Spyglass: A System for Ontology Based Document Retrieval and Visualization

John Rushing, Todd Berendes, Hong Lin, Cody Buntain and **Sara Graves**
Information Technology and Systems Center
University of Alabama in Huntsville
Huntsville, AL 35899
{jrushing, tberendes, alin, cbuntain, sgraves}@itsc.uah.edu

Abstract

This paper describes the Spyglass tool, which is designed to help analysts explore very large collections of unstructured text documents. Spyglass uses a domain ontology to index documents, and provides retrieval and visualization services based on the ontology and the resulting index. The ontology based approach allows analysts to share information and helps to ensure consistency of results. The approach is also scalable and lends itself very well to parallel computation. The Spyglass system is described in detail and indexing and query results using a large set of sample documents are presented.

Introduction

Many enterprises have large collections of documents that they need to manage, search and analyze. Documents vary in the amount of structural metadata that is associated with them, from purely unstructured text, to text annotated with semantic metadata tags, to highly structured forms where each field has a specific well known meaning. In general, unstructured text documents present the greatest challenge because all information used to index, search and analyze the documents must be derived from the documents themselves. A wide variety of approaches have been used for retrieval and analysis of unstructured text documents, including keyword search, clustering (Zhao and Karypis 2005), supervised classification (Joachims 1998), ontology based approaches (Castells, Fernandez, and Vallet 2007) (Fluit, Sabou, and van Harmelen 2005), and others.

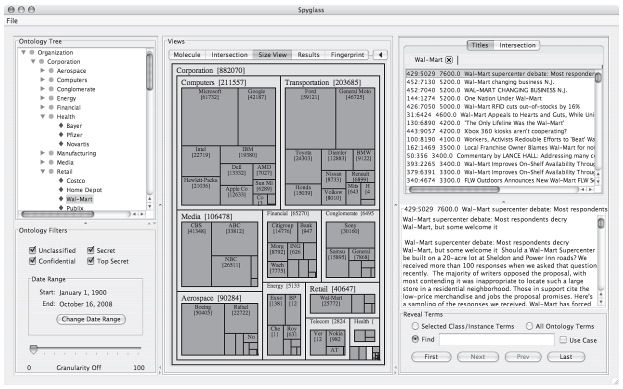
Ontological approaches to document retrieval have key advantages over pure keyword based searches. Keyword based systems leave it up to the users to define every possible term needed to find the entities of interest, with the result that analysts end up duplicating effort and possibly missing important terms. Consider for example an analyst that wishes to find all documents that refer to energy companies. There are hundreds of such companies, and each has a name, stock symbol, address, president and so on. Constructing a keyword query that includes all of those names (including singular and plural forms and common misspellings) would be quite cumbersome.

Ontological approaches solve this problem by providing a structure and vocabulary that is used to describe the domain of interest. Ontologies consist of classes (general categories of things), and instances (the things themselves). The entities in the ontology have properties that describe their characteristics, and links to other entities in the ontology. Ontologies are typically created by domain experts, sometimes with the assistance of knowledge engineers. Using an ontology based approach, the required information is specified once, and then that information is available for use in all subsequent queries. The ontology provides a means for analysts to share their knowledge about their domain of interest.

Ontological approaches also have advantages over black box analysis methods such as clustering and classification. The primary disadvantage of these methods is that the analyst does not have any understanding of how the system is functioning, and if they notice that it is making mistakes it is hard for them to fix it. The similarity measures chosen by the machine learning methods may be statistically relevant, but may be of no interest to the analysts. Ontology based systems on the other hand are easy for the analysts to understand because the information used to index and retrieve documents is explicitly specified, and can be updated and changed as needed. Another disadvantage of clustering and classification methods is that they are typically based on TF/IDF features. Although efficient parallel methods for computing these features exist (Rushing, Berendes, and Graves 2007), the dictionaries and indexes required by these methods are far larger than those generated for ontology based methods since they include every term that appears in the corpus rather than only those of interest.

The main disadvantage of ontology based approaches is the amount of effort required to create the ontology. Ontology based systems have been very successful in focused domains such as medicine (Logantharaj and Narayan 2006), but may be difficult to employ in other areas where the number of potential entities of interest is too large to specify. In those cases, it is possible to partially automate the ontology construction process by using the relationships present in the documents themselves to learn ontological structures (Bloehdorn, Cimiano, and Hotho 2005). However, even when automated methods are used work is still required to ensure quality and correct errors.

