Practical Evaluation of IR within Automated Classification Systems

R. Dolin* J. Pierre M. Butler R. Avedon
Datafusion, Inc.
139 Townsend Street, Suite 100
San Francisco, CA 94107

Ron.Dolin@kp.org, {jpierre,mbutler,ravedon}@datafusion.net

Abstract

This paper describes some of the work we have done to evaluate and compare the use of three IR systems (Verity, LSI, and SMART) as black boxes within an automated classification environment. We use automated classification to make a quantitative comparison of the effectiveness of the systems within this context. In so doing, we also develop criteria for the construction of a useful training set. These results lead to metrics useful in the integration of IR systems into larger applications. We conclude with an initial API for an IR component within an automated classification architecture.

KEYWORDS: IR evaluation, automated classification, training sets.

1 Introduction

A library environment includes such tasks as metadata generation, finding relationships between various thesauri and/or classification schemes, and document classification. There are many commonly used controlled vocabularies and classification schemes in industry, such as ICD9[1] for medicine, SIC[2] for market analysis, and the West Key System[3] for law. Identifying documents, companies, and people related to categories in these classification schemes requires processing many data elements. In a large-scale commercial setting, this must be done in as close to an automated manner as possible.

*Currently with Kaiser Foundation Health Plan, Inc.

[1]ICD9, the International Classification of Diseases (Ninth Revision), is used, for example, for public health statistics at the U.S. Centers for Disease Control.

[2]The Standard Industrial Classification Code, 1987.

[3]The West Key System is a legal classification generated by West Publishing to classify elements of case law.

In a digital environment, information retrieval (IR) is typically used and designed for search and retrieval. However, IR systems are also used as classification tools. In general, automated classification necessitates the integration of a general IR system into a comprehensive production environment. In such a setting, the requirements on an IR system are potentially very different from the usual evaluation criteria used in a more academic environment. For example, issues related to cost, functionality, platform, robustness, maintainability, and ease of use (e.g. documentation, API, etc.) are paramount. In terms of evaluation, the question is not simply how these systems might compare in TREC [12], but how these systems are likely to perform when used on arbitrary data. Furthermore, while TREC shows results generated by each of the IR systems' builders and/or expert users, it does not necessarily indicate how these systems might compare when deployed by independent developers.

As Harter and Hert state, "In addition to users, a number of other players (e.g., managers, system designers, vendors, content providers) have a stake in the success (variously defined) of an IR system. In general, evaluation from their perspectives has not been addressed much in the IR literature" [6]. In this paper, we focus on the development of a methodology for practical evaluation of IR systems, mainly from the perspective of system designers, within the context of an automated classification environment. We attempt to view such systems as black boxes, and set out to determine their effectiveness as a single component of a large system.

We seek to identify effective evaluation metrics that do not rely on relevance (such as precision and recall do). As Harman states in the TREC-3 Overview, "The relevance judgements are of critical importance to a test collection" [5]. There are several problems with such an evaluation methodology. First, it is not feasible to have such relevance judgements for arbitrary new collections, applications, etc. Second, relevance is not definable without knowledge of the application, let alone the user's intentions [10]. Finally, tuning parameters for a given collection are not necessarily applicable to new collections.

We simultaneously attempt to answer several related questions. First, how can we compare different IR systems used for automated classification, working on loosely controlled, semi- or un-structured data? We would like to derive a straightforward evaluation methodology. Second, in working with automated classification, how can we determine if our training set data is sufficient, or at least reasonably self-consistent? Third, how might we determinê appropriate parameters for the IR systems, as well as thresholds for result weights, to be able to apply evaluation results to other data sets?

In the next section, we present an overview of the IR systems used in this study. The full evaluation methodology is described in Section 3, followed by an analysis of the experimental results in Section 4. Finally, we conclude in Section 5 with a discussion of the results, a preliminary API for the IR component of an automated classification application, and future directions.

2 Overview of the IR Systems

Table 1 shows various characteristics of IR systems, focusing on functionality.[4,5] The first column lists the name of the system, followed by several possible functional features. The appearance of a 'y' denotes that the feature is available (possibly optionally), while an 'n' denotes that the system does not use that feature at all. Similarly, 'many' means that several methods are available. A '?' by itself indicates our uncertainty as to the capability of the system regarding the particular feature, while a '?' next to another mark indicates our untested belief about the availability of the feature.

