

Propagation-based structured text retrieval

Karen Pinel-Sauvagnat
IRIT-SIG, 118 route de Narbonne,
31062 Toulouse Cedex 4, FRANCE
sauvagnat@irit.fr

Encyclopedia of Database Systems, Springer, 2009, pages 2197-2201

SYNONYMS

Relevance propagation; Score propagation

DEFINITION

Evaluate the relevance score of text components. Approaches using propagation view the logical structure of a structured document as a tree whose nodes are components of the document and whose edges represent the relationships between the connected nodes. A document component can be either a leaf or an inner node. Leaf nodes are document components that correspond to the last elements of hierarchical relationship chains and that contain raw data (textual information). With propagation, relevance scores are first calculated for leaf components. They are then propagated upwards in the document tree structure to calculate relevance scores for the inner components.

HISTORICAL BACKGROUND

With appropriate query languages, users may want to exploit the structure of structured text documents to perform fine-grained and flexible retrieval. Instead of treating documents as atomic units, structured text retrieval systems aim thus at retrieving document components that answer a given information need in the most specific way. Classical Information Retrieval (IR) models (e.g. the vector space model, the proba-

bilistic model, language models) have been extended in order to take into account the structural dimension of documents. IR research has shown that document term weighting is a crucial concept for effective retrieval, thus classical formalisms were adapted with additional weighting parameters: component type, number of descendant components, frequency of the component type, to name a few. However, most of these adaptations do not use (or make little use of) the tree representation of structured documents to identify the most specific components.

The idea behind relevance propagation method is to follow the way relevance changes in a document tree to estimate the relevance of the document components. Indeed relevance in structured text documents has been expressed in terms of specificity and exhaustivity: the specificity (its coverage of the topic and nothing else) of components in the tree typically decreases as one moves up the tree, and when a component has multiple relevant descendants, its exhaustivity (coverage of the topic) usually increases compared to that of each of the descendant elements.

In propagation methods, relevance scores are thus first computed at leaf level, and then propagated up the document tree: to allow the identification of the most specific components, the relevance score of leaf components should be somehow decreased while being propagated upwards in the document tree, and to allow the identification of the most exhaustive components, relevance scores may be aggregated (a parent score should be evaluated using its children

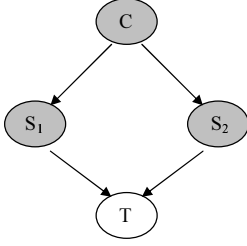


Figure 1: A simple network

scores). A naive solution would be to just sum the relevance scores of each inner component relevant children. However, this would ultimately result in root components being returned at top ranks although they may not be the most specific components for the given information need.

Few relevance propagation approaches were proposed before 2002 (when INEX, the evaluation Initiative for XML Retrieval, was set-up). One can however cite the method presented in [1] using inference nets. The retrieval process is applied to SGML documents but can be extended to any type of structured documents. The basic retrieval strategy is to calculate the degree to which a component at any level of the document hierarchy satisfies the query by considering what components are contained in that component. This strategy makes it possible to systematically calculate the expected relevance of a component at any level, taking into account its relationship with other components in the hierarchy. Let us consider the following net made of two section components S_1 and S_2 that are the parent node of a term node T and the children nodes of the component C (see figure 1).

The retrieval process is performed in a bottom-up fashion, because of the way documents are represented in the inference net: no components other than leaf components contain actual text, and the retrieval process must start from leaf components whose text contains a query term.

The degree of belief of T given the network topology is computed with a simplified formula as follows (the simplification comes from considering only positive

events):

$$\begin{aligned}
 B(T|C) &= P(T|S_1) * P(S_1) + P(T|S_2) * P(S_2) \\
 &= P(T|S_1) * P(S_1|C) * P(C) \\
 (1) \quad &+ P(T|S_2) * P(S_2|C) * P(C)
 \end{aligned}$$

Children components S_i and their parent C are represented as $S_i = \langle s_{i1}, s_{i2}, \dots, s_{in} \rangle$ and $C = \langle c_1, c_2, \dots, c_n \rangle$ respectively, where n is the number of index terms and s_{ij} and c_i are calculated using standard term frequency (TF) and inverse document frequency (IDF) statistics. The probability $P(S_i|C)$ of observing S_i given C (or the degree of belief that C supports S_i) is calculated as a similarity between the two vectors:

$$(2) \quad P(S_i|C) = \lambda_i * (S_i \cdot C)$$

where λ_i is a weight associated to the component type, representing its overall importance relative to other types of components sharing the same parent. This computation incorporates both the content and the type of components.

