SUPPORTING DOMAIN SPECIFIC WEB-BASED SEARCH USING HEURISTIC KNOWLEDGE EXTRACTION

A Thesis

by

SUDHARSAN GUNANATHAN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2008

Major Subject: Computer Science

SUPPORTING DOMAIN SPECIFIC WEB-BASED SEARCH USING HEURISTIC KNOWLEDGE EXTRACTION

A Thesis

by

SUDHARSAN GUNANATHAN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Frank Shipman
Committee Members,	Richard Furuta
	Kuang-An Chang
Head of Department,	Valerie Taylor

August 2008

Major Subject: Computer Science

ABSTRACT

Supporting Domain Specific Web-Based Search

Using Heuristic Knowledge Extraction. (August 2008)

Sudharsan Gunanathan, B. En.,

Anna University, India

Chair of Advisory Committee: Dr. Frank Shipman

Modern search engines like Google support domain-independent search over the vast information contained in web documents. However domain-specific information access, such as finding less well-known people, locations, and events are not performed efficiently without users developing sophisticated query strategies. This thesis describes the design and development of an application to support one such domain-specific information activity: for insurance (and related) companies to identify weather and natural disaster damage to better assess when and where personnel will be needed. The approach presented to supporting such activity combines information extraction with an interactive presentation of results. Previous domain specific search engines extract information about papers, people, and course information using rule-based or learning-based techniques. However they use the results of information extraction in a typical query and list of results interface. They fail to address the need for interaction based on the extracted document features. The domain specific web-based search application developed in this project combines information extraction with the interactive display of results to facilitate rapid information location. A heuristic evaluation was performed to determine whether the application met the design goals and to improve the design.

Thus the final application has an unconventional but interactive presentation of the results with the use of tree based display. The application also allows options for user specific results caching and modification of the search and caching process. With a heuristic based search process it extracts information about place, date and damages regarding a specific disaster using a bank of search heuristics developed.

DEDICATION

To my parents S. Gunanathan and G. Murugeswari, sister G. Suganya

And my grandparents S. Kamraj and K. Rajamani

ACKNOWLEDGEMENTS

First I would like to sincerely thank my advisor Dr Shipman for his invaluable guidance for this thesis. I would be forever grateful to him for his constant support and enormous patience during the course of my research. I would also like to thank my other committee members, Dr Richard Furuta and Dr Kuang-An Chang for their guidance.

I am also very grateful to Dr Waran, WARFT for introducing me to the world of research.

I would also like to thank the evaluators for sparing their time to evaluate the application giving valuable comments.

Next I would like to specially thank Nivedita, for her constant support through out my times of hardship and helping me make this thesis possible. I would also like to thank my other friends Sudharsan R, Ram, Sridhar, Arun and Anandh.

Last, but certainly not the least, I would like to express my greatest gratitude to my parents and Lord Vinayaka, without whom nothing would have been possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

A large portion of the world’s networked information is in the form of unstructured natural language content in web pages, news articles, research reports, e-mail, blogs, and variety of other documents. Modern search engines like Google provide domain-independent search in these web related documents effectively and efficiently. In such cases, search results are based on the user’s query strings. Domain-specific searches, such as finding less well-known people, locations, and events are not performed efficiently without users developing sophisticated query strategies. In our context, for example, an insurance company wants to identify weather and natural disaster damage to better predict when and where personnel will be needed. This task is currently done by having an employee browse and search news websites, like www.cnn.com and www.theeagle.com, and news aggregators, like news.google.com.

A well-trained analyst could generate a query to news aggregators or web search engines that identifies specific sources (news agencies) and dates along with the keywords representing their interest. Still, the analyst would be left to determine if the articles returned were duplicates, alternate reports of the same event, or discussions of an event in the past that is unrelated to current personnel needs.

This thesis follows the style and format of *IEEE Transactions on System, Man, and Cybernetics*.

Fielded search, supporting range-based or join-based structured queries, could solve this problem but require domain-specific knowledge representation. While Semantic Web efforts aim to make such search possible, they rely on either information providers/brokers adding that domain-specific metadata or automatic means to extract this information. Between the cost structures involved with human generation of metadata for all articles and the heuristic nature of natural language processing, tools are needed to support domain-specific information access that do not assume high-quality metadata extraction.

This thesis work explores the design of one such tool. The thesis concentrates on domain-based searches for information regarding natural calamities such as hurricanes, storms, blizzards, and earthquakes throughout the United States of America from news articles. This task was motivated by an industrial representative describing their company's current practice and desire for improved tools. In general, such information would be of value to insurance (and other) companies when allocating personnel. For this task, information that is of interest includes the place and date of the occurrence, the damage in terms of deaths, injuries, and cost, and the type of event.

2. APPROACH

The general approach to supporting domain-specific information location explored in this thesis combines aspects of information retrieval, information extraction, and interactive system design. The next subsections provide an overview of information retrieval (IR), information extraction (IE), and natural language processing technologies [1].

2.1 Information Retrieval

Information retrieval [2] deals with searching for information or metadata in documents, or searching within databases, or even searching in the highly unstructured World Wide Web. The best known examples of information retrieval systems are web search engines such as Google. An information retrieval system takes in an input from the user for searching, in the form of queries which describe the information needed. These could vary from formal database access queries as in SQL to highly random text as in web-based search engines. For many IR models, queries do not provide a Boolean assessment of whether each object matches a domain but result in a set of objects that partially match the query. In systems based on such models, the results from the query are determined by ranking the objects using various metrics.

The performance of an information retrieval system is evaluated using measures such as prescription, recall and fall out. Table 1 shows the formula for calculating them.

Table 1

Performance measures of information retrieval

Measure	Formula
Precision	$\frac{\lvert\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}\rvert}{\lvert\{\text{retrieved documents}\}\rvert}$
Recall	$\frac{\lvert\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}\rvert}{\lvert\{\text{relevant documents}\}\rvert}$
Fall-out	$\frac{\lvert\{\text{non-relevant documents}\} \cap \{\text{retrieved documents}\}\rvert}{\lvert\{\text{non-relevant documents}\}\rvert}$

The best search engines are the ones which can maximize the retrieval of relevant documents and minimize the retrieval of non-relevant documents as far as possible [3]. This corresponds to the terms recall and precision. Recall is the percentage of the relevant documents retrieved by the search among the total relevant documents in the search domain. If the total number of relevant documents in the search domain is 50, and if the search retrieved 10 documents from this then the recall is 20%. Outside of a controlled experimental setup, as is found in the TREC competitions, it is not feasible to know the number of total relevant documents. Thus, recall is not often used in practice.

Precision is the percentage of relevant documents among the documents retrieved. If the search retrieves 100 documents and 15 of these are relevant, then the precision is 15%. Fallout is the percentage of irrelevant documents returned by a search. If the total number of irrelevant documents in the search domain is 1000 and the search retrieved 10 of the irrelevant documents then the fallout is 1%. Fallout gives the measure of how badly the system performs by retrieving unrelated documents. It becomes a problem as the size of the input domain increases.

2.2 Information Extraction

While information retrieval [4], in the form of a standard search engine, can be used to locate documents of potential interest, it cannot be used to examine the results based on domain-specific features, such as the location, date, costs, and type of event. Extraction systems are often built to either populate structured domain-specific databases or to enable question answering, where the system answers domain-specific questions posed by the user. Thus information extraction can be defined as a type of information retrieval which is used to automatically extract structured information from unstructured machine readable documents.

2.2.1 Extraction Process

The subtasks of information extraction [5] include segmentation, classification, association, normalization and de-duplication which are discussed below.