Spyglass is an ontology based document indexing and



```
429:5029  7600.0  Wal-Mart supercenter debate: Most responder
452:7130  5200.0  Wal-Mart changing business N.J.
452:7040  5200.0  WAL-MART CHANGING BUSINESS N.J.
144:1274  5200.0  One Nation Under Wal-Mart
426:7050  5000.0  Wal-Mart RFID cuts out-of-stocks by 16%
31:6424  4600.0  Wal-Mart Appeals to Hearts and Guts, While Uni
130:6890  4200.0  'The Only Lifeline Was the Wal-Mart'
443:9057  4200.0  Xbox 360 kiosks aren't cooperating?
100:8190  4100.0  Workers, Activists Redouble Efforts to 'Beat' Wa
162:1469  3500.0  Local Franchise Owner Blames Wal-Mart for no
50:356  3400.0  Commentary by LANCE HALL: Addressing many c
393:2265  3400.0  Wal-Mart Improves On-Shelf Availability Throu
379:6391  3300.0  Wal-Mart Improves On-Shelf Availability Throu
340:4674  3300.0  FLW Outdoors Announces New Wal-Mart FLW S
```

429:5029 7600.0 Wal-Mart supercenter debate: Most respondents

Wal-Mart supercenter debate: Most respondents decry Wal-Mart, but some welcome it

Wal-Mart supercenter debate: Most respondents decry Wal-Mart, but some welcome it Should a Wal-Mart Supercenter be built on a 20-acre lot at Sheldon and Power Inn roads? We received more than 100 responses when we asked that question recently. The majority of writers opposed the proposal, with most contending it was inappropriate to locate such a large store in a residential neighborhood. Those in support cite the low-price merchandise and jobs the proposal promises. Here's a sampling of the responses we received. Wal-Mart has forced

Figure 1: Spyglass Client User Interface

retrieval system with associated visualization capabilities designed to facilitate analysis of unstructured textual data. Spyglass allows analysts to visually explore potential relationships between ontology entities in the document sets of interest, and quickly access documents or contexts containing those relationships. Both indexing and query performance are primary concerns and the system is designed to scale well to very large document repositories.

Spyglass consists of two components: a client side graphical user interface and a server side indexing component. The server consists of an indexer and a web-based query service that supports both retrieval of documents and visualization. The indexer is capable of running both in single processor environments and on computing clusters using the Message Passing Interface (MPI) for communication between nodes. The Spyglass client is implemented as a Java application and has been tested on a variety of platforms including Microsoft Windows XP and Vista, RedHat and SUSE Linux and Mac OS X. A Spyglass demo is available for download at http://www.itsc.uah.edu/spyglass.

Spyglass Client

The Spyglass user interface is designed for ontology based document search and retrieval. The Spyglass user interface is shown in Figure 1. The interface consists of several components: an ontology tree browser (upper left), a title and document browser (right), a set of document filters (lower left), and a set of visualization panes (center). The tree browser shows the structure of the ontology and is used to select ontology items. The title and document browser is used to show the results of searches and to select and view documents. The document filters limit the scope of searches and visualizations, allowing the user to constrain the search by date, classification level and matching distance for intersection queries. The visualization panes show different views of the structure of the ontology and the relationships between ontology items and the document set.

The views interact with one another via ontology item selections. When a class or instance is selected in one view, it shifts the focus of the other views to the same item whenever that is possible. The primary way to select items is by

