

# Improving Hypertext Presentation by Structuring Information Space

Douglas Grundman and Andrea Michalek

Infonautics Corporation

April 2000

## *Introduction*

Displaying generated hypertext documents such as search results in a manner that is easily assimilated by users is hard. These documents are typically represented as relatively simple lists of search engine hits. The display of any kind of information that lets the user understand how the referred-to documents may relate to the query is usually lacking. Thus, even if a search result page can be thought of as a document about a single topic, the user must read each of the entries on it independently to ascertain that fact.

Exposing some structure that shows how documents in a result list relate to the query can lead to a better navigation experience for users. In this paper, we will take the approach that structuring the result list is most powerfully and naturally achieved by *pre-structuring* the information being presented. Additionally, when the structure of the information is already known, it becomes much easier to structure its presentation.

We will determine the structure of the information we wish to present by defining *vertical slices of information*, and will extract our desired structure from these slices.

## *How Users Search for Information*

Users of Internet search engines typically ask queries composed of three or fewer words. These short query phrases are frequently incomplete or ambiguous, but the user almost invariably has a particular meaning in mind. Unfortunately, the search engine rarely knows which meaning the user intends, and the information returned by the search engine in the context of the various meanings is probably best displayed in different ways. The problem might be largely solved if only we could identify these various contexts ahead of time.

In fact, we can take some fairly large steps in exactly that direction. Users ask textual queries. Collecting and analyzing query logs, one notices that many of these queries are specific instances of relatively concrete classes. Examples of these *query classes* might be “company names”, “sports teams”, or “movie stars”; the queries we’re referring to are the obvious specific instances of these classes. The set of documents related to the queries in a query class is a very useful concept, and we will use it to define our contexts for searching.

## ***Vertical Slices of Information***

We define a *vertical slice of information* to be a set of documents related to a query class when that collection of documents forms a single context that a large number of users share an interest in. Note that we make the notion of “interest” a central part of the definition here – we may be interested in defining such sets of documents for query classes such as “sushi bars in California”, but not for abstractly defined query classes such as “all queries starting with the letter T.”

When a query is submitted to a search engine in the context of a particular vertical slice, the set of documents searched and presented to the user is restricted to the documents in that slice. At first glance, it might seem that this would serve end-users poorly by restricting the information to which they have access, but this is not necessarily so. By restricting the document set to one in which users are known to have particular interest, we can do better at presenting that information to those users. This restriction can either be done dynamically at search time, or statically by creating entirely separate products tailored to meet end users’ needs in particular vertical market segments.

As an example, consider Infonautics’ CompanySleuth.com Web product. This product locates and presents information about public corporations to investors. By restricting its domain to business and company information, CompanySleuth.com is able to do a far better job of presenting users with relevant information on that topic than is an unrestricted search engine.

Infonautics’ SleuthCenter.com product goes one step farther, addressing five separate vertical slices. It does this by exposing doorways to the five separate services, each of which specializes in presenting a particular vertical slice.

### ***Structuring the Display in the Simplest Case: All Queries are Predetermined***

Our overall objective is to find ways to better present information to users. The simplest case to consider is when all the possible queries are known ahead of time. (This is in fact true in the case of the CompanySleuth.com site, since there are only about 10,000 public corporations in existence in the United States.)

In this case, one can use the above-mentioned technique to define a vertical slice of information to address, then divide that slice into sub-areas of interest determined by the slice at hand (or rather, by the interests of its users). For example, the vertical slice of “information about companies” might be divided into general news, buzz on the Web, intellectual assets, corporate publications, and what external observers are saying – and each of these might be subdivided further. Documents about each company named in the associated query class would then be sorted into one of these hierarchically organized buckets. Finally, any particular company’s information could be displayed as a hypertext page organizing that information in terms of those subcategories. This would provide easy location of and navigation to those documents. Figure 1 shows a screenshot of CompanySleuth.com where this organization has been implemented.

Figure 2 depicts the process creating these pages. The vertical slice is defined as a set of content (here, from the Internet). This content is then sorted into several “horizontal” buckets within that vertical slice. Finally, when information about a particular entity is requested, that information is presented in a structured form where the structure mirrors the way the buckets are organized.

