A Patent Search and Classification System

Leah S. Larkey
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, Mass 01003
larkey@cs.umass.edu

ABSTRACT

We present a system for searching and classifying U.S. patent documents, based on Inquery. Patents are distributed through hundreds of collections, divided up by general area. The system selects the best collections for the query. Users can search for patents or classify patent text. The user interface helps users search in fields without requiring the knowledge of Inquery query operators. The system includes a unique "phrase help" facility, which helps users find and add phrases and terms related to those in their query.

Keywords

Digital libraries, systems, information retrieval, text categorization, classification, patents, applications.

1. INTRODUCTION

At the Center for Intelligent Information Retrieval (CIIR) at the University of Massachusetts we are working with the U.S. Patent and Trademark Office (USPTO) on a project involving the retrieval and classification of U.S. Patent texts, patent images, and trademark images.

This paper describes a web-based system for the retrieval and classification of patent text that we have implemented for the USPTO. The intended users for the system are patent examiners and classifiers at the USPTO. The goal is to partially automate two aspects of the routine work done by patent office personnel: (1) searching for prior art, that is, finding existing patents related to a claimed new invention and (2) assigning the appropriate patent class and subclass to patents to be issued.

Notable features of the system include:

- Large, Distributed Collection. The collection has about 1.5 million patents, about 55 Gigabytes of raw data, distributed across 400 collections.
- Collection Selection. Collection selection technology chooses the best collections for a query so only a limited number of collections have to be searched.
- Fields. 50 fields are indexed, so users can search for patents by a particular inventor, or assigned to a particular company, etc.
- Choice of query syntax. The user interface allows users to search via "natural language" queries, or boolean and field operators, or form fill-ins to do field searches and boolean combinations of them.
- Choice of text sources. A user can enter queries by typing into the form, by browsing and selecting text from an ascii file, or selecting a patent from the collection to use as a query.
- Choice of actions on text. The system can search or classify the query.
- Phrase Help. A unique "phrase help" facility provides users lists of phrases and terms related to query phrases, and allows them to choose items from the list to add to their query.
- Phrase and Compound Handling. Automatic processing of queries to handle compounds and phrases in natural language queries.

In what follows we will first describe U.S. patent documents and the USPTO's patent classification scheme. Next, we will describe the user interface and query modification our system carries out. Finally, we will describe the search and classification components of our system.

2. U.S. PATENTS

There are over 5 million U.S. patents, consisting of 100-200 gigabytes of text. (There are also more than 40 million pages of bitmap images, making up 4-5 terabytes of data which we are not searching here). The multidatabase system we describe here includes 1.5 million patents from 1980-1996. This is a little over one fourth of all U.S. utility patents, and fills about 100 gigabytes, 55 of text and 45 of indexes. Our system also includes a smaller single database covering two years of patents, 1995 and 1996, consisting of around 220,000 documents and about 16 gigabytes in text and indices.

Patents range in size from a few kilobytes to 1.5 megabytes. They are represented in Greenbook format [11], which tags hundreds of fields at two levels. We represent about 50 of these fields. A large number of these fields are small and not text-like, containing information like application number, patent number, dates of application, of issue, number of figures. Another large number of fields are small and contain specific pieces of text information, like the names and addresses of the authors, assignees, patent examiners, and patent attorneys. There are a few large narrative text fields, which dominate the influence on natural language queries:

- Abstract
- Background Summary

- Detailed Description
- Claims

As in many other real-world classification and retrieval domains, patents present a severe vocabulary mismatch problem. Patents or patent applications about similar inventions can contain very different terminology. To compound the problem, some inventors intentionally use nonstandard terminology so their invention will seem more innovative and to prevent search systems from finding prior art. The claims in a patent are written in legalistic language and can be quite different in style from the rest of the patent. Idiosyncratic legal styles and terminology can lead to spurious similarities between patents based on style rather than content.

2.1 U.S. Patent Classes and Subclasses

U.S. Patents have been manually classified by the USPTO into a scheme containing around 400 classes and around 135,000 subclasses. The classes and subclasses form a hierarchy, with subclasses of subclasses of subclasses, etc. The tree goes as deep as 15 levels, but the depth varies greatly. In some domains there is only one level of subclasses below a class, and in many places there are only three or four levels. Subclasses at any level can be assigned to patents. That is, even if a subclass has subclasses of its own, the parent subclass can be assigned to a patent. The patents that USPTO personnel place into higher level, more general nodes in the subclass tree, tend to be unusual patents that don't fit well into a more specific subclass.

A patent belongs to one class/subclass called its *original reference*. In addition, it can have cross-references to other class/subclasses. The average patent has three cross-references. In the present system, we are attempting only to place patents into their unique original reference subclass. The techniques we use to classify patents are discussed in the Classification section below.

