Patent Space Visualization for Patent Retrieval

A. W. McLean

Canon Research Centre Europe Ltd, Guildford Research Park, Surrey GU2 7YP, UK

July 14, 2000

Abstract

We present work describing a novel architecture and user interface for patent queries and visualization of query results. Existing patent retrieval engines such as QPAT[1] allow users to enter text queries, retrieve a set of results and then refine these queries. However this approach and method of presentation of results is not necessarily very informative for the user. We gathered an initial set of requirements from patent engineers at our company and explored techniques that allow a user to build up a stack of queries which allow real time updates of any part of the query and display the results over a 2D map. This allows the user to gain a better understanding of the patent space (the search space for this type of data related to his query). The architecture allows users to develop their own query sets and different query modules (such as more sophisticated clustering/analysis algorithms) can be easily plugged-in. A *query-part* can be a normal retrieval query (eg, search on keyword/assignee), domain specific (eg search using the International Patent Classification (IPC) hierarchy for related information), input (obtain patents from this site, at this time interval), processing (analyse this set using this algorithm), or visualization (use this process/algorithm to display these results).

1 Introduction

We describe here a software architecture and a prototype patent visualization and retrieval application that, based on a generic architecture, *integrates* retrieval with interaction. Figure 1 shows the basic notion of interactive information retrieval. In our architecture each query module is displayed in a stack along with its parameters. Reformulation of a query is achieved by adding a new query module, removing one or altering the parameters of a module (see figure 3). The controls to modify the parameters can be general UI widgets (buttons, sliders, combo-boxes etc). This integration gives the user a mode of *browsing* the data by rapidly changing the query and its parameters and the effects of one query-part on the results from the entire query can be readily examined.

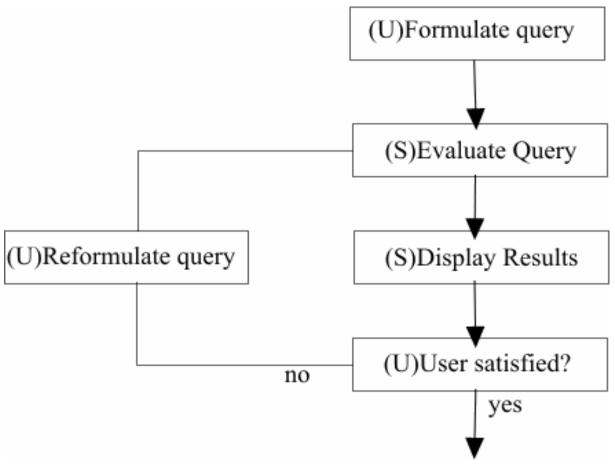


Figure 1: Interactive refinement in information retrieval

The traditional approach to patent retrieval is based upon database querying. Systems range from a simple full text retrieval based on an unique patent identification number to full blown SQL-type query input. In general, a query is formed by entering query terms into at least one field. Often, systems such as QPAT (`www.qpat.com`) allow the user to enter a single query within which query terms are assigned to specific fields using application specific codes. As an example one might enter:

```
'Philip'/inv and 'Canon'/pan
```

which says you want to search for "Philip" in the inventor field and "Canon" in the assignee field. Search results are returned as a list and are labeled by your

[1]QPAT, a patent search engine owned by Questel-Orbit

current query id. For example, if the above was our first query then the result set would be labeled `S1`. In QPAT you can then refine you search using previous search result sets. For example:

```
(‘‘high definition’’) AND S1
```

will search within the result set from the first query for all patents containing the phrase “high definition”.

Although this ability to save older result sets is useful, we believe it does not go far enough. We desire a system that can:

- enable the user to see how his query is performing
- give him an understanding of the patent space (see below)
- allow him to interactively change the parameters of his query (including those representing earlier query sets)
- quickly view different aspects of the result sets.

