1998-12 (Partially updated on 2009-7)

Fine-grained Full-text Search

Yasusi Kanada

Central Research Laboratory, Hitachi Ltd.
Higashi-Koigakubo 1-280, Kokubunji, Tokyo 185, Japan
E-mail: kanada@crl.hitachi.co.jp

Keywords:

IR theory and models (general), passage retrieval, user interface (UI) design for IR (general).

Abstract

Most conventional text retrieval methods are designed to search for *documents*. However, users often do not require documents themselves, but are searching for specific *information* that may come from a large collection of texts quickly. To satisfy this need, we have developed a model and two methods for *fine-grained searching*. The unit of search in this model is called an atom, and it can be a sentence or smaller syntactic unit. A score, i.e., a relevance value, is defined for each atom and for each query, and the score is propagated between atoms. By using the two methods, excerpts from texts surrounding the search-result items and/or hyperlinks to the document parts that include the items are displayed. Multiple topics in a document can be separately listed in a search result. Evaluation of two prototypes, using a conventional full-text search engine as is or with only a small modification, has demonstrated that these methods are feasible and can decrease the search cost in terms of time and effort for users.

1 Introduction

Most conventional text retrieval methods are able to search for *documents*, but not for smaller units of text. Although document retrieval is sufficient for users who want to read entire documents, many users are only interested in specific *information* within documents rather than the documents themselves. Such users want to find needed or interesting information from a large collection of documents quickly. If a document contains multiple interesting topics, they should be listed separately in the search results for the benefit of such users. To satisfy these user needs, information retrieval methods must be developed that can

- Locate text parts, that contain information relevant to the user, and/or display excerpts from around the text parts.
- Separately list multiple topics within single documents.

We call this type of text search fine-grained searching.

A passage retrieval method can be the basis for a fine-grained searching function. In passage retrieval, a document is divided into smaller units, such as chapters, sections, paragraphs, and so on. The text units are characterized by a vector of term frequencies or similar quantities, and text units that have characteristics that match the query are obtained as the search-result items. Although passage retrieval can be used to retrieve text units that are smaller than documents, this approach has almost always been used to improve *document* retrieval performance [Woo 97].

There are three problems in using passage retrieval to retrieve fine-grained text units, such as words or sentences. First, statistics play an important role in passage retrieval, but they do not work well when the unit of search is too small because the statistical errors becomes excessive. Second, the text units must be fixed before beginning a retrieval. Thus, the text units may be too large or too small, or the text may be inappropriately divided. If the text unit can vary depending on the query, interesting topics may be listed separated in the search result even if the text units are not fine-grained. However, the topics cannot be separated well in passage retrievals. Third, although an ad hoc technique that can handle the relationships between text units can be introduced into a passage retrieval method, there is no unified framework to do so.

We have developed two search methods that satisfy the two needs stated above and that do not involve the above three problems. These methods are called fine-grained search methods, and have two main features.

The first feature is that a document is regarded as consisting of atoms, which are small text units, such as sentences, words, or characters. Atoms are the units of search. An item in a search result contains the text in an atom or the text surrounding the atom, and/or contains a hyperlink to the atom in the original text. So that part of the document or an excerpt that contains the atom can be displayed to the user. Also, if a topic is equal to or larger than an atom, multiple topics can be separated in the search result.

The second feature is that a score, which is a measure of the relevance to the query, is assigned to each atom, and the propagation mechanism of the value between atoms is introduced as the framework for describing the relationship between atoms.

In this paper, the model of fine-grained searching is described, then two methods for implementing the model are explained and compared, and results assigned when implementing and evaluating these methods are shown.

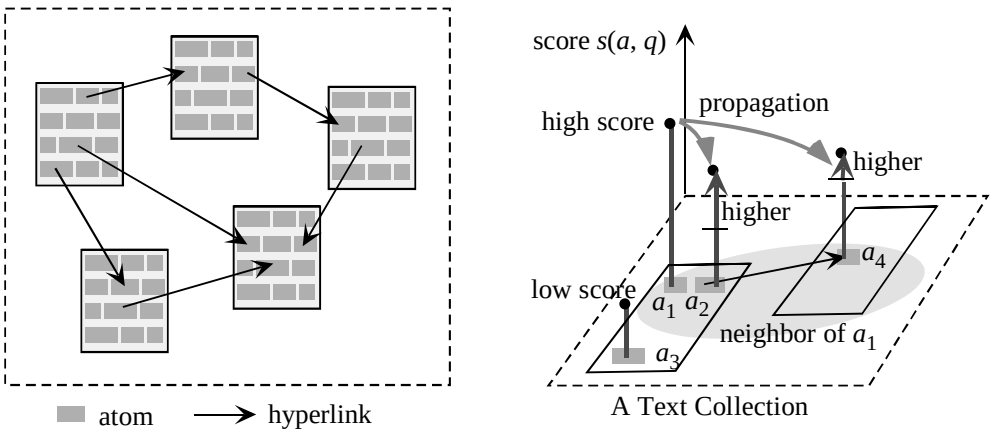
2 The Fine-grained Search Model

A model for fine-grained searching is explained in this section in the following order: (1) the most generalized model, (2) the basic function in a more specific model, in which input is limited as character strings, (3) the function of conjunctive and disjunctive searching and searching that reflects term frequency.