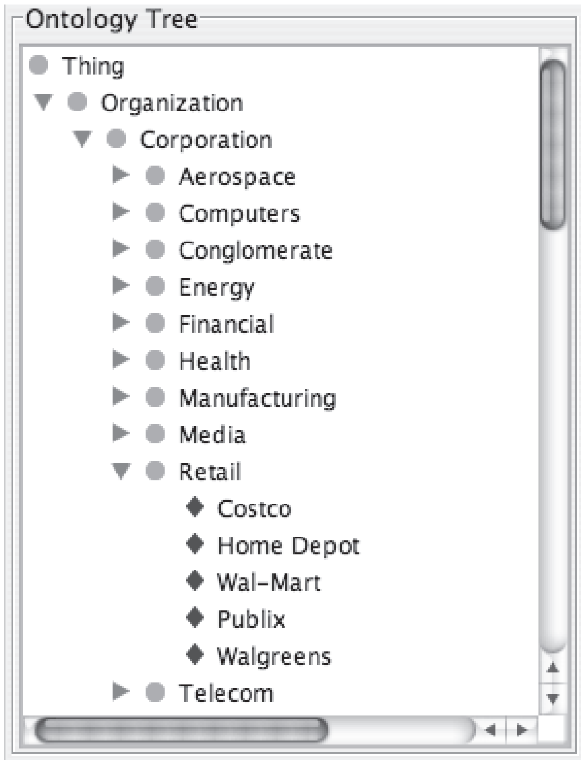


Figure 2: Ontology Tree View

double clicking on them in the tree view. Some of the visualizations also support selection while others are there merely to provide information.

Ontology Exploration

Spyglass includes several views of the ontology, each of which may be used to select classes and instances for visualization and retrieval. The ontology views show both the structure of the ontology and how the documents in the collection map onto the ontology. These views allow the analyst to see what information is available and also to identify gaps or errors in the ontology.

Ontology Tree View The ontology tree view shows the ontology in a traditional hierarchical fashion. Classes are distinguished by a yellow circle, while instances are shown with purple diamonds. Classes may be expanded to show their subclasses or component instances. The ontology tree view is the primary means of selecting ontology items. Double clicking on any item in the tree view selects that item in the remainder of the user interface. The selected class or instance is the object of focus in the visualization panes, and a list of documents ranked by relevance to the selected item is shown in the title browser. An example of the tree view is shown in Figure 2.

3D Structural View The 3D View pane is designed to facilitate a user's better understanding and intuitive feeling of the ontology displayed in the tree browser. To this end, the

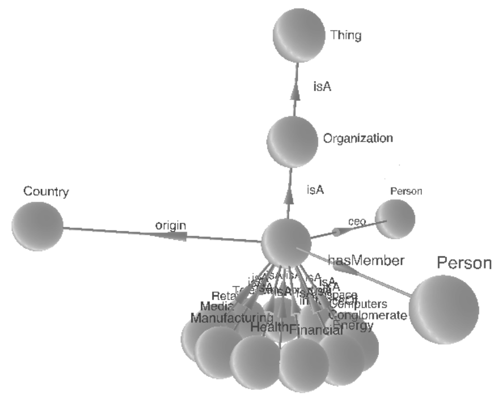


Figure 3: 3D View of the Corporation Class

scene displays the selected class or instance from the Tree Browser in the center of the view and connects that central instance with all other classes or instances to which it is related. As shown in Figure 3, the hierarchy of specialization and inheritance is shown by the "isA" relationships above the central object. Further specializations of the central object are also visible as "isA" relationships below the central object. The relations between the central object (in this case, the "Corporation" class) and other classes are denoted by the horizontal radial connections spanning out from the central object. Lastly, when the scene is focused on a class object, the instances of that class are contained within the small, purple sphere (representing composite objects) connected to the central object by the "instanceOf" relationship. This view and two other 3D views available in Spyglass are described in more detail in (Buntain 2008) and are documented in the Spyglass User Guide.

Size View The size view is designed to show at a glance how the documents in the collection map onto the ontology. The size view pane shows the number of documents associated with each class and instance in the ontology in a hierarchical fashion. Each class is displayed as a yellow box, and each instance is displayed as a green box. The structure of the layout mirrors the structure of the ontology, so the box for each class includes all of its subclasses and instances. The boxes are sized so that the size of the box on the screen is roughly proportional to the number of documents associated with the class or instance that the box represents. It is possible to zoom in or out to focus on a specific class or instance. The selections made in the filter pane limit the number of documents included in the counts, so it is possible to see the document mapping for a specific date range. An example of the size view is shown in Figure 4.

