Confucius and Its Intelligent Disciples: Integrating Social with Search

Xiance Si
Google Research
Beijing 100084, China
sxc@google.com

Edward Y. Chang
Google Research
Beijing 100084, China
edchang@google.com

Zoltán Gyöngyi
Google Research
Mountain View CA, USA
zoltang@google.com

Maosong Sun
Tsinghua University
Beijing 100084, China
sms@tsinghua.edu.cn

ABSTRACT

Q&A sites continue to flourish as a large number of users rely on them as useful substitutes for incomplete or missing search results. In this paper, we present our experience with developing Confucius, a Google Q&A service launched in 21 countries and four languages by the end of 2009. Confucius employs six data mining subroutines to harness synergy between web search and social networks. We present these subroutines' design goals, algorithms, and their effects on service quality. We also describe techniques for and experience with scaling the subroutines to mine massive data sets.

1. INTRODUCTION

Despite the success of Internet search engines, Q&A sites continue to attract a large number of users. For instance, Yahoo! Answers serves an estimated 1 billion pages to 80 million unique visitors each month.[1] Users visit Q&A sites either because the sought-after information may not be available on the Web, or because the available information is not summarized in a conveniently accessible way. For example, when a very recently launched mobile device fails to boot, it may happen that the fix to the problem is not yet published on the Web. Even when relevant information is available, it may be buried in some long product specification in a way that makes it difficult to locate. Furthermore, in many Asian, Middle Eastern, South American, and Eastern European countries with emerging Internet presence, the number of local Web pages available to search engines for indexing is relatively low (when compared to that in the US and Western Europe). In these countries, Q&A sites are often a prime source of online information. Indeed, for instance, in China, 25% of Google's top-search-results pages contain at least one link to some Q&A site. Thus, Q&A sites are not only useful when some information is not present on the Web, but also constitute a versatile source of indexable content for search engines.

In this paper, we present the key technologies that we relied on for designing and developing *Confucius*, a Google Q&A system that is meant to leverage the social network of Q&A users to synergistically produce high-quality content that may benefit Web search. The performance goal of Confucius is to *provide high-quality answers in a timely fashion*. In building a system that meets such a goal, we encountered four particular technical challenges. First, typical users are seldom willing to provide answers without incentives. Second, in presence of incentives, abusers/spammers are always quick to emerge. The abusers/spammers may lead to a degradation in service quality that can discourage knowledgeable users from contributing. Third, when a question is asked, it should be routed quickly to a domain expert to reduce answering delay. Finally, should an answer to a question already be available, the system must surface it to avoid redundant work (and unnecessary delay).

The first two of the challenges affect answer quality, while the last two affect the timeliness of answers. Besides answer quality and timeliness, we discovered that the nature of the questions has great impact on the quality of the collected Q&A corpus, and thus on the overall user experience. When we first launched Confucius in 2007, as a service on an existing social network, most askers sought subjective information, such as relationship advice. The consequent interactions did not yield generally relevant, high-quality content. As one of the stated goals of Confucius is to improve Web search quality by contributing useful content, we since enhanced the system so that it attracts more questions that have objective, factual answers.

To tackle the aforementioned technical challenges, we developed six system components—six disciples of Confucius—that help us establish a healthy ecosystem spanning both *social* and *search*. These components and their purposes are as follows:

Search integration: We integrate Confucius with the Google search engine. When search terms indicate a *wh*-query (*when*, *where*, *why*, etc.) or when the search engine cannot return sufficiently relevant results (e.g., the content overlap between query and

[1] As reported by the DoubleClick AdPlanner tool (`http://www.google.com/adplanner/site_profile #siteDetails? identifier=answers.yahoo.com`), visited on March 7, 2010.

Proceedings of the VLDB Endowment, Vol. 3, No. 2
Copyright 2010 VLDB Endowment 2150-8097/10/09... $ 10.00.

potential top result pages is low), a Q&A session is recommended, encouraging the searcher to post a question on Confucius.

Question labeling: Once a question has been typed in, a set of category labels are suggested. Labels are useful for organizing the question corpus, for providing a subscription mechanism, and for routing questions to answerers. We employ a parallel implementation of Latent Dirichlet Allocation (PLDA) [12] to make label suggestions.

Question recommendation: Given a question, this subsystem finds similar earlier questions and their already available answers. Such prior information often reduces the time it takes for a user to obtain a satisfactory answer. PLDA is used once more to accomplish this recommendation task.

Answer quality assessment: We evaluate the quality of each answer based on several factors including its relevance to the question and its originality. Being able to assess answer quality in a semi-automatic way is a key requirement in identifying top contributors and in curbing spam.

User ranking: We rank users based on their contributions in a domain specific way. User ranking is used for identifying top contributors within each domain so that we can provide appropriate incentives, enhance the ranking of Q&A pairs in search, and route questions to domain experts.

NLP-based answer generation: Though Natural Language Processing (NLP) research is still in relative infancy, it provides avenues leading to an improved Q&A experience. In particular, we employ NLP techniques to automatically generate answers. Note that a typical user of a Q&A service can tolerate a delay of several minutes (as opposed to the instantaneously expected search results). This relaxed latency constraint allows us to run time-consuming, complex NLP algorithms that may yield good answers. Also note that NLP answers are optional, to the extent that the system may always elect to discard low-quality generated results and simply wait for an actual human response. Accordingly, we can harvest the best NLP results without compromising overall quality. NLP techniques also prove to be helpful in assessing the type of questions and the quality of user-provided answers.

In addition to being accurate, the subroutines underlying our system's components must also be able to deal with massive amounts of data, in the order of terabytes. To address the scalability issues, our foundational algorithms (such as PLDA [12] and UserRank [17]) can be executed on thousands of machines in parallel.

Since 2007, Confucius has been deployed in 21 nations in four languages. In China, for example, the usage of Confucius quadrupled in 2009. Confucius content has a marked presence in Google search results now. Each of the subsystems played a clear role in the quality and timeliness improvements that lead to such a wide adoption.

The rest of this paper focuses on the six subroutines. First, we review the other existing Q&A services and related research results (Section 2). Next, we introduce the Confucius system and its subroutines in detail (Section 3). We continue by presenting key usage statistics (Section 4), revealing how the presented subroutines have helped improve content quality and traffic. Section 5 discusses the lessons we learned and some future research/development directions. Finally, we provide concluding remarks in Section 6.

2. RELATED WORK

More than ten large online Q&A services were launched since 1998. Not all of these systems are alike—they differ mainly depending on how they approach three key issues: *question submission*, *question routing*, and *incentives*. The approaches to these three issues can be regarded as independent variables, which affect dependent variables, such as *answer presentation*.

Name	Key elements				Year
	Question submission	Question routing	Incentives	NLQ&A	
Internet Oracle	by email	to experts	virtual	no	1989
Ask.com	by search	N/A	N/A	yes	1996
WikiAnswers	in community	to public	virtual	no	2002
Yahoo! Answers	in community	to public	virtual	no	2005
Baidu Zhidao	in community	to public & experts	virtual & $	no	2005
Google Confucius	by search & in community	to public & experts	virtual & $	yes	2007
Aardvark [6]	by IM	to friends	virtual	no	2009
Powerset	by search	N/A	N/A	yes	2005
Quora	in community	to friends	virtual	no	2009

Table 1: Representative Q&A sites.

Question submission: A user can ask a question through two access points: within a search engine (like with Confucius' search integration) or on a community site. A post-search question tends to seek facts or objective answers, whereas an asker within a community tends to expect more personalized answers tailored to his/her particular situation.

Question routing: When the answers are provided by users, a question can be routed to domain experts, social contacts, or both. The choice of the routing algorithm affects the nature of the gathered answers. When a question is routed to an expert, the answer tends to be formal and objective. When an answer is provided by a friend, it tends to be more informal and personalized. For example, assume that the question is "Which are the must-see sights in Singapore?" If the asker is known to the answerer, the answerer may tailor the answer to focus exclusively on, say, natural landmarks, historical sights, or contemporary attractions. In contrast, a domain expert may choose to pursue a more balanced approach.

Incentives: Without incentives, few users would provide high-quality answers. Incentives can be built around community reputation or monetary rewards and provided to askers or answerers. Content quality is the primary consideration, but incentives may also help induce bias on content type or category. Let us assume that a site aims to collect facts to eventually build a Wikipedia-like service. Correspondingly, questions with objective answers may be more encouraged on the site. In contrast, a site interested in collecting diverse opinions on restaurants, fashion, or travel, may encourage questions and answers on these topics by differentially rewarding them.

