The Morpheus Visualization System: A General-Purpose RDF Results Browser

by

Akash G. Shah

S.B., Electrical Engineering and Computer Science, M. I. T., 2008
S.B., Management Science, M. I. T., 2008

Submitted to the Department of Electrical Engineering and Computer Science in Partial Fulfillment of the Requirements for the Degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2009

Author.....
Department of Electrical Engineering and Computer Science
December 17, 2008

Certified by.....
Michael Stonebraker
Adjunct Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by.....
Arthur C. Smith
Professor of Electrical Engineering
Chairman, Department Committee on Graduate Theses

The Morpheus Visualization System: A General-Purpose RDF Results Browser

by

Akash G. Shah

Submitted to the Department of Electrical Engineering and Computer Science on December 17, 2008 in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Electrical Engineering and Computer Science

Abstract

As the amount of information available on the deep web grows, finding ways to make this information accessible is growing increasingly problematic. As some have estimated that the content in the deep web is several orders of magnitude greater than that in the shallow web, there is a clear need for an effective tool to search the deep web.

While many have attempted a solution, none have been successful in effectively addressing the problem of deep web searching. The Morpheus project presents a unique approach to the problem as it integrates the deep web with the shallow web while preserving the semantics of the deep web sites it accesses. At the heart Morpheus is its visualization system which allows users to access the deep web information. The visualization system makes use of clustering algorithms, visual information techniques, as well as the semantics of the deep web sites stored by Morpheus to present deep web results to users in an effective manner.

User testing was also conducted to identify problematic areas of the system during development as well as to evaluate the usability of the system's design. Results indicate that users find that the Morpheus visualization system is a highly usable and learnable interface for searching the deep web for results as well as for processing those results.

Thesis Supervisor: Michael Stonebraker
Title: Adjunct Professor of Electrical Engineering and Computer Science

Acknowledgements

I would like to thank Mike Stonebraker, my advisor and mentor, without whom this thesis would not have been possible.

Table of Contents

List of Figures

List of Tables

1 - Introduction

1.1 - Motivation

Most search engines have focused on providing search capability for the shallow web, i.e. those static web pages that are findable by a conventional crawler. In contrast, there is also a deep web of information, whose pages are dynamically generated from data entered into forms. The deep web includes most travel sites, weather.com, most shopping sites, 411.com, Mapquest, currency converters, all credit card sites, etc. It has been estimated that the deep web is at least 100 times the size of the shallow web. [3, 27, 59]

Information in the deep web is invisible to most crawlers, so current search engines must either resort to other techniques for access or ignore information in the deep web. Often, only a small collection of deep web sites, such as weather.com, are supported with a site-specific algorithm. Alternately, a larger collection of sites may be utilized, but only a sample of the information on the site is surfaced. In either case, a user looking for deep web information is likely to be frustrated, unless he knows and visits the specific site that holds the data of interest. [3, 5, 27, 39, 55, 59]

The Morpheus system represents an attempt to solve the problem of indexing the deep web. Originally designed as a data integration tool, the system utilized a repository of transform functions and a database to achieve this integration. Now, however, the second iteration of the Morpheus system attempts to address the problem of indexing the deep web by treating a web form as a series of inputs and outputs, which Morpheus handles by creating a function called a "wrapper." A Morpheus wrapper maps a collection of a web form's inputs to its outputs, allowing the web form itself to fetch the information. Therefore, the system is well suited to wrapping pages from the deep web. Moreover, the advantage Morpheus has over other efforts to index the deep web is that Morpheus preserves the semantics of the pages indexed in the deep web for query exploration. As the web forms are wrapped

by a human in a manner that records the associated metadata, or context, of the form, the Morpheus system is able to preserve the meaning of the web form. For instance, the type of information required as input or produced as output, as well as a description of the purpose of the web form being wrapped are stored in Morpheus. Using this metadata, results returned from the repository for a user query are in the RDF data format [64]. The RDF format contains a set of subject, predicate, object triples that allow machines to understand meaning and relationships. In other words, with RDF the system is able to distinguish different types of information from one another, recognizing the difference between an address and a telephone number, for instance.

This data format then allows the system to make inferences about relationships among the results. For instance, using these semantic relationships the system can deduce that several results obtained are very similar in nature, allowing the system to place them under a single grouping. In the example above, all addresses may be grouped separately from all telephone numbers, for instance.

1.2 - The Morpheus Visualization System: A General Purpose RDF Results Browser

From a user's perspective, integrating the shallow and the deep web is useless if the user does not have a proper tool to view the results of his queries. The user must have a way to understand and process the results obtained from a search. Therefore, the results browser is an integral component of the revised Morpheus system. This browser seeks to display results in an orderly and meaningful fashion to the user.

The primary goal of the visualization system is to provide usability. This is accomplished by providing tools to make searching for results an efficient, orderly, and manageable process. Efficiency is important because Morpheus would not represent a viable replacement for current deep-web query techniques if it took users too long to find information. Order and manageability are important as well

because Morpheus becomes unusable if finding a particular result among the many returned from a query becomes an infeasible or unmanageable task because there is too much information on the screen.

In support of these goals, the visualization system makes several tools available to the user. Many of these tools are aimed at providing information through visual cues, such as positioning, color, size, and proximity. Others aimed at providing alternative views on the same data include a textual view, clustering, and search facilities. Additional tools help focus on manageability by allowing the user to prune the search results.

Together, these tools all work to make the task of searching the deep web through Morpheus a viable replacement for current methods of scouring the shallow web for web forms and testing their outputs. What follows is a description of the implementation of the visualization system for Morpheus. Chapter 2 discusses related work on the problem of searching the deep web while Chapter 3 gives an overview of the Morpheus approach to attacking the deep web problem. Chapter 4 gives background on RDF visualization systems, clustering algorithms, as well as visualization and user interface designs while Chapter 5 details the solution for the visualization system in Morpheus. Finally, Chapter 6 contains an evaluation of the system and Chapter 7 offers some conclusions to be drawn from this work.

2 - Background on the Deep Web

2.1 - Deep Web

In contrast to the shallow web, which contains primarily static textual information that is indexed by conventional crawlers and made available to users via a search engine, the deep web represents a collection of web pages that remain largely hidden from most traditional search engine crawlers as they contain information only accessible through forms or other types of data structures or content delivery (e.g. AJAX enabled pages). Experts have estimated that the amount of information contained in the deep web is orders of magnitude greater than that of the shallow web and that users are generally searching only a small percentage of the information on the web when using traditional search engines. While most web users utilize search engines to find information, the majority of these users also have problems finding desired information. Currently, the only effective solution for searching the deep web is for a user to memorize the location of a web form or web object that could potentially satisfy his web query. Thus many have said it is imperative to have a search engine which accesses both shallow and deep sites simultaneously. [3, 21]

2.2 - Related Work on Deep Web Integration

There has been a large volume of research on the topic of data integration. Most of this has been directed at federating disparate schemas, such as would be found inside an enterprise or between collaborating enterprises [9, 13, 14, 18, 25, 33, 34, 36, 48, 56]. However, none of this body of literature tries to integrate the disparate data in the deep web or to integrate the deep web with the shallow web.

Four research or commercial efforts are closer to our goals. Google [39, 40], Yahoo [11, 12] and Endeca [59] have active commercialization efforts in this area. There is also an active research project at the University of Illinois [7, 8]. We describe each and indicate how the Morpheus approach is quite different.

Halevy's approach at Google to deep web integration [39] is to crawl the web looking for web forms. However, he then proposes repeated call to the URL with disparate input data, collecting the corresponding output pages and indexing them using the standard Google system. Hence, he will find example input (as much as possible) and collect output in advance, which is then keyword-indexed. This approach has the significant advantage that no wrappers are required. Hence, it has the possibility of scaling to the whole web, which is the primary Google objective.

In contrast, Morpheus requires wrappers, which limit its scalability to specific subsets of the web. However, the Google approach has three notable disadvantages. First, it loses all semantics from the deep web and second, it will not work for dynamic web pages (such as the flight status of UA 179) which change rapidly. Lastly, there is no guarantee that it will find information stored in the deep web. For example, using Google to search for students at Bucknell University, some students are located while others are not found. Complete coverage is difficult for deep web sites with a large number of inputs when one is attempting to enumerate all possible inputs.

Ramakrishnan's approach at Yahoo is to build vertical market portals, as evidenced by his DBMS researcher portal [12]. Hence, he picked a narrow focus and then built a custom portal for this restricted market, with many months of programming effort. He then hopes to "stamp out" sites for other markets with less effort. In essence, he is trying to find a way to build narrow sites such as Orbitz or Zillow.com, which cover modest number of deep web pages. In contrast, Morpheus is a broader integration effort, which has a single interface for everybody. Of course, custom sites could be built on top of Morpheus.

Endeca is a privately held company in Cambridge, MA that has focused on faceted search, appropriate for the e-commerce market, where they have been quite successful. Of course, their intent is to move beyond the consumer e-commerce market to broader data integration issues. According to their CTO [59], they plan to pick a few vertical markets (e.g. drug discovery, news reporting in financial

services) and build custom vertical-market portals. They intend to choose markets carefully, so that limited customization for each enterprise is required. They plan to integrate enterprise data that is typically not available over the web and provide a unified search engine for this information. In contrast, Morpheus is focused on the broad market and on web-accessible data, whether structured, semi-structured or unstructured.

Chang at Illinois is also focused on integrating the deep web [7, 8]. Like Morpheus he crawls the web and builds wrappers. Unlike Morpheus, he has no meta-data repository of semantic information. Hence, he has no way of capturing the meaning of the returned data. In contrast, Morpheus is focused on capturing the semantics of the deep web for query exploitation.

3 - Morpheus

3.1 - System Design

The overall Morpheus system consists of three main components, a frontend GUI and visualization system, a query engine, as well as a Postgres backend. While the visualization and query engine are application code, the wrappers within the Morpheus system allow the Postgres backend to interpret information contained within the deep web. Morpheus also contains Postgres functions, called "wrappers" in Morpheus, which map a web form's set of inputs to its outputs, wrapping the function of a web form. Additionally, each of these wrappers is placed into a category hierarchy, allowing a user of Morpheus to either narrow or broaden the scope of a query by looking into the category hierarchy. For instance, a wrapper searching a generic travel website may be categorized under the "Travel" category, while a more specific wrapper searching only tropical island destinations may be categorized under the "Tropical" sub-category within the larger "Travel" category.

The interaction model between these components is fairly simple. Once a user submits a query, the query is sent to the query engine which decides which wrappers to execute from the category hierarchy and how to run the corresponding Postgres queries to obtain the search results. Then, the Postgres backend utilizes the stored wrappers to execute the query in the deep web, and the results are ultimately handed off to the visualization data model which processes these results, and are displayed to the user in a graphical form.

What follows is a more detailed explanation of the components of the Morpheus system.