An interesting system, described in [5][14], allows users to visualize relationships between tables in a database. The queries are generated using a visual query environment and the user can choose how to represent the relationship (join) between the tables (views). The operations they allow include colouring a data object according to an attribute value, changing its size and filtering according to value limits. One very useful feature is the ability to interact with the data itself and generate input sets from direct interaction with the interface (see section 5). The system is built upon relational database data models and this is reflected in its interface. The user can for example navigate from the visual representation of a database object to other related objects and so the system is tightly integrated with the underlying database and its schema.

A slightly different application is described in [8] which proposes a set of techniques that aid the debugging of user interfaces. These *lenses* are similar to the lenses described in [2] which operate on application objects visible in the user interface (e.g.: magnifying lens, outline lens, drop shadow lens, achromatic lens). In general these lenses add information to the display, remove all but the selected information or make arbitrary changes to the display. The user places a lens over an object in the interface to modify the display of the object by, for example, revealing “hidden” features or performing some operation. In the former the lenses show debugging information and have additional controls that the user can interact with. We believe this is a useful concept which our architecture incorporates.

Another useful insight is described in the work by McGuinness and Manning (e.g.: [6]) and their “findUR” system. This uses background ontologies to assist in the search. We also use such an approach in the “international patent classification” module where we employ the IPC hierarchy to give the search module knowledge about the relationships between the query IPC and other IPCs in the classification scheme.

InfoCrystal, described in [16], is a tool for visually creating queries and viewing their results. The emphasis here is in displaying all possible combinations of the boolean query and allowing the user to examine subsets of the results from such a query. A *query spreadsheet* allows users to visualize all the possible boolean relationships among N variables and it allows users to specify *boolean* queries graphically. The visualization is based upon Venn diagrams but instead of using traditional geometric shapes to represent a set, a range of iconic descriptors are used. Each icon employs various features to encode information about the subset it represents such as the actual shape displayed, the proximity of one shape to another, the distance from the centre of the display. Furthermore colour, texture, orientation and size/brightness can be used to represent different features of the subset. As an example, if a query is *A and B and* C then the visualization will be a triangle with each vertex representing one concept; the results of the actual query will be displayed by an icon in the centre; results of other queries such as *(A and B) and (not C)* will be displayed by other icons in the appropriate positions. Each icon can be clicked on to create a new query. We see this approach as a possible useful extension to our work where we have concentrated on general interactive queries rather than the visualizations of all boolean combinations of a single query.

One of the most sophisticated software tools that is currently available is IBM’s Data Explorer [1] which allows users to visually construct complex data analysis programs. Once constructed, the program is executed in order to view the results. Application areas of this tool include oceanography, computational fluid dynamics and meteorology. Again we moved away from this “program/execute” organisation and concentrated on interactive querying. Our architecture will also accept any data type and each query module uses reflection to determine whether it can operate on the data coming into it.

We therefore propose a system that allows a user to build up a query consisting of *query-modules* each of which has an interactive user interface, a set of tools that allow a user to visualize the result set as a whole according to different algorithms and/or the attributes of the result set and the patents themselves. The framework surrounding the query modules should allow the user to easily change the query parameters and the query itself and allow him to save the query for later use on different data sets. Finally we also provide navigational tools such as *pan* and *zoom* modules which can also be added into the query. A more sophisticated module in this set would be something along the lines of the *fish-eye lens* [11].

The structure of the remainder of the paper is as follows: Section 2 describes our initial requirements and the architecture of the system and section 3 describes some query modules specific to the problem of patent retrieval. We then show some results and finish with a discussion on future work.

2 Architecture

In designing the system we took design rules from early user interface designers such as [3] and from the requirements of the people in our Intellectual Property department. We therefore had, briefly, the following initial requirements:

- Clear conceptual interface: The user should have a clear mental model of the system in order to effectively interact with it. It will allow the user to predict the behaviour of the system, to plan for novel tasks and to deal with error situations.
- Simple access to task interface: The system is designed to help users perform large difficult intellectual tasks as well as simple, routine tasks. The interface should allow the user to build these more sophisticated tasks easily and to be able to return to them.
- Support for browsing, search and reviewing[9]:
  - Browsing allows users to gain an overview of what is available and how it relates to its surroundings.
  - Searching is an information retrieval process where the items of interest are well defined but their location is not.
  - Reviewing occurs when items have been retrieved and it is necessary to examine the results in more detail.
- The ability to examine one's query history and see the results of modifying earlier query parts is important.