2.1 A generalized model

The fine-grained search model, at a generalized level, is explained using **Figure 1**. In this model, a text collection consists of documents and hyperlinks between document parts (Figure 1(a)). A *document* is a sequence of atoms. An *atom* may be a character, word, sentence or larger unit of text. A *hyperlink* is a directed edge between two atoms. To simplify the model, the anchor texts of the hyperlink, in the sense of the Web, are asserted to be an atom in this paper.[1] A *score*, i.e., a relevance value, is defined for each query and for each atom (Figure 1(b)). A score function $s(a, q)$ is thus defined for each atom $a \in$ A in the text collection and each possible query $q \in$ Q.

Text collections can be embedded into a continuous space. This means that the domain of the score function, Q×A, can be continuous. However, only the above discrete model, in which both Q and A are discrete, is explained and used in this paper.



(a) A text collection (b) Score and value propagation

Figure 1. Fine-grained text searching

The function of fine-grained text searching is explained using **Figure 2**. A search operation is to find a set of atoms $\{a_1, a_2, \ldots, a_n\}$ (or a set of atom sequences) that have scores higher than a

[1] If the anchor texts contain multiple atoms, a hyperlink must be defined as a directed edge between two sets of sequences of atoms. If the anchor texts are parts of an atom, this fine-grained search model should be redefined to be exact.

threshold. The search system inputs query q and outputs a list of search-result items, each of which corresponds to a_i (i = 1, 2, …, n). A search-result item contains a copy of the text in the atom or the text surrounding the atom. The item also contains a hyperlink to the atom in the original text.[1] This output is similar to the conceptual index [Woo 97]. The user can follow this hyperlink and see the text surrounding the atom or the whole document.

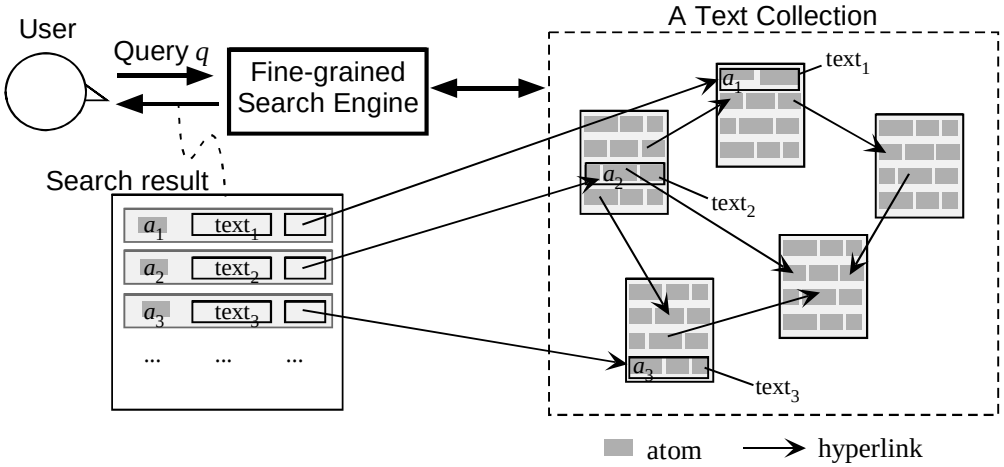
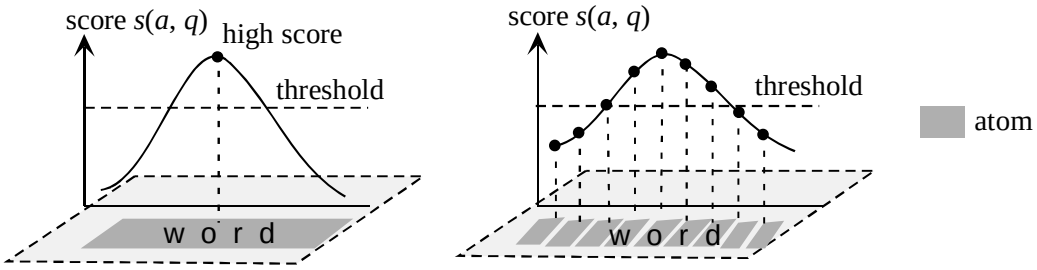


Figure 2. The function of fine-grained searching

2.2 A string-list-input model

From this section, a query is asserted to be defined by character strings to be searched for and possibly operators, such as AND or NOT. The syntax of queries is not specified and it can be more general in the abstract model described in the previous section.. If an atom or a sequence of atoms contains a specified character string (or a string close to the specified one), the score of the atoms is high (**Figure 3**). If an atom is a word or sentence and the specified string is a word, the atom can contain the string (see "word" in Figure 3(a)). If an atom is a character, it cannot contain a string that consists of multiple characters (see "word" in Figure 3(b)).



(a) Case 1: atom is larger (b) Case 2: search string is larger

Figure 3. Relative size of atoms and search string and scores

If the score of an atom is high, the scores of the neighboring atoms become higher in this model. This means that a score value is *propagated* to the syntactic neighbors (see a_2 in Figure 1(b)).

[1] It is possible to include only an excerpt or a hyperlink in the search-result item, instead of both.

Propagated values from different atoms are additive.[1] Several propagation methods are shown here. The first method is the *one-time propagation method*. Atoms a_{i1}, a_{i2}, … are the neighbors of a_i, and $s_0(a_i, q)$ is the score function without the propagation, then the score function with the propagation, $s(a_i, q)$, may be computed by

$$s(a_i, q) = s_0(a_i, q) + \sum_{i \neq k} c_{ik}\, s(a_{ik}, q) \quad \text{(a summation over all the neighbors of } a_i\text{)},$$

where c_{ik} $(0 \leq c_{ik} < 1,\ i = 1, 2, \ldots)$ are constants. The second method is the *relaxation method*. Scores are repeatedly propagated to nearest atoms until the scores converge. The third method is the *decay function method*. Score $s(a_i, q)$ is computed by

$$s(a_i, q) = \sum_{j \geq i} d(a_i, a_j)\, s_0(a_j, q) \quad \text{(a summation over all the atoms } a_1, a_2, a_3, \ldots\text{)},$$

where $d(a, a')$ $(0 \leq d(a, a') \leq 1$ and $d(a, a) = 1)$ is a decreasing function of distance between a and a'. This function is called a decay function.

