Processing Structural Constraints

Andrew Trotman
Department of Computer Science
University of Otago
Dunedin
New Zealand

SYNONYMS

None

DEFINITION

When searching unstructured plain-text the user is limited in the expressive power of their query – they can only ask for documents that are about something. When structure is present in the document, and with a query language that supports its use, the user is able to write far more precise queries. For example, searching for smith in a document is not necessarily equivalent to searching for smith as an author of a document. This increase in expressive power should lead to an increase in precision with no loss in recall. By specifying that smith should be the author, all those instances where smith was the profession will be dropped (increasing precision), while all those in which smith is the author will still be found (maintaining recall).

HISTORICAL BACKGROUND

With the proliferation of structured and semi-structured markup languages such as SGML and XML came the possibility of unifying database and information retrieval. The Evaluation of XML Retrieval (INEX) was founded in 2002 to examine the use of semi-structured data for both tasks. It was expected that the use of structure would not only unify the two technologies but would also improve the performance of both.

SCIENTIFIC FUNDAMENTALS

User Querying Behavior

When using an information retrieval search engine the user typically has some information need. This information need is expressed by the user as a keyword query. There are many different queries that could be drawn from the same information need. Some might contain only keywords, others phrases, and others a combination of the two. It is the task of the search engine to satisfy the information need given the query. Because the task is not to satisfy the query, the terms in the query can be considered nothing more than hints by the user on how to identify relevant documents. It is likely that some relevant documents will not contain the user's keywords, while others that do might not be relevant.

If the user does not immediately find an answer they will often change their query, perhaps by adding different keywords, or by removing keywords. If the query syntax permits they might add emphasis markers (plus and minus) to some terms.

With a few exceptions semi-structured search engines remain experimental, so user behavior cannot be studied in a natural environment. Instead the user behavior is expected to mirror that of other search engines, or search engines that include some structural restriction.

The model used at INEX is that a user will give a query containing only search terms, then, if they are dissatisfied with the results, they might add structural constraints to their query. The keyword searches are known as Content Only (CO) and when structure is added as Content Only + Structure (CO+S) or Content And Structure (CAS) queries. Just as the keywords are hints, so too are the structural constraints. For this reason they are commonly referred to as structural hints.

The addition of structure to an otherwise content only search leads to a direct comparison of the performance of a search engine before and after the structural constraint has been added. The two queries are instantiations of the same information need, so the same documents or document components are relevant to each query making a direct comparison meaningful.

The analysis of runs submitted to INEX 2005 (against the IEEE document collection) showed no statistical difference in performance between the top CO and top CO+S runs – having structural hints in the query did not improve performance [12]. Even at low levels of recall (1% and 10%) no significant improvement was seen. About half the systems showed a performance gain, the other half no gain.

There are several reasons why improvements are not seen: first it could be a consequence of the structure present in the IEEE collection; second (and more likely) it could be that users are not proficient at providing structural hints.

The result was backed up by a user study [15] in which users were presented with three ways of querying the document collection: keywords, natural language (including structure), and Bricks [14] (a graphical user interface). 16 users each performed 6 simulated work tasks, 2 with each interface. The same conclusion is drawn, no significant improvement was seen when structure was used in querying.

Structural Constraints

There are two reasons a user might add structural constraints to a query. The first is to constrain the size of the result. When searching a collection of textbooks it is, perhaps, of little practical use to identify a book that satisfies the user need. A better result might be a chapter from the book, or a section from the chapter, or even a single paragraph. One way to identify the best granularity of result is to allow the user to specify this as part of their query. These elements are known as target elements.

The user may also wish to narrow the search to just those parts of a document they know to be appropriate. In this case the user might search for smith as an author in order to disambiguate the use from that as any of: an author, a profession, a street, or a food manufacturer. Restricting a query to a given element does not affect the granularity of the result; instead it lends support on where to look so such elements are known as support elements. Both target elements and support elements can appear in the same query.

It is not at all obvious from a query whether or not the user expects the constraint to be interpreted precisely (strictly) or imprecisely (vaguely). In the case of smith as an author, it is likely that smith as a profession is inappropriate, but smith as an editor might be appropriate. If

the target element is a paragraph, then a document abstract (about the size of a paragraph) is likely to be appropriate, but a book not so.

The four possible interpretations of a query were examined at INEX 2005 [11]. Runs that perform well with one interpretation of the target elements do so regardless of the interpretation of the support elements. The interpretation of the target element does, however, matter. The consequence is that the search engine needs to know, as part of the query, whether a strict or vague interpretation of the target element is expected by the user.