3.2 - Postgres Backend

The Morpheus Postgres backend is largely a collection of Morpheus wrappers, which are represented by Postgres user defined functions. All return and input types within this database are

represented by user-defined complex types within Postgres. Consequently, all wrapper logic is pushed down into the database itself and running a query appears to the system as if the database is running a stored procedure.

All Morpheus information is stored in Postgres tables within the database, called a repository within Morpheus. The repository holds all Morpheus objects: wrappers, data types, and category hierarchies. Each wrapper also has its own metadata, or information describing its purpose and function, in addition to information about the page it wraps. For each wrapper and data type, the Postgres backend stores the name of the person who created the wrapper, the date it was created, a description of what the wrapper does, as well as its classification in the category hierarchy. Each wrapper also has information about the types it accepts as input and produces as output.

Also contained in the Postgres database is the category hierarchy. This hierarchy helps the Morpheus system categorize objects and helps focus a user's search. While many objects can clearly be classified under a single domain, Morpheus also allows a single object to be categorized under multiple domains.

This database acts as the means through which results are obtained from the deep web. Once the user submits a search query, the appropriate wrappers are found in the database. Using this information, the system is then takes the user's input values, feeds them to the wrappers, and the examines the output produced by the wrappers. As mentioned above, to the rest of the system, it simply appears as if a stored procedure has been called.

3.3 - Query Engine

The goal of the query engine is decide which wrappers in the category hierarchy to execute and then to run the corresponding Postgres queries that will perform the correct actions, one per wrapper. Three pieces of information are passed to the query engine as part of a Morpheus query. The first

represents the parameters passed to the wrappers as inputs, the second represents what the user is looking for, and the third represents the characteristics of the information the user is searching for. This last piece is what the system uses to determine which wrappers to use.

First, the query engine performs a relevance calculation for each wrapper, favoring wrappers that are deep in the category hierarchy over more shallow ones, and favoring wrappers that match more of the desired information over ones that match less. The current query engine uses standard information retrieval techniques (e.g. stop words and synonyms) to assist in this calculation. It attempts to limit the number of wrappers it queries much as possible in order to reduce complexity and time. Once wrappers are selected, all chosen wrappers are executed by running the appropriate Postgres queries.

Finally, the system will inevitably encounter the scenario where it is unable to narrow the field of potential wrappers to query. This case can arise, for instance, if the user attempts to run a query that is too general or lacks enough information. In such a case, rather than querying every possible wrapper it can find, it asks for more information. In a similar fashion, if Morpheus cannot find any wrappers to query, it comes back with an immediate response that the system could not find that information, rather than providing the user with a set of irrelevant responses. This differs from the approach of most search systems which attempt to produce some results without regard to relevance.

3.4 - The Visualization System

The result of any Morpheus query is a collection of information returned from these Postgres queries. The returned information can be easily converted into RDF triples [64]. Each output value is the "object" of an RDF triple, the function name is the "verb", and the input parameter is the "subject". Hence, the result of the collection of Postgres queries is a collection of RDF triples. Notice that these triples do not obey any global schema; in fact, it is silly to think there might be a global schema for the

deep web. As such, there is no guarantee that the Morpheus outputs presented mean the same thing or are in the same units. These RDF triples form the basis for the groupings and visual cues created by the system.

This visualization system acts as the start and end points to a user query. The graphical user interface, which is part of the visualization system, contains an area for a user to perform query operations such as executing, clearing, saving, or loading queries. Here, the user submits a query before it is passed to the query engine. Once a response is received to the query, the visualization system takes over once again.

Next, the results to a query are processed by the visualization system to analyze relationships among the results in an effort to group similar results together. Once the processing is complete, the results are displayed within the visualization system. The visualization system outputs a graph of shapes, representing results, and links, representing relationships, which the user can interact with by manipulating the objects on the screen. The graph contains an anchor node at the center of the space for the input parameter. For instance, if the user searched for the phone number of a friend, the center node would represent his friend since it is the only piece of information the system knows that all results have in common. While duplicate values are suppressed, the system keeps track of and displays the number of times a result occurred if there are duplicates.

The visualization also attempts to place the most relevant information at any given time closer to the center of the screen, by drawing stronger results or results the user has interacted with closer the center. Additionally, the visualization system attempts to reduce clutter in the space by creating a hierarchy of objects to be displayed on screen, at which point the details of each result can be seen by simply clicking on the node representing the cluster which performs a "zoom" operation.

This separation between the data processing and front end within the visualization system was created to allow for better reuse of code. The visualization system attempts to follow a model-view – controller design paradigm. In its implementation, the data model which processes the results obtained by the Postgres runtime environment is separated from the view which displays the information and interacts with the user. The controller, which is intended to handle user interaction, is effectively coupled into the view as its functions are very closely the job of displaying the information. This added layer of abstraction allows for the potential for extensibility, allowing for the possibility of replacing just the view to create a web-based interface to Morpheus, for instance.

4 - Related Work

4.1 - RDF Visualization

Several models have been proposed in the past for visualization of RDF data. These models range from ones tailored to specific applications to general purpose tools, which are able to display data with any semantics. Each approach has its strengths however and the designs generally face a tradeoff between usability and flexibility.

The most narrowly focused type of RDF visualization tools come in the form of interfaces that are tailored to just one application and can display relevant information in an extremely meaningful manner. For instance, a user could encounter a photo annotation tool based on RDF [20, 30, 45], with annotations such as the time and location the photograph was taken. Since the number of properties associated with the set of data is limited, the results can be displayed meaningfully to the user as a map or a timeline with points of interest [20]. This type of visualization provides a highly meaningful output to the user who can get a sense of where and when the images were taken. This visualization makes use of the available semantics to create a tailored user interface. However, this approach becomes much more difficult as one tries to generalize the dataset.

With a more generic set of data, it is increasingly difficult to determine the context of an individual result. Moreover, it is difficult to produce a single visualization that can be as tailored as the example photo application when there are multiple types of results in the set of results. For instance, one result may be a photo object with location and time information while another result may represent an MIT student with class standing and GPA information. Assuming the photo is unrelated to the student, in this example it is difficult to come up with a single visualization that takes advantage of the semantics for both types of results [49].

A first attempt at generalizing the visualization comes in the form of a faceted approach [4, 31, 44]. In this approach, the tool contains a set of templates, called facets, for displaying the information. Based on the type of information contained in a result, a different visualization template is used. In the previous example of the photograph and the MIT student, the tool utilizes one template to display photograph information on a map, for instance, and another tool to display biographical data when the student is selected. Therefore, while this approach is able to accommodate varying types of data, this approach can be utilized only if there are a limited number of result types as a new facet must be created for each result format. Moreover, the expected types of results must be known in advance to allow the implementer to create the facets. Therefore, it is clear that the faceted approach cannot generalize to arbitrary data sets.

To accommodate generality, a class of browsers has been created that are generic RDF browsers. These browsers do not need any prior knowledge about the context of the result data that is being displayed and can display results in a form that is general enough for any context. However, this flexibility comes at the expense of usability of the interface.

The simplest type of browsers within this class are text-based RDF browsers [58]. These browsers display results as text lists. Information about a specific result can generally be found by clicking on the text description of a result. Related objects may also be found by clicking on properties of a result. However, these text-based browsers, while simple, have the limitation that it is difficult for a user to visualize relationships among multiple results. For instance it would be more difficult to see that three results were related with a text-based approach than with a more visual approach.

Therefore, perhaps the best way to visualize RDF data of an unknown context are graph-based RDF visualizations. Since any RDF dataset can be represented as a graph, these visualizations take advantage of this property and create either a tree [52] or a graph of interconnections [16, 19, 23, 26,

41] for the RDF data. These visual cues allow the user to better narrow down the result set to find results of interest. However, with these interconnections, the visualization soon becomes unmanageable for the user as the number of connections simply create a tangled web for the user to look at.

Fortunately, some of these graph-based RDF browsers take steps to reduce this complexity. For instance some browsers attempt to group related results into clusters in an effort to present a cleaner visualization for the user. This feature allows the user to focus on just a few aspects rather than forcing the user to compare all the results at once. Other enhancements include the ability to re-center the visualization, zoom into the image, as well as manipulate the nodes in the graph. These enhancements make navigating the graph a more manageable task for the user in large datasets. [16]

Since the results returned by Morpheus do not fit a single schema description, the narrowly-focused approaches will not work for the system. Moreover, because there is essentially no limitation on the context of a single result, the faceted approach also fails to meet the needs of Morpheus. The faceted approach would require designing a template and visualization for every possible result context, and this task is simply infeasible. Therefore, the RDF results browser interface will need to implement one of the more generic RDF browsers, which are able to accommodate any set of results.

4.2 - Clustering Algorithms

Numerous methods for clustering data have been developed in the past. These methods group data points together based on their similarity, often utilizing a distance metric to evaluate their proximity along some dimension. At the highest level, these clustering approaches can be broken into two groups: hierarchical and partitional clustering methods [34].

Because they are often easier to implement, partitional clustering methods are often a more popular choice, with the k-means clustering algorithm being the most popular of the group [38, 67]. The k-means algorithm attempts to minimize the squared error from the centroid, or mean, of each cluster. The process starts with a random assignment of data points to clusters and the algorithm keeps reassigning data points to different clusters based on their distance from the mean. This process continues until there is no further reassignment of data points from one cluster to another. However, this k-means algorithm has its flaws. First, if the initial assignment of data points is not chosen carefully, the algorithm could end up hitting a local minimum of the squared error rather than the optimal solution. Second, and more significantly, the k-means algorithm draws hard lines between its clusters, where a value can be in at most one cluster and cannot be shared by multiple clusters. This strict clustering approach can prove problematic when dealing with the sets of results in Morpheus, since multiple clusters can contain results that take on the same value. This problem can arise, for example, if a user searches for an individual's email address and the wrappers return information about a collection of individuals. In this case, if gender is one of the types returned, there are only two possible values this type can take on, male or female. The k-means algorithm will try to place all information about males into one cluster and all information about females into a second, without considering the fact that that none of the other information belongs together and that this is not a logical grouping. Thus, k-means would not suffice for Morpheus. [38, 67]

Fuzzy c-means clustering improves upon the k-means algorithm in this respect. The fuzzy approach creates a mapping from each data point to the set of clusters it belongs to. This fuzzy algorithm addresses the concern in the example above, allowing the same value to be a part of many clusters. However, the fuzzy c-means method has its own problems. Like the k-means algorithm, it can still hit a local minimum rather than the optimal assignment if the initial assignment is not chosen carefully. More relevant to the goal of Morpheus, however, are the difficulty in creating a mapping

function to determine which clusters a point belongs to, the difficulty in determining the number of clusters to use, as well as the difficulty in formulating a distance metric. [38, 67]

There has been much work done on the problem of determining the number of clusters. One common method of solving for the number of clusters involves no longer adding clusters when the benefit to the quality of the model is low. More specifically, the process involves graphing the explained variance and finding the point where the marginal benefit to adding a cluster begins to decrease [50]. In other words, clusters are added until the rate of improvement of the clustering begins to decline. This method has been shown to be both efficient and effective in determining the number of clusters [50]. However the problem with using this "L-method" is tied to one of the other reasons the c-means algorithm falls short for the purposes of Morpheus: the inability to formulate a distance metric for results in Morpheus. [21, 35, 50]

