\* \* \*

**Comparable Entity Mining from Comparative Questions**

**1. INTRODUCTION**

Comparing alternative options is one essential step in decision-makings that we carry out every day. For example, if someone is interested in certain products or services such as digital cameras or treatments, he or she would want to know what the alternatives are and how they compare to each other before making a purchase decision. This type of comparison activity is very common in our daily life but requires high knowledge skill. Magazines such as Consumer Reports and PC Magazine and online media such as CNet.com strive in providing editorial comparison content and surveys to satisfy this need. In the World Wide Web era, a comparison activity typically involves: search for relevant web pages containing information about the targeted products, find competing products, read reviews, and identify pros and cons. In this paper, we focus on finding a set of comparable entities given a user’s input entity. For example, given an entity, Nokia N95 (a cellphone), we want to find comparable entities such as Nokia N82, iPhone and so on. In general, it is difficult to decide if two entities are comparable or not since people do compare apples and oranges for various reasons. For example, “Ford” and “BMW” might be comparable as “car manufacturers” or as “market segments that their products are targeting,” but we rarely see people comparing “Ford Focus” (car model) and “BMW 328i.” Things also get more complicated when an entity has several functionalities. For example, one might compare “iPhone” and “PSP” as “portable game player” while compare “iPhone” and “Nokia N95” as “mobile phone.” Fortunately, plenty of comparative questions are posted online, which provide evidences for what people want to compare, e.g., “Which to buy, iPod or iPhone?”. We call “iPod” and “iPhone” in this example as comparators. In this paper, define comparative questions and comparators as

* Comparative question. A question that intends to compare two or more entities and it has to mention these entities explicitly in the question.
* Comparator. An entity which is a target of comparison in a comparative question.

The goal of this work is, mining comparators from comparative questions and furthermore, provides and rank comparable entities for a user’s input entity appropriately. The results would be very useful in helping users’ exploration of alternative choices by suggesting comparable entities based on other users’ prior requests. To mine comparators from comparative questions, we first have to detect whether a question is comparative or not. According to our definition, a comparative question has to be a question with intent to compare at least two entities. Please note that a question containing at least two entities is not a comparative question if it does not have comparison intent. However, we observe that a question is very likely to be a comparative question if it contains at least two potentially comparable entities. We leverage this insight and develop a weakly supervised bootstrapping method to identify comparative questions and extract comparators simultaneously.

**Objective of the Project**

Comparing one thing with another is a typical part of human decision making process. However, it is not always easy to know what to compare and what are the alternatives. In this paper, we present a novel way to automatically mine comparable entities from comparative questions that users posted online to address this difficulty. To ensure high precision and high recall, we develop a weakly supervised bootstrapping approach for comparative question identification and comparable entity extraction by leveraging a large collection of online question archive. The experimental results show our method achieves F1-measure of 82.5 percent in comparative question identification and 83.3 percent in comparable entity extraction. Both significantly outperform an existing state-of the-art method. Additionally, our ranking results show highly relevance to user’s comparison intents in web.

**2. LITERATURE SURVEY**

**Relational Learning of Pattern-Match Rules for Information Extraction**

Information extraction systems process natural language documents and locate a specific set of relevant items. Given the recent success of empirical or corpus-based approaches in other areas of natural language processing, machine learning has the potential to significantly aid the development of these knowledge-intensive systems. This paper presents a system, Rapier that takes pairs of documents and filled templates and induces pattern-match rules that directly extract fillers for the slots in the template. The learning algorithm incorporates techniques from several inductive logic programming systems and learns unbounded patterns that include constraints on the words and part of-speech tags surrounding the filler. Encouraging results are presented on learning to extract information from computer job postings from the newsgroup misc.jobs.offered.

The ability to extract desired pieces of information from natural language texts is an important task with a growing number of potential applications. Tasks requiring locating specific data in newsgroup messages or web pages are particularly promising applications. Manually constructing such information extraction systems is a laborious task; however, learning methods have the potential to help automate the development process. The Rapier system described in this paper uses relational learning to construct unbounded pattern-match rules for information extraction given only a database of texts and filled templates. The learned patterns employ limited syntactic and semantic information to identify potential slot fillers and their surrounding context. Results on extracting information from newsgroup jobs postings have shown that for one realistic application, fairly accurate rules can be learned from relatively small sets of examples. Future research will hopefully demonstrate that similar techniques will prove useful in a wide variety of interesting applications.

**Collective Information Extraction with Relational Markov Networks**

Most information extraction (IE) systems treat separate potential extractions as independent. However, in many cases, considering influences between different potential extractions could improve overall accuracy. Statistical methods based on undirected graphical models, such as conditional random fields (CRFs), have been shown to be an effective approach to learning accurate IE systems. We present a new IE method that employs Relational Markov Networks (a generalization of CRFs), which can represent arbitrary dependencies between extractions. This allows for collective information extraction" that exploits the mutual influence between possible extractions. Experiments on learning to extract protein names from biomedical text demonstrate the advantages of this approach.

We have presented an approach to collective information extraction that uses Relational Markov Networks to reason about the mutual influences between multiple extractions. A new type of clique template the logical OR template was introduced, allowing a variable number of relevant entities to be used by other clique templates. Soft correlations between repetitions and acronyms and their long form in the same document have been captured by global clique templates, allowing for local extraction decisions to propagate and mutually influence each other.

**Identifying Comparative Sentences in Text Documents**

This paper studies the problem of identifying comparative sentences in text documents. The problem is related to but quite different from sentiment/opinion sentence identification or classification. Sentiment classification studies the problem of classifying a document or a sentence based on the subjective opinion of the author. An important application area of sentiment/opinion identification is business intelligence as a product manufacturer always wants to know consumers’ opinions on its products. Comparisons on the other hand can be subjective or objective. Furthermore, a comparison is not concerned with an object in isolation. Instead, it compares the object with others. An example opinion sentence is “the sound quality of CD player X is poor”. An example comparative sentence is “the sound quality of CD player X is not as good as that of CD player Y”. Clearly, these two sentences give different information. Their language constructs are quite different too. Identifying comparative sentences is also useful in practice because direct comparisons are perhaps one of the most convincing ways of evaluation, which may even be more important than opinions on each individual object. This paper proposes to study the comparative sentence identification problem. It first categorizes comparative sentences into different types, and then presents a novel integrated pattern discovery and supervised learning approach to identifying comparative sentences from text documents. Experiment results using three types of documents, news articles, consumer reviews of products, and Internet forum postings, show a precision of 79% and recall of 81%. More detailed results are given in the paper.

This paper proposed the study of identifying comparative sentences. Such sentences are useful in many applications, e.g., marketing intelligence, product benchmarking, and e-commerce. We first analyzed different types of comparative sentences from both the linguistic point of view and the practical usage point of view, and showed that existing linguistic studies have some limitations. We then made several enhancements. After that we proposed a novel rule mining and machine learning approach to identifying comparative sentences. Empirical evaluation using diverse text data sets showed its effectiveness.

**Mining Comparative Sentences and Relations**

This paper studies a text mining problem, comparative sentence mining. A comparative sentence expresses an ordering relation between two sets of entities with respect to some common features. For example, the comparative sentence “Canon’s optics are better than those of Sony and Nikon” expresses the comparative relation: (better, {optics}, {Canon}, {Sony, Nikon}). Given a set of evaluative texts on the Web, e.g., reviews, forum postings, and news articles, the task of comparative sentence mining is (1) to identify comparative sentences from the texts and (2) to extract comparative relations from the identified comparative sentences. This problem has many applications. For example, a product manufacturer wants to know customer opinions of its products in comparison with those of its competitors. In this paper, we propose two novel techniques based on two new types of sequential rules to perform the tasks. Experimental evaluation has been conducted using different types of evaluative texts from the Web. Results show that our techniques are very promising.

This paper studied the new problem of identifying comparative sentences in evaluative texts, and extracting comparative relations from them. Two techniques were proposed to perform the tasks, based on class sequential rules and label sequential rules, which give us syntactic clues of comparative relations. Experimental results show that these methods are quite promising.

**Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs**

We present a novel approach to weakly supervised semantic class learning from the web, using a single powerful hyponym pattern combined with graph structures, which capture two properties associated with pattern-based extractions: popularity and productivity. Intuitively, a candidate is popular if it was discovered many times by other instances in the hyponym pattern. A candidate is productive if it frequently leads to the discovery of other instances. Together, these two measures capture not only frequency of occurrence, but also cross-checking that the candidate occurs both near the class name and near other class members. We developed two algorithms that begin with just a class name and one seed instance and then automatically generate a ranked list of new class instances. We conducted experiments on four semantic classes and consistently achieved high accuracies.

Combining hyponym patterns with pattern linkage graphs is an effective way to produce a highly accurate semantic class learner that requires truly minimal supervision: just the class name and one class member as a seed. Our results consistently produced high accuracy and for the states and countries categories produced very high recall. The singers and fish categories, which are much larger open classes, also achieved high accuracy and generated many instances, but the resulting lists are far from complete. Even on the web, the doubly anchored hyponym pattern eventually ran out of steam and could not produce more instances. However, all of our experiments were conducted using just a single hyponym pattern. Other researchers have successfully used sets of hyponym patterns (e.g., (Hearst, 1992; Etzioni et al., 2005; Pas¸ca, 2004)), and multiple patterns could be used with our algorithms as well. Incorporating additional hyponym patterns will almost certainly improve coverage, and could potentially improve the quality of the graphs as well. Our popularity-based algorithm was very effective and is practical to use. Our best-performing algorithm, however, was the 2-step process that begins with an exhaustive search (reckless bootstrapping) and then ranks the candidates using the Out degree scoring function, which represents productivity. The first step is expensive, however, because it exhaustively applies the pattern to the web until no more extractions are found. In our evaluation, we ran this process on a single PC and it usually finished overnight, and we were able to learn a substantial number of new class instances. If more hyponym patterns are used, then this could get considerably more expensive, but the process could be easily parallelized to perform queries across a cluster of machines. With access to a cluster of ordinary PCs, this technique could be used to automatically create extremely large, high-quality semantic lexicons, for virtually any categories, without external training resources.