Class	Description
2	Apparel
4	Baths, Closets, Sinks, and Spittoons
5	Beds
7	Compound Tools
8	Bleaching and Dyeing: Fluid Treatment and Chemical Modification of Textiles and Fibers
12	Boot and Shoe Making
14	Bridges
15	Brushing, Scrubbing, and General Cleaning
16	Miscellaneous Hardware
19	Textiles: Fiber Preparation
23	Chemistry: Physical Processes
24	Buckles, Buttons, Clasps, etc.
…	…
395	Information Processing System Organization
396	Photography
399	Electrophotography
…	…

Table 1: A sample of patent classes

Table 1 shows a small part of the list of 400 patent classes. Table 2 shows some of the subclasses of one of those classes. In Table 2, hierarchical level is indicated by indentation. Note that the subclass numbering scheme does *not* reflect the hierarchical relations among subclasses. In our classification research (reported elsewhere [7]), we have been focusing on these speech-related subclasses of class 395, *Information Processing System Organization* because they are particularly difficult. In the current system, however, we use all 135,000+ subclasses.

The set of classes and subclasses is dynamic. A subclass can contain up to 2000 patents. The patent office tries to keep it down to a maximum of 200 by making new subclasses. In fact, the vast majority of subclasses have fewer than 20 patents in them, which makes training classifiers difficult. New inventions require the continual creation of new subclasses. Periodically, the PTO carries out a reclassification. Sometimes existing classes are subdivided into new subclasses. Sometimes a set of subclasses of a class are merged together, then subdivided again in a different manner. After new subclasses are formed, the patents involved may or may not be assigned to the new subclasses.

2.090 SPEECH SIGNAL PROCESSING
- **2.1 For storage or transmission**
  - …
- **2.4 Recognition**
  - 2.41 Neural network
  - 2.42 Detect speech in noise
  - 2.43 Normalizing
  - 2.44 Speech to image
  - 2.45 Specialized equations or comparisons
    - 2.46 Correlation
    - 2.47 Distance
    - 2.48 Similarity
    - 2.49 Probability
    - 2.50 Dynamic time warping
    - 2.51 Viterbi trellis
  - 2.52 Creating patterns for matching
    - 2.53 Update patterns
    - 2.54 Clustering
  - 2.55 Voice recognition
    - 2.56 Preliminary matching
    - 2.57 Endpoint detection
    - 2.58 Subportions
    - 2.59 Specialized models
  - 2.6 Word recognition
    - 2.61 Preliminary matching
    - 2.62 Endpoint detection
    - 2.63 Subportions
    - 2.64 Specialized models
      - 2.65 Markov
      - 2.66 Natural language
- **2.67 Synthesis**
  - …
- **2.79 Application**
  - …

Table 2: Some subclasses of class 395

Because of the dynamic nature of the subclass system, we ignore the subclass information inside the Greenbook format documents, and refer instead to an associated set of data files from the USPTO. These files can be easily replaced with updated versions in order to keep the subclass system current, without changing the data in the collections of patents.

3. SYSTEM ARCHITECTURE

The system has a client/server architecture, with the data and server software currently residing on a 4-processor Sun Ultra-Enterprise running Solaris 2.5. The data consist of 401 collections of patents. One collection (the single database system) contains all the patents from the years 1995 and 1996. The other 400 collections (the multidatabase system) cover the years 1980-1996, divided up according to patent class. Different collections could be at different sites, but they happen to be on the same server at present. The Inquery client software (including the web server) is also currently running on a Sun Solaris system. The user interface runs in a web browser (Netscape or Internet Explorer), communicating with the web server via CGI (Common Gateway Interface).

The user interface is written in Javascript and dynamic HTML. The underlying search, collection selection, and classification engines are written in C. The CGI portions of the system are written in C, Perl, gawk, and shell scripts.

4. USER INTERFACE

The system includes a user interface that allows users to search for patents or find subclasses in several different ways. Figure 1 shows the main screen of this interface. First, the user can issue a query in so-called *natural language*, for example, “I want technology that parents can use to control television content,” or “energy-efficient windows.” Users can also enter a query using Inquery query operators, such as:

scroll bar #field(INVT Smith) #field(ASSG Microsoft)

to look for patents mentioning scroll bars assigned to Microsoft, invented by Smith. The same query could otherwise be entered using field fill-ins as shown in Figure 1. Other alternatives are to enter a patent number or to take text from a file, via a browsing mechanism.

The main screen in Figure 1 is what the user sees when the system starts up, providing for a standard search. The user can click radio buttons to choose to search the single database for the years 1995 and 1996, or the multiple database system covering 1980-1996. The tabs along the top allow them to select alternate ways to search. Advanced search is like standard search, except that the screen shows a larger number of specific fields. The patent number tab provides a form for the user to enter a particular patent number.

