Cocobean
A Socially Ranked Question Answering System

Ethan DeYoung
Jerry Ye
Jimmy Chen
Srinivasan Ramaswamy

Advisor: Professor Marti Hearst

May 8, 2008

Masters Project Report

School of Information

University of California, Berkeley

Contents

Cocobean:
A Socially Ranked Question Answering System *

Ethan DeYoung, Jerry Ye, Jimmy Chen, Srinivasan Ramaswamy
School of Information
University of California, Berkeley
{ethan, jerryye, jimmy, srini}@ischool.berkeley.edu

May 12, 2008

Abstract

Search in its current form is still in its infancy. Finding exactly what one wants is usually a process of constantly refining ones query with additional words. Traditional search engines have no inherent understanding of what a user wants and matches documents to queries based on the presence of or lack of search terms. In natural language search, a user prompts the system with a coherent question and the system returns a single or a set of candidate answer sentences. By making search more natural and intuitive, natural language search is fundamentally changing how we search the web and delivering higher quality results.

Our project consists of a natural language search engine on the Wikipedia dataset, utilizing both algorithmic search as well as methods for users to socially influence the ranking of results. One of the main components of our system consist of a traditional question answering system based on a paper by Microsoft Research. The paper calls for the creation of question-to-answer transformations to figure out how a candidate answer might look like given a question. We use Lucene to find relevant documents in our corpus and extract candidate answers. Our system then learns which transformations worked well by pairing candidate answers with their transformation and having users vote on the correctness of the answer. A machine learned ranking technique utilizing both algorithmic and social features is used to reorder the results.

1 Introduction

Traditional search engines work by treating each term in a query independently and attempts to match documents to the query based on some similarity measure. One of the main advantages of this type of system is that this entirely automatic process returns answers to users instantly. User queries are instantly compared to documents in the corpus and ranked results are returned. However, since terms are treated independently and there is no semantic understanding of the query or the documents in the corpus, this method can suffer by returning irrelevant results.

Question answering systems have a wide range of complexity, ranging from usage of simple regular expressions to statistical methods utilizing language models and parse trees. Question answering systems can work well on the domain it was designed for but may miss entirely for unexpected questions. More automated systems might not have the domain problem but may require retraining of the system to changing datasets. For some systems that presents multiple choices, there is usually no mechanism for users to select the correct choice.

On the other end of the spectrum is community driven question answering systems. These systems are entirely human driven and questions are answered only when a person responds to it. Inherently, these systems require users to wait until someone else responds and there is no guarantee of accuracy or relevancy of the response.

*available at cocobean.com

Our project attempts to merge aspects of all these systems into a socially ranked question answering system by combining algorithmic ranking with social ranking. Using the Wikipedia corpus as a knowledge base, we indexed everything in Lucene to provide an algorithmic ranking measure. Given a question, we generated potential answer patterns from several transformations. Each of these potential answers were passed into Lucene to yield a ranked ordering of candidate documents and snippets. Users of the system are then given the chance to vote on which result they believed answered their question. These responses are aggregated across all users and a reputation score is generated for each user. A machine learned ranking approach aggregates these features, among others, and predicts a relevancy score for each answer, directly influencing the order of answers on the results page.

2 Related Work

Attempts at question answering has taken many forms, as interfaces to highly structured databases, highly manual methods that required human constructed patterns, statistical methods that leverages natural language processing, or completely human based systems that relies on social networks. Each method has its own benefits, with more manual systems providing more accurate answers than automated methods but suffering scalability issues that the later does not have.

Since the 1960's, natural language interfaces to databases (NLIDB) allowed for natural language queries to be used when searching highly structured databases. Early methods relied heavily on simple pattern matching and although sometimes working well, they were domain specific and failed miserably for queries that didn't match existing patterns [4]. Another approach to NLIDBs are statistical learning systems with the goal of learning rules for parsing sentences into traditional relational database queries. Zelle and Mooney were early pioneers of such learning systems, relying logic programming and shift-reduce parsers [12]. More recently, there has been interest in automatically learning an intermediate language using machine learning. Zettlemoyer and Collins introduced a supervised approach of mapping natural language sentences into lambda calculus using a log-linear model [13]. The main difference between NLIDBs and what we have done is that we don't have a highly structured database of facts. Wikipedia is mostly unstructured and our system has to handle both parsing of the query sentence as well as the candidate answers.

Dumais et al [6] explore how the redundancy of information on the web could enhance the accuracy of question-answering systems. The occurrence of multiple linguistic formulations of same answers can increase the chance to find an answer that occurs within the context of a simple pattern match with the query. The system reformulates the questions using hand-coded rules to get multiple answers from the search engine. A corresponding set of filters was developed to help extract good answers based on question types. The result shows that the redundancy in the web significantly improved the accuracy.

In contrast to manually created transformation rules, some works focus on learning question-answer patterns. Ravichandra and Hovy [10] proposed a simple learning approach that requires a lot less effort than manually creating the rewrite rules. The system constructs a table of answer patterns that are learned from the question-answer pairs and the corresponding sentences in the returned documents. To find the answers, the system takes the sentences in the returned documents to search in the pattern table. The answer term in the sentences would be extracted according to the pattern.

Agichtein et al [3] propose a technique to learn question to query transformations from question-answer pair dataset. In this approach various possible transformations associated with the identified question phrases are learned by mining the the first n bytes of the answer. The transformations are further filtered and weighted for further query expansion. These transformations are tested with a search engine and the transformations are re-weighted. This approach is interesting and the evaluation shows better performance over ordinary search engines, but the scalability of this approach to new questions should be examined in more detail.

Roussinov and Robles [11] proposes a self-learning approach to learn patterns for open domain question answering on the web. General patterns are learned for different question types by replacing the question and answer keywords with symbols. The results retrieved by querying a search engine with questions formed by those patterns are refined by means of scoring sub-phrases, where high scores are assigned to longer

sub-phrases. The answers are checked for convergence (triangulation) by measuring its similarity with other answers and filtered based on its type. This approach is claimed to be an improvement over Dumais et al [6] who uses rule based rewrites. The evaluation does not show very good results but we hope it has a lot of room for improvement. The automatic identification and training of patterns seems interesting and we might use it in future.

Some works use NLP techniques to get the semantic meaning of questions and answers. This could facilitate the process of question parsing, classification, and formulation. Kwok el al [8] introduced the Mulder QA system that targeted the web as the information source and applied NLP techniques such as part-of-speech tagging and WordNet to question classification/transformation and answer extraction. The questions are classified to determine what type of information is being requested so that the system may better recognize an answer. From the returned search summary, the expected answer type such as noun phrases, numbers, and dates are extracted. There is a nice voting procedure that ranks and clusters the candidate answers so as to keep the good answers and remove noise data.

To identify the correct answer for a question, it would be better to identify the type of question thereby the task of finding the answer becomes easier. Li and Roth [9] has demonstrated this by means of a hierarchical question classifier using SNoW learning architecture, which supports multiple labeling. The features extracted from each question are words, POS tags, named entities, chunks and head chunks, named entities and semantically related words. The questions are classified into coarse classes by the first classifier, which in turn generates the candidate labels for the next level. The second classifier classifies the question again into even finer classes. The authors show that the machine learning based classifier works very well, but the coarse level classifier didn't help much in terms of performance. We are using the dataset proposed in this paper and initially we are planning to focus only on the coarse categories.

Zhang and Lee [14] proposes that Support Vector machines outperforms all other algorithms even with very simple surface text features for Question Classification. For each question they extracted simple features like bag-of-words and bag-of-ngrams. They also use a tree kernel to exploit the syntactic information available in the question. It is shown that the tree kernel has improved the performance of the coarse classifier but there was not a significant difference in the fine-grain classifier. This paper provided inspiration for us to use Support Vector Machines with very simple surface text features as it reportedly outperforms all other classifiers.

Harabagiu et al [7] propose a question answering system with lightweight, knowledge based NLP techniques, where even a shallow, surface based approach surprisingly produces quality results. The question types are identified based on a comparison of the semantic representation of the question and the nodes in the question taxonomy. A set of keywords are generated based on the identified class for each question, which in turn is used to query a search engine to get multiple sets of answers. These set of answers are further refined by the justification technique, which uses a mixture of axioms derived from the answer and the knowledge base. Though this approach is interesting it requires high computational power for parsing and generation and processing of axioms. We believe that our social ranking approach is an effective but less computational replacement to this approach.

On the other end of the spectrum are question answering systems that rely entirely on user interaction. Websites like Yahoo! Answers allows users to post questions to its entire community of users. Receiving an answer to ones question depends on the knowledgability of the community and there might be latency between posting and receiving an answer [1]. Recent approaches to improving search engine relevancy has focused on utilizing implicit user feedback. Click-through information has been shown to improve ranking, particularly in the relevance of the first result [2]. There are currently two search engines that combine traditional search with a social ranking system similar to Digg. While implicit information is definitely informative and we would like to be able to leverage this in the future, we focus on explicit votes from users. As far as we can tell, there are currently two commercial search engines that mixes traditional information retrieval and social ranking. Sproose (http://www.sproose.com) and OOMZ (http://www.oomz.com) returns search results similar to Google while also allowing for users to vote for the relevancy of a result.

3 Methods

In our implementation we use the Wikipedia corpus as our knowledge base. We chose Wikipedia mainly because of its size, the ease of obtaining the dataset, and the lack of intellectual property restrictions. We used the open source search engine Apache Lucene to index and search through the dataset.

Our transformation system conducts question classification using libSVM and our transformations are generated using python. All transformations for the system are stored in a MySQL database and ranking for the system is done in php and utilizes an AJAX interface to update transformation weights in the MySQL database.

Our system consists of two important modules, the user interface that handles user interaction and the server end module that processes queries and returns the ranked answers. We describe the user interface design process in Section 4. Section 5 details the server end module and what happens between our system receiving a query and candidate answers being returned to the user.

4 User Interface Design

4.1 Initial Interviews

How we developed personas

We began by interviewing three individuals who were at the very least, familiar with social ranking websites. We choose such candidates because our target audience is people who read or frequent websites were there is a social ranking component. Two out of the three people we interviewed were active participants in such websites, while the third was almost exclusively a lurker (someone who actively reads social websites but rarely, if ever, participates). The one interviewee who does not participate in social ranking sites declines to do so because of the extra effort needed to register, check email for confirmation and then login. However, this interviewee would be willing to participate in such websites if there were no overhead involved.

For the interviewees that do participate in social ranking websites, we asked them to talk about their voting habits. The two were split in their patterns of voting; one preferred to act as a kind of filter, down-voting any bad articles that he felt should not be featured. Furthermore, this interviewee rarely voted good articles up. The second interviewee who participates in social ranking websites primarily voted good articles up and only occasionally voted bad articles down. One interesting point that both people mentioned is that they do not vote an article up or down after it has received a larger number of votes. They both said they felt it already had enough votes and their vote no longer really mattered.

Because it is directly related to our project, we were also interested to learn how such people use the Internet to find information, specifically answers to questions, and how they formulate queries. Not surprisingly, all of the interviewees used keywords to formulate their queries, although one mentioned that he would intentionally translate a natural language query to a keyword query in his head. This same person also used the longest keyword queries out of all the interviewees.

We found that everyone we interviewed was aware of natural language queries and/or search, but none of them used natural language in their queries. When we inquired as to the reason for this the most common answers were that typing a query using natural language was simply too long and that keyword search typically worked well enough. One of the interviewees mentioned he would use natural language queries if he felt confident the search engine would understand the query. Based on our interviews, we developed three personas that we felt would most accurately represent potential users of our system.

4.2 Personas

Jason
Lurker - Only read but never contributes

Jason is a 26 year old second year graduate student in the computer science department at Stanford University. His primary focus of research is Human Computer Interaction and he is working with his professor as a graduate student researcher in a research project to develop the next generation interface for interacting with computers. In addition to his work as a researcher, he is taking two graduate level courses at school to learn more about the field. He has a tight schedule between his research work and his classes.

Jason uses the Internet (his default source of information) to find most of the literature related to his research. He visited the university library only during his orientation and he felt comfortable with the information he found online. His default webpage is Google, which he uses to access all other websites, even websites like his school home page or other familiar websites. Though he is a predominant user of the Internet, he never had a major 'web presence.' He had a modest student profile page and was a member of a few of the major social networking websites. He communicates and collaborates online only with people whom he already knew. He consumes a lot of information from the Internet everyday ranging from assignments to online news, but he never contributes content to any website or web community. Occasionally he uses participatory media sites like digg, yelp an reditt but again, only to consume information. Bogged down by the numerous accounts which he has to remember for each and every site he uses regularly, he prefers not to register for any additional websites unless there is an absolute necessity. He openly admits that he has never once contributed information to the web and feels that there really is no reason to do so.

Linda (*Primary Persona*)
Casual User - Participates and contributes occasionally

Linda is 20 years old and is a 2nd year literature major at the University of Maryland. She is a very active person and loves to socialize with her friends. She strikes a good balance between her academic and social life. She just recently bought a MacBook and feels really happy about the switch from Windows machines. She has a busy course load of 4 classes this semester which occupies much of her time. Despite the heavy course load, she attends a ball room dance class at the university and has even signed up for the dance team and is eager to participate in competitions in the future. She is also a member of a music club which does occasional performances on the campus.

Linda is a regular user of the Internet, checking her email about once an hour and checking her facebook account every 30 minutes. She has a long list of friends in her IM list, and gets frequently interrupted by some of her friends. She uses the search toolbar built into the top of her browser because she doesn't want to spend the extra time typing the address of the search engine. She uses very few keywords in her search and feels comfortable with keyword search. She tried a few natural language systems but it didn't work well for her and she never used it again. She also uses multiple social networking sites to keep in touch with multiple friends from high school, freshmen year and her dance classes and music clubs. She likes Reditt due to its 'geeky' content, but rarely visits Digg. She participates by both voting and commenting on those websites whenever she feels compelled to do so, but its not usually very often. She often prefers to vote down the articles if she feels that it doesn't deserve to be at the top or when she feels the article is not credible. She does so because she feels that the best content should not get lost in the midst of 'junk.' She doesn't spend a lot of time on these sites, she simply votes on few articles when she feels really compelled to do so.

Lee
Hardcore User - Active participant

Lee is a first year graduate student in the Robotics Department, School of Computer Science at Carniege Mellon Univesity. He started he started taking courses this fall and is slowly getting used to the campus. Because of his unfamiliarity with the campus, he is only taking two classes to keep his coursework light in the first semester. He lives far away from campus and takes the campus shuttle to go to and from school everyday. He prefers staying a little late on campus and gets back home in the late evening. He is fond of doing interesting applications and projects.

Lee has mastered many computer languages and software, owns a Macbook Pro and carries it around with him all the time. Internet is his primary mode of communication and he talks to most of his friends on IM more than by phone. He is highly skilled at multi-tasking and uses his notebook in an efficient manner. To keep himself up to date with both the current technology and news, he has subscribed to several RSS feeds from multiple blogs and reads them constantly using a blog reader. He can seamlessly locate information online and gives credit for this ability to the latest improvements in search. He participates in a lot of social websites such as Digg, Reditt and Wikipedia. He diggs the articles he finds interesting and quality articles worthy of being voted up. He does not like digging articles which have already received many votes. He is also a constant contributer to articles on Wikipedia. Recently he stopped contributing to Wikipedia (but still consumes information) and attributes it to the recent policing issues and laments that users do not have freedom anymore. He is also a member in many technical discussion forums, takes pleasure in answering questions and enjoys the sense of community within the forums. To find information on the Internet, he typically uses Google, but he is also open to using other services as well. He is an early adoptor and enjoys experimenting with new systems if he feels they are really cool and interesting. Sometimes he uses natural language queries, especially when he thinks about it as a question, otherwise he is comfortable with keyword search. In his opinion, natural language question answering will get better in the future as keyword search has improved in the previous decade. He is waiting for a good system which can answer natural language questions.

Primary Persona - Justification

We choose not to focus on the hardcore users such as Lee, because we felt that they will likely contribute to a social website regardless of the voting interface. Additionally, we felt that some hardcore users would long for a sense of community with social websites, something we are not focusing on. We felt it would be difficult to convince the lurker (Jason) to contribute to a social website, his long record of only reading social websites and his unwillingness to register for new websites present strong obstacles. We feel that users who often use websites with a social ranking component and are willing to contribute would be ideal candidates for our system. We therefore selected Linda as the primary persona because she is a outgoing person who often reads social websites and occasionally contributes.

4.3 Experiment Design

Before testing our low-fi prototype we gave considerable thought to what we wanted to test and how best to go about it. As a group, we had decided on two possible voting interfaces that we felt were compelling enough to warrant user testing. A 2-icon voting interface using thumbs up and thumbs down (shown in Figure 1) and a 3-icon voting interface using up and down arrows and an "X" for remove (shown in Figure 2). Because all of our tasks were going to focus on these two voting interface types, we wanted to design our test with a minimum amount of learning effect. We also were interested in observing how participants would vote on question and answer pairs relative to their position to other question and answer pairs. Thus, we came up with four variables to test: 2-icon voting interface, 3-icon voting interface, question/answer position 1, question/answer position 2 (the difference was in the position of the most correct answer).

With the four people available to us for low-fi testing, and using four tasks, we used blocking to minimize participant learning effects of the system and set up our test as shown in Table 1 where UI-n is voting interface type and Pn is position of correct answers.

4.4 Low-fi Testing

4.4.1 Setup

Search involves a lot of information and the participant might be influenced by the different elements present in the interface. To keep the test focused, we wanted to test only the participant experience related to finding the answer, relevance of results and voting. We started with low-fi testing due to its simple nature and its focus on the core design and not on aesthetics. Our low-fi prototype consisted of individual results pages printed on paper for a set of questions. We prepared prototypes for the two voting interfaces we decided

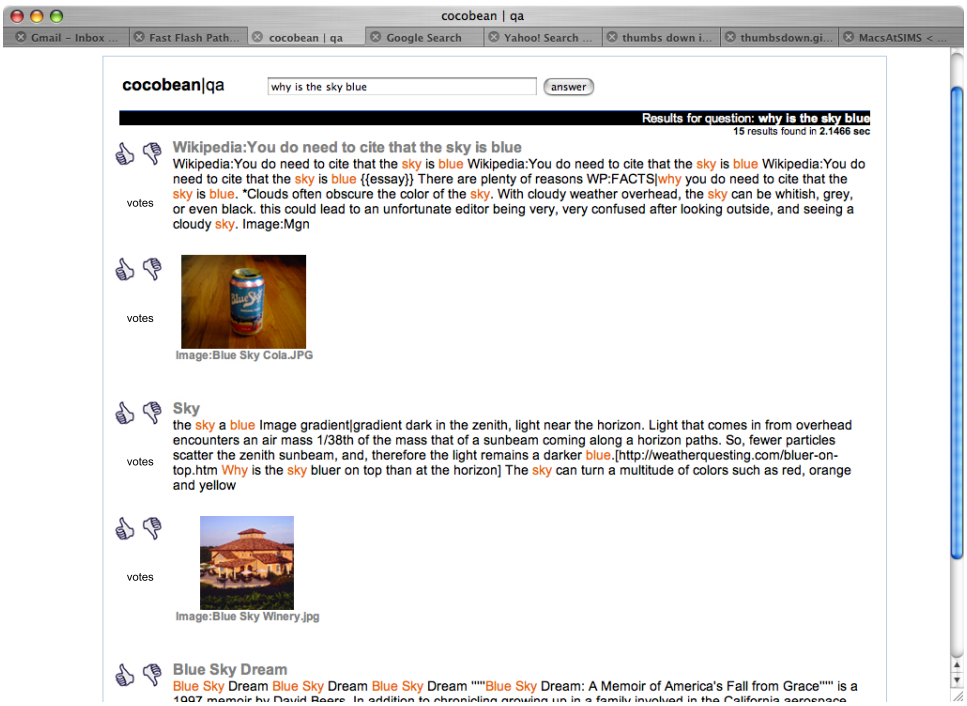

Figure 1: Thumbs up/down (good answer/bad answer) voting interface.

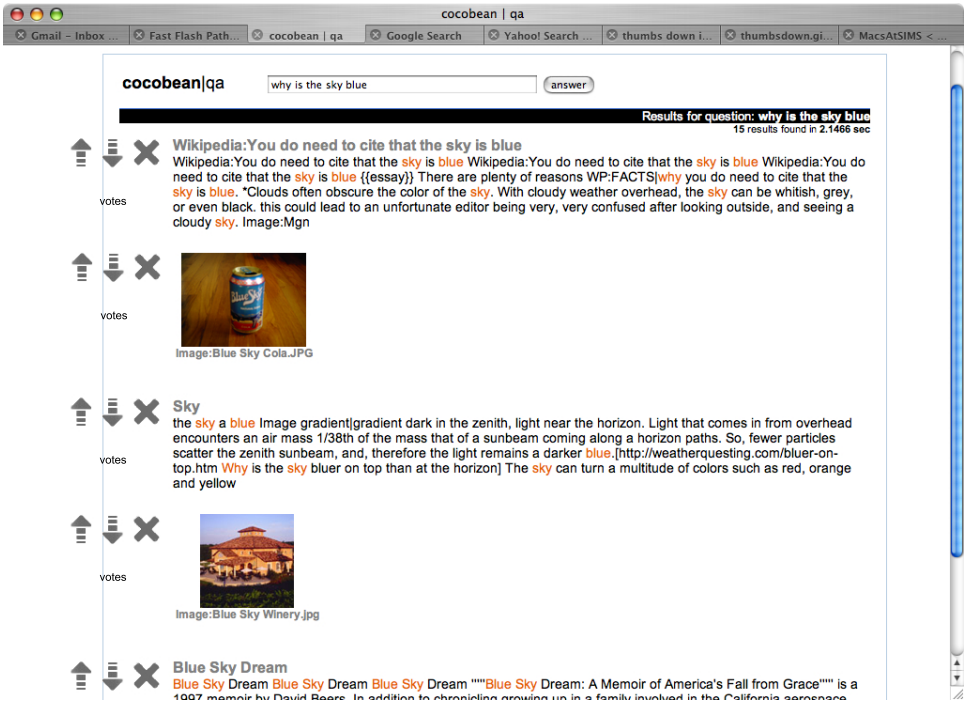

Figure 2: Up/Down/Remove voting interface.

Table 1: Experiment design matrix.

	Task 1	Task 2	Task 3	Task 4
Participant 1	UI-1, P1	UI-1, P2	UI-2, P1	UI-2, P2
Participant 2	UI-2, P2	UI-1, P1	UI-1, P2	UI-2, P1
Participant 3	UI-2, P1	UI-2, P2	UI-1, P1	UI-1, P2
Participant 4	UI-1, P2	UI-2, P1	UI-2, P2	UI-1, P1

to test. The interaction was simulated manually, using tiny icons to highlight the votes. We provided the participants with pre-selected questions due to the limitations of a low-fi prototype and to observe the same experience across different participants. We performed our low-fi testing based on the experiment designed above and conducted individual tests with different types of participants representing our personas.

4.4.2 Observations

Our user study provided us with a lot of valuable observations and some interesting feedback from the participants. Some of the important observations are described below.

- Two of the four participants initially expected direct answers for the questions asked. Only later on did they understand that our system produces a list of possible answers.
- Participants scanned the results in the first pass and when the article title is of the same type as the answer, a few of them initially assumed it was the answer. This might be due to the learning in the keyword based search engines like Google, Yahoo, etc.
- They were confused by some metadata text present in some of the results. This was leftover wiki code that was not completely removed.
- Participants trusted the answers which came from a relevant article, rather than an irrelevant article. For example, if they were asking, "Who killed Abraham Lincoln?" they trusted the answer from the article 'History of United States' more than the answer from the article 'self-surgery' though they both contained the correct answer.
- One of the participants accidentally clicked on the wrong vote icon and wanted to recast their vote.
- In the 3-icon interface, some of the participants didn't find the interface intuitive at first. They became more familiar with it through trial and error.
- Half of our participants used the remove icon in the 3-icon interface to completely vote down highly irrelevant answers. They expected that the remove icon should remove that result at least from their view of the result.
- When an answer already has votes, the participants' attention was directed to those particular answers. They tended to agree with the answer the majority voted on and did not vote on any other answers.
- When there were no prior votes on the answers, participants read the answers sequentially, at least in the first pass, until they found the best answer.
- When using the 3-icon interface one participant mentioned they imply immediate movement after voting.
- When the answer is not complete, participants would click on the article title and go to the source of the answer and read the whole article. Some participants were not sure if the article title was a link to the entire article or not.
- Most of the participants tend to vote down a result if it was wrong or did not answer their question.
- Three out of four participants said that the 2-icon interface was intuitive and matched their natural way of thinking and made the decision making really easy.

- Participants found that pictures made more sense when they are exactly relevant, but not so when they are not relevant. They immediately down voted/removed such results when they found an irrelevant picture.
- One of the participants commented that if the picture could be displayed for questions for which a picture is likely a good answer, then it might be more relevant.

4.4.3 Results

These set of observations showed us hidden difficulties with our system. We brainstormed on how to improve the system and after a thoughtful discussion derived a list of solutions keeping in mind our observations of the participants and their comments. It was evident from the study that the 2-icon (Thumbs up/down) interface was well received and the participants concluded it was more intuitive than the 3-icon (arrows) voting interface. We also debated about including the remove icon (from the 3-icon interface) along with the thumbs, but later decided not to as some of the participants did not use it and others mentioned they did not completely understand its functionality.

The test brought to our attention that we needed to clean the answer snippets to remove any leftover Wikipedia markup which was hindering the participants ability to scan the text for correct answers. It was also clear that participants were unsure whether they could click on the article names (which are links to the source article). Hence we underlined the article title and increased the size of the font slightly to communicate the link functionality explicitly. In order to help the participant locate the answer easily, we improved the highlighting in the answer snippets in two ways. First, when a query matched a rule, we only highlighted the rule in the answer snippet (instead of all the keywords). If the query did not match a rule, we resorted to keyword highlighting. The second improvement to the highlighting of answer snippets we added was 'potential answer' highlighting. This feature highlights the nouns and numbers in close proximity to the matched rule in the answer with a different color than that of the matched phrases or keywords.

Unlike other social ranking websites, we wanted to provide participants with the ability to toggle their votes (if they made a mistake in voting) and for participants to be able to cancel their vote altogether. However, after further investigation of this functionality, we decided not to provide the option at this time as it posed security issues with our current system architecture.

The functionality of this system is a little different than most mainstream search engines and some participants may not be familiar with this concept. To alleviate this problem, we decided to include a brief description of the functionality of the system that is accessible from all pages. We also found that some participants had difficulties in getting to the home page from the search results page, so we adjusted the color of the 'cocobean' logo to give participants an affordance that the logo was interactive.

The user interface design process helped us to considerably improve the system from our initial design as shown in Figure 3 to our current design as shown in Figure 4. The pictures below illustrate the evolution of our interface.

5 System Architecture

Figure 5 illustrates what happens in our system when a user asks a question. Questions are sent to two modules, a question classifier and a question transformation module. The question classifier identifies the type of answer the question is referring to and sends the result on to the re-ranking module. The question transformation module tranforms the question into multiple answer templates and submits all of them to lucene as queries, the results are sent to the re-ranking module. The re-ranking module uses several features some of which it fetches from our database. The re-ranking module then computes a score for each answer and re-orders the results. These results are then cleaned, the answer highlighted and finally displayed to the user. Once a user has results displayed, voting on an answer up or down will send a request to the database to add the vote and an updated count is returned to the user.

Figure 3: Previous design.

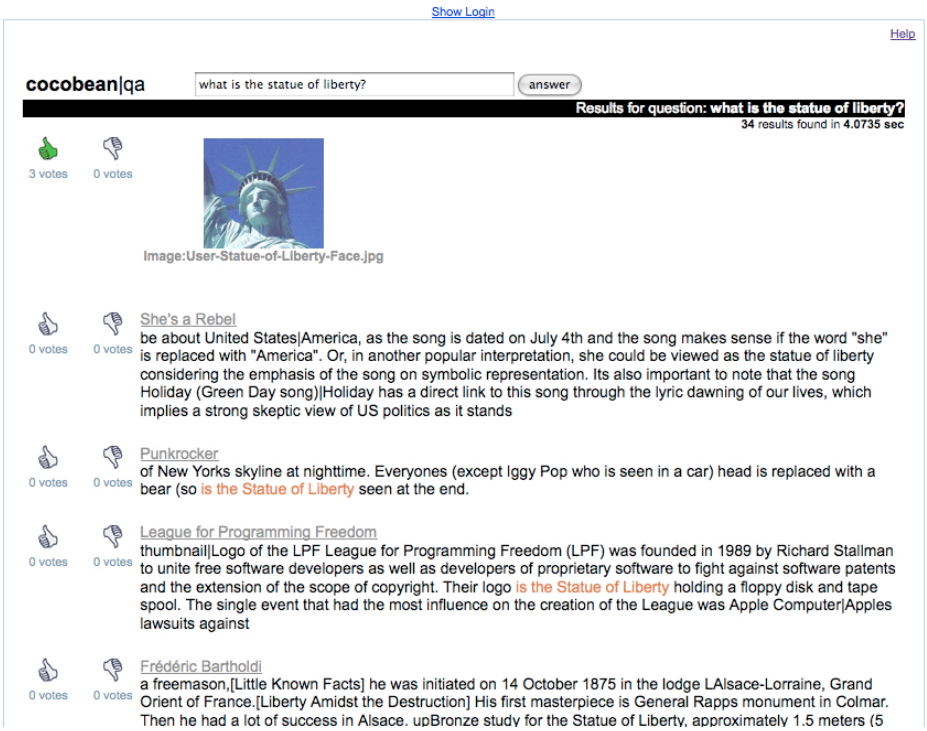

Figure 4: Current design (after user study).

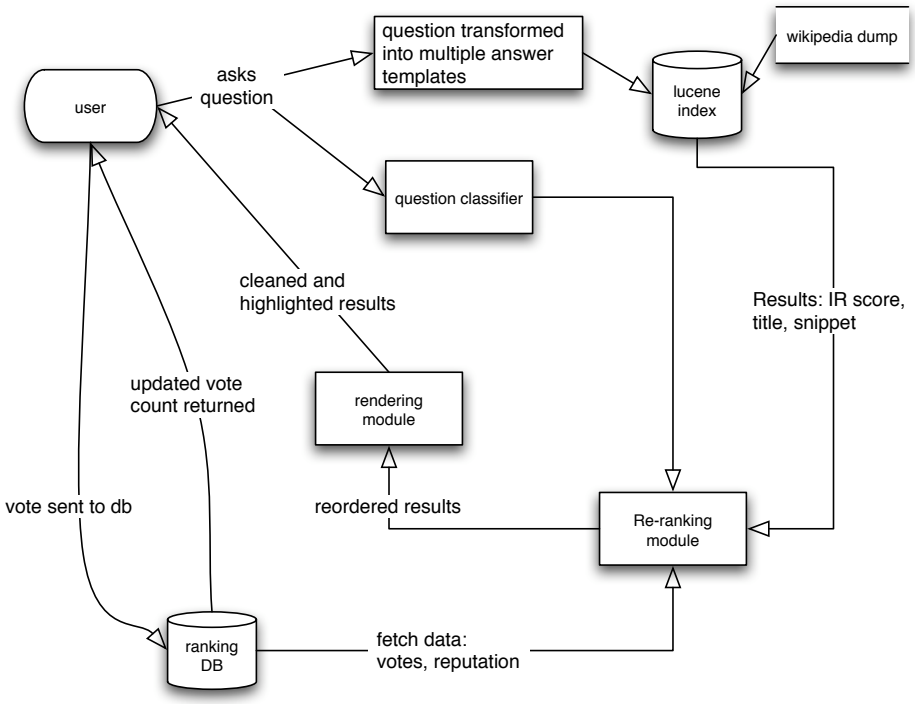

Figure 5: Process flow of system (high level).

5.1 Knowledge Base

5.1.1 Wikipedia

We downloaded the Wikipedia dump, which is a large XML file that contains a snapshot of all the Wikipedia articles. We used an XML parser to parse the XML and extract the title and text of each article. The title and text of each article were indexed into our IR engine.

The system retrieved the category of each Wikipedia article, such as person, location, entity, etc. The category information of the articles could help improve the ranking of the search results, which will be explained in Section 5.8. We wanted to get the higher level category information from a category hierarchy. However, we found Wikipedia's category structure too complicated for the scope of our project. Each article could belong to multiple category hierarchies, and there's no easy way to retrieve the category information.

5.1.2 Freebase

We decided to utilize Freebase to get Wikipedia article category information. Freebase is an open and shared database that provides structured metadata for Wikipedia and other online sources. It contains cleaner category information - a single category hierarchy for each Wikipedia article. For example, the category of an article about New York City is "/location/citytown". The system looked up the database, retrieved the root level category "location" and indexed it along with title and text.

5.2 Information retrieval engine

We implemented our search engine using Lucene, an open source information retrieval library. We used Lucene to index Wikipedia articles to a index file. Because the system needs to do phrase search that matches every word (including stop words) in the transformed query string, we indexed every word from the text. More details about query transformation is described in Section 5.4.

The title and text of the Wikipedia articles were indexed. To gain more precision of the search result, we gave the title three times more weight than the text because the Wikipedia title usually represents the theme of the article.

5.3 Database

The database stores user information (user name, reputation, etc), transformation rules, votes, questions and answers that were voted on. For more details please see to the ER diagram in Figure 10.

5.4 Query Transformation

To retrieve answers from the Wikipedia corpus, our system transforms input questions into answer strings that are likely to be found in source documents. The answers may appear in the corpus in various forms. For instance, the two sentences "The Statue of Liberty is located on Liberty Island" and "The Statue of Liberty, and its location on Liberty Island, appear in scores of posters, pictures, motion pictures, and books" both contain the location information of the Statue of Liberty. In order to retrieve these various forms of answers, the system generates multiple transformed queries based on a set of transformation rules. As shown in Table 5, we have 60 hand-coded rules used for query transformation. The transformed queries are string-based manipulation enhanced by the part-of-speech of the question terms[5]. To make the rules reusable, we separated the rules from the code and stored them in the database.

5.4.1 String-based transformation

The system first recognizes the type of question based on its interrogative term, such as "what", "where", "who". Then it applies the rules associated with the type of question to the original queries. For example, the question "Where is the Statue of Liberty" is recognized as a "location" question because of the "where" term. The entity of this question "the Statue of Liberty" is then recognized and used for the transformation. We use the string-based manipulation approach to transform the question into the following strings:

1. "the Statue of Liberty is in"
2. "the Statue of Liberty is located in"
3. "the Statue of Liberty is situated in"
4. "the Statue of Liberty lies on"
5. "the location of the Status of Liberty"

These transformed queries are submitted to the search engine. We use Lucene's phrase search function so that the documents that contain these strings will be retrieved. The matched sentences, such as "the Statue of Liberty is located in New York City" are returned as the search result.

The system deals with other types of questions such as "who", "what", "when"and "how many" that require manually added rules. The transformation of "who" and "what" questions removes the interrogative word and then places the auxiliary word ("is", "was", etc.) in the beginning and end of the string. For example, the question "Who is the mayor of New York" is transformed as "the mayor of New York is" and "is the Mayor of New York". For the "when" questions, the system removes the wh-phrase, i.e., the interrogative word followed by an auxiliary word, and then appends "at" or "in" to the query to form the answer strings with time information. For instance, the question "When was George Washington born" will be transformed into "George Washington was born in". For the "how many" questions, the query string is added "the number of" to beginning. For example, the question "how many states are there in the US" will be transformed into "the number of states in the US".

5.4.2 Part-of-speech transformations

In addition to the simple string-based manipulation, we empowered the transformation based on part-of-speech tagging on the question terms [7]. Using part-of speech information could make the transformations

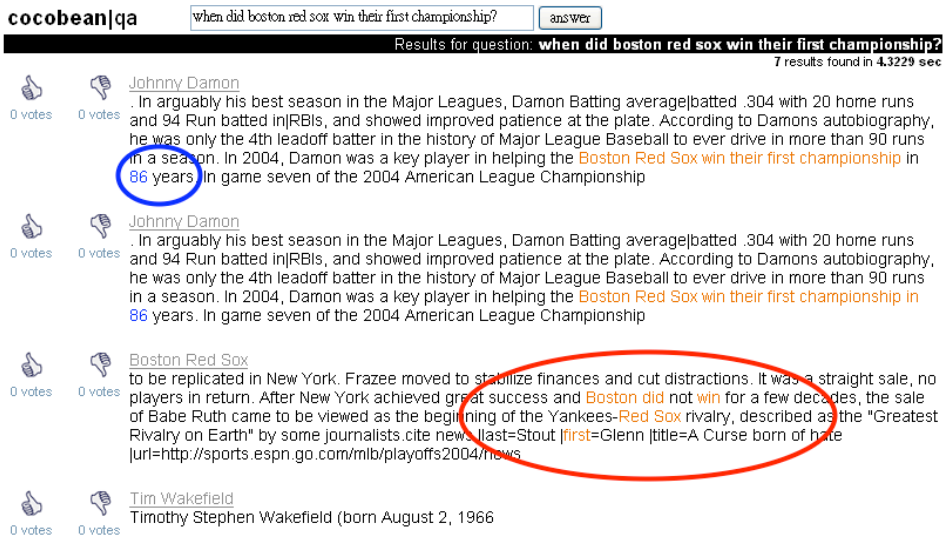

Figure 6: The text circled in blue (smaller circle) shows the highlighted potential answers generated from the process described in Section 5.7. The text circled in red (larger circle) shows keyword highlighting using the backoff rule described in Section 5.6

more flexible and could avoid the chore of exhaustively adding rules based on the question terms. For instance, the question "Who made the Statue of Liberty disappear" contains a verb "made" after "who". This corresponds to the rule "who + verb", which removes "who" to transform the query as "made the Statue of Liberty disappear".

Another use of part-of-speech tagging is to detect the auxiliary verbs like "do", "does", and "did". The system will remove the interrogative word and the following auxiliary verb. The tense of the following verbs in the sentence will be changed according to the auxiliary verb. For example, the question "When did the Cubs last win the World Series" will be transformed to "the Cubs last won the World Series in".

5.5 Title query rule

In addition to searching through the text, the system also searches the title of the article. The Wikipedia titles are useful as they clearly indicate what the article is about, such as a person, an event, etc. The title query rule extracts the subject phrase in the question and forms it as the transformed query. Then the system retrieves the documents with their titles matching the transformed query. For example, the question "What is cosine similarity" will be transformed into "cosine similarity". The articles with titles similar to "cosine similarity" will be retrieved.

5.6 Backoff rule

Because we use the phrase search to match the transformed queries with the sentences in the documents, it is possible that there are no such forms of answer strings in the corpus. This is because there may not be enough variation of answer forms in the Wikipedia corpus as compared to the Web. To avoid the problem of having no results returned, the system has a default backoff rule that retains the non-stop words in the original query to form an AND query. This rule is particularly necessary given the fact we don't have a really large corpus and much information redundancy. Figure 6 demonstrates the answers generated by the backoff rule.

5.7 Answer Highlighting

Given a query, the system returns a list of answers, each consisting of a title of the article and the snippet of the text that contains the potential answers. Initially the system highlighted all the keywords from the query

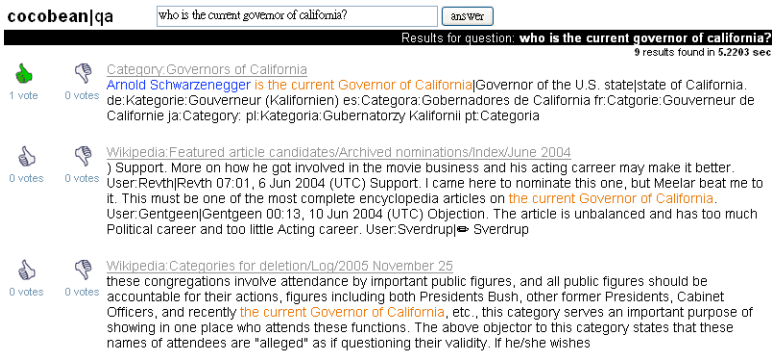

Figure 7: Answer highlighting: the answer "Arnold Schwarzenegger" is highlighted in the returned snippets.

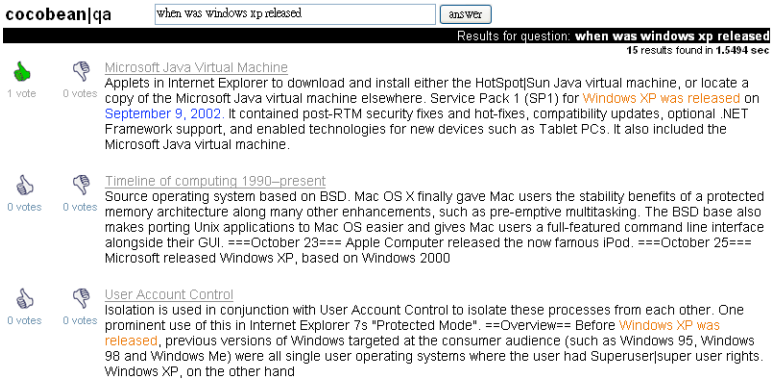

Figure 8: Answer highlighting: the answer "September 9, 2002" is highlighted in the returned snippets.

string, just like mainstream search engines do. However, we want to locate the answers words, which usually exist around the transformed query. Simply highlighting the keywords could be distracting when many keywords densely exist in the snippet. Therefore our system only highlights the transformed query sentences that appear in the text, not every key word. Only the backoff queries would highlight each keyword in the snippets.

In addition to highlighting the transformed queries, the system highlights the potential answer words around the queries in the text with a different color. The transformation rules are set to capture the answers by highlighting a few words to the left or right of the queries. The system will highlight the surrounding words with specific surface string features. The "Where" and "Who" questions will look for capitalization, and the "When" and "How many" questions will capture digits and date/time formats. For instance, the question "Who is the current Governor of California" will highlight three words with capitalization to the left of the transformed query "is the current Governor of California." The question "When was windows XP released" will highlight five words with digits or date format to the right of the transformed query "windows XP was realeased in." Figure 7 and Figure 8 show the results of the above two questions.

5.8 Question Classification

In a keyword search there are numerous answers returned and the user has the burden of scanning through results to identify the potential good answer. Furthermore, the results might not reflect the domain the query was intended for. On the other hand, a question answering system can answer the question directly and help the answers to be more relevant, as the question provides more clues than a simple query. Question classification helps us identify the type of the potential answer. Especially in fact based questions, where the user is looking for a specific answer, keyword search is not very helpful. For example, if the user asks, "When was the first space mission to the moon launched?", typically they might issue a query "first space mission moon year" which is not guaranteed to answer the question properly. With the help of question classification, our system can identify that the question is asking for a "date" and answer the question appropriately. We

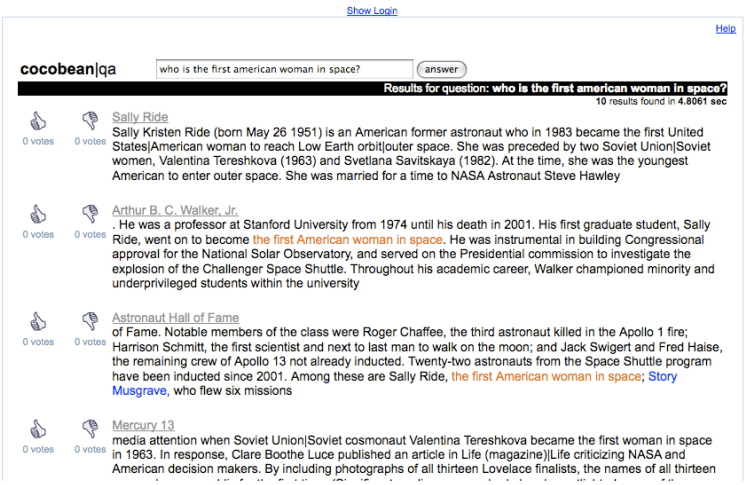

Figure 9: Results for a search query for the first american woman in space.

designed a question classifier as described below.

5.8.1 Classifier

Training and Testing
We used the question classification database made available by Li and Roth [9]. The dataset has 5500 questions with manually labeled question types. The question types have a coarse and a fine layer. Currently, we are only using the coarse layer for our categorization such as ABBREV, ENTITY, DESCRIPTION, HUMAN, LOCATION and NUMERIC and OTHERS. We also remapped one of the finer categories 'SPORTS' under the 'ENTITY' as a top level category. The OTHERS class is to represent data from other categories or if the classifier is not sure about the prediction of the class. The test dataset comprises of 500 labeled data from TREC10 QA track.

For question classification, support vector machines outperform all the other classifiers with very simple features [14]. The two major features used in our classification are bag-of-words and bag-of-bigrams. We used LIBSVM [5] to build our classification model. The classifier was trained with the 5500 questions from the training data set and the cross validation accuracy with 10 folds is 85.785%. The trained classifier is then tested with the test data which comprises of 500 questions. There were 52 misclassified instances out of the 500 total test questions, giving us an error rate of 10.4%. We believe that the performance of the classifier could be improved. The training data is limited and the coverage of different types of quesitons is also inadequate. Additional resources would definitely help to boost the performance of the system. For example, adding question type 'Image' to the system would help our system in matching images to relevant questions. Techniques such as active learning might help us improve the training and we would like to explore more in that direction.

Application
In our system, the user question is passed to the question classification module, which in turn predicts the type of answer into one of the seven classes. In addition, it also returns a probability distribution of the classes to the re-ranking module. This is used as one of the primary features in the re-ranking module along with other features. Initially we tried using the classes to influence the results directly, but later we found that using it as one of the features in the re-ranking process would be helpful.

5.8.2 Answer Filtering

For any given query transformation a few results are returned based on the rules applicable to that particular type of question. In many cases the transformation rule selected would result in retrieval of results which are irrelevant to the question. Answer filtering can help to prevent such irrelevant answers. In Figure 9, the

question "Who is the first American woman in space" is asking for a person's name and hence reduced the ranking weight for any other type of results.

To locate the answer we are using Freebase type tags to determine their type. We used an indicator feature when the question type and answer type matched in our machine learned ranking module.

5.9 User Voting

Voting is one of the primary aspects of the system that uses user's votes to improve the ranking of the results. User voting captures subtle details which might be really hard for a natural language system. There can also be disambiguation issues when answering a question. But a community of users can easily separate the good and bad answers. Community voting has already proven to be highly successful in systems like Digg, Reddit and others.

Our system allows an answer to be voted as 'good' or 'bad'. The user votes along with the question and answer pair and they are both recorded in the database when a user votes on an answer. These votes are used to train our re-ranking model as described in the section below. The system in turn learns from the user votes and the learning is propagated to the future questions of same type.

The system provides the option of user accounts, to track the reputation of the user based on votes. If a user consistently votes in the same manner for question answer pairs that the majority has also voted for, his reputation score increases, which in turn gives higher weight for his future votes. Conversely a vote by an user with lower reputation would be give a lower weight. The reputation score is passed as a feature to our ranking module. The votes are limited to one vote (either good or bad) per answer for a user, to prevent bombing/spamming of the system. Guests (users who are not logged in) are tracked as individual users based on their IP address.

5.10 Ranking

For each potential answer, we keep track of the type of question and which type of transformation was used to generate it. Voting icons corresponding to each potential answer are shown on the result page allowing users to instantly update the ranking database.

User interaction helps the system to tune itself by promoting transformations that work well for a certain type of question. The motivation behind the system is that unlike traditional search, users of our system will be able to easily recognize that an answer is correct by looking at the snippet. Since users are looking for a specific answer, it should be easy to recognize it and to vote the result up. Conversely, users can recognize bad answers and filter them out, allowing the system to learn appropriate negative rankings. We believe that this setup will encourage more explicit user feedback than with traditional search systems.

At the moment, our system allows users to influence results by voting on a question and transformation pair once per unique IP address and username. The transformations are then aggregated over all users for a particular type of question. For instance, a "who is" question classified as regarding a person entity would be grouped with similar questions about other people. Voting on any similar question automatically influences the order that the system uses the transformation. This type of social ranking for question and transformation pairs allows our system to automatically propagate answers chosen from successful transformations. Since all similar question types are affected, even never before seen questions can have the most probable answer propagated up.

5.10.1 Machine Learned Ranking

Our system generates various information about questions and answers on our system, including relevancy scores from Lucene and classification tags. In addition, various usage metrics about the system is collected and derived features such as user reputations are generated. The goal of using a machine learned ranking approach in our system is to intelligently predict the relevancy of answers given a question and its set of candidate answers. By training a model using the various features described in Section 5.10.2, we are able

to assign importance weights to each of the features and assign a score to each question answer pair. The order of the results is then ordered by these scores and presented to the user.

Since relevancy is inherently a numeric attribute rather than categorical, we modeled our ranking module as a regression task as opposed to classification. The training of our model was done using WEKA, an open source data mining and machine learning software package. While there were numerous regression algorithms available in WEKA, we focused mainly on Boosted Decision Trees (BDT) and a variant of Support Vector Machines(SVM) for regression. Due to the complexity of implementing a BDT applier and virtually no difference in performance between BDT and SVM, we decided to use the SVM regression model for ranking in our system.

As described in Section 6.1, data was collected using Mechanical Turk from Amazon.

5.10.2 Features

Images
The image feature is a binary attribute that checks if a particular result is a image.

Votes up
The numeric votes up feature is a raw count of how many up votes users gave to a particular answer.

Votes down
The numeric votes down feature is a raw count of how many users voted down on a particular answer.

Reputation of users
The reputation of a user is an implicit feature that is a combination of how many times the user voted with the majority and how many votes the user has placed on the system. User reputations are recomputed on a nightly basis. The reputation score is used to weigh the up and down votes features. The formula for computing a user's reputation is:

$$0.8 * \frac{\text{user's votes in the majority}}{\text{user's total votes}} + 0.2 * \log(\frac{\text{user's total votes}}{\text{average votes per user}} + 1)$$

As shown in the formula above, the reputation score of a user ranges between 0 and 1. The majority of the weight goes toward how many times the user votes with the majority. A current limitation of this reputation formula is that user votes are considered to be in the majority even though it is the sole vote. A possible fix for this would be to have a threshold on the number of votes before a question is considered in the computation of the reputation score (e.g. a question must have at least 5 votes and the user must be in the majority for it to count). Another approach can be to normalize the number of votes the user has in the majority by the average number of votes for questions on our system, effectively increasing the contribution a vote in the majority has as the question gets more votes. However, we believe that gaming of reputation systems are a constant threat regardless of the formula but our exposure is limited because user reputations are hidden on our system.

Lucene Relevancy
The lucene relevancy feature is the tf*idf ranking of the answer returned by lucene given a query.

Question type
The question type is returned by a multi-class SVM classifier.

Answer type
The answer type is the type tagged by Freebase.

Question type == Answer type
This binary feature is on when the classified question type matches the type of the answer article as tagged

by freebase.

Rule Type
The rule type features are binary features for each rule type that was used to find the answer.

5.10.3 Feature Importance

Using a Chi-squared evaluation of feature importance, the ranked attributes are shown in Table 2.

Table 2: Feature rankings using χ^2 evaluation of importance.

76.567	Rule
31.498	Up_votes
16.961	isImage
6.74	sameType
0	relScore
0	down_votes

From evaluating feature importance, we can see that the rule feature was by far the most important attribute. We believe that this made sense since the type of rule had the most significance in determining the snippet returned for each answer candidate. The second most important feature turned out to be the number of up votes. We found this to be in line with what we predicted assuming that vote quality of the users were sufficient. The third most significant feature was the binary feature identifying whether an answer is an image or not. We are not exactly sure why this feature was important but it does appear that the image tag was useful for certain types of rules such as "who is" or "what is" type questions.
The other important feature indicated whether the classified question is of the same type as the article that the answer came from. We believe that this makes sense since certain question types such as "What sport does Kobe Bryant play", is looking for an answer that is classified as sport rather than other categories such as location or people.

It is worth noting that relevancy scores from lucene and down votes did not have much of an impact. We believe this was the case because rule usage was a better indicator of answer quality. Since whether a rule that matches perfectly with a snippet or not is mostly independent of the lucene relevancy score, this result was expected.

The low importance of down votes surprised us but we believe that the result was mostly due to the larger number of down votes, leading to the diminished distinguishing factor of the attribute.

5.10.4 Training

To determine the ranking of our results, we relied on a machine learned ranking approach. Given a training set of vote data collected on mechanical turk as well as features that we added, we trained SVM and Boosted Decision Tree models for regression. Once we had a model, it predicted the ranking of an answer between 1 and 5 relying only on the features that we gave it. Since both the SVM and BDT models performed relatively the same with 70% accuracies using 10-fold cross validation when we treated the task as multi-class classification, we opted to use the SVM because online usage of the model was easier to implement. We wish to clarify that although we did a dry run training a classification model to gauge performance, we focused the majority of our time developing regression models and eventually implemented a regression model in our system.

6 Experiments

6.1 Dataset

6.1.1 Gold Standard

In order to determine the weights of each of the features in our system, we used WEKA to train an SVM model for regression based on the set of features as descried in Section 5.10.2. To do this, we needed accurately labeled question and answer pairs, which would become our gold standard. The four of us each separately wrote queries that included question types such as Who, What, When and Where, and How. We then input these queries into our system and recorded the results as well as the transformation rule that was used to generate the candidate answers. The next step was to manually rate the relevance of each candidate answer for a given query on a 5 point Likert scale. In total we had 38 gold standard questions, each with a varying number of candidate answers; the total number of question and answer pairs in our gold standard was 237.

6.1.2 mTurk Setup

Because we wanted a large number of users to evaluate our question and answer pairs, we chose to use Amazons Mechanical Turk for evaluations. For a small fee users of Mechancial Turk will perform Human Intelligence Tasks (HITs), tasks that only humans can perform. Each HIT consisted of one question and all of its candidate answers. We asked that users vote on the answers based on the following criteria; "If you feel an answer is correct and/or should be higher up on the list below, please choose "Vote Up" for that answer (even if it is in the first position). If you feel an answer is partially or somewhat correct but should be further down on the list, please choose "Vote Down" for that answer. If you feel an answer is completely incorrect and should not even be present on the list, please choose "Vote Wrong" for that answer. For each question, you may vote on as few or as many answers as you like."

To allow for the possibility that a user may not know the correct answer to the question, we included a note giving them permission to use any source they wish to verify any facts they wish before voting.

We assigned each question HIT (again 38 questions in total) to 10 Mechanical Turk users, which gave us a large number of voting overlap. We also kept track of each Mechanical Turk users ID in order to use reputation as a feature (which we will discuss later on in the paper).

6.2 Ranking Evaluation

6.2.1 DCG

The metric that we used to evaluate relevancy of our system was the discounted cumulative gain (DCG). The main idea behind DCG is that the results set, as well as its ordering, contributes a certain amount of gain. Gain is tiered depending on the relevancy of the answer (e.g. a perfect answer contributes a 5 while a excellent result might contribute a 4). The gains are also discounted depending on position. A perfect answer contributes a discounted gain if it is in position 2 as opposed to position 1. Finally, this metric is cumulative as it is the sum of discounted gains of all results. For our evaluation, we measured the DCG10, the DCG of the top 10 results.

$$NDCG = Z * \sum_{i=1}^{k} score(i) * \frac{\log 2}{\log(i+1)}$$

In the above equation, we are computing the Normalized DCG of the top k answers, we used $k = 10$ for our experiments. The ordering of the k answers is determined by the sorted order, from highest to lowest, given the predicted relevancy of the answers from our SVM model. Given the ordering, the $score()$ function returns the gain of the answer where the gain is 10 for answers rated 5 in the gold standard, 7 for answers rated 4, 3 for answers rated 3, 0.5 for answers rated 2, and 0 for answers rated 1. The discounting factor is accomplished with the log function on the right which decreases from 1 when $i = 1$. Z is the normalization factor chosen as the maximum DCG obtained using a perfect ranking, where the predicted scores are equal to the golden set.

6.2.2 Evaluation

We evaluated our ranking function based on gain to our DCG function between our default ranking from lucene and the DCG when using our trained model for reranking. As a sanity check, using reranking on the training set nearly doubled the DCG over the default ranking system that we had. The DCG was also significantly better than randomly ranked results. Our DCG of 12.79 closely approaches the absolute maximum of 13.68 for a perfect ranking on the set.

For a more reasonable assessment of system performance, we evaluated the trained model on a 103 instance held out set. The test set ranking had a DCG of 15.85, significantly better than the default ranking of 10.95 or random ranking of 11.28. With a maximum DCG of 17.38 for the test set, we find the results of the test to be extremely encouraging.

Table 3: DCG10 on training set.

	DCG	NDCG
Default	6.93	0.51
Random	8.14	0.60
Best Possible	13.68	1.00
Reranking	12.79	0.93

Table 4: DCG10 on test set.

	DCG	NDCG
Default	10.95	0.63
Random	11.28	0.65
Best Possible	17.38	1.00
Model only w/o votes	13.67	0.79
Reranking after votes	15.85	0.91

7 Discussion

7.1 Limitations

The system supports several types of questions, such as Who, Where, What, When, and How. We found some types of questions usually return good results, but there are other types that at times, returned less than satisfactory results. For the Where, Who and When questions, our system highlights the potential answers more precisely than What and How questions, this is because some surface text features like capitalization and digits are usually found around the answer templates. For some What questions, people intend to ask for a description, which is difficult for our fact-based question answering system to handle. For the How/adjective question, our query transformation would limit the size of the returned answer set because the system is not able to convert the adjective to its attribute nouns.

One area of concern is whether users would actually provide explicit feedback by voting on the results. Research has shown that explicit feedback is hard to come by, as participation rates are usually low. However, we believe that the unique aspect of our system where we provide users with the answer as well as the immediate context allows users to easily identify whether an answer is correct or not. We believe that being able to easily identify the answer encourages users to vote. Furthermore, our AJAX voting interface also makes it easier to vote by immediately registering votes without reloading the page.

Using manually created transformations as described in the AskMSR paper worked well when dealing the entire web as a corpus. This duplication and redundancy of information on the web makes it easier to find answers that correspond to a particular manually added transformation. A disadvantage of our system and choice of Wikipedia as a dataset is that there is usually only one instance where a fact is stated, making our manually added transformations a hit or miss. We hope to resolve this issue by using more natural language techniques and to move away from using manual transformations.

String-based manipulation of manual transformations is straightforward and it is easy to add new rules. However, its simplicity limits itself to capture a broad range of answer representation. It is sometimes meaningless to reorder the words in the questions because these words might not appear again in the answers. For example, the query transformation is not able to do well for the "how + adjective" question as the adjective does not reappear in the answer sentences. For the question "How tall is Mount Everest?", we may find the answer in the form, "The height of Mount Everest is . . . ', but we won't see "tall" again in the answer. In order to overcome this limitation, we need to know the attribute nouns of the adjectives in the questions and place the attribute nouns in the transformed queries. Some external NLP tools, such as WordNet, would be helpful to solve this problem.

Highlighting answers around the transformed queries may work well if the queries exist exactly in the text. However, the simple approach that captures capitalized words and numbers around the queries does not work if the sentence structure of the answer is more complicated. For example, our system is not able to consistently extract the answer from sentences such as, "Who is the first American in space?" because it would look for the capitalized words before the transformed query "is the first American in space," which are not always the correct answer. In the answer sentence, "Alan Shepard was an American astronaut, the first American in space" the word "American" will be highlighted instead of "Alan Shepard." Also, the system may fail to extract the answers using the surface string feature, such as word capitalization and digits. It will be helpful if the system could get the part-of-speech information so as to capture the nouns and noun phrases in the surrounding text.

7.2 Lessons Learned

Throughout the course of this project we learned a lot from both the technical aspect and the social ranking part of the system. We found that a well designed user interface would significantly improve the usability of our system. Based on feedback from the user study, we found that users are more comfortable with voting interfaces that reflect their natural way of thinking. At times it would be surprising that what appears to be very simple to the designer, might not occur to the user in the same manner.

Some users consider creating an account or logging in to a search system as high overhead and are reluctant to do so. Taking this into consideration, we allow guests to vote on answers and compute their reputation by their IP address. We have learned that the combination of a nicely designed UI and well thought out back-end is the key to a successful system such as ours.

From the techinal side of things, we learned that question answering based on answer templates which are based on rules returns highly relevant answers, but it also highly depends on the redundancy and the type of data available. Keyword search as a back-off rule works very well in combination with a rule based system and returns fairly relevant results in the event that no rules match.

We also found that interleaving text and image results together works better when the image is highly relevant, but not the other way around. Users have a high sensivity to the relevance of images, perhaps more so than with text.

Data backup is important! Unfortunately we ignored this part, and we encountered some problems during the system implementation. First, we mistakenly dropped the whole database. The entire schema and data were gone in a blink. Because we didnt have a backup, we had to re-create everything (schema, test data) from scratch again. The second problem we faced was a hard disk crashed while we were indexing the Wikipedia dump. All the index files were lost, which we also didnt have any backup for. In the end we bought a new hard drive and re-indexed the entire Wikipedia dump, which cost us a lot of time. In addition to these two

problems, our server was located in one of our members office, which was not air-conditioned. So we were always wary of another data crash again and realized the importance of the data backup plan. From then on we backed-up our database and files constantly to other machines.

8 Future Work

As question classification determines the semantic type of the question which is used in filtering the answers, the precision of this module is highly important. To improve the question classification further and make it more precise we would like to include finer classes as proposed in Li and Roth[9]. This would give more coverage and improve the answer filtering. We would also like to use an active learning based approach to acquire more training data for our question classification system. We also want to implement an intelligent answer extraction algorithm to help users identify the correct answer easily.

We have found the Wikipedia dataset contains limited variations of answer forms. For the next phase of the the project we plan to extend our corpus to the web, leveraging its nature of information redundancy to retrieve more good answers.

As our system functionality is fairly different from most existing question answering systems, conducting an elaborate usability study with more participants could help us further refine our interface.

References

[1] Yahoo! answers, http://answers.yahoo.com.

[2] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–26, New York, NY, USA, 2006. ACM Press.

[3] E. Agichtein, S. Lawrence, and L. Gravano. Learning search engine specific query transformations for question answering. In *World Wide Web*, pages 169–178, 2001.

[4] I. Androutsopoulos, G. Ritchie, and P. Thanisch. Natural language interfaces to databases–an introduction. *Journal of Language Engineering*, 1(1):29–81, 1995.

[5] C. C. Chang and C. J. Lin. *LIBSVM: a library for support vector machines*, 2001.

[6] S. Dumais, M. Banko, E. Brill, J. Lin, and A. Ng. Web question answering: is more always better? In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 291–298, New York, NY, USA, 2002. ACM Press.

[7] S. M. Harabagiu, M. A. Paşca, and S. J. Maiorano. Experiments with open-domain textual question answering. In *Proceedings of the 18th conference on Computational linguistics*, pages 292–298, Morristown, NJ, USA, 2000. Association for Computational Linguistics.

[8] C. C. T. Kwok, O. Etzioni, and D. S. Weld. Scaling question answering to the web. In *World Wide Web*, pages 150–161, 2001.

[9] X. Li and D. Roth. Learning question classifiers, 2002.

[10] D. Ravichandran and E. Hovy. Learning surface text patterns for a question answering system. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 41–47, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[11] D. Roussinov and J. Robles. Learning patterns to answer open domain questions on the web. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 500–501, New York, NY, USA, 2004. ACM.

[12] J. M. Zelle and R. J. Mooney. Learning to parse database queries using inductive logic programming. In *Proceedings of the 14th National Conference on Artificial Intelligence*, pages 1050–1055, Portland, OR, 1996. AAAI Press/MIT Press.

[13] L. S. Zettlemoyer and M. Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the 21th Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 658–66, Arlington, Virginia, 2005. AUAI Press.

[14] D. Zhang and W. S. Lee. Question classification using support vector machines. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 26–32, New York, NY, USA, 2003. ACM.

Table 5: Rules used for query transformation.

Original question form	**Transformed question form**	**Example**
where <be> <subject>	<subject>	Where is UC Berkeley -> UC Berkeley
where <be> <subject>	<subject> <be>	Where is UC Berkeley -> UC Berkeley is
where <be> <subject>	<subject> <be> in	Where is UC Berkeley -> UC Berkeley is in
where <be> <subject>	<subject> <be> at	Where is UC Berkeley -> UC Berkeley is at
where <be> <subject>	<subject> <be> near	Where is UC Berkeley -> UC Berkeley is near
where <be> <subject>	<subject> <be> located in	Where is UC Berkeley -> UC Berkeley is located in
where <be> <subject>	<subject> <be> situated in	Where is UC Berkeley -> UC Berkeley is situated in
where <be> the <subject>	the <subject>	Where is the Space Needle -> the Space Needle
where <be> the <subject>	the <subject> <be>	Where is the Space Needle -> the Space Needle is
where <be> the <subject>	the <subject> <be> in	Where is the Space Needle -> the Space Needle is in
where <be> the <subject>	the <subject> <be> at	Where is the Space Needle -> the Space Needle is at
where <be> the <subject>	the <subject> <be> near	Where is the Space Needle -> the Space Needle is near
where <be> the <subject>	the <subject> <be> located in	Where is the Space Needle -> the Space Needle is located in
where <be> the <subject>	the <subject> <be> situated in	Where is the Space Needle -> the Space Needle is situated in
where <be> <subject> <vbd>	<subject> <be>	Where is UC Berkeley located -> UC Berkeley is
where <be> <subject> <vbd>	<subject> <be> in	Where is UC Berkeley located -> UC Berkeley is in
where <be> <subject> <vbd>	<subject> <be> near	Where is UC Berkeley located -> UC Berkeley is near
where <be> <subject> <vbd>	<subject> <be> <vbd> in	Where is UC Berkeley located -> UC Berkeley is located in
where <be> <subject> <vbd>	<subject> <be> <vbd> at	Where is UC Berkeley located -> UC Berkeley is located at
where did <subject> <vb>	<subject> <vb>	Where did the ancient Greek Olympics take place -> the ancient Greek Olympics take place
where did <subject> <vb>	<subject> <vb> in	Where did the ancient Greek Olympics take place -> the ancient Greek Olympics take place in
where did <subject> <vb>	<subject> <vb> at	Where did the ancient Greek Olympics take place -> the ancient Greek Olympics take place at
where did <subject> <vb>	<subject> <vb> near	Where did the ancient Greek Olympics take place -> the ancient Greek Olympics take place near
where did <subject> <vb>	<subject> <vbd>	Where did the ancient Greek Olympics take place -> the ancient Greek Olympics took place near
where did <subject> <vb>	<subject> <vbd> in	Where did the ancient Greek Olympics take place -> the ancient Greek Olympics took place in
where did <subject> <vb>	<subject> <vbd> at	Where did the ancient Greek Olympics take place -> the ancient Greek Olympics took place at
where did <subject> <vb>	<subject> <vbd> near	Where did the ancient Greek Olympics take place -> the ancient Greek Olympics took place near
Who <be> <subject>	<subject>	Who is George Bush -> George Bush
Who <be> <subject>	<be> <subject>	Who is George Bush -> Is George Bush
Who <be> <subject>	<subject> <be>	Who is George Bush -> George Bush is
Who <be> the <subject>	The <subject>	Who is the tallest player in NBA -> The tallest player in NBA
Who <be> the <subject>	The <subject> <be>	Who is the tallest player in NBA -> The tallest player in NBA is
Who <be> the <subject>	<be> the <subject>	Who is the tallest player in NBA -> Is the tallest player in NBA
Who <verb> <object>	<verb> <object>	Who killed Abraham Lincoln -> Killed Abraham Lincoln
When <be> <subject>	<subject> <be>	When is Christmas -> Christmas is
When <be> <subject>	<subject> <be> <on>	When is Christmas -> Christmas is on
When <be> <subject>	<subject> <be> <in>	When is Christmas -> Christmas is in
When did <subject> <vb> <object>	<subject> <vb> <object>	When did Japan attack Pearl Harbor -> Japan attack Pearl Harbor
When did <subject> <vb> <object>	<subject> <vbd> <object>	When did Japan attack Pearl Harbor -> Japan attacked Pearl Harbor
When did <subject> <vb>	<subject> <vb> in	When did Japan attack Pearl Harbor -> Japan attack Pearl Harbor in
When did <subject> <vb>	<subject> <vbd> in	When did Japan attack Pearl Harbor -> Japan attacked Pearl Harbor
What <be> <subject>	<subject>	What is cosine similarity -> Cosine similatiry
What <be> <subject>	<subject> <be>	What is cosine similarity -> Cosine similatiry is
What <be> <subject>	<be> <subject>	What is cosine similarity -> Is Cosine similatiry
What <be> the <subject>	the <subject>	What is the statue of liberty -> The statue of liberty
What <be> the <subject>	the <subject> <be>	What is the statue of liberty -> The statue of liberty is
What <be> the <subject>	<be> the <subject>	What is the statue of liberty -> is the statue of liberty
What did <subject> <vb>	<subject> <vbd>	What did Historical Swords Weigh -> Historical Swords weighed
How many <subject>	<subject>	How many players in a baseball game -> players in a baseball game
How many <subject>	number of <subject>	How many players in a baseball game -> number of players in a baseball game
How <adj> <be> <subject>	<subject > <be>	How tall is Yao Ming -> Yao Ming is

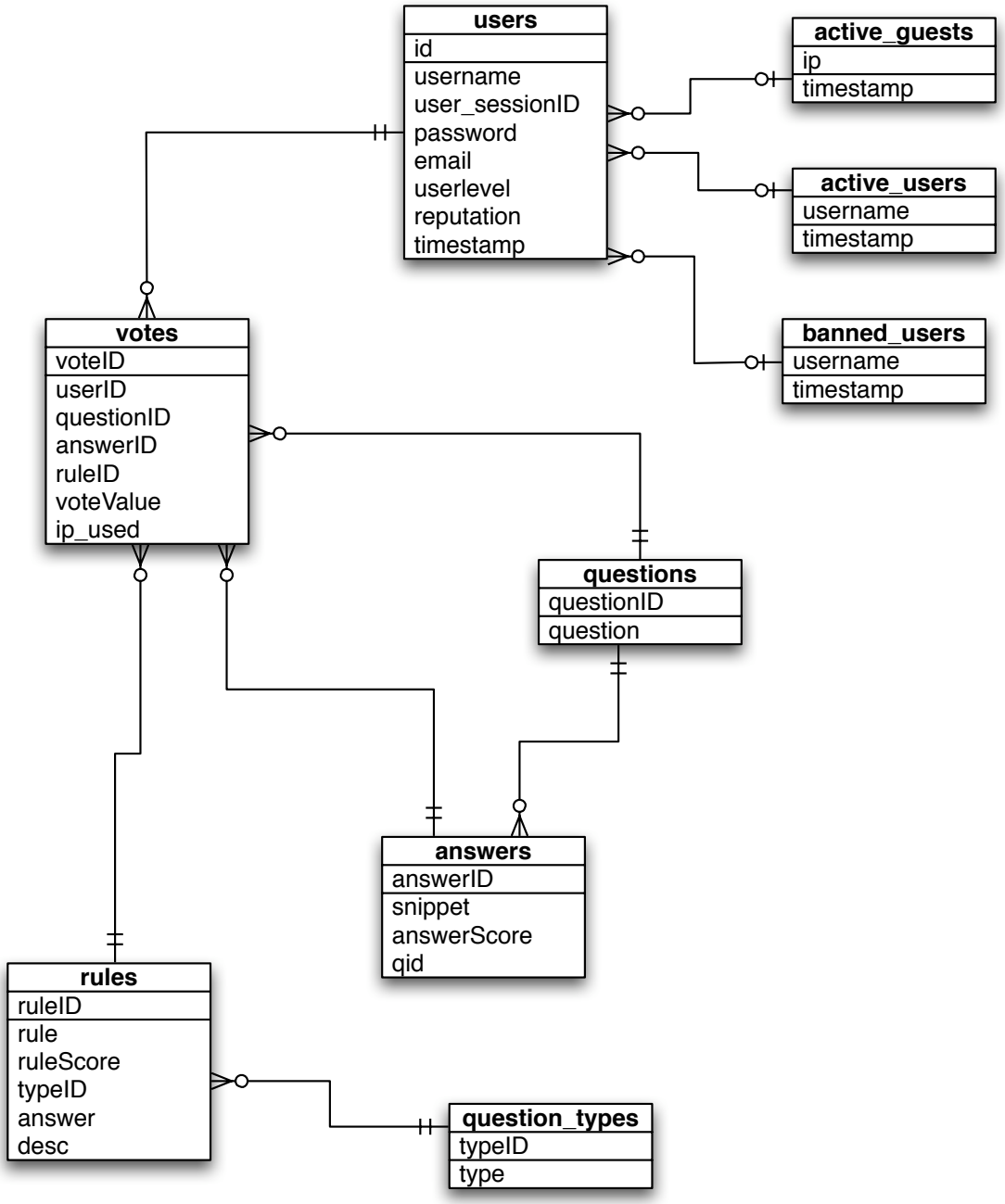

Figure 10: ER diagram of Database.

Table 6: Project Work Distribution

Person	Module
Srini	UI Design, User Study, Question Classification, Answer Filtering
Jimmy	Backend System, Question Transformation, Lucene (Indexing), Answer Highlighting
Jerry	Backend System, Ranking, Training, Voting
Ethan	UI Design, User Study, Reputation and Login system, Mechanical Turk Experiment