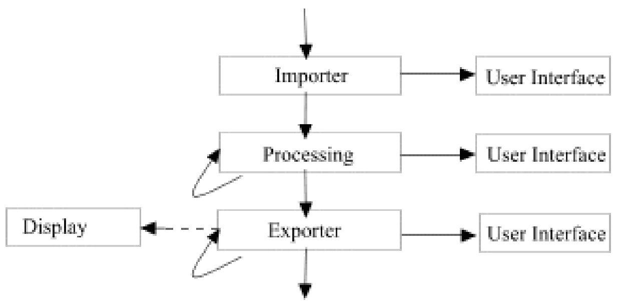
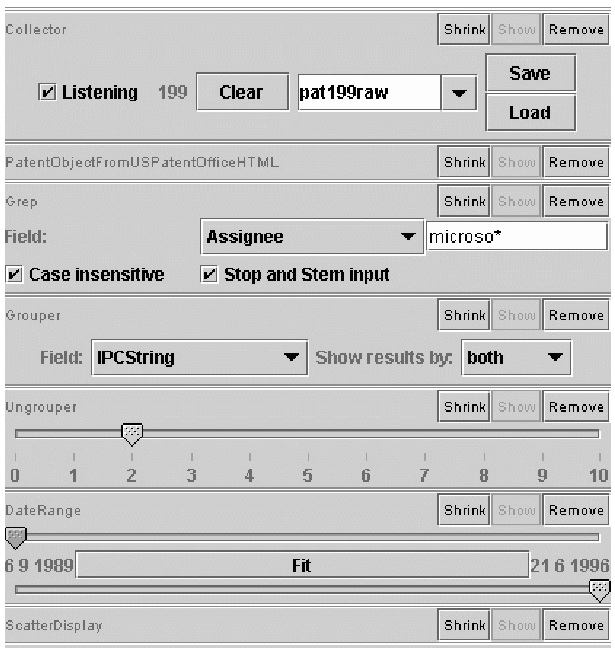


Figure 2: The architecture of our system.

- The ability to examine different views of the same data using different attributes of the data.
- The ability to save search queries for later use on different data.
- The ability to use application specific knowledge to improve the search (such as IPC hierarchy).
- The ability to interactively observe how additional query parts affect the result set.

After consultations with the IP department we designed a system that allowed users to visually build queries based on a stack metaphor and view the current results from the combined query on an adjacent window. We divide the query modules that can be used on the stack into three generic types: *import, processing, export.* There is one import module, one or more processing modules and one or more export modules (see figure 2) and a set of modules define an application. Import modules allow the user to bring data into the system, processing modules allow the user to apply some function to the data, either to each element or to the collection as a whole and export modules allow the user to display, view or save the resulting data and query stack. In general processing modules augment (e.g.: by adding keyword vectors) or change the data either within each element (e.g.: its location after being clustered) or to the collection as a whole (e.g.: by adding element-element similarity information) or they can remove elements from the collection that do not satisfy the modules criteria (e.g.: filtering on an IPC class). Each module defines its own user interface which is exported to the framework when the user instantiates the module. Instantiated modules are placed on the *stack* (see figure 3).

This architecture is a generic one and can accept any type of data. Indeed within the patent application itself we accept HTML data (containing patents)



Collector

✓ Listening 199 pat199raw

PatentObjectFromUSPatentOfficeHTML

Grep

Field: Assignee microso*

✓ Case insensitive ✓ Stop and Stem input

Grouper

Field: IPCString Show results by: both

Ungrouper

DateRange

6 9 1989 21 6 1996

ScatterDisplay

Figure 3: The stack.

from remote sites. The modules and system are designed so that every query module can be plugged into any other query module without giving exceptions or errors. If the particular module can not operate on the data coming into it, it simply ignores it and lets it pass through unchanged. In this way we give the user maximum flexibility in how he puts together his queries.

