Hyperindices: A Novel Aid for Searching in Hypermedia

Peter D. Bruza

Department of Information Systems
University of Nijmegen
Toernooiveld, 6525 ED Nijmegen
The Netherlands
E-mail: peterb@cs.kun.nl

ABSTRACT: In this article the formal basis of hyperindices is given. Hyperindices are a new means for supporting effective search in hypermedia. The basis of the hyperindex, the so called index expression is treated in detail. It is shown how the hyperindex can be constructed using the structural properties of the index expression. The hyperindex is placed in a general framework for indexes which features quantitative and qualitative criteria with which index effectiveness can be judged.

KEY WORDS: hypertext, information retrieval, indexing.


1 Introduction

People wishing to retrieve information are sometimes confronted with a problem at the beginning of their search. It may be that they do not have a clear idea of their information need and thus begin to arbitrarily browse or ask queries in the hope that they will find what they want. Alternatively, they may have a clear idea of their information need, but are unable to formulate it in a query. This may be because they are not proficient with the query language at hand, or that the query language is not sufficiently powerful to specify their information need.

A useful aid in the above situations is an index. The term index is heavily taxed, so for this reason the reader can for the moment acquaint themselves with this notion by imagining indexes such as those at the back of a book. The advantage of the index is that the searcher is not faced with the problem of having to express their information need in the form of a query. Also,
for those searchers who have no clear notion of their information need, an index is often helpful to clarify this need.

In this article the formal basis of a new type of index is presented. This index, called the hyperindex, has the indexing information organized in the form of a hypertext. This form of index has already been less formally introduced in [BvdW90] where it is presented in the framework of a novel two level architecture. (See figure 1).

Figure 1: The Architecture of Two Level Hypermedia

The hyperbase corresponds to a hypermedia as is typical in current systems [Con87][SK89]. The searcher can navigate structurally over the information, for example, by moving from a chapter to a section, or by following cross references. The latter is referred to as associative navigation.

The hyperindex forms the top level of the architecture. It is a hypertext of index terms which index the underlying hyperbase. The feature of the hyperindex is its structure. (See figure 2). The structure provides the opportunity to search through the index terms in an organized fashion. This facet not only facilitates information retrieval but can guide a possibly distracted or lost searcher. The basis of the organized search capability is the fact that any focus (the current index term being scanned by the searcher), can be refined (context contraction) or enlarged (context extension). For ex-
ample, given the fragment of the hyperindex depicted in figure 3 the current focus can be refined to become attitudes to courses of students or attitudes to courses in universities. Conversely, the current focus can be enlarged to become attitudes or courses.

When an index term is found which describes the information need, the objects from the underlying hyperbase which are characterized by this index term can be retrieved and examined. This operation is referred to as a beam down because the searcher is transferred from the hyperindex to a view at the hyperbase level which is constructed from the relevant objects. Navigating through the hyperindex and retrieving information in the above way has been coined Query By Navigation [BvdW90].

The hyperindex is an extension of the lattice-based model of information retrieval presented in [GGP89]. This model consists of a lattice of index terms over a document set. The searcher interacts with the system essentially by walking over the lattice. A node in the lattice represents a query. The query can be enlarged or refined by moving to descendant or ancestor nodes respectively.

```
most general

refine

most specific

enlarge
```

Figure 2: Conceptual View of a Hyperindex

This article focuses on the hyperindex and the so called index expressions which form its basis. Section 2 introduces the notion of an index and some associated terminology. Section 3 formally defines the index expressions. Also in this section it is shown how the structural properties of the index expressions can be employed to construct the hyperindex. In section 4 the
user interface of the hyperindex is discussed and in section 5 the effectiveness of the hyperindex is discussed in terms of a general index effectiveness framework. Finally, in section 6 some conclusions and directions for further research are given.

