sub-queries: \( Q^{co}: \) //\*[about\_1, route planning]\)
\( Q^{scas}: \) //article[about\_(//title, overview) and about\_(//intelligent transportation system)]

\( Q^{co} \) searches for all the XML nodes relevant to ‘route planning’; and \( Q^{scas} \) searches for article nodes relevant to ‘intelligent transportation system’ with a descendant node title about ‘overview’.

3.2 Retrieval
After the query decomposition step, we use an existing XML IR engine to process the CO sub-query using CO retrieval and the SCAS sub-query using SCAS retrieval. We use our XML IR engine [8] to perform both retrievals. Our VCAS retrieval approach, however, can be used by any XML IR engine. In the following, we first overview our ranking model, and then describe how we apply this model to rank the CO and SCAS retrieval results.

3.2.1 Ranking model
The ranking model used in our XML IR engine is called the extended vector space model. This model measures the relevancy of an XML node \( v \) to an \( about \) function \( \alpha \), where \( v \) satisfies the path condition in \( \alpha \). The model consists of two components: weighted term frequency (wtf) and inverse element frequency (ief).

**Weighted term frequency.** Given a term \( t \) and an XML node \( v \), suppose there are \( m \) different descendant nodes of \( v \), say \( v_1', \ldots, v_m' \), that contain term \( t \) in their texts. Let \( p_i (1 \leq i \leq m) \) be the path from node \( v \) to node \( v_i' \) and \( w(p_i) \) be the weight of path \( p_i \), then the weighted term frequency of term \( t \) in node \( v \), denoted as \( wtf(t, v) \), is:

\[
twf(t, v) = \frac{\sum_{i=1}^{m} w(p_i) \cdot wtf(t, v_i')}{\sum_{i=1}^{m} w(p_i)}
\]

That is, the weighted term frequency of a term \( t \) in an XML node \( v \) is the sum of the frequencies of \( t \) in the text of \( v_i' \) adjusted by the weight of the path from \( v \) to \( v_i' \). The weight of a path is the product of the weights of all the nodes on the path, where the weight of a node is user-configurable.

**Inverse element frequency.** The inverse element frequency of a term \( t \) in an \( about \) function \( \alpha \), denoted as \( ief(t, \alpha) \), is:

\[
ief(t, \alpha) = \log \frac{n_1}{n_2}
\]

where \( n_1 \) is the number of XML nodes that satisfy the path condition in the \( about \) function \( \alpha \), i.e., \( \alpha.path \); and \( n_2 \) is the number of XML nodes that satisfy \( \alpha.path \) and contain \( t \) in texts.

**Relevancy score function.** The relevancy score of an XML node \( v \) to an \( about \) function \( \alpha \), denoted as \( score(v, \alpha) \), is the sum of all the query terms’ weighted frequencies in node \( v \) adjusted by their corresponding inverse element frequencies. That is,

\[
score(v, \alpha) = \sum_{te\alpha, cont} wtf(te, v) \cdot ief(te, \alpha)
\]

The extended vector space model is effective in measuring the relevancy scores of XML nodes to \( about \) functions in SCAS queries [8]. Relevant nodes to such \( about \) functions, however, usually are of relatively similar sizes because these nodes must satisfy the path conditions of the \( about \) functions. For example, all the relevant nodes to the \( about \) function \( about(/\title, overview) \) are title nodes. This, however, may not be the case for the \( about \) function in a CO sub-query \( Q^{co} \). The path condition of the \( about \) function in \( Q^{co} \) is a wildcard, which is so general that all XML nodes are exact matches to the path condition. Thus, nodes relevant to the \( about \) function in \( Q^{co} \) are of varying sizes. The larger a node, the less specific it is to an \( about \) function. Thus, to compute the relevancy of an XML node \( v \) to an \( about \) function \( \alpha \) either in a CO or a SCAS sub-query, we modify the score function in (3) to:

\[
score(v, \alpha) = \sum_{te\alpha, cont} \frac{wtf(te, v) \cdot ief(te, \alpha)}{\log_2 \text{wsize}(v)}
\]
where \( \text{wsize}(v) \) is the weighted size of a node \( v \). Given an XML node \( v \), suppose \( v \) has \( r \) different child nodes \( v_1, v_2, \ldots, v_r \). Let \( \text{size}(v) \) be the number of terms in the text in node \( v \), then \( \text{wsize}(v) \) is recursively defined as follows:

\[
\text{wsize}(v) = \text{size}(v) + \sum_{i=1}^{r} (\text{wsize}(v_i) \ast w(v_i))
\]

That is, the weighted size of a node \( v \) is the text size of node \( v \) plus the sum of the weighted sizes of its child nodes \( v_i \) adjusted by their corresponding weights.