To enter a query, the user either types the query into the text box labeled “enter query below,” or clicks “load from file” to browse for a file. They can further constrain the query by specifying terms in the illustrated field fill-ins. They can select various operators for the field fill-ins, *prefer*, *require*, or *reject*.

Underneath the field fill-ins is a pop-up menu for the user to select how many documents they want displayed at a time. The default is 10.

Once a user enters a query by any of the above means, any of three actions can be taken. They can either search for related patents, or attempt to find the correct patent subclass for the query, or they can request “phrase help,” to help them get additional phrases for their query.

4.1 Query Processing

Recent research on automatic query expansion has shown slight improvements, on average, when terms related to query terms are automatically added to queries. For individual queries, however, these techniques help in some cases and hurt in others [10][15]. In the present system, we decided to take a conservative approach to query expansion and provide two separate query expansion components, one automatic and one user-guided, as shown in Figure 2.

Figure 1: Main Screen

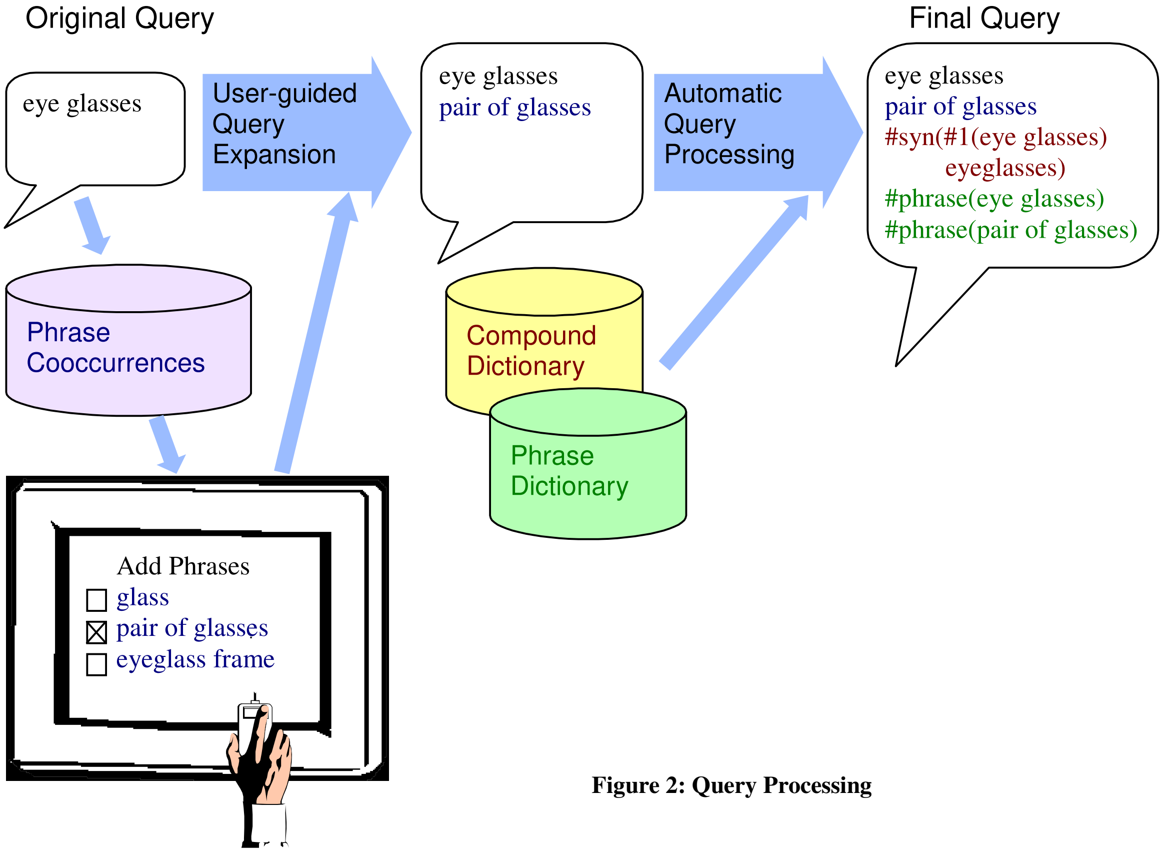

Figure 2: Query Processing

The automatic component adds only phrases and compounds whose terms are already in the query. The user-guided component presents a wider-ranging set of additional terms and phrases related to the query, but only adds them to the query if the user explicitly selects them.

Both classes of additions depend upon data structures in which information about phrases and their cooccurrences have been precompiled, as explained below.