Document Browse and Retrieval

The document browse pane shows the lists of documents associated with the selected class or instance, and the results of intersection queries. A tab is created for each class or instance selection or each intersection query. Each tab con-

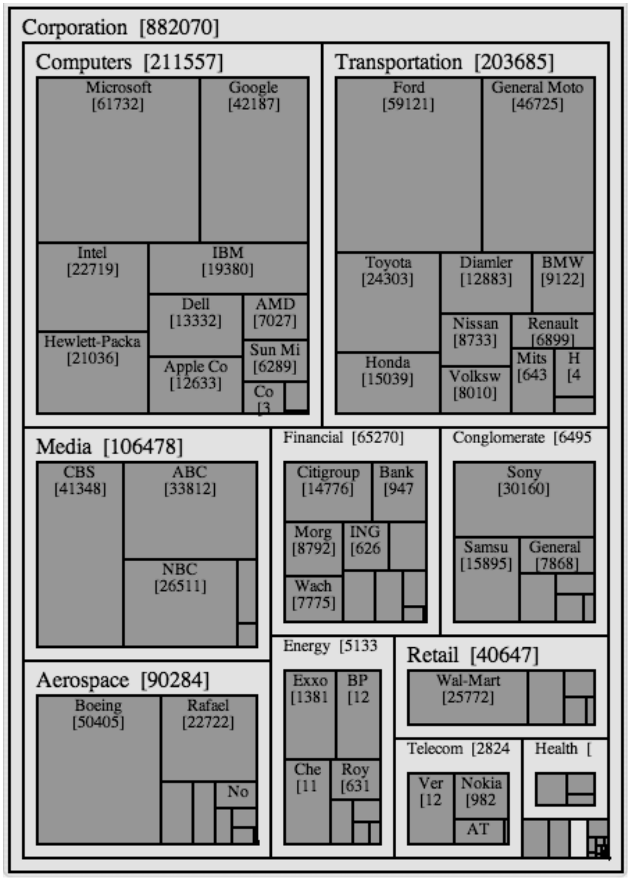
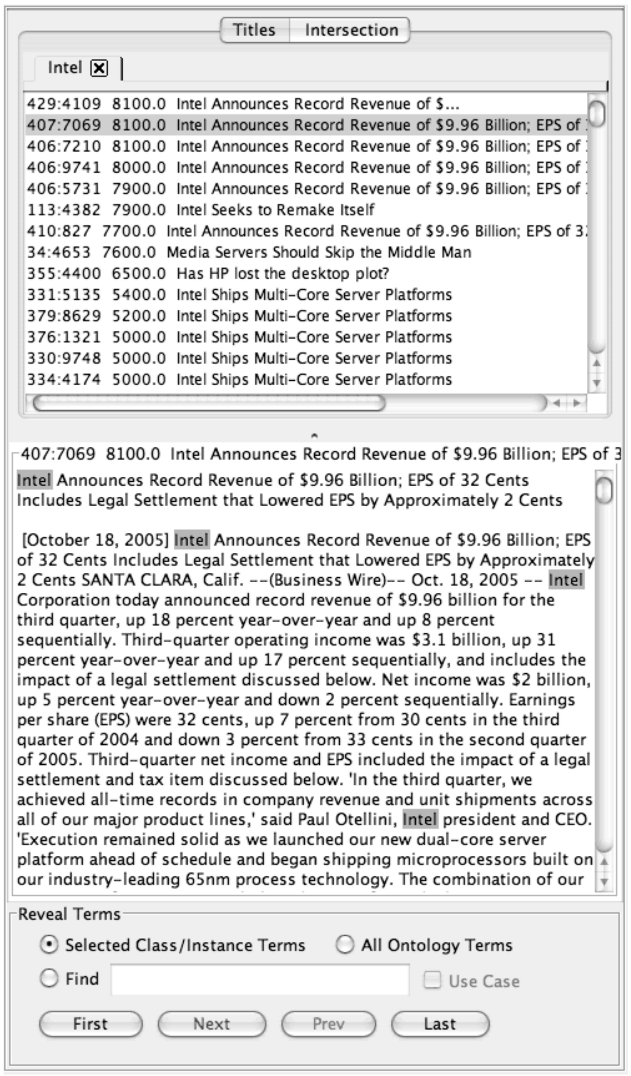


Figure 4: Size view with focus on corporation class

tains a sorted list of document headers, where each header contains the document id, document title, and score for the document with respect to the selected item. The document filters limit which documents appear in the list, so the searches can be limited by date or classification level.

