# Navigating large-scale semi-structured data in business portals

Mani Abrol Neil Latarche Uma Mahadevan Jianchang Mao Rajat Mukherjee
Prabhakar Raghavan Michel Tourn John Wang Grace Zhang

Verity, Inc. 892 Ross Drive, Sunnyvale, CA 94089
{nlatarch,jmao,rmukherj,pragh,jwang,gzhang} @verity.com

## Abstract

This paper presents several paradigms by which users of Verity business portals (from within as well as from outside an enterprise) discover and navigate relevant semi-structured data in corpora with millions of documents. These paradigms include (i) combining free-text and structured queries, (ii) automatic classification, and (iii) personalization.

## 1 Introduction

It is estimated that there are about 4 billion static web pages on the internet. The web is rapidly growing at a rate of about 7.3 million new pages per day [2]. In fact, the information in enterprises dwarfs the volume and growth on the web, as evident from trends in the growth of storage shipments. The enormous growth stems from documents on corporate file systems, application servers, and various data bases. In order to provide users unified access to this vast amount of information, business portals have become increasingly popular.

The following characteristics are typical of business portals:

1. The need to access information in diverse repositories including file systems, HTTP web servers, Lotus Notes, Microsoft Exchange, content management systems such as Documentum, as well as relational databases.
2. The need to respect fine-grained individual access control rights, typically at the document level; thus two users issuing the same search/navigation request may see differing sets of documents due to the differences in their privileges.
3. The need to index and search a large variety of document types (formats), such as PDF, Microsoft word and Powerpoint files, etc., and different languages (such as, English, European and Asian languages).
4. The need to seamlessly and scalably combine structured (e.g., relational) as well as unstructured information in a document for search, as well as for organizational purposes (clustering, classification, etc.) and for personalization.

We briefly review how these features are addressed in Verity's flagship K2 Enterprise product for business portals. The first two items above are handled through "gateway" interfaces for each repository, ingesting access control information as documents are spidered from the repository. Verity provides state-of-the-art spider and gateway technologies which allow users to gather and index documents from HTTP web servers, file systems, Lotus Notes, Microsoft Exchange, Documentum, and various ODBC databases. K2 Enterprise incorporates Verity's KeyView technology that can automatically detect and filter over 250 document formats. Comprehensive multiple-language capabilities allow users to search documents in 24 languages. This paper will not discuss these technologies. Our focus here will be on the fourth item: organizing, searching and navigating a corpus of millions of documents containing semi-structured information.

We discuss three principal methods for this purpose, drawn from Verity's K2 Enterprise portal infrastructure product.

1. Arbitrary combinations of text and data cube queries. Consider the query: *Give me documents that contain the term "LAN", are of type "product sheet", language "French", format "pdf", product family "routers", speed* $10Mbps$*, price* $<$ 1000. Clearly this query has a text/unstructured



**Proceedings of the 27th VLDB Conference, Roma, Italy, 2001**

component (to match the term "LAN") as well as a structured component (the restrictions on type, language, format, product family, speed and price). We discuss how Verity's Parametric Search Engine handles this type query in Section 2.

2. Automatically classifying documents by textual content as well as structure derived from meta-data. Verity's automatic classifier can classify documents based on any combination of human-maintained rules or rules "learned" from exemplary documents; depending on the domain and expertise of those maintaining the information in a business portal, one or the other method is emphasized. This is discussed in Section 3.
3. Delivering personalized views of information by re-ranking search results, recommending documents to a user, locating experts on the query subject, and discovering community. Both explicit profile information and implicit behavioral patterns are exploited in the personalization engine. This is discussed in Section 4.

## 2 Combined Text and Structured Queries

In product catalog databases, each document (corresponding to a product in the catalog) has some unstructured text as well as a number of structured attributes. Note that some of these attributes may assume numerical values; for instance, a description of an automobile might have the year in which the car is manufactured and its price. A typical query is a conjunction of an arbitrary text query (e.g., the query mentioned in the introduction). Whereas the text query is the purview of classical information retrieval systems, the parametric query is traditionally handled using relational database systems. In a number of emerging applications, it is desirable to retrieve and rank documents that simultaneously meet both the unstructured and the structured query components, without recourse to two retrieval systems. Such applications include electronic commerce and marketplaces where scalability and performance play critical roles. Using an RDBMS to solve the problem would result in query responses that would be unacceptably poor. In addition to retrieving documents that meet the text and parametric queries in a scalable fashion, it is important to be able to rank the results not only by text query scores, but also by sorting along field values (e.g., price). This allows for the efficient navigation of a results list, allowing the user to further refine (or relax) the query at hand.

Our solution consists of augmenting a full-text index with an auxiliary parametric index that allows for fast search, navigation and ranking of query results. We now describe the elements of this index, and how they are organized in order to support all of the operations described above.

Each field is represented as a set (known as a BucketSet) and each unique item in that field is represented as a member of that set along with position and frequency information (this datum is known as a Bucket). A set of BucketSet structures make up a completed structure known as the parametric index which maps directly to the corpus from which it was generated. Extra meta-data are stored along with the BucketSets to allow ranges (either text, numeric or date) to be extracted for speed, enhanced data retrieval and clarity.