2 Introduction to Indexes

The basis of all indexes is a set of objects. An object is typically a document, or a structural element of a document, for example, a chapter or section. From this point these structural elements will be referred to as molecules. Objects are represented in a particular medium, for example text, video, audio etc. The set of objects is termed the object base. An example of an object base is a base of documents or a library of books. Another example is a hyperbase. The objects in the hyperbase are information fragments, nodes and molecules.

An index is an aid that facilitates the location of objects. It is a set of index entries where an index entry is composed of two parts: a descriptor and a set of locators. The descriptor is a compact, incomplete description of the content of an object. The locator is a pointer to an object which is characterized by the associated descriptor.

A descriptor can be a single term, a series of terms, or more complicated descriptions such as use of indexes for effective information retrieval.

The descriptors conform to a certain syntax. The language specified by this syntax is referred to as the indexing language. The descriptors are used to order the index entries to facilitate the quick location of relevant index entries. For example, in book indexes the index entries are typically ordered alphabetically on the descriptors.

The index is constructed by indexing each object in the object base.

Figure 3: Example Fragment of a Hyperindex
Indexing is a process in which a set of descriptors are produced for that object. These descriptors are sometimes derived directly from the objects themselves. In this case each object is scanned, either manually or automatically, to identify or derive good characterizations of the content of the object. Indexing can also be performed by assigning the object a so-called *string* and using this as basis for the generation of a set of descriptors for that object. Typically the string is permuted and manipulated according to certain rules. For example, the string *use of animals in experiments* generates the following set of descriptors according to rules which will be presented later in this document: { *use*, *animals*, *experiments*, *use of animals*, *animals in experiments*, *use of animals in experiments* }. There are many variations and possibilities in string indexing. See [Cra86] for more details.

If the searcher finds a descriptor in the index which they deem relevant to their information need, then the associated objects can be retrieved by using the locators. In other words, an index entry can be considered as a predefined query with associated result. This type of information retrieval is commonly referred to as precoordinate retrieval [Rij79]. Precoordinate retrieval has the advantage that the searcher does not have to have knowledge of some query language and as all query results are *a priori* known, this can be used to advantage with regard to optimization.

Indexes may also contain cross references which are used to unidirectionally associate related index entries. For example, See also: *use of chimpanzees in research on AIDS*.

Finally, another facet of indexes is their *presentation*. This is how the index is displayed to the user.

### 3 Descriptors: Terms to Index Expressions

#### 3.1 Term and Term Phrase Descriptors

There are a number of forms that a descriptor can take. The most elementary form of descriptor is a keyword, or *term*. Term descriptors have the advantage that there are a number of straightforward indexing methods to derive them from the objects [Sal89]. The disadvantage of terms descriptors is that they can lack *specificity*, particularly in large object bases. The specificity of a descriptor refers to how detailed or specific the descriptor is. It can be expressed in terms of the probability of the descriptor’s occurrence. The smaller the probability of occurrence, the greater the specificity [Sal83]. Specificity is an important criterium for judging the effectiveness of a descriptor. Effectiveness issues are dealt with in section 5.
A term phrase is an extension of the term descriptors to increase their specificity. For example, the term phrase computer programming is more specific than the term computer or programming. A term phrase is a series of two or more terms. The most common term phrases consist of two terms. Even though straightforward indexing methods exist for the generation of term phrases [Sal89], these methods often suffer from the problem that either too many non-meaningful phrases are generated, or conversely a large percentage of the phrases are meaningful but the resulting indexes lack exhaustivity. (See section 5 for the definition of exhaustivity).

3.2 Index Expression Descriptors
An extension to the term phrases are the so-called index expressions. In an index expression descriptor the relationships between the terms are denoted. For this reason they are more specific than term phrase descriptors. In contrast to term descriptors or term phrase descriptors, index expressions have a structure. This structure can be used to derive a lattice of descriptors which supports query by navigation. More details about are given later in this section.