Clicking on a title in a list displays the contents of the document in the document pane directly below the title list. There are several options for highlighting terms in the selected document and navigating within the document. The user can highlight the terms associated with the selected class or instance, highlight all ontology terms, or search the document for a text string by keying it in. By using the "First", "Next", "Prev", and "Last" buttons, the user can jump directly to the first, next, previous, and last highlighted terms in the text. Additionally, the arrow keys, page up, page down, home, and end keys can be used to navigate the text highlighting. An example of the document browse pane is shown in Figure 5.

Document Fingerprint View Once a document has been selected, it is possible to see its contents at a glance using the document fingerprint view. The view contains one line for every instance in the ontology that occurs in the document, sorted by the number of times the instances occur. The instances are color coded based on the class that the in-



```
429:4109  8100.0  Intel Announces Record Revenue of $...
407:7069  8100.0  Intel Announces Record Revenue of $9.96 Billion; EPS of
406:7210  8100.0  Intel Announces Record Revenue of $9.96 Billion; EPS of
406:9741  8000.0  Intel Announces Record Revenue of $9.96 Billion; EPS of
406:5731  7900.0  Intel Announces Record Revenue of $9.96 Billion; EPS of
113:4382  7900.0  Intel Seeks to Remake Itself
410:827  7700.0  Intel Announces Record Revenue of $9.96 Billion; EPS of 3
34:4653  7600.0  Media Servers Should Skip the Middle Man
355:4400  6500.0  Has HP lost the desktop plot?
331:5135  5400.0  Intel Ships Multi-Core Server Platforms
379:8629  5200.0  Intel Ships Multi-Core Server Platforms
376:1321  5000.0  Intel Ships Multi-Core Server Platforms
330:9748  5000.0  Intel Ships Multi-Core Server Platforms
334:4174  5000.0  Intel Ships Multi-Core Server Platforms
```

407:7069 8100.0 Intel Announces Record Revenue of $9.96 Billion; EPS of 3

Intel Announces Record Revenue of $9.96 Billion; EPS of 32 Cents Includes Legal Settlement that Lowered EPS by Approximately 2 Cents

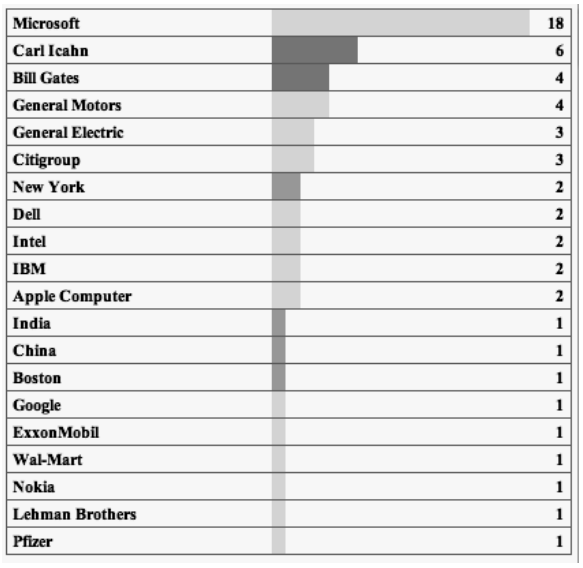
[October 18, 2005] Intel Announces Record Revenue of $9.96 Billion; EPS of 32 Cents Includes Legal Settlement that Lowered EPS by Approximately 2 Cents SANTA CLARA, Calif. --(Business Wire)-- Oct. 18, 2005 -- Intel Corporation today announced record revenue of $9.96 billion for the third quarter, up 18 percent year-over-year and up 8 percent sequentially. Third-quarter operating income was $3.1 billion, up 31 percent year-over-year and up 17 percent sequentially, and includes the impact of a legal settlement discussed below. Net income was $2 billion, up 5 percent year-over-year and down 2 percent sequentially. Earnings per share (EPS) were 32 cents, up 7 percent from 30 cents in the third quarter of 2004 and down 3 percent from 33 cents in the second quarter of 2005. Third-quarter net income and EPS included the impact of a legal settlement and tax item discussed below. 'In the third quarter, we achieved all-time records in company revenue and unit shipments across all of our major product lines,' said Paul Otellini, Intel president and CEO. 'Execution remained solid as we launched our new dual-core server platform ahead of schedule and began shipping microprocessors built on our industry-leading 65nm process technology. The combination of our