1. Segmentation: The first step in the information extraction is the segmentation. It involves identifying the documents which are to be searched and which are not, thus making the search more efficient. For example, consider searching technical papers in the web. Good sources for these papers are conference and journal websites. Segmentation here refers to finding specific web pages with information about papers to be presented at conferences or published in journals. Segmentation is not just locating a web site. In this example, the searching task is reduced by avoiding non-relevant web pages of the conference like keynote speaker pages, submission details pages etc. Similarly, in the case of calamity search project this segmentation is applied to separate the news pertaining to calamities from other news articles. Segmentation also selects the content within each document that is examined by the later stages of information extraction process. In the news domain, the initial paragraph(s) is often all that is needed to extract domain specific features.

2. Classification: The text that is selected through segmentation is then classified by assigning one or more categories to words in the selected text. In the case of a pure grammatical classification, as in natural language processing, this process classifies the words based on their part of speech, e.g. noun, verb, adverb and adjective. For the above example of searching for technical papers, this includes segregation of the extracted information into various fields such as paper title, authors, category etc. This also includes searching inside papers to classify metadata information. The paper’s sections might be classified into abstract, introduction, results, references, etc. In the calamity project, when the document is searched for the event date, the extraction usually returns a set of dates such as published date, related event dates, etc. Classification of these dates into categories such as, event date, published date, or any other date of reference to some other events is performed to extract the actual event date.

3. Association: This determines which set of classified words or text can be associated together to determine relationships. For the publications example described above a variety of associations is done on the classified documents or text. Association is done on the pool of extracted paper title, authors, category to match them to determine complete paper information. Also association is done on different paper titles to group papers according to various categories and subjects. Also related papers such as improvement papers, critique papers can also be grouped using this. With respect to the calamity project, consider the case of determining the cost of a calamity. The cost may be the number of damaged homes, estimated property damage, or the number of deaths or injuries. These are usually defined by quantitative terms inside the same paragraph. Here association is done to attach each quantitative figure to a cost type.

4. Normalization: Once the classification and association is performed the text extracted need to be normalized. This is the process of converting the extracted text to standard formats so that they can be compared to each other efficiently. For example the published papers have authors and references attached to them. Different conferences have different formats for references and authors use different variations of their names. Normalization converts these varied formats into standard format after extraction so that comparisons could be made more efficiently. In the date extraction process of the calamity project, the extracted dates could be in different formats such as “2/2/2008”, “February 2nd 2008” or “2nd Feb 08”. One of the methods to find the date of the article is to determine the most referred to date in the article. For this normalization of the dates is performed to find the similar dates in the article.

5. Deduplication: This step uses the data from the prior steps to remove duplicate information. Continuing the publications example, once the references for the papers are normalized, the same paper is likely to be referred to by different published papers. Deduplication on them might be done so as to remove redundancies, so a single reference to each paper is provided to the user. In the calamity news extraction project the same article is often found on multiple news sites that acquire stories from the same sources (e.g. Associate Press). These are recognized and the duplicates can be hidden or removed. Actually these duplicate search hits can be organized together and displayed as a single unit for user reference and comparisons.

2.2.2 Extraction Algorithms

The information extraction process described above results in a set of entities and relationships. The particular algorithms for classification and association building can be based on either a knowledge engineering approach or a statistical or training-based approach. These approaches are described below.

Knowledge engineering approach: This is also known as grammar or rule-based extraction approach. A person or team of people who have domain expertise and an understanding of the types of unstructured documents that will be encountered writes rules for extracting the desired information. The rules are corpus and domain specific. The iterative process of writing the rules starts with building of rules on a subset of corpus. This is tested against a testing set and modifications are made to the rules based on unexpected behavior.

The performance of the system depends on the skill and experience of the person creating the system's extraction component. The human effort for hand-crafting information extraction rules yields better performance in domain specific systems. However the rule creation, testing, and debugging process can be prohibitively costly and may be unsuccessful due to a lack of required expertise. Additionally, modifications to the system specifications can be difficult to instantiate.

Training-based approach: The training-based approach to information extraction requires a set of unstructured documents and the structured information that should be extracted from them. The system uses a learning algorithm to create a statistical model that matches characteristics of the documents to extracted features. The model is then applied to documents not part of the training set to identify the desired characteristics.

This approach eliminates the task of writing domain-specific extraction rules. Instead, this approach relies on people to generate training data sets, which is a process of interpreting and annotating texts with metadata rather than requiring a person or team with both domain expertise and rule-authoring expertise. For example consider the case of named entity extraction [6] [7]. Here a person with good knowledge about the specific names which are needed is enough for annotating the text. The hurdle to this approach lies in the training data. Sometimes training data may be hard to get or may be costly. Insufficient training data may lead to less efficient extraction systems. Specification changes can lead to re-annotation of different and large amount of new training data.

2.3 Natural Language Processing

Given that this thesis explores the use of information extraction to aid access to domain-specific natural language texts, an approach to information extraction that was initially considered was natural language processing (NLP). NLP is used for understanding, processing and interpreting human language using syntactic and semantic analysis of the input text. NLP involves structuring the input text, identifying the various entities mentioned in the text and relationships among them, and providing a representation of the patterns discovered. The process involves lexical analysis, name recognition, and syntactic analysis.

Lexical analysis involves dividing the input sentences into parts and assigning parts of speech tags and features to those. This is done using dictionary lookup and morphological analysis.

Name recognition involves identification of parts of the text as proper nouns, dates, formats such as currencies of specific names such as company names.

Syntactic analysis identifies meaningful groups of nouns, verbs, etc. This looks for specific patterns in the text in order to extract the desired information.

Early on in the design of this domain-specific information system, two NLP packages were explored for performing information extraction: Minor Third and LingPipe.

Minor Third [8] is an open source package for storing, annotating, and learning to extract entities and categorizing text. It was written in Java and was developed at

Carnegie Mellon University in the Center for Automated Learning and Discovery. Unlike other NLP packages it has combined tools for annotating and visualizing text. It also has methods to visualize training data and the performance of classifiers that are meant to help improve application performance.

LingPipe [9] is a collection of java classes for linguistic analysis of human language. LingPipe can extract named entities; find relations between entities and actions; classify sentences according to language, genre, topic, or sentiment; perform spell checking; group documents by common topic, generate a time line of events; and provide part-of-speech tagging and phrase chunking. LingPipe's architecture enables multi-lingual, multi-domain, multi-genre models; new training for new tasks; statistical confidence estimates; and online training.

Early design and development of the information extraction component of this thesis was built on the above NLP packages. While both Minor Third and LingPipe can be used for information extraction, they are designed to perform a complete analysis of the text. Building effective models for such a thorough analysis of the text is time consuming and difficult. The English language itself is complex, ambiguous and dynamic. Lots of new words are constantly added to the vocabulary. A list of disputes in the English language can be found at [10]. All these features of the language make NLP a difficult task especially when applied on data from the web [11].

Due to the extra overhead associated with full NLP packages for functionality not central to the task of providing domain-specific information access, the final design uses the knowledge engineering approach to information extraction. This enables the focus to be on the portion of news articles where the desired information is often found and the use of HTML markup.

3. RELATED WORK

This section discusses work related to domain-specific information access on the Web including meta-search engines and domain specific search engines.

3.1 Meta Search Engines

Meta search engines are used to locate information from web-based sources but do not search the web on their own. Instead they take a user-defined query and pass variations of it to available search engines based on domain-specific or search-engine-specific characteristics. These engines make decisions like which standard search engine(s) to query for a particular user input query and how to modify the user query before submitting it to the standard search engines. These engines also re-rank the results returned by the various search engines and normalize the presentation of the outputs from the different engines. Meta search engines are particularly important for domain-specific search because of their ability to integrate results from search engines that cover different domain-specific collections (e.g. the ACM digital library and the IEEE digital library). Inquirus 2 and Brainboost are examples of meta-search engines discussed below.

Inquirus 2 is a meta-search engine [12] was developed at NEC research institute. This search engine takes user preferences as input to direct the search process. The user input determines how a query is modified by appending terms, how results are ranked, and how to select the search engines. Hence when using Inquirus 2, the same search queries can yield different results based on the user preferences.