The largest obstacle to formulating a distance metric stems from fact that Morpheus acts as a generic search tool for the deep web. As described earlier, the general purpose nature of Morpheus makes it impossible to determine the context of a user's query. This is problematic for two reasons. First, it is entirely infeasible to create a distance metric which could potentially compare results of the same type for every possible data type that Morpheus may return. Not only is it difficult to determine what types exist in this space, but the space is also continually growing as new wrappers are created. Second, it is also infeasible to be able to compare two results of different types. In a set of Morpheus results, two results of differing types could easily be more closely related than two results of the same type. For instance, an individual's name and phone number would be more closely related than the name of that individual and that of another individual. Thus, in order to have a usable distance metric for use in k-means, c-means, or other partitional algorithms, distance metrics would need to be created

for every combination of every possible return type in Morpheus, a task that is highly impractical. [24, 67]

Thus, the focus for Morpheus turns to hierarchical clustering methods. The first place to look was the agglomerative clustering technique [17, 22, 24, 67]. Generally, the algorithm begins by assuming each result is a separate cluster. It then determines how similar two clusters are and can merge their two clusters to form a larger one. This process continues until some termination condition is met. However, most traditional hierarchical clustering methods still require distance metrics [34]. Moreover, the use of such distance metrics has been shown to be inappropriate for boolean or categorical attributes, such as those dealt with by Morpheus, because they cannot be readily placed on a coordinate plane like numerical attributes can. Fortunately, these techniques can be adapted to look at relationships among links between results [24, 66]. By examining what neighbors nodes in the graph have in common, agglomerative clustering techniques are able to produce superior results for categorical data [24]. Therefore, this agglomerative option is a suitable choice for Morpheus.

A similar technique to agglomerative clustering is divisive clustering. This technique begins by grouping all results into one cluster, then proceeding to break off dissimilar results to form new clusters [22, 51]. As was the case with agglomerative clustering, this technique can be adapted to examine link structures as well, choosing weakly linked results to be broken off from a cluster.

To address the problem of finding a suitable metric, one can turn to a ranking system developed by the founders of the popular search engine, Google. The search engine ranks pages based on an algorithm that treats searching the web as a random walk. Web pages are ranked higher if there is a higher probability of randomly ending up on a given page [5]. This concept can be extended to Morpheus, where each link can be treated as having equal weight, but the probability of ending up at a given node is based on the number of incoming links to the node. However, since links in Morpheus are

symmetric, one can make the simplification that the probability of transitioning from one cluster to another is the fraction of links going to the second node out of all links outgoing from the first. This probabilistic model allows for a simple and intuitive metric for determining similarity between clusters and results. Putting this metric together with the hierarchical clustering techniques mentioned above results in effective clustering techniques for Morpheus.

4.3 – Visualization and User Interface Techniques

Many visualization techniques have been evaluated in the past. Of the more effective methods, many provided inspiration for the design of the Morpheus system. Included in these tools are the model view controller [10], center stage [62], and direct manipulation [32] interface design paradigms as well as techniques to display visual cues to a user of Morpheus.

Many effective design paradigms have been created for visualization systems. However, most of these patterns are not relevant to Morpheus. For instance, Morpheus intends on communicating a lot of results to its user. This information is at the center of the user's interest, and if it is not easily visible or accessible, the Morpheus system is of little use to the user since it would be more effective for the user to use Google to find a web form at that point.

The center stage design pattern is perhaps the most relevant to this goal of making the results the center of a user's attention. Coined by Jenifer Tidwell, the Center Stage pattern is a paradigm where the most important part of the UI is put into the largest subsection of the interface while secondary tools are clustered around it in smaller panels [62]. This is shown to be effective when the primary goal of the interface is to present information to the user because it guides the user's attention immediately to what is most relevant, quickly establishes the purpose of the UI, and creates a visual hierarchy in the interface with the center stage sitting at a higher level than the toolbars, menus, and forms, which are secondary [62]. This concept is reflected in Morpheus with the visual results browser taking the

majority of the space on screen with the search panel, textual results panel, and toolbar taking up space along the perimeter, effectively focusing the user on the results in the visualization.

Another design principle utilized by many well-designed interfaces is the model view controller paradigm. In this concept, all data elements, event and interaction handlers, and visual elements are decoupled, allowing the interface to be flexible and extensible [10]. Morpheus uses a slightly altered form of this design, with the controller tightly integrated with the view, as it is both difficult and often inefficient to separate the two given modern graphics packages. However, since independence is largely preserved, this gives Morpheus the flexibility to potentially be accessed through a variety of interfaces. For now, however, only one interface exists.

This interface follows a direct manipulation design pattern. This model requires that there be continuous visual representation, physical actions to manipulate the interface, as well as immediately visible and reversible results to the manipulation [32]. These interfaces have been shown to be generally effective because of the consistency achieved with the user being able to interact with items as if they were in the real-world as well as the minimal cognitive effort required to operate the interface. Morpheus follows a direct manipulation interface as the shapes within the graph view can be interacted with as if they were physical objects.

Finally, visual cues have been shown to be effective in conveying information and are, therefore utilized by Morpheus. For instance, color plays an important role in conveying information in Morpheus. Color has been shown to be an effective way to act as a signpost, or a clue to the user about where he is in the visualization or what he is looking at [28]. Since colors work visually instead of verbally, while they may not be noticed immediately, once the user is cued into the colors, he can use them effectively as they draw clear information boundaries and make it easy to map out information [28, 62]. Additionally, as size and proximity have been shown to convey important clues about the relationships

among objects, these cues are also used within Morpheus as heuristics to determine what information is most important [57].

5 – Visualization System

5.1 – System Goals

The goal of the Morpheus visualization system is to offer a usable interface to quickly and efficiently access information contained in the deep web. As the system attempts to aggregate a wealth of information, the system must also be able to present this information in a manageable form. The primary goal of the visualization system is to create the most usable interface for Morpheus. The two factors that Morpheus focuses on in improving usability are manageability and efficiency. For instance, Morpheus, provides tools to allow the user to separate and parse through large sets of data to make the task more manageable, while providing other tools to allow the user to locate results of interest efficiently.

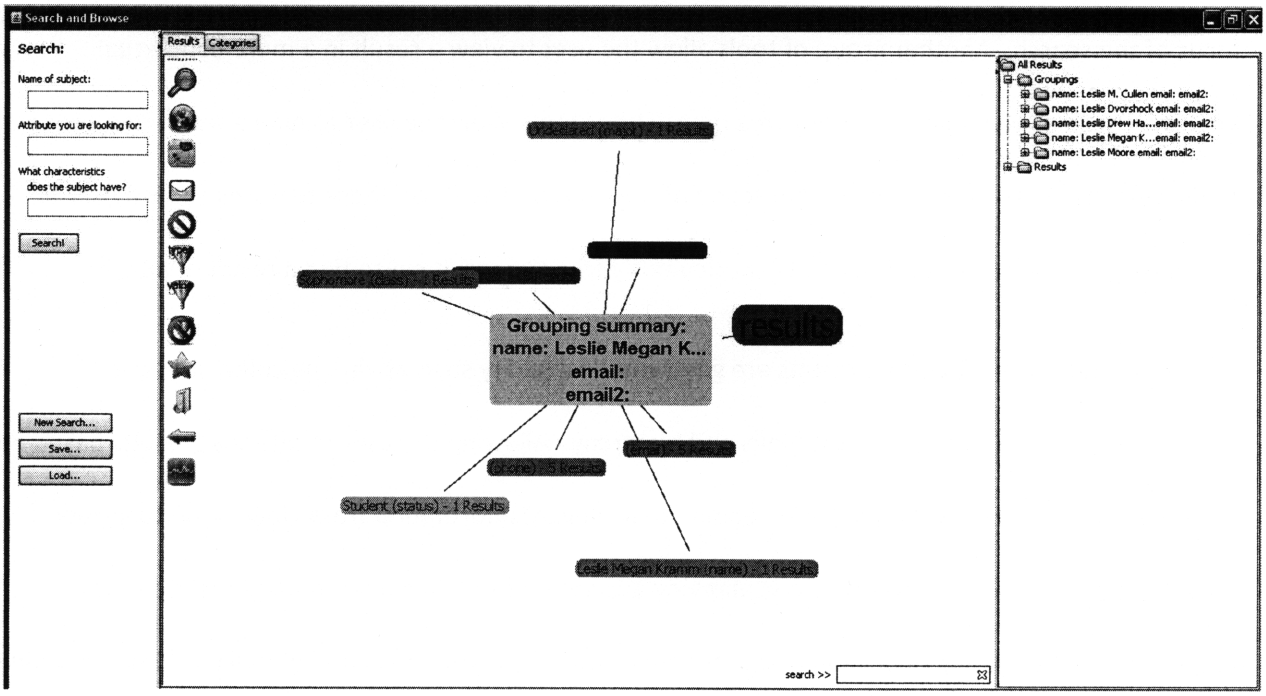

Figure 1 – The Morpheus visualization system.

Figure 1 shows the interface of the Morpheus visualization system. The search panel is located on the left, the text view is on the right, and the direct-manipulation graph view interface is located in

the center. The toolbar on the left-hand side offers users tools to manipulate the graphical view. Size, color, and dimension convey information about relevance to the user. Once a search is performed through the search panel, the graph of results is then displayed in the center panel. At the same time, the text view section within the right-most panel is populated with text descriptions of the results and clusters. Clicking on one of the buttons on the toolbar or a right-click menu option either affects the way the graph of results is drawn or it pops up a modal dialog box, such as those described in the bookmarks, filtering, ratings, and charting sections of this paper. Similarly, modal dialogs are shown when clicking the "Save" or "Load" buttons on the search panel, asking the user for the name of a query.

The visualization system provides the user a range of tools, described in detail in section 5.3 and 5.4. In providing manageability, the visualization system makes several tools available. For instance, results are grouped in an effort to create a small hierarchy, allowing one to systematically narrow the focus of the search. Another group of tools allow the user to view a result in a more contextual interface, such as viewing an address on a map. Other tools allow the user to hide a result or a set of similar results from view to narrow the scope of the search. Together, these features allow a user to eliminate irrelevant information and to meticulously trim the result set to find a result of interest.

To help with efficiency, results are given dimensionality such as varying color, size, and positioning, in addition to their groupings and hierarchy. Allowing the user to filter out results based on their type, value or rating, helps quickly eliminate irrelevant information from view. A textual view provides a duality to the graphical view, allowing a user to jump to a corresponding result in the graphical view by clicking on a label within the text view. Finally, searching and bookmarking tools allow help with efficiently tracking items of interest.