4.2 Automatic phrase and compound processing

Automatic additions take place only if the query in the query window contains no explicit Inquery operators. The system consults a compound dictionary and a phrase dictionary. If the query contains a word sequence like *eye glass* that can be found in the phrase dictionary, the phrase is added to the query, e.g. **#phrase (eye glass)**. The operator **#phrase** gives a higher score to documents in which the terms occur in proximity. If the word sequence can be found as a single word in the compound dictionary (*eyeglass),* then the compound is added to the query inside a synonym operator, indicating that either the compound or phrase form would satisfy it. The right hand side of Figure 2 shows this processing.

The phrase and compound dictionaries were built automatically from patent text, using a set of heuristics aimed at finding noun phrases. First, a large sample of text was extracted from the corpus of 17 years of patents, consisting of titles, abstract, part of the background summary, and the claims from each patent. The text was segmented wherever items from a special list of delimiters were found. The delimiters included stopwords, punctuation, irregular verbs, company names, auxiliary verbs, and many other categories. The terms in the resulting sequences were assigned parts of speech using WordNet [5]. The sequences were retained as phrases only if they satisfied rules defining noun phrases, and met certain other criteria.

The dictionary of compounds was made from the phrase dictionary by checking every two-word phrase and every hyphenated term. If the combined form (without space or hyphen) was in the WordNet dictionary, the combined form was added to the compound dictionary.

4.3 User-guided Phrase Help

User-guided phrase help was added at the request of the patent office. They were interested in a facility that would suggest class-specific related phrases for a phrase or term that the user typed. If the user clicks the *Phrase Help* button on the main screen, a phrase help window appears, showing the patent class the system selected for the phrase. (Class selection is based on the collection selection algorithm described below and can be overridden by the user). The user can choose to see phrases *containing* their query phrase, or phrases *associated* with their query phrase. Check

