Search Considered Integral

A combination of tagging, categorization, and navigation can help end-users leverage the power of enterprise search.

Most corporations must leverage their data for competitive advantage. The volume of data available to a knowledge worker has grown dramatically over the past few years, and, while a good amount lives in large databases, an important subset exists only as unstructured or semi-structured data. Without the right systems, this leads to a continuously deteriorating signal-to-noise ratio, creating an obstacle for busy users trying to locate information quickly. Three flavors of enterprise search solutions help improve knowledge discovery:

Raw engines. These are toolkits that developers can use to embed high-powered search into their applications. A popular upcoming implementation is Lucene, an open source Apache project available in Java, .NET, Python, C++, and others. You need to implement the crawling, parsing, and UI yourself; however, the engine handles Boolean logic, fuzzy queries, stemming, and hit highlighting. Commercial offerings, such as Verity (bought by Autonomy), Fast, or Coveo, provide the same core

features as Lucene, but may add an entity extractor, thesaurus, automated classification, and a number of other leading features.

Intranet appliances. These are low-customization boxes that you simply plug into your network, point to a data source, and let index. The UI, crawling, and indexing can be tweaked; however, features can rarely be added or changed. Depending on the ranking algorithms used by the vendor, there is huge variation in the relevancy of results provided by each implementation. Google and Thunderstone both offer enterprise search appliances that have received positive reviews.

Desktop search. A large number of personal search engines are available that index e-mails, local files, instant messaging history, Web history, contacts, and more. These engines are very popular for the home computer market. Enterprises, however, must address security and privacy concerns, while also dealing with the performance impact to desktops and mail servers. Most companies strive to keep data off local machines; however, the desktop looks to be the best place to aggregate personal

RYAN BARROWS and JIM TRAVERSO, MORGAN STANLEY

Search Considered Integral

data stores with larger, centralized repositories.

All three search solutions are likely to show up in an enterprise with massive information management challenges. At Morgan Stanley we have had a group working on intranet search and raw search engines for more than five years and have been experimenting with desktop search since 2004. A fourth piece of this puzzle has yet to be popularized: combining tagging, categorization, and navigation to improve the overall experience for the end user. This piece is needed, as machine-relevance algorithms alone are not good enough to produce high-quality intranet results. In this article we discuss what such a system looks like, with a particular emphasis on solving enterprise-scale problems.

EVOLVING SEARCH INTERFACES

Many groups within Morgan Stanley have built search into their enterprise applications. For some, a search UI is the key distribution mechanism, while for others search is just a small added feature. With literally thousands of different applications, it is hard to generalize how we use search interfaces, but two applications illustrate how search has evolved.

The screen shown in figure 1 was from one of our most popular client-facing applications in 2000. It allowed clients to get detailed research reports about hundreds of public companies using an advanced search interface. Each report was tagged with enough metadata to allow extremely focused searches.

In 2004, one of our groups built a Web application to allow multifaceted browsing of training materials, shown in figure 2. Feedback from users was highly positive; their reactions showed they quickly found the items they were seeking; discovered unsought items of interest; successfully searched for items even when it was difficult to articulate what was sought; never hit dead ends with no idea of how to proceed; and were satisfied when their search was finished that they had found all the objects that matched the characteristics in which they were interested.

This is very much in line with the results of other

research projects, such as a 2002 survey in *Communications of the ACM* (<http://www.sims.berkeley.edu/~hears/papers/cacm02.pdf>).

A feature that was notably lacking in this application, however, was full text search. Users wanted to run a full text search and then browse through the results by facet, or browse to a set of documents and then run a full text search from there.

Although an advanced search interface provides deeper functionality than faceted browsing, users find it difficult to use. If they are not able to choose the right set of filters on the first shot, many are unwilling to keep trying with different parameters. To end users, the value of the tool drops with each click that does not help them accomplish their goals. For many use cases, faceted navigation became a welcome improvement to usability. In fact, many then wanted to apply this system to other applications within the company.