The 'idf' column denotes the use of some form of inverse document frequency applied to term weights; 'norm' denotes the use of document length normalization. The 'phrase' column represents the ability to index and query with multi-word phrases taken as a single entity. 'fields' denotes the ability to query within particular fields, as well as the ability to combine the results of multiple fields. 'POS' describes a system's use of part-of-speech tagging: for example, the ability to extract noun phrases and proper names. 'stem' implies that the system performs some type of word stemming; however, not all stemming is equal. The use of a lexicon generally yields substantially better results than the application of simplistic language rules. The 'trm-co' column denotes the use of term co-occurrences.

[4]Space does not permit us to discuss issues of cost, ease of use, etc.

[5]It is important to note that the information in this table has been derived from our using the systems and/or trying to decipher the documentation. In that sense, it reflects the degree to which these features are clearly presented.

'wts' denotes the availability of result weights or scores. As will be discussed later, this is very important for the purposes of automated classification. Finally, 'platform' shows the platforms on which the system is able to run (we were only interested in Unix and Windows environments in this study).

Verity

We used version 2.3 of the Search'97 Developer's Kit. Training set documents were indexed using their default stopword list. We attempted to submit query documents using query-by-example (i.e. the "<LIKE>" operator) but found the results to be unacceptable due the fact that only 15 terms were extracted for each query. After much effort and experimentation we were able to obtain acceptable results by first preprocessing the query into a weighted term vector (ignoring stopwords) and then submitting the top 50 terms and their weights using the "<ACCRUE>" operator.

LSI

Latent Semantic Indexing (LSI) [4] employs the vector model of IR in coordination with singular value decomposition (SVD) for dimensionality reduction. Each dimension in the reduced space tends to represent those terms that are more likely to co-occur in the collection. As a result, LSI does not by default use word stemming to identify related terms.[6] We used "log entropy" weighting with cosine similarity measures. Results improved dramatically after increasing the number of factors from the default of 100 to 150 (with no change in going to 200).

SMART

SMART [2] is an implementation of the vector space model of information retrieval. In this study we used version 11 with stemming turned on, the default stopword list, and the "atc" weighting for both documents and queries.[7] Training set documents and query documents were indexed and treated as an experimental collection. Queries were submitted in batch against the training set, and a full result list of matching documents and scores was output from SMART.

Others

We had originally hoped to include both Cheshire [9] and InQuery [1] in this study. Unfortunately, we were not able to do so within the time-frame of the submission. Cheshire, for example, was not originally designed to handle large queries - some of our query documents

[6]For example, non-lexicon algorithmic stemming might view doctor, doctors, and doctoral as variants of "doctor". LSI, however, might find that doctor and doctors tend to occur in the same documents, while doctoral does not. As a result, LSI tends not to benefit from the use of stemming.

[7]This basically corresponds to the usual tf/idf with cosine normalization.

Tool	idf	norm	phrase	fields	POS	stem	trm-co	wts	platform
Verity	n	n	y	y	n	y	n	y	Unix,NT
LSI	many	cos	n	n	n	n	y	y	Unix,NT
InQuery	many	y?	y	y?	y	y	y?	n	Unix,Win
Cheshire	y	y?	?	y	n	y	?	n	Unix
SMART	many	many	?	y	n	y	?	y	Unix,?

Table 1: Various Functionality of IR Systems

were of the order of a megabyte. InQuery, as distributed by Sovereign-Hill (now Dataware), was designed around a web search-engine. As a result, it was difficult to package up batch-mode queries of the style required by these experiments. Neither Cheshire nor InQuery provided document weights in the query result set. We were able to work with the authors of Cheshire to change this, but were not able to completely debug it in time. We are working to include these systems in future studies. We will discuss in the conclusion some of the usability issues related to all five of the systems we have considered so far.

3 Methodology

The approach developed below relies on the creation of a 'training set' of documents that represent each classification category. The document to be classified is submitted as a query to the system. The result of the query is a ranked list of categories. A full description of our methodology follows.