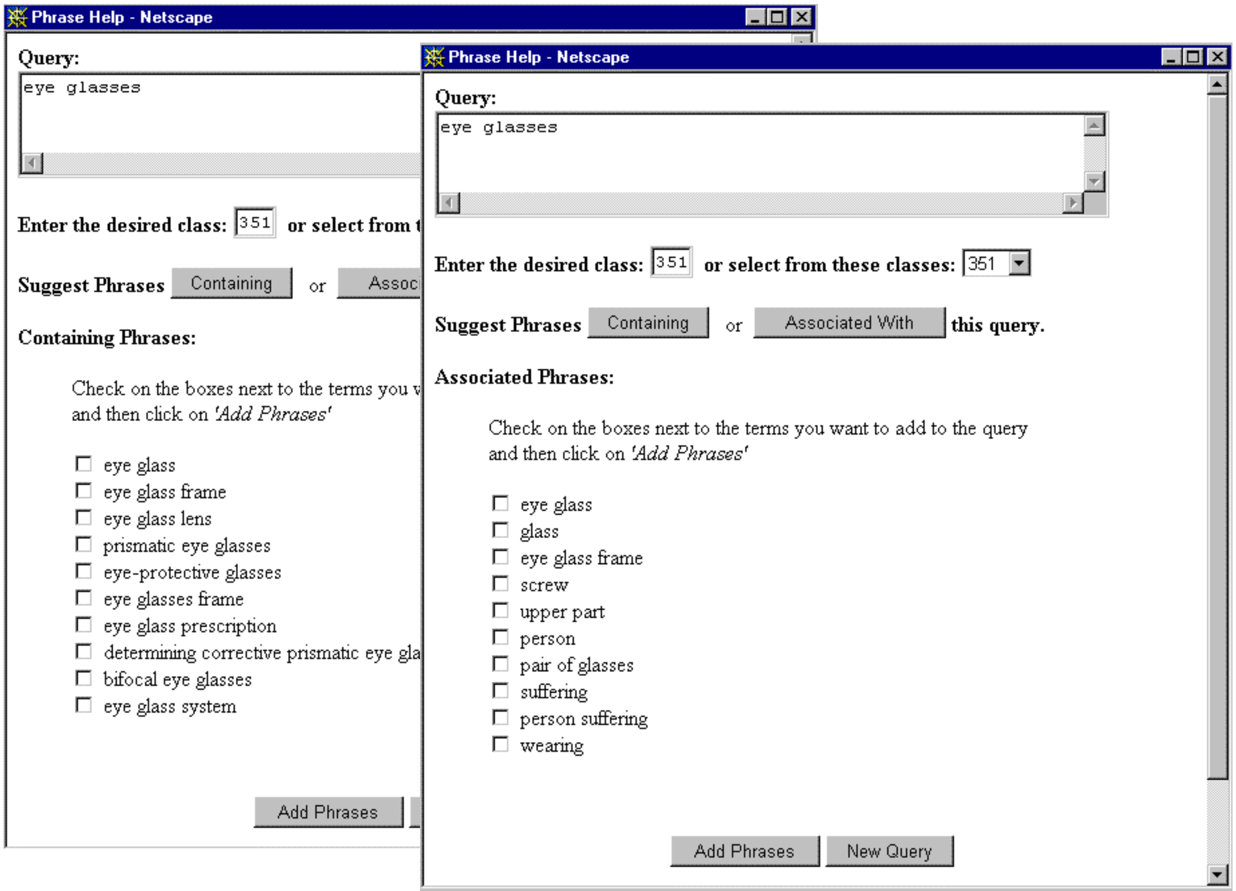

Figure 3: Phrase Help Screens

boxes allow them to choose phrases from the list to add to their main query. Examples are shown in Figure 3.

These lists come from a set of data structures made offline. These structures allow us to access any phrases containing a query phrase, and any phrases cooccurring with a query phrase. The phrase cooccurrence data structures were built for each class as follows. Any phrases from the global phrase dictionary described above and any single terms and subphrases in those phrases were candidates for inclusion. The phrases were subjected to "simple stemming," which combined upper and lower case, hyphenated and non-hyphenated forms, and singular and plural forms. Each patent in the class was divided into three sections: (1) title and abstract (2) background summary and (3) claims. We accumulated a list of all the phrase pairs that occurred together in a document section and a count of how many times the pair occurred. A cooccurrence score was then computed for each pair:

$$\frac{C_{ab}\text{-}O_a O_b / N}{O_a + O_b}$$

where C_{ab} is the number of occurrences of the phrase pair in the class, O_a is the number of occurrences of one of the phrases in the class, O_b is the number of occurrences of the other phrase in the class, and N is the number of passages (sections) in the class.

Cooccurrence information was stored in Inquery databases of pseudo-documents. The title of each pseudo-document was a phrase, and the body text of each pseudo-document was a list of phrases that cooccurred with the title phrase, provided the pair met a frequency of occurrence threshold and a cooccurrence score threshold. This implementation allowed us to find all the phrases related in two different ways to a query phrase: phrases containing the query phrase, and phrases cooccurring with a query phrase.

After the user has formulated a query and expanded it with help if desired, they are ready to search for patents, or classify the text of the query. (Classifying a query may seem strange, but the subclass found in this way can provide an alternate way to find patents relevant to a query.)

5. PATENT SEARCH

Both retrieval and classification components use Inquery, a probabilistic information retrieval system based on Bayesian networks that uses *tf·idf* weighting [2]. Inquery can take structured queries, as exemplified in section 4. In addition to the **#field** operator shown in that example, there are operators available to specify Boolean operations, proximity and other constraints. The system allows users to search either the single collection containing the 200,000 patents from 1995 and 1996, or the larger set of 1.5 million patents from the years 1980-1996. This set is distributed across 400 collections, divided up by patent class.

In searching a single collection, the query is submitted to the server. The requested number of documents is returned and displayed in a ranked list, according to their *tf·idf* scores. The display includes links that the user can click to see parts of the text of the patent.

5.1 Indexing Individual Collections

Searching by fields is made possible by the manner in which the collections are indexed. Around 50 of the more important Greenbook tags were processed to mark fields in the patents. In addition to the usual index which stores the locations of each term in the text, each collection also has a field index. The field index stores information about terms in fields, so a user can look for a query term (e.g. a name) in the Inventor field, for example. We also took the standard steps of eliminating words on Inquery's standard 418 word stopword list, and combining related forms by stemming the remaining words using the *kstem* stemmer [6].

5.2 Collection Selection

A search of the larger corpus is really a search of 400 different collections. It would be too costly and time consuming to actually perform a search on each of 400 collections. Instead, we use *collection selection*, a technique for selecting the best collections for a query.

Collection selection has grown out of the need to search large numbers of collections distributed across networks [3][12]. However, the techniques are useful in dealing with large numbers of collections whether they are distributed across many sites or at a small number of sites. Collection selection attempts first to rank collections according to their relevance to a query in much the same way that documents are ranked according to relevance. Then, documents are retrieved from only a small number of the top-ranking collections.

This ranking of collections requires that a collection selection index be constructed, in which each collection is treated as a virtual document. The virtual document lists each term in the collection with a count of how many documents in that collection contain the term. The virtual documents are indexed as if they were actual documents.

In searching the large corpus of 400 collections, the query is submitted to collection selection, which ranks the collections in order of relevance to the query. Then documents are retrieved from the best 10 collections via the same query, and ranked. Figure 4 illustrates this two-stage search using collection selection.