Independently of the above three issues, some services provide machine-generated-answers via search or NLP. NLQ&A is a branch of the general NLP and AI research with more than ten years of history [4]. One NLQ&A approach is to generate answers from a formatted knowledge base, such as Wikipedia or relational databases. A typical system consists of three parts: question analysis, information retrieval, and answer selection. During the question analysis step, the system classifies a question by the expected type of the answer, using regular patterns [16] or machine learning [11]. Then, the question is used as a query to search the document set and possible answers of the expected type are extracted from the search results. Third, the system ranks the candidate answers by frequency, sophisticated linguistic features [9] or information distance [20].

A different NLQ&A approach is based on finding similar questions within an existing Q&A database. Lai et al. [10] proposed a method to mine questions and answers in Web FAQs. Jeon [7] modeled the similar question finding problem as a machine translation task, and showed such modeling to be more effective than some other similarity measures. Xue et al. [19] extended previous work by combining the language model and IBM's translation model. Murdock et al. [13] used non-content features to help retrieve answers for procedural questions. Surdeanu et al. [18] also presented a system that leverages NLP features to improve the accuracy of question retrieval. Agichtein et al. [1] reported their experience in assessing the quality of existing answers, which helps NLQ&A systems rank matching answers. Since no human is involved in the answering process, NLP is more suitable for answering questions seeking factual answers rather than opinions.

Table 1 lists the main Q&A services and their positioning with respect to the three key issues, together with the answer sources. For instance, Confucius aims to collect factual information by providing *search integration*, *question-label suggestion*, *expert identification and routing*, and *quality-based incentives*.

3. THE CONFUCIUS SYSTEM

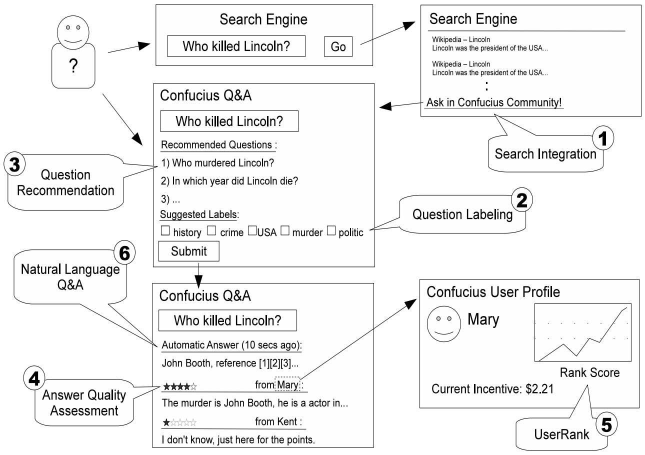


Figure 1: The Confucius Q&A system.

Figure 1 presents the question answering work flow, including steps that involve Google search and the six main subroutines of Confucius (represented by numbered bubbles). For convenience, we introduce the following acronyms for the corresponding subroutines:

SI : Search Integration,

QL : Question Labeling,

QR : Question Recommendation,

AQ : Answer Quality Assessment,

UR : User Ranking, and

NL : Natural Language Q&A.

Currently, there are two access points to Confucius: through the Google search engine and through the Confucius site. After a user issues his/her search query, SI (Figure 1(1)) decides whether the search engine should recommend the user to initiate a Q&A session. Once the user submits a question, the QL subroutine (Figure 1(2)) suggests a list of category labels to the user to select from. The QR subroutine (Figure 1(3)) compares the submitted question with existing ones and, in case of a match, recommends the corresponding answers to the user. If the user finds the matching question and answers satisfactory, he/she can terminate the Q&A session by canceling the new question asking process. Otherwise, the question is routed to the users who have subscribed to some of the chosen category labels and/or to domain experts identified by the system. The asker can optionally select preferred notification mechanisms such as email or instant messaging for when an answer becomes available. Askers may also provide feedback on the submitted answers, in the form of best-answer selection or agree/disagree votes. However, in practice most answers will lack such feedback. To assess the quality of answers with a broader coverage, Confucius implements the AQ subroutine (Figure 1(4)), which learns the relationship between different features and the quality of the answer as indicated by user votes, and uses the learned model to assign quality scores to answers without explicit feedback. Quality scores are then aggregated by UR (Figure 1(5)), which quantifies user contributions and ranks users in a topic-dependent manner. Finally, the NL subroutine (Figure 1(6)) provides a machine-generated answer when it is capable of. In the remainder of this section, we describe the goals, functionalities, and algorithms of each key subroutine.

Search Integration subroutine (Online)

Given: A SVM model obtained offline M_{SI}, a threshold of PageRank value ϵ

Input: A Web query q

Output: Decision δ, show the triggered link if $\delta = true$

```
if $M_{SI}$ classify $q$ as a question
    $\delta = true$
else if $\max\{PageRank(r) | r \in FirstPageResults\} < \epsilon$
    $\delta = true$
else if the user clicked Next on the first page
    $\delta = true$
else
    $\delta = false$
```

Figure 2: SI subroutine.

3.1 Search Integration (SI)

There are two goals of SI : improving search experience and enhancing search quality. To improve search experience, SI provides the user an opportunity to ask a question when search results are not satisfactory. The goal of enhancing search quality is more subtle but equally important. A search user tends to seek factual information. Thus, should he/she ask a question, the desired answers are probably objective, fact-based ones. If such fact-based answers are provided, they may be indexed and used for providing better search results to subsequent similar queries (thus potentially eliminating the need for a later repeated question submission by another user). Currently, SI triggers a Q&A session under the following two conditions:

1. When the query is in a question form, or
2. When the computed relevance of all the search results to the query (whether in a question form or not) falls below some threshold.

Confucius classifies search keywords into eleven categories, listed in Table 2. We assembled the list of categories by sampling a set of questions and manually annotating them. The resulting categories are not exhaustive, though they still cover the majority of the questions. SI uses a multi-class SVM classifier with word features. The classifier is trained offline. When the terms of a search query can be classified into one of these eleven categories with a high SVM margin, SI triggers a Q&A session.

The search result evaluator uses two signals to identify queries result sets that are not relevant enough. The first signal is based on the maximum query-independent quality of any page in the result set. The query-independent page quality can be approximated using algorithms such as the well-known PageRank algorithm [3]. If

Category	Purpose	Expected answer
REC	Recommendation	Suggested item and the reason
FAC	Seeking facts	Date, location, or name
YNO	Yes/No decision	Yes/no and reason
HOW	How-to question	The instructions
WHY	Seeking reason	The explanation
TSL	Translation	Translated text in target language
DEF	Definition	The definition of given entity
OPN	Seeking opinion	Opinions of other users
TRA	Transportation	Navigation instructions
INT	Interactive	Discussion thread
MAT	Math problem	Solution with steps

Table 2: Categories of questions in NL.

Rank	From Community	From Web Users
1	Gossip	Medicine
2	Web	Software
3	Software	News
4	Online Games	Hardware
5	Relationship	Engineering
6	Entertainment	Web
7	Downloads	Online Games
8	Mood	Technology
9	Travel	Economy
10	Medicine	Products

Table 3: Top labels of questions, from community users and from SI-directed users.

for a query q, no result page has a high PageRank score, one can naturally expect user dissatisfaction. The second signal is the presence or absence of a user click on the *Next* link on the search result page. Moving to the second result page or beyond implies that the user was less than perfectly happy with the top results.

When at least one of the two triggering conditions is satisfied, the search results page will include an additional link to Confucius at the bottom of the result list, with a suggestion like "Would you like to ask a question about *search terms* on Confucius?" Upon clicking on the link, the user is led to Confucius' question submission page. On that submission page, the user can refine the question and obtain possible existing answers by browsing previously-asked, similar questions on the right-hand side. If the user has not registered on Confucius, he/she may choose to ask the question anonymously, and leave an email address for receiving notifications when answers become available.

SI may appear to be a straightforward integration, but it yields a variety of significant benefits. First, it provides a substantial flow of questions to Confucius. At a reasonable question triggering rate, SI can bring tens of thousands of questions to Confucius per day. Also, as we mentioned, search integration yields not only quantity, but quality as well. Table 3 shows the most frequent question labels collected on our Chinese site, from community users and from SI-directed users, respectively. Before search integration was launched, 90% of the submitted questions were of subjective nature, soliciting opinions on gossips, relationships, fashion, and entertainment. With SI-directed questions, Confucius has begun to cover more objective topics, such as medicine, engineering, and economy.