3.1 Selection of Test Data

We first pulled over the SEC 10K filings for 1997 from the SEC's Web site[8]. These documents are similar in nature to annual reports. Within each document, "Item 1" includes a description of the type of business performed by the company. Furthermore, each document is also associated with a four digit SIC category (we denote this as the "assigned category", or AC, of the SEC document). After bringing over the documents, we extracted the "Item 1" sections and some header information (e.g. the company name, ID number, SIC category). Each filtered document was then grouped according to SIC category. As a result, each category is represented by a collection of SEC documents, thereby forming a training set (TS).

Our project focuses on developing practical solutions to automating the process of classifying. In order to do so, we must understand the ways in which classifications are actually used in the world.

We do not assume that a classification provides comprehensive coverage of its domain. Nor do we assume that there is only one correct category assignable to each item. Instead, we assume the meaning of a class is defined by its usage, not by the definition provided by the authors of the classification. In this case, the usage is in the SEC's classification of companies by SIC code. Each company is assigned exactly one four digit SIC number. The number is assigned by the SEC based on the SIC assignment in Standard and Poor's Compustat database. The SEC will revise an SIC assignment to a company, if the company asks for a review.

The documents used in the training set are submitted by a company to the SEC. No validity checking is performed by the SEC on the SIC code provided in the documents. Thus we are assuming that the SIC code included in a document is the assigned SIC code for a company.

Furthermore, the assignment of a single four digit SIC code to a large company with diverse business interests assumes the company can be represented by a single SIC code. The level of granularity represented by a four digit code may not allow for an accurate categorization of the company. Results might differ if we were attempting to categorize at the three digit level using our training set based on four digit categories.

In performing automated classification, we are not concerned with assigning the universally correct or best SIC code to a company. A correct assignment in this case is relative to the language used in the documents in the training set. The language in the document is merely representative of the 'correct' definition of the category. Yet, operationally, a correct assignment is one that reads an SEC document describing a company and assigns the SIC code given to that company. This circular definition raises interesting questions about defining a category on the basis of a particular training set, and then using that definition to classify documents for some possibly unrelated purpose.

3.2 Training Set Construction

The methodology involves classifying a query by using the various IR systems to perform similarity matches between the query and the category representations. This work is similar to that in [8]. This is in contrast to, for example, the method of [7], which matches a query directly to the individual documents, and then derives a ranked list of categories by aggregating the results of the top ranked documents.

[8]http://www.sec.gov/

We next sought to evaluate how many documents needed to be associated with a category in order for the results to be sound.[9] We thus performed experiments varying the number of documents in each category, and measuring the resulting change in the evaluation metrics. If the TS was completely empty or composed of completely random text, then one would expect that the assignment of queries to categories would be random. As a result, both the mean and median of the ranks of the AC's in the query result sets would be approximately one half of the total number of categories. On the other extreme, if the TS was composed of 'perfect' mutually exclusive discriminatory text, and each query fit one and only one category, then both the mean and median of the AC ranks would be exactly 1. As we go from an empty training set to a better one, the AC median rank decreases from half of the total number of categories to something approaching 1.[10] Thus we can see if our TS is improving by observing a decrease in the median AC rank.

One way to make such a test is to start with only the category description as the TS and to measure the median AC rank. We then add a single document (keeping the description) to each category and measure the median AC rank again. We continue doing this until adding more documents has only a negligible effect on the median AC rank. Suppose that this happens within N documents. This requires that we use only those categories with at least N documents in the TS[11]. Here we are faced with the problem of maintaining enough categories to allow for a meaningful statistical analysis. We started out with 411 categories. However, many of them had 6 or fewer documents per category. If we ruled those out, we would be left with only 206 categories, which begins to reduce the confidence level of our measurements. As a result, we simultaneously wanted to decrease the maximum number of TS documents per category (enough to keep the number of categories as high as possible), and increase it (enough to know that adding more documents per category made little difference in the median AC rank).

Previous work using LSI with LCC and MARC records [3] indicated that as few as four documents per category might be sufficient for accurate classification. Taking this as an estimate, we decided to use those 206 SIC categories that had at least 7 documents – one for the query and 6 for the training set.

[9]It is perhaps worth pointing out that the size of the included "Item 1" section of the TS documents varied widely, from a single paragraph to many tens of pages.