If an atom is connected by a hyperlink from another atom, the former is regarded as a neighbor of the latter. The value is thus propagated by hyperlinks as well as through syntactic neighbors (see a_4 in Figure 1(b)).[2]

Due to the score propagation, a document can be regarded as a "smooth" object. This means that each atom in the document is in a text flow (of syntax or semantics) among atoms, and the neighboring atoms do not suddenly have very high or very low scores. In other words, although the text units are very small in the fine-grained search method, are not independent texts but they are related by the neighboring relationships or hyperlink relationships. This is in contrast to passage retrieval or conventional document retrieval in which the text units are basically independent of each other.

The most distinct feature of the fine-grained search method is that a unified means, i.e., propagation, is used to represent the relationships between text units. Other effects caused by score propagation are explained in the next section.

However, two problems may be caused by score propagation. The first problem is the *redundancy in search results*. The fine-grained search algorithm may output all the neighboring atoms of an atom that has a high score value (Figure 3(b)). This makes the search result very redundant because the user will often see the neighbor atoms by following the text or links from an atom. It would be better for the search algorithm to return only representative atoms.

The second problem is the *computation overhead*. A large amount of computation is necessary if the score value is propagated throughout a document or an entire set of linked documents. This is especially so when using the relaxation method. The model must be designed so as to keep the amount of computation under a reasonable limit.

2.3 Extended search function

In the fine-grained search model, functions similar to disjunctive searching (i.e., searching with OR), conjunctive searching (i.e., searching with AND), and searching that reflects term frequencies can be roughly simulated by using the score propagation, as explained below.

---

[1] Thus, theoretically, it is natural to represent the fine-grained search method by a neural net, in which atoms are represented by neurons.

[2] The neighboring relationship can be defined by the edges in syntactic trees or case structures.

- Simulated OR searching

  If multiple strings are specified in the query, the score of an atom or a sequence of atoms that contains at least one of the strings will be high. A disjunctive search can thus be simulated by simply specifying multiple strings.

- Simulated AND searching

  If two or more specified strings occur in an atom or a sequence of atoms, the score will be higher than that in single occurrence cases due to the additivity of propagation. Thus, if an appropriate propagation and score calculation method is introduced, and an appropriate threshold is set, only atoms that are close to all the search strings will be included in the search result. A conjunctive search can thus be roughly simulated. A similar result can be obtained by limiting the range of score propagation instead of adjusting the threshold. (An example is given later.)

- Searching that reflects term frequency

  When a specified string occurs several times in an atom or a sequence of atoms, the score is higher than when there is a single occurrence. This is also due to the additivity of propagation. The score of an atom in a document or a part of a document in which the term frequency (*tf*) is high will therefore be higher.

In conventional text retrieval, the score of a search-result item is often a function of the inverse document frequency (*idf*). However, in a fine-grained search, no such global information is included in the score, because the score propagation is a local effect.

The simulation of a conjunctive search is explained in more detail here. In a simulated conjunctive search, unlike a conventional conjunctive search in document retrieval, the score does not rise due to multiple strings appearing in a document if the strings are far apart. Thus, if multiple strings appear in a document and the score is high, the strings are close each other and probably related to the context.

The threshold must be specified so that the score value exceeds the threshold when multiple search strings occur within a limited distance. For example, the decay function may be defined as $\exp(-x^2/4)$, where the number of search strings n_s is asserted to be two to four, and the threshold is defined as $n_s - 1$. If all n_s strings occur in the same atom or neighboring atoms (i.e., the distance is zero or one), the score is above the threshold. If $n_s - 1$ strings occur in the same atom, the score is equal to $n_s - 1$, but is not above the threshold.

However, other conditions, such as the term frequency, may cause high scores. So an atom may be included in the search result even when the conjunctive condition is not satisfied. To realize crisp conjunctive searching without such events, a method different from the decay function or relaxation method may be introduced. In this method, the score propagation can be defined as follows. The score value of an atom is propagated only within the distance m (given with atoms as the unit). Only the score values of atoms that contain the search string become high. Then, it becomes possible to select only the atoms that satisfy the conjunctive conditions. Thus, the query $\mathrm{AND}_m(s_1, s_2, \ldots, s_n)$ means that all of strings, $s_1, s_2, \ldots, s_n$, occur in an existing sequence of atoms with length m. If this condition is satisfied for a sequence of atoms in the text collection, the scores of the atoms (after propagation) will be high.