5.2 - System Design and Overview

Like the separation between the visualization system and Morpheus, a separation also exists within the visualization system. The data model which provides the basis for the result processing and analysis is separated from the portion which provides the graphical interface to the user, called the "view." Thus, queries are created by the user within the "view" before being passed to the Morpheus database. Once the results are obtained, the data model takes over, first taking the raw results and putting all the relevant information into objects representing each result and transform. Next, the data model clusters these results together, grouping similar results. Finally, the data model generates a graph structure and a tree structure and hands the information off to the view.

The view, which is comprised of the graphical and text-based displays, is based on the open source Prefuse visualization toolkit [29]. Thus, using this toolkit, the view creates an animated display of the graph structure that has been created by the data model. The view also uses the tree structure to create a textual display for the user. These two displays are integrated with a category hierarchy in a center stage design pattern, with the graphical and category displays providing a direct manipulation interface for the user to interact with. Moreover, all three displays are linked, with actions performed in one display having an effect on the others. These displays sit in a window along with a search panel that allows the user to create new queries as well as save and load existing queries.

5.3 - Data Model

5.3.1 - Graph Model

Results returned by queries to Morpheus are represented by a graph of interconnected shapes and edges. At the center of the graph is an anchor node which represents the search query, or the piece of information known to link all results. At the next level are nodes represented by clusters, each of which is linked to the anchor. These clusters are generated by examining similarities among results in an

effort to group related results. Linked to each cluster node are the nodes representing the results which make up the cluster. Results are linked together by examining their similarities or by the fact that they were produced by the same wrapper or web result. Finally, linked to each result are nodes representing the wrappers that produced the results. Thus, this graph model gives a hierarchy to the results obtained from Morpheus queries. Every cluster is comprised of a group of results, every result has a wrapper as its source, and every wrapper has its properties.

5.3.2 - Wrappers

The wrappers within the Morpheus system each produce a set of results to a given query. These wrappers act as sources of the data and provide results in response to a user query. While each wrapper is represented as its own node within the graph, these nodes are liked to their respective results, indicating that a given wrapper is the source of the information of interest. Also in the graph are nodes representing properties of each wrapper, such as the data types the wrapper outputs, along with other metadata. These descriptive nodes are also connected to the wrapper nodes.

5.3.3 - Results

Results obtained from wrappers are represented as nodes in the graph. Identical results that may be found by multiple sources are collapsed into a single node. These result nodes form the basis of the data model as all operations and analysis are based on the result, their frequencies of occurrence, and their relationships among each other. In addition to keeping track of these relationships, each result node also keeps track of its data type.

5.3.4 – Clusters

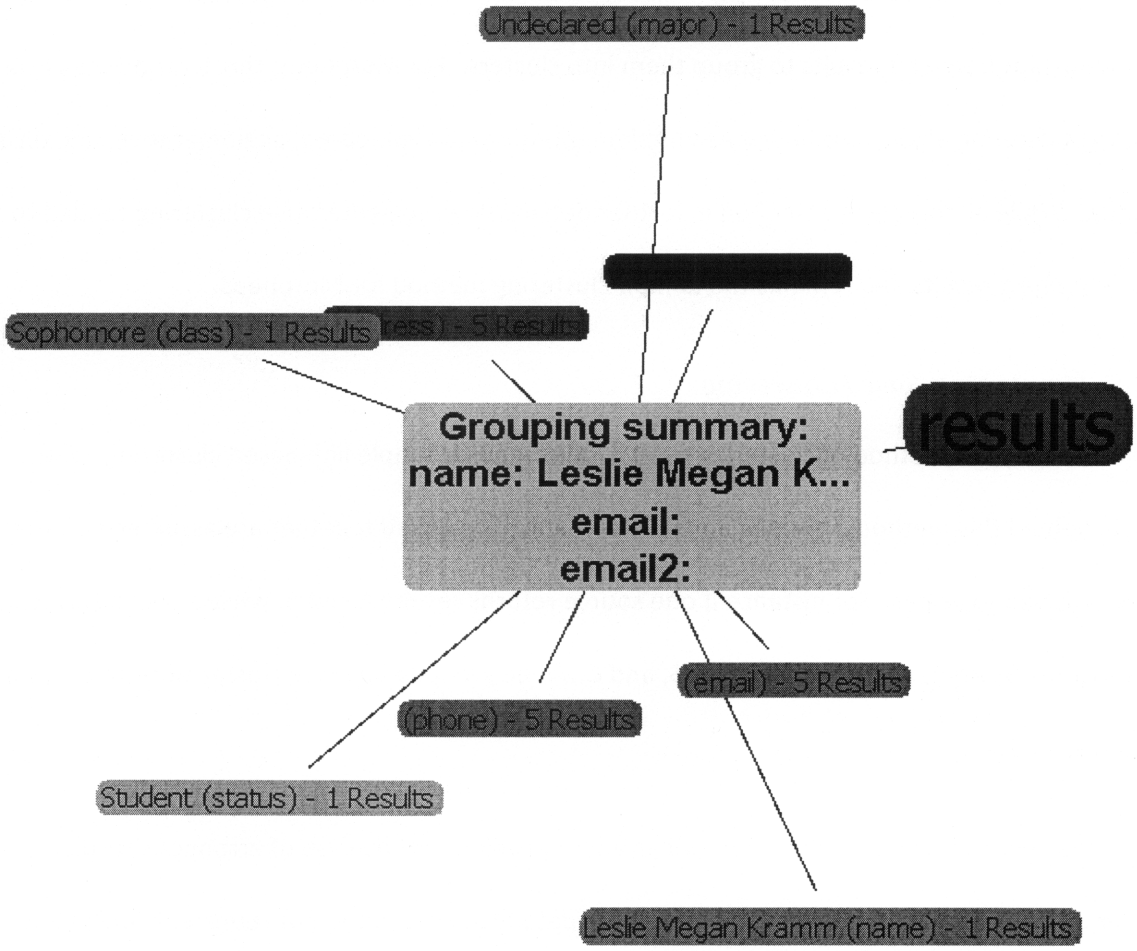

Figure 2 – A Cluster of results

Clusters of results form the basis of the visual representation to the user. The clustering system allows similar or related results to be grouped together. While a cluster is represented largely by a collection of graph links and result nodes in the data model, the availability of such a hierarchy and assimilation allows for greater efficiency when searching for a particular result. For instance, Figure 2 displays a group of results that have been clustered together under a single "cluster" node. The node at the center of the image represents the cluster, while the surrounding nodes represent the results which make up the cluster.

5.3.4.1 - Clustering Algorithms

The Morpheus system relies upon link-based clustering algorithms. These algorithms analyze the relationships between results to group them into clusters. For Morpheus, three variations of such link-based clustering algorithms have been implemented: simple link-based, agglomerative, and divisive clustering. While all three are described in further detail below, agglomerative clustering tended to produce the best results and remains the default clustering method for Morpheus.

5.3.4.2 - Simple Link-Based Clustering

The simplest method of clustering used by Morpheus is simple link-based clustering. The principle behind this method is simple: any results that can be linked together are assumed to be related and are grouped together. For instance if one source returns results A and B, while a second source returns results B and C, all three results, A, B, and C will be grouped into one cluster since they are linked through result B.

However, this clustering method brings about problems in the case of erroneous results. Consider in the example that result C is of the same data type as result A, but represents outdated information. Since the process does not discriminate the results, the two results are treated equally. While this method is more valuable than not having any clustering, this approach is overly simplistic in nature and does not account for the imperfect nature of the results of a Morpheus query.

5.3.4.3 - Agglomerative Clustering

Agglomerative clustering follows the basic principles of the simple link-based approach, but adds tolerances for imperfect data. The agglomerative method starts by assuming that all results are unrelated and places each into its own cluster. It then takes several passes through the results in the graph and groups similar nodes together. However, instead of simply grouping any linked results, the algorithm looks for results that have a strong probability of being related. This probability of being

related is determined based on the strength of a link calculated based on the percentage of links two clusters share out of the total number of outgoing links in a cluster. In other words, the likelihood of two clusters being related is equal to the probability of moving from one to the other if a random outgoing link were to be traversed in a random walk. Once it is determined that two clusters are similar, these two clusters are merged into one larger cluster. The process continues until it is determined that all existing clusters are not linked strongly enough to be merged.

5.3.4.4 - Divisive Clustering

The divisive clustering idea is similar to the agglomerative clustering, but this algorithm essentially does the process in reverse. To start, all results are considered to be related and are grouped into a single cluster. Next, clusters are split into two, with each of the new clusters containing results which are deemed to be similar to the others in that new cluster. Similarity is once again determined by the likelihood of ending up at a node when randomly selecting an edge of the graph to traverse from the starting node.

5.3.4.5 - Comparison of Clustering Algorithms

While it is clear that the simple link-based clustering algorithm is overly simplistic in its nature, it seems that divisive clustering tends to be a little more problematic than agglomerative clustering. The process of dividing a cluster into two groups is complicated. While based on the same principles as agglomerative clustering, borderline results may be erroneously grouped, and there may be no way to re-combine the results together. While the same may be said about the ability of the agglomerative clustering algorithm to separate two improperly grouped results once the grouping has occurred, the separation is more of a significant step because of the tools available to the user in the view, particularly the facilities to eliminate a node from the view. In other words, with agglomerative clustering, a false positive is a problem, while with divisive clustering, a false negative is problematic, and in this case, it is

easier to account for false positives since most of the information should not be placed in the same cluster. This difference represents the primary reason why the agglomerative clustering method yields better results than the divisive clustering process, as mistakes are less costly in the agglomerative method.

Based on the lessons learned from the clustering algorithms that have been implemented, one may argue that there must be a better way to cluster results, perhaps an algorithm that uses both agglomerative and divisive techniques to reduce errors. While such a combination would likely reduce the error rate, the errors made by the agglomerative method have minimal consequence and are tolerable. Moreover, the additional computational burden imposed by the hybrid method would also inhibit the performance of the Morpheus system.

5.3.5 – Bookmarks

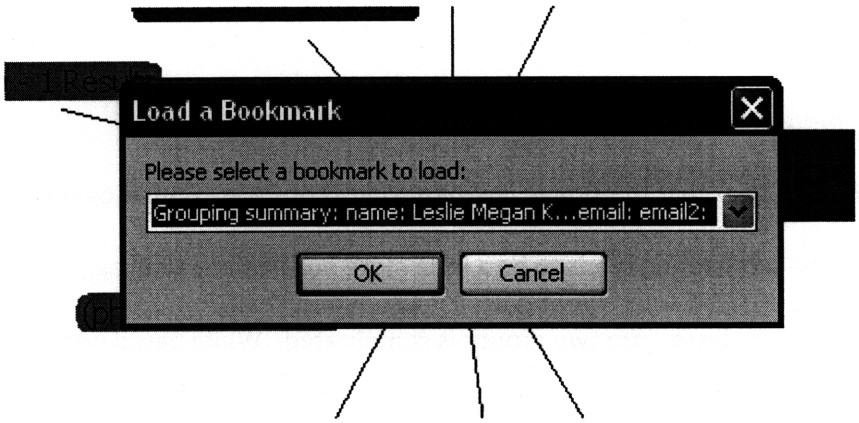

Figure 3 – Loading a bookmark.