BUILDING AN ENTERPRISE METADATA CATALOG

After helping many groups follow the same steps (in different data domains), we realized that there was a significant opportunity for optimization in front of us. We looked at all these projects and extracted the common work to the following steps:

1. **Define a metadata schema.** My application may care about the date, author, subject, and keywords of documents. If these are research reports about public companies, we may also have ticker and industry.
2. **Index a set of documents.** This includes not only

2000 – Advanced Search

FIG 1

extracting all text for regular search, but also extracting the metadata into custom fields.

3. Write a UI for querying and displaying results. This usually involves hard-coding input fields to allow sorting and filtering by the relevant metadata for this domain.

As an infrastructure team, we are trying to create a low-cost platform that can allow business users to solve what used to be an “IT-required” problem. After evaluating the relevant products on the market, we decided to assemble the overall solution, leveraging a number of third-party components.

ONTOLOGY MANAGEMENT

To support the hundreds of different business applications that could benefit from a metadata catalog, each group must be able to create and maintain its own *ontology*—a dictionary representing the metadata for a particular domain. A particular category or field in an ontology is a *facet*. It may have one or more values and be strongly typed, such as a date, number, or Boolean.

Although programmers traditionally do this work, there is substantial value in allowing business users to do it themselves. Not only will this decrease the cost of each project, but it will also speed delivery and lower the barrier to entry for launching new initiatives. Building this ontology manager right means handling a number of tricky situations:

- There are different types of metadata (date, string, number, etc.). This affects UI, as well as comparability for search.

- Different users may want to assign different values to the same metadata category (e.g., subject).
- Some metadata items need to allow for multiple values (e.g., keywords).
- Groups may want to own some metadata, requiring an approval process for any changes.
- Since each group wants to see a different set of facets, your metadata UI has to be metadata-driven itself.
- A category of metadata may need to be limited to the choices in a tree; this is known as a taxonomy. An example of a taxonomy value is Departments → Technology → Enterprise Infrastructure → Search.
- Different validations are required for different metadata (e.g., six-character alphanumeric or any valid stock ticker).

EXTRACTING METADATA

In addition to information about an author, we can get a lot of other metadata from documents:

- Entity extraction allows us to pull out metadata from well-defined domains. For example, we can easily pull a list of company names and stock tickers from documents.
- An AD or LDAP directory can tell us who your manager is, what mail groups you are in, what region you are in, etc. All of this is potentially useful when we later want to match people with similar interests.
- Custom database information can be repurposed. For example, we have a database that holds metadata about thousands of multimedia files. Each presentation is tagged by language, series, business unit, date, speakers, etc. An XML file configures our database crawler to map columns in a table to values in the ontology.
- When all else fails, there is always the possibility of screen scraping. While a major ISV cannot build these scripts for every custom application a customer may use, we can. Further, we can prioritize the work by the added value of having the metadata extracted for any particular system.
- Many vendor applications, such as Microsoft’s SharePoint, already capture a significant amount of metadata from users. We can go one step further than the crawler integration by sticking ourselves into the “update pipeline” so that we can send the changes to our metadata catalog in realtime.

2004 – Faceted Navigation

FIG 2

USER INTERFACE

Metadata extraction is never perfect, and even if it were, users would still want to modify the metadata that was found. For a system to work in the real world, it needs to

Search Considered Integral

allow end users to add their own metadata, a process that gets complicated quickly. Usability is also very important for browsing results. The challenges include:

- Full text search must be able to be combined with faceted browsing, in relevant order.
- Distinct URLs must be available for every results page to enable sharing via e-mail, IM, or posting on a blog.
- Metadata updates must be made as easy as possible—AJAX ratings à la Netflix, at a minimum. Side-by-side preview with metadata editing would be ideal.
- Mass updates occur much more frequently in an enterprise than with consumers, so there must be support to allow this without too many clicks.
- There must be notifications of changes in a saved search via RSS or e-mail alerts.