### 3.2.2 CO retrieval

An XML node \( v \) is relevant to a CO sub-query \( Q^\text{co} \) if either the text of \( v \) or that of any descendant node of \( v \) satisfies the content condition in \( Q^\text{co} \). For example, for the CO sub-query \( Q^\text{co} \), the text of nodes 4, 9 and 18 as well as their ancestor nodes 1, 2, 5, 8, 10 and 17 are relevant to \( Q^\text{co} \).

A CO sub-query \( Q^\text{co} \) contains only one about function. Thus, the relevancy score of an XML node \( v \) to \( Q^\text{co} \), denoted as score \((v, Q^\text{co})\), is the relevancy score of \( v \) to the about function in \( Q^\text{co} \), which can be calculated using (4).

### 3.2.3 SCAS retrieval

An XML node \( v \) is relevant to a SCAS sub-query \( Q^\text{scas} \) if it strictly conforms to the structure conditions in \( Q^\text{scas} \) and approximately satisfies the content conditions in \( Q^\text{co} \). For example, nodes 1 and 10 in Figure 1 are relevant to \( Q^\text{scas} \). This is because both nodes strictly conform to the structure conditions: both are article nodes with a descendant node title. For example, node 1 has a descendant node title (i.e., node 3). Also both article nodes are about ‘intelligent transportation system’ and both title nodes are on ‘overview’.

During query processing, if an XML node \( v \) is a match to a query node with an about function \( \alpha \), then the relevancy score of \( v \) to \( \alpha \) is calculated using (4). The relevancy score of a SCAS result \( v \) to \( Q^\text{scas} \), denoted as score \((v, Q^\text{scas})\), is the sum of all the relevancy scores of the corresponding nodes to the about functions in \( Q^\text{scas} \). For example, there are two about functions in \( Q^\text{scas} \):

\[\alpha_1: \text{about}(/\text{/article, intelligent transportation system})\]

\[\alpha_2: \text{about}(/\text{/article/title, overview})\]

The relevancy score of a SCAS result, say node 1 in Figure 1, to \( Q^\text{scas} \) is the relevancy score of node 1 to \( \alpha_1 \) plus the relevancy score of node 3 to \( \alpha_2 \).

### 3.3 Combination

After the retrieval step, we have two lists of results: one list of results from the CO retrieval, \( R^\text{co} \), and another list of results from the SCAS retrieval, \( R^\text{scas} \). Each result is a pair of \((v, s)\), where \( v \) is an XML node and \( s \) is the score indicating the relevancy of \( v \) to a sub-query. For example, for the sample query \( Q_1 \), we have two result lists, \( R^\text{co} \) and \( R^\text{scas} \), one for each of its sub-queries. \( R^\text{co} = \{(v_4, s_4), (v_5, s_5), (v_6, s_6), (v_7, s_7), (v_8, s_8), (v_{10}, s_{10}), (v_{11}, s_{11})\} \) and \( R^\text{scas} = \{(v_1, s_1), (v_{12}, s_{12})\} \), where \( v_i \) denotes node \( i \) in Figure 1 and \( s_i \) is the score for \( v_i \).

Results from the SCAS retrieval are answers to one part of the original query and results from the CO retrieval are approximate answers to the remaining part of the original query. Thus, results from both retrievals can be combined to produce approximate answers to the original query. To do so, we focus on results from the CO retrieval because they are the nodes “matching” the original query’s target. For each CO result \( v_\text{co} \), let \( v_\text{scas} \) be a SCAS result such that \( v_\text{co} \) and \( v_\text{scas} \) are in the same document, then the relevancy of \( v_\text{co} \) to a query \( Q \), denoted as score \((v_\text{co}, Q)\), is:

\[
\text{score}(v_\text{co}, Q) = \text{f}(v_\text{co}, P(Q)) \ast \text{score}(v_\text{co}, Q^\text{co}) + \text{score}(v_\text{co}, Q^\text{scas})
\]

where \( \text{f}(v_\text{co}, P(Q)) \) is a target path similarity with a value between 0 and 1 that measures how well an XML node \( v_\text{co} \) satisfies the target path in \( Q \), i.e., \( P(Q) \).