[10]Assuming, of course, that the categories are correctly assigned to the test queries.

[11]Actually, $N+1$, if we take into account the randomly selected query document

3.3 Query Generation

We sought to get a good representative collection of queries. Given that we had no *a priori* reason to favor one category over another, we decided to use a stratified random query selection mechanism. We randomly selected one document from each category as a query, giving us 206 queries in all. We first used a training set consisting solely of the SIC category descriptions. These were indexed within each IR system. We then took a query document and used the IR system to provide a similarity match against each category. The result of this matching was a list of categories ranked by their calculated similarity to the query. Somewhere in this ranked list was the category actually assigned to the original SEC document (the AC). The position of the AC in the ranked list is called the AC rank, and its score is called the AC score. Similarly, the score at the top of the list (i.e. the top ranked category) is called the Top score. We used all 206 queries from our stratified random sample and combined the results within our evaluation metrics.

We then re-did the entire procedure with a new training set: one that included, in addition to the category descriptions, a single randomly selected document from each category. These documents were different from the documents in the query set. We continued to perform the experiment with a larger and larger training set until the last run, in which each category's training set was composed of the category's description and six randomly selected documents within the category, but all different from the query document used for that category.

Thus for a single set of 206 query documents taken from the 206 allowable categories, we ran seven sets of experiments and calculated the various evaluation metrics as a function of the training set size – the number of documents in each category. In order to allow for possible peculiarities in the query documents and/or the selected training set documents, we re-did the seven experiments three more times, with three different randomly selected query documents and training set documents.

4 Analysis

As an example of a successful run, Figure 1 shows the results of using LSI with six documents per category. It is clear from this figure, with 45% of the assigned categories automatically chosen as the top ranked category out of 206, that there is an overall consistency and discriminatory vocabulary within the training set data.

We present a summary of the results in Figures 2 through 4. The results are the averages and 90% confidence intervals of four different trials of 206 queries

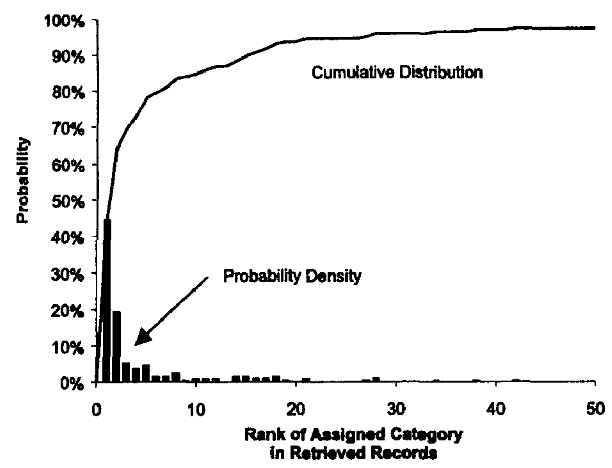

Figure 1: Distribution of Ranks for LSI, Six Documents per Category

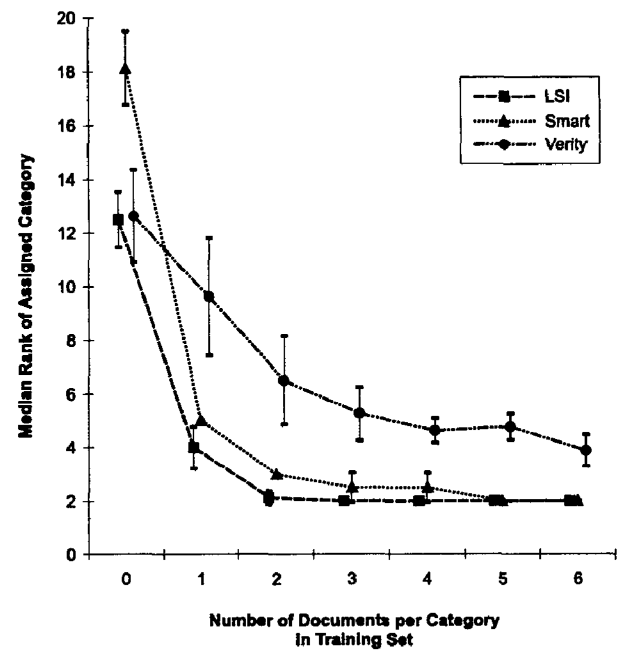

Figure 2: Summary of Results – Median AC Ranks

each, as described in the previous section.[12] The rank information allows us to compare the different IR systems, while the score information is more useful for integrating the systems within various applications.