Since we develop query modules that can handle generic data, we need a mechanism to determine the attributes of the data that is flowing into the module. The module can then work out if it can operate on it or not. To this end we use Java Reflection which allows us to query the data objects entering the module. We can determine their accessor methods and the types returned. In the *keyword search* module described below this mechanism is used to dynamically build the user interface for the module. This module examines the different fields that the data object exposes and which can be searched on using regular expressions. This list is used to populate a *combo box* in the module's user interface. The user can then select one of the data object's attributes to search on. This filter is shown in figure 3 (second filter down) where the user is searching for "canon*" in the *assignee* attribute.

3 Filters for Patent Retrieval

We now describe some of the application specific patent filters we have written. A requirement for our system was the ability to automatically download a specific set of full text patents into the system. The particular patents are determined by a monthly patent watch service from which we obtain a file containing URLs to the patents at the online US patent office database. This file is read by an import filter which then automatically downloads the patent (in HTML format) corresponding to each URL. We convert this HTML document into a patent object using a series of regular expression queries. The output is then a collection of patent objects. Other import modules include reading data from a file, from a URL, and from an object database containing patent objects.

We wrote a set of processing filters for the patent applications, most of which we later generalised so that they can be used on other data. The only remaining application-specific filter is the IPC filter. When used this first of all loads in a hierarchy of IPC classes. Users can search for patents in a specific IPC class, or in that class and any below that class and can generate a simple boolean expression such as "G06F+ AND NOT G06F 011/00+" which says give me all patents in or below class G06F except for all those in or below class G06F 011/00. Figure 4 shows an example of such a query.

A slightly more sophisticated IPC filter allows the user to broaden his search using relationships between IPC classes discovered in a data set. We used the 1998 MicroPatent US Full Text data set and generated a set of correlations between IPC classes (from patents filed under more than one class). We can then recursively broaden our search by including classes related to the input class and limit the search by a maximum search depth and a relavence threshold (related to co-occurrance of the IPC classes). We demonstrate this in figure 5 where we have used a polar plot and colouring describe the relationship. The polar plot shows how each patent is related to the initial input term where the radial parameter corresponds to the number of IPC classes that patent is filed under and the angular parameter corresponds to the closeness (in edit-distance terms) to the input IPC class. Thus we can see different trails corresponding to patents filed under 1,2,3 or 4 IPC classes. The patents at the anti-clockwise end of each trail are closer to the input class which is highlighted using the colour shading.
We can combine the IPC module with a term searching module. In figure 6 we have added a module to

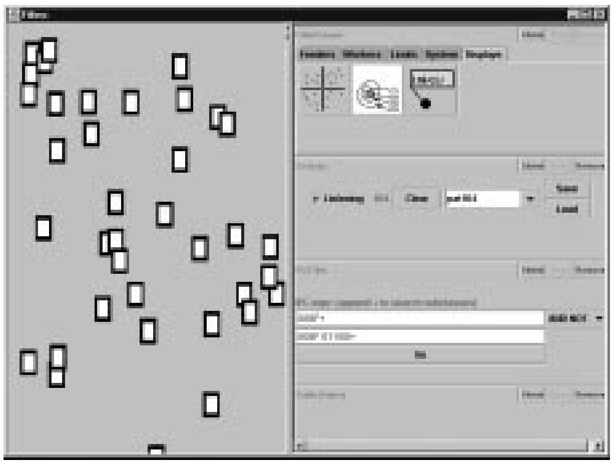


Figure 4: An example of searching for patents under specific IPC classes

search for the word "machine" in the assignee field. We thus end up with only those patents that Canon have filed in the above IPC classes. Remember that every module is *active*: the user can easily modify the IPC term and view the new results. [ht]