The probability $P(T|S_i)$ of observing a term T given a component S_i (or the degree of belief that S_i supports T) is estimated with:

$$(3) \quad P(T|S_i) \cong IDF_T * TF_{i,S_i}$$

$P(C)$ is the probability of observing C assuming that C is the root node (i.e. document) in the inference net. In the implementation, it is set to 1. This approach was however not evaluated, since no suitable test collection was available.

Other approaches presented in the rest of the entry were developed and evaluated in the context of the INEX evaluation campaign, which is concerned with the evaluation of XML retrieval. In this case, document components correspond to XML elements.

SCIENTIFIC FUNDAMENTALS

As all IR approaches, propagation-based approaches can be characterized in terms of indexing and scoring methods (here both for leaf and inner elements).

Indexing

Approaches using relevance propagation consider indexing units as disjoint units: the text of each element is the union of one or more of its disjoint parts. Thus, as textual information is only present in leaf elements, inverted index only concern leaf elements. Propagation accounts then for the fact that the text in a given leaf element is also contained in its ancestors. The document structure is generally stored in a separate index and used to build the document trees.

Relevance scores evaluation for content-only queries

In the rest of the entry, the following notations are used. Let q be a query composed of k terms t_1, \dots, t_k . In the document tree, le is a leaf element and e an inner element having n children. $RSV(q, le)$ is the relevance score of the leaf element le with respect to query q and $RSV(q, e)$ is the final score of element e with respect to query q .

Scoring leaf elements

To evaluate leaf element scores ($RSV(q, le)$), approaches found in the literature have used the following parameters:

1. frequency of term t_i in query q or leaf element le
2. frequency of term t_i in the whole collection
3. number of leaf elements containing t_i
4. length of leaf element
5. average length of leaf elements
6. inverse element frequency ief , which is similar to idf (inverse document frequency) but that takes into account the collection of leaf elements instead of the collection of documents.

In [2], the weighting scheme used to compute the relevance score of leaf elements also uses the cross-structural importance of t_i relative to le and q , which allows to increase or decrease the importance of a term depending on its location in the query (some terms may be more important than others in

a content-and-structure query). In [3], the frequency of terms in the leaf elements and in the whole collection are used with a parameter scaling up the score of elements having multiple query terms. The formula penalizes elements with frequently occurring query terms (frequent in the collection), and rewards elements with more unique query terms within a result element. One can also cite the leaf elements weighting scheme presented in [5], where term frequency and inverse element frequency ief are applied together with idf . This allows to take into account the importance of terms in both the collection of leaf elements and the collection of documents.

Propagating relevance scores

Once the relevance score of the leaf elements has been calculated, they are used to evaluate the relevance score of the inner elements. The main issue here is how to combine these scores. As already said, a naive solution that simply sums the relevance scores of leaf elements will result in root elements being ranked at the top of the result lists, although they are likely to not constitute the most specific elements to the query.

In [2] the relevance of inner elements is for instance evaluated using the maximum of their leaf element scores. Other approaches use a weighted sum of leaf or children element scores, and make some assumptions related to the document tree structure. They are described below.

For example, in the GPX approach [3], a heuristically derived formula is proposed to evaluate the scores of inner elements that accounts for specificity and exhaustivity:

$$(4) \quad RSV(q, e) = D(n) \sum_{l=1}^n RSV(q, l)$$

where n is the number of children elements, $D(n) = 0.49$ if $n=1$, 0.99 otherwise, and $RSV(q, l)$ is the relevance score of the l^{th} child element.