**Comparable Entity Mining from Comparative Questions**

Comparing one thing with another is a typical part of human decision making process. However, it is not always easy to know what to compare and what are the alternatives. To address this difficulty, we present a novel way to automatically mine comparable entities from comparative questions that users posted online. To ensure high precision and high recall, we develop a weakly-supervised bootstrapping method for comparative question identification and comparable entity extraction by leveraging a large online question archive. The experimental results show our method achieves F1- measure of 82.5% in comparative question identification and 83.3% in comparable entity extraction. Both significantly outperform an existing state-of-the-art method.

In this paper, we present a novel weakly supervised method to identify comparative questions and extract comparator pairs simultaneously. We rely on the key insight that a good comparative question identification pattern should extract good comparators, and a good comparator pair should occur in good comparative questions to bootstrap the extraction and identification process. By leveraging large amount of unlabeled data and the bootstrapping process with slight supervision to determine four parameters, we found 328,364 unique comparator pairs and 6,869 extraction patterns without the need of creating a set of comparative question indicator keywords.

**Mining Knowledge from Text Using Information Extraction**

An important approach to text mining involves the use of natural-language information extraction. Information extraction (IE) distills structured data or knowledge from unstructured text by identifying references to named entities as well as stated relationships between such entities. IE systems can be used to directly extricate abstract knowledge from a text corpus, or to extract concrete data from a set of documents which can then be further analyzed with traditional data-mining techniques to discover more general patterns. We discuss methods and implemented systems for both of these approaches and summarize results on mining real text corpora of biomedical abstracts, job announcements, and product descriptions. We also discuss challenges that arise when employing current information extraction technology to discover knowledge in text.

In this paper we have discussed two approaches to using natural-language information extraction for text mining. First, one can extract general knowledge directly from text. As an example of this approach, we reviewed our project which extracted a knowledge base of 6,580 human protein interactions by mining over 750,000 Medline abstracts. Second, one can first extract structured data from text documents or web pages and then apply traditional KDD methods to discover patterns in the extracted data. As an example of this approach, we reviewed our work on the DiscoTEX system and its application to Amazon book descriptions and computer-science job postings and resumes. Research in information extraction continues to develop more effective algorithms for identifying entities and relations in text. By exploiting the lastest techniques in human-language technology and computational linguistics and combining them with the latest methods in machine learning and traditional data mining, one can effectively mine useful and important knowledge from the continually growing body of electronic documents and web pages.

**Object-Level Ranking: Bringing Order to Web Objects**

In contrast with the current Web search methods that essentially do document-level ranking and retrieval, we are exploring a new paradigm to enable Web search at the object level. We collect Web information for objects relevant for a specific application domain and rank these objects in terms of their relevance and popularity to answer user queries. Traditional PageRank model is no longer valid for object popularity calculation because of the existence of heterogeneous relationships between objects. This paper introduces PopRank, a domain-independent object-level link analysis model to rank the objects within a specific domain. Specifically we assign a popularity propagation factor to each type of object relationship, study how different popularity propagation factors for these heterogeneous relationships could affect the popularity ranking, and propose efficient approaches to automatically decide these factors. Our experiments are done using 1 million CS papers, and the experimental results show that PopRank can achieve significantly better ranking results than naively applying PageRank on the object graph.

This paper studies how to calculate the object popularity scores of Web objects based on their Web popularity and the object relationship graph. Traditional PageRank algorithms are no longer valid because of the existence of heterogeneous relationships between objects. We propose to automatically assign a popularity propagation factor for each type of object relationship. Specifically the contributions of the paper are: (i) A PopRank model which considers both the Web popularity of an object and the object relationship graph to calculate the PopRank score of the Web object; (ii) An automated approach for assigning popularity propagation factors for different types of object relationships using partial ranking lists from domain experts. We propose to use a subgraph of the entire object relationship graph to efficiently search for good propagation factors; (iii) The experiments are done in the context of Libra, an object-level Web search prototype indexing 1 million papers.

**Probabilistic Question Answering on the Web**

Web-based search engines such as Google and NorthernLight return documents that are relevant to a user query, not answers to user questions. We have developed an architecture that augments existing search engines so that they support natural language question answering. The process entails five steps: query modulation, document retrieval, passage extraction, phrase extraction, and answer ranking. In this article, we describe some probabilistic approaches to the last three of these stages. We show how our techniques apply to a number of existing search engines, and we also present results contrasting three different methods for question answering. Our algorithm, probabilistic phrase reranking (PPR), uses proximity and question type features and achieves a total reciprocal document rank of .20 on the TREC8 corpus. Our techniques have been implemented as a Web-accessible system, called NSIR.

We presented a probabilistic method for Web-based Natural Language Question Answering. It has been implemented in a robust system and has been tested on a realistic corpus of questions. One thing we didn’t address in this article is the scalability issue. Even though the current system performs relatively faster than other Web-based question answering systems, the current system’s performance for real-time question answering remains to be improved. One thing that deserves further attention is that, after extensive testing, we found that many preprocessing steps such as page downloading, sentence segmentation, part of speech tagging, and so on, take most of the response time. Even though parallel processing can be used to speed up the downloading phase, the dependence of existing Web search engines as answer sources is really the bottleneck of our system. We expect to improve performance significantly by using a prebuilt snapshot of a search engine’s content. NSIR currently takes between 5 and 30 seconds per question depending on the (user-specified) number of documents to be downloaded from the Web and on the (again userspecified) number of phrases to extract. The current version of NSIR doesn’t include query modulation (Radev et al., 2001b; the process of converting a question to the best query for a given search engine).

**Learning Subjective Nouns using Extraction Pattern Bootstrapping**

We explore the idea of creating a subjectivity classifier that uses lists of subjective nouns learned by bootstrapping algorithms. The goal of our research is to develop a system that can distinguish subjective sentences from objective sentences. First, we use two bootstrapping algorithms that exploit extraction patterns to learn sets of subjective nouns. Then we train a Naive Bayes classifier using the subjective nouns, discourse features, and subjectivity clues identified in prior research. The bootstrapping algorithms learned over 1000 subjective nouns, and the subjectivity classifier performed well, achieving 77% recall with 81% precision.

This research produced interesting insights as well as performance results. First, we demonstrated that weakly supervised bootstrapping techniques can learn subjective terms from unannotated texts. Subjective features learned from unannotated documents can augment or enhance features learned from annotated training data using more traditional supervised learning techniques. Second, Basilisk and Meta-Bootstrapping proved to be useful for a different task than they were originally intended. By seeding the algorithms with subjective words, the extraction patterns identified expressions that are associated with subjective nouns. This suggests that the bootstrapping algorithms should be able to learn not only general semantic categories, but any category for which words appear in similar linguistic phrases. Third, our best subjectivity classifier used a wide variety of features. Subjectivity is a complex linguistic phenomenon and our evidence suggests that reliable subjectivity classification requires a broad array of features.

**3. ANALYSIS**

**Introduction**

The Systems Development Life Cycle (SDLC), or Software Development Life Cycle in [systems engineering](http://en.wikipedia.org/wiki/Systems_engineering), [information systems](http://en.wikipedia.org/wiki/Information_systems) and [software engineering](http://en.wikipedia.org/wiki/Software_engineering), is the process of creating or altering systems, and the models and [methodologies](http://en.wikipedia.org/wiki/Methodologies) that people use to develop these systems. In software engineering the SDLC concept underpins many kinds of [software development methodologies](http://en.wikipedia.org/wiki/Software_development_methodologies). These methodologies form the framework for planning and controlling the creation of an information system the [software development process](http://en.wikipedia.org/wiki/Software_development_process).

**Existing System**

In the World Wide Web era, a comparison activity typically involves: search for relevant web pages containing information about the targeted products, find competing products, read reviews, and identify pros and cons. For example, given an entity, Nokia N95 (a cellphone), we want to find comparable entities such as Nokia N82, iPhone and so on. In general, it is difficult to decide if two entities are comparable or not since people do compare apples and oranges for various reasons.

**Disadvantage:**

1. Things also get more complicated when an entity has several functionalities.
2. difficult to decide if two entities are comparable or not

**Proposed System**

To our best knowledge, this is the first attempt to specially address the problem on finding good comparators to support users’ comparison activity. We are also the first to propose using comparative questions posted online that reflect what users truly care about as the medium from which we mine comparable entities. Our weakly supervised method achieves 82.5 percent F1-measure in comparative question identification, 83.3 percent in comparator extraction, and 76.8 percent in end-to-end comparative question identification and comparator extraction which outperform the most relevant state-of-the-art method by Jindal and Liu significantly.

**Advantages of Proposed System:**

1. Provide and rank comparable entities for a user’s input entity appropriately.
2. The results would be very useful in helping users’ exploration of alternative choices by suggesting comparable entities based on other users’ prior requests.

**3.4. PROCESS MODEL USED WITH JUSTIFICATION**

**SDLC (Umbrella Model):**

**Umbrella Activity**

**Umbrella Activity**

**Umbrella Activity**

1. Feasibility Study
2. TEAM FORMATION
3. Project Specification PREPARATION

Business Requirement Documentation

ANALYSIS & DESIGN

CODE

UNIT TEST

DOCUMENT CONTROL

ASSESSMENT

TRAINING

INTEGRATION & SYSTEM TESTING

DELIVERY/INSTALLATION

ACCEPTANCE TEST

Requirements Gathering

SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

**Stages in SDLC:**

* Requirement Gathering
* Analysis
* Designing
* Coding
* Testing
* Maintenance

**Requirements Gathering** **stage:**

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.



These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term *requirements traceability*.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

* Feasibility study is all about identification of problems in a project.
* No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
* Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

**Analysis Stage:**

The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.



The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

**Designing Stage:**

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.

  
When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

**Development (Coding) Stage:**

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.



The RTM will be updated to show that each developed artifact is linked to a specific design element, and that each developed artifact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

**Integration & Test Stage:**

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.



The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

* **Installation & Acceptance Test:**

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.



The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labor data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

**Maintenance:**

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will under go training on that particular assigned category.