The visualization system allows users to bookmark nodes. Figure 3 shows Morpheus' interface for allowing users to load previously bookmarked graph states and nodes which is shown after the user activates the bookmark function through either the toolbar or right-click menu. These bookmarks are stored in the data model and are represented by the name or value of a node. Along with these values,

a hash table is created that maps the value of the node to the Prefuse object that represents the bookmarked node in the graph[29]. These pairs of stored data allow the system to display a meaningful list of values to the user when requested while allowing the system to quickly load a new visual state without delay.

5.3.6 – Ratings

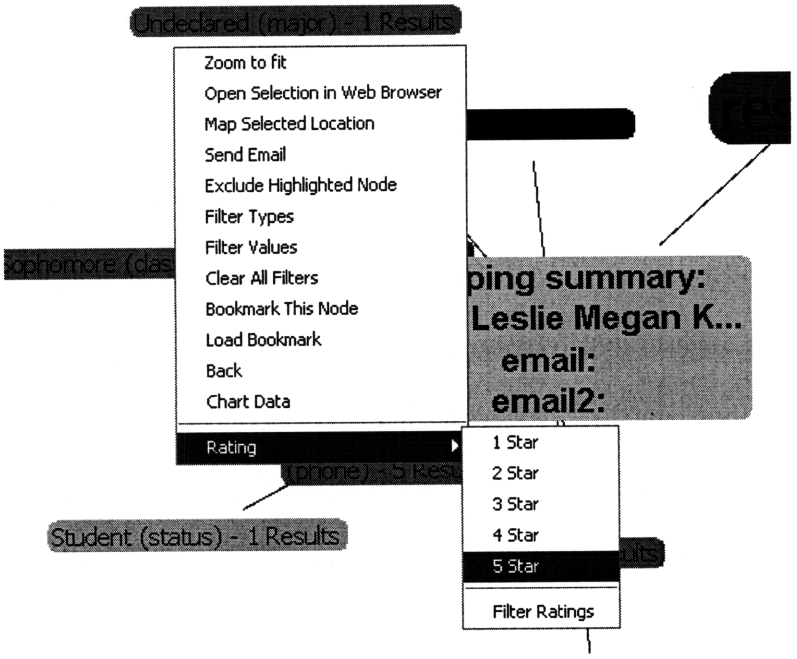

Figure 4 – The ratings interface.

The Morpheus visualization allows users to store ratings of nodes. Figure 4 shows the interface for rating a node. Nodes can be given a numerical rating on a scale from one to five, with five being the best. Thus, once a user rates the quality of a node by selecting the node's rating from the right-click context menu, that rating is stored in the data model and can be used as a criteria for filtering out nodes from the view. These node ratings are stored in a table that maps a nodes value to its rating. Later, this information can be used to filter out poorly-rated results from view.

5.3.7 – Filtering

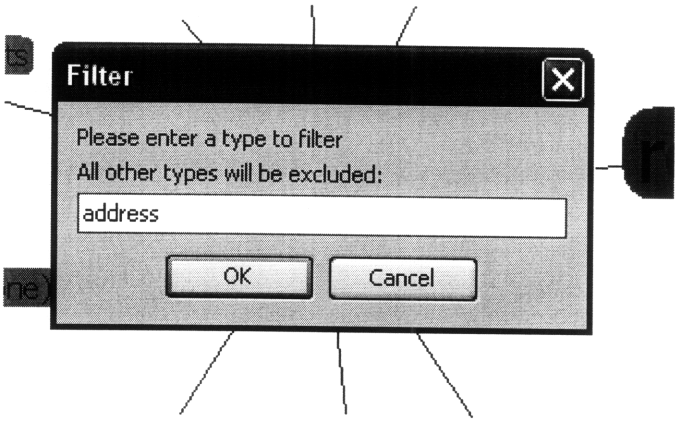

Figure 5 – The filtering interface.

The Morpheus visualization system allows for filtering of data as well. Information can be filtered by data type, as illustrated in Figure 5, by value, and by rating. Once a filter is received, the filter is propagated down the hierarchy, first by being passed to all the clusters, then by having each cluster propagate the filter down to the results. However, when a filter is created, data is not discarded, but rather copied to a new variable. The advantages to this copying approach are twofold: first, the user is able to undo a filtering operation very quickly without performing a new search; second, as opposed to simply hiding the data from view, copying the data allows the clustering algorithms to be run on the filtered data and to produce more meaningful output by accounting for all of the available information.

5.3.8 – Saving and Loading Queries

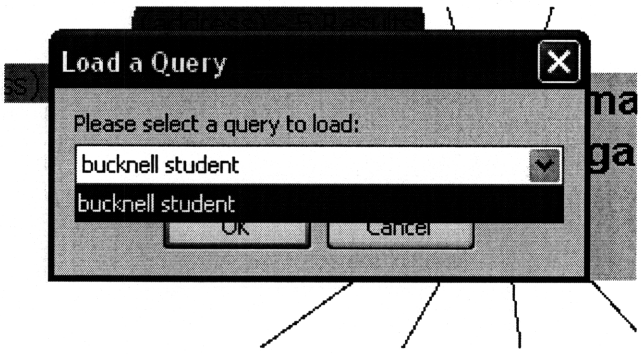

Figure 6 – The interface for loading a query.

The Morpheus data model allows for the saving and loading of queries. Queries can be saved in the Morpheus database, and upon request, these queries are fetched, allowing the user to load a previously-saved query. Figure 6 shows an example of loading a saved query. Queries that are saved during the current session are also stored in a hash table within the visualization system, reducing the load on the database during a request to load queries.

5.4 – View

The user interface of the visualization system is comprised of two main elements. The first is a search panel which allows a user to perform tasks related to queries. The second is the center stage which makes a direct manipulation interface available to the user through which the user can view results as well as narrow a search by manipulating the category hierarchy. These elements which the user is able to see graphically and to interact with are referred to as the "view."

5.4.1 – Search Panel

The search panel represents the area a user will first focus on when running the Morpheus visualization system. This search panel has two main functions. First, the panel allows a user to enter a query to search for information through the Morpheus system. Second, the panel allows a user to

perform operations on these queries. The search interface reduces complexity by containing a minimal set of objects, such as a few buttons and a few text fields. Each of these functions are localized within the search panel, allowing the user to focus on one task at a time.

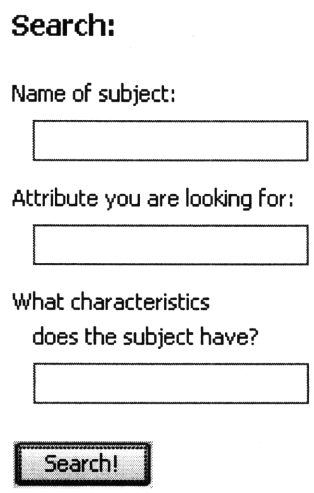

Figure 7 - The interface for entering a search query.

The search area is intended to be quickly learnable as it will most likely represent the first point of interaction for a user. As shown in Figure 7, the user is asked to provide three pieces of information to Morpheus: the name of the subject of the query, the attribute of interest for that subject, and some characteristic of that subject which could help narrow the search field. In early iterations of the visualization system, users were required to provide these pieces in one area through an encoding. While we found this system efficient for advanced users, we found the setup had low learnability and it was difficult to teach a user how to perform a search. The next iteration separated these three pieces into different fields, each labeled with a noun describing what information belonged in each field. However, we found that this was also not very learnable. The final iteration settled on three separate fields, but provided meaningful and descriptive labels to accommodate new users. These fields are labeled in plain English, with tags such as "What characteristics does the subject have?" to avoid

confusing new users with unnecessary jargon. In testing, we discovered that users were able to quickly learn this setup and to quickly perform searches during their first interactions with the interface, without any help.

The search panel also contains a section to allow a user to perform operations on these queries. A user can, for instance, clear a query by clicking on the "New Search" button. Additionally, a user can save a query by clicking on the "Save" button which will save the values currently in the search fields to the database. Finally, the user can also load previously-saved queries by clicking a button as well.

5.4.2 – Text View

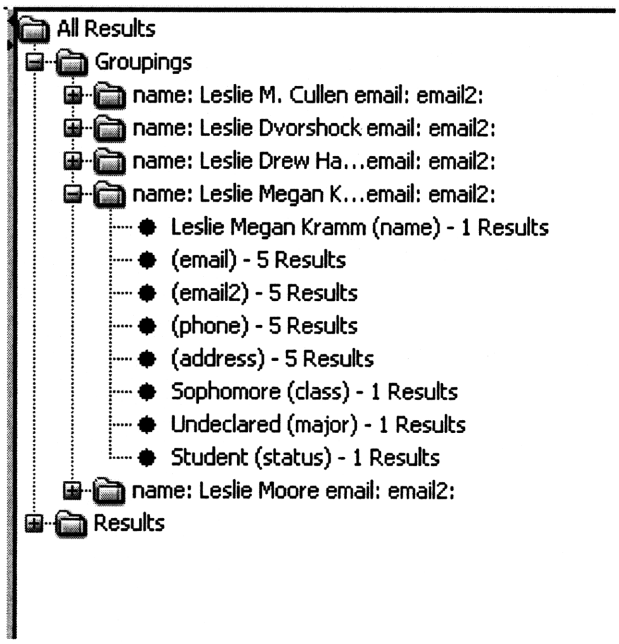

Figure 8 - The text view interface.

The simplest component of the center stage area is the text view, shown in Figure 8. This view provides a contrast to the graphical browser, which gives the user many visual cues. This display takes advantage of the hierarchy created in the graph model that exists once results are grouped into clusters. The text view therefore takes the summaries generated by the clusters as well as the values and frequencies of the results to generate a tree structure. In this tree structure, users can browse results either by first narrowing down results by cluster, then examining the results within each cluster or by

simply viewing a list of all the results. Expanding a node in the tree allows the user to move down the hierarchy, from clusters to results, for instance. In many cases, these textual listings of results and clusters can provide an efficient way for users to narrow down search results and find meaningful information. This view can also give the user a way to view all results in a single display, which is an overwhelming task when using the graphical interface.

However, the most powerful aspect of the text view is the duality it provides with the graph view. Upon clicking on a result in the textual tree structure, the graph view is immediately changed to put the selected result at the focus of the user's attention. Thus, the text view not only provides an alternative to using the graph view, but it acts as an effective supplement to the graph view, as actions taken in this mode are reflected the graph view.