Brainboost [13] is a metasearch engine for answering natural language queries. As with other meta search engines, the user query in Brainboost is used to generate different queries which are submitted to standard search engines. The results returned by the search engines are then scanned using NLP techniques to locate answers for the original natural language query. Finally, the answers are ranked according to the AnswerRank algorithm and displayed to the users according to the rankings. The engine works best on questions that have been previously answered on the web.

In general, the main advantage of meta-search engines over a standard search engine is the increased coverage and automatic integration of results. However, results from meta search engines are more likely to be irrelevant due to their lack of knowledge about the internal functioning of the standard search engines they query. Also, aggregating results from multiple search engines requires re-ranking, which is difficult across search engine results as the meta search engine lacks the information used by the search engines to perform their initial rankings (e.g. inverse document frequency).

3.2 Domain-Specific Search Engines

Domain specific searches limit the collection of materials covered to those in a particular domain and thus have the advantage of increased precision. The ease of development and maintenance depends on the domain for which it is developed. Domain specific search engines have been used to find information about courses in [5]; to find information about conferences, their dead lines and submission guidelines in [14]; to summarize medical records involving tests and diagnosis; and to list job openings from advertisements, company websites, job posting sites. Several domain specific searching applications are discussed below.

FlipDog [15] is a job listing website. Instead of looking for information from newspaper classified ads and other collections of job openings from various companies, it performs information extraction from 6000 company web sites. It extracts the job title, description, location, requirements, and contact information from the company web sites.

Zoominfo [16] extracts information about people, creating profiles which include job references and job history, contact information, and education. The data is collected by mining biographies from companies, news articles, press releases, and personal web pages referred to by company pages. This information is used in sales, recruitment and also in background checks.

Citeseer [17] is used to assist researchers in searching for computer and information science research materials, especially publications. CiteSeer extracts paper titles, authors, references and gives users the ability to navigate from paper to author and from paper to reference. It also gives suggestions by also generating a list of relevant papers using similarity measures. Papers are segmented into blocks and displayed for the users to get the gist out of them instead of straight away downloading them. Users are also provided with options for downloading papers in different formats.

DEADLINER [14] is a domain specific search engine that extracts information from conferences and workshop in the form of speakers, location, topics, deadlines, committees, abstracts, etc. It doesn't use manual text extraction solutions, or natural language processing. It is based on the assumption that the WWW has documents with enough structure, i.e. link information, formatting structure and keyword fields; that target information can be extracted without NLP. It uses Bayesian integration of simple extractors.

The above projects take different collections as inputs, use different search mechanisms, and extract different features depending on the domain. As such, they inform the design of the domain-specific search application. However these projects do not focus on user's interactions with the extracted information and how the extracted information can be used to support user interaction. All of these websites present the result in the normal format of snippet listings. Here the user actions are limited to the input query formulation and choosing between different orderings of the results. The domain specific web-based search application developed in this project combines information extraction with the interactive display of results to facilitate rapid information location.

The following sections describe the design of the system for the calamity search, along with the heuristic evaluation conducted and the improvements made to the system.

4. DESIGN OF A DOMAIN-SPECIFIC SEARCH APPLICATION

The goal of this project is to design a tool to support rapid access to domain-specific information in the domain of weather and other calamities. The users of the system are presumed to be insurance (or other) companies that have to allocate personnel and resources based on the type and size of event. To support these users, the application locates news articles about events of interest, performs domain-specific information extraction on these articles, and presents the results in an interface that allows users to rapidly assess the articles.

The application consists of a backend that does preliminary information selection and processing and a front-end that supports user interaction. The backend locates news articles related to events of interest to insurance companies and performs information extraction. The front end presents the articles located and information extracted, providing users with flexibility and ease of navigation.

4.1 Backend Design

The backend of the system retrieves articles of interest from RSS feeds from news sites. *XML and Website parsers* are used to select components from the web pages that contain the articles that will be the input for the domain-specific information extraction. Information is extracted via rule-based algorithms in a *heuristic search bank* consisting of topic and source-specific approaches for extracting date, place and cost features. A *heuristic matcher* is used to select a heuristic search function for a particular combination of input article, calamity and feature to be extracted. Appendix D lists the pseudo code used for the backend design.

4.1.1 XML and Website Parser

The main source for the extraction is the articles describing calamities from the various online news websites such as CNN.com, CNBC.com, and Foxnews.com. These news websites differ greatly in their structure and organization. So a single search strategy for article location will not work for all these websites. However, most news websites have one thing common; they provide RSS feeds to provide notifications of new articles and updates. These news sites provide multiple RSS feeds where articles are divided into categories such as top stories, US news, world news, and weather, and are in fixed format. RSS feeds generally follow a standard XML format that includes

1. Publication date
2. Link to the main article
3. Description
4. Headline

Using this metadata, news describing calamities are chosen from the universal set of news articles. The description and the title are used to identify whether a news article is related to an event of interest. This is done using a simple string matching technique that looks for domain-specific keywords.

A website parser is then used to parse the news website to extract the article on which information extraction will be performed. The web pages that contain the news articles have considerable unrelated content, such as pointers to recent or related articles, advertisements, general header, footer, and navigation information. The website parser filters out this “noise” from the content of the main article. Two methods are used for locating the article among the extra content:

HTML tag based retrieval: Here the contents of the web page are extracted by using the information in the HTML tags. The tags represent various parts of the webpage and these parts can be segregated using knowledge of the web sites’ common use of tags.

Visual cues Based retrieval [18]: Here the structural representation of the webpage is used to separate it into various parts. These could include change in fonts inside a page or an introduction of a line.

The extracted news articles are passed to the extraction mechanisms for identifying date, place and cost information about the calamity from the articles.

4.1.2 Rule-Based Feature Extraction – Heuristic Search Bank

This project originally began by using NLP packages to extract the domain-specific features desired for the application. It became clear that the overhead of such an approach, both in terms of development time and speed of the application, was unacceptable and not necessary for this particular domain and application design. Instead, the application uses rule-based feature extractors hand-coded for the domain and news sources.

Once articles for information extraction are identified, these are searched for information in terms of date, place and cost of the event. Heuristics are used for extracting this information. A pool of heuristics is developed for each calamity feature. A heuristic matcher is also developed to match the heuristic rule for a news website with respect to a calamity type and the feature being extracted. The various heuristics that are used for the 3 features are described below.

Date

Pattern based date search: The news articles specify dates in some specific formats such as – “16th July 2006”, “02/16/2006”, “16-JUL-06”. So pattern matching here is done in order to recognize such patterns. When these patterns are matched and specific date formats are found the dates are converted into a standard format – Gregorian calendar format. This conversion of different formats of dates into fixed format is an example of normalization discussed under the information extraction topic in Section 2.2. However, there can be many dates mentioned inside a single document. So further processing of the outputs is needed to determine the date which best describes the event. The first technique used is the spatial ranking of the dates collected. Spatial ranking relies on the knowledge that most articles include the date of the article and primary events being described at the beginning of the document, in the first few paragraphs. This information is used to determine the date of the article. Another technique is to take the most mentioned date in the article. While both techniques were instantiated, the application uses the most referred to date as the event date.

Relative references to time: In news articles, dates are often referred to by relative terms, such as “yesterday”, “last Tuesday”, “today”, “last week” etc. By extracting these terms and the publication date of the article, it is possible to determine the dates being mentioned in the text.

Tag and field specified: Some news sites embed date information in hidden fields to pass this information between the pages in the news web site. When the parameters in hidden fields are parsed, the date information could be extracted. This is more likely to be a specification of the publication date than the event date.