2.4 Examples of fine-grained searching

As an example of fine-grained searching, a search result for the query AND_5(impressionist, music) (translated from Japanese) in the World Encyclopædia [HDH 98] is shown in **Table 1**. The result shown is a translated output of the search system that implements the atom-document search method explained in Section 3.2. In Table 1, as well as in the article titles (document titles) and the excerpts (copies of text that the atoms contain), the headings of the sections that contain each atom are shown. Hyperlinks to the original text are embedded in the excerpts. The heading column is empty

Table 1. The search result of query AND_5(impressionist, music) in the World Encyclopædia*

#	Article title	Heading	Excerpt	Score
1	Impressionism		The concept of impressionism is also used for **music**.	0.88
2	Impressionism	[Origin and pioneers]	Partially, a movement that leads to the **impressionist** movement can sometimes be seen at the end of the 18C.	0.88
3	Impressionism	[Impressionism and modern art in Japan]	It had a strong tendency toward expressionism, and it is the special condition of Japan, in which the introduction of **impressionist** was going to tie to avant-garde painting movement.	0.88
4	Impressionism	**[**Music**]**	**[Music]**	0.99
5	Impressionism	**[**Music**]**	who borrowed the concept of impressionism from paintings and applied it to **music**,	0.99
6	Impressionism	**[**Music**]**	To describe strictly in the **music** mode,	0.89
7	Symphonic poem		Debussy, as an **impressionist**, composed "Prélude à l'après-midi d'un Faune",	0.97
8	Symphonic poem		This genre, which was derived from the romantic **music**al thought,	0.97
9	Samuel Courtauld		He affected by his wife, who had a profound knowledge of **music** and art,	0.98
10	Samuel Courtauld		French **impressionist**s' works, including van Gogh's "Sunflowers", at the Tate Gallery in 1923,	0.99
11	Samuel Courtauld		He donated fund for purchasing paintings of Post-**Impressionists**.	0.99
12	Charles Camille Saint-Saëns		He did not reach really original **music**al expression,	0.98
13	Charles Camille Saint-Saëns		He established the National **Music** Society with R.Bussine, Faure, C. Franck, etc. (1871),	0.98
14	Charles Camille Saint-Saëns		He stood against Wagner and **impressionist**s as a writer.	0.98
15	French movie	[French impressionists and movie art movement]	Movie artists, who intended to create poems and **music** by images, became the mainstream.	0.98
16	French movie	[French impressionists and movie art movement]	The movie writers, who George Sador called "French **impressionist**s", brought forth an age.	0.98
17	Édouard Lalo		He greatly succeeded the first time, and was applauded by "the king of chair" as theater **music** (premiere on 1888),	0.98
18	Édouard Lalo		His style was quite different from Franck's school or the **impression-ist**s,	0.98

* All of the result was translated from Japanese.

if there is no heading except the article title. There were 18 search-result items, and six documents. Articles 2, 3, 9, 10, 11, 17, and 18 are considered to be non-relevant.[1]

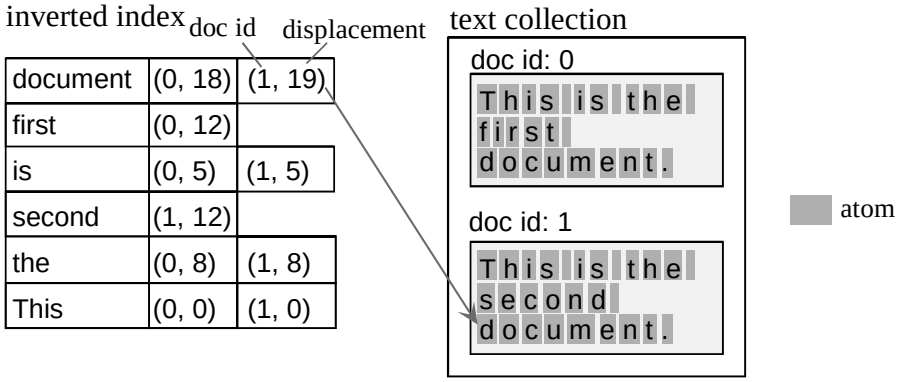
[1] The relevance of the search-result items was judged in an ad hoc manner here, and was not very objective. Items 14 and 15 in Table 1 implies that Saint-Saëns was an anti-impressionist, so he was regarded as being related to the impressionists, and the items were judged to be relevant. On the contrary, in items 18 and 19, it

3 Two methods

Two methods for fine-grained full-text searching are explained and compared in this section.

3.1 The atom-addressing search method

The first method, the atom-addressing search method, is explained using **Figure 4**. In this method, the address of each atom is defined. If the atom is a character, as shown in Figure 4(b), the address of atom a_i can be specified by a pair, (d_i, l_i), where d_i is the identifier of the document that contains the character and l_i is the displacement of the character from the beginning of the document (see Figure 4(a)). The displacement may be measured by the number of bytes, number of characters, or number of some other unit of text. If the entire text is included in a file, no document identifier is required. This means the address may be a displacement from the beginning of the file.



(a) Inverted index (b) Text collection

Figure 4. Inverted index and addresses of atoms for the atom-addressing method

A full-text search system generates and refers to an inverted index of words or characters. The index contains the locations of all the occurrences of words or characters (Figure 4(a)). The index can be designed to contain the addresses of atoms, as illustrated in Figure 4(a), or to be able to calculate them from the index. The full-text search algorithm must return the displacements, as well as the document identifiers.

A technique for implementing the atom-addressing method using a conventional full-text search system is explained here. Conventional full-text search engines only return document identifiers. Even if such an engine is used, the displacement of the atom can be obtained by scanning whole of the documents found. However, this scanning requires a very high cost in terms of machine time, it is complicated when multiple search strings are specified, and a considerable amount of computation is still required to use the scanned result. It is inefficient because the scanning and computation must be done during the search time instead of during the index-generation time. However, although the displacements of words or characters are not returned by the application programming interface (API), an inverted index usually holds the displacements, as shown in Figure 4(a), and,

---

is stated that Lalo's music is different in nature from that of the impressionists, so these items are not actively related to impressionists, and were judged to be non-relevant. However, these judgments are affected by the intention of the query, and, thus, are very delicate.

thus, we can easily design the search system to return displacements of atoms. This makes it easy to redesign or modify the search engine to return the positions of the atoms.