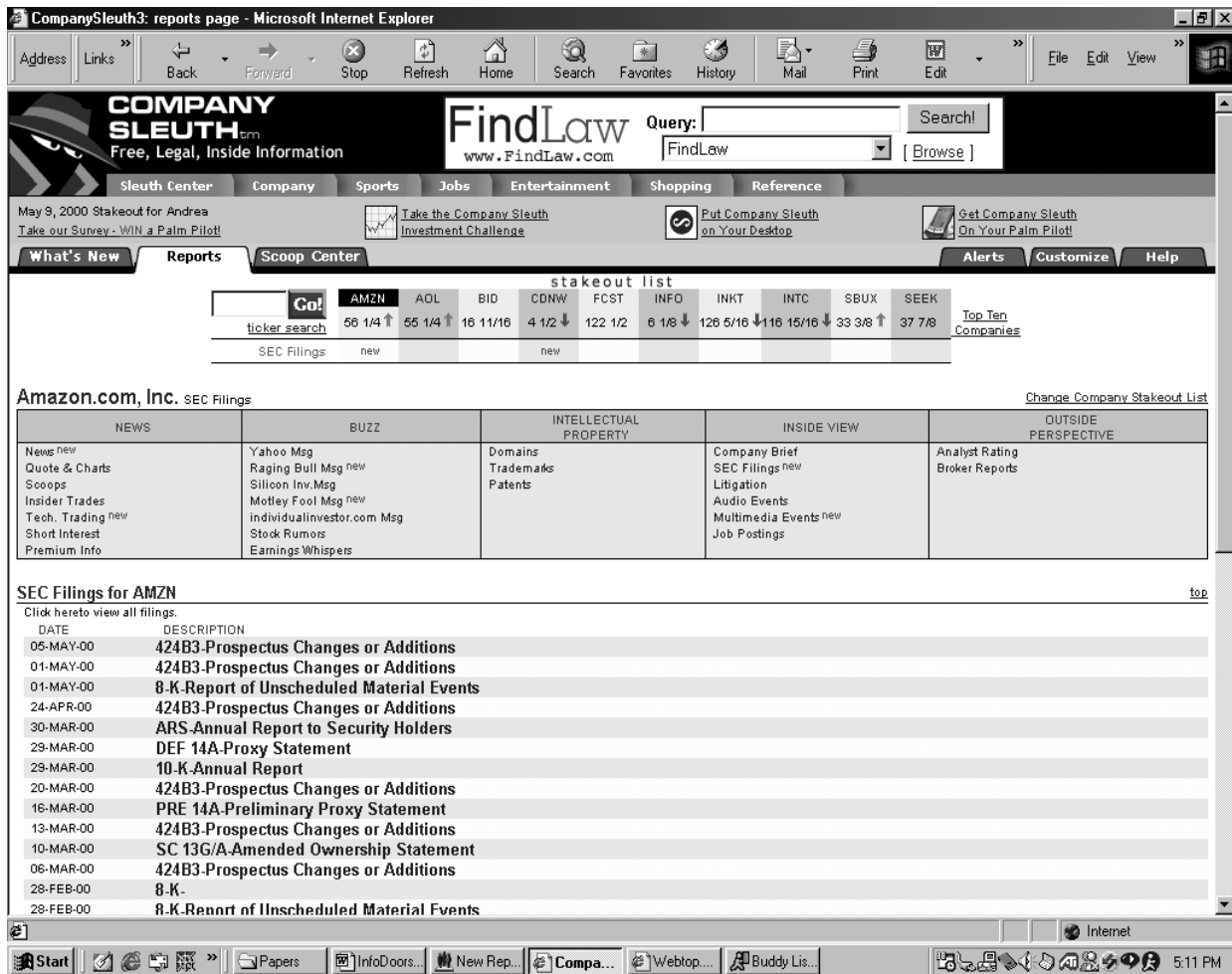


Figure 1 – CompanySleuth.com

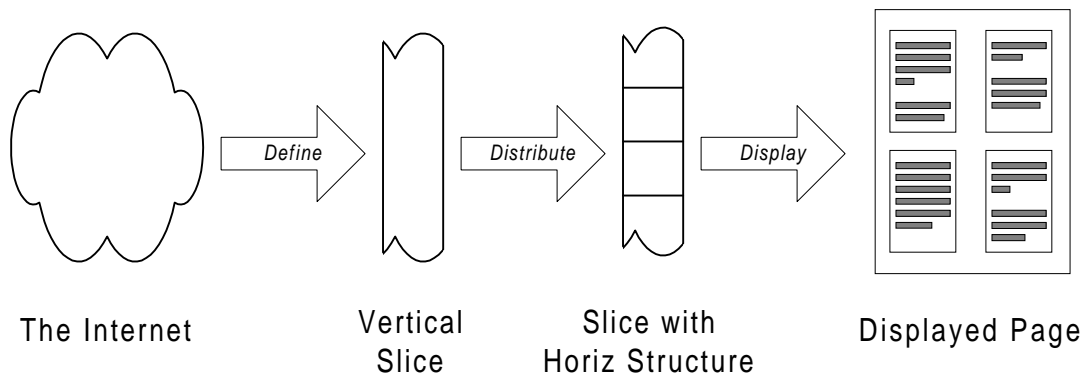


Figure 2 – Extracting Display Structure from a Slice

It is interesting to note that the information that is extracted and presented to the user is extremely dense, yet very navigable. In the CompanySleuth.com example above, the five-entry table in the middle of the page shows the structure that has been pre-defined for public company information. Each horizontal bucket there (such as “News”, “Quotes & Charts”, “Scoops”, etc.) refers to a particular subset of results. In this interface, clicking on any one bucket causes its result subset to be displayed below the table. Those result subsets are each formatted in a meaningful way for that type of information – for example, the subset shown is that for company SEC filings, where time-ordering of the results is important so as to preserve the sequencing of events for the reader. Patents owned by a company would instead be displayed in patent number order, etc.

Despite the fact that the set of queries is predetermined, the results of a given query are not constant – the corpus in question is undoubtedly changing constantly, usually faster than the set of entities of interest to the user base. Thus, it is not unreasonable to assume a constant predefined set of queries and a fixed structure for answers in the given domain of discourse.

***Categorizing Documents within a Vertical Slice via Cross-document Metadata***

The above approach maps documents to queries by attending to where documents originate. This mapping of documents in a vertical slice may also be determined by an automated analysis of their text. For example, documents that contain a particular product name, the name of a particular football player, or the name of a particular city may be thought of as answering the appropriate query. By examining documents to extract features that are meaningful to users in the appropriate domain of discourse, these mappings of documents may be discovered.

More formally, derived metadata may be thought of (and implemented) as an association of a predefined finite set of queries to documents. The statement, “document X has piece Y of metadata” means that document X will result (possibly among others) from submitting query Y to a suitable search engine into which document X has been indexed. “Cross-document metadata” is metadata that applies to more than one document. Each piece of cross-document metadata then identifies a particular subset of documents in a vertical slice of content. This explicitly assigns documents to particular queries in a query class. In fact, since the individual metadata entities need not be defined prior to construction of the service (though they are defined by the builders of the service), it is not necessary to know the set of possible queries *a-priori*.