Processing Structural Constraints

Given a CO search engine, the strict interpretation of target elements can be satisfied by a simple post-process eliminating all results that do not match. As just discussed above, strictly processing support elements has been shown to be unnecessary.

Several techniques for vaguely satisfying target element constraints have been examined including ignoring them, pre-generating a set of tag-equivalences, boosting the score of elements that match the target element, and propagating scores up the document tree.

Ignoring Structural Constraints

Structural constraints might be removed from the query altogether and a Content Only (CO) search engine used to identify the correct granularity of result.

Tag Equivalence

A straightforward method for vaguely processing structural constraints is tag equivalence. A set of informational groups are chosen *a priori* and all tags in the DTD are mapped to these groups. If, for example, `<p>` is used for paragraphs and `<ip>` is used for initial paragraphs, these would be grouped into a single `paragraph` group.

Mass & Mandelbrod [5] *a priori* choose appropriate retrieval units (target elements) for the document collection and build a separate index for each. The decision about which units these are is made by a human before indexing. A separate index is built for each unit and the search is run in parallel on each index. Within each index the traditional vector space model in used for ranking. The lexicon of their inverted index contains term and context (path) information making strict evaluation possible. Vague evaluation of paths is done by matching lexicon term contexts against a tag-equivalence list.

Mihajlović *et al*. [6] build their tag equivalence lists using two methods, both based on prior knowledge of relevance. For INEX 2005 they build the first list by taking the results from INEX 2004 and selecting the most frequent highly and fairly relevant elements and adding the most frequently seen elements from the queries. In the second method they take the relevant elements from previous queries targeting the same element and normalize a weight by the frequency of the element in the previous result set (the training data, in this case INEX 2004). Using this second method they automatically construct many different tag equivalence sets using the different levels of relevance seen in the training data.

In a heterogeneous environment in which many different tags from many different DTDs are semantically but not syntactically identical techniques from research into schema-matching [1] might be used to automatically identify tag equivalence lists.

Structure Boosting

Van Zwol [13] generates a set of results ignoring structural constraints then boosts the score of those that do match the constraints by linearly scaling by some tag specific constant. The consequence is to boost the score of elements that match the structural constraints while not removing those that do not. A score penalty is also used for deep and frequent tags in the expectation of lowering the score of highly frequent (and short) tags. A similar technique is used by Theobald et al. [9] who use it with score propagation.

Score Propagation

Scores for elements at the leaves of the document tree (that is, the text) are computed from their content. Scores for nodes internal to the document tree are computed from the leaves by propagating scores up the tree until finally a score for the root is computed. Typically as the score propagates further up the tree its contribution to the score of an ancestor node is reduced (see the entry on *Propagation* for details).

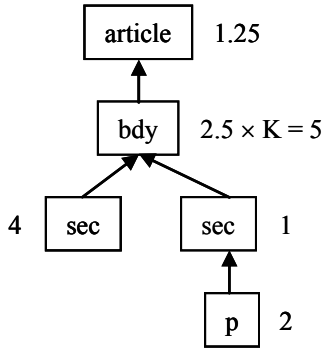

Figure 1: Score Propagation, each time a score is propagated the score is weakened (in the example: halved), but at target nodes it is boosted (in the example: K=2).

Figure 1 illustrated score propagation. A search term is found to occur two times in the `p` element, and four times in the (left) `sec` element. With a decay factor of 0.5, the score of the `bdy` element is computed as 4 *0.5 + 2 * 0.5 * 0.5 = 2.5. The score for the article element is computed from that score likewise. If the target element is `bdy` and the score is boosted by *K*=2, then the element with the highest score is that element. The score for K and the propagation value are chosen here for illustrative purposes only and should be computed appropriately for a given document collection.

Hurbert [3] uses score propagation with structure reduction - if a node in the tree does not match a constraint in the query then the score there is reduced by some factor. In this way all nodes in the tree obtain scores but those matching the constraints are over-selected for. Sauvagnat *et al*. [8] use score propagation in a similar way but in combination with tag equivalence.

KEY APPLICATIONS

Retrieval of document components from structured and semi-structured document collections.

FUTURE DIRECTIONS

The best performing search engines that interpret structural constraints have not yet significantly outperformed those that ignore them. Several reasons have been forwarded.

There is evidence to suggest specifying a structural constraint is difficult for a user. Studies into the use of structure in INEX queries suggest that even expert users, when asked to give structured queries, give simple queries [7]. This is inline with studies that show virtually no use of advanced search facilities on the web.