For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

**3.5. Software Requirement Specification**

**3.5.1. Overall Description**

A Software Requirements Specification (SRS) – a [requirements specification](http://en.wikipedia.org/wiki/Requirements_specification) for a [software system](http://en.wikipedia.org/wiki/Software_system) is a complete description of the behavior of a system to be developed. It includes a set of [use cases](http://en.wikipedia.org/wiki/Use_case) that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. [Nonfunctional requirements](http://en.wikipedia.org/wiki/Non-functional_requirements) are requirements which impose constraints on the design or implementation (such as [performance engineering](http://en.wikipedia.org/wiki/Performance_engineering) requirements, [quality](http://en.wikipedia.org/wiki/Quality_%28business%29) standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A [business analyst](http://en.wikipedia.org/wiki/Business_analyst), sometimes titled [system analyst](http://en.wikipedia.org/wiki/System_analyst), is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the [systems development lifecycle](http://en.wikipedia.org/wiki/Systems_development_life_cycle) domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

* [Business requirements](http://en.wikipedia.org/wiki/Business_requirements) describe in business terms *what* must be delivered or accomplished to provide value.
* Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
* Process requirements describe activities performed by the developing organization. For instance, process requirements could specify .Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:
* **ECONOMIC FEASIBILITY**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

* **Operational Feasibility**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

* **TECHNICAL FEASIBILITY**

Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to .the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security.

**3.5.2. External Interface Requirements**

**User Interface**

The user interface of this system is a user friendly Java Graphical User Interface.

**Hardware Interfaces**

The interaction between the user and the console is achieved through Java capabilities.

**Software Interfaces**

The required software is JAVA1.6.

**Operating Environment**

Windows XP, Linux.

**HARDWARE REQUIREMENTS:**

# Processor - Pentium –IV

* Speed - 1.1 Ghz
* RAM - 256 MB(min)
* Hard Disk - 20 GB
* Key Board - Standard Windows Keyboard
* Mouse - Two or Three Button Mouse
* Monitor - SVGA

**SOFTWARE REQUIREMENTS:**

* Operating System : Windows XP
* Programming Language : JAVA
* Front End : AWT, Swing
* Back End : MySql

**4. DESIGN**

**UML diagrams**

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

* + **User Model View**
    1. This view represents the system from the users perspective.
    2. The analysis representation describes a usage scenario from the end-users perspective.
  + **Structural Model view**
    1. In this model the data and functionality are arrived from inside the system.
    2. This model view models the static structures.
* **Behavioral Model View**

It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

* **Implementation Model View**

In this the structural and behavioral as parts of the system are represented as they are to be built.

* **Environmental Model View**

In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

**4.1 Class diagram:-**

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. A class with three sections, in the diagram, classes is represented with boxes which contain three parts:

* The upper part holds the name of the class
* The middle part contains the attributes of the class
* The bottom part gives the methods or operations the class can take or undertake

**Class diagram:**

****

**4.2 Use case diagram:-**

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.

**4.2.1 Use case diagram:**



Use case Diagram for User:



**4.3. Sequence Diagram:**

A **sequence diagram** is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams**, **event scenarios**, and timing diagrams.

**4.3.1 Sequence diagram:**



**4.4 Collaboration diagram**

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behavior of a system.

**4.5.1 Collaboration diagram:**

****

**4.6 Component Diagram**

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.

**4.6.1 Component diagram:**

Component Diagram for Admin:

****

Component Diagram for User:



**4.7 Deployment Diagram**

A **deployment diagram** in the Unified Modeling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.

**4.7.1 Deployment diagram:**

Deployment Diagram for Admin:



Deployment Diagram for User:

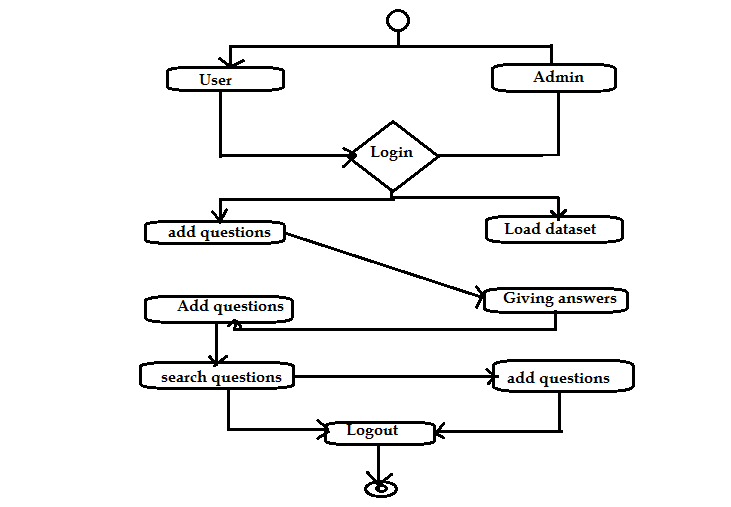


**4.8 Activity diagram:**

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system.

So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent.

**4.8.1 Activity diagram:**



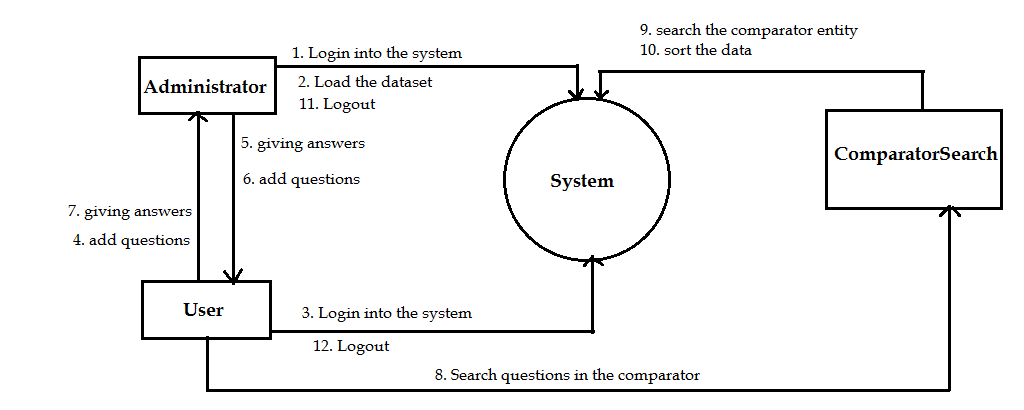
**4.9 Data Flow Diagram:**

[Data flow diagrams](http://www.edrawsoft.com/Data-Flow-Diagrams.php) illustrate how data is processed by a system in terms of inputs and outputs.

Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.

**Data Flow Diagram:**

****

**5. IMPLEMENTATION**

**5.4. Sample code**

**ComparatorSearch.java**

package com;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import java.util.ArrayList;

import java.util.Map;

import java.util.HashMap;

import org.jfree.ui.RefineryUtilities;

public class ComparatorSearch extends HttpServlet{

public void doPost(HttpServletRequest request, HttpServletResponse response)throws ServletException, IOException{

response.setContentType("text/html");

PrintWriter out = response.getWriter();

String question = request.getParameter("t1").trim().toLowerCase();

HashMap<String,Integer> result = new HashMap<String,Integer>();

try{

ArrayList<String> cq = PosTagger.cq;

HashMap<String,String> map = PosTagger.map;

for(int i=0;i<cq.size();i++){

String cqq = cq.get(i);

String value = map.get(cqq);

if(value != null){

String arr[] = value.split("#");

if(arr[0].trim().toLowerCase().equals(question)){

result.put(arr[0],(result.containsKey(arr[0]) ? 1 + result.get(arr[0]) : 1));

result.put(arr[1],(result.containsKey(arr[1]) ? 1 + result.get(arr[1]) : 1));

}

if(arr[1].trim().toLowerCase().equals(question)){

result.put(arr[0],(result.containsKey(arr[0]) ? 1 + result.get(arr[0]) : 1));

result.put(arr[1],(result.containsKey(arr[1]) ? 1 + result.get(arr[1]) : 1));

}

}

}

ArrayList<ComparatorSort> list = new ArrayList<ComparatorSort>();

for(Map.Entry<String,Integer> entry : result.entrySet()){

ComparatorSort cs = new ComparatorSort();

cs.setComparator(entry.getKey());

cs.setCount(entry.getValue());

list.add(cs);

}

java.util.Collections.sort(list,new ComparatorSort());

if(list.size() > 0){

Chart chart1 = new Chart("Comparator Frequency Chart",list);

chart1.pack();

RefineryUtilities.centerFrameOnScreen(chart1);

chart1.setVisible(true);

}

HttpSession session = request.getSession();

session.setAttribute("list",list);

RequestDispatcher rd = request.getRequestDispatcher("ComparatorSearchResult.jsp");

rd.forward(request, response);

}catch(Exception e){

e.printStackTrace();

}

}

}

**ComparatorSort.java**

package com;

import java.util.Comparator;

public class ComparatorSort implements Comparator<ComparatorSort>

{

String comparator;

int count;

public void setComparator(String comparator){

this.comparator = comparator;

}

public String getComparator(){

return comparator;

}

public void setCount(int count){

this.count=count;

}

public int getCount(){

return count;

}

public int compare(ComparatorSort b1, ComparatorSort b2){

double s1 = b1.getCount();

double s2 = b2.getCount();

if (s1 == s2)

return 0;

else if (s1 > s2)

return 1;

else

return -1;

}

}

**6. TESTING**

**Implementation and Testing:**

Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

## Implementation

The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the system should be modifies as a result of a programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

## Testing

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. Actually testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period.