Place

Pattern matching: In most news articles the place of occurrence is described in the first paragraph with specific formatting such as a bold font or followed by a hyphen. These formatting conventions are not same for articles on different news websites. By making use of this feature and the pattern in the article, the place field can be extracted. However, use of such mark-up to locate the place information sometimes extracts additional text along with the location. This text often identifies the news agency or source, e.g. “AP”, “Reuters” etc. The result is that pattern matching removes such names from the date string during extraction.

Tag and field specified: As with the date values placed in hidden tag values to pass information from one page to another, such tags occasionally pass location information.

Noun/preposition based extraction: The terms or words which describe a place usually come after prepositions such as “at” and “in”. By extracting terms which are in proximity to these prepositions in the title or in the first paragraph of the article, place information can be extracted. However this technique produces noisy results because of the many other uses of this form of prepositional phrase. Thus, the application does not use the results of this approach directly to extract place information but to strengthen the results found using other techniques when more than one potential location is identified.

Cost

Usually costs are described by terms such as “killed”, “deaths”, “injuries”, “damages”, etc. By looking for such terms in the documents the cost information can be extracted. However sentences describing costs many also include other quantitative information. For example consider the sentence “Two powerful earthquakes ranging 7 to 7.2 on the Richter scale hit 4 Indonesian islands killing 340 people and injuring thousands”. This sentence has 5 quantitative phrases but only two related to costs. A shortest distance mechanism and a phrase-based mechanism were developed.

Shortest distance mechanism: According to this approach the cost descriptor is paired with the quantity which is nearest to the cost in the sentence. The nearness is measured in terms of the number of intermediate words. For example consider the above described sentence: “Two powerful earthquakes ranging 7 to 7.2 on the Richter scale hit 4 Indonesian islands killing 340 people and injuring thousands”. Here the nearest word to the cost key term “killing” is “340” and for the cost key word “injuring” is “thousands”.

Phrase-based mechanism: In this mechanism the boundaries of phrases containing cost-related quantities are used to bind the pairing of quantity and cost terms. For example in the sentence “Two powerful earthquakes ranging 7 to 7.2 on the Richter scale hit 4 Indonesian islands killing 340 people and injuring thousands”. The boundary for killing starts from “two” and ends at “and”. Whereas the boundary for “injuring” starts at “and” and ends at “thousands”.

4.1.3 Heuristics Matcher

The above section describes the heuristics developed to extract domain-specific information from news articles. The heuristics used for extraction depend on the source of the article (the website it comes from) and the feature being extracted. The heuristics matcher is a table which determines which heuristic is to be used for a particular news website for a particular feature.

4.2 Front End Design

The front end of the domain-specific search application is where the user specifies information needs and receives search results. Many domain-specific applications provide faceted search but still display results in the form of snippet lists. Results in such a view can span pages (e.g. “next 10”) making the user navigate from one page to another to examine the results. This application’s interface provides a more compressed view of results and enables users to sort and categorize the results based on the features extracted. A common interface providing such capabilities are interactive trees, such as found in Microsoft’s Windows Explorer.

Fig. 1 shows the front end of the application. Capabilities associated with locating and organizing results are found beneath the standard menubar and toolbar. On the left, the user is given with the list of calamities such as: all, tornado, cyclone, storm, blizzard, forest fire, snow storm, landslide, heat wave, hurricane, volcano, earthquake, flood, and tsunami, to search from and also to include the calamities not listed. Next to this the user is can select the order in which results should be presented. The options are:

1. Date - Ascending / Descending
2. Place - Ascending / Descending
3. Calamity Type - Ascending /Descending

The user is also given the flexibility to choose whether to search the web or to use the cache of articles already retrieved. Searching the live RSS feeds takes time as it involves considerable web scanning. However after every search, the results are stored into a cache. In cases where the user wants to explore the same query results again, searching the cache is much faster. Obviously, the value of the cache depends on the time interval between the search queries. If the time interval is short then the probability that the RSS links have new information is low and hence the cache itself is sufficient. The cached data is refined periodically so that old news is removed.

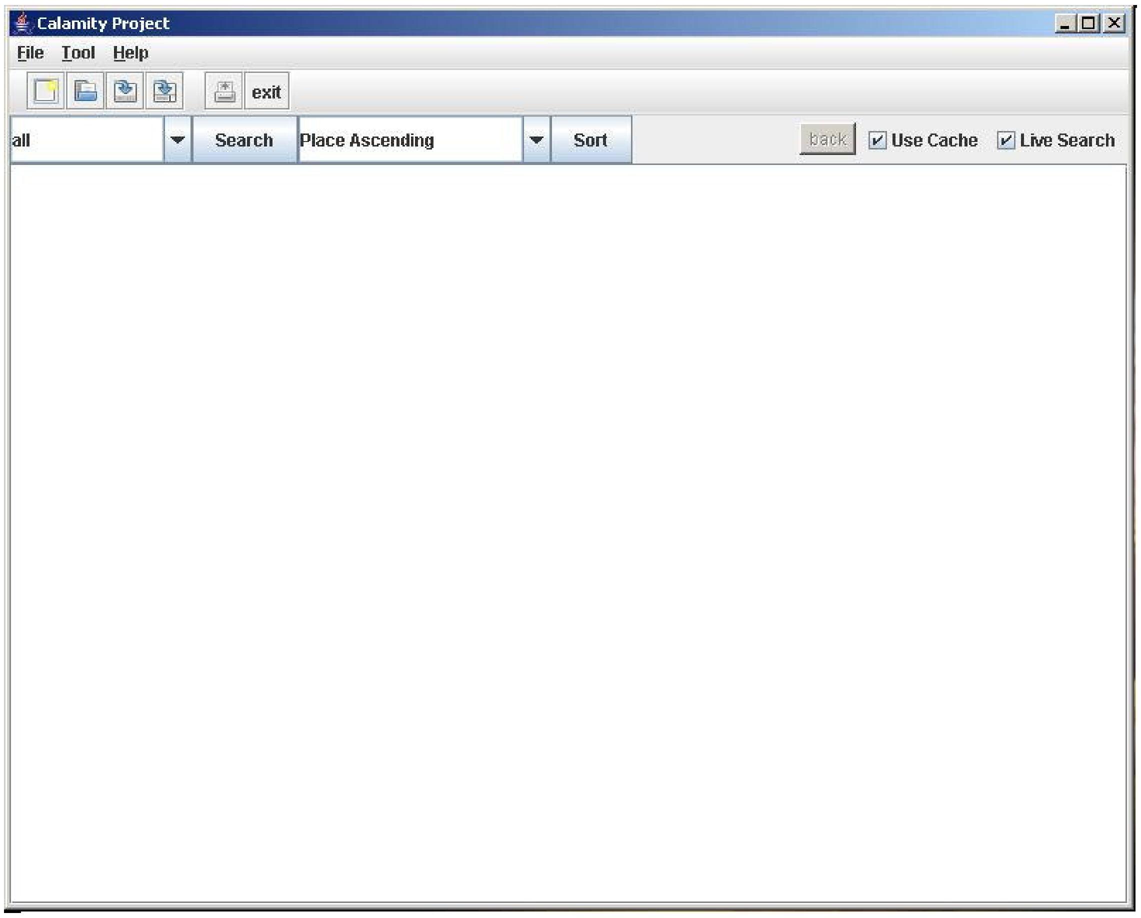
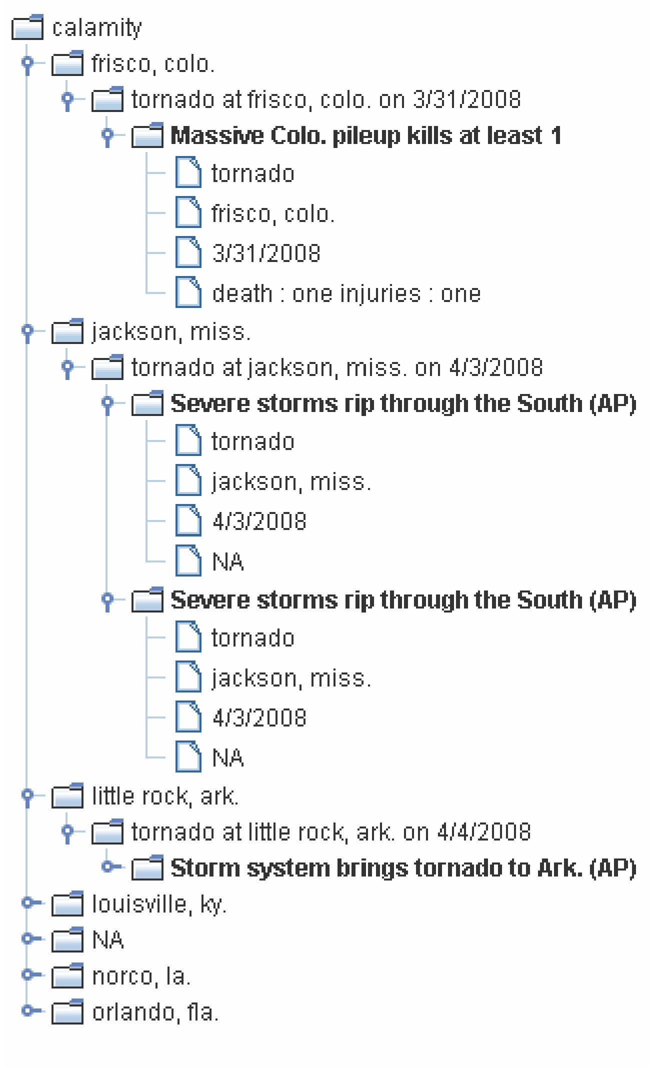