However, determining appropriate metadata to extract is a key factor in the successful structuring of a particular vertical slice. “Interesting” cross-document metadata should be present across sub-groupings of documents within a vertical slice. Once the type of metadata relevant to a vertical slice is chosen, a variety of automated extraction mechanisms can be utilized. For example, Infonautics’ Entertainment.Sleuth.com web product does this by using a metadata set consisting of the names of famous celebrities in the entertainment world. This set of people changes frequently, but since automated methods are used to extract the names from documents, the system needs no reconfiguration when such changes happen.

Another point to note is that it is not necessary to address complex metadata such as hierarchical or 2-Dimensional relationships to derive significant value from this approach. Often, extracting information that can be organized into simple linear lists is a powerful and straightforward tool for organizing information in a vertical slice.

### ***Structuring the Display of the Results of Free-text Queries***

Extending the system to work with free-text queries is now seen to be relatively easy. One still needs to find a class of queries that a community of users is interested in asking, to define a vertical slice from that query class, and to determine a structure for displaying documents from that vertical slice. One can then index the documents in that vertical slice and allow free querying across it. All documents returned may be displayed according to the relevant structure no matter what method is used to locate them within the slice.

Note that with all of these ways of structuring results, the structure is predefined by the designer of the service according to what makes sense in the domain of discourse – the vertical slice of information under scrutiny.

### ***Conclusion***

We have defined a *query class* as a well-behaved set of closely related queries that users are interested in asking, and a *vertical slice of information* as a set of documents answering the queries in a query class. Given a vertical slice, an Information Retrieval System designer can determine a structure suitable for answering queries in the query set in an understandable and information-rich way. Any query posed to the system may be answered through that structure.

The Web is a very big place, and there are a great many useful and interesting vertical markets there that people have interest in. We have shown how, by structuring the kind of content people can ask for, or by structuring the information that people find, one can create a user experience that far surpasses that produced by standard search and retrieval user interfaces.

Finally, note that if a search service also serves the actual documents to end users, it can exploit our approach in another interesting way – by annotating each returned document with some of the information that was available when that document was located through a search. Relevant information might include links to closely related documents, such as some subset of those that appeared on the dynamic result page that lead the user to this document. This information would naturally be displayed in a structured form, perhaps similar to the format used on result pages. If this is done, every ordinary document serves as the basis for a dynamically generated structured hypertext, thus broadening the applicability of our approach far beyond just result pages.