The scoring mechanism can be designed as follows. A decay function may be used for scoring. For example, if an atom is a character, function $d(x) = \max(0, 1 - 10^{-5}x^2)$ can be used for the decay function between two atoms in a document, where x is the distance measured by the number of characters. To reduce the computation overhead, only the beginning positions of strings that match the search strings are evaluated, and the propagation is computed only between them. To reduce the redundancy in search results, an elaborate method should be deviced.

3.2 The atom-document search method

The second method, the atom-document search method, is explained using **Figure 5**. In this method, each atom is regarded as a document (Figure 5(b)). A conventional full-text search algorithm, which returns document identifiers, is used. The inverted index contains a list of document identifiers of the atoms (Figure 5(a)). Only a string that is wholly included in an atom can be searched. Thus, the atom length must be larger or equal to the word length, as in Figure 5(b). If the atom is a character, only a single character could be searched, and the search system would be useless.

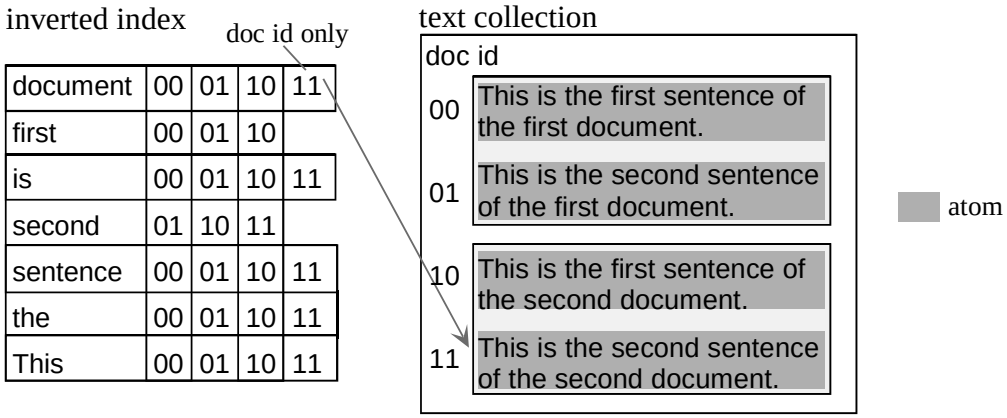


Figure 5. Inverted index and addresses of atoms for the atom-document method

A reasonable size for an atom is a sentence. The redundancy in search results is not excessive in this case. If a sentence is long, it is better to split the sentence into several atoms. If a sentence is an atom, though, the number of "documents" becomes huge. The performance may, thus, become very low with some algorithms. However, the performance of full-text search systems based on inverted indices does not decrease in principle.

The scoring method can be designed as follows. A decay function is used for scoring. An example of a decay function between two atoms (i.e., sentences here) in a document is $d(x) = 8 / (x + 8)$, where x is the distance measured by the number of sentences. To reduce the computation overhead, only the atoms contained in the search result are evaluated and propagation is computed only between them.

3.3 Comparison

The atom-addressing and atom-document search methods are compared here.

- Use of a conventional full-text search engine

  A conventional full-text search engine (API) that only returns document identifiers can be used for the atom-document method. However, it cannot be used for the atom-addressing method because it requires the locations of atoms in a document.

- Tolerance to text changes

  Even if only a space character is inserted into a document, all the index entries generated by the atom-addressing method after the space will be invalidated (if the spaces are counted when computing displacements) because the addresses will have shifted. This is especially so if the text is in ISO-2022-JP code, which is a code used for Japanese where there are characters that have the same meaning but a different size (one byte or two bytes). For example, "A" is a single-byte character and "Ａ" is a double-byte character. This often causes shifts in the displacements when the text format is changed. The task of maintaining an index may become heavy if the addresses are shifted. The atom-document method is more durable to format changes because the search result does not change regardless of the place where the terms occur unless the terms in the documents change.

- Ease of displaying excerpts

  As explained in Section 3.1, the text that is contained in the atom or that surrounds the atom is copied into the search-result item. An appropriate size for an atom in the atom-document method is a sentence, because a sentence is a unit that is bound to a meaning, and it is easy for the system to display the sentence in the atom as is. On the contrary, a smaller unit size that is not bounded to a meaning is used in the atom-addressing method. Thus, to display a text from which the user can obtain useful information, the system must extract and display an appropriate text fragment surrounding the atom. This may not be a very easy task.

The atom-document method seems to perform better in terms of these three criteria. However, the atom-addressing search method can still be a candidate.

4 Evaluation

Both the atom-addressing and atom-document methods have been implemented and evaluated. In this section, the atom-document method and conventional document full-text search method are compared by using a user cost model, and the actual performance of the fine-grained search method is evaluated. Although information retrieval methods are usually evaluated in terms of precision and recall, these properties are not considered here because it is semantically very subtle and difficult to objectively decide whether a text part is relevant when using the fine-grained search method.[1]

[1] The footnote in Section 2.4 pointed out that the relevance judgment is delicate and less objective. However, the relevance in a document retrieval can be judged by the relationship between the query and the document subject. Items 14, 15, 18, 19 in this example were judged as non-relevant, and such a delicate problem did not occur. This means the difficulty of relevance judgment is caused by searching multiple topics from a document.