4 Results

This section demonstrates some of the queries and results that we typically use in the patent retrieval application. As shown above we can search on keywords and IPC classes and we have also implemented several filters for determining and visualizing document similarity. In order to compute document similarity we need to do some more processing on the data set. The basic keyword generation module creates a keyword vector for each patent based on term frequency and inverse document frequency[7]. We can then compute document similarity using this vector according to traditional vector cosine algorithms. We implemented a multi-dimensional scaling algorithm[10], [15] which uses these vectors to display on a 2D map the document similarity. A more sophisticated keyword generation module uses *wordnet* to compute the keyword vector. This time we determine the synset of each term and use the *information value* of that term to determine its discriminatory power. Our vector is composed of the top N synsets – a more useful set as we can now combine words with similar senses (eg: house, dwelling) into one element.

Another clustering module uses a statistical approach based on tri-grams. We compute a vector based on the frequency of occurance of tri-grams in the full text of a document. We input the re-

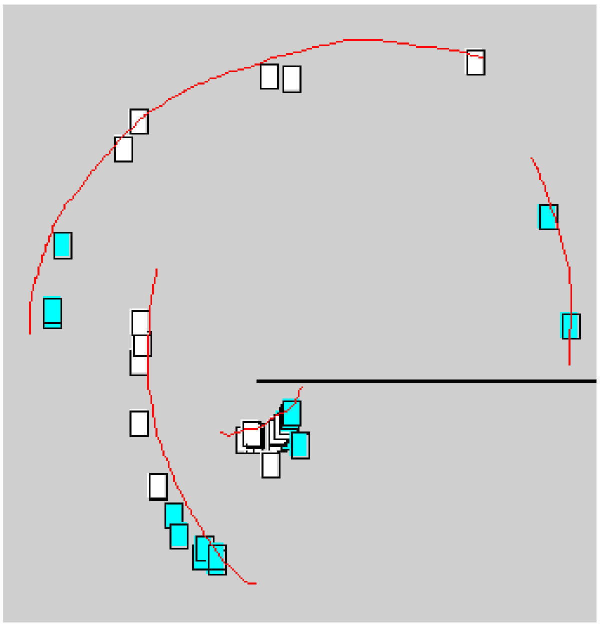


Figure 5: Example showing IPC broadening and polar plot.

quired number of clusters and iteratively minimize the "global distortion". Each cluster is associated with a centroid which is simply the mass center of the cluster. We iteratively move points from one cluster to another until the decrease in distortion per interaction falls below some tolerance. The global distortion is $D = \sum_j \min_i(\|x_j = c_i\|^2)$. Figure 7 displays the results of this filter. We also show in figure 8 all the patents filed by Canon in the last month. We have clustered them using the tri-gram approach on their full text and then coloured them by IPC class. This shows an interesting feature of the data set for we have discovered patents filed under the same IPC can have suprisingly different content. In the figure, the single patent in its own cluster is about user interfaces and layout but has the same IPC class as one other patent in the main cluster that is about operating system extensions. Another example is shown in figure 9. Here the user has been looking for the filing history of a particular person and has been interested in the companies he filed those patents for. In the query stack we have asked for all patents filed by inventor "Kim" (note: in this example we are using surname only[2]). We then colour the results using the assignee attribute and ask for this set to be displayed in a time line. We can see that the inventor has filed three patents at "Daewoo" in the area of image processing. He (or

[2]We have not yet implemented a search on full names which would clearly improve this search.

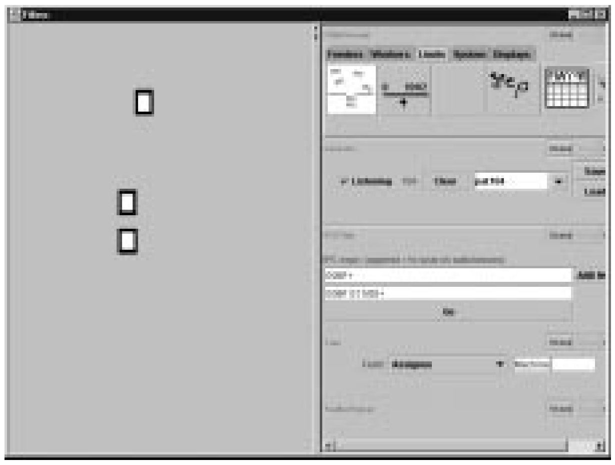


Figure 6: Here the user has first searched for all patents in class G06F and below and then refined the search using a keyword search in the assignee field.