Fig. 1. Front end of the calamity search project

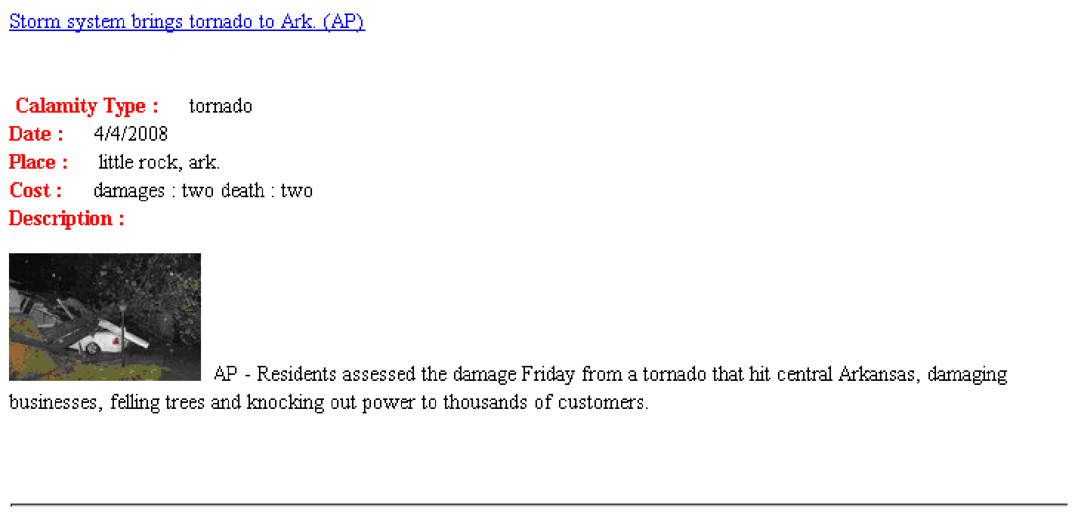


```
calamity
    frisco, colo.
        tornado at frisco, colo. on 3/31/2008
            Massive Colo. pileup kills at least 1
                tornado
                frisco, colo.
                3/31/2008
                death : one injuries : one
    jackson, miss.
        tornado at jackson, miss. on 4/3/2008
            Severe storms rip through the South (AP)
                tornado
                jackson, miss.
                4/3/2008
                NA
            Severe storms rip through the South (AP)
                tornado
                jackson, miss.
                4/3/2008
                NA
    little rock, ark.
        tornado at little rock, ark. on 4/4/2008
            Storm system brings tornado to Ark. (AP)
    louisville, ky.
    NA
    norco, la.
    orlando, fla.
```

Fig. 2. Tree view

4.2.1 Tree View

Fig. 2 shows an example of a tree view for a search query. The tree structure irrespective of the search condition and sort type has 4 levels. The first level is the general level comprising of all the results. The second level consists of either the calamity types, place or date in ascending or descending order according to the search type and the sorting type selected by the user. In Fig. 2 the list is generated after searching for “all” the calamities and sorting the results according to the place in ascending order. The third level in the tree has the calamity news for a corresponding calamity type, place and date. This is the level that best corresponds with what people would likely consider events. News articles that seem to mention the same event, including duplicate articles are grouped together. The fourth level of node corresponds to the actual news articles. For each article, there are four leaves showing the calamity type, place of occurrence, time and the cost (deaths, injuries, damages etc) related to the calamity. The fourth levels nodes can be used to navigate to the actual news snippet page. The page consists of the news headline as a link to the news websites article and the source of the information. A brief description of the page along with the search results - calamity type, place, date and cost. An image corresponding to the calamity is also displayed if one such is available. This image is also clickable and takes one to the news article page. The Fig. 3 shows one such snippet view.



Storm system brings tornado to Ark. (AP)

Calamity Type : tornado
Date : 4/4/2008
Place : little rock, ark.
Cost : damages : two death : two
Description :

AP - Residents assessed the damage Friday from a tornado that hit central Arkansas, damaging businesses, felling trees and knocking out power to thousands of customers.

Fig. 3. Snippet view

4.2.2 Deduplication

Because different news feeds are combined by this application, and because news articles are shared among publications (e.g. AP Newswire), there is a good chance that multiple instances of the same news article will be located. The interface will not remove duplications but will group duplicates together as the bottom level of the tree view. Keeping the multiple pointers provides alternate access paths in case a news article is no longer available from one source. Currently, the duplicate identification algorithm compares the place, date, and calamity type. This approach groups articles about a single event. If the application was collecting news from many different news sources, similar to news aggregators like Google News, then it would be valuable to hide instances where the same article is published by multiple sources in an additional level of the tree.