4.1 Comparison with document full-text searching

In this subsection, a model of searches using the fine-grained search method is described. A model of the user cost (i.e., the time and effort required from the user for the search) is also described, and the cost using the fine-grained search and that using a conventional full-text search are compared.

The model of the searches is explained here. In this model, it is asserted that the user reads only small part from each document. If this hypothesis does not hold, i.e., if the user reads the entire document, this model does not apply. The user types search strings and gets a search-result list. Then it is asserted that the user reads all the atoms and their neighbors of all the occurrences of the search strings in the document collection. The cost of reading is estimated in this model. A search-result list contains the atom, or a link to the atom.

In a document search, it is asserted that the user scans the text from its beginning. Because the search string may occur several times in the text, the whole text is asserted to be scanned. If the document is short and the search string is highlighted in the result display, this scanning would not be necessary, and, thus, this hypothesis would not hold. However, if the document is long, a type of scanning, such as window scrolling, is required, and, if the search string is not highlighted or is weakly highlighted, text scanning is required even if the document is short. Therefore, the hypothesis that scanning is required is appropriate. If the user reads the text surrounding the atom and finds that the search-result item can satisfy the requirement, the user will probably read the text further. For example, the user may read the entire document. However, the cost of further investigation is not counted here.

The model of search cost is described below. The number of documents that occur in the search-result list is defined as n. The number of search strings that occur in the search-result list is defined as N. (If two or more search strings are specified, N is the summation of their occurrences.) The average cost (time) of reading the text surrounding an atom in the original text is defined as Cr. The cost of reading an item in a search-result list is defined as Cl. The cost of scanning the whole text, per atom, is defined as Ct.

The number of items in a search-result list is n in a conventional document search, so the cost of reading the whole list is $Cl\ n$. The number of atoms to be read is N. So the cost to read all of them is $Cr\ N$. If the number of atoms in document i is defined as a_i, then the number of atoms is $\sum_i a_i$. So the total cost for the scanning is $Ct\sum_i a_i$. The total cost Cd is expressed by

$$Cd = Cl\ n + Cr\ N + Ct\sum_i a_i .$$

The number of items on the search-result list is N in the fine-grained search, so the cost of reading the whole list is $Cl\ N$. If the user *always* reads the original text surrounding the atom, its cost will be $Cr\ N$, which is the same as the cost of the document search.[1] There is no need to scan the text to find the search string, so the total cost Cf consists of two terms:

$$Cf = (Cl + Cr)\ N.$$

The difference between the document search cost and the fine-grained search cost, ΔC, is expressed as

[1] The actual cost is less than $Cr\ N$ because a search-result item can be found to be non-relevant by reading only the text in the item. However, the cost is asserted to be $Cr\ N$ here for the sake of simplicity.

$$\Delta C = Cd - Cf = Ct\sum_i a_i - Cl\,(N - n).$$

Re is defined as $Re \equiv Ct/Cl$; the ratio of the cost to scan the text in an atom and the cost to read an item in the search-result list is defined as *Re*. The sufficient condition that the cost of the fine-grained search is lower, i.e., $\Delta C > 0$, is expressed as

$$Re > (N - n) / \sum_i a_i .$$

This condition is not defined when the search-result list is empty, i.e., $\sum_i a_i = 0$.

The CD-ROM World Encyclopædia [HDH 98], which is written in Japanese, was used as the corpus. Each encyclopedia article was regarded as a document in the document searches. The atom-document search method, in which an atom is usually a sentence, was used for the fine-grained search. A sentence was an atom if the sentence is short, but was divided at a comma if longer than 32 bytes (16 characters). The number of documents was 85,387, and the total number of sentences was 2,696,147. The average number of sentences per document was, thus, 31.6. The decay function $d(x) = 8 / (x + 8)$ was used for scoring. The search engine embedded in the World Encyclopædia was used for the document search. Thirty queries and the measured values of $(N - n) / \sum_i a_i$ are shown in **Table 2**. The set of queries consisted of twelve queries that contained only one search string, twelve queries that were conjunctions of two search strings, and six queries that were conjunctions of three search strings. All the strings were in Japanese. Some conjunct strings occurred adjacently, but others did not.

The value of *Re* will vary between user interfaces. In this experiment using the World Encyclopædia, *Ct* was 0.3-0.6 s, *Cl* was 2-4 s, and *Re* was 0.1-0.2.[1,2] Therefore, the values of $(N - n) / \sum_i a_i$ were smaller than *Re* in all cases, and the fine-grained search was found to cost less. However, in the queries in which the term frequency of the search strings is high (i.e., $\sum_i a_i / N$ is smaller), such as "AND_5(cocoa, chocolate)", the difference between the document search cost and the fine-grained search cost becomes smaller.

If *N* is much larger than *n*, the fine-grained search is less advantageous because the search-result list becomes lengthy. However, *N* is at most three times larger than *n* in this measurement.

4.2 Performance evaluation of the fine-grained searches

Both the atom-addressing and atom-document search methods were implemented and the basic performance measured. Both were an implementation of the axis-specified search [Kan 98a] [Kan 98b], which is a higher search function based on the fine-grained search. However, only the fine-grained search function of the implementation is used here. To compare the results, the performance of a full-text document search using the search engine embedded in the CD-ROM World Encyclopædia was also measured. The atom-addressing and atom-document searches are not compared here, because their implementation methods are very different.

---

[1] Although scrolling or paging takes time if the entire search-result list cannot be displayed in the window, its cost is not included in *Cl*.