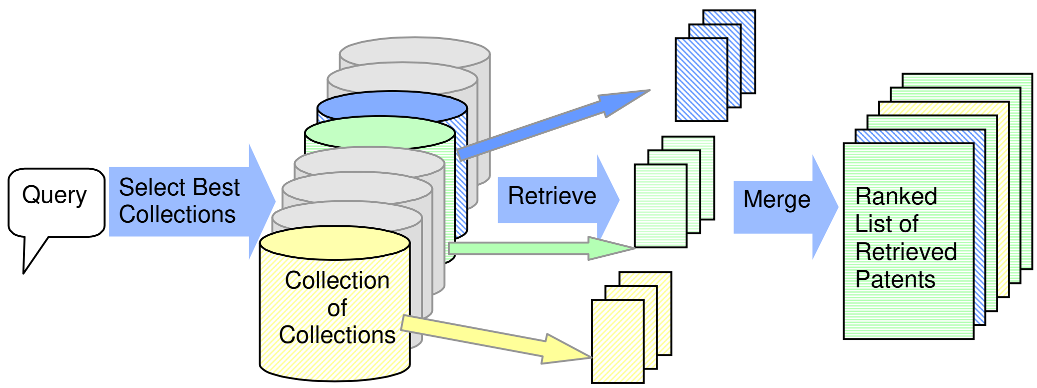

Figure 4: Collection Selection

6. CLASSIFICATION

The classification part of the system uses a *k*-nearest-neighbor algorithm, in much the same form as we have used it in our research on assigning of diagnostic codes to patients' medical records [9]. In general, the *k*-nearest-neighbor algorithm assigns a category to an item by computing a distance (similarity measure) between the item and a corpus of items of known category. It assigns the new item to the majority category among the closest *k*

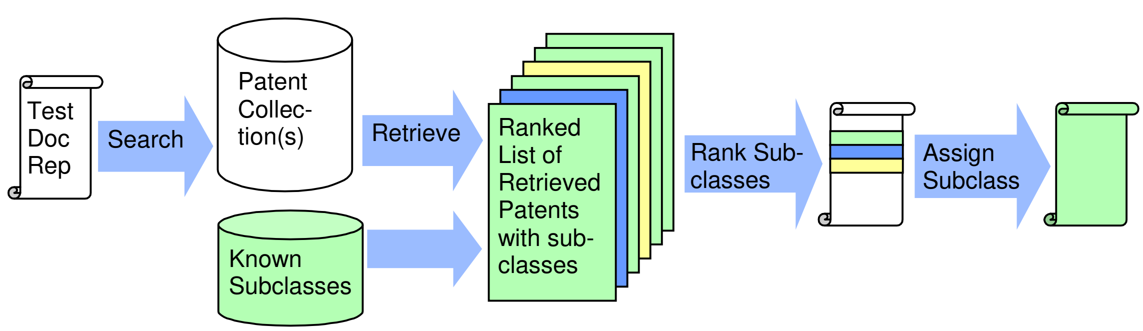

Figure 5: *k*-nearest-neighbor classification

known items [4]. Various refinements can be used to avoid ties and take into account the ranks and distances to the nearest neighbors. We use *k*-nearest-neighbor because it does not require much training up front, and because it scales up well from small to large data sets [13]. In the case of our patent system, the distance metric is Inquery's belief score.

After a query is formulated from a document to be classified, the initial stages of *k*-nearest-neighbor classification are identical to those for searching. The query is submitted to Inquery. The retrieval engine returns a ranked list of documents and scores. Rather than simply counting the number of neighbors in each subclass, we sum the scores of the neighbors in each subclass, and then rank the subclasses by this sum. We then assign the top-ranking subclass to the test document. This process is illustrated in Figure 5.

The user sees a ranked list of patent subclass codes and the name of the subclass.

6.1 Representation of Patent Documents for *k*-nearest neighbor classification

A crucial component of *k*-nearest classification of patent text is the formulation of a query from a patent or patent application.

One example of a query made from a patent for a motorcycle theft alarm can be seen in Figure 6. It illustrates the use of two Inquery operators, **#wsum**, a weighted sum, and **#1**, a proximity operator requiring that terms occur adjacent to each other.

#wsum (1 11 alarm 10 switch 10 horn 10 device
6 motorcycle 6 kickstand 5 vehicle 5 button 4 lock
4 invention 4 circuit 4 battery 3 theft 3 require
3 cycle 3 close 2 weight 2 warn 2 usually
5 **#1**(kickstand switch) 5 **#1**(horn button)
5 **#1**(alarm device) 4 **#1**(lock switch) 3 **#1**(theft alarm)
3 **#1**(cycle theft alarm) 3 **#1**(cycle theft))

Figure 6: A Query Formed from a Patent

We have been investigating the following choices in converting the document to such queries:

- What part of the patent to use,
- whether features should be single terms only, or terms and phrases,
- how to determine which terms (or phrases) are the best ones,
- how many terms or phrases to include,
- how to weight the features in the vector,
- how to discover and represent the relative importance of different sections of the document.