For example, for the sample query \( Q_1 \), node 4 in Figure 1 is a result for its CO sub-query \( Q^\text{co} \). Node 1 in Figure 1 is a result for the SCAS sub-query \( Q^\text{scas} \), which is in the same document as Node 1. Thus, the relevancy of node 4 (i.e., \( v_4 \)) to \( Q_1 \) can be computed using (6). That is, score \((v_4, Q_1) = \text{f}(v_4, P(Q_1)) \ast \text{score}(v_4, Q^\text{co}) + \text{score}(v_4, Q^\text{scas}) \). The target path similarity, \( \text{f}(v_\text{co}, P(Q)) \), is the key in the combination step. If the label path of an XML node \( v_\text{co} \) is an exact match to \( P(Q) \), then \( \text{f}(v_\text{co}, P(Q)) = 1 \). It’s often the case that the label path of a CO retrieval result \( v_\text{co} \) may not be an exact match to a query target path. In such cases, we compute the target path similarity for a CO retrieval result \( v_\text{co} \) to be the maximum similarity between \( v_\text{co} \) and an XML target node \( v_\text{ts} \) where \( v_\text{ts} \) is an exact match to \( P(Q) \), denoted as \( \text{sim}(v_\text{co}, v_\text{ts}) \). That is,

\[
\text{f}(v_\text{co}, P(Q)) = \max \{ \text{sim}(v_\text{co}, v_\text{ts}) | v_\text{ts} \text{ is an exact match to } P(Q) \} \]

For example, the target path similarity for node 4 (i.e., \( v_4 \)) is the maximum of \( \text{sim}(v_4, v_9) \) and \( \text{sim}(v_4, v_8) \) since both nodes \( v_9 \) and \( v_8 \) match \( Q_1 \)’s target path exactly.

For a given query \( Q \) and an XML data tree \( D \), there are usually many nodes in \( D \) whose label paths match the target path in \( Q \) exactly. For example, there are about 65470 different nodes in the INEX collection that exactly match the target path in \( Q_1 \). Thus, to reduce computations, we cluster XML nodes in \( D \) with the same label paths into groups similar to DataGuides[7]. For example, Figure 2 is a group representation of the XML data tree in Figure 1. Each rectangle represents a group with its identifier and label next to the rectangle. The numbers inside each rectangle are the identifiers of the nodes in the rectangle.

![Figure 2: A group representation of the XML tree in Figure 1.](image-url)
For example, the target path similarity of node 4 (i.e., $v_4$) is $\text{sim}(g_{co}, g_4)$ since node $v_4$ is inside group $g_4$ and all the nodes that are exact matches to $P_1(\emptyset)$, i.e. node 6 and 8, are in group $g_6$. In the following, we introduce two methods to compute group similarities by considering groups’ path and content aspects.

### 3.3.1 Path similarity

The similarity between a group $g_{co}$ and a target group $g_{t}$, $\text{sim}(g_{co}, g_{t})$, can be computed based on the similarity between their corresponding label paths. Let $p_{gco}$ and $p_{gt}$ be the label path of group $g_{co}$ and $g_{t}$ respectively. The greater number of common prefix nodes these two paths share, the more similar the two groups are. Thus, $\text{sim}(g_{co}, g_{t})$ is:

$$\text{sim}(g_{co}, g_{t}) = \frac{|p_{gco} \cap p_{gt}|}{|p_{gco}| + |p_{gt}| - |p_{gco} \cap p_{gt}|}$$  \quad (9)

where $|p_{gco} \cap p_{gt}|$ represents the number of common prefixing nodes between $p_{gco}$ and $p_{gt}$; $|p_{gco}|$ and $|p_{gt}|$ denote the number of nodes on the paths $p_{gco}$ and $p_{gt}$ respectively. The denominator in (9) is used for the normalization purpose such that $\text{sim}(g_{co}, g_{t}) = 1$ when $g_{co} = g_{t}$.