3.2.1 Base Index Expressions
Index expressions are constructed from the so-called base index expressions. These expressions first appeared as linked phrases in [Cra78] and have the following syntax (in extended BNF format):

\[
\begin{align*}
\text{BaseExpr} & \rightarrow \text{Term} \{ \text{Connector BaseExpr} \}^* \\
\text{Term} & \rightarrow \text{string} \\
\text{Connector} & \rightarrow \text{string}
\end{align*}
\]

A term corresponds to a noun, noun-qualifying adjective or noun phrase. A connector denotes a relationship type between two terms and is basically restricted to the prepositions and a so-called null connector which is denoted by ". Refer to [Bru90] for a complete description of the allowable connectors and the relationship types they denote.

Base index expressions are sometimes ambiguous. For example, consider the base index expression attitudes to courses of students in universities.

In this example, are the attitudes, university student attitudes to courses, or, are the attitudes, university attitudes to student courses? To resolve ambiguity, base index expressions are sometimes represented as ordered trees. Each representation establishes a unique interpretation for the associated
expression. The tree representation of the second interpretation is depicted in figure 4.

![Diagram](image)

Figure 4: Example Base Index Expression Representation

Dealing with different interpretations of an index expression is complicated. For this reason, we demand that every base index expression \(B\) has a unique representation, denoted by \(\rho(B)\). Brackets are used to de-ambiguate the expression. For example, \textit{attitudes to (courses of students) in (universities)} corresponds to the representation depicted in figure 4.

3.2.2 Power Base Expressions

Using the base index expression, the notion of a \textit{power base expression} is introduced. This notion is similar to the power set concept. The power base expression of a base index expression is the set of all index sub-expressions including the empty base expression, which is denoted by \(\epsilon\). More formally, 

**Definition 3.1** Let \(B\) be a base index expression, the power base expression of \(B\), denoted by \(\mathcal{P}(B)\) is the set

\[
\mathcal{P}(B) = \{\text{preorder}(T) | T \text{ subtree of } \rho(B)\}
\]

Like the powerset, the power base expression forms a lattice where the underlying ordering relation can be interpreted as \textit{is index sub-expression of}. The top of the lattice is the full base expression and the bottom is the empty base expression, \(\epsilon\). Lattices (and partial orders in general) can be elegantly represented as Hasse diagrams. The Hasse diagram of the power index expression represented in figure 4 is depicted in figure 5.

The lattice has some useful properties. If we consider that every vertex is a potential focus of the searcher then the surrounding descriptors of the focus are enlargements or refinements of the context represented by the focus. The searcher can thus move across the lattice by refining or enlarging the current focus until a focus is found which is relevant to their information
need. Searching the hyperindex is then reduced to an organized form of browsing which is an easily understood form of searching behaviour.

3.2.3 Index Expressions
Index expressions are constructed from base index expressions by means of conjunction. The following grammar defines the language of the index expressions:

\[
\text{IdxExpr} \rightarrow \text{BaseExpr} \{\text{Conjunctor BaseExpr}\}^* \\
\text{Conjunctor} \rightarrow \text{AND}
\]

An example of an index expression is effective retrieval systems AND people in need of information.

The power base expression can be extended to the notion of a power index expression. The power index expression of an index expression is defined in terms of the power base expressions of the base expressions of which the index expression is composed. Given an index expression \( I = b_1 \text{AND} \ldots \text{AND} b_n \) where \( b_i \) is a base expression, then the power index expression of \( I \) is defined as \( P(I) = \bigcup_{i=1}^n P(b_i) \cup \{I\} \).

Unlike the power base expression, the power index expression is not always a lattice. (This is because the union of lattices does not necessarily result in a lattice). It is however lattice-like, namely it is a partial order with a single maximal element, \( I \), and a single minimal element, \( \epsilon \).
ordering relation is the *is index sub-expression of* relation introduced earlier. Power index expressions can also be represented as Hasse diagrams. The Hasse diagram of the example index expression is depicted in figure 6. In this diagram one can see how the power index expression is constructed from power base expressions. The left hand side of the diagram consists of the descriptors which are generated from the base index expression *effective information retrieval* and the right hand side consists of the descriptors generated from the base index expression *people in need of information*. Note that the power base expressions overlap via the descriptor *information*.