4.1 IR System Comparison

The trend in the data indicates increased classification accuracy as the number of training set documents per category are increased. However, depending on the particular IR system, the results do not change appreciably after three or four documents per category. These graphs imply several results.

First, previous to these experiments, we were not sure whether the data were classifiable. That is, we had no way of knowing *a priori* if the documents and classification scheme would yield an effective system for classifying anything, even the documents within the training set. The results indicate that there is enough consistency in the documents' assignment to their respective categories that the query documents can, for the most part, be associated with their correct categories.

The second result is that, as evidenced by both the median and mean AC rank values, we can compare accuracy between the various IR systems within this context. In this case, LSI does better than SMART, which does substantially better than Verity. Without loss of generality, we can take as our baseline results whichever IR system gives us the best values. That is, we can assume in this case, given the data, that the best possible automated classification results are those of LSI. Any system's performance can then be compared with that. The difference between an AC median rank of 2 for

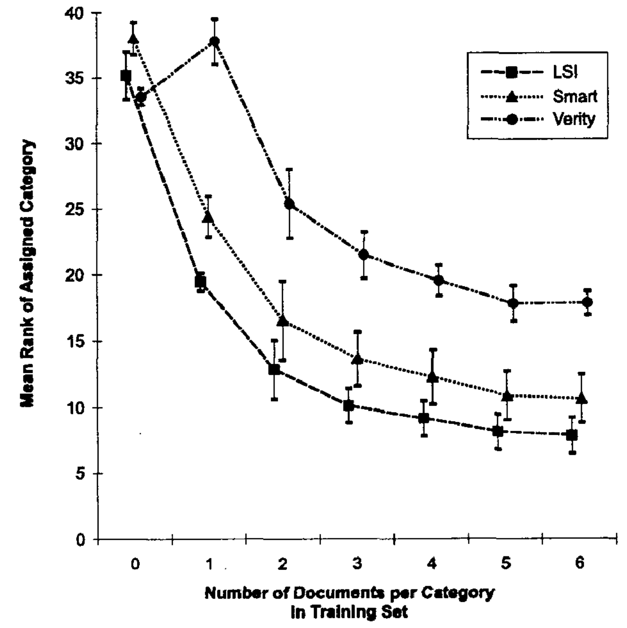

Figure 3: Summary of Results – Mean AC Ranks

[12]Each trial consisted of randomly generated queries and training set documents. Although there was a small degree of overlap between the different trials, the largest percentage of overlap in either queries or training set documents between any two trials was 5%, and only 1% among the intersection of all four trials.

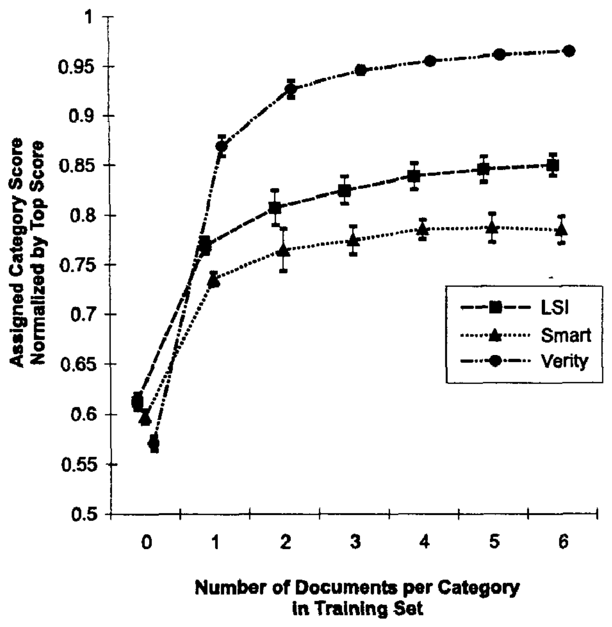

Figure 4: Summary of Results – Normalized AC Scores

LSI and SMART and that of 4 for Verity is not trivial, depending on the applications. For example, consider using the IR systems for semi-automated classification, in which the top ranked categories are presented to a human to select the most appropriate one(s). In determining how many categories should be examined by hand, the data indicates that, using LSI for this data, there is roughly a 75% probability of finding the best categories within the first four listed. The same probability requires looking over approximately three times as many categories using Verity.