### System Testing

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to user the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

### Module Testing

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. The comparison shows that the results proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

### Integration Testing

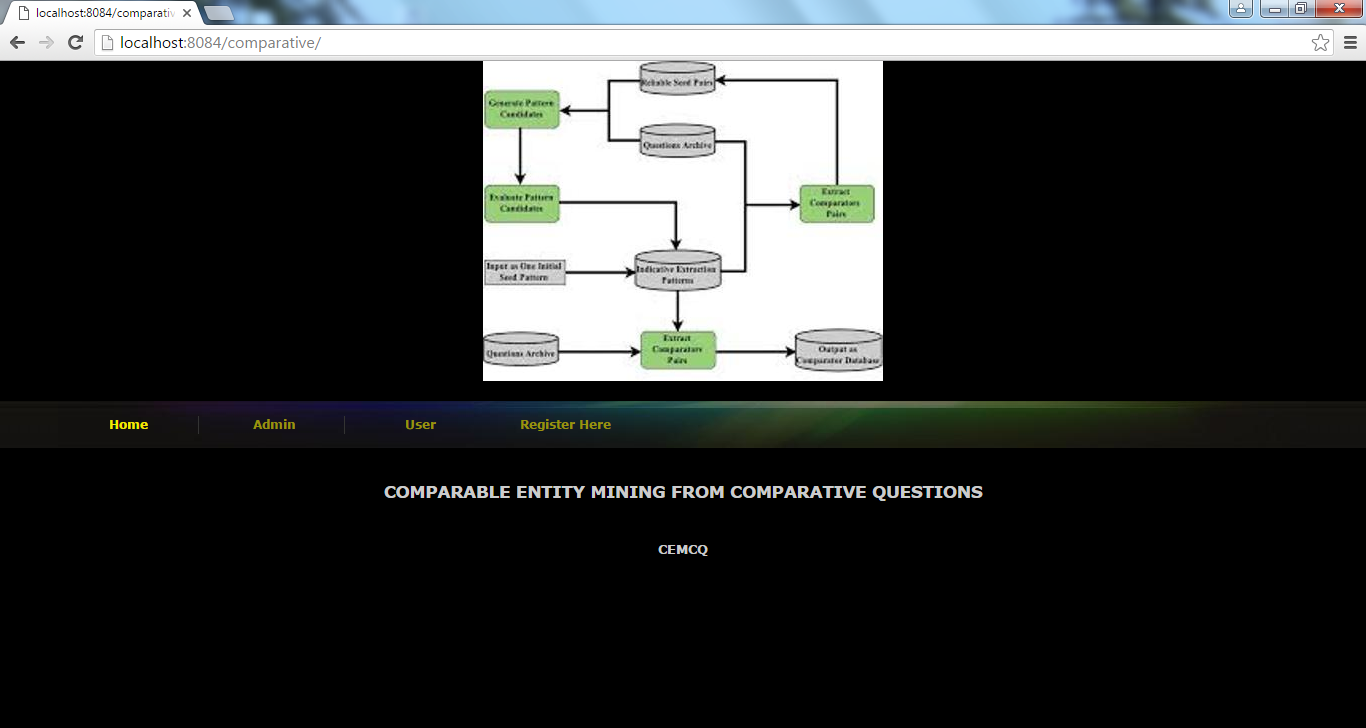
After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

### Acceptance Testing

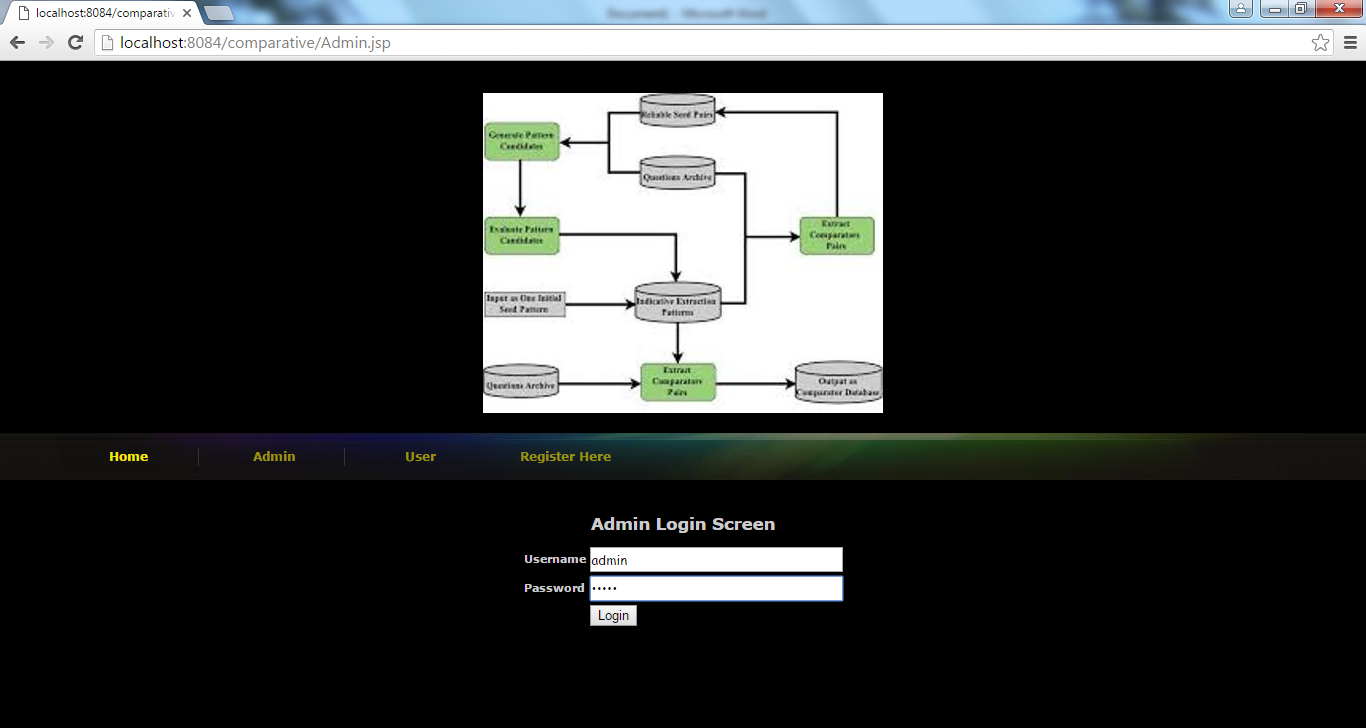
When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