![Hasse diagram](image)

Figure 6: Example Power Index Expression

### 3.2.4 The Construction of the Hyperindex

The index expressions and their structural properties form the basis of the hyperindex. The hyperindex is the power index expression of an index expression, denoted by $T$, which essentially characterizes all objects in the object base. This index expression is constructed in the following way. Every object in the object base is characterized by an index expression. Say that there are $n$ objects and the $i$th object is characterized by index expression $I_i$. The index expression $T$ is equal to $(I_1 \text{AND} \ldots \text{AND} (I_n)$ and the hyperindex is thus equal to $P(T)$.

As the hyperindex is a power index expression it is a lattice-like structure. This structure demarcates a space of descriptors within which the searcher
can navigate via enlargement and refinement as described earlier.

4 The Presentation of the Hyperindex

The presentation of the hyperindex consists of a set of nodes connected to each other by means of structural links. The relationship between the hyperindex and its presentation can be understood by considering the Hasse diagram of the hyperindex. Every vertex in the diagram has a corresponding hypertext node. The node consists of a single descriptor and a number of buttons. The descriptor corresponds to the current focus of the searcher in the hyperindex. The buttons are descriptors which denote enlargement and refinement possibilities of the focus. By activating such a button the searcher traverses a structural link to another hypertext node. This corresponds to moving along an edge in the Hasse diagram. Such a move is demonstrated in figure 7 and figure 8, using the Hasse diagram shown in figure 6 as the hyperindex.

![Hasse diagram of the hyperindex](image)

Figure 7: The situation as the button is activated

When the searcher decides that the current focus is relevant to his or her information need, the associated objects can be retrieved. In the case of the hyperindex this is not done by means of locators, but by the so called beam down operation. (Refer to figure 1). This operation results in a hypermedia view composed of the objects which are characterized by the
focus of the beam down. The beam up and beam down operations as well as view structures in relation to the two level architecture are documented in [BvdW90].

5 The Effectiveness of Indexes

The effectiveness of an index is directly related to the effectiveness of the descriptors. As the descriptors are in essence queries, the standard information retrieval effectiveness criteria of precision and recall are applicable [Sal89]. Given that \( N \) is the set of objects that best corresponds to the searcher’s information need, \( d \) is a descriptor, and \( res(d) \) is the set of objects associated with \( d \), then

**Precision:** The precision of a descriptor is the ratio of the number of relevant objects associated with the descriptor, to the total number of objects associated with the descriptor. More formally, \( \text{precision}(d) = \frac{|N \cap res(d)|}{|res(d)|} \).

**Recall:** The recall of a descriptor is the ratio of the number of relevant objects associated with the descriptor to the total number of relevant objects. More formally, \( \text{recall}(d) = \frac{|N \cap res(d)|}{|N|} \).

More specific descriptors give better precision at the expense of recall and vice versa. The hyperindex thus offers both good recall and precision as it
contains descriptors which range in specificity.

Another criterion from the field of information retrieval which can be applied to indexes is the following:

**Exhaustivity**: The exhaustivity of an index is the degree to which the content of the objects are reflected in the descriptors of the index.

The exhaustivity of the hyperindex is therefore directly related to the ability of the index expressions to describe the contents of objects. One can conclude that hyperindices are more exhaustive than indexes based on terms or term phrases because more of the content can be described by virtue of the connectors. We agree with [Far80] who states the term relationships contribute significantly to the content of objects.

As searching an index implies scanning individual descriptors, there are criteria with which descriptors can be judged in regard to their scannability. The first is the notion of the power of a descriptor. This is a quantitative criterium based on the notion of descriptor succinctness introduced in [Cra86].