It is the mechanism by which the BucketSets and Buckets interact (through their parametric representation) that allows complex set operations (based on nested intersects and unions) to be applied to them. This mechanism results in: (a) real time cardinality statistics for these operations, and (b) an iterative refinement of the operations applied to the database, which can result in the reduction of old Buckets and the inclusion of new Buckets (depending on the semantics applied). In the simple case where A and B are both Buckets of the same set, "$A \bigcap B$" can remove items, whilst "$A \bigcup B$" can add items.

We can also express the text search in the parametric domain (i.e., a text search maps to a BucketSet where each item (hit) is considered as a separate unique Bucket in that set with no context until applied to a parametric index). Both kinds of BucketSets can interact with the same set of operators. Extra speed is obtained by performing query optimizations in real time – for instance, by examining the relative sizes and complexities of the BucketSets when intersecting them and ensuring that the smallest or least complex (when taking into account the free text representation) controls the query optimization. The text search component is built on Verity's VDK kernel [5] with a K2 search engine on top [3].

The system has been tested on numerous examples with over a million documents with 6-10 fields of structured attributes. Interactive performance is feasible for several concurrent users, using a standard desktop PC as the server.

## 3 Automatic Classification

While searching provides an efficient way for users to find relevant information in business portals if they know what to search for, there is a different need for browsing and navigating information. Taxonomies are the most popular way for organizing documents into navigable structures. With a taxonomy, users can easily navigate / sift through the category hierarchy to find relevant information. Search within a category typically produces more relevant results than unscoped search.

There are two main phases in deploying taxonomies in a portal environment: taxonomy construction, and maintenance. The former defines the category hierarchy, and the latter populates documents into the tax-

onomy once it is built, and modify the taxonomy structure when needed. Verity refers to a populated taxonomy as a *knowledge tree* which consists of a taxonomy of browsable categories and databases that store the relationships of documents and categories.

Taxonomy construction and maintenance is a challenging problem that works best with sufficient domain knowledge. Fully automatic construction and maintenance often leads to unsatisfactory results. Consequently, most taxonomies are built and maintained manually by human experts. Well-known examples include the directory structures of Yahoo! and Open Directory Project. Manual classification produces higher accuracy than machine because library scientists understand the content, context and nuances of domain information. However, manual population of a large volume of documents is a prohibitive task, and is very costly.

Verity's Intelligent Classification technology combines human intellect and machine efficiency, thus strikeing a balance between manual construction and automatic generation. Depending on the domain and expertise of those maintaining the information in a business portal, one or the other method can be emphasized.

Verity Intelligent Classifier [6] provides several taxonomy construction methods, each of which can also be used to categorize documents into a taxonomy, i.e, to generate a knowledge tree:

1. Construct a taxonomy manually through Verity Intelligent Classifier's graphic user interface;
2. Extract a hierarchy of categories of a taxonomy from URL paths or file paths; This method enables administrators to mirror the web site or file system structure in the taxonomy.
3. Fetch categories from a collection field (e.g., meta data); This method can be used when the categories are explicitly listed in a field in the collection. This is the case, for example, when categories are specified in a META tag inside HTML documents that is indexed into a collection field.
4. Generate a taxonomy from document clusters produced by Verity's clustering algorithm. Each document in the collection (or in a selected subset) is represented by a feature vector. The clustering algorithm groups similar documents into clusters. A hierarchy of clusters are generated by recursively breaking large clusters into smaller clusters.

Verity Intelligent Classifier provides the following methods for defining classification rules for each category in the taxonomy.

1. Manually construct classification rules using Verity's powerful query language;
2. Learn classification rules for categories automatically from exemplary documents associated with these categories in the taxonomy;
3. Interactively refine classification rules by providing relevance feedback to the test results of previously built rules.

The core technology that supports these automatic features is Verity's regularized Logistic Regression Classifier (LRC). LRC automatically learns a classification rule from a set of documents that are labelled as relevant or irrelevant to a category. Let a document be represented by a feature vector $\mathbf{x} = [t_1, t_2, \cdots, t_d]^T$ , and $r$ be the relevancy measure of the document with respect to the topic, $0.0 \leq r \leq 1.0$. Given a set of relevant documents and a set of irrelevant documents for a category, the LRC learning algorithm learns a regression function

$$log(r/(1-r)) = w_1 t_1 + w_2 t_2 + \ + w_d t_d + b = f(\mathbf{w}, \mathbf{x})$$

such that the separation between these two sets of documents is maximized. This is done by introducing a regularized term $||w||^2$ in the cost function. Structure risk minimization theory [7] guarantees a minimized upper bound on the classification error on future documents.

Once the regression function is determined, a future document can be assigned to the category if the relevancy score $r$ for the document is greater than a pre-specified threshold (usually 0.5). The relevance score can be computed as follows.

$$r = 1/(1 + \exp\{-f(\mathbf{w}, \mathbf{x})\})$$

This classification rule can be conveniently represented by Verity's powerful query language. Our experiment shows that on Reuter's bench-marking data set, LRC achieves the state-of-the-art performance at 88% precision-recall break-even rate [1].