The value of the decay factor D depends on the *number of relevant children* that the inner element has. If the element has one relevant child then the decay constant is 0.49. An element with only one

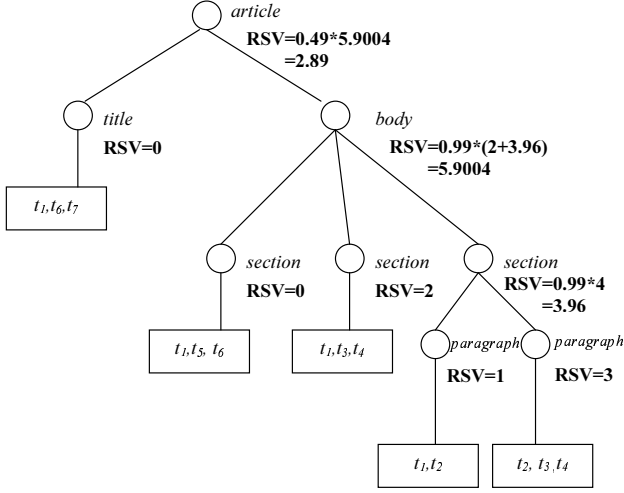


Figure 2: Relevance propagation according to [3]

relevant child will be ranked lower than its child. If the element has multiple relevant children the decay factor is 0.99. An element with many relevant children will be ranked higher than its descendants. Thus, a section with a single relevant paragraph would be judged less relevant than the paragraph itself, but a section with several relevant paragraphs will be ranked higher than any of the paragraphs. The method is illustrated in figure 2, with the content-only query ‘ $t_2 t_3 t_4$ ’. To simplify, the weight of a term in a leaf element is equal to 1 if it is a query term and 0 otherwise. For example the `/article[1]/body[1]/section[3]/paragraph[2]` element that contains 3 query terms has thus a score of 3. Its parent (`/article[1]/body[1]/section[3]`) contains two relevant children with scores 1 and 3. Its score is then calculated with a decay factor equal to 0.99 as follows: $0.99*(1+3)$. To evaluate the score of the root element (which only contains one relevant child), the decay factor used is 0.49, and the root element has consequently a lower score than its child `/article[1]/body[1]`. As a result of the propagation, the `/article[1]/body[1]` element has the highest score and will be ranked first by the GPX system.

The distance between an element and its descendant leaf elements has also been used as a weight in the weighted sum that calculates the relevance score of inner elements. Terms that occur close to the root of a given subtree are more significant to the root element than ones at deeper levels of the subtree. It seems therefore that the greater the distance of an element from its ancestor, the less it should contribute to the relevance of its ancestor. This can be modeled with the $d(x, y)$ parameter, which is the distance between elements x and y in the document tree, i.e. the number of arcs joining x and y .

In [4], the relevance score of an inner element takes into account this distance and also the distance separating the root element and leaf elements. The latter is used as a normalization factor. The relevance score of an inner element e is calculated as follows:

$$RSV(q, e) = \sum_{le_j \in L_e} \left(1 - 2\lambda \cdot \frac{d(e, le_j)}{d(e, le_j) + d(root, le_j)}\right)^2 \times RSV(q, le_j) \quad (5)$$

λ is a constant coefficient ≥ 0 , le_j are leaf elements being descendant of e , and L_e is the set of leaf elements being descendant of e . This process tends to consider an element having a relevant descendant element less relevant than the descendant element itself, which is comparable to the approach followed in [3], as above described.

Finally, in [5], the distance between elements and the number of relevant descendant leaf elements are used together with an additional parameter, β , in the weighted sum. The β factor captures the assumption that small elements may be used by authors to highlight important information (*title* elements, *bold* elements,...). *Small elements can therefore give important indications on the relevance of their ancestors* and their importance should be increased during propagation. The relevance value of an element e is thus computed according to the following formula:

$$RSV(q, e) = |L_e^r| \cdot \sum_{le_j \in L_e} \alpha^{d(e, le_j) - 1} * \beta(le_j) \times RSV(q, le_j) \quad (6)$$

where α is a constant coefficient $\in]0, 1]$ used to tune the importance of the distance $d(e, le_j)$ parameter and $|L_e^r|$ is the number of leaf elements being descendant of e and having a non-zero relevance value.