Structure aware search engines are not as mature as web search engines and as yet the best way to use structural constraints (when present in a query) is unknown. The annual INEX workshop provides a forum for testing and presenting new methods.

Improvements were not seen when the IEEE document collection was used, but this result may not generalize to all collections. Alternative collections including newspapers, radio broadcast, and television, have been suggested [7; 10]. In 2006 INEX switched to using the Wikipedia as the primary test collection for *ad hoc* retrieval, but a comparative study on that collection has not yet been conducted.

Relevance feedback including structural constraints has been examined. Users might provide feedback on both the desired content and the preferred target element, or just one of these. Evidence suggests that including structure in relevance feedback does improve precision.

EXPERIMENTAL RESULTS

Evidence that use of structure increases precision is tentative. In the XML search engine of Kamps *et al*. [4], no significant difference is seen overall, however significant differences are seen at early recall points (the first few tens of documents). The search engine of Geva [2] recently performed better without structure than with.

At INEX 2005 a comparative analysis of performance with and without structural constraints on the same set of information needs was performed [12]. The best structure run was compared to the best non-structure run and no significant difference was found. Not even a significant difference at early recall points was found. On a system by system basis about half the search engines show a performance increase.

CROSS REFERENCES

Content-Only Queries,
Content-And-Structure Queries,
Evaluation Initiative For XML Retrieval (INEX),
MAP (Mean Average Precision),
Narrowed Extended XPath I (NEXI),
Propagation,
Semi-Structured Text,
Text Indexing & Retrieval,
XML

RECOMMENDED READING

[1] Doan, A., & Halevy, A. Y. (2005). Semantic integration research in the database community: A brief survey. *AI Magazine,* 26(1):83-94.

[2] Geva, S. (2006). GPX - gardens point XML IR at INEX 2006. In *Proceedings of the INEX 2006 Workshop*, 137-150.

[3] Hubert, G. (2005). XML retrieval based on direct contribution of query components. In *Proceedings of the INEX 2005 Workshop*, 172-186.

[4] Kamps, J., Marx, M., Rijke, M. d., & Sigurbjörnsson, B. (2006). Articulating information needs in XML query languages. *Transactions on Information Systems,* 24(4):407-436.

[5] Mass, Y., & Mandelbrod, M. (2005). Using the INEX environment as a test bed for various user models for XML retrieval. In *Proceedings of the INEX 2005 Workshop*, 187-195.

[6] Mihajlovic, V., Ramírez, G., Westerveld, T., Hiemstra, D., Blok, H. E., & de Vries, A. P. (2005). Vtijah scratches INEX 2005: Vague element selection, image search, overlap, and relevance feedback. In *Proceedings of the INEX 2005 Workshop*, 72 –87.

[7] O'Keefe, R. A. (2004). If INEX is the answer, what is the question? In *Proceedings of the INEX 2004 Workshop*, 54-59.

[8] Sauvagnat, K., Hlaoua, L., & Boughanem, M. (2005). Xfirm at INEX 2005: Ad-hoc and relevance feedback tracks. In *Proceedings of the INEX 2005 Workshop*, 88-103.

[9] Theobald, M., Schenkel, R., & Weikum, G. (2005). Topx and xxl at INEX 2005. In *Proceedings of the INEX 2005 Workshop*, 282-295.

[10] Trotman, A. (2005). Wanted: Element retrieval users. In *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology, Second Edition*, 63-69.

[11] Trotman, A., & Lalmas, M. (2006). Strict and vague interpretation of XML-retrieval queries. In *Proceedings of the 29th ACM SIGIR Conference on Information Retrieval*, 709-710.

[12] Trotman, A., & Lalmas, M. (2006). Why structural hints in queries do not help XML retrieval. In *Proceedings of the 29th ACM SIGIR Conference on Information Retrieval*, 711-712.

[13] van Zwol, R. (2005). B^3-sdr and effective use of structural hints. In *Proceedings of the INEX 2005 Workshop*, 146-160.

[14] van Zwol, R., Baas, J., van Oostendorp, H., & Wiering, F. (2006). Bricks: The building blocks to tackle query formulation in structured document retrieval. In *Proceedings of the 28th European Conference on Information Retrieval (ECIR 2006)*, 314-325.

[15] Woodley, A., Geva, S., & Edwards, S. L. (2007). Comparing XML-IR query formation interfaces. *Australian Journal of Intelligent Information Processing Systems,* 9(2):64-71.