other Kim's) have filed patents at a later date at several other companies, two more of which are related to image processing. From this result set it would be easy to determine if the "Kim" at Daewoo moved to another company and continued his work there. Another way of viewing time data is through the date filter. In figure 10, we show a different example where we use the sliders on the date filter to examine when patents were filed by Canon. We have also coloured the data with the IPC attribute. As we move the date filter through time we see that their two patents in class G06F were filed last and were filed within 6 months of each other.

5 Future Work

In order to further this prototype, the most important aspect that we now need to do is obtain feedback from the IP engineers. We would hope to gain knowledge of particular sets of queries that are the most used for the type of searches they perform. Short-cuts or "macro-modules" could be used to improve search efficiency. Another important aspect we would like to test is the usability of the system in terms of its flexibility. Users can plug any module into any other even if this makes no sense. There are no errors, the module that does not understand the data coming into it simply lets it pass through. This has the obvious disadvantage of confusing the user but we don't want to build in connection constraints as this may prevent a user from using the system in a novel way. However, it will be necessary to provide some sort of help more sophisticated in nature than our basic help module.

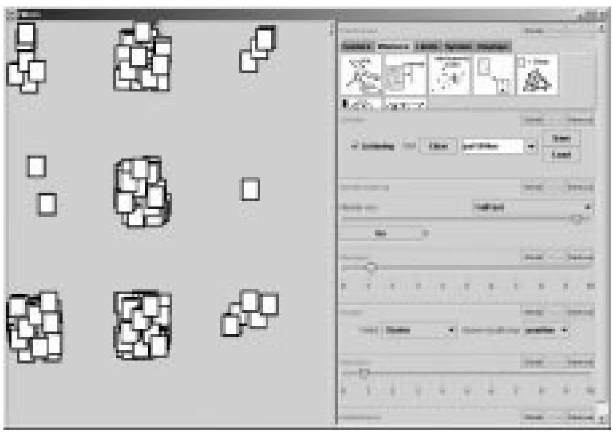


Figure 7: The tri-gram clustering algorithm. It is interesting to note that the cluster with one member is a patent on chromosones and the cluster with 3 members relate to electronic communication (mail or notes). There are no other patents of this nature in the dataset.

A starting point may be [12] who describe a method of building up a query based on goals, decision nodes and retrieval nodes. Background knowledge sources enable the system to perform some inference and determine what to retrieve. We have several sources of knowledge particular to the patent retrieval application, such as IPC hierarchy, examiner decisions, and company information that we could employ to extend our system for more intelligent retrieval.

One assumption that we have made is that the initial patent data set is relatively small (of the order of a few thousand patents). We assume that an initial general query to a traditional database will provide the context (or initial patent search space) in which this system will operate. We have used the example of our monthly patent search results; others include an assignee (e.g. searching for all Microsoft patents on QPAT returns 1303 patents), searching for (high ADJ definition OR hd) and (television OR TV) in the abstract returns 292 and 5250 in all fields. If this turns out not to be sufficient for general use then it will be necessary to work on the scalability of our solution. This shouldn't be a huge problem as the modular nature and data-flow pipeline lends itself to parallel processing techniques.

We would also like to increase our library of document analysis modules. As an example a Latent Semantic Indexing module such as that presented in [4] would be a useful addition. In it a method for visualizing document spaces and co-citation networks is

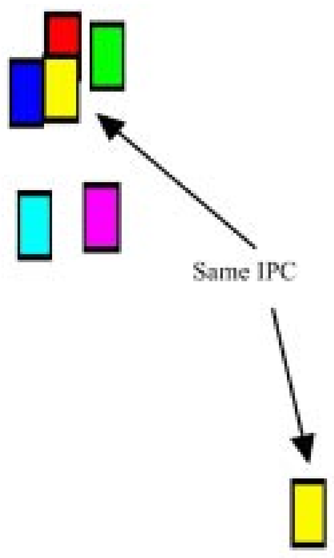


Figure 8: The results of the examination of Canon patents filed in the last month.