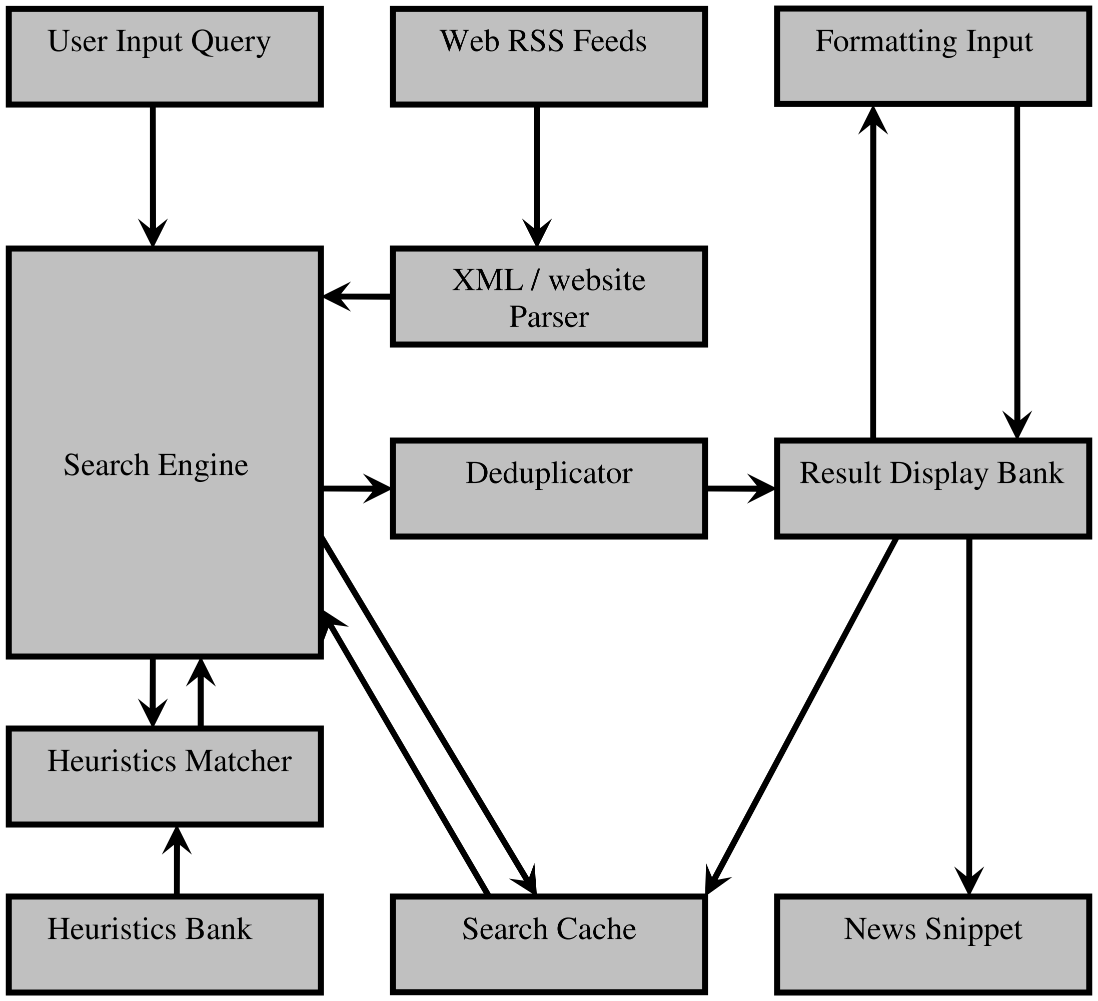


Fig. 4. Search system model

4.2.3 Help System

The application includes a basic help system to explain: search conditions, sort conditions, search modes (cache/live search), the tree view, and the snippet view. This help pages featured in the application are shown in appendix A

4.3 System Architecture

Fig. 4 presents the overall architecture of the application. There are two control inputs from the user, one for the initial search, and other one for formatting the result set. The main data input is from the RSS feeds from which the features are extracted. The secondary data input comes from the cache of the already searched data. The search is carried out using heuristics available in the heuristic bank determined by the heuristic matcher. The output of the search engine goes through the deduplicator which finds duplicated articles.

5. HEURISTIC EVALUATION

A heuristic evaluation of the overall application was done to assess the design of the domain specific search application [19]. A heuristic evaluation consists of having designers not associated with the design identify issues with the design. According to Nielsen, five designers who have expertise in the field related to the software are enough to get a good feedback. For this study, five computer science graduate students with expertise in computer-human interaction and information systems design were given a set of tasks that users of the system were expected to perform. They were asked to identify problems with the initial design. The heuristic evaluation form given to them is shown in Appendix B. The Appendix C lists the set of improvements given by the evaluators. Some of the problems and improvements that were identified and incorporated into the modified system are discussed below.

Remembering the tree structure – In the initial design, whenever there was navigation to and from the snippet page the tree structure went back to the compressed form. This removed any state information indicating what the user had looked at so far, requiring them to manipulate the tree view to get back to their prior state. This was modified in the newest version of the application. Now the application preserves the state of the tree structure between navigations.

Option to add new RSS feeds – The initial system design limited access to news from RSS feeds from a specific set of news websites. In the newest design the user is given the option to add new RSS feeds in addition to the existing ones. New RSS feeds are assigned default information extraction techniques.

Option to set the time for cache cleaning up – In the initial design the cache clears its contents every week. However users may want to have a smaller period of time covered by the cache if they perform the search daily. Similarly, they may want to have a longer period of time to compare and analyze a larger set of events. So in the newest design the user is given the options to set his own cache clearance time.

Preserving activity – The original design was similar to a web search engine in that each time the application was started the activity had to begin from the start (except the user could select to search the cache instead of the live web). Evaluators indicated that users should be able to save the state of their activity across application uses. In the newest design, users can save, load, and print the state of the search activity. This is different from normal search engines where bookmarks and prior searches are saved and results and views are recomputed. Users can remember their status and locate previously visited documents based on their memory of the tree structure and its current view.

Spatial and user control changes – Some of the evaluators specified spatial improvements like positioning of the back button, arrangements in the snippet page which could aid the user in better navigation and easy view. These changes were made in the improved design. The navigation controls for accessing snippet pages was also changed in the improved design.

6. FUTURE WORK

6.1 Adaptive Heuristic Matcher

The system's current heuristic matcher hard codes the selection of an extraction technique to a news source and document feature. This functionality can be improved by making it adapt based on the relative success of different algorithms. One approach to such adaptation is using a finite automata based matcher in the heuristic matcher. By preserving the history of the failure/success of the different algorithms in different conditions, the selection process can evolve if a news site changes its presentation of articles. A difficulty of such an approach is gaining information regarding the success of the algorithms. If an algorithm does not identify the desired information it can be assumed to have failed but user input is needed to determine whether extracted features are indeed appropriate for the article. Users may not provide such feedback.

6.2 Map Oriented Model for Front End

The current front end presents the results in an interactive tree view. However, given the geographic nature of this application domain, a map based display would also make sense. This map would show states with the corresponding markings to show location of events. This would mean a place based sorting of the results. A timeline could be provided to provide a more intuitive time-based presentation of the results.

6.3 Automation of Things to Give Automatic Updates

The current system needs the users to select a category and search for to get an updated list of articles in the tree. Incrementally updating the tree view based on new articles in the selected RSS feeds could help users in situations where they are monitoring for activity rather than performing a limited search task. The system could automatically search for calamities at regular intervals and preserve articles located in the cache for faster access. Users could be notified of new events or new articles about already known events through modifications to the current view and audio cues.

7. CONCLUSIONS

This thesis explores the design of a domain-specific search application based on information extraction. The application design uses a rule-based approach to extract information in the form of location, time and cost of recent calamities. As part of this research, a set of heuristic algorithms for information extraction is developed. The application selects among the set of heuristics depending on the type of calamity, the feature to be extracted, and the source of the article. The interface to the documents retrieved and information extracted is made to facilitate rapid browsing of the selected documents based on any of the features extracted and an initial assessment of document duplication and multiple documents covering the same event. A heuristic evaluation of the application indicated that the application was capable of supporting the desired activities but could be improved by preserving state during tasks and by facilitating more user control over the source of articles, their storage, and their presentation.

REFERENCES

[1] R. Gaizauskas, and Y. Wilks. “Information Extraction: beyond document retrieval”, in *Proc. of Computational Linguistics and Chinese Language Processing*, August 1998, vol. 3, no. 2, pp. 17–60.

[2] http://en.wikipedia.org/wiki/Information_retrieval. Accessed on 06/10/2008.

[3] V. V. Ragavan and G. S. Jung, “A critical investigation of recall and precision as measures of retrieval system performance”, *ACM Transactions on Information Systems*, vol. 7, no. 3, pp. 205–229, 1989.

[4] http://en.wikipedia.org/wiki/Information_extraction. Accessed on 06/10/2008.

[5] A. Mccallum, “Information”, *ACM Queue*, vol. 3, no. 9, pp. 48–57, 2005.

[6] http://en.wikipedia.org/wiki/Named_entity_recognition. Accessed on 06/10/2008.

[7] D. M. Bikel, R. Schwartz, and R. M. Weischedel, “An algorithm that learns what is in a name”, *Machine Learning Archive*, vol. 34, Issue 1-3, pp. 211–231, 1999.