[2] For example, the search strings were highlighted in coloring them by red in the CD-ROM World Encyclopædia. However, it is not always easy for a user to distinguish red characters from black characters. So the value of *Re* seems to become lower.

Table 2. The values of $(N - n) / \sum_i a_i$ in the World Encyclopædia search

Query	Number of document search results n	Number of search string occurrences in fine-grained search N*	$\sum_i a_i$	$(N - n) / \sum_i a_i$
Shikarami-soshi (しからみ草紙)	3	4	116	0.0086
Presley (プレスリー)	6	14	860	0.0093
Iridium (イリジウム)	28	40	2347	0.0068
Cocoa (ココア)	44	71	4352	0.0062
Yoshinobu (慶喜)	75	129	6893	0.0078
Chocolate (チョコレート)	84	127	11690	0.0037
Asano (浅野)	223	325	16041	0.0064
Computer (コンピュータ)	708	2015	100087	0.0131
Life (生命)	1386	2325	183998	0.0051
Tokugawa (徳川)	1613	2215	116928	0.0051
Philosophy (哲学)	2146	5065	177064	0.0165
America (アメリカ)	9785	22523	709151	0.0180
AND(Cocoa, Chocolate)**	7	39	375	0.0853
AND(Elvis, Presley)**	5	5	821	0
AND(Asano, Soichiro (総一郎))**	9	16	423	0.0165
AND(Asano, Naganori (長矩))**	12	44	546	0.0586
AND(Pre (プレ), Sley (スリー))**	45	49	2803	0.0014
AND(Tokugawa, Yoshinobu)**	58	129	5512	0.0129
AND(Art (アール), Nuvo (ヌーボー))**	69	132	7786	0.0081
AND(Computer, Communication (通信))**	155	516	20600	0.0175
AND(Life, Philosophy)**	190	225	10233	0.0034
AND(Tokugawa, Ieyasu)**	769	1337	53574	0.0106
AND(Japan (日本), America)**	4859	13315	377248	0.0224
AND(Ame, Rica)**	9807	23609	710462	0.0194
AND(Asano, Soichiro, Bank (銀行))**	4	7	79	0.0380
AND(Cocoa, Chokolate, Japan)**	4	13	128	0.0703
AND(Computer, Media (メディア), Economy (経済))**	22	3*	193	–0.0984
AND(Art, Nuvo, Painting (絵画))**	27	30	2439	0.0012
AND(Tokugawa, Ieyasu, Hidetada (秀忠))**	108	198	3560	0.0253
AND(Japan, America, Treaty)**	527	1262	38027	0.0193

* These ANDs are interpreted as AND_5 in the atom-document searches. That means that N is counted only when all the search strings occur within five atoms in conjunctive searches.

** Because these ANDs are interpreted as AND_5 in the atom-document searches, $n > N$ may hold.

In the implementation of the atom-addressing method, JPerl5, which is a Japanese version of Perl5 language, was used. A non-volatile hashing table package, GNU DBM, was used to build the inverted indices. The search engine, which is a character bi-gram engine, was also implemented using JPerl5. Because Perl programs are executed by an interpreter, the search time was an order of magnitude greater than with search engines using native code. Score values were propagated throughout a document, but no propagation through hyperlinks was introduced. Because the size of each document was relatively small (i.e., 2 kB on average) the search time was not greatly lengthened by the score propagation. The measurement results are shown in **Table 3**.

The Inprise Delphi developing environment was used for implementing the atom-document search method. The character uni-gram search engine used in the full-text document search was also used for the atom-document search. Score values were propagated only between matched strings with no other matched string between. These results are also shown in Table 3.

Table 3. Search time in two implementations of the fine-grained search (Unit: second)

Query	Atom-addressing method**		Atom-document method†		Document full-text search method†	
	CPU time	Elapsed time	CPU time	Elapsed time	CPU time	Elapsed time
Shikarami-soshi (しからみ草紙)	12.30	13	1.8	4.6	1.2	4.5
Presley (プレスリー)	2.03	2	1.1	4.5	0.9	4.2
Iridium (イリジウム)	1.11	1	1.2	3.2	0.7	3.1
Cocoa (ココア)	0.62	1	1.0	2.3	0.6	1.7
Yoshinobu (慶喜)	0.49	1	0.7	1.4	0.1*	0.5
Chocolate (チョコレート)	1.94	2	1.1	2.5	0.7	1.9
Asano (浅野)	0.66	1	0.7	1.2	0.1*	0.8
Computer (コンピュータ)	5.43	6	1.3	2.8	0.9	2.7
Life (生命)	2.49	3	0.9	1.8	0.1*	1.2
Tokugawa (徳川)	2.23	2	1.0	2.5	0.1*	0.6
Philosophy (哲学)	5.16	5	1.1	2.1	0.1*	0.8
America (アメリカ)	30.99	31	2.2	2.7	0.6	2.9
AND(Elvis, Presley) ††	-	-	2.0	3.9	1.1	2.5
AND(Cocoa, Chocolate) ††	-	-	1.9	3.8	0.9	5.0
AND(Asano, Soichiro (総一郎)) ††	-	-	1.4	2.3	0.3	2.2
AND(Asano, Naganori (長矩)) ††	-	-	1.3	3.1	0.1*	1.0
AND(Pre (プレ), Sley (スリー)) ††	-	-	2.1	5.0	0.9	3.4
AND(Tokugawa, Yoshinobu) ††	-	-	1.2	1.8	0.1*	0.9
AND(Art (アール), Nuvo (ヌーボー)) ††	-	-	2.0	2.4	1.2	2.4
AND(Computer, Communication (通信)) ††	-	-	1.9	3.4	0.9	3.2
AND(Life, Philosophy) ††	-	-	1.5	2.3	0.3	1.8
AND(Tokugawa, Ieyasu) ††	-	-	1.4	2.1	0.1*	1.4
AND(Japan (日本), America) ††	-	-	3.2	4.0	0.9	2.9
AND(Ame, Rica) ††	-	-	3.2	4.0	0.7	2.4
AND(Asano, Soichiro, Bank (銀行)) ††	-	-	1.5	2.5	0.4	2.5
AND(Cocoa, Chokolate, Japan) ††	-	-	2.5	5.0	1.2	5.9
AND(Computer, Media (メディア), Economy (経済)) ††	-	-	2.1	4.7	1.1	4.7
AND(Art, Nuvo, Painting (絵画)) ††	-	-	2.4	3.3	1.2	3.7
AND(Tokugawa, Ieyasu, Hidetada (秀忠)) ††	-	-	1.5	2.6	0.1*	2.6
AND(Japan, America, Treaty) ††	-	-	2.9	7.5	0.9	4.8