**7. SCREEN SHOTS**

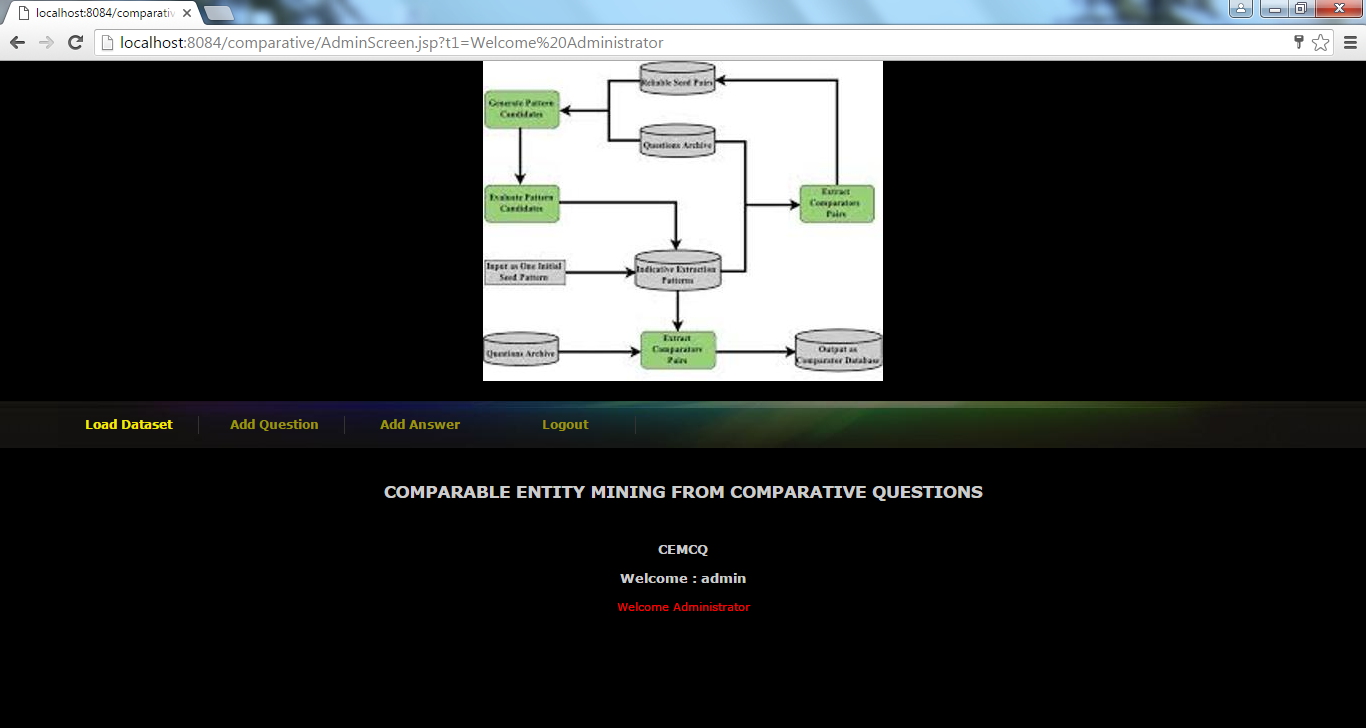
Index page



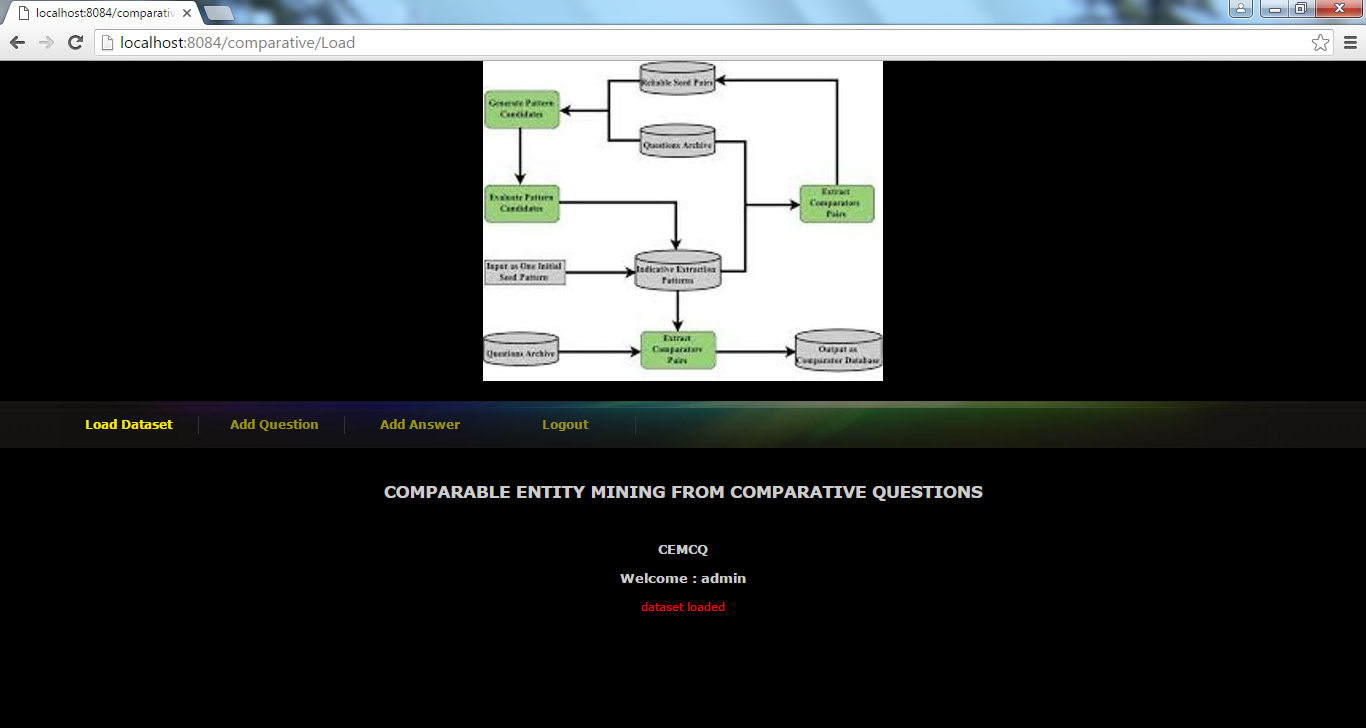
Admin Login



Admin Screen

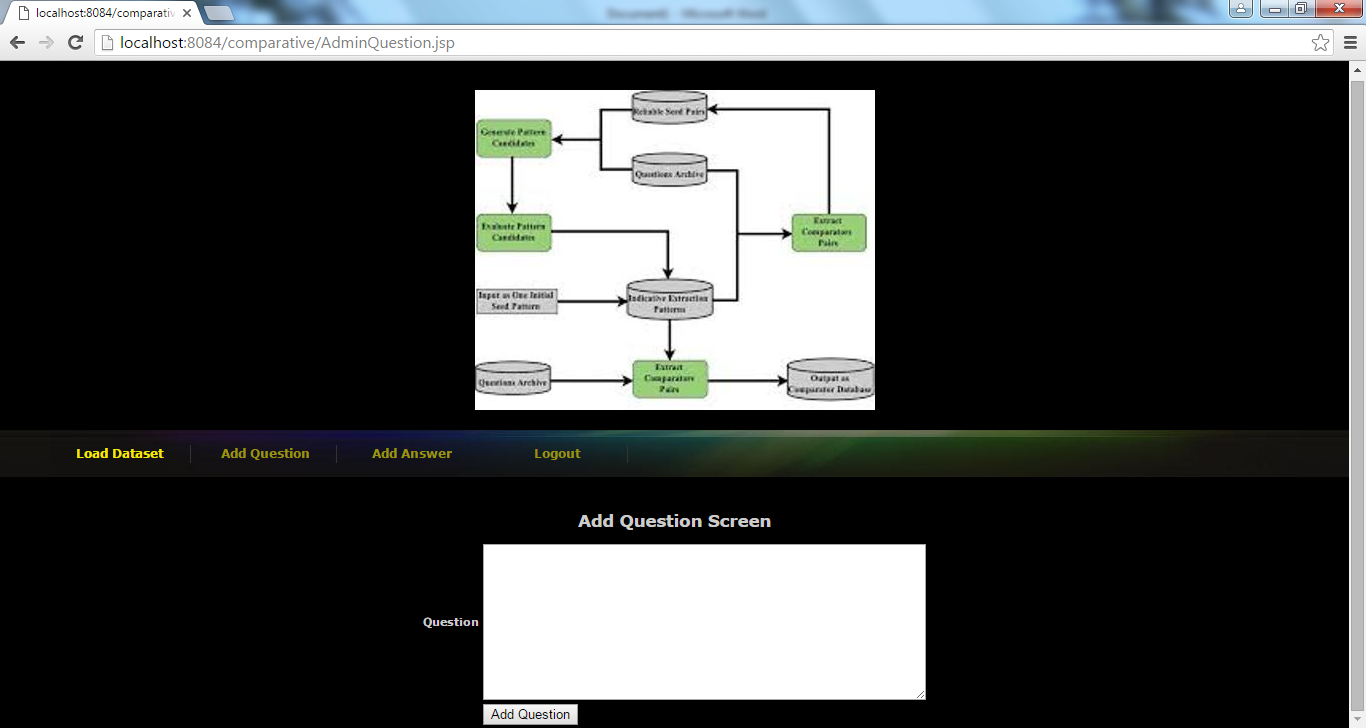


Using load dataset function admin load dataset questions and all other questions users or admin has added manually. While loading dataset application will take only comparative entities