In our previous work classifying patient medical records [9] and student essays [8], we used the entire test document as a query. For classifying medical records we used Inquery operators to differentially weight different sections of the document. For patents we do not use the entire document, or even entire sections, because many of them are too large and full of detail that does not aid classification. Instead, the system selects certain sections (fields) and portions of sections, then removes stopwords and stems the remaining terms, as in indexing the documents. Then a vector of terms and phrases is made from the reduced document, and assign term weights that reflect both the relative importance of the different sections the terms come from and the term frequency in those sections.

The weights on features (stemmed terms and phrases) depend upon what section of the patent it came from, and how many times it occurred in that section. A weight for the section is multiplied by the number of occurrences of the feature in the section to get a per-section feature weight; then the weights for that feature are summed across sections. The features are then ranked by this weight, and a threshold (maximum number of terms) is applied to retain up to the threshold number of terms which have a weight of at least 2.

When phrases were included as features, they were chosen as follows. First, part-of-speech tags were assigned to the original document via the *jtag* tagger [13], and any noun phrases were flagged as potential phrases. As with the single terms, each phrase received a weight consisting of the section weight multiplied by the number of occurrences of the phrase in that section, and the weights for each phrase were summed across sections. The phrases were ranked by this weight and a threshold (possibly different from the threshold for single terms) was applied to retain up to the threshold number of phrases with a weight of at least 2.

Concerning representation of patents for *k*-nearest-neighbor classification, we have settled for the present on a very small portion of each patent document. Our research has shown the best performance using a vector made up of the most frequent terms from the title, the abstract, the first twenty lines of the background summary, and the exemplary claim(s), with the title receiving three times as much weight as the rest of the text. We are including only single terms because we have not found that the addition of phrases is better than using just single terms. This somewhat surprising result is in contrast with what we have found for searching, where phrases do improve performance, at least on very short queries.

7. SYSTEM PERFORMANCE

7.1 Speed

Retrieval time depends upon many factors, including the load on the server and the speed of the network, and the size of the query. In our environment, it usually takes around 2 or 3 seconds to retrieve a list of documents from the single database in response to a normal, short query. It takes around 5-10 seconds to retrieve a list of documents from the multidatabase. The first query after a long hiatus can take much longer – up to 30 seconds - when some large data structures are not cached. This retrieval time includes retrieving and ranking the documents, and for the top-ranked documents, getting class and subclass information from an auxiliary database, getting all the text of the documents, and formatting the text for display, for example, highlighting query terms. If a user then clicks on a link to see the text of a particular patent, the response is well under a second.

Classification time is comparable to retrieval time because it is dominated by the retrieval of *k* documents required for *k*-nearest-neighbor classification. We currently use the single 1995-1996 collection for classification, so classification of a short query takes 2 or 3 seconds. Classification of a patent-length document takes around 45-60 seconds.

7.2 Accuracy

It is difficult to measure how well the retrieval side of the system performs. There is no corpus of queries with manual relevance judgments to allow the measurement of recall and precision. We are presently developing such a set of queries and judgments for our work on collection selection.

We have not yet measured this system's accuracy in classifying patents into the complete set of 135,000 subclasses. We have done a large amount of research measuring the accuracy of *k*-nearest-neighbor classification on smaller sets of patent subclasses, and in comparing *k*-nearest-neighbor accuracy with that of Bayesian independence classifiers, to be reported elsewhere. On small sets (4 to 6 subclasses) we get performance on the order of 80-100%. Usually the Bayesian classifiers perform a little better than the *k*-nearest-neighbor on these sets.

The largest set we have tested so far is the entire set of 76 speech signal processing subclasses under subclass 2.09 of class 395. This is a very difficult set in the same way that the complete set is difficult. Many of the subclasses are extremely similar to each other, and many of the subclasses have very little training data. On this set, we get classification accuracy ranging from 25% to 32%, depending on many different factors in how the classifiers are built. On the *k*-nearest-neighbor classifiers comparable to those used in the present system, accuracy is 30.9%. It is difficult to interpret these numbers, however, because they take no account of near misses. A misclassification accrues the same penalty regardless of how close the automatically-assigned subclass is to the correct subclass.

We often find that several subclasses closely related to the correct assignment appear on the ranked list of subclasses presented to the user. Such a list may be still be a significant aid to a classifier even when the correct subclass is not ranked first. For this reason, accuracy is not the best evaluation metric.