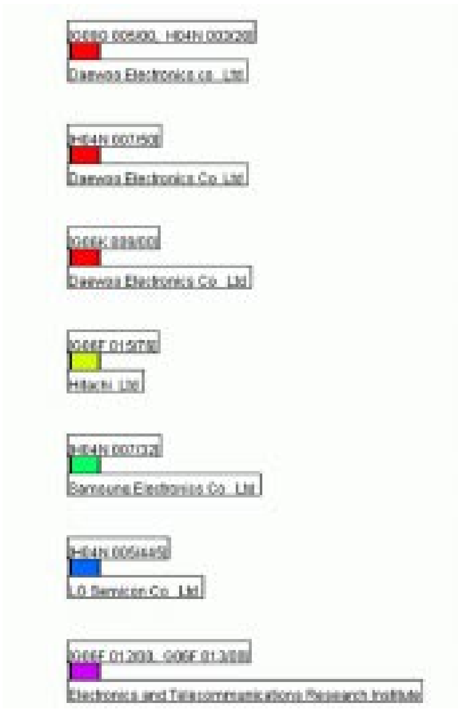

G06Q 005/00, H04N 003/20
Daewoo Electronics co. Ltd
H04N 007/50
Daewoo Electronics Co. Ltd
G06K 009/00
Daewoo Electronics Co. Ltd
G06F 015/78
Hitachi, Ltd
H04N 007/32
Samsung Electronics Co. Ltd
H04N 005/445
LG Semicon Co. Ltd
G06F 01.203, G06F 01.3/00
Electronics and Telecommunications Research Institute

Figure 9: A search for patents filed by a particular inventor over time showing the company he was working for at the time of invention.

presented and LSI is used as a way of reducing the dimensionality of the document space. A pathfinder network algorithm is applied, originally developed for the analysis of proximity data in psychology, to reduce the complex network of proximity data to a simple network where only the most important links in the network are preserved.

An area of improvement is to introduce a cluster description so that upon viewing the result of a clustering algorithm the user can browse cluster properties which are determined by that cluster's elements. In [13], Pirolli describes his "scatter/gather" browser where documents are clustered according to their similarity (here, normalized correlation of their word-frequency vectors) and these clusters of documents are represented by "meta-documents" which contain profiles of topical words and the most typical titles. This system does not improve the retrieval itself but gives users a greater understanding of the structure of an information database. We could use patent specific information to improve the description.

6 Conclusions

We have presented work describing a novel architecture and user interface for patent queries and visualization of query results. We have built a prototype for patent retrieval and visualization, based on the notion of *integrating fully* retrieval with interaction, that extends the functionality of existing patent retrieval systems by enabling interactive querying and browsing of the patent space. It is a simple matter to obtain different views of your search results.

The architecture is very general, will accept any data type, allows arbitrary controls for a query part and allows users to develop their own query sets and different query modules (such as more sophisticated clustering/analysis algorithms). It is simple, for example, to write a module that analyses a dataset using a Kohonen network to display the contents of a document space. The (software engineering) interface of this module is simply one function "filterData" that accepts a collection of generic data objects and returns a new collection of generic data objects. The user must also write the GUI for his module.

7 Acknowledgments

I would like to thank Dr. Martin Portman for his ideas and work on the filters architecture, Alex Dodge and Bill Imlah for their help defining the patent application.