[8] http://minorthird.sourceforge.net/ . Accessed on 06/10/2008.

[9] http://alias-i.com/lingpipe/. Accessed on 06/10/2008.

[10] http://en.wikipedia.org/wiki/Disputes_in_English_grammar. Accessed on 06/10/2008.

[11] S. Soderland, “Learning to extract text based information from the World Wide Web”, in *Proc. of Third International Conference of Knowledge Discovery and Data Mining,* 1997, pp. 251–254.

[12] E. J. Glover, S. Lawrence, M. D. Gordon, W. P. Birmingham, and C. L. Giles, “Web Search - Improving Web searching with user preferences, your way”, *Communications of the ACM*, vol. 44, no. 12, pp. 97–102, 2001.

[13] http://www.answers.com/bb/. Working on 06/10/1998.

[14] A. Kruger, C. L. Giles, F. M. Coetzee, E. Glover, G. W. Flake, S. Lawrence and C. Omlin, “DEADLINER: Building a new niche search engine", in *Proc. of Ninth International Conference on Information and Knowledge Management*, 2000, pp. 272–281.

[15] http://www.flipdog.com/. Accessed on 06/10/2008.

[16] http://www.zoominfo.com/. Accessed on 06/10/2008.

[17] K. D. Bollacker, S. Lawrence, and C. L. Giles, “CiteSeer: An autonomous web agent for automatic retrieval and identification of interesting publications”, in P*roc. of the Second International Conference on Autonomous Agents*, 1998, pp. 116–123.

[18] S. Yu, D. Cai, J. R. Wen, and W. Y. Ma, “Improving pseudo - relevance feedback in web information retrieval using web page segmentation”, in *Proc. of the Twelfth International Conference on World Wide Web*, 2003, pp. 11–18 .

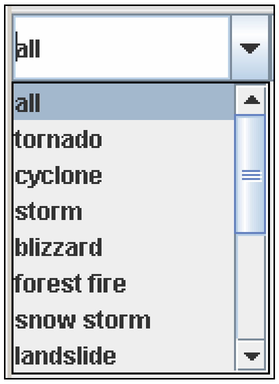
[19] J. Nielsen, & R.L. Mack, *Usability Inspection Methods*, New York, John Wiley and Sons, 1994.

APPENDIX A

1. Help System

Searching

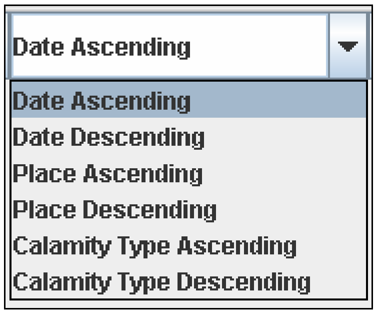
To search for a particular calamity, select the calamity type from the drop down menu.



Sorting

A sorting type can also be selected along with the search condition.

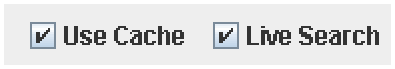
Sorting can also be done after searching on the found set.



Search modes:

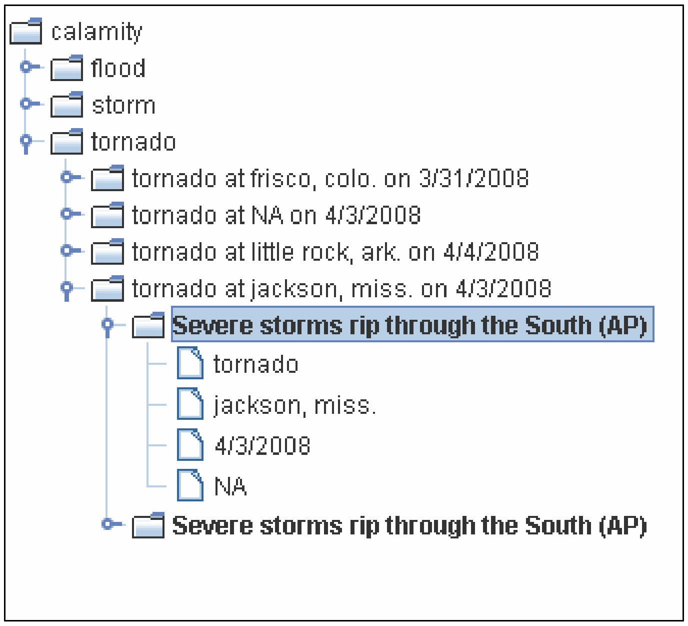
The search modes can be fixed to be either from cache alone or from live web and cache by selecting the corresponding check boxes.

Live search mode search takes time in the order of a few minutes for searching for news across the web.



Tree view

Once searched the results are displayed in the form of a tee view.



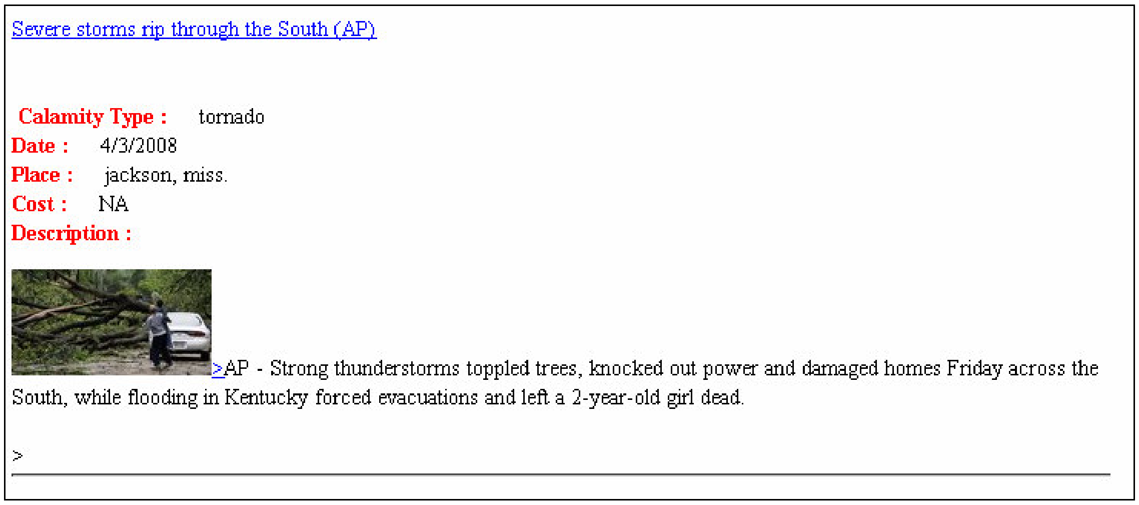
```
calamity
  flood
  storm
  tornado
    tornado at frisco, colo. on 3/31/2008
    tornado at NA on 4/3/2008
    tornado at little rock, ark. on 4/4/2008
    tornado at jackson, miss. on 4/3/2008
      Severe storms rip through the South (AP)
        tornado
        jackson, miss.
        4/3/2008
        NA
      Severe storms rip through the South (AP)
```

The tree view has the news information at the lowest level, showing details for the calamity type, place of occurrence, date and the cost of the calamity.

Double clicking on the calamity new headline at the level four in the tree takes one to the actual news snippet.

Snippet View

The news snippet has a link to the original news. The news snippet displays the details of the calamity such as the type, date, place and cost and a brief description of the calamity along with a picture if available.



Severe storms rip through the South (AP)

Calamity Type : tornado
Date : 4/3/2008
Place : jackson, miss.
Cost : NA
Description :

>AP - Strong thunderstorms toppled trees, knocked out power and damaged homes Friday across the South, while flooding in Kentucky forced evacuations and left a 2-year-old girl dead.

>

A back button appears in the snippet view which takes one to the previous tree view,

APPENDIX B

Heuristic Evaluation

Heuristic evaluation of calamity search project

The application is mainly for insurance companies to get information about calamites so that they would be able to process the levels of damage and extend help.