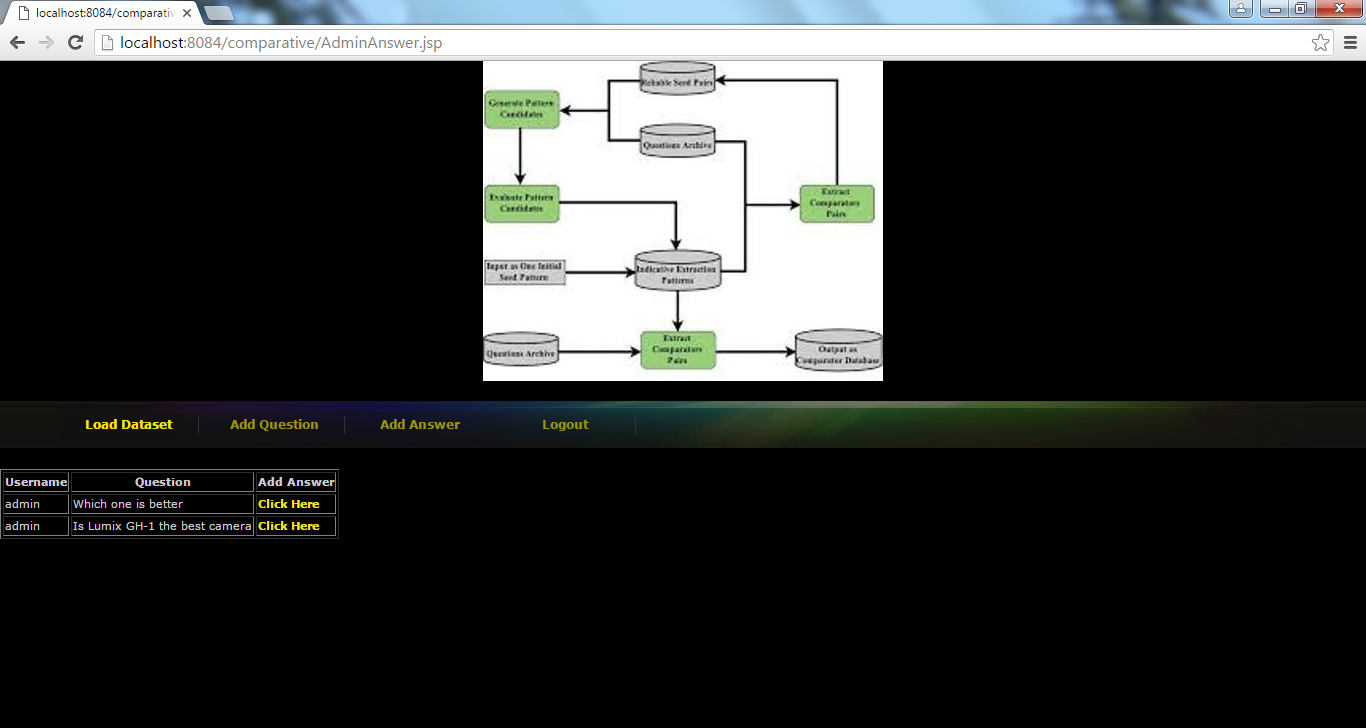


Above screen after loading dataset

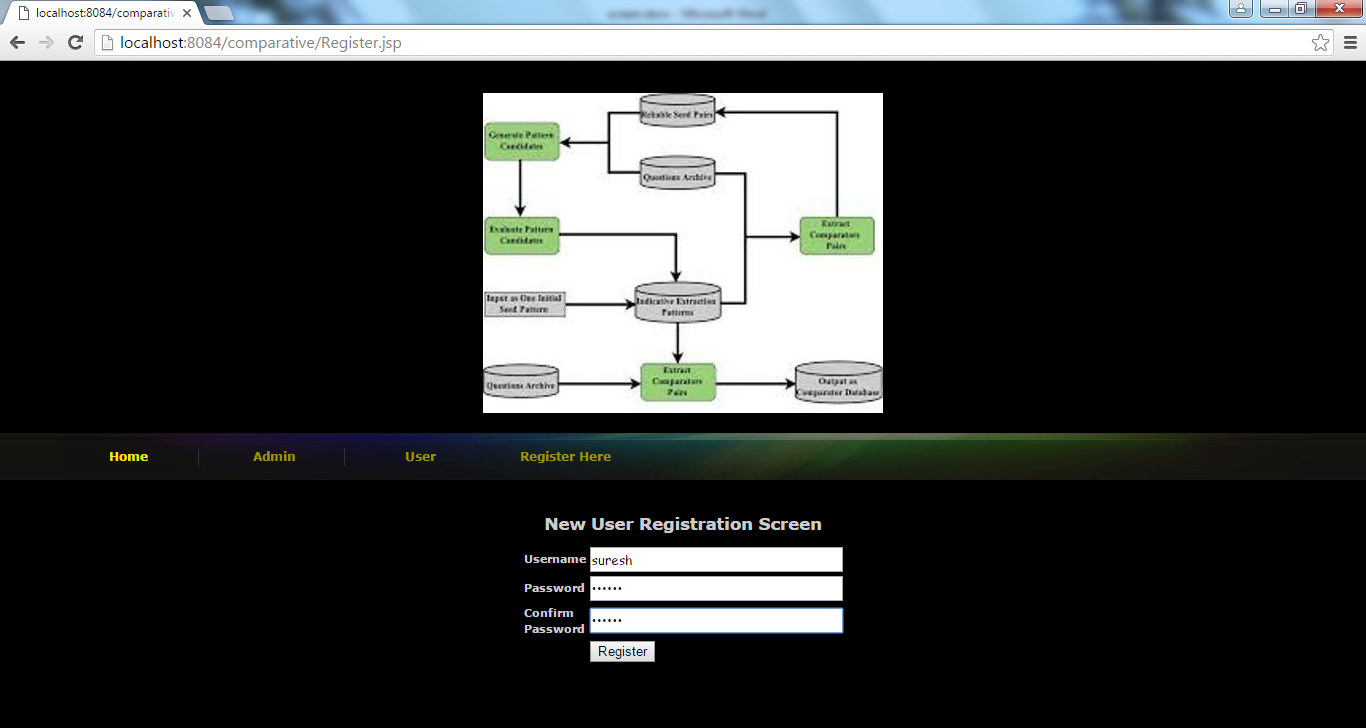
Admin to add question



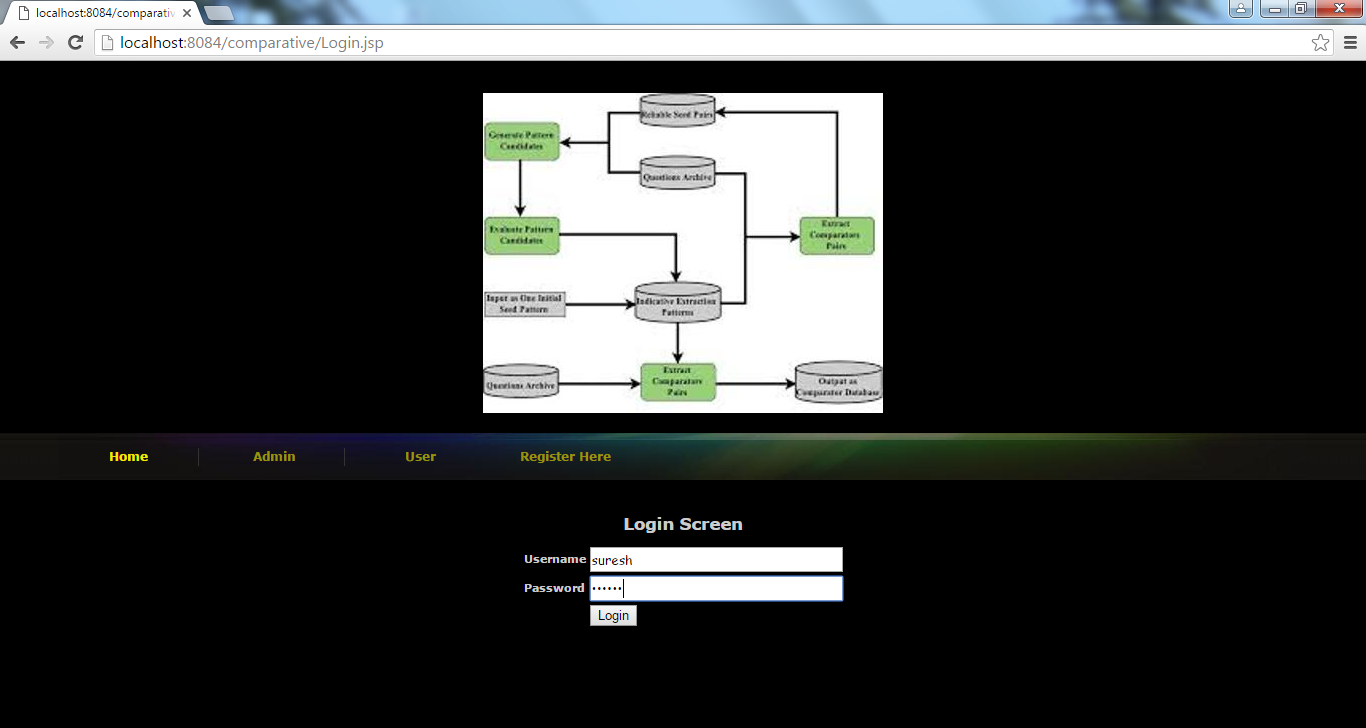
Admin to give answer



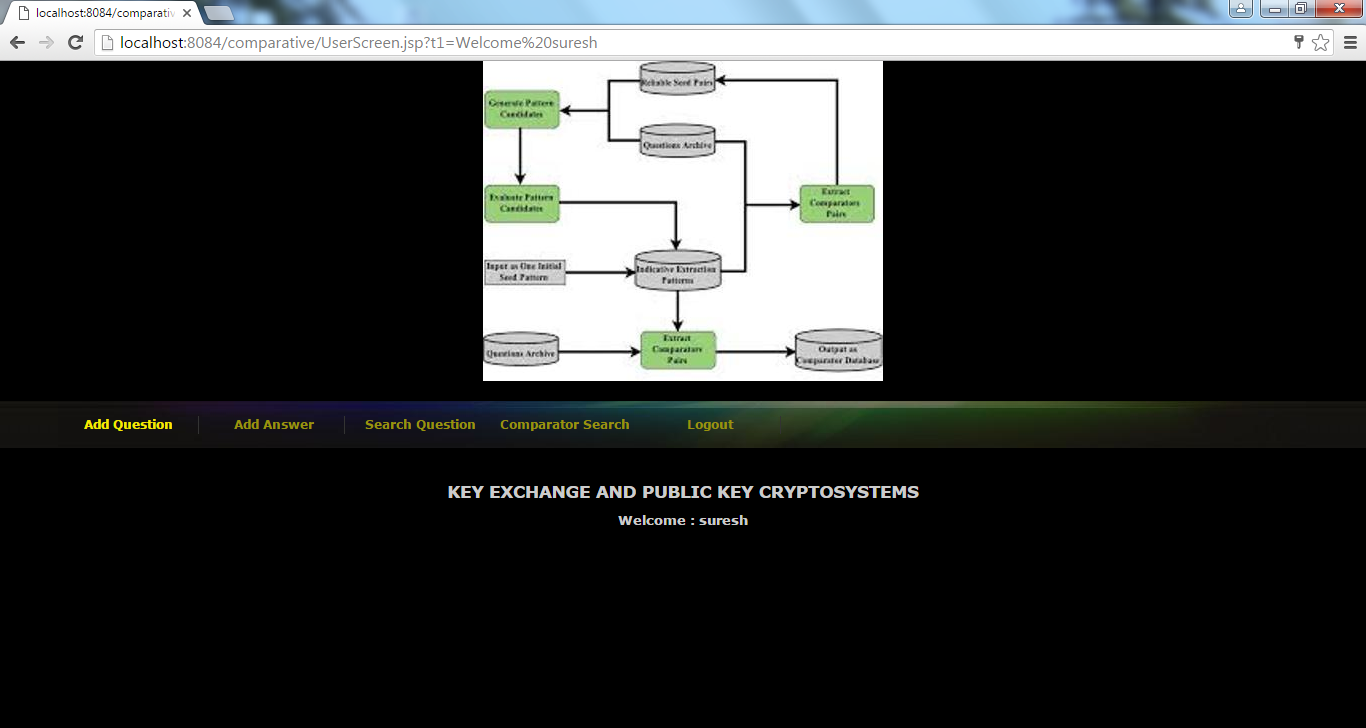
New user registration



New user login

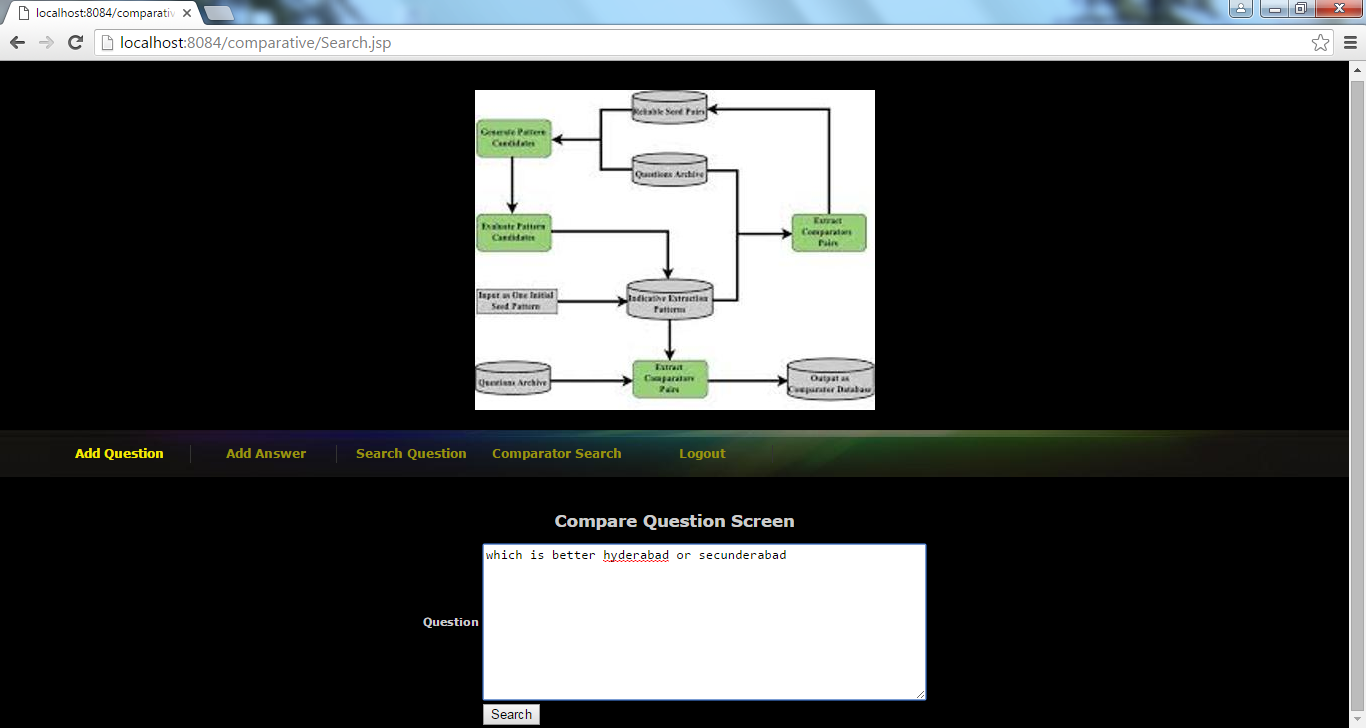


User screen

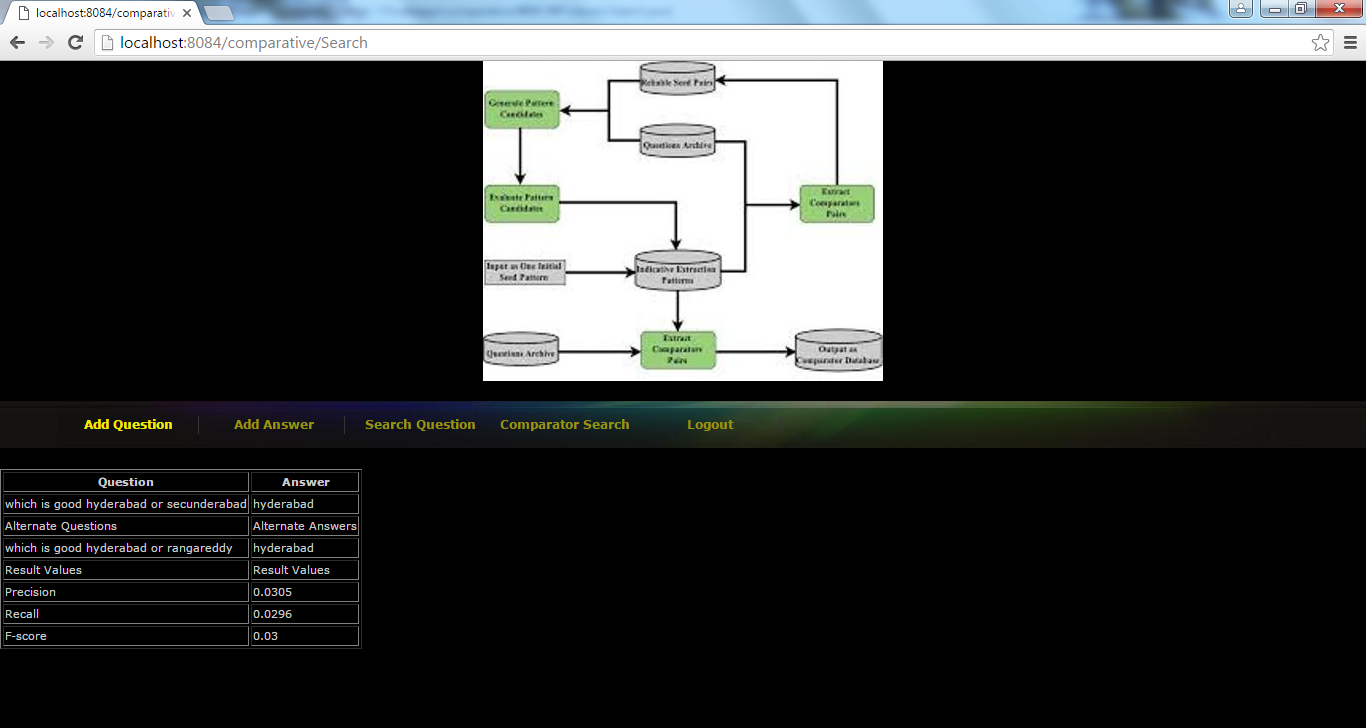


Users also can add and answer question

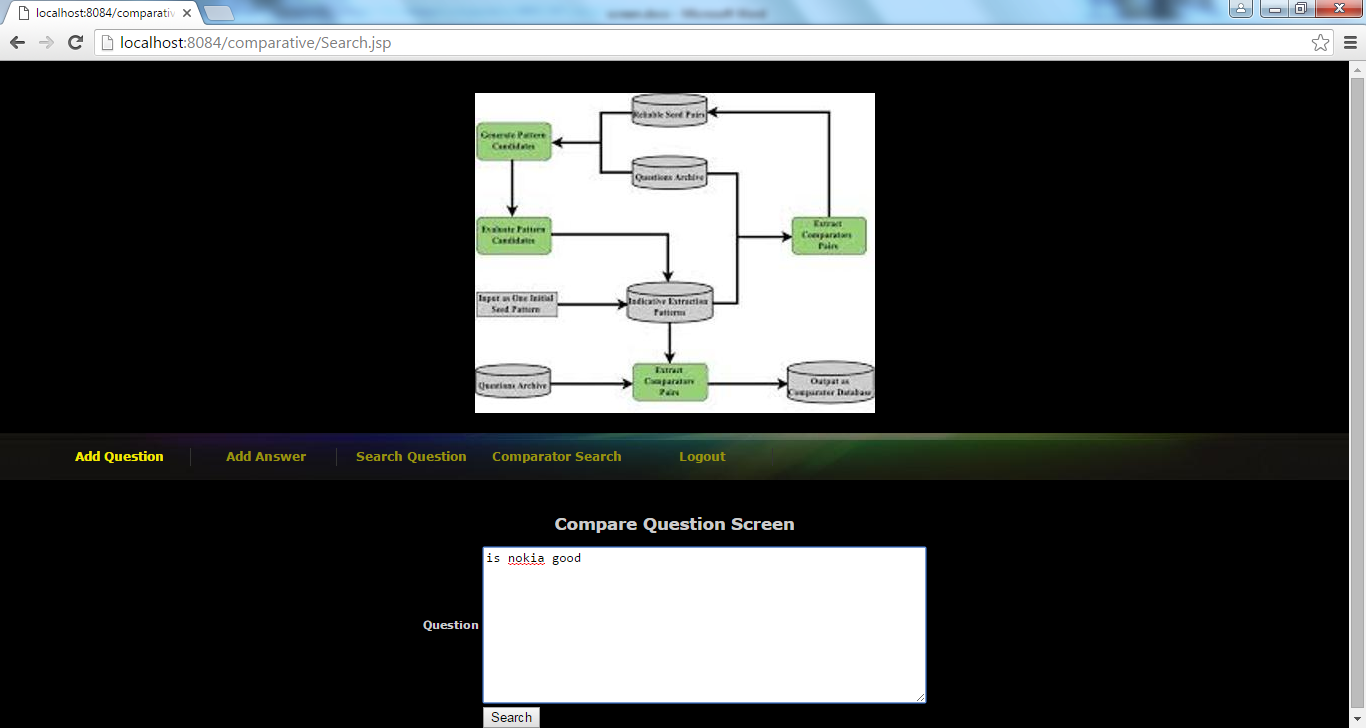
Search question screen



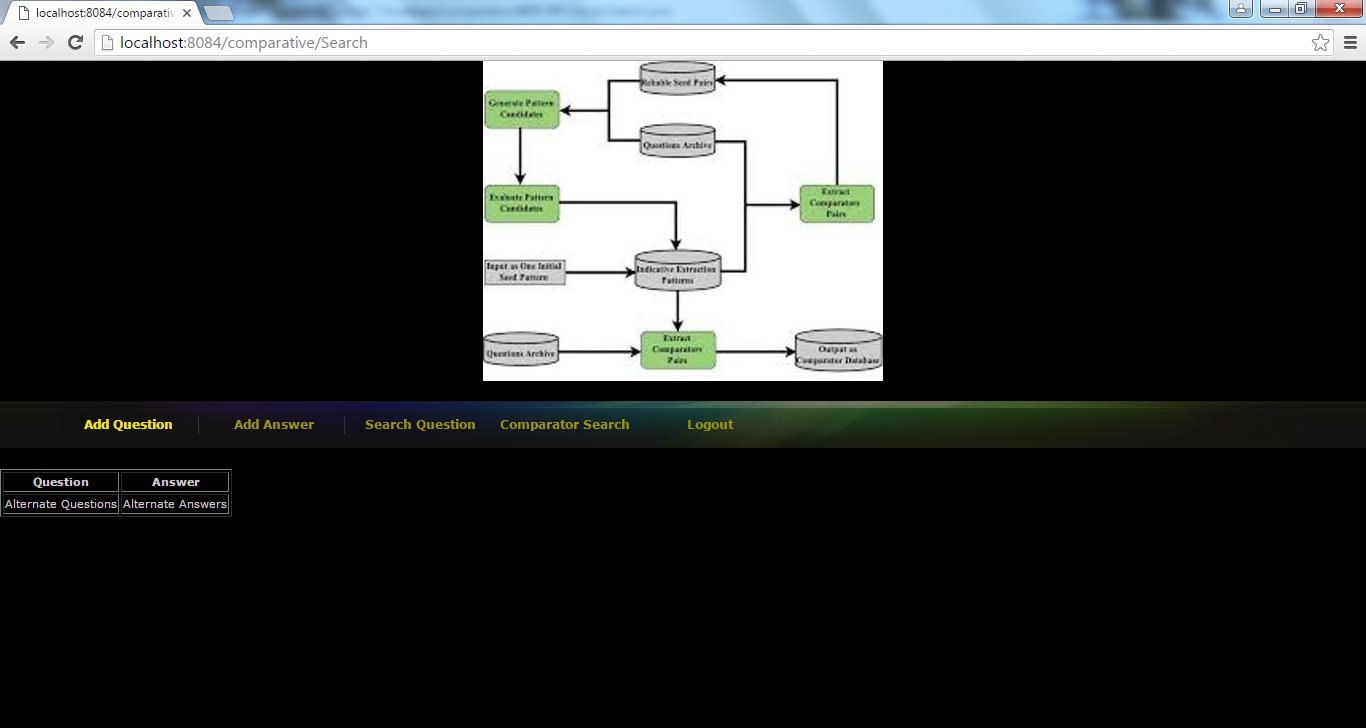
Query answer



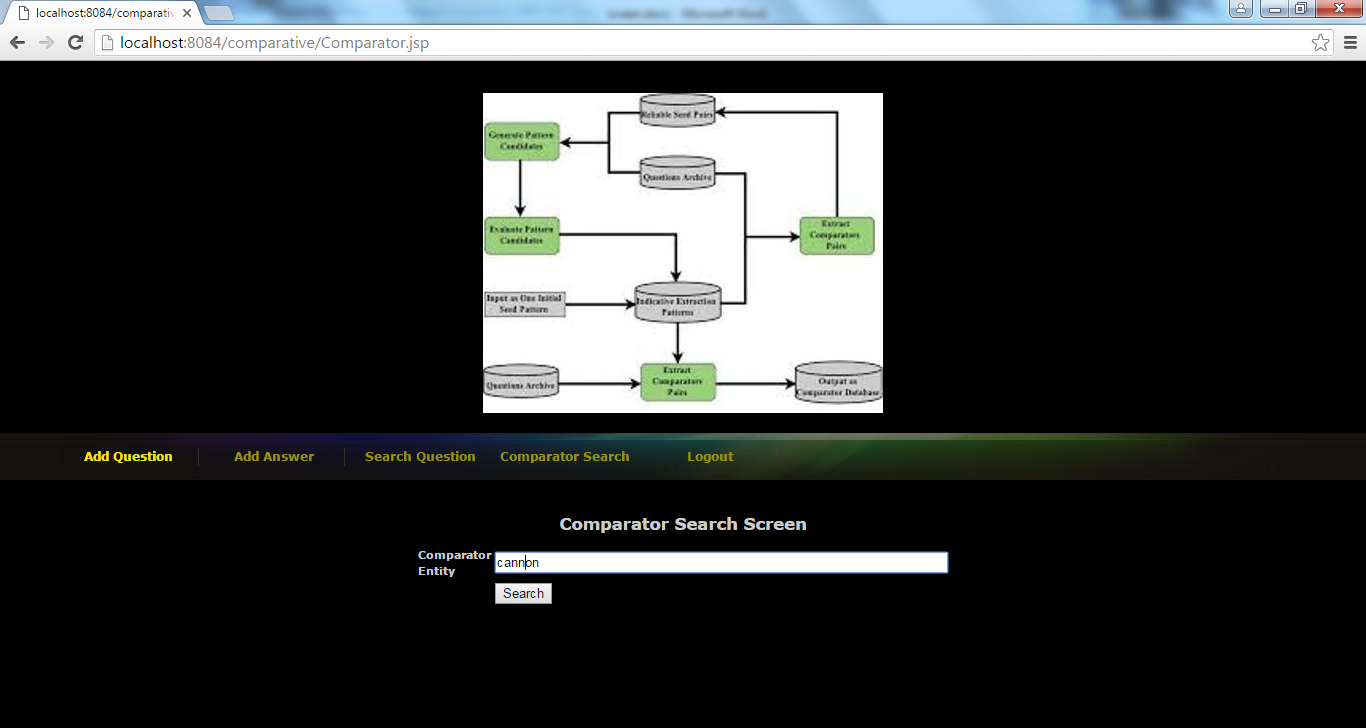
Un-comparative question



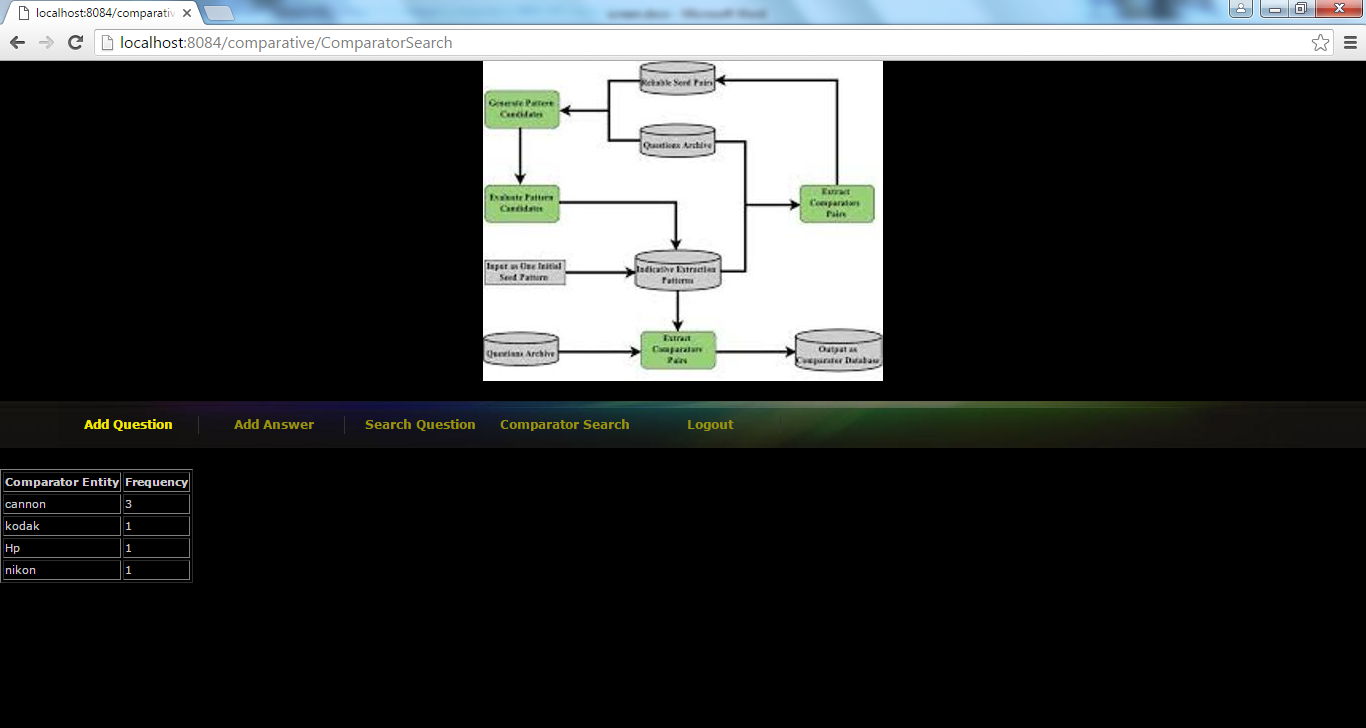
No answer for above queries



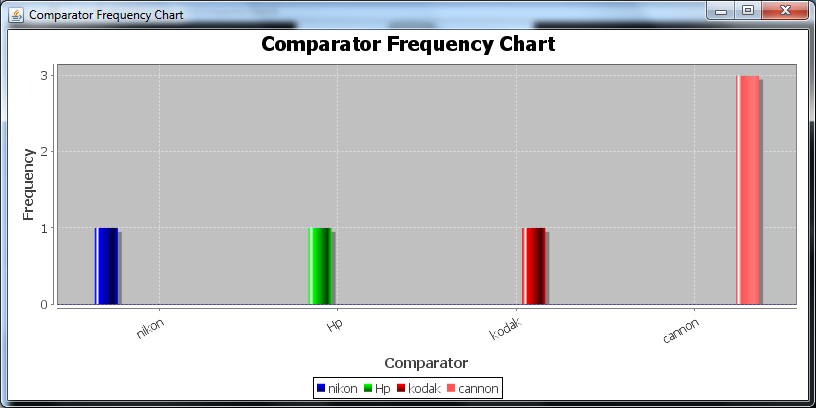
Comparator search screen



Answer for above comparator



Graph for above answer



**8. CONCLUSION**

In this paper, we present a novel weakly supervised method to identify comparative questions and extract comparator pairs simultaneously. We rely on the key insight that a good comparative question identification pattern should extract good comparators, and a good comparator pair should occur in good comparative questions to bootstrap the extraction and identification process. By leveraging large amount of unlabeled data and the bootstrapping process with slight supervision to determine four parameters, we found 328,364 unique comparator pairs and 6,869 extraction patterns without the need of creating a set of comparative question