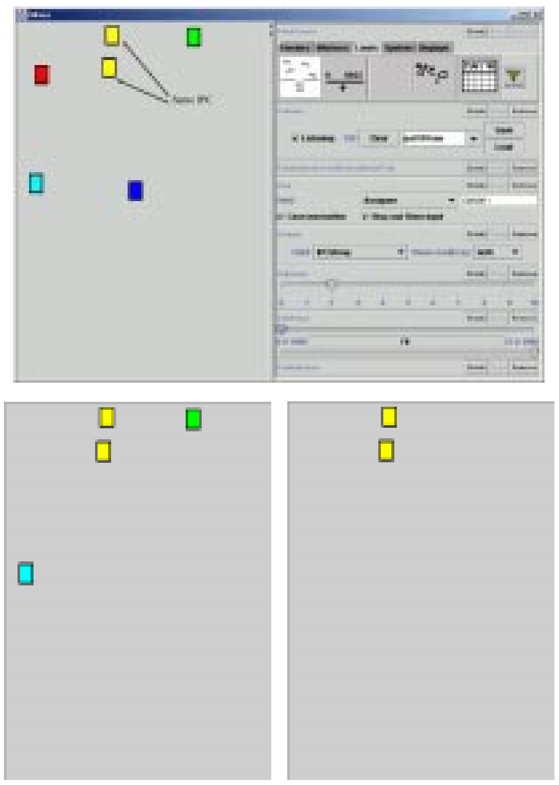


Figure 10: The date filter showing when patents were filed.

References

[1] In *Proceedings of the 1996 IBM Visualization Data Explorer Symposium*, October 1996.

[2] E. Bier, M. Stone, W. Buxton, and T. DeRose. Toolglass and magic lenses: The see-through interface. In *SIG-GRAPH*, pages 73–80, August 1993.

[3] S. Card and T. Moran. User technology: From pointing to pondering. In A. Goldberg, editor, *A History of Personal Workstations*. ACM Press, 1988.

[4] C. Chen. Visualising semantic spaces and author co-citation networks in digital libraries. *Information Processing and Management*, 35:401–420, 1999.

[5] M. Derthick, J. Kolojejchick, and S. F. Roth. An interactive visual query environment for exploring data. In *Int. Conference on User Interface Software and Technology*, pages 179–187, October 1997.

[6] D.McGuinness, H. Manning, and T. Beattie. Knowledge assisted search. In *Int. Joint Conf. Artificial Intelligence*, August 1997.

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

[8] S. Hudson, R. Rodenstein, and I. Smith. Debugging lenses: A new class of transparent tools for user interface debugging. In *Int. Conference on User Interface Software and Technology*, pages 189–198, October 1997.

[9] G. Jorna, M. Wouters, P. Gardien, H. Kemp, J. Mama, I. Mavromati, I. McClelland, and L. V. Matzen. The multimedia library: The centre of an information rich community. In *Human Factors in Computing Systems Design Briefings*, March 1997.

[10] J. B. Kruskal. Multi-dimensional scaling by optimizing goodness-of-fit to a nonmetric hypothesis. *Psychometrika*, 29:1–27, 1964.

[11] J. Lamping, R. Rao, and P. Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Int. Conference on Human Factors in Computing Systems*, pages 401–408, May 1995.

[12] C. Lee and Y.-T. Chen. Distributed visual reasoning for intelligent information retrieval on the web. *Interacting with Computers*, 12:445–467, 2000.

[13] P. Pirolli. Computational models of information scent-following in a very large browsable text collection. In *Int. Conference on Human Factors in Computing Systems*, pages 3–10, March 1997.

[14] Steven Roth, John Kolojejchick, and Jade Goldstein. Interactive graphic design using automatic presentation knowledge. In Mark T. Maybury and Wolfgang Wahlster, editors, *Intelligent User Interfaces*, pages 237–242. Morgan kaufmann, 1998.

[15] R.N. Shepard. The analysis of proximities: Multidimensional scaling with an unknown distance function, i and ii. *Psychometrika*, 27:125–140, 219–246, 1962.

[16] A. Spoerri. Infocrystal: a visual tool for information retrieval management. In *Proceedings of the Second Int'l conference on Information and Knowledge Management*, November 1993.