Possible tasks

The users of the system should be able to perform tasks such as,

1 Searching for the calamities news using both live and cache searches and be able to produce a list of calamities which are related to one another. For example a storm in a northern state could trigger a snow storm in its southern state and further triggering heavy rains and thus floods in further south.
2 Finding the most affected state/city in the recent calamities
3 Finding the most commonly occurring calamity in the US
4. Classification of the certain calamities according to season and place
5. Should be able to navigate from the application to the corresponding news article and extract other information. One such example could include finding the intensity of earthquakes in Richter scale.

The following sections describes the Nielsen's heuristic guidelines, the help for using the application, and finally a section for your heuristic evaluation

Nielsen's heuristic guidelines

Visibility of system status	The system should always keep the user informed about what is going on by providing him or her with appropriate feedback within reasonable time.
Match between system and the real world	The dialogue should be expressed clearly in words, phrases, and concepts familiar to the user rather than in system-oriented terms.
User control and freedom	A system should never capture users in situations that have no visible escape. Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue.
Consistency and standards	Users should not have to wonder whether different words, situations, or actions mean the same thing. A particular system action - when appropriate - should always be achievable by one particular user action. Consistency also means coordination between subsystems and between major independent systems with common user population
Helping users recognize, diagnose, and recover from errors	Good error messages are defensive, precise, and constructive. Defensive error messages blame the problem on system deficiencies and never criticize the user. Precise error messages provide the user with exact information about the cause of the problem. Constructive error messages provide meaningful suggestions to the user about what to do next.
Error prevention	Even better than good messages is a careful design that prevents a problem from occurring in the first place
Recognition rather than recall	The user's short-term memory is limited. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate. Complicated instructions should be simplified.
Flexibility and efficiency of use	The features that make a system easy to learn -such as verbose dialogues and few entry fields on each display - are often cumbersome to the experienced user. Clever shortcuts - unseen by the novice user - may often be included in a system such that the system caters to both inexperienced and experienced users.

Aesthetic and minimalist design	Simple things should be simple. Things should look good, keep graphic design simple, and follow the graphic language of the interface without introducing arbitrary images to represent concepts.

Please evaluate the application based on the Nielsen's 9 guidelines. Provide your evaluations and suggestions for categories which are applicable.

Visibility of the system

Match between system and the real world

User control and freedom

Consistency and standards

Helping users recognize, diagnose and recover from errors

Error prevention

Recognition rather than recall

Flexibility and efficiency of use

Aesthetic and minimalist design

Please also provide your evaluations, suggestions for improvements for the following if any.

Search conditions

Results display format

Help system

Snippet display page

Other you think, not mentioned here

APPENDIX C

The following is the list of the comments given by the evaluators,

1. To include options for user to enter values for new RSS feed links
2. To include options for user to set values of the cache clearance interval
3. To make the tree remember the structure between navigations
4. Changing the back button to prominent place for easy access
5. Changing the click event of the tree to navigate to the snippet page
6. Adding functionalities to save and load the search process
7. Adding functionalities to print the tree.
8. Having a progress bar for the live search process during the search time
9. Giving flashing updates regarding new news and out of date cache
10. Including sorting on the basis of intensity of the calamity and searching based on region
11. Giving preference to the users to set the features of the tree view like size, color, fonts for different levels.
12. Changing the tree icons to correspond to the information they carry

APPENDIX D

Pseudo Code

```
list generateSearchResultList(rssFeeds)
{
    newsArticleList = empty;
    newsArticles = xmlParser(rssFeeds)
    for each article in newsArticles
    {
        trimContents = websiteParser(article)
        featureSet = getFeatureSet (trimContents, article meta data)
        add (featureSet, article metadata) to newsArticleList
    }
    return deduplicator(newsArticleList)
}


list xmlParser( (rssFeeds)
{
    calamityItemList = empty;
    for each item in XML format of rssFeeds do
    {
        description = extract the meta data of the current item
        if(description meta data has calamity describing term)
            add item to calamityItemList
    }
    return calamityItemList
}


article websiteParser(item)
{
    load HTMLTAGSLIST
    url = extract URl information from item's metadata
    contents = download the contents of the url
    contents = contents trimmed using HTMLTAGSLIST
    contents = trim using content semantics using HTMLTAGSLIST
    return contents
}
```

```
featureSet getFeatureSet(input, item)
{
        place = getPlace(input, item)
        date = getPlace(input, item)
        cost = getPlace(input, item)
        return set(place, date, cost)
}


place getPlace(input, item)
{
        function = heuristicMatch(item, place)
        return function (input)
}


date getDate(input, item)
{
        function = heuristicMatch(item, date)
        return function (input)
}


cost getCost(input, item)
{
        function = heuristicMatch(input, date)
        return function (input)
}


function heuristicMatch(item, feature)
{
        load FUNCTIONSET[feature][articleSource]
        for each element in FUNCTIONSET do
        {
                if(FUNCTIONSET.feature == feature &&
                                FUNCTIONSET .articleSource == item.source)
                        return FUNCTIONSET.function
        }
}
```

```
list deduplicator(newsArticelList)
{
    for each pair(article1, article2) in newsArticleList
    {
        if(calamityMatch(article1, article2) == 1 && cumulative (
        datePercentageMatch(article1, article2), palcePercentageMatch(article1,
        article2)) > THRESHOLD)
            Mark (article1, article2) as duplicates
    }
    return newsArticleList
}
```

functionSet:

```
date patternMethod(input)
{
    set dateSet = empty;
    for each word in article do
    {
        if(word matches date pattern)
        {
            dateSet.add(convertToGregorianFormat(date))
        }
    }
    return rankedDate(dateSet)
}
```

```
date referenceWordsMethod(input)
{
    load  REFERENCEWORDLIST
    set dateSet = empty;
    For each word in article do
    {
        if(word matches a element in REFERENCEWORDLIST)
        {
            dateSet.add(compareToGregorianFormat(
                convertToGregorianFormat(date)))
        }
        return rankedDate(datSet)
    }
```

```
}


date rankedDate(dateSet)
{
        for each element in dateSet do
        {
                spatialWeight = percentage of distance from the last word in the article
                occuranceWeight = percentage of number of times the date occurs
                                        corresponding to others
        }
        return max(spatialweighted + WEIGHT * occuranceWeight among the dateSet)
}


place getByFormat(input)
{
        load FORMATSET
        For each word in article do
        {
                If(word matches format)
                {
                        Return cleaneUp(word)
                }
        }
}


cost getByBoundarMarkingMethod(article)
{
        Pair costQuanPair = empty
        load COSTQUANTITATIVETERMS
        load COSTTERMSSET
        For each sentence in article do
        {
                if(sentence has COSTQUANTITATIVETERMS and COSTTERMSSET)
                        sentence = current Sentence
        }
        for each costTerm in sentence do
        {
                Boundary = get pair of opposite nearest cost terms from the costTerm
                costQuanPair.add(costTerm, nearest Quantitave term
                        within boundary}
        }
```

```
    Return costQuanPair
)
```

cost getByDistanceMethod(article)

```
{
    pair costQuanPair = empty
    load COSTQUANTITATIVETERMS
    load COSTTERMSSET
    for each sentence in article do
    {
        if(sentence has COSTQUANTITATIVETERMS and COSTTERMSSET)
            sent = current Sentence
    }
    for each costTerm in sent do
    {
        costQuanPair.add(costTerm, nearest Quantitave term)
    }
    Return costQuanPair
)
```

VITA

Name:	Sudharsan Gunanathan
Address:	12/A7, Lakshmi Street, 100 Feet Road,
	Nanganallur,
	Chennai,
	India – 600061.
Email Address:	sudharsan_svce@yahoo.com
Education:	B.En., Computer Science and Engineering, Anna University
	M.S., Computer Science, Texas A&M University