No matter how usable an interface is, groups are guaranteed to want their own changes. We are solving the last problem by providing a “very good” UI out of the box, but allowing any group to customize its own version. We can enable this using a templating engine. This allows each group to selectively override different aspects of the UI: browse top facets and search results.

Although it requires knowledge of HTML on the part of the user, this choice will allow us to go beyond skinning the same UI with CSS (cascading style sheets) to a point where new content and integrations can be added to a page without the platform team doing any work. If this is still not enough, we will also support a SOAP API so all of the data can be repurposed by another IT team.

THINGS TO WATCH OUT FOR

We have been presented with a few challenges while building this metadata catalog.

Scalability. Our initial goal is for the platform to be able to index 1 million documents, each containing metadata in various categories. This is enough to show the value of the system, but will not even come close to capturing as many documents as we would like.

Security. Security around data is a significant concern in the financial services industry. For this system, we face two distinct issues:

- **Who is allowed to update a piece of metadata?** Since each business unit is able to create its own taxonomies, it has a reasonable expectation of being able to control who can update its fields and what values any particular person is able to assign them. For example, our training department puts out “tech tip” documents for the firm. Each business, however, has a designated approver of the tips that it sees as relevant. It then adds its business unit to the metadata category “Targeted Business Unit.”
- **Who is allowed to view a particular piece of metadata about a document (or know that the document exists at all)?** While the previous security concern raises a number of policy issues, this one raises some technical limitations. It is hard enough to decide who has access to view a particular document, but if you allow this type of access control to exist per facet, the problem is immediately scaled by a large factor, depending on the number of facets in the system.

Query optimization. Until now, we have conveniently ignored an important step that happens in most successful search projects: query optimization. Users and administrators alike will want to know why a document is not ranked higher for a particular query, or why a seemingly irrelevant one got returned. These one-off issues can be fixed by tweaking the indexing, thesaurus, or ranking algorithms. Although any particular request does not take long to handle, the time commitment quickly adds up. Further, a proactive system owner should regularly go through query logs to see what users are searching for, what produces zero or few results, and which items are being clicked on the most. This is hard enough to do for a single application, but figuring out how to make this scale to hundreds of applications on the same core platform is, needless to say, quite a challenge.

FUTURE WORK

Audit trails. The financial industry is under tight regulatory control. It is essential to know who accessed or modified what content and when. Any new application must be designed with these constraints in mind, requiring a complete audit trail. The following are some of the reports that people want to include:

- For this particular document, show me every piece of metadata that has been associated with it, who added it, and when.
- For this particular category of metadata, show me which documents have been tagged, which values were given, by whom, and when.
- For this particular user, show me all the documents that were accessed within these dates, with these keywords.

Collaborative Metadata Tagging (Folksonomy). We have a few high-profile metadata catalogs that have been built over the past few years. Each, however, has required a dedicated person to finalize and approve all updates. While this worked well for these projects, it creates a high barrier to entry for projects that may not have as much funding or a well-defined structure.

Ratings. Community ratings have proven to be very successful at sites such as digg, Amazon, and Netflix; very

few implementations exist in enterprise applications, however. While the Internet offers some degree of anonymity, this is rare in a corporation. Even when encouraged by open dialogs and constructive feedback, many are still understandably uncomfortable with the potential adverse effects of an open and public rating system.

Assuming this issue is addressed, our first implementation will likely be a simple “average rating” from one to five. Each time a new rating is added, we will recompute

applications

Del.icio.us / Dog Ear

1. Metadata Schema

URL Single Line of Text
Tags Multivalue, Single Line of Text

2. Adding Data

The best tools to get data into the system are

- A bookmarklet

```
[javascript:location.href='http://server/add?url='+location.href+'&tags='+document.title]
```

- A desktop importer that crawls through a user's bookmarks. It can also add a tag for the folder name.

3. Browse Results

- Although a standard UI would work, we probably want to add a line that creates a link for each tag in each result.