### 3.3.2 Content similarity

For document-centric XML collections, the path similarity may not be very accurate in estimating group similarity. For example, given three paths $p_1$: /article/body/section/title, $p_2$: /article/body/section and $p_3$: /article/body/section /paragraph, $p_2$ is as similar to $p_1$ as $p_2$ to $p_3$ according to (9). If a user is looking for a section regarding specific content, then according to (9), a title will have the same target path similarity as a paragraph. Compared to a title, a paragraph, however, is a better approximation for a section. This is because the content of a paragraph is much closer to that of a section than the content of a title to that of a section.

This motivates us to measure the similarity between two groups based on their corresponding content. We describe the content of a group $g_{t}$ via a $N$-vector $g_{t} = (f_{t1}, f_{t2}, ..., f_{tN})$, where $N$ is the total number of distinct terms in an XML collection and $f_{tk} (1 \leq k \leq N)$ represents the frequency of term $f_{tk}$ in group $g_{t}$. With this vector representation of a group’s content, the content similarity between two groups, $g_{co}$ and $g_{t}$, can be estimated via the cosine of their corresponding content vectors:

$$\text{sim}(g_{co}, g_{t}) = \frac{g_{co} \cdot g_{t}}{\sqrt{g_{co} \cdot g_{co}} \cdot \sqrt{g_{t} \cdot g_{t}}}$$  \quad (10)

For example, using (10), we find that the similarity between a section group and a section’s title group in the INEX document collection is 0.4196, while the similarity between the section group and a section’s paragraph group is 0.991.

### 4. EXPERIMENTAL STUDIES

#### 4.1 Experimental Dataset

We use the INEX 04 dataset and all the 33 VCAS queries to evaluate the effectiveness of our VCAS retrieval methodology. The INEX 04 dataset, around 500MByte in size, consists of over 12,000 computer science articles from 21 IEEE Computer Society journals. The documents are marked with about 170 different tags. A document contains 1532 elements on average and an element has an average depth of 6.9.

#### 4.2 Test Runs

The following four runs are used to study the effectiveness of our VCAS retrieval methodology. All the experiments use the same node weight configurations: uniform weights. That is, $w(v) = 1$ for any node $v$ in the dataset.

- **CO run.** In this run, we ignore the structure conditions in a query and use the query’s content conditions to perform CO retrieval. This run is used as the baseline for testing the effectiveness of our VCAS retrieval methodology.

- **VCAS-1 run.** In this run, we perform the VCAS retrieval with $f=1$ for all results. The run is used as a base line to compare the effectiveness of the path similarity and content similarity metric.

- **VCAS-path run.** In this run, we perform the VCAS retrieval using the path similarity in (8) as the target path similarity.

- **VCAS-cont run.** In this run, we perform the VCAS retrieval using the content similarity in (9) as the target path similarity.

#### 4.3 Result Evaluation and Analysis

To evaluate the relevancy of an XML document component to a query topic, the relevance assessment working group in INEX has proposed a two-dimension relevancy metric (exhaustiveness, specificity). Exhaustiveness measures the extent to which the document component discusses the topic of request and specificity measures the extent to which the document component focuses on the topic of request. This two-dimension metric is then quantized to a single relevancy value between 0 and 1. In this paper, we use two of the most frequently used quantization methods: strict and generalized. A relevancy value is either 0 or 1 with a strict quantization; while it could be 0, 0.25, 0.5, 0.75 or 1 with a generalized quantization.

In our experiments, we use the INEX relevance assessment set version 3.0 and compute each run’s mean average precision (MAP) using INEX on-line evaluation tools. Table 1 presents mean average precisions over all of the 33 query topics using both strict and generalized quantization methods. The corresponding ranks compared to all the 51 official submissions returned by other INEX participating systems are also included.