* Below the measurable limit.

** The server used for these measurements had a Pentium Pro 200 MHz CPU, 192 MB main memory, and an UltraWide 7200 rpm hard disk. Both CPU and elapsed time were measured on the server-side.

† The computer used for these measurements had an AMD K6 233 MHz CPU, 128 MB main memory, and an EIDE 5400 rpm hard disk. The CPU time could not be measured directly. So the elapsed time from the beginning of the search to the beginning of the results display was measured for both when the index is loaded on the cache and when it was not loaded, and the former was regarded as the CPU time and the latter was regarded as the elapsed time.

†† These ANDs are interpreted as AND_5 in the atom-document searches.

The atom-addressing and atom-document search methods were compared here. On average, the CPU time of the former was 2.7 times longer, and the elapsed time was 1.2 times longer than the latter. In some queries, though, the CPU time was much longer. However, the performance of the atom-addressing method in terms of elapsed time is sufficient for practical searches, in spite of the interpreter overhead.

5 Related Work

The conceptual index by W. A. Woods [Woo 97] is a method for retrieving "concepts" that the user specified, where the hyperlinks to the occurrence positions of the concepts in the document collection are retrieved. This method can be regarded as a type of fine-grained search. However, there is no propagation-like mechanism in the conceptual index method.

6 Conclusion

This paper contributes to the development of information retrieval, both in modeling and in practice, as follows.

- A model of a new search method called the fine-grained search method, has been proposed. In this method, each text to be searched is a sequence of atoms, and hyperlinks connect the atoms. The relationship between atoms are represented by a unified mechanism, i.e., score propagation.
- Two methods of fine-grained searching have been developed: the atom-addressing search and the atom-document search methods. These methods can be implemented by using a conventional full-text search engine as is or with only a small modification. These methods are compared and the latter is better in several criteria. Both have been implemented and shown to be practical.
- A search model of fine-grained searching was described, and the user cost in the search model was measured. Retrievals can be performed with less cost (faster) by using the fine-grained search rather than the conventional full-text search.

However, although the user cost has been decreased by introducing the fine-grained search method, this method is not sufficient for searching the huge text collections that can be supplied by the Internet, DVD-ROM, and other media. Search methods with a function to organize search results will be required. The axis-specified search [Kan 98a] [Kan 98b] is a candidate for this. However, a more general framework and/or a range of organizing search methods should be developed.

Acknowledgment

I am grateful to Yasufumi Fujii and others at the Hitachi Digital Heibonsha Company for allowing use of the text of the World Encyclopædia.

References

[Cut 92] Cutting, D. R., Karger, D. R., Pedersen, J. O., and Tukey, J. W.: Scatter/Gather: a cluster-based approach to browsing large document collections, *15th Int'l ACM SIGIR conference on Research and Development in Information Retrieval*, 318–329, 1992.

[Cut 93] Cutting, D. R., Karger, D. R., and Pedersen, J. O.: Constant interaction-time scatter/gather browsing of very large document collections, *16th Int'l ACM SIGIR conference on Research and Development in Information Retrieval*, 126–134, 1993.

[Fra 92] Frakes, W., and Baeza-Yates, R., ed.: *Information Retrieval: Data Structures & Algorithms*, Prentice Hall, 1992.

[HDH 98] *DVD/CD-ROM World Encyclopædia, version 2*, Hitachi Digital Heibonsha, 1998.

[Kan 98a] Kanada, Y.: Axis-specified Search: A New Full-text Search Method for Gathering and Structuring Excerpts, *3rd Int'l ACM Conf. on Digital Libraries*, 108–117.

[Kan 98b] Kanada, Y.: Axis-specified Search: A Method for Searching and Ordering Excerpts from A Document Collection, *Information Processing and Management*, submitted.

[Mor 68] Morrison, D. R.: PATRICIA — Practical Algorithm to Retrieve Information Coded in Alphanumeric, *Journal of the ACM*, 15:4, 514–534, 1968.

[Mor 95] Morohashi, M., and Takeda, K.: Information Outlining — Filling the Gap between Visualization and Navigation in Digital Libraries, *Int'l Symp. on Research, Development and Practice in Digital Libraries 1995*, pp. 151–158, Univ. of Library and Information Science, 1995.

[Woo 97] Woods, W. A.: Conceptual Indexing: A Better Way to Organize Knowledge, *SML Technical Report*, Sun Microsystems Laboratories, 1997.