The third result is that we can estimate the minimum size of the training set required to achieve accurate classification. We can thus, for example, exclude categories with an insufficient number of training set documents. This is very helpful in determining which sets of classified documents might be useful in an automated classification environment.

It is important to note the relationship between these results and the results from TREC-3 [5], the last TREC in which these IR systems were simultaneously included. In TREC-3, their relative effectiveness, as applied to the routing problem,[13] was similar to their relative effectiveness reported here. In the TREC case, LSI and SMART were among the best systems in routing capabilities, with LSI doing better at ranking the list of documents it had determined were relevant. Verity was not among the best performers, even with hand-crafted "TOPICS" to aid in query expansion. We were able to arrive at results consistent with TREC-3, albeit less extensive, but without necessitating relevancy judgements of our test collection beyond the originally assigned categories.[14]

4.2 Implications to Applications

Figure 4 shows the average ratio of AC scores to Top scores. These ratios remain fairly consistent, or, within their estimated errors, monotonically increase. They show that the spread of scores between the Top scores and the AC scores are fairly narrow. Thus within each IR system, there are potential score thresholds that can be used to determine acceptable similarity measures.

These results are useful in determining how to integrate individual IR systems into various applications. Consider the example of trying to automatically associate relevant categories to a set of documents.[15] We have no way of knowing which, if any, categories might be relevant to a given document, let alone how many might be appropriate. As a first approximation, we can use the data in Figure 4 as an indication of the score threshold to use in determining "relevance" (i.e. sufficient similarity) for a particular category. Given that the average ratio of AC score to Top score is, say, 0.85, we might want to associate a category to a document if the similarity measure is at least 0.85 of that category's Top score. We might also want to exclude categories that receive particularly low scores in the training set experiments; the implication is that the training set data of these categories is not sufficiently discriminatory to be helpful for automated classification.

5 Conclusion

This paper described the use of automated classification results as an evaluation tool for comparing IR systems as a component within, for example, a knowledge management environment such as a digital library. As complete systems for automatic classification, we found all of the IR systems that we worked with to be lacking in some way or another. This was mostly due to assumptions made in their system design relating to their intended use as tools for search and retrieval based on interactive human queries. A basic framework for a generic automatic classification system can be described

[13]The routing problem is very similar to an automated classification problem. In the former, the categories are defined by the pre-specified queries, and the training set is composed of the set of documents assigned 'relevant' to each query.

[14]Determining the relevancy of documents to queries is similar to assigning one or more existing categories to a document. However one major difference, in terms of search behavior, is that of off-line versus on-line analysis. Thus, while it is obviously impossible to pre-determine how to relate documents to a query until the query is known, it is normal to classify documents in advance. Part of the query process then becomes selecting relevant categories, where each category has already been associated with appropriate documents.

[15]Taking into account the possible variability discussed in Section 3, such as the genre of these documents and the differences between them and the training sets.

by a set of modules:[16]

1. Assemble training set for categories:

   A set of representative training set documents must be selected and collected for each category from an existing document corpus.

2. Pre-process the training set:

   This module is only necessary if the original form of the training set documents is not appropriate for indexing. Pre-processing may include structural changes such as the extraction of sections of text, and removal or addition of fields or markup tags. It may also involve lexical processing such as stemming, part-of-speech tagging, noun phrase extraction, or the removal of punctuation. The amount of pre-processing required will vary depending on the capabilities of the IR search engine used.

3. Index the training set:

   This module essentially builds an index of concepts with a list of associated documents, but the details and algorithms used depend on the IR search engine and its underlying model for information retrieval.

4. Stream of query documents to be classified:

   This module accepts an incoming collection of query documents, from some external source (file system, e-mail server, etc.), and submits them to the queue for automatic classification.