$\beta(le_j)$ is experimentally fixed and allows to increase the role of elements smaller than the average leaf element size in the propagation function.

[5] also propose a backward propagation after the first propagation, in order to account for the whole document relevance (and thus for the element context) in the calculation of the relevance score of inner elements (see the contextualization entry for other approaches using elements context).

Content-and-structure queries processing

In most of the approaches using relevance propagation, results elements are simply filtered to satisfy the structural constraints of content-and-structure queries [3, 4]. The approach described in [7] however uses relevance propagation to process structural constraints. Queries may have several constraints, and each constraint is composed of both a content and a structure condition, one of them indicating which type of elements should be returned (target elements). For each constraint, propagation starts from leaf nodes answering the content condition and goes until an element that matches the structure constraint is found. The score of result elements of each constraint is then propagated again to elements belonging to the set of targeted structures.

More details can be found in the Processing structural constraints entry.

KEY APPLICATIONS

The access to structured text documents such as SGML or XML documents is the main application of the relevance propagation approach described in this entry. Propagation can also be used to access HTML documents in the context of web retrieval, for example to determine among a set of related web pages, which one corresponds to the best entry in the set. Moreover, propagation can also be applied to distributed IR, as presented in [8].

FUTURE DIRECTIONS

Propagation methods presented in this entry are relatively independent of the DTDs of collections, since most of them do not use the type of elements to estimate relevance. However, propagation cannot be done in the same way on small document collections and large document collections. Methods should be adapted to process large document collections and efficiency issues that follow.

Relevance propagation may evolve in the future with the use of another source of evidence to calculate inner element relevance score, for example, link information. Indeed, web approaches using relevance propagation have also used links to find relevant web pages (a relevant page may be linked to other relevant pages) [9]. The same approaches could be used for structured text retrieval.

EXPERIMENTAL RESULTS

Good performances were achieved with relevance propagation methods in the INEX evaluation campaign. For example, the approach in [3] was ranked in the top 5 for the focused retrieval task (which aims at targeting the appropriate level of granularity of relevant content that should be returned to the user for a given topic) and comparable results were achieved by the approach described in [5] using content-and-structure queries.

CROSS REFERENCE

AGGREGATION-BASED STRUCTURED TEXT RETRIEVAL

INEX

RELEVANCE

SPECIFICITY

RECOMMENDED READING

- [1] Myaeng S.-H., Jang D.-H., Kim M.-S., Zhoo Z.-C. (1998): A Flexible Model for Retrieval of SGML Documents. Proceedings of SIGIR 1998, Melbourne, Australia: 138-145 SIGIR 1008
- [2] Anh V. N., Moffat A. (2002): Compression and an IR approach to XML Retrieval. INEX 2002 Workshop Proceedings, Dagstuhl, Germany.
- [3] Geva S. (2005): GPX - Gardens Point XML IR at INEX 2005. INEX 2005 Workshop Proceedings, Dagstuhl, Germany: 240-253.
- [4] Hubert G. (2005): XML retrieval based on direct contribution of query components. INEX 2005 Workshop Proceedings, Dagstuhl, Germany: 172-186.
- [5] Sauvagnat K., Hlaoua L., Boughanem M. (2005): XFIRM at INEX 2005: adhoc and relevance feedback tracks. INEX 2005 Workshop Proceedings, Dagstuhl, Germany:88-103.
- [6] Ogilvie P., Callan, J. (2005): Parameter estimation for a simple hierarchical generative model from XML retrieval. INEX 2005 Workshop Proceedings, Dagstuhl, Germany: 211-224.
- [7] Sauvagnat K., Boughanem M., Chrisment C. (2006): Answering content-and-structure-based queries on XML documents using relevance propagation. Information Systems, Special Issue SPIRE 2004, 31:621–635.
- [8] Baumgarten C. (1997): A Probabilistic Model for Distributed Information Retrieval. Proceedings of SIGIR 1997, Philadelphia, U.S.A.: 258-266.