8. FUTURE WORK

We are currently engaged in evaluating the accuracy of this system, as mentioned in the previous section.

In addition, there are several parts of this system which we believe could be improved, involving the phrase help subsystem and the text classification subsystem.

We believe that building the phrase cooccurrence databases class by class was too fine-grained. Some of the classes are very closely related, and hence cover many of the same concepts. If we were to combine some of them we would have a more stable and reliable set of cooccurrence statistics. This combination could be based either upon clustering, or could be done manually with the PTO's guidance about what classes can go together.

We are currently experimenting with Bayesian classifiers, and intend eventually to combine these with the *k*-nearest-neighbor classifiers, as in Larkey and Croft [9]. The Bayesian classifiers should be able to distinguish closely related subclasses, due to the selection of negative training examples from closely related subclasses. They can refine the selection made by the *k*-nearest-neighbor classifier, which tries to distinguish each subclass from all the other subclasses at once.

There are additional available sources of information that could be used to aid text classification. In particular, each classification schedule includes a classification index, which is a list of phrases and a pointer to an appropriate subclass. The list is far from exhaustive, but we believe that in cases of a short query which happens to match one of the index items, this might be a more accurate means of finding the subclass.

9. Acknowledgments

Many people contributed to this system. Donald Byrd helped in the design of the user interface, Kamal Souccar and Michael Phillips contributed to the design and implementation of the user interface. Margie Connell incorporated collection selection into the system. Jinxi Xu and Fang-fang Feng contributed to the query processing and phrase help facilities.

This material is based on work supported in part by the National Science Foundation, Library of Congress and Department of Commerce under cooperative agreement number EEC-9209623, and also supported in part by United States Patent and Trademark Office and Defense Advanced Research Projects Agency/ITO under ARPA order number D468, issued by ESC/AXS contract number F19628-95-C-0235.

Any opinions, findings and conclusions or recommendations expressed in this material are the author's and do not necessarily reflect those of the sponsors.

10. References

[1] Allan, J., Callan, J., Croft, W. B., Ballesteros, L., Broglio, J., Xu, J., and Shu, H. 1997. INQUERY at TREC-5. In *Proceedings of The Fifth Text REtrieval Conference (TREC-5),* 119-132. Gaithersburg, MD: NIST special publication 500-238.

[2] Callan, J., Croft, W. B., and Broglio, J. 1994. TREC and TIPSTER Experiments with INQUERY. *Information Processing and Management,* 31(3):327-343.

[3] Callan, J., Lu, Z., and Croft, W.B. 1995. Searching distributed collections with inference networks. In *Proceedings of the 18th Annual ACM-SIGIR International Conference on Research and Development in Information Retrieval*, 21-28.

[4] Duda, R.O, and Hart, P.E. 1973. *Pattern Classification and Scene Analysis.* New York: John Wiley & Sons.

[5] Fellbaum, Christiane. 1998. *WordNet: An Electronic Lexical Database*, Cambridge: MIT Press.

[6] Krovetz, R. 1993. Viewing Morphology as an Inference Process. In *Proceedings of the 16th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, 191-203. Pittsburgh: ACM Press.

[7] Larkey, L. S. 1998. Some Issues in the Automatic Classification of U.S. Patents. In Learning for Text Categorization. Papers from the 1998 Workshop. AAAI Press, Technical Report WS-98-05, pp. 87-90.

[8] Larkey, L. S. 1998. Automated Essay Grading using Text Categorization Techniques. In *Proceedings of the 21st Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, 90-95.

[9] Larkey, L. S., and Croft, W.B. 1996. Combining Classifiers in Text Categorization. In *Proceedings of the 19th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, 289-297.

[10] Mitra, M., Singhal, A., and Buckley, C. Improving Automatic Query Expansion. In *Proceedings of the 21st Annual*

International ACM-SIGIR Conference on Research and Development in Information Retrieval, 206-214.

[11] U.S. Patent and Trademark Office. 1985. Patent Full-Text APS File. Technical Report, Office of Information Products, Administrator for Dissemination.

[12] Xu, J. and Callan, J. 1998. Effective Retrieval with distributed collections. In *Proceedings of the 21st Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, 112-120.

[13] Yang, Y. 1997. An Evaluation of Statistical Approaches to Text Categorization. Technical Report, CMU-CS-97-127, Computer Science Department, Carnegie Mellon University.

[14] Xu, J. and Croft, W. B. 1994. The Design and Implementation of a Part of Speech Tagger for English. CIIR Technical Report, IR-52, Dept. of Computer Science, Univ. of Massachusetts.

[15] Xu, J. and Croft, W.B. 1996. Query Expansion using Local and Global Document Analysis. In *Proceedings of the 19th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, 4-11.