indicator keywords. The experimental results show that our method is effective in both comparative question identification and comparator extraction. It significantly improves recall in both tasks while maintains high precision. Our examples show that these comparator pairs reflect what users are really interested in comparing. Our comparator mining results can be used for a commerce search or product recommendation system. For example, automatic suggestion of comparable entities can assist users in their comparison activities before making their purchase decisions. Also, our results can provide useful information to companies which want to identify their competitors. The experimental results show that our method is effective in both comparative question identification and comparator extraction. It significantly improves recall in both tasks while maintains high precision. Our examples show that these comparator pairs reflect what users are really interested in comparing. Our comparator mining results can be used for a commerce search or product recommendation system. For example, automatic suggestion of comparable entities can assist users in their comparison activities before making their purchase decisions. Also, our results can provide useful information to companies which want to identify their competitors.

**9. BIBLIOGRAPHY**

[1] M.E. Califf and R.J. Mooney, “Relational Learning of Pattern- Match Rules for Information Extraction,” Proc. 16th Nat’l Conf. Artificial Intelligence and the 11th Innovative Applications of Artificial Intelligence (AAAI ’99/IAAI ’99), 1999.

[2] C. Cardie, “Empirical Methods in Information Extraction,” Artificial Intelligence Magazine, vol. 18, pp. 65-79, 1997.

[3] D. Gusfield, Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge Univ. Press, 1997.

[4] T.H. Haveliwala, “Topic-Sensitive Pagerank,” Proc. 11th Int’l Conf. World Wide Web (WWW ’02), pp. 517-526, 2002.

[5] G. Jeh and J. Widom, “Scaling Personalized Web Search,” Proc. 12th Int’l Conf. World Wide Web (WWW ’02), pp. 271-279, 2003.

[6] N. Jindal and B. Liu, “Identifying Comparative Sentences in Text Documents,” Proc. 29th Ann. Int’l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR ’06), pp. 244-251, 2006.

[7] N. Jindal and B. Liu, “Mining Comparative Sentences and Relations,” Proc. 21st Nat’l Conf. Artificial Intelligence (AAAI ’06), 2006.

[8] Z. Kozareva, E. Riloff, and E. Hovy, “Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs,” Proc. Ann. Meeting of the Assoc. for Computational Linguistics: Human Language Technologies (ACL-08: HLT), pp. 1048-1056, 2008.

[9] S. Li, C.-Y. Lin, Y.-I. Song, and Z. Li, “Comparable Entity Mining from Comparative Questions,” Proc. 48th Ann. Meeting of the Assoc. for Computational Linguistics (ACL ’10), 2010.

[10] G. Linden, B. Smith, and J. York, “Amazon.com Recommendations: Item-to-Item Collaborative Filtering,” IEEE Internet Computing, vol. 7, no. 1, pp. 76-80, Jan./Feb. 2003.