5.4.3 – Category Hierarchy

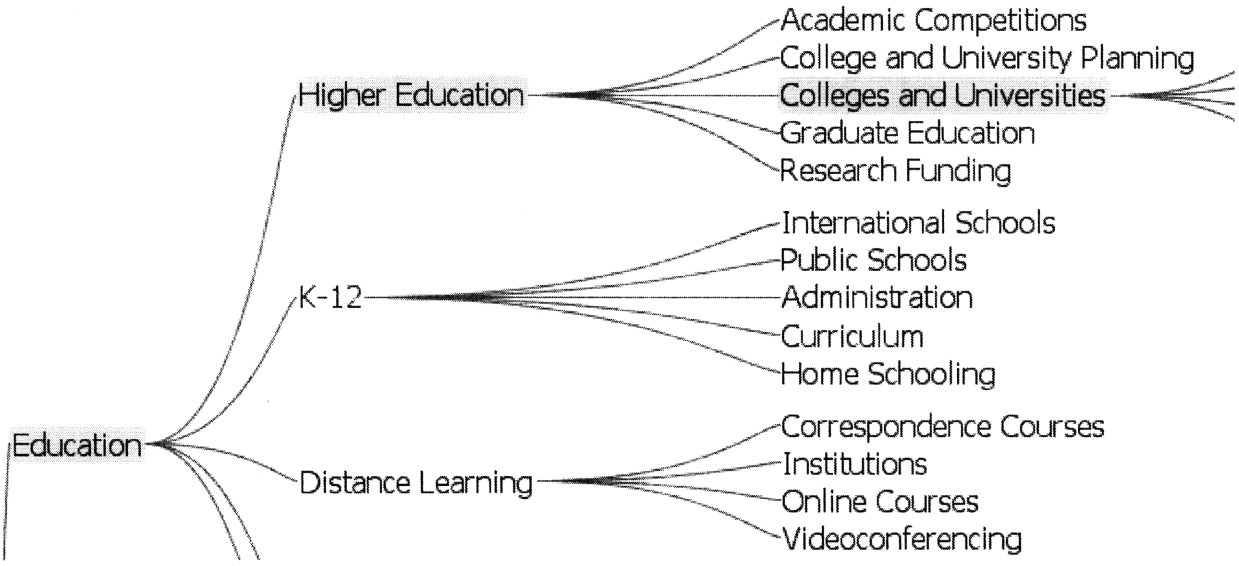

Figure 9 – A category hierarchy.

The category hierarchy is available to the user through a tabbed interface within the center stage. Each wrapper in the Morpheus system is categorized within this hierarchy upon its creation. The user is then allowed to browse this category hierarchy through a direct manipulation interface. The

hierarchy is once again represented as a tree, but the category tree dynamically displays additional details to the user based on where the user is browsing within the tree. Figure 9 shows an example of a category hierarchy. In this example, the wrapper is categorized under the "Colleges and Universities" sub-category within the larger "Education" and "Higher Education" categories. Once a search is performed, the depth of the queried wrappers in the tree is highlighted. Using this hierarchy, a user can choose to either widen or narrow the scope of their query. This direct manipulation interface along with the intuitive hierarchical tree structure provides a simple and intuitive way for users to fine-tune their search results by changing the scope of the search through this category hierarchy.

5.4.4 - Graph View

The graph view is the focal point of the center stage area for the user. This graph view is once again a direct manipulation interface. Here, the user is able to navigate a graph by clicking on nodes shown on the screen and dragging the objects around the display. The graph generated from the data mode is displayed here. This graph is laid out using an algorithm where each node is modeled to have a uniform mass, each edge has its own length and spring constant, and the layout and is determined based on how real-world springs and masses with these properties would interact. To aid visibility, only nodes adjacent to the current node in focus are visible to the user at any time. Clicking on one of these adjacent nodes now shifts the focus to this new node, exposing its neighbors. Moreover, this focus node automatically moves to the center of the user's field of view, to allow the user to easily focus on the current view. Figure 10 shows an example of results shown in the graph view. Here, all results have been grouped into five clusters of results.

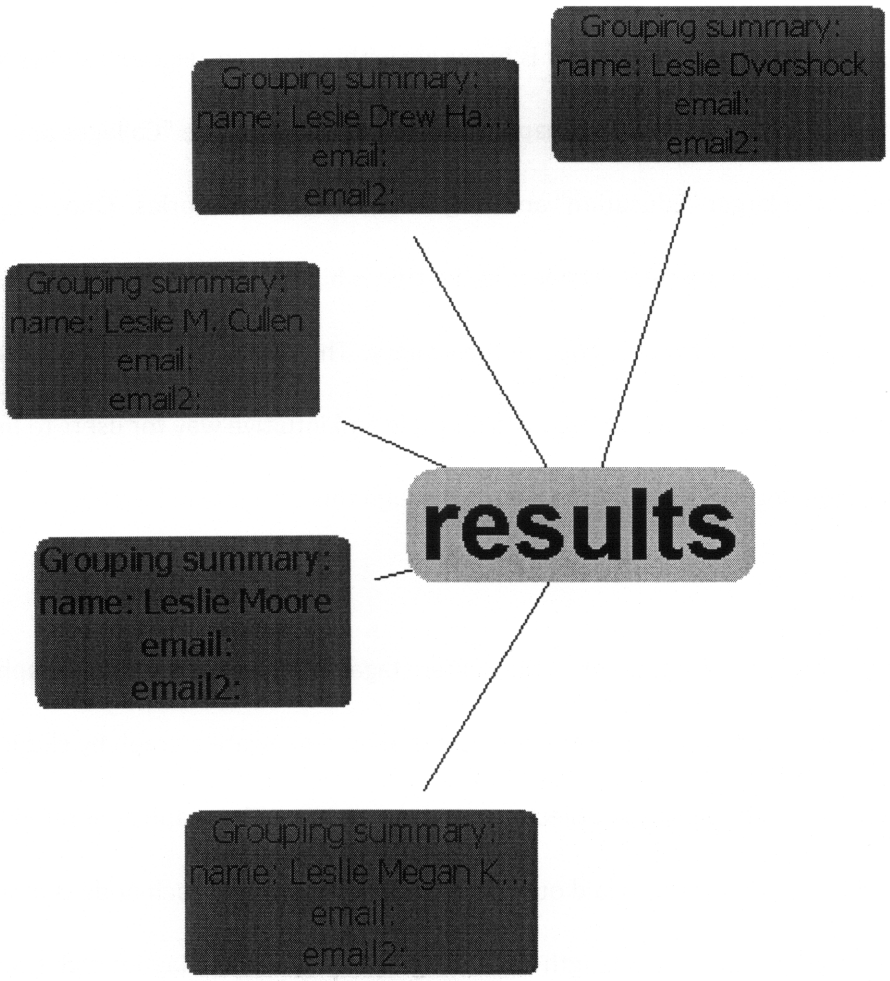

Figure 10 – Results in the graph view.

Several tools also aid users in searching for data. Size, color, and distance are visual cues that help the user identify what information is most relevant. Other features such as charting and zooming tools, which are described in detail later, help the user narrow the field of results and locate the most appropriate result. Additionally, to provide more room for the visualization, all panel dividers on the screen can be collapsed with a single click to create maximal space for the graph view.

5.4.4.1 – Menus and Functions

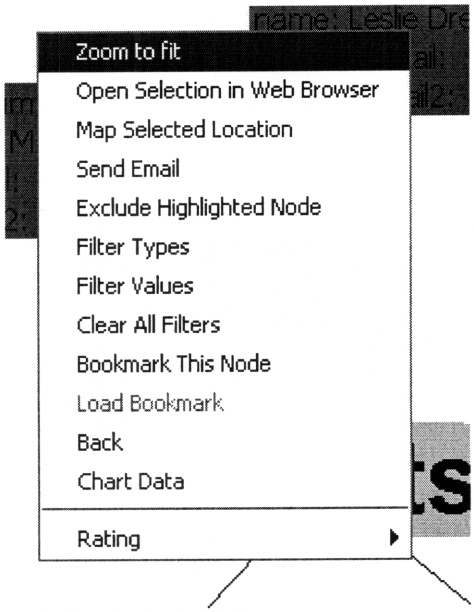

Figure 11 –Right-click context menu.

Important functions are made available to the user either through a right-click context menu, as shown in Figure 11, or through a toolbar located on the display. While the toolbar contains largely the same functionality as the context menu, it does so in a very accessible and highly visible manner, using informative but simple icons to convey the meaning of the function, as shown in Figure 1. These functions include the filtering functions as well as the operations discussed below.

5.4.4.2 – Dimensions

Size

Figure 12 – Use of size in the graph view.

Size acts as an important dimension in the visualization system. With the intent to emphasize more relevant information, this size dimension is primarily reflected in the sizing of the clusters in the graph. The example in Figure 12 shows the use of size to indicate relevance. The cluster at the bottom of the image is drawn with a larger size to indicate it contains more information or higher quality information. Each cluster varies in size, varying linearly with the number or results within the cluster as well as the number of transforms producing those results. Thus, the size of the node representing a cluster aptly reflects the amount of information available within the cluster. However, while size is a

good indicator of the availability of content within a cluster, it was discovered during user testing that nodes can easily grow too large by taking up all of the screen space or becoming unreadable. Therefore, once the cluster sizes are determined, they are normalized within a usable range so nodes are not drawn so large that they no longer fit within the display.

Distance

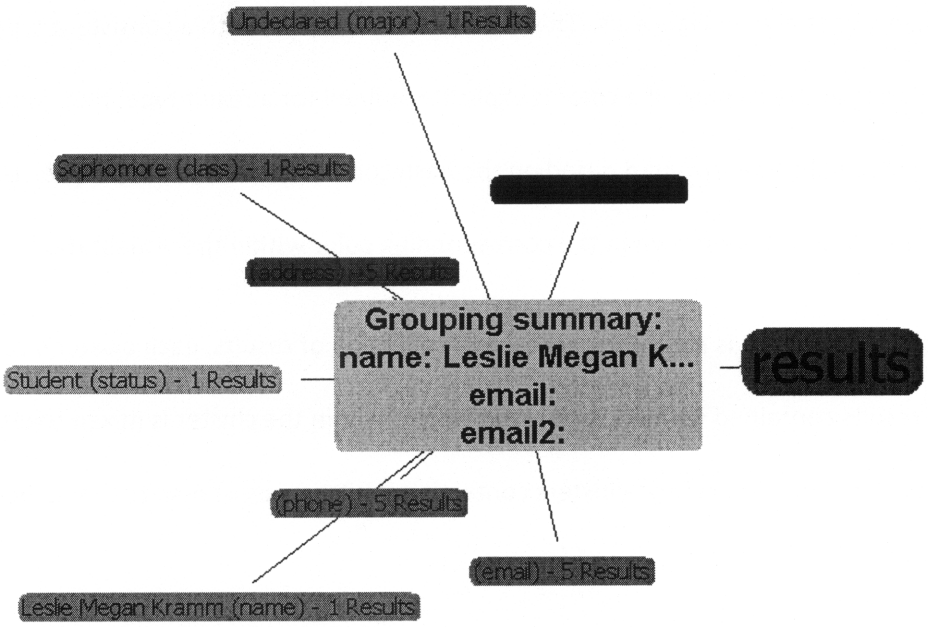

Figure 13 – Use of color and distance in the graph view.