This templating script would look something like:

```
[ foreach $tag in $Tags: <a href='browse?tags=$tag'>$tag</a> ]
```

Equity Research Reports

1. Metadata Schema

URL Single Line of Text
Analyst Single Line of Text
Industry Single Line of Text
Company Multivalue, Single Line of Text
Ticker Multivalue, Single Line of Text
Region Multivalue, Single Line of Text
Date of Report Date

2. Adding Data

Because research reports may already exist in another system, the best way to bring them into another catalog is to write a database crawler and use the SOAP API.

3. Browse Results

For this application, the out-of-the-box facet browsing works well.

Search Considered Integral

the average and stick that value into a facet. This allows it to be browsed and narrowed by the user just like all other facets.

Although average ratings should be helpful, a much greater jump in utility will come from the second phase. This will be when we take into account profile data to provide custom rankings unique to each user. For example, the ratings made by users within the same department should be weighted more than ratings from across the company. This system will allow us to move further along in sophistication for knowledge discovery, from “Here’s how relevant I think this document is for you” to “Here are some documents that you probably will be interested in.”

Approval workflows. Each group wants to use its own unique approval workflow for adding metadata to documents. Relieving the platform team of this work can allow quicker delivery of customized requirements. To allow full flexibility, we will integrate with a full workflow engine to abstract out the approval per ontology.

Any time someone tries to update the metadata in a particular ontology, we can start an approval workflow. For the simplest implementation, an e-mail is sent to an approval group that will simply need to click an Approve or Reject button. More complicated examples include logic such as “If someone is trying to tag a customer with the gold label, require any officer in the group to approve. If someone wants to tag a customer with the platinum label, require a managing director to approve and then send a notification e-mail to a mail group.” The business rules for this can become very complicated when dealing with complex financial information.

Consistent alerts. Users want to be proactively notified when an item is added to the system that may be of interest to them. Subscribing to search results, however, is still relatively new to most business users. Although RSS seems like the obvious solution, many traditional users do not regularly use an aggregator. This means that we need to support a number of ways to alert users. We have or plan to build modules for RSS; alert box on an intranet portal; e-mail (realtime and daily digests); and IM pop-ups.

THE RIGHT TIME

While search has been an important topic for decades, only recently has it become integral to every enterprise application. A few significant events over the past few years include:

Explosion of data. With e-mail, instant messaging, RSS, and a growing intranet, there is now so much data that many users feel they suffer from information overload.

Pervasive desktop search. Google, Microsoft, and other major players offer free desktop search solutions that greatly improve knowledge worker productivity. They also allow plug-ins to be written, which companies such as IBM have already used to connect to back-end systems.

Lucene. This is an open source search package that is seeing increasing use by third-party applications including Eclipse, Lookout, and CNET.com. Overall, 187 groups are using the Java version. Ports are also available in Perl, Python, C++, .NET, and Ruby.

Autonomy. Autonomy recently bought Verity, creating a combined set of enterprise search features that look very promising. These include categorization, clustering, and even automatic ontology creation.

Microformats. This grassroots effort is bringing structure into HTML documents on the Web. As more and more data producers provide these clues, entity extraction tools can easily increase their relevance.

OWL (Web Ontology Language). The Semantic Web is based on RDF (Resource Description Framework), but that does not solve problems for any particular domain. For this, ontologies are created, often in OWL (<http://www.w3.org/TR/owl-features/>).

Knowledge workers need better tools to handle the dramatic increase in data that they deal with daily. While much progress has been made in the market, innovation is still needed to solve the challenges of a large enterprise. The usability of tagging and classification, along with the quality of search results, will be key factors for success. When done right, this will enhance productivity for those who must transform a sea of data into actionable information. ☺

LOVE IT, HATE IT? LET US KNOW

feedback@acmqueue.com or www.acmqueue.com/forums

RYAN BARROWS is an associate in Morgan Stanley’s institutional securities technology department.

JIM TRAVERSO is an executive director in Morgan Stanley’s institutional securities technology department.

© 2006 ACM 1542-7730/06/0500 \$5.00