5. Pre-process query documents:

   The module is similar to the pre-processing of the training set described above. An important requirement is that the size of the query documents be scalable. Several of the query documents in our experiment were on the order of 1 MB. We had great difficulty using IR systems such as Verity and Cheshire, due to inherent limits on query size in these systems.

6. Compare queries to training set:

   In order to automatically classify a large number of incoming documents, it should be possible to submit queries to the IR system through some kind of batch mode, without the need for human interaction.

7. Ranked list of results including scores:

   The output from the IR system should include the relevance scores as well as the rank and ID for every possible matching category. Though the absolute value of the scores may be unimportant, we feel that an analysis of the relative scores in a results list is essential for setting thresholds for category assignment as well as assessing the overall quality of the classification.

8. Assign categories based on thresholds:

   An algorithm for assigning categories is highly dependent on subjective requirements such as the desired degree of precision and recall, "fuzziness" of categories, and the maximum number of acceptable assigned categories. In our work we have not yet explored the use of such an algorithm, but this is an interesting area for further research.

[16]This is an adapted version of the TIPSTER "Generic Information Retrieval System" [11] for document routing.

This framework for generic automatic classification allows us to easily integrate an IR system as a black box into a knowledge management environment. Existing IR systems need to be designed to support this functionality, in addition to their standard retrieval roles. The methodology of developing a training set and a query set allows us to compare IR systems, test for self-consistency within the training set, and begin to develop parameters for automated classification thresholds.

In this paper, we looked at using a training set of one type of document to predict the classification of other instances of that document type. Since our definition of a category is based on usage, we are curious to see if a training set built with one type of document can classify a different type of document in the same domain. Can we build a training set of SEC documents and use it to predict the assignment of SIC codes in non-SEC documents? Similarly, can a training set composed of a collection of different document types with the same classification be used to predict the classification of other document types. Is there a point at which the usages average out? Future research will attempt to answer these questions.

References

[1] J. Broglio, J. Callan, and W.B. Croft. INQUERY system overview. In *Proceedings of the TIPSTER Text Program (Phase I)*, pages 47–67, San Francisco, CA, 1994.

[2] C. Buckley. Implementation of the SMART information retrieval system. Technical Report 85-686, Computer Science Department, Cornell University, 1985.

[3] R. Dolin. *Pharos: A Scalable, Distributed Architecture for Locating Heterogeneous Information Sources.* PhD thesis, University of California, Santa Barbara, Santa Barbara, CA 93106, June 1998.

[4] S. Dumais et al. Using latent semantic analysis to improve access to textual information. In *Proceedings of the 1988 CHI Conference*, 1988.

[5] D. Harman. Overview of the Third Text REtrieval Conference (TREC-3). In *Third Text REtrieval Conference (TREC-3)*, Gaithersburg, MD, 1994. http://trec.nist.gov/pubs/trec3/overview.ps.

[6] S. Harter and C. Hert. Evaluation of information retrieval systems: Approaches, issues, and methods. In *Annual Review of Information Science and Technology*, volume 32, pages 3–94. Information Today, Inc. (on behalf of the American Society for Information Science), Medford, NJ, 1997.

[7] L.S. Larkey. Some issues in the automatic classification of U.S. patents. In *Proceedings of the AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[8] R. Larson. Experiments in automatic Library of Congress Classification. *JASIS*, 43(2):130–148, 1992.

[9] R.R. Larson, J. McDonough, P. O'Leary, L. Kuntz, and R. Moon. Cheshire II: Designing a next-generation online catalog. *JASIS*, 47(7):555–567, 1996.

[10] L. Schamber. Relevance and information behavior. In *Annual Review of Information Science and Technology*, volume 29, pages 3–48. Learned Information, Inc. (on behalf of the American Society for Information Science), Medford, NJ, 1994.

[11] TIPSTER Text Program. *Generic Information Retrieval System.* Washington, D.C., 1995. http://www-nlpir.nist.gov/related_projects-/tipster/gen_ir.htm.

[12] E. Voorhees and D. Harman. Overview of the Sixth Text REtrieval Conference (TREC-6). In *Sixth Text REtrieval Conference (TREC-6)*, Gaithersburg, MD, 1997. http://trec.nist.gov/pubs/trec6-/papers/overview.ps.