Distance is another dimension used as an indicator of relevance for the user. Edge lengths in the visual representation of the graph often have meaning. For instance, cluster nodes have varying distance from center anchor node (labeled "results" in Figure 13) depending on the cluster's contents. As with cluster size, the number of results and wrappers used are used as indicators for the relevance of a cluster. Clusters containing more information are drawn with shorter edges connecting the nodes to the anchor node in the center, as illustrated in Figure 13. Additionally, edges between clusters and their results are adjusted as well. This edge length is an inverse function of the number of times the result's value was found by all wrappers during the search. Thus, since the node in focus is always centered in

the display, the more relevant information in the graph view will be displayed closer to the center of the screen and to the node in focus.

Color

The Morpheus visualization system also utilizes a color mapping system to give the user visual cues about the types of results that have been returned. To start, each result is mapped to a color based on its type, as shown in Figure 13. This is done so that results with a consistent type all have a consistent coloring in the display. If a color is explicitly defined for a result type, that color will be used. Otherwise, a color will be determined based on the hash code of the result's data type. Using these mappings, all result nodes are drawn in the corresponding color within the visualization.

Next, since a cluster is meant to represent a collection of results, each cluster's color reflects the colors of the results contained within. Each result's color within the cluster is mixed together to produce a cluster's color. Therefore, clusters containing similar types of results should be drawn with similar colors.

This coloring based on the types of data will allow the user to distinguish clusters of different types. To see where this would be useful take, for instance, the scenario where a user is searching for a restaurant named Paris. Paris not only represents the name of a famous city in France, but also the name of a large resort in Las Vegas, NV, each of which contains restaurants. Assuming the city of Paris has different properties than the resort in Las Vegas, and that both would have different properties from restaurants named Paris, each of the relevant clusters would have different colors, with all restaurants named Paris having similar colors, and the clusters representing the resort and city having very different colorings.

Finally, all nodes representing wrappers and their properties are assigned colors as well, allowing the user to distinguish the nodes that are part of the Morpheus system from those that are a function of the search query.

5.4.4.3 – Operations

Zooming

The user is also given the tools within the visualization system to zoom into and out of the display to allow for a better view of the graph. Users have a few options available to re-size the display. First, users can make use of the scroll function on a mouse or track pad to zoom in and out of the visualization. Next, the user can hold down the right mouse button and drag the mouse pointer up and down to zoom out and in, respectively. Third, the user can right-click and select the "Zoom to Fit" option to allow the system to select the appropriate zoom level so the user can view all of the data. Finally, the user is able to click on the same option through the toolbar

Exclusion

Perhaps the most fundamental operation available to the user is the ability to exclude a node. If the user has determined that a node does not contain useful information, the user can choose to hide this node from view. Once the operation is selected through the toolbar or the context-menu, the current node will no longer appear in the visualization, reducing the complexity of the graph and allowing the user to focus on results which have better potential for being meaningful to the user.

Browsing History

The visualization system also keeps track of a user's browsing history within the graph, storing the last ten states the user has viewed. Currently, this only means that a 'Back' button and context-menu item are made available to the user as an easy way of backing out of a complex state within the graph. Because of the lack of depth and interconnections within most result graphs, further

manipulations based on browsing history are largely unnecessary, but can easily be implemented as the system tracks the last ten states of the display.

Web Operations

The visualization system provides simple facilities for web operations. For instance, if a result node is in the format of an email address, the user is able to select the function to send an email to the address, which opens user's default email program to send an email. The user is able to perform similar operations on web URLs and addresses, to open the URL in the default browser or to map the location using Google Maps, respectively.

Incremental Search

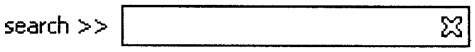

Figure 14 – Incremental search.

An incremental search function is made available on the display. Using this, a user can type in words or part of a word in the field, and any nodes containing that search string will be immediately highlighted in a red color to bring the nodes to the user's attention. This function can be useful when there is a crowded display of results, helping to highlight potentially useful information. Figure 14 shows the incremental search box located at the bottom of the visualization.

Contextual Help

The display also provides contextual help to the user in a two ways. First, every node is clearly labeled with a textual description. Clusters are labeled with a summary, which is generated by looking at the results contained within. These labels help the user understand what each node represents.

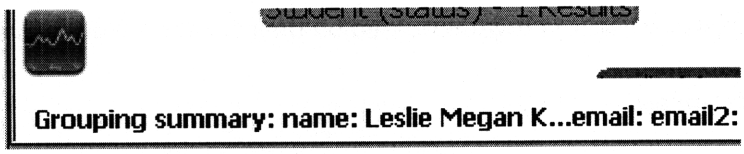

Figure 15 – Contextual help.

In addition to these labels, a status bar, as shown in Figure 15 and located at the bottom of the display, gives some further clues to the type of node and the type of information a node represents in addition to what is contained in the label. These contextual help cues are displayed when a user hovers over a node, providing a convenient, visible, and consistent location for the user to turn to when in need of help.

Charting Data

Figure 16 – Charting tool.

The charting tool, shown in Figure 16, was created to make a more contextual display available for numerical data types. This tool gathers data for every result, grouping them by type. When activated, the charting tool displays an empty chart. At the top of the chart window are two drop down boxes, one for each axis. With the boxes, the user can select one data type to plot along each axis.

Results of different data types are paired together based on their clusters, allowing a creation of a full scatter plot once the axes are set.

6 - Experimental Results

To get a measure of the impact these visualization techniques had on the usability and effectiveness of the interface, we asked several MIT students to perform a set of tasks with the visualization system. These tasks were performed on three different iterations of the system from distinct stages of its development.

6.1 - Methods

Ten users were asked to participate in this study. Nine of these users were undergraduates and one was a graduate student. None had prior knowledge of the workings of RDF triples, deep-web searches, or visualization techniques. Moreover, none of the users had a background in Electrical Engineering and Computer Science and had taken at most an introductory programming class. We felt such users represented our target audience, as the Morpheus visualization system is meant to be a general-purpose system for average individuals.

These users were asked to perform a pre-specified search query and to track down a specific result from the results obtained through the visualization system. In order to judge the effectiveness of the query interface, these tasks were given in plain English. For instance, users were asked, "Find the email address of a sophomore named Leslie, whom you know is a Bucknell student." Users were asked to perform three such tasks, one task for each iteration of the interfaces used in testing. None of the users had ever used Morpheus prior to the study, and users were given no direction, assistance, or training in learning the interface. All users were given the same set of tasks to control for varying difficulty.

The first interface used represents an early-stage iteration of the Morpheus visualization system. While this iteration clustered results and adjusted size and distance of results in the graph, many of the tools and features were not yet implemented. For instance, coloring, the text view, and

filtering tools were not yet implemented. The interface represented the bare-bones RDF visualization system with a simple view on the results. Additionally, the querying interface required users to type in a query using an esoteric encoding.

The first interface evolved into what became the second iteration after most of the available tools and features were implemented, with the text-view and toolbar being the major exceptions. However, while these tools were implemented, these implementations were unrefined. Therefore, while this iteration was clearly superior to the first, it was clear that the visualization featured needed fine-tuning.

These features were refined based on the feedback from testing the second iteration to create the third iteration. Incorporating suggestions from the users about how to improve usability, this third iteration was the result of most of the visualization features, including clustering algorithms, being tweaked for more ease of use.

As usability was a primary goal of the visualization system, to get a more quantitative measure for the improvement in usability through these iterations, users were asked to rate the usability of each interface on a numeric scale from zero to ten, with ten being the highest rating. To control for a user's perceptions, these ratings were taken after the user had tested all three iterations, making sure the ratings were on relatively consistent scale with other iterations.

6.2 - Results

Table 1 shows the results for users' rankings of the usability of the Morpheus visualization system. The raw ratings are shown on the left half of the table while the improvements are shown on the right half. Values represent users' ratings on a scale from 0 - 10 for three successive iterations of the interface.

Table 1 – Usability Ratings.

Iteration	early	late	final	late - early	final - late	final - early
	0	3	6	3	3	6
	1	5	9	4	4	8
	0	5	8	5	3	8
	0	2	7	2	5	7
	0	4	8	4	4	8
	0	3	7	3	4	7
	1	4	8	3	4	7
	1	4	9	3	5	8
	0	4	7	4	3	7
	1	5	8	4	3	7
average	**0.4**	**3.9**	**7.7**	**3.5**	**3.8**	**7.3**
std dev	**0.516**	**0.994**	**0.949**	**0.850**	**0.789**	**0.675**

With an average rating of 0.4, the first iteration was clearly not usable at all. User's rated the interface poorly, with very little variation in the scores. While this represented a disappointment in terms of achieving our usability goals, this result was entirely expected, as very little user feedback had been incorporated at this point. Some comments from the users on this early iteration were, "I don't get it," "What's going on here?" and "I don't understand what I am supposed to do." Based on some of the comments from this interface, we decided on the direction to take in further developing the visualization system.

The second, "late" stage prototype, included most of the features included in the final iteration. While the ratings of the iteration themselves varied from one individual to the next, there was less variation in the measured improvement of the usability of the system. This means that while the absolute ratings were subjective and dependent on a user's knowledge and mindset, the improvement in score was more a function of the improvement of the interface. Also noteworthy is the fact that the ratings universally improved from the initial iteration, showing that these features made a meaningful impact on usability.

In this second stage, much of the feedback varied depending on the knowledge of the user. While scoring for the second iteration was only within a small range, users gave varying reasons to justify these scores. For instance one user focused on the initial learning curve of the system, stating that his major difficulty in usage came at the onset as it took him some poking around to be able to navigate the system. Consequently, this user suggested that better contextual information and alternate views be provided to shorten the learning curve, leading to the creation of the toolbar, contextual help, and text view features. Meanwhile, another user focused on superficial aspects of the interface, suggesting better visibility. These comments contributed to the design of the contextual help feature as well as a refinement of the clustering, sizing, and distance calculations. This variety of comments provided a challenge as it indicated that each user was comfortable with most features, but had major difficulty with just a few areas. Consequently, we set about developing ways to improve usability without bringing about new usability problems with the fixes while accommodating each user's feedback.

Finally, the third and final iteration of the visualization system yielded the highest ratings of the group. These ratings mark a significant improvement over the early iteration as well as the late stage iteration. Moreover, the improvement from the late stage iterations to the final stage was higher than that from the early to the late stage. While there was once again a fair amount of variation in the absolute ratings, the finding that the score improvements showed less variation shows more clearly what effect the evolution of the system had on users. An average improvement from the first to last iteration of 7.3 points out of a possible total improvement of just 9.6 on a scale to 10 illustrates the impact of incorporating user feedback into the design of the system.

6.3 - Reflection

The fact that scores improved by a larger average amount on and a larger percentage from the late stage iteration to the final iteration was very interesting. While the final iteration included a couple new features from the late iteration, the majority of the differences between the two designs were represented by relatively small refinements to existing features. In other words, while the majority of the work involved in designing the system came between the first and second iterations, the majority of the benefit came from addressing feedback on this design. Thus, while each of the features implemented were effective to a certain extent, their designs were initially slightly off-target and with another pass, these features were able to be more effective in addressing users' needs.