3.2 Question Labeling (QL)

The goal of QL is to help organize and route questions with automatically recommended labels. The QL subroutine takes a question as the input and outputs an ordered list of labels that best describe the question. Labels consist of a set of words or phrases that best describe the topic or type of the question. Confucius allows at most 5 labels per question, but puts no limit on the size of the global label vocabulary. Confucius organizes the most important category labels into a two-layer hierarchy, in order to provide a better browsing experience. QL is used by two other subroutines: UR and QR . When ranking users, UR uses popular labels to compute the topic-dependent rank scores. QR assigns questions to users via either subscription or expert identification, during which labels generated by QL are used for matching. The precision and recall of suggested labels are two important metrics for measuring QL performance. Precision measures the correctness of suggested labels, while recall measures the completeness. In Confucius, the precision and recall of QL are measured by online user feedback, which we present in Section 4.

```
Question Labeling Subroutine (Offline Training)
Input: Questions $Q = \{q_1, \ldots, q_n\}$, in which $q_i = \{w_{i,1}, \ldots, w_{i,n}\}$, Labels $L = \{l_1, \ldots, l_m\}$ and their relationship with labels $R \in Q \times L$
Output: LDA models $M = \{(\theta, \phi, k)\}$
1: for $l \in L$
2:     $d_l = \{w | w \in d, \forall (d, l) \in R\}$
3: Remove stop words and rare words from all $d_l, l \in L$
4: $M = \{\}$
5: for $k \in \{32, 64, 128, 512\}$
6:     Train LDA model $(\theta, \phi, k)$, with $\{d_l\}$ and $k$ topics
7:     $M \leftarrow (\theta, \phi, k)$

Question Labeling Subroutine (Online)
Given: LDA Models M = $\{(\theta, \phi, k)\}$
Input: Question $q = \{w_1, \ldots, w_{|q|}\}$
Output: Suggested Labels $L_q = \{l_1, \ldots, l_n\}$
1: Infer $\theta_{q,k}$ with $M$
2: $S_k(\theta_q, \theta_{d_l}) = CosSim(\theta_{q,k}, \theta_{d_l}), \forall l \in L$
3: $S(q, l) = \overline{\{S_k(\theta_{q,k}, \theta_{d_l,k})\}}$ /* Mean similarity */
4: $L_q = \{l | |\{l' | S(q, l') > S(q, l)\}| < N\}$ /*Top N labels*/
```

Figure 3: QL subroutine.

Figure 3 shows the two parts of QL: offline training and online suggestion. In the offline training part, we employ Latent Dirichlet Allocation (LDA) [2], a latent semantic model, to model the relationship between words and topic labels. The training data is the existing set of questions with user-submitted labels. First, we merge all questions with the same label l into a meta-document d_l, and form a set of meta-documents $\{d_l\}$ (Figure 3, steps 1-2). Second, we remove all stop words and rare words to reduce the size of each meta-document (step 3). Third, we use $\{d_l\}$ as the corpus to train LDA models (steps 5-6). The label corresponds to the document in LDA definition, while the words in the meta-documents correspond to the words. The resulted LDA model decomposes the probability $p(w|l)$ to $\sum_z p(w|z)p(z|l)$—this is similar to the factor model in recommendation algorithms, expressed in terms of probabilities. Instead of a single model, QL trains several LDA models with different number of latent topics. Using multiple LDA models with different k-s is known as *bagging*, which typically outperforms a single model and avoids the difficult task of setting an optimal k, as discussed by Hofmann [5].In the current QL system, the following numbers of topics are used: $k = 32, 64, 128$ and 256. We collect all LDA models into a set M (step 7) and save it to disk. The training part works offline. To handle large training data, we use a parallel implementation called PLDA [12] on thousands of machines in order to maintain training time within the range of a few hours.

The online suggestion part assigns labels to a question as the user types it. The bottom half of Figure 3 depicts the suggestion algorithm. First, we use each LDA model in M to infer the topic distributions $\{\theta_{q,k}\}$ of the question q (step 1). Then, we compute the cosine similarity $CosSim(\theta_{q,k}, \theta_{d_l,k})$ between $\theta_{q,k}$ and $\theta_{d_l,k}$ (step 2). Third, we use the mean similarity over different values of k as the final similarity $S(q, l)$ between a question and a label (step 3). Finally, we sort all $l \in L$ by $S(q, l)$ in descending order, and take the first N (say ten) labels as recommended ones (step 4).

Using PLDA for QL has two benefits: semantic matching and scalability. PLDA decomposes each question and answer into a distribution over a set of latent topics. When encountering ambiguous words, PLDA can use the context to decide the correct semantics. For example, QL suggests only labels such as ‘mobile’ and ‘iPhone’ to the question “*How to crack an apple?*”, although the word *apple* also means the fruit “apple.” In addition, PLDA is designed to scale gracefully to more input data by employing more machines. The scalable implementation enables SI to handle fast-growing training sets to deliver more accurate suggestions.

3.3 Question Recommendation (QR)

```
Question Recommendation Subroutine (Offline Training)
Input: Questions $Q = \{q_1, \dots, q_n\}$, in which $q_i = \{w_{i,1}, \dots, w_{i,n}\}$. Answers
$A = \{a_1, \dots, a_n\}$, in which $a_i = \{w_{i,1}, \dots, w_{i,n}\}$. Labels $L = \{l_1, \dots, l_n\}$.
The relation $R_{QA} = Q \times A$ between $Q$ and $A$, and the relation $R_{LQ} = L \times Q$
between labels and questions
Output: LDA model $M_{q2q} = (\theta, \phi, k)$, inverted word index $I_w \in W \times Q$, and
inverted label index $I_l \in L \times Q$
1: Remove stop words from all $q \in Q$
2: Train LDA model $(\theta, \phi, k)$, with $\{q\}$ as the corpus and $k$ topics
3: $I_w = \{\}$ /* Build word index */
4: for $q \in Q$
5:     for $w \in q$
6:         $I_w \leftarrow (w, q)$ /* Build index with words of the question*/
7:     for $w \in a, \exists (q, a) \in R_{QA}$
8:         $I_w \leftarrow (w, q)$ /* and its answers */
9: $I_l = \{\}$ /* Build label index */
10: for $q \in Q$
11:     for $l, (l, q) \in R_{LQ}$
12:         $I_l \leftarrow (l, q)$
```

```
Question Recommendation Subroutine (Online)
Given: LDA Model $(\theta, \phi, k)$ and inverted index $I_w, I_l$, the current time $t_now$,
the asking time $t_q$ of each question $q$ as days, Each word's inverted document
frequency $IDF = \{idf_w\}$, Time-sensitive words $W_t$
Input: Question $q = \{w_1, \dots, w_{|q|}\}$
Output: A list of recommended questions $Q' = \{q'_1, \dots, q'_n\}$
1: Build candidate set $Q' = \{q' | (w, q') \in I_w, \forall w \in q, or (l, q') \in I_l\}$
2: Infer topic distribution $\theta_q$ with LDA model $(\theta, \phi, k)$
3: Build TF*IDF word vector $\gamma_q$ with $IDF$.
4: Find question location $l_q$
5: for $q' \in Q$
6:     Infer topic distribution $\theta_{q'}$ with LDA model $(\theta, \phi, k)$
7:     Build TF*IDF word vector $\gamma_{q'}$. with $IDF$
8:     $Sim(q, q') = CosSim(\theta_q, \theta_{q'})^{\alpha_1} CosSim(\gamma_q, \gamma_{q'})^{\alpha_2}$
9:     Find question location $l'_q$
10:     $LocationBoost(q, q') = EuclideanDistance(l_q, l_{q'})^{-1}$
11:     if $q \cap W_t \neq \varnothing$
14:         $Freshness(q') = (t_{now} - t_{q'})^{-1}$
13:     else
12:         $Freshness(q') = 1$
15:     $Quality(q') = max\{quality(a), a \in q'\}$
16:     $Rank(q') = Sim(q, q')^{\lambda_1} LocationBoost(q, q')^{\lambda_2}$
    $Freshness(q')^{\lambda_3} Quality(q')^{\lambda_4}$
17: sort all $q' \in Q'$ by $Rank(q')$ in descending order
```

Figure 4: QR subroutine.

QR recommends similar, existing questions to a user when he/she issues a question. The goal is to recommend relevant questions that already have good answers so that the user can access the solicited information faster. QR formulates the question recommendation problem exactly in the same way as QL formulates the label suggestion. Given a question, QL assigns relevance scores to all existing questions and ranks them accordingly. The input to the QR subroutine is a question, the output is a list of questions sorted by their relevance scores. Similarly to QL, QR also employs PLDA and consists of an offline training part and an online recommendation part.

Figure 4 presents the training and recommendation algorithms. The offline part builds an LDA model and an index for all questions (Figure 4 top). First, we remove all stop words from the questions and train a PLDA model (steps 1-2). Then, we build an inverted word index of all questions (steps 3-8). The purpose of inverted indexing is to shrink the candidate question set, since computing similarity between all questions is a computationally intensive task. However, the word index may miss candidates, since a typical question contains very few words, and the same meaning may map to multiple words. To enrich the data for an answered question, we also include the words in its answers when indexing it (steps 7-8). QR also indexes questions using labels generated by QL (steps 9-12).

After a user has typed in a question but before he/she hits the submit button, QR recommends related questions (Figure 4 bottom). QR uses the words in the question and its labels to query the index. The top N results are returned as the recommended questions (step 1). QR then adds additional factors to rank these returned questions. The considered factors include similarity, freshness, quality, and location boost (steps 8-12), detailed as follows:

Similarity. Similarity is the main factor in candidate question ranking. Questions are short, introducing the word sparsity problem in similarity computation. We take a corpus-based approach in question similarity computation using PLDA. The similarity score is computed in step 8, where $Sim(q, q')$ is the similarity score between questions q and q', and θ_q is the topic distribution vector of q. Parameters α_1, α_2 are blending factors for topic-based and word-based similarities. The similarity measure uses both words and topics—the word vector is used to emphasize exact matching, while the topic vector is used to capture non-overlapping synonyms.

Freshness. For some questions, up-to-date answers are preferable to old ones. For instance, for “*Who is the winner of the latest Olympic gold in men’s 100m sprint?*.” QR emphasizes recent answers through freshness scores. QR uses a set of manually selected trigger words W_t, such as *today* and *recently*, to identify questions that expect a recent answer. If the input question contains a trigger word, QR computes the freshness score of each candidate question as the reciprocal of the days since the question was answered (step 12). The newer the question q’s answer(s), the higher the freshness score $Freshness(q)$ (Freshness is set to one when an answer was provided today.) If the input question does not contain any trigger words, QR sets all freshness scores of the answers to 1, so as not to induce suffer unwarranted demotion in ranking (step 14).

Quality. Besides freshness, QR also considers the quality of the recommended questions. If two perfectly matching questions exist, one with a best answer, and the other with a spam answer, we would like to display the best-answered one first. $Quality(q)$ measures the quality of a question by the quality of its answers (step 15), where the answer quality $quality(a)$ is provided by user feedback and our answer quality predictor (Section 3.4). If an answer is marked as a best answer, we set $quality(a) = 1$. If an answer was reported as spam, we set $quality(a) = 0$. Otherwise, we use the automatic quality predictor to predict the quality of all answers to that question, then use the maximum answer quality as the question’s quality score. The quality of an unanswered question is set to zero.

Factor	Parameter	Description
Relevance	#rword	Computed as the cosine similarity between TF*IDF weighted word vectors of the question and the answer.
	#rlda	Computed as the cosine similarity between the PLDA latent topic distributions of the question and the answer.
Originality	#origt	Compared to earlier answers for the same question.
	#origu	Compared to earlier answers from the same user.
Timeliness	#prompt	The time from when the question was asked to when the answer was provided.
Coverage	#cov	The sum of the unique words' IDFs.
Spam	#nau	Total number of answers provided by the answerer in the past.
	#npbau	Percentage of best-answers in all closed threads in which the answerer has participated.
	#naul	Number of answers posted by the answerer to questions on the same topic.
	#npbaul	Percentage of best answers posted by the answerer in closed threads on the same topic.
	#entropy	The entropy of the user's topic distribution of answers.

Table 4: Quality factors and parameters in AQ .

Location Boost. Some questions are closely related to geographical locations, such as questions asking for the best restaurant in a city. To emphasize location in question matching, we add location boost to the final rank score (steps 9-10). The location mentioned in the question is extracted using a standard named-entity extractor, such as one for country and city names. Then, we convert the location description to geographical coordinates (steps 4 and 9). Location boost is computed as the reciprocal of the geographical distance between the current question and an earlier candidate question. If no location-related entities are detected in the original question, we consider it to be a location-independent question, and set all candidate questions' location boost to 1.

In step 16, QR combines four factors into a final score $Rank(q)$ using weights λ, which are chosen empirically. In the final step (step 17), QR sorts all candidate questions in descending order by their $Rank(q)$, and returns the top ones to the user.

3.4 Answer Quality Assessment (AQ)

User feedback, such as best-answer votes, is a commonly utilized way of assessing answer quality in community Q&A sites. However, the majority of answers do not have enough votes, and fresh answers are not likely to obtain any votes soon. We leverage an automatic answer-quality assessment routine AQ to produce quality ratings. Since AQ can be trained and executed off-line, its primary design goal is accuracy over speed.

We model the AQ task as a binary classification task of best and non-best answers. Each instance is an answer, represented by a set of features. The classifier outputs a confidence score for an answer being a best answer, which can be deemed as a quality score. The training data is Q&A threads with user annotated best answers. We consider five groups of quality factors as features: *relevance*, *originality*, *timeliness*, *coverage*, and *spam*. These quality factors are quantified by parameters enumerated in Table 4.

Relevance. An answer must be relevant to the question in a Q&A thread. We use the same approach as in question recommendation to compute the relevance between the question and the answer (the bottom of Figure 4, step 9). We also share the same offline models with question recommendation.

Originality. Content copying is widespread and mostly unavoidable on Q&A sites. Some duplication—such as straightforward plagiarization of other users' answers or using the same text to answer multiple questions—is clearly unwanted and the authors' reputation should be diminished accordingly. Parameter #origt indicates how an answer differs from other answers on the same Q&A thread. Parameter #origu indicates how an answer differs from the other answers provided by the same user. To quantify content originality, we leverage BleuScore [15], which is a standard metric in machine translation for measuring the overlap between n-grams of two text fragments.

Timeliness. The time it takes to answer a question is relevant to quality. However, a short response time may not always indicate high quality. Whereas some questions can be answered very quickly, well thought-out answers to some others (think college-level science and mathematics, for instance) may take time. For this latter kind of questions, users providing prompt answers are more likely to be spammers than genii. Therefore, though the parameter #prompt is simple to compute, its quality interpretation may be topic-dependent. Rather than relying on an explicit rule for applying the timeliness factor, we use supervised learning to weight this factor through the training process.

Coverage. The length of an answer can imply high quality if the response time is not too short, the answer is highly relevant, and the content is original. Since we want to emphasize the *useful* part of an answer, we measure content coverage by summing the inverse document frequencies (IDF) of its unique words (#cov).

Spam. Spam signals are an important factor in determining answer quality. Some individuals may have strong economic incentives to attract post views by users. To achieve their goal, spammers generate a large number of posts using multiple accounts. To curb common spam, such as commercial and porn spam, we train targeted porn and commercial classifiers for identifying corresponding posts. Identified questions/answers can be traced back to the perpetrating users. Then, once a spammer is identified, the rest of his/her content can be surfaced in a transitive manner.

Besides the monetization of content visibility, potential reputation within a community may also motivate spammers. This is particularly true if a service offers financial rewards to its power users, as we do in Confucius. To identify active spammers, an #entropy score is computed for each user based on the topics that his/her answers cover. Let $\boldsymbol{\theta}$ be the topic distribution of an answer that is inferred by PLDA. Then, the focus score is the average entropy of all answers of that user, $\#entropy(u) = \sum_{i=0}^{N}(-\sum_{j=0}^{k}\theta_{i,j}log(\theta_{i,j}))$, where N is the number of answers of the user and $\theta_{i,j}$ the probability that the i^{th} interaction is on the j^{th} topic. In other words, $\#entropy(u)$ measures the uncertainty in user u's latent topic distribution. Thus, a higher $\#entropy(u)$ means that the user tends to answer more types of questions, whereas a low $\#entropy(u)$ indicates that a user focuses on a small set of topics.

With our training data and features, we take a data-driven approach to finding an appropriate prediction algorithm. AQ uses cross-validation to choose the most appropriate classification algorithm. The cross-validation accuracy of different classifiers is presented in Section 4.

3.5 User Ranking (UR)

The goal of the UR subroutine is to find the high quality users on different topics. As depicted in Figure 5, UR consists of three main steps. First, we convert the Q&As to topically weighted interactions between users (step 1). Then, we generate the user activity graph G from the interactions (step 2). Finally, we run a weighted and topic-sensitive HITS computation on G (step 3), and combine

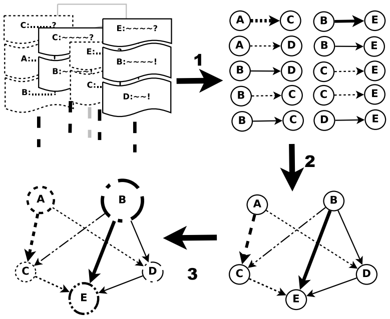


Figure 5: The UR subroutine. Two different topics are represented by solid and dotted lines. Line widths convey the weights of the interactions.

the hub and authority scores into a final UR score.

Each pair of question and answer is an interaction between the asker and the answer provider, denoted as $(u_f, u_t, \mathbf{w})$, where u_f is the answer provider who followed in this interaction, u_t is the asker who started the interaction, and $\mathbf{w}$ is the topical weight. To get topical weighting, we use the LDA model from QL once again to infer for each interaction the topic distribution of the question, denoted as $\boldsymbol{\theta}$. Then the topical weight vector $\mathbf{w}$ can be computed as $\mathbf{w}_i = w_q \theta_i$, where w_q is the quality score computed by AQ in Section 3.4.

We use the user activity graph to connect users by interactions. An edge from u_f to u_t represents one or more interactions from u_f to u_t. Each edge has an assigned weight vector $\mathbf{w}^{(u_f,u_t)}$, where $w_i^{(u_f,u_t)}$ represent the weight for the i^{th} topic. We compute the edge weight from interactions as:

$$\mathbf{w}_i^{(u_f,u_t)} = \sum_{\forall (u_f,u_t,\mathbf{w}') \in Y_{u_f,u_t}} \frac{w'_i}{|Y_{u_f,u_t}|}, \tag{1}$$

where Y_{u_f,u_t} represents the set of all interactions from u_f to u_t. Accordingly, for each topic, we use the mean weight of the interactions as the edge weight. The mean is preferable over the sum, as it reduces the effect of a large set of low-quality interactions.

We adopt Ng et al.'s randomized HITS [14], which adds a universal random jump probability to standard HITS [8]. To incorporate topic distribution, we use vectors to represent the topic-specific hub and authority scores and edge weights, namely, $\mathbf{h}$, $\mathbf{a}$ and $\mathbf{w}^{(u_i,u_j)}$. The number of dimensions is the number of topics k. The values in $\mathbf{w}^{(u_i,u_j)}$ represents the topic-specific weight of the interactions between user u_i and u_j. We use the following vector-based HITS formula to get $\mathbf{h}$ and $\mathbf{a}$:

$$\mathbf{h}_u^{(n+1)} = (1-\epsilon) \sum_{u' \in in(u)} \frac{\mathbf{w}^{u',u} \cdot \mathbf{a}_{u'}^{(n)}}{\sum_{u^* \in out(u')} \sum_{i=0}^{k} w_i^{u',u^*}} + \epsilon \tag{2}$$

$$\mathbf{a}_u^{(n+1)} = (1-\epsilon) \sum_{u' \in out(u)} \frac{\mathbf{w}^{u,u'} \cdot \mathbf{h}_{u'}^{(n)}}{\sum_{u^* \in in(u')} \sum_{i=0}^{k} w_i^{u^*,u'}} + \epsilon \tag{3}$$

where $(\cdot)$ represents the Hadamard (or entry-wise) product of two vectors. Weighted topic-specific HITS propagates topic-specific hub and authority scores through the edges. The final $\mathbf{h}$ and $\mathbf{a}$ reflect the user's reputation scores for different topics. When there is only one topic, weighted topic-specific HITS becomes weighted HITS.

The result of HITS on the user activity graph captures two aspects of user reputation: the hub score represents a user's ability to reach out to other users, and the authority score represents a user's ability to gain attention from the others. In Confucius, providing more good answers is encouraged, so hub scores may be valued higher. We take a linear combination of the hub score and the authority score to produce a single final UR score. The final topic-sensitive score vector is $\boldsymbol{\phi}_u = \gamma \mathbf{h}_u + (1-\gamma)\mathbf{a}_u$, where γ is the mixture weight. There are two ways to choose γ. If one has some training data (that is, a list of manually ranked users) readily available, one can learn the most appropriate γ by regression. If one does not have training data and it is expensive to obtain it, one can set γ empirically. For example, one can use $\gamma = 1$ to emphasize answering behavior.

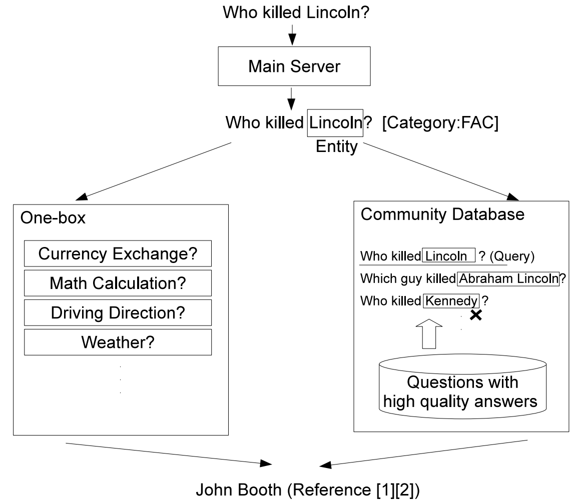

Who killed Lincoln?
Main Server
Who killed Lincoln? [Category:FAC]
Entity
One-box
Currency Exchange?
Math Calculation?
Driving Direction?
Weather?
Community Database
Who killed Lincoln ? (Query)
Which guy killed Abraham Lincoln?
Who killed Kennedy ?
Questions with high quality answers
John Booth (Reference [1][2])

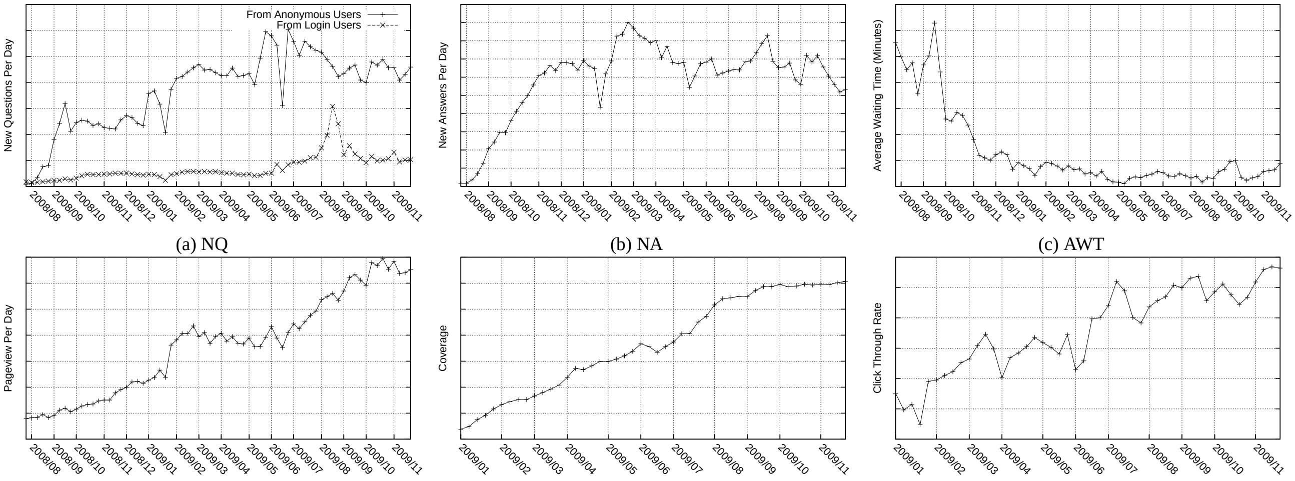
Figure 6: The NL subroutine.

3.6 NLP-based Answer Generation (NL)

NL generates an answer from existing documents to answer a question. Figure 6 shows the four key steps of the NL subroutine. First, NL classifies a question as we discussed for the SI subroutine. Based on the category, NL delivers the question to two sub-servers: the one-box server and the community database server. Each sub-server uses its own method to generate an answer a with some confidence score $C(a) \in [0,1]$. Besides categorization, NL employs a standard *named-entity recognizer* to extract named entities from the question. Named entities may include locations, dates, company names, person names, and product names. Entities are provided to the sub-servers to perform further processing.

The first sub-server is the one-box sub-server. One-boxes are common features of web search engines, which aim to provide direct answers instead of page snippets to simple and structured queries, such as "*San Francisco weather*" or "*movies in Beijing*." Usually, the one-box server can answer queries such as time, date, currency exchange rates and other financial data, math calculations, route planning, machine translation, and weather. NL sends TSL, TRA and MAT questions (see Table 2) to the one-box sub-server. If the one-box server finds an answer, we set the confidence score to 1, otherwise 0.

The second sub-server is the community database sub-server, which indexes all questions with high quality answers in the Confucius database, and provides the answer by finding the matching question. The basic routine for finding similar questions mimics the QR subroutine (Section 3.3). However, while QR focuses on providing a list of related questions to the user, NL focuses on provid-



(a) NQ (b) NA (c) AWT

(d) NPV (e) COV (f) CTR

Figure 7: Overall statistics for Confucius China. Following the initial deployment, SI was launched in August 2008, UR was launched in September 2008, and QR was launched in February 2009.

ing accurate answers. Thus, we consider two more factors on top of the QR subroutine: entity matching and category matching. Entity matching is computed as the number of matched entities between the input question and existing questions. Category matching is an indicator function, which outputs one if the category of the input question is the same as that of the existing question, and zero otherwise. We send REC, YNO, FAC, HOW, WHY, INT, and OPN questions to the community database sub-server. The confidence score of a returned answer is computed as $Rank(q, q') * quality(a)$, where $Rank(q, q')$ is the similarity rank between the question q and the indexed question q', and $quality(a)$ is the quality of the answer provided by the AQ subroutine.

After collecting all sub-servers' responses, the main server sorts all answers by their confidence scores. The answer with the highest confidence score is returned to the user. NL cannot cover all questions, especially those seeking non-factual answers. If the confidence scores of all sub-server answers are lower than a pre-determined threshold, NL will not return any answers.

4. PRODUCT STATISTICS AND EXPERIENCE

This section reports some key statistics and evaluation results of the Confucius system. In particular, we present how the six subroutines have helped us harness the synergy between search and community.

4.1 Overview

Confucius was launched in 21 countries so far. Among these countries, Russia was the first one, followed by China, Thailand, and Egypt. We use the statistics collected for Confucius China to report on our experience. Since some raw performance figures constitute confidential proprietary information, we opt to report percentages of improvement rather than absolute numbers on question/answer volume, page views, and click-through rates.

Figure 7 shows the overall statistics for Confucius China, along with the approximate launch time of the six subroutines. QL was launched at the very beginning, so it is not shown separately. We discuss several individual metrics as follows:

NQ: Referring to the number of new questions per day, it represents a system growth indicator. Note that Confucius features SI, which allows anonymously posted questions. Hence, we show NQ for logged-in and anonymous users separately in Figure 7 (a). Compared to logged-in users, NQ for anonymous users is much higher since SI was launched. This indicates that SI is the source of a large number of new questions.

NA: Referring to the number of new answers per day, it represents another growth indicator. As the number of questions increases, the number of answers grows too. Figure 7 (b) shows NA for Confucius China.

AWT: AWT is the average waiting time, which is computed as the number of minutes between the submission of the question and the first answer. AWT measures the timeliness of answers, which is a central goal in Confucius. Figure 7 (c) shows the AWT for Confucius China. Along with the growth of the community, AWT drops gradually. The steady decrease of AWT shows once again that the community is capable of handling questions from SI. Typically, a new question in Confucius China will receive its first answer within one hour. The shortening AWT provides askers with positive feedback and encourages them to ask more questions.

NPV: NPV is the number of page views per day, a widely used popularity metric for web sites. For a Q&A site, more questions and answers are only part of the goal, while letting more users benefit from the Q&A database is also important. Figure 7 (d) shows the NPV for Confucius China, indicating that Confucius is becoming more popular. Notice that NPV increased significantly around February 2009, which corresponded with the launch of QR . Besides showing QR recommendations on the question submission page, Confucius also put QR on every Q&A thread page. Allowing users who browse existing Q&As to review related questions boosted the NPV of Confucius, and let users spend more time on the site. NPV also relates to content quality: should users encounter low quality content on Confucius over and over again, they would not come back and NPV would stop increasing. This is certainly not the case at this time.

COV: The coverage of Confucius content in web search result, is computed as the percent of queries that contain results from Confucius on the first page (10 results). High COV is another indicator of content quality, since site reputation and overall quality are important factors in the independently generated search engine ranking.

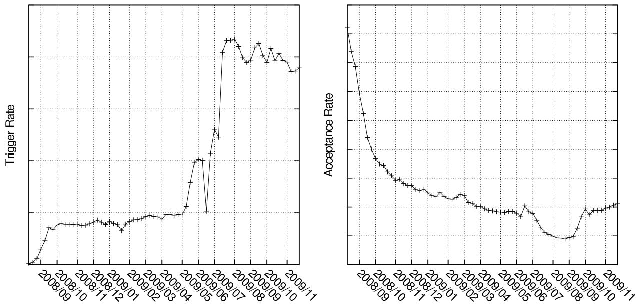
Figure 7 (e) shows COV for Confucius China in 2009, indicating that content quality is increasing and more users get to see results from Confucius while searching the web.

CTR: The click-through rate of Confucius results in web search is also a direct measure of the content quality in Confucius. Figure 7 (f) shows that the CTR for Confucius China was increasing over the last year.

Next, we show the performance of each subroutine with detailed evaluation data.

4.2 Search Integration

SI leads unsatisfied users from Web search to community Q&A. We use trigger rate and acceptance rate to measure SI 's performance. Trigger rate is computed as the percentage of queries that trigger the link to Confucius. Acceptance rate is computed as the percentage of triggered links that user actually clicked. The higher the acceptance rate, the more users expect SI links to be useful. On the one hand, we want SI to cover as many queries as possible; on the other hand, showing the Confucius link for completely irrelevant queries hurts the user experience, which we want to avoid. Figure 8 shows the two rates of SI of Confucius in China.



(a) Acceptance

(b) Trigger Rate

Figure 8: Acceptance(a) and Trigger Rate(b) of SI .

Recall that SI uses two algorithms to decide about the visibility of the Confucius link: question recognizer and search result relevance assessor. At first, we enabled the question recognizer only, without the search result relevance assessor (before May 2009). The coverage reached a stable level quickly after launch. The CTR was very high in the first several months, then returned to a stable value. Users' curiosity was the main reason for the higher CTR in the beginning, as they were not familiar with the Confucius link, and might have expected instant and correct answers by following the link. After users became familiar with the functionality of the Confucius link, they tended to click less often, only when it was necessary to fall back on community answers. In May 2009, we launched the search result relevance assessor in SI to work along with the question recognizer. The first thing that changed was the coverage, which quickly increased by four times. As the coverage increased, the CTR dropped temporarily. However, after users had adapted to the SI logic change, the CTR returned to its previous level. So, the overall result of launching the second algorithm was an increase in the number of questions initiated from web search, which did not compromise the other metrics.

Besides yielding more questions, SI also helps us bridge the needs of web users and community members. Prior to SI , all questions in Confucius were submitted by logged-in community users. The topics of interest of community users, often biased towards subjective ones, are not necessarily the same as those of web users. Table 3 (presented in Section 3.1) shows the label distributions of questions from community users and web users. SI favors questions that require objective answers (such as medical and business related), whereas community users tend to favor qeustions with subjective answers.

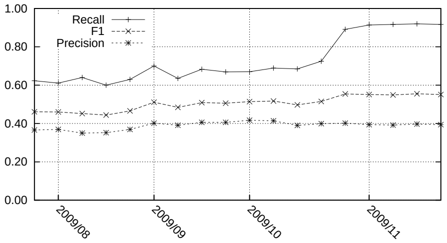


Figure 9: Precision, recall and F1 measure of QL .

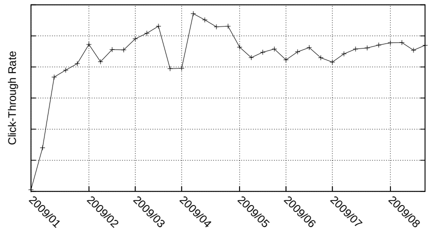


Figure 10: CTR of recommended questions by QR.

4.3 Question Labeling

We measure the effectiveness of QL by user feedback. For each new question, QL outputs the suggested labels as an ordered list. The user can choose from the recommended labels, or add new labels manually. The user can choose at most five labels for one question. We keep the log of what labels are suggested and what labels the users used, so as to make automatic evaluation possible.

We use precision, recall, and F1 to measure the effectiveness of QL . Precision measures the correctness of suggested labels. For each question, precision is defined as $(|L_S| \cap |L_U|)/|L_S|$, where L_S is the set of suggested labels and L_U is the set of user-chosen labels. Recall measures the completeness of suggested labels and it is defined as $(|L_S| \cap |L_U|)/|L_U|$. F1 is the geometric mean of precision and recall, computed as $2PR/(P+R)$. Precision, recall, and F1 are widely used metrics for recommender systems. Since at most five labels are allowed, we compute precision@5, recall@5 and F1@5.

Figure 9 shows the precision, recall, and F1 of QL for Confucius China. First, the recall of QL was high and increasing, from 0.6 to near 1. Higher recall means that users tend to select from among the recommended labels instead of creating new ones. As QL only recommends labels with high frequency, such behavior helps to limit/reduce the size of the label vocabulary. Considering that the mean number of labels for a question is 2.32 (which suggests that users prefer fewer labels), the precision at around 0.4 is already quite promising.

4.4 Question Recommendation

Figure 10 shows the CTR of recommended questions after the launch of QR . The higher the CTR, the more useful the recommended questions to the user. As we can see, the CTR increased quickly after the launch, then stabilized after the first month. When stablized, the CTR is quite high (around 0.38). The rapid adoption of QR means that the users found it helpful in identifying related questions. QR brings two benefits to Confucius. First, if the user finds a related question on the question submission page, he/she

Table 5: Side-by-side experimental results for UR on improving search results. All control groups are Google Search results. Impact indicates the fraction of queries that differ in quality between the experimental and control groups. Mean score is the sum of rating scores divided by the number of queries with agreement. Larger mean score is better.

Experimental Groups	Number of Queries at Different Ratings									Impact	Mean Score
	-2	-1.5	-1	-0.5	0	0.5	1	1.5	2		
UR without graph	0	1	0	18	237	31	0	0	0	0.174	0.017
UR without weight	0	6	2	30	178	44	9	6	0	0.353	0.050
UR without topic	5	6	9	16	205	21	21	4	5	0.298	0.039
Full UR	4	8	6	17	230	31	16	6	5	0.215	**0.054**

Table 6: The accuracy of best/non-best answer classification, averaged over 10-fold cross-validation.

Algorithm	Accuracy
Random Forest	**0.770**
AdaBoost	0.753
Logistic Regression	0.739
SVMs (RBF)	0.698
SVMs (polynomial)	0.696
SVMs (linear)	0.661
Perceptron	0.641
SVMs (sigmoid)	0.527
Naive Bayes	0.499

may get to the answer directly without submitting the question. This reduces the redundancy in the Q&A database. Second, within the community, active users are the source of new answers. More clicks on QR links keep the users engaged within the community for longer periods of time, thus increasing the probability that they contribute answers.

4.5 Answer Quality Assessment

We prepared the training data for AQ from user voted Q&A threads. Specifically, we used the Q&A threads that contain asker-selected best answers. Threads that contain less than 3 answers were not used, since the less participation, the more risk of it being spam. For each thread, we used its best answer as the positive example, and randomly sampled an answer not marked as the best as the negative example. We collected $100,000$ pairs of positive and negative samples for the experiment.

As described in Section 3.4, AQ uses a data-driven approach to select the best algorithm for answer quality prediction. Thanks to our massive distributed computing facility, we could test a number of algorithms in a short time, covering all possible parameter choices. We included Naive Bayes, Logistic Regression, Decision Tree, Perceptron, AdaBoost, and SVMs in the selection pool. For SVMs, we used common kernels, including linear kernel, RBF kernel, polynomial kernel, and sigmoid kernel. Table 6 lists the 10-fold cross-validation accuracy of best/non-best answer classification, showing each algorithm at its best across different parameter values.

The best result came from the Random Forest, a Decision Tree-based probabilistic classifier, which outperformed the others, yielding an accuracy of 0.77. We noticed that the non-linear methods, such as Random Forest and AdaBoost, outperformed linear ones. This result suggests that the target prediction function is not linear.

4.6 User Ranking

To test UR, we incorporated it into web search result ranking. Lacking internal links, community Q&A sites, such as Confucius, can barely benefit from PageRank or similar link-based ranking algorithms. We used UR as the query-independent quality indicator for web search, and performed a side-by-side experiment to examine the changes in the search results.

We sampled 300 queries from Confucius' search log, and retrieved the corresponding search results from Google. We used site search restrictions[2] to limit the result to the Q&As from Confucius. For each search result, we combined the relevance score and UR score into a reordering score $\lambda Rel + (1 - \lambda) UserRank$, where the linear mixing factor $\lambda = 0.9$ was selected by a small scale cross-validation.

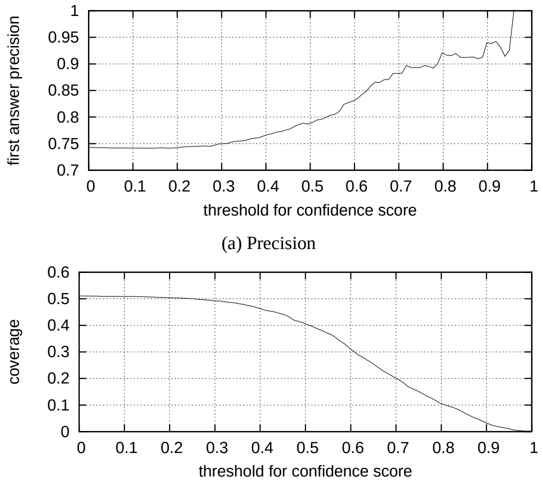
We used a side-by-side experiment to evaluate the effectiveness of UR. Our side-by-side experiments were a form of double-blind testing. Two groups of search results were shown to human raters in the same format and in random order. In the experimental group, the search results were ordered by the combined score, while the control group used the original ordering of Google search. The raters were not able to identify which one is the experimental group. They were given the task to assign each query a score: Score value 2 means that side A is much better than side B, -2 means that side B is much better than A, and other scores captures levels in between. We let multiple raters rate the same query until there was an agreement, or at most three raters had participated but still had no agreement.

UR scores are determined by three factors: the graph of users, the quality of answers, and the topic of interactions. We also evaluated each factor's contribution by temporarily remove them from the computation. Table 5 shows the results of the four side-by-side experiments. Impact measures how many results are affected by the reordering and it is computed as $|Q_{nonzero}|/|Q_{agreed}|$, where $|Q_{nonzero}|$ is the number of agreed queries with non-zero ratings and $|Q_{agreed}|$ is the total number of agreed queries. The greater the impact, the more queries are affected. Mean score measures the improvements for the agreed queries and it is computed as $(S_+ + S_-)/|Q_{agreed}|$, where S_+ is the sum of positive rating scores, and S_- is the sum of negative rating scores. The higher the mean score, the better the experimental group is, compared to the control group. Queries with rating disagreement are ignored when computing impact and mean score. All control groups are Google Search results. "UR without graph" means computing UR without the user activity graph; "UR without weight" means computing UR without the quality weight on each edge; "UR without topic" means computing topic-independent rank scores for each user.

4.7 NLQ&A

We evaluated the NL subroutine by a set of 2500 questions sampled from web queries. We used only the queries that would trigger SI to ensure that they are questions. We use precision and coverage as the evaluation metrics. Precision measures the proportion of correctly answered questions among all questions that have been answered. Coverage measures the proportion of answered questions

[2] Added `site:wenda.tianya.cn` to each Google query. `wenda.tianya.cn` is the URL of Confucius China.



(a) Precision

(b) Coverage

Figure 11: Precision (a) and coverage (b) of NL.

among all questions. We asked three human raters to rate each answer provided by NL as good or bad. We treated an question as correctly answered only if all raters said its top-ranked answer was good.

Precision and coverage are not weighted equally in practical applications. In Confucius, we favor the precision over the coverage, since giving the wrong answer hurts the user experience more than keeping silence. We used the threshold mentioned in Section 3.6 to control the tradeoff, that if the confidence score of the top-ranked answer was below the threshold, NL would refuse to answer that question.

Figure 11 shows the precision and coverage of our preliminary NL subroutine with respect to different threshold values. The precision increased as the threshold went higher (Figure 11 (a)), while the coverage drops (Figure 11 (b)). When using 0.8 as the threshold, NL can answer 10% of the questions with the precision 0.9, which is an acceptable performance for Confucius.

5. DISCUSSIONS

Though significant progress has been made, several challenges remain to be tackled to maintain high service quality.

Opinion Questions

Many users ask for recommendations on things to buy or for advice on life matters. Examples include "Who are more dependable, women or men?" and "Which camera brand should I go with, Canon or Nikon?." We call these types of questions *opinion questions*. Most opinion questions do not have an objective best answer. At best, there may exist a handful of main-stream opinions, with a large number of users supporting each. At worst, users may have significantly diverging subjective opinions. Each party may then list some supportive evidence, which can trigger a debate in the thread. If such opinion questions are ever closed, the best answer is typically chosen based on the asker's arbitrary perception of the outcome of the argument.

Opinion questions present challenges to automatic answer quality assessment. The model training step presented in Section 3.4 relies on best answers as positive samples and non-best answers as negative samples. However, for opinion questions, the distinction between a best and a non-best answers is subjective. To handle them adequately, we would need a way of capture the asker's subjectivity, which is not possible with the current set of features. Indeed, to our best knowledge, constructing complex user models that encapsulate such bias remains a grand open problem in natural language processing.

Factoid Questions

We rely on relevance, coverage, and originality for AQ. However, for certain factoid questions, these factors may be insufficient. Factoid questions — about weather conditions, the solution to a particular math problem, or driving directions — typically have an objective, short, and specific "correct" answer. For instance, the answer to "*Did it snow in Beijing yesterday?*" is simply yes or no.

First, relevance does not work for these extremely short answers. Both word-relevance and topic-relevance require some topically related terms to appear in the answer. Yes, no or the number 43 are not likely to be found in the question, nor do they provide any evidence for topic inference. Second, higher coverage does not necessarily mean better answer quality for these questions. In contrast, long answers for simple factoid questions may be spam. Third, since the form of the answer to simple factoid questions is quite rigid, most answers will be identified as copies, although the answerer is earnest.

There are possible ways to deal with factoid answer assessment. First, we can leverage the question classifier in NL to decide a question's type. Then, if the question type is factoid, we can check if the answer is within the desired class and format. For instance, when checking the answers to "*Did it snow in Beijing yesterday?*", we can expect a positive or negative reply, like "*yes*" or "*didn't*." NL also provides a way to assess answers to factoid questions. Though not perfect, NL's accuracy on simple factoid questions is promising, as reported in TREC QA [4]. Checking the answers with NL's output might help judge the quality of the answer.

Best Answer Spam

Early in the history of Confucius, spammers identified best answers as a prime spam target. As we rely on best answer labels in training, our ranking is somewhat susceptible to this type of spamming. In order to generate fake best answers, a spammer creates multiple user accounts first. Then, he/she uses some of the accounts to ask questions, and others to provide answers (most of which are typically concealed ads). If uncaught, spammers may generate prodigious amounts of fake best answers, which could have a non-trivial impact on the quality of the machine learning model.

When the spammer's agenda is direct advertising, we can often identify telltale signs of his/her activity: repeated phone numbers or URLs help us detect much of the best answer spam. However, when the spammer's intention is to obtain higher status within the community, and the perks that come with it, the spam content may lack easily identifiable patterns. Such a spammer may post low quality answers to his/her own questions, and select those as best, despite the presence of other, truly better answers in the thread.

There are clues that may help identify the best answer spammers. First, the spammers have an incredible high best answer rate, compared to normal users. Screening the best answer rate list of users can thus help identify them. Second, to be efficient, best answer spammers tend to answer their own question quickly. We can cluster the users by their time of receiving best answer label to find the spammers.

Question Spam

Spammers also target questions. First, they find a large document collection to use as the source, such as an FAQ list or an archive of historical documents (common sources include history/geography documents and government reports). Then, they use one account to ask questions replicating the headings in the document collection, and use another account to submit corresponding document sections as the answer. While such question-answer pairs are a perfect match, they do little to help the community beyond boosting the spammers' reputation.

Question spam will often result in irregular patterns in the distributions of both timestamps and interaction counts with different users. These patterns help us identify spammers when performing manual evaluation. We are considering adopting automatic routines once the patterns stabilize.

Answerer Targeting

The most prominent characteristics of a Q&A site are anchored in its answer collection. Without users providing high-quality and timely answers, the system surely alienates its adopters quickly. To motivate users to provide high-quality answers, a Q&A system should provide certain incentives. In this respect, Wikipedia may be considered a most successful product (even if not a classical Q&A system), which attracts volunteers to provide quality content via authorship honor. However, transferring that success to a Q&A site faces several challenges. First, Wikipedia decides its "questions" whereas Q&A accepts questions. Second, the quality bar for Wikipedia is set high, and the low quality content can be edited away easily. Users who are not qualified to provide quality content are discouraged and eventually turned away. In comparison, the quality bar for a Q&A site is much lower. Virtually anyone can answer a question. Therefore, both the question flow and the answer flow of a Q&A system are more "open" than Wikipedia—and this openness inevitably reduces overall content quality.

To address the quality challenge, one can consider providing more explicit virtual or monetary incentives. Such incentives often invite spammers. One way to deal with low-quality content and spam is to strengthen content ranking algorithms so that low-quality questions and answers can be "buried" while high-quality posts are promoted. Another way to improve quality is to target experts or friends as potential answerers. Targeting experts can surely improve quality, but experts may demand incentives. Targeting friends may or may not be helpful, depending on the goal of the service. If the primary goal is to collect content to augment web search, targeting friends, which typically yields subjective content, may not be particularly beneficial. At the same time, if the goal is to increase usage, targeting friends may turn a Q&A site into a forum site—trading content quality for higher degree of user interaction. Nevertheless, no matter what the goal is, our developed algorithms can be adjusted to meet the system requirements.

6. CLOSING REMARKS

In this paper, we presented our mature Q&A service along with its six key subroutines. We pointed out the primary design goals of Confucius and showed how the subroutines had helped to meet them. We also discussed how the subroutines could be adjusted to position the service as being oriented more toward social interactions or toward high-quality content generation.Our future work will focus on improving the key subroutines.

Acknowledgement

We would like to thank the engineering, research, and product management teams for their contributions. In particular, we would like to thank Jim Deng, Feng Hong, Greg Coladonato, Jerry Smith, Jun Wu, Dekang Lin and Shamayn Teh for their innovative ideas.

7. REFERENCES

[1] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. Finding high-quality content in social media. In *Proceedings of WSDM '08*, pages 183–194, 2008.

[2] D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent Dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

[3] S. Brin, L. Page, R. Motwani, and T. Winograd. The PageRank citation ranking: bringing order to the web. In *Proceedings of ASIS98*, pages 161–172, 1998.

[4] H. T. Dang, D. Kelly, and J. Lin. Overview of the TREC 2007 question answering track. In *NIST Special Publication: SP 500-274 The Sixteenth Text REtrieval Conference (TREC 2007) Proceedings*, 2007.

[5] T. Hofmann. Probabilistic latent semantic analysis. In *Proceedings of UAI'99*, 1999.

[6] D. Horowitz and S. D. Kamvar. Anatomy of a large-scale social search engine. In *Proceedings of WWW'2010*, 2010.

[7] J. Jeon, W. B. Croft, and J. H. Lee. Finding similar questions in large question and answer archives. In *Proceedings of CIKM'05*, pages 84–90, 2005.

[8] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5), 1999.

[9] J. Ko, E. Nyberg, and L. Si. A probabilistic graphical model for joint answer ranking in question answering. In *Proceedings of SIGIR'07*, 2007.

[10] Y. S. Lai, K. A. Fung, and C. H. Wu. FAQ mining via list detection. In *Proceedings of the Workshop on Multilingual Summarization and Question Answering*, 2002.

[11] F. Li, X. Zhang, J. Yuan, and X. Zhu. Classifying what-type questions by head noun tagging. In *Proceedings of COLING 2008*, 2008.

[12] Zhiyuan Liu, Yuzhou Zhang, Edward Y. Chang, and Maosong Sun. PLDA*: Parallel latent dirichlet allocation with data placement and pipeline processing. *ACM Transactions on Intelligent Systems and Technology (Accepted)*, 2010.

[13] V. Murdock, D. Kelly, W.B. Croft, N.J. Belkin, and X. Yuan. Identifying and improving retrieval for procedural questions. *Information Processing and Management*, 43(1), 2007.

[14] A.Y. Ng, A.X. Zheng, and M.I. Jordan. Stable algorithms for link analysis. In *Proceedings of SIGIR'01*, 2001.

[15] K. Papineni, S. Roukos, T. Ward, and W. J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL'02*, pages 311–318, 2001.

[16] D. Radev, W. Fan, H. Qi, H. Wu, and A. Grewal. Probabilistic question answering on the web. In *Proceedings of WWW '02*, 2002.

[17] X. Si, Z. Gyongyi, E. Y. Chang, and M.S. Sun. Scalable mining of topic-dependent user reputation for improving user generated content search quality. Technical report, Google, 2010.

[18] M. Surdeanu, M. Ciaramita, and H. Zaragoza. Learning to rank answers on large online QA collections. *Proceedings of ACL-08: HLT*, 2008.

[19] X. Xue, J. Jeon, and W. B. Croft. Retrieval models for question and answer archives. In *Proceedings of SIGIR '08*, 2008.

[20] X. Zhang, Y. Hao, X. Zhu, M. Li, and D. R. Cheriton. Information distance from a question to an answer. In *Proceedings of KDD '07*, pages 874–883, 2007.