**Power**: The power of a descriptor is a ratio of its specificity to its length (in characters).

The power of a descriptor is inversely proportional to its length because the longer the descriptor, the more taxing it is for the searcher to scan it. This means that for two descriptors with equal specificity, the longer is less powerful. By virtue of the connectors, index expressions are more powerful than term phrases. See [Bru90] for a formal exposition of the power of index expressions.

Eliminability and Clarity are also introduced in [Cra86] as qualitative criteria for judging the effectiveness of descriptors. Eliminability means how quickly a searcher can determine the irrelevance of a descriptor and thus break off scanning it. Clarity refers to how readily searchers grasp the intended meaning of the descriptor. Index expressions afford good clarity by virtue of the connectors. In [Cra86] research is cited which concludes that searchers prefer having connectors in descriptors. This is in part due to the fact that without connectors the relationship between terms is sometimes not clear.

A important criterium in judging the effectiveness of an index is that of predictability. This notion is defined in [Cra86] as follows:

**Predictability**: The predictability of an index is a measure of the extent to which a searcher can predict where relevant descriptors can be found in the index.
Predictability is related to the number of descriptors the searcher has to scan before a relevant descriptor is found. In a predictable index, relatively few descriptors should have to be scanned to find a relevant descriptor. The hyperindex is predictable due to its underlying lattice-like organization which supports refinement or enlargement of the current descriptor.

Once a searcher has found a relevant descriptor, the question arises as to whether the descriptor found is most relevant with respect to searcher’s information need. In other words there could be a descriptor in the index which better describes the information need of which the searcher is not aware. It is therefore important that relevant descriptors be near each other in the index so that when the searcher finds a relevant descriptor, other relevant descriptors are also found. The criterion of collocation addresses this aspect of an index.

**Collocation:** The collocation of an index is the measure of the extent to which relevant index items are near each other in the index.

The collocation of an index can be quantified by averaging the collocation around each descriptor. Given that \( R \) is a set of descriptors relevant to an information need, \( f \in R \) is the current focus (that is the searcher has found a relevant descriptor), and \( dist(i, j) \) is a measure of the distance between descriptors \( i \) and \( j \) in the index, then the collocation around descriptor \( f \) is the average distance from \( f \) to another relevant descriptor in the index. More formally, \( C(f) = \frac{\sum_{d \in R} dist(f, d)}{|R|} \). The notion of distance is dependent on the type of index. For example, if the index is a book index, then the distance between two descriptors can be defined as the number of intervening index entries.

One of the main advantages of the hyperindex is its high collocation. This is due to each descriptor in the hyperindex being surrounded by those descriptors that overlap with it. There is a strong likelihood that given a descriptor \( d \) which is relevant to a given information need, then some of the descriptors that overlap with \( d \) will also be relevant.

6 Conclusions and Further Research

In this article the hyperindex has been formally defined. The descriptors of the hyperindex, the so-called index expressions have been described in detail. It is further shown how the hyperindex can be derived from the structural properties of the index expressions. In addition a framework has been proposed in which the effectiveness of indexes can be judged. In
the context of this framework it can be concluded that the hyperindex is sufficiently promising to warrant further investigation. In this regard we are building a prototype system to obtain some practical results. It can also be concluded that hyperindices are likely to be superior to indexes based on terms or term phrases, particularly in regard to collocation and exhaustivity.

There are a number of open problems. The first is the indexing of objects which result in index expression descriptors. Thus far, this has been performed manually. Initial investigations involving the titles of objects show that these are often in a form which is amenable to a more or less direct translation to an index expression.

Another problem is that the number of buttons in the nodes of the presentation of the hyperindex is determined by the fan in and fan out of the vertices in the lattice structure. This is particularly a problem for the start node, because the bottom element has a potentially huge fan out. It may be possible to group the descriptors to result in a set of smaller nodes. Another possibility is to prune the lattice by removing terms that are not so ‘good’. There are statistical techniques to identify good terms [Sal89].

References