<table>
<thead>
<tr>
<th>Run</th>
<th>MAP</th>
<th>Rank</th>
<th>Generalized MAP</th>
<th>Generalized Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO</td>
<td>0.064</td>
<td>11</td>
<td>0.0716</td>
<td>7</td>
</tr>
<tr>
<td>VCAS-1</td>
<td>0.0844 (+31.88%)</td>
<td>5</td>
<td>0.0878 (+22.63%)</td>
<td>5</td>
</tr>
<tr>
<td>VCAS-path</td>
<td>0.0886 (+38.44%)</td>
<td>4</td>
<td>0.0887 (+23.88%)</td>
<td>5</td>
</tr>
<tr>
<td>VCAS-cont</td>
<td>0.0946 (+47.81%)</td>
<td>4</td>
<td>0.094 (+31.28%)</td>
<td>5</td>
</tr>
</tbody>
</table>
From Table 1, we note that our VCAS retrieval approach significantly outperforms the CO approach. The VCAS-1 run outperforms the CO run by 31.88% using the strict quantization metric. This is because the CO approach ignores XML structures for simplicity but loses the precision benefit provided by XML structures. Further, by comparing the VCAS-1 run with the VCAS-path and VCAS-cont runs, we note that similarity measures further improve our VCAS retrieval precision. Also, the content similarity provides more precision improvement than the path similarity for the INEX VCAS retrieval task. We note that the mean average precisions of our VCAS retrieval approach are relatively high compared to all the 51 official INEX submissions. For example, the mean average precision of the VCAS-cont run ranks top 4 (5) using the strict (generalized) quantization method. We have also observed similar results using other quantization methods.

5. RELATED WORKS
There is a large body of work on XML information retrieval (e.g.,[3-6, 8-13]), most of which focuses on effective XML CO retrieval and SCAS retrieval. For example, Sigurbjörnsson et al propose a general methodology for processing content-oriented XPath queries [11]. The key difference between [11] and our methodology is that: [11] focuses on extending IR engines designed for CO retrieval to support SCAS retrieval; while our methodology extends XML engines designed for CO and SCAS retrievals to support VCAS retrieval.

XML VCAS retrieval is a new task in INEX 04. Many teams within the INEX initiative conducted VCAS retrievals by ignoring the query structure conditions (e.g., [12]). In [9], S. Geva proposed a VCAS retrieval approach by decomposing a query into multiple sub-queries, where each sub-query contains one structure filter and one content filter. An XML element is a result for a sub-query if it satisfies the content filter, but does not necessarily have to satisfy the structure filter. Results from different sub-queries are merged and sorted by the number of filters they satisfy. This approach is simple and effective. Our work differs [9] in two aspects: the query decomposition strategies are different; and two similarity metrics are proposed to measure the relevancy of a VCAS result to a query target path for improving retrieval precision. No such measure is used in [9].

Query relaxation is also related with XML VCAS retrieval. S. Amer-Yahia et al have some seminal studies on XML query relaxation in [1, 2]. They model a XML query as a tree and relax node and/or edge constraints on the query tree to derive approximate answers. Algorithms have been proposed to efficiently derive top-k approximate answers. Our work differs from [1, 2] in that while they focus more on the efficiency aspect, we focus on the effectiveness (i.e., retrieval precision) aspect.

6. CONCLUSION
In this paper, we propose an approach for processing XML vague content and structure (VCAS) retrieval. A content and structure (CAS) query consists of two parts, i.e., support and target, where each part contains both path and content conditions. To derive approximate answers to a query, we decompose a query into two sub-queries: one sub-query consisting of support path and content conditions (a SCAS sub-query) and another sub-query consisting of the target content condition (a CO sub-query). We then process the SCAS sub-query by SCAS retrieval and the CO sub-query by CO retrieval. Results from both retrievals are combined to produce approximate results to the original query. To improve retrieval precision, we adjust the score of a CO retrieval result by the relevancy of the result to the target path condition of the original query, which is measured by target path similarity. We propose a path similarity and a content similarity metric to compute the target path similarity. We evaluate our VCAS retrieval approach and the similarity metrics through extensive experiments on the INEX 04 dataset and all the 33 VCAS queries. Our experimental results demonstrated that: 1) our VCAS retrieval approach, by taking advantage of XML structures, significantly outperforms the content-only approach; and 2) the path and content similarity metrics are effective in estimating the relevance of CO sub-query results to a query target path constraint. Therefore, they can be used to further improve the accuracy of the ranking of the retrieved results.

REFERENCES