Once a knowledge tree is defined in Verity Intelligent Classifier, it can be exported to Verity Knowledge Organizer [4]. Administrators can configure how to display search results and the category structure to end users. End-users can browse through the documents in a collection by category and drill down to a subject of interest. They can also limit the scope of their searches only to the categories of interest.

## 4 Personalization Engine

The Verity Personalization Engine enhances the end-user's information discovery, or product discovery experience to the next level beyond search and taxonomies. It uses a tensor space model that represents different entities in the system, such as products, documents, users and queries as vectors in the vector space. These vectors adapt to latent patterns in user behavior in order to dynamically personalize the results of subsequent searches in different ways, as outlined below.

By tracking user choices, or transactions, in a configurable manner, the engine allows for refinement of

search, allowing usage feedback to overcome the vagaries intrinsic in data sets, and allowing search results to adapt over time to domain specific knowledge. Further, by tracking individual user behavior, the engine effectively constructs user profiles so that effective and representative recommendations can be served.

A Transaction comprises a set of vectors that are updated in the engine in such a way that reduces the distance between them. Some examples of transactions are:

User <**joe**> bought <**Toaster X**> upon query <**"toaster"**>

User <**john**> viewed <**Document Y**> upon query <**"personalization"**>

Figure 1 depicts the vector space, with multiple vector entities, and the effects on this vector space of transactions that explicitly feed back user choices to the personalization system.

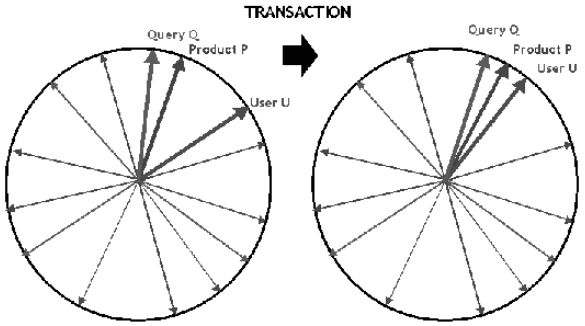


Figure 1: Effects of Transactions on Vector Space.

Figure 2 shows the relationships between the entities that the Personalization Engine currently supports. Other types of vector entities can be supported by this technology in the future, e.g., cluster/category vectors, etc. These relationships result in the following features that the personalization engine supports.

- Adaptive Ranking - multiple selections of an item for a given query causes the relevance of the item to be boosted for all users.
- Product/Document Recommendation - Products/Documents are recommended based on a combination of the current query and the user information, based on the user's past behavior.
- Document/Product Similarity - Documents or products that are similar to a selected item are recommended.
- Expert Location - Based on a document that is being viewed, or a query that has been issued, experts/users in the organization are recommended, based on the proximity to the subject area.
- Community - A dynamic user community can be presented, based on the current user's profile, and possibly, other information.

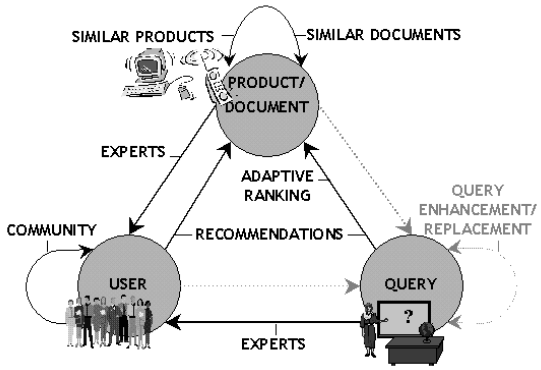


Figure 2: Functions Supported by the Personalization Engine.

## 5 Conclusions

We have presented three principal methods, drawn from Verity's K2 Enterprise family of portal infrastructure products, for searching, navigating, organizing, and personalizing information resided in a corpora of millions of semi-structured documents. Verity's Parametric Search Engine is capable of serving arbitrary combinations of text and data cube queries without hitting any back-end databases. Verity's Intelligent Classifier balances the best of human expertise and the state-of-the-art machine learning algorithm to efficiently and accurately classify information for easy navigation and scoped searches. Our Personalization Engine uses an innovative tensor space model to provide various personalized services, including adaptive re-ranking, recommendations, expert location, and community discovery.

## References

[1] Susan T. Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In Georges Gardarin, James C. French, Niki Pissinou, Kia Makki, and Luc Bouganim, editors, *Proceedings of CIKM-98, 7th ACM International Conference on Information and Knowledge Management*, pages 148–155, Bethesda, US, 1998. ACM Press, New York, US.

[2] Cyveillance Inc. Sizing the internet. July 2000.

[3] Verity Inc. *K2 Toolkit (K2): API Reference Guide. V2.0.* 1999.

[4] Verity Inc. *Knowledge Organizer, V2.0.* 2000.

[5] Verity Inc. *Verity Developer Kit (VDK): API Reference Guide. V3.1.* 2000.

[6] Verity Inc. *Intelligent Classifier, V2.6.* 2001.

[7] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, 1995.