While the final ratings represent a significant improvement over the initial iteration, these scores are still far from perfect. The visualization system is hindered by a lack of information, and the fact that it is designed to be a general-purpose system, capable of handling any data types or queries is a significant burden for the system. If the visualization were targeted for a specific type of information, a faceted browser could allow for true tailoring of the interface for the given context.

7 - Conclusion

The task of deep-web integration represents is an area of significant importance as it represents the next logical step in the evolution of search technology. While information on the deep web has grown to account for orders of magnitude more information than the shallow web, little has been done to provide users with increased access to this information. With the rise of web forms and AJAX applications, the need for such search technology is growing every day. In a web where access to the deep-web currently means memorizing web addresses, Morpheus represents a novel approach to solving the deep web problem. While there have been previous attempts to integrate the deep web with the shallow web, Morpheus has a distinct advantage in this area as it preserves the meaning of the deep web pages it wraps.

In order for this deep-web information to be manageable, users need a highly usable interface which makes finding a single result of interest in a field of information an efficient and manageable task. Many RDF result browsers have been created in the past, but many of these approaches are either unsuitable for Morpheus or are not made for the average user. Each of the tools available in the Morpheus visualization system are geared towards these goals of providing usability through efficiency and manageability, and the seamless manner in which they are all integrated together creates a highly learnable and intuitive interface for the user.

In our evaluation of the interface, we saw that this system represents a highly usable interface for the average user. Additionally, we saw that while the stages of the project that involved most of the work in implementing our designs produced improvements in usability, more significant benefits were attained through relatively effortless refinements to the existing features.

However, there remain areas to explore for future growth of the Morpheus visualization system. For instance, faceted approaches can be explored to provide users with an interface tailored to specific

purposes and contexts. Additionally, we have discussed the possibility of extracting components of the visualization system to create a web interface which would be accessible to any user from any computer, potentially furthering the reach of Morpheus. Nonetheless, the Morpheus visualization system is an important piece of the larger Morpheus system, which has the potential to revolutionize the web search industry.

Bibliography

[1] <http://www.chacha.com>.

[2] <http://www.dmoz.org>.

[3] Bergman, Michael. "The Deep Web: Surfacing Hidden Value." 7.1 (2001).

[4] Berners-Lee, Tim, et al. Tabulator: Exploring and Analyzing linked data on the Semantic Web. Cambridge, MA: Massachusetts Institute of Technology, n.d.

[5] Brin, Sergey and Lawrence Page. "The Anatomy of a Large-Scale Hypertextual Web Search Engine." Stanford University (n.d.).

[6] Chang, S. et al. "Context-Aware Wrapping: Synchronized Data Extraction." VLDB Conference (2007).

[7] Chen, T. and K. Chang. "Entity Search Engine: Toward Agile Best-Effort Information Integration over the Web." CIDR Conference (2007).

[8] Cheng, T., X. Yan and K. Chang. "Supporting Entity Search: A Large-Scale Prototype Search System." SIGMOD Conference (2007).

[9] Davidson, S., et al. "K2/Kleisli and GUS: Experiments in Integrated Access to Genomic Data Sources." IBM Systems Journal (2001).

[10] Deacon, John. "Model-View-Controller (MVC) Architecture." 07 12 2008 <http://www.jdl.co.uk/briefings/MVC.pdf>.

[11] DeRose, P. et al. "Building Structured Community Portals: A Top-Down, Compositional, and Incremental Approach." VLDB Conference (2007).

[12] Derose, P. et al. "DBLife: A Community Informaiton Management Platform for the Database Research Community." CIDR Conference (2007).

[13] Dhamankar, R., et al. "IMAP: Discovering Complex Mappings between Database Schemas." SIGMOD Conference (2004).

[14] Doan, A., P. Domingos and A. Halevy. "Reconciling Schemas of Disparate Data Sources: A Machine Learning Approach." SIGMOD Conference (2001).

[15] Dobbins, Pete, et al. "Morpheus 2.0: A Data Transformation Management System." VLDB. 2007.

[16] Dokulil, Jiri and Jana Katreniakova. "Visualization of Large Schemaless RDF Data." International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (2007).

[17] El-Hamdouchi, A and P Willet. "Comparison of Hierarchic Agglomerative Clustering Methods for Document Retrieval." The Computer Journal (1987).

[18] Gannon, T., et al. "Semantic Information Integration in the Large: Adaptability, Extensibiilty, and Scalability of the Context Mediation Approach." (2006).

[19] Gassert, Hannes and Andreas Harth. "From Graph to GUI: Displaying RDF Data from the Web with Arago." (n.d.).

[20] Gemmell, Jim, Aleks Aris and Roger Lueder. "Telling Stories with Mylifebits." IEEE (2005).

[21] Gouttea, Cyril, et al. "On Clustering f-MRI Time Series." Neroimage (2002).

[22] Gowda, Chidananda K. and T. V. Ravi. "Divisive Clustering of Symbolic Objects Using the Concepts of Both Similarity and Dissimilarity." Pattern Recognition (1995).

[23] Goyal, Sunil and Rupert Westenthaler. RDF - Gravity. January 2008 <http://semweb.salzburgresearch.at/apps/rdf-gravity/index.html>.

[24] Guha, Sudipto, Rajeev Rastogi and Kyuseok Shim. "ROCK: A Robust Clustering Algorithm for Categorical Attributes." (n.d.).

[25] Hammer, J., et al. "Integrating and Accessing Heterogeneous Information Sources in TSIMMIS." AAAI Symposium on Information Gathering (1995).

[26] Hayes, Jonathan. "Bipartite Graphs as Intermediate Model for RDF." ISWC. 2004.

[27] He, Bin, et al. "Accessing the deep web." Communications of the ACM 50.5 (2007): 94-101.

[28] Healey, Christopher and James Enns. "Large Datasets at a Glance: Combining Textures and Colors in Scientific Visualization." IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS (1999).

[29] Heer, Jeffrey. Prefuse. <prefuse.org>.

[30] Heim, Philipp. "Graph-Based Visualization of RDF Soccer Data and Interaction Possibilities on a Handheld." (2007).

[31] Hildebrand, Michiel, Jacco van Ossenbruggen and Lynda Hardman. "/facet: A Browser for Heterogeneous Semantic Web Repositories." (n.d.).

[32] Hutchins, Edwin, James Hollan and Donald Norman. "Direct Manipulation Interfaces." Human-Computer Interaction (1985): 50-62.

[33] IBM. "Using the Federated Database Technology of IBM DB2 Information Integrator." IBM White Paper (2003).

[34] Jain, A.K., M.N. Murty and P.J. Flynn. "Data Clustering: A Review."ACM Computing Surveys (1999).

[35] Josifovski, V., et al. "Garlic: A New Flavor of Federated Query Processing for DB2." SIGMOD Conference (2002).

[36] Ketchen, David Jr and Christopher L. Shook. "The Application of Cluster Analysis in Strategic Management Research: An Analysis and Critique." Strategic Management Journal (1996).

[37] Lakshmanan, L. V. S., F. Sadri and I. N. Subramanian. "Schema SQL 0 A Language for Interoperability in Relational Multi-database Systems." VLDB Conference (1996).

[38] Likas, Aristidis, Nikos Vlassis and Jakob Verbeek. "The Global k-means Clustering Algorithm." Pattern Recognition (2003).

[39] Madhavan, J. et al. "Google's Deep Web Crawl." VLDB Conference (2008).

[40] Madhavan, J. et al. "Web Scale data Integration: You Can Afford to Pay as You Go." CIDR Conference (2007).

[41] Mazzocchi, Stefano and Paolo Ciccarese. Welkin. January 2008 <http://simile.mit.edu/welkin/>.

[42] Miles, Alistair. "Representing RDF." (n.d.).

[43] Mutton, Paul and Jennifer Golbeck. "Visualization of Semantic Metadata and Ontologies." Proceedings of the Seventh International Conference on Information Visualization. 2003.

[44] Oren, Eyal, Renaud Delbru and Stefan Decker. "Extending Faceted Navigation for RDF Data." ISWC (2006): 559-571.

[45] Ouwerkerk, Arthur and Heiner Stuckenschmidt. "Visualizing RDF Data for P2P Information Sharing." n.d.

[46] PLS Inc. PLS. 07 12 2008 <http://web.archive.org/web/19971021232057/www.pls.com/news/pr961212_at1.html>.

[47] Quan, Dennis and David Karger. "How to Make a Semantic Web Browser." (n.d.).

[48] Rahm, E. and P. A. Bernstein. "A Survey of Approaches to Automatic Schema Matching." VLDB Journal (2001).

[49] Rutledge, Lloyd, Jacco van Ossenbruggen and Lynda Hardman. "Making RDF Presentable." WWW (2005).

[50] Salvador, Stan and Philip Chan. "Determining the Number of Clusters/Segments in Hierarchical Clustering/Segmentation Algorithms." Florida Institute of Technology (n.d.).

[51] Savaresi, Sergio M., et al. "Cluster Selection in Divisive Clustering Algorithms." (n.d.).

[52] Sayers, Craig. Node-centric RDF Graph Visualization. Palo Alto, 2004.

[53] Schraefel, m.c., Maria Karam and Shendong Zhao. "mSpace: interaction design for user-determined, adaptable domain exploration in hypermedia." <http://beta.mspace.fm/>.

[54] Schreiber, A. Th., et al. "Ontology-Based Photo Annotation." IEEE Intelligent Systems (2001): 66-74.

[55] Shestakov, Denis. "Search Interfaces on the Web: Querying and Characterizing." (2008).

[56] Sheth, A. and J. A. Larson. "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases." ACM COMputing Surveys (1990).

[57] Spoerri, Anselem. "InfoCrystal: A Visual Tool for Information Retrieval." IEEE (1993).

[58] Steer, Damian. BrownSauce RDF Browser. January 2008 <http://brownsauce.sourceforge.net/>.

[59] Stonebraker, Michael. Integrating the Deep Web with the Shallow Web. Cambridge, MA, 2007.

[60] Stuckenschmidt, Heiner. "Exploring Large Document Repositories with RDF Technology: The Dope Project." IEEE Intelligent Systems (2004): 22-28.

[61] Telea, Alexandru, Flavius Frasincar and Geert-Jan Houben. "Visualisation of RDF(S)-based Information." (n.d.).

[62] Tidwell, Jennifer. Designing Interfaces. O'Reilly, 2006.

[63] W3C. IsaViz: A Visual Authoring Tool for RDF. 2001. January 2008 <http://www.w3.org/2001/11/IsaViz/>.

[64] W3C. "Resource Description Framework (RDF)." <www.w3.org/RDF/>.

[65] W3C. Tabulator: Generic data browser. January 2008 <http://www.w3.org/2005/ajar/tab>.

[66] Wang, Yitong and Masaru Kitsuregawa. "Link Based Clustering of Web Search Results ." Lecture Notes in Computer Science (2001).