Reveal Terms

[✓ Selected Class/Instance Terms] All Ontology Terms

Find   Use Case

Figure 5: Document Browse Pane



Microsoft	18
Carl Icahn	6
Bill Gates	4
General Motors	4
General Electric	3
Citigroup	3
New York	2
Dell	2
Intel	2
IBM	2
Apple Computer	2
India	1
China	1
Boston	1
Google	1
ExxonMobil	1
Wal-Mart	1
Nokia	1
Lehman Brothers	1
Pfizer	1

Figure 6: Fingerprint View

stances belong to. A histogram is shown on the right side of the view that shows the relative frequency of occurrence of the instances in a graphical fashion. Other researchers have used similar views for literary analysis (Keim and Oelke 2007), mapping contents of a document onto a a color coded grid. These views allow the user to quickly compare document contents visually without having to read the entire document. An example of the document fingerprint view is shown in Figure 6.

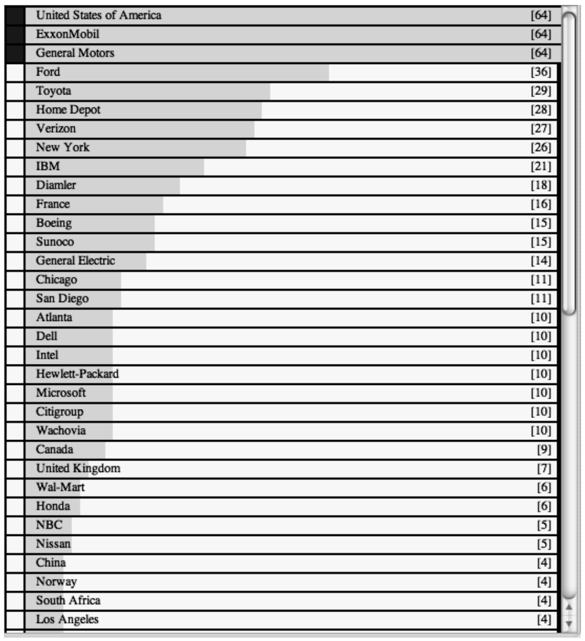
Exploring Relationships Between Entities

Spyglass has several views dedicated to exploring relationships between ontology entities. One purpose of these views is to help the analyst identify places where different topics of interest appear in the same context. The views can also give an indication of which entities may be related to one another by showing how often the entities co-occur in the document repository.

Intersection View The purpose of the intersection view is to locate documents that contain multiple entities of interest, and to see which instances occur together in one or more documents. When an instance is selected, the view shows the number of documents associated with that instance. In addition, the view shows the number of documents that contain both the selected instance and every other instance in the ontology. The other instances are sorted by the frequency of cooccurrence with the selected instance. When multiple instances are selected, the view shows the number of documents that contain both all of the selected instances and any of the other instances in the ontology.

Selected instances are denoted by filled boxes to the left of the instance name. Instances can be added to the intersection or removed from it by clicking on the box. When instances are selected in the intersection view, a list of documents containing those instances is displayed in the intersection tab on the document browse pane. It is possible to limit the scope of the intersection search by date range and by granularity using the options set in the filter pane. If a date range is specified, only those documents that match the date range will be returned. The granularity slider can be used to control the size of the context in which to search for intersections. The granularity can be set from document level (default) down to a context of a few words. This is useful especially when the corpus has large documents that may have multiple topics.

An example of the intersection view is shown in Figure 7. In this view, the instances “United States of America”, “ExxonMobil” and “General Motors” are selected. A total of 64 documents in the repository contain references to all three of those instances. On the next line down the entry for “Ford” indicates that 36 documents contain references to “Ford” and the three selected instances. The view shows all the possible candidates for narrowing the search.



United States of America	[64]
ExxonMobil	[64]
General Motors	[64]
Ford	[36]
Toyota	[29]
Home Depot	[28]
Verizon	[27]
New York	[26]
IBM	[21]
Diamler	[18]
France	[16]
Boeing	[15]
Sunoco	[15]
General Electric	[14]
Chicago	[11]
San Diego	[11]
Atlanta	[10]
Dell	[10]
Intel	[10]
Hewlett-Packard	[10]
Microsoft	[10]
Citigroup	[10]
Wachovia	[10]
Canada	[9]
United Kingdom	[7]
Wal-Mart	[6]
Honda	[6]
NBC	[5]
Nissan	[5]
China	[4]
Norway	[4]
South Africa	[4]
Los Angeles	[4]

Figure 7: Intersection View

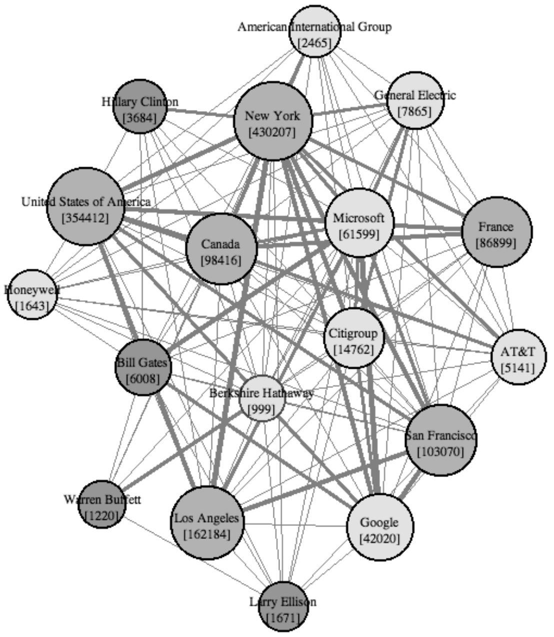


Figure 8: Information Molecule View

Information Molecule View The information molecule view provides a visual means of exploring how ontology instances are related in the document set. The view shows a circle for each selected instance where the size of the circle is proportional to the log of the number of documents related to that instance. A line is drawn between each pair of selected instances where the thickness of the line is proportional to the log of the number of times the instances cooccur. As in the intersection view, the granularity slider is used to determine the distance in the document used to check for matches.

Instances can be added to the view by selecting those instances from the tree browser on the left hand side of the interface. In addition, there are several shortcuts that can be used to construct the view. If a class is selected, all instances of that class are added to the view. There is an "Expand" feature that adds to the view all instances that have cooccurrences with the selected instance. There is also a "Purge" feature that removes from the view all instances that do not have any cooccurrences with the selected instance. These features serve a similar purpose to the interactive list view of the Jigsaw tool (Stasko et al. 2007), allowing users to iterate through multiple related entities or concepts to find the topics of interest.

When instances are added they are placed relative to the instances already in the view. The placer puts instances close together if they share documents, and also tries to spread the view out so that no instances overlap. The view can be automatically replaced with the selected instance in the center by right clicking anywhere in the view. An example of the information molecule view is shown in Figure 8.

Spyglass Server

The Spyglass server is a web based service that accepts queries from the client and returns documents and other information used for the visualizations. The server comes with a set of utilities to read an ontology and index a collection of unstructured text documents. The documents must be packaged and indexed before the server can make them available to the client. The server components are all implemented in C++ and have been built and tested on a variety of UNIX-like environments including RedHat Linux, SUSE Linux and Mac OS X.

Ontology Based Indexing

The Spyglass server begins by packaging the input text files into binary packages, typically with about 10000 documents per package. This is done for performance reasons, as most file systems do not handle large numbers of files very well. For example, the news data set typically used to test the Spyglass application has just over 4.5 million documents in it, and it takes about 10 hours to read the documents and create the text packages. Once the packages are created, the indexer is able to read and index the entire document collection in less than ten minutes. The packaging takes place once as the documents are added to the system.

Once the binary text packages are created, the ontology is used to index the packages. Spyglass currently works with ontologies created by the Protege tool (Protege 2007), although ontologies in other formats may be supported in the future. The indexer constructs a state based parser directly from the terms in the ontology. All of the instance and class names are incorporated directly. In addition, the indexer looks for a special slot labeled "aliases" that contains additional names for the instance or class. Typically, the aliases would contain alternate names, nicknames, acronyms, common misspellings and any other terms that could by used to identify the object in question.

The state based parser used by the indexer requires only one pass through the text no matter how many terms are in the ontology. For each term match, the indexer writes the document id, term id, and offset within the document to a binary file. The indexer produces a file containing all the hits ordered by document, and then uses that to produce additional indexes ordered by instance and by class. If new documents are added to the repository, the indexer will work incremntally, running only on those documents that have not already been indexed.

Web Based Services and Queries

The Spyglass server is implemented as a web based service. Initial versions of the server were based on relational databases, but the queries required by the intersection and molecule views required custom services to achieve suitable performance. The intersection view requires the counts for the number of documents or contexts shared by the selected instances and every other instance in the ontology. The molecule view requires the intersection counts for every pair of instances in the ontology. These values cannot be precomputed because they are different for any combination of filter parameters (date range and granularity) required. The current implementation is based on shared memory. The table required for the molecule and intersection queries is loaded into a shared memory segment on the first invocation of the server. The table remains in shared memory for all subsequent invocations. Some of the queries supported by the server include:

- Retrieve the files for an ontology
- Retrieve the text for a document by document id
- Get scores for each instance and class for a document
- Get a sorted list of documents associated with a class
- Get a sorted list of documents associated with an instance
- Get the number of common documents or contexts for every pair of instances
- Get the number of common documents or contexts for a set of instances intersected with every other instance

For the test document set of 4.5 million documents indexed with an ontology containing 1882 terms the server is able to perform all but the pairwise intersection queries in one second or less. The pairwise intersection query takes about three seconds on average, but it is required only when the filter parameters change, as the pairwise intersection counts are cached on the client. The results were achieved using a server running on an iMac with an Intel Core 2 Duo processor running at 2.16 GHz and 2 GB of RAM.

Conclusions

The Spyglass system for indexing, retrieval and visual analysis of unstructured text documents has been described. Spyglass is a scalable platform independent software tool capable of indexing millions of documents and supporting interactive query response for visualization. Spyglass is based on domain specific ontologies, which allow analysts to specify entities of interest and share information with one another. The use of ontological indexing allows the system to scale better than products based on TF/IDF features or other similar structures because only terms of interest are indexed rather than all terms. The ontological structure also provides a starting point for visualization, and when the documents are mapped onto the ontology it is possible to see relationships among the entities that are revealed by the cooccurrence of those entities within the document set. Spyglass has a rich set of visualization tools that help analysts identify and explore these relationships.

References

Bloehdorn, S.; Cimiano, P.; and Hotho, A. 2005. Learning ontologies to improve text clustering and classification. In *Proceedings of the 29th Annual Conference of the German Classification Society (GfKl)*. Springer.

Buntain, C. 2008. 3d ontology visualization in semantic search. In *The 46th ACM Southeast Conference*.

Castells, P.; Fernandez, M.; and Vallet, D. 2007. An adaptation of the vector-space model for ontology-based information retrieval. *IEEE Transactions on Knowledge and Data Engineering* 19(2):261–272.

Fluit, C.; Sabou, M.; and van Harmelen, F. 2005. Ontology-based information visualization: towards semantic web applications. In Chen, C., and Geroimenko, V., eds., *Visualizing the Semantic Web*. Springer-Verlag.

Joachims, T. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, 137–142.

Keim, D. A., and Oelke, D. 2007. Literature fingerprinting: A new method for visual literary analysis. In *IEEE Symposium on Visual Analytics Science and Technology*, 115–122.

Loganantharaj, R., and Narayan, V. B. 2006. Sempub: An ontology based semantic literature retrieval system. In *Proceedings of the 19th IEEE Symposium on Computer-Based Medical Systems*, 875–880.

Protege Project: Stanford University. 2007. *http://protege.stanford.edu*.

Rushing, J.; Berendes, T.; and Graves, S. 2007. Efficient parallel computation of inverse document frequency features for text mining. In *The 2007 International Conference on Data Mining*.

Stasko, J.; Gorg, C.; Liu, Z.; and Singhal, K. 2007. Jigsaw: Supporting investigative analysis through interactive visualization. In *IEEE Symposium on Visual Analytics Science and Technology*, 131–138.

Zhao, Y., and Karypis, G. 2005. Hierarchical clustering algorithms for document datasets. *Data Mining and Knowledge Discovery* 10(2):141–168.