Comparable Entity Mining from Comparative Questions

Shasha Li, Chin-Yew Lin, *Member*, *IEEE*, Young-In Song, and Zhoujun Li, *Member*, *IEEE*

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

Index Terms—Information extraction, bootstrapping, sequential pattern mining, comparable entity mining

1 Introduction

Comparing alternative options is one essential step in decision-makings that we carry out every day. For example, if someone is interested in certain products or services such as digital cameras or treatments, he or she would want to know what the alternatives are and how they compare to each other before making a purchase decision. This type of comparison activity is very common in our daily life but requires high knowledge skill. Magazines such as *Consumer Reports* and *PC Magazine* and online media such as *CNet.com* strive in providing editorial comparison content and surveys to satisfy this need.

In the World Wide Web era, a comparison activity typically involves: search for relevant webpages containing information about the targeted products, find competing products, read reviews, and identify pros and cons. In this paper, we focus on finding a set of comparable entities given a user's input entity. For example, given an entity, Nokia N95 (a cellphone), we want to find comparable entities such as Nokia N82, iPhone and so on.

In general, it is difficult to decide if two entities are comparable or not since people do compare apples and oranges for various reasons. For example, "Ford" and "BMW" might be comparable as "car manufacturers" or as "market segments that their products are targeting," but we rarely see people comparing "Ford Focus" (car model) and "BMW 328i." Things also get more complicated when an entity has several functionalities. For example, one might compare "iPhone" and "PSP" as "portable game player" while compare "iPhone" and "Nokia N95" as "mobile phone." Fortunately, plenty of comparative questions are posted online, which provide evidences for what people want to compare, e.g., "Which to buy, iPod or iPhone?". We call "iPod" and "iPhone" in this example as comparators. In this paper, define comparative questions and comparators as

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

According to these definitions, Q1 and Q2 below are not comparative questions while Q3 is. "iPod Touch" and "Zune HD" are comparators.

Q1. "Which one is better?"
Q2. "Is Lumix GH-1 the best camera?"
Q3. "What's the difference between iPod Touch and Zune HD?"

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

To our best knowledge, this is the first attempt to specially address the problem on finding good comparators

- *S. Li is with the Department of Computer Science, National University of Defense Technology, Hunan 410073, China. E-mail: shashali@nudt.edu.cn.*
- *C.-Y. Lin and Y.-I. Song are with the Microsoft Research Asia, Beijing 100190, China. E-mail: {cyl, yosong}@microsoft.com.*
- *Z. Li is with the State Key Laboratory of Software Development Environment and Beijing Key Laboratory of Network Technology, Beihang University, Beijing 100191, China. E-mail: lizj@buaa.edu.cn.*

Manuscript received 16 Oct. 2010; revised 25 July 2011; accepted 29 Aug. 2011; published online 5 Oct. 2011.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2010-10-0552. Digital Object Identifier no. 10.1109/TKDE.2011.210.

to support users' comparison activity. We are also the first to propose using comparative questions posted online that reflect what users truly care about as the medium from which we mine comparable entities. Our weakly supervised method achieves 82.5 percent F1-measure in comparative question identification, 83.3 percent in comparator extraction, and 76.8 percent in end-to-end comparative question identification and comparator extraction which outperform the most relevant state-of-the-art method by Jindal and Liu [7] significantly.

The rest of this paper is organized as follows: the next section discusses previous works. Section 3 presents our weakly supervised method for comparator mining. Section 4 discussed how to rank comparable entities for a user's input entity and build a comparator database. Section 5 reports the evaluations of our techniques, and we conclude the paper and discuss future work in Section 6.

2 Related Work

2.1 Overview

In terms of discovering related items for an entity, our work is similar to the research on recommender systems, which recommend items to a user. Recommender systems mainly rely on similarities between items and/or their statistical correlations in user log data [9]. For example, Amazon recommends products to its customers based on their own purchase histories; similar customers' purchase histories, and similarity between products. However, recommending an item is not equivalent to finding a comparable item. In the case of Amazon, the purpose of recommendation is to entice their customers to add more items to their shopping carts by suggesting similar or related items. While in the case of comparison, we would like to help users explore alternatives, i.e., helping them make a decision among comparable items.

For example, it is reasonable to recommend "iPod speaker" or "iPod batteries" if a user is interested in "iPod," but we would not compare them with "iPod." However, items that are comparable with "iPod" such as "iPhone" or "PSP" which were found in comparative questions posted by users are difficult to be predicted simply based on item similarity between them. Although they are all music players, "iPhone" is mainly a mobile phone, and "PSP" is mainly a portable game device. They are similar but also different therefore beg comparison with each other. It is clear that comparator mining and item recommendation are related but not the same.

Our work on comparator mining is related to the research on entity and relation extraction in information extraction [1], [2], [15], [16], [17]. Specifically, the most relevant work is by Jindal and Liu [6], [7] on mining comparative sentences and relations. Their methods applied class sequential rules (CSRs) [6] and label sequential rules (LSRs) [6] learned from annotated corpora to identify comparative sentences and extract comparative relations respectively in the news and review domains. The same techniques can be applied to comparative question identification and comparator mining from questions. However, their methods typically can achieve high precision but suffer from low recall [7]. However, ensuring high recall is crucial in our intended application scenario where users can issue arbitrary queries. To address this problem, we develop a weakly supervised bootstrapping pattern learning method by effectively leveraging unlabeled questions.

Bootstrapping methods have been shown to be very effective in previous information extraction research [8], [11], [14], [15], [16]. Our work is similar to them in terms of methodology using bootstrapping technique to extract entities with a specific relation. However, our task is different from theirs in that it requires not only extracting entities (comparator extraction) but also ensuring that the entities are extracted from comparative questions (comparative question identification), which is generally not required in IE task.

2.2 Jindal and Liu [6], [7]

In this section, we provide a brief summary of the comparative mining method proposed by Jindal and Liu [6], [7], which is used as baseline for comparison and represents the state-of-the-art in this area. We first introduce the definition of CSR and LSR rule used in their approach, and then describe their comparative mining method. Readers should refer to Jindal and Bing's Method (J&L's) original papers for more details.

2.2.1 CSR and LSR

CSR is a classification rule. It maps a sequence pattern $S(s_1 s_2 \ldots s_n)$ (a class C. In our problem, C is either comparative or noncomparative. Given a collection of sequences with class information, every CSR is associated to two parameters: support and confidence. Support is the proportion of sequences in the collection containing S as a subsequence. Confidence is the proportion of sequences labeled as C in the sequences containing the S. These parameters are important to evaluate whether a CSR is reliable or not.

LSR is a labeling rule. It maps an input sequence pattern $S(s_1 s_2 \ldots s_i \ldots s_n)$ to a labeled sequence $S'(s_1 s_2 \ldots l_i \ldots s_n)$ by replacing one token s_i in the input sequence with a designated label (l_i). This token is referred as the anchor. The anchor in the input sequence could be extracted if its corresponding label in the labeled sequence is what we want (in our case, a comparator). LSRs are also mined from an annotated corpus, therefore each LSR also have two parameters: support and confidence. They are similarly defined as in CSR.

2.2.2 Supervised Comparative Mining Method

J&L treated comparative sentence identification as a classification problem and comparative relation extraction as an information extraction problem. They first manually created a set of 83 keywords such as beat, exceed, and outperform that are likely indicators of comparative sentences. These keywords were then used as pivots to create part-of-speech (POS) sequence data. A manually annotated corpus with class information, i.e., comparative or noncomparative, was used to create sequences and CSRs were mined. A Naïve Bayes classifier was trained using the CSRs as features. The classifier was then used to identify comparative sentences.

Given a set of comparative sentences, J&L manually annotated two comparators with labels $ES1 and $ES2 and

TABLE 1
Candidate Indicative Extraction Pattern (IEP) Examples of the Question "Which City Is Better, NYC or Paris?"

Sequential Patterns
<#start which city is better, $C or $C ? #end>
<, $C or $C ? #end>
<#start $C/NN or $C/NN ? #end>
<which NN is better, $C or $C ?>
<which city is JJR, $C or $C ?>
<which NN is JJR, $C or $C ?>
...

the feature compared with label $FT for each sentence. J&L's method was only applied to noun and pronoun. To differentiate noun and pronoun that are not comparators or features, they added the fourth label $NEF, i.e., nonentity-feature. These labels were used as pivots together with special tokens li & rj[1] (tokens at specific positions), #start (beginning of a sentence), and #end (end of a sentence) to generate sequence data, sequences with single label only and minimum support greater than 1 percent are retained, and then LSRs were created. When applying the learned LSRs for extraction, LSRs with higher confidence were applied first.

J&L's method have been proved effective in their experimental setups. However, it has the following weaknesses:

- The performance of J&L's method relies heavily on a set of comparative sentence indicative keywords. These keywords were manually created and they offered no guidelines to select keywords for inclusion. It is also difficult to ensure the completeness of the keyword list.
- Users can express comparative sentences or questions in many different ways. To have high recall, a large annotated training corpus is necessary. This is an expensive process.
- Example CSRs and LSRs given in Jindal and Liu [7] are mostly a combination of POS tags and keywords. It is a surprise that their rules achieved high precision but low recall. They attributed most errors to POS tagging errors. However, we suspect that their rules might be too specific and overfit their small training set (about 2,600 sentences). We would like to increase recall, avoid overfitting, and allow rules to include discriminative lexical tokens to retain precision.

In the next section, we introduce our method to address these shortcomings.

3 Weakly Supervised Method for Comparator Mining

Our weakly supervised method is a pattern-based approach similar to J&L's method, but it is different in many aspects: instead of using separate CSRs and LSRs, our method aims to learn sequential patterns which can be used to identify comparative question and extract comparators simultaneously.

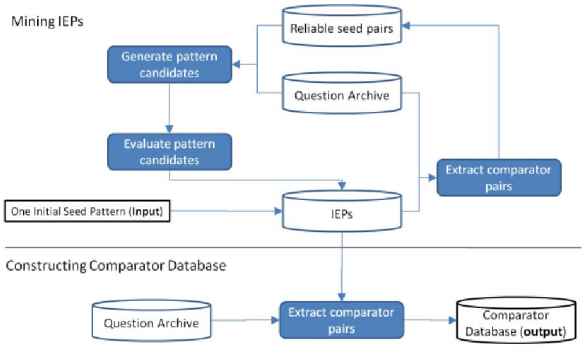

Fig. 1. Overview of the bootstrapping algorithm.

In our approach, a sequential pattern is defined as a sequence $S(s_1 s_2 \ldots s_i \ldots s_n)$ where s_i can be a word, a POS tag, or a symbol denoting either a comparator ($C), or the beginning (#start) or the end of a question (#end). A sequential pattern is called an indicative extraction pattern (IEP) if it can be used to identify comparative questions and extract comparators in them with high reliability. We will formally define the reliability score of a pattern in the next section.

Once a question matches an IEP, it is classified as a comparative question and the token sequences corresponding to the comparator slots in the IEP are extracted as comparators. When a question can match multiple IEPs, the longest IEP is used.[2] Therefore, instead of manually creating a list of indicative keywords, we create a set of IEPs. We will show how to acquire IEPs automatically using a bootstrapping procedure with minimum supervision by taking advantage of a large unlabeled question collection in the following sections. The evaluations shown in Section 5 confirm that our weakly supervised method can achieve high recall while retain high precision.

This pattern definition is inspired by the work of Ravichandran and Hovy [14]. Table 1 shows some examples of such sequential patterns. We also allow POS constraint on comparators as shown in the pattern "<, $C/NN or $C/NN ? #end|>". It means that a valid comparator must have an NN POS tag.

3.1 Mining Indicative Extraction Patterns

Our weakly supervised IEP mining approach is based on two key assumptions.

- If a sequential pattern can be used to extract many reliable comparator pairs, it is very likely to be an IEP.
- If a comparator pair can be extracted by an IEP, the pair is reliable.

Based on these two assumptions, we design our bootstrapping algorithm as shown in Fig 1. The bootstrapping process starts with a single IEP. From it, we extract a set of

1. l_i marks a token is at the ith position to the left of the pivot and r_j marks a token is at jth position to the right of the pivot where i and j are between 1 and 4 in J&L [7].

2. It is because the longest IEP is likely to be the most specific and relevant pattern for the given question.

Algorithm 1 Weakly-Supervised Model

Input: CP, G
Initialize solution: $Q \leftarrow \{\}, P \leftarrow \{\}, P_{new} \leftarrow \{\}, CP_{new} \leftarrow CP$
1. **Repeat**
2. $P \leftarrow P + P_{new}$
3. $Q_{new} \leftarrow ComparativeQuestionIdentify(\ CP_{new})$
4. $Q \leftarrow Q\ +\ Q_{new}$
5. **for** $q_i \in G$ **do**
6. **if** $IsMatchExistingPatterns(P, q_i)$ **then**
7. $Q \leftarrow Q - q_i$
8. **end if**
9. **end for**
10. $P_{new} \leftarrow MineGoodPatterns(Q)$
11. $CP_{new} \leftarrow \{\}$
12. **for** $q_i \in G$ **do**
13. $cp \leftarrow ExtractComparableComparators(P, qi)$
14. **if** $cp \neq NULL$ **and** $cp \notin CP$ **then**
15. $CP_{new} \leftarrow\ CP_{new} + \{cp\}$
16. **end if**
17. **end for**
18. **until** $P_{new} = \{\}$
19. **return** P

Fig. 2. Pseudocode of the bootstrapping algorithm.

initial seed comparator pairs. For each comparator pair, all questions containing the pair are retrieved from a question collection and regarded as comparative questions. From the comparative questions and comparator pairs, all possible sequential patterns are generated and evaluated by measuring their reliability score defined later in the Pattern Evaluation section. Patterns evaluated as reliable ones are IEPs and are added into an IEP repository.

Then, new comparator pairs are extracted from the question collection using the latest IEPs. The new comparators are added to a reliable comparator repository and used as new seeds for pattern learning in the next iteration. All questions from which reliable comparators are extracted are removed from the collection to allow finding new patterns efficiently in later iterations. The process iterates until no more new patterns can be found from the question collection.

The pseudocode of the algorithm is shown in Fig. 2.

There are two key steps in our method: 1) pattern generation and 2) pattern evaluation. In the following sections, we will explain them in details.

3.1.1 Patterns Generation

To generate sequential patterns, we adapt the surface text pattern mining method introduced in [14]. For any given comparative question and its comparator pairs, comparators in the question are replaced with symbol $Cs. Two symbols, #start and #end, are attached to the beginning and the end of each sentence in the question. To reduce diversity of sequence data and mine potential patterns, phrase chuncking is applied. In this paper, we have just simply tried to use some heuristic rules for the task. Those rules are listed in Table 2.

TABLE 2
Heuristic Rules for Phrase Chuncking

Heuristic rules for phrase chuncking
NP*->NP
NN*->NN
NN+NNS->NNS
NP+NPS->NPS
More+ADJ->JJR
Most+ADJ->JJS
...

Then, the following three kinds of sequential patterns are generated from sequences of questions:

- **Lexical patterns.** Lexical patterns indicate sequential patterns consisting of only words and symbols ($C, #start, and #end). They are generated by suffix tree algorithm [3] with two constraints: a pattern should contain more than one $C, and its frequency in collection should be more than an empirically determined number β.
- **Generalized patterns.** A lexical pattern can be too specific. Thus, we generalize lexical patterns by replacing one or more words/phrases with their POS tags. $2^n - 1$ generalized patterns can be produced from a lexical pattern containing N words excluding $Cs.
- **Specialized patterns.** In some cases, a pattern can be too general. For example, although a question "ipod or zune?" is comparative, the pattern "<$C or $C>" is too general, and there can be many noncomparative questions matching the pattern, for instance, "true or false?". For this reason, we perform pattern specialization by adding POS tags to all comparator slots. For example, from the lexical pattern "<$C or $C>" and the question "ipod or zune?", "<$C/NN or $C/NN?>" will be produced as a specialized pattern.

Note that generalized patterns are generated from lexical patterns and the specialized patterns are generated from the combined set of generalized patterns and lexical patterns. The final set of candidate patterns is a mixture of lexical patterns, generalized patterns and specialized patterns.

3.1.2 Pattern Evaluation

According to our first assumption, a reliability score $\mathrm{R}^{\mathrm{k}}(\mathrm{p_i})$ for a candidate pattern $\mathrm{p_i}$ at iteration k can be defined as follows:

$$\mathrm{R}^{\mathrm{k}}(\mathrm{p_i}) = \frac{\sum_{\forall \mathrm{cp_j} \in \mathrm{CP}^{\mathrm{k-1}}} \mathrm{N_Q}(\mathrm{p_i} \rightarrow \mathrm{cp_j})}{\mathrm{N_Q}(\mathrm{p_i} \rightarrow *)}, \tag{1}$$

where $\mathrm{p_i}$ can extract known reliable comparator pairs $\mathrm{cp_j}$. $\mathrm{CP}^{\mathrm{k-1}}$ indicates the reliable comparator pair repository accumulated until the $(\mathrm{k}-1)\mathrm{th}$ iteration. $\mathrm{N_Q(x)}$ means the number of questions satisfying a condition x. The condition $\mathrm{p_i} \rightarrow \mathrm{cp_j}$ denotes that $\mathrm{cp_j}$ can be extracted from a question by applying pattern $\mathrm{p_i}$ while the condition $\mathrm{p_i} \rightarrow *$ denotes any question containing pattern $\mathrm{p_i}$.

However, (1) can suffer from incomplete knowledge about reliable comparator pairs. For example, very few reliable pairs are generally discovered in early stage of bootstrapping. In this case, the value of (1) might be underestimated which could affect the effectiveness of (1)

on distinguishing IEPs from nonreliable patterns. We mitigate this problem by a lookahead procedure. Let us denote the set of candidate patterns at the iteration k by $\hat{P}^k$. We define the support S for comparator pair $\widehat{cp}_i$ which can be extracted by $\hat{P}^k$ and does not exist in the current reliable set

$$S(\widehat{cp}_i) = N_Q(\hat{P}^k \rightarrow \widehat{cp}_i), \quad (2)$$

where $\hat{P}^k \rightarrow \widehat{cp}_i$ means that one of the patterns in $\hat{P}^k$ can extract $\widehat{cp}_i$ in certain questions. Intuitively, if $\widehat{cp}_i$ can be extracted by many candidate patterns in $\hat{P}^k$, it is likely to be extracted as a reliable one in the next iteration. Based on this intuition, a pair $\widehat{cp}_i$ whose support S is more than a threshold α is regarded as a likely reliable pair. Using likely reliable pairs, lookahead reliability score $\hat{R}(p_i)$ is defined

$$\hat{R}^k(p_i) = \frac{\sum_{\forall \widehat{cp}_i \in \widehat{CP}^k_{rel}} N_Q(p_i \rightarrow \widehat{cp}_i)}{N_Q(p_i \rightarrow *)}, \quad (3)$$

where $\widehat{CP}^k_{rel}$ indicates a set of likely reliable pairs based on $\hat{P}^k$.

By interpolating (1) and (3), the final reliability score $R(p_i)^k_{final}$ for a pattern is defined as follows:

$$R(p_i)^k_{final} = \lambda \cdot R^k(p_i) + (1 - \lambda) \cdot \widehat{R}^k(p_i). \quad (4)$$

Using (4), we evaluate all candidate patterns and select patterns whose score is more than threshold γ as IEPs. All necessary parameter values are empirically determined. We will explain how to determine our parameters in Section 4.

3.2 Comparator Extraction

By applying learned IEPs, we can easily identify comparative questions and collect comparator pairs from comparative questions existing in the question collection. Given a question and an IEP, the details of the process for comparator extraction are shown as follows:

1. ""Generate sequence for the comparative question. If the IEP is a pattern without generalization, we just need to tokenize the questions and the sequence is the list of resulted tokens. Otherwise, phrase chuncking is necesary. The sequence is a list of resulted chunks. Take the question "Which is better phone iphone or nokia n95 ?" for example. If we apply the pattern "Which is better $C or $C?" to the question, we generate a sequence "which|is|better|phone|iphone|or|nokia|n95|?". If we apply pattern "Which is better NN $C or $C?" to the question, a sequence "Which/WP | is/VBS| better/JJR|phone/NN|iphone/NP|or/CC|nokia n95/NP|?/?".
2. Check whether sequence of the question matches the given pattern. If IEP is a specialized pattern, the POS tag sequence of extracted comparators should follow the constraints specified by the pattern.

However, about 67 percent comparative questions can match to multiple patterns, and from 11 percent comparative questions, we can extract different comparator pairs. Take the question "Which is better phone iphone or nokia n95 ?" for example. If we apply the pattern "Which is better $C or $C?" to the question, the wrong comparator pair, "phone iphone" and "nokia n95," will be extracted. However, if we use pattern "Which is better NN $C or $C?", we can gather the correct comparator pairs "iphone" and "nokia n95" from the question.

Thus, developing a strategy to determine which comparators to extract is necessary. Given a question, two factors are important for this issue.

1. How reliable that a pattern is when it is used to identify comparative question and extract comparators. Take the question "What do you prefer? Coke or Pepsi? And the reason?" for example, both the pattern "? $c or $c ?" and "NN or $c ? and $c ?" can be applied to the question. However, the former one is more reliable than the later one. In the example, when the former one is applied to the question, the extracted comparators ("Coke" and "Pepsi") are more reliable.
2. How well pattern is suitable for a question. Take the question "Which is better, iphone or mytouch or HTC G2?" for example, both the pattern ", $c or $c or NN?" and ", $c or $c ?" can be applied to the question. If we applied the pattern ", $c or $c or NN?", four tokens in the question can be exactly covered, namely "," "or" (the first one), "or" (the second one), and "?". However, if we use ", $c or $c ?", only three tokens can be exactly covered, namely ",", "or" (the first one), and "?". That means that the patter ", $c or $c ?" is more suitable for the example question.

According to above observation, we examined the following strategies:

- **Random strategy.** Given a question, randomly select a pattern among patterns which can be applied to the question.
- **Maximum length strategy.** Given a question, select the longest one among patterns which can be applied to the question. According to the discussion above, the longer the pattern is, the more tokens in the question can be exactly covered which means that the pattern is more suitable for the question.
- **Maximum reliability strategy.** Given a question, select the most reliable one among patterns which can be applied to the question.

4 Comparator Ranking

The remaining issue is to rank possible comparators for a user's input. The following ranking models are examined for this issue.

4.1 Comparability-Based Ranking Method

Intuitively, a comparator would be more interesting for an entity if it is compared with the entity more frequently. Based on this intuition, we define a simple ranking function $R_{freq}(c; e)$ which ranks comparators according to the number of times that a comparator c is compared to the user's input e in comparative question archive Q:

$$R_{freq}(c; e) = N(Q_{c,e}), \quad (6)$$

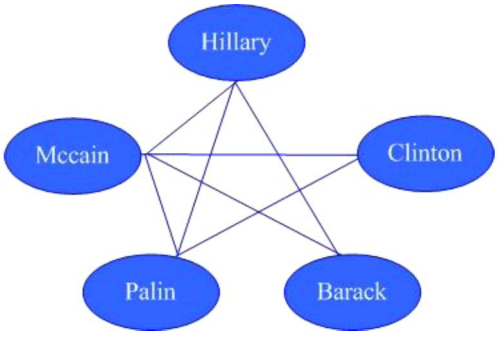

Fig. 3. Example graph for user's input "Obama."

where $Q_{c,q}$ is a set of questions from which c and e can be extracted as a comparator pair. In later, we will call this ranking function as Frequency-based Method.

Also, we can define another ranking function R_{rel} by combining reliability scores estimated in comparator mining phase

$$R_{rel}(c;e) = \sum_{q \in Q_{c,e}} R(p_{q,c,e}), \quad (7)$$

where $p_{q,c,e}$ means the pattern that is selected to extract comparator pair of c and e from question q in comparator mining phase. This ranking function will be denoted as Reliability-based Method.

4.2 Graph-Based Ranking Method

Though frequency is efficient for comparator ranking, the frequency-based method can suffer when an input occurs rarely in question collection; for example, suppose the case that all possible comparators to the input are compared only once in questions. In this case, the Frequency-based method may fail to produce a meaningful ranking result. Then, ***Representability*** should also be considered. We regard a comparator ***representive*** if it is frequently used as a baseline in the area the user is interested in. For example, when one wants to buy a smart phone and he/she is considering "Nokia N82," "Nokia N95" is the first one he/she wants to compare. That's because "Nokia N95" is a well-known smart phone and it's usually used as a baseline to help users know the performance of other smart phones better.

One possible solution to consider representability can be to use graph-based method such as PageRank. If a comparator is compared to many other important comparators which can be also compared to the input entity, it would be considered as a valuable comparator in ranking. Based on this idea, we examine PageRank algorithm to rank comparators for a given input entity which combine frequency and representability.

We define a graph $G = (V, E)$. In the graph, V is the set of nodes v, which consists of comparable comparators of the input. The edge e_{ij} between v_i and v_j means that the two comparators are compared in our comparator pair repository. Fig. 3 shows one example of a graph for an input entity "Obama."

A transition probability $P(v_i|v_j)$ is defined as follows:

$$P(v_i|v_j) = \frac{Count(v_i, v_j)}{Count(v_j, *)}, \quad (8)$$

$Count(v_i, v_j)$ is the frequency of comparator pairs v_i and v_j in our repository. For a user input entity e, the PageRank score $S(v_i)$ for node v_i is defined as follows [12]:

$$S^{(k)}(v_i) = \lambda \cdot S^{(0)}(v_i) + (1-\lambda) \cdot \sum P(v_i|v_j) \cdot S^{(k-1)}(v_j), \quad (9)$$

where

$$S^{(0)}(v_i) = \frac{Count(e, v_j)}{Count(e, *)}. \quad (10)$$

An initial score of each node is initialized as (10) and scores for nodes are iteratively calculated based on (9) until they are converged. In our algorithm, λ is set as 0.8. We will call this ranking method as PageRank-based method.

5 Experiments

5.1 Experiment Setup

5.1.1 Source Data

All experiments were conducted on about 60M questions mined from Yahoo! Answers' question title field. The reason that we used only a title field is that they clearly express a main intention of an asker with a form of simple questions in general.

5.1.2 Evaluation Data for Comparator Extraction

Two separate data sets were created for evaluation. First, we collected 5,200 questions by sampling 200 questions from each Yahoo! Answers category.[3] Two annotators were asked to label each question manually as comparative, noncomparative, or unknown. Among them, 139 (2.67 percent) questions were classified as comparative, 4,934 (94.88 percent) as noncomparative, and 127 (2.44 percent) as unknown questions which are difficult to assess. We call this set SET-A.

Because there are only 139 comparative questions in SET-A, we created another set which contains more comparative questions. We manually constructed a keyword set consisting of 53 words such as "or" and "prefer," which are good indicators of comparative questions. In SET-A, 97.4 percent of comparative questions contains one or more keywords from the keyword set. We then randomly selected another 100 questions from each Yahoo! Answers category with one extra condition that all questions have to contain at least one keyword. These questions were labeled in the same way as SET-A except that their comparators were also annotated. This second set of questions is referred as SET-B. It contains 853 comparative questions and 1,747 noncomparative questions. For comparative question identification experiments, we used all labeled questions in SET-A and SET-B. For comparator extraction experiments, we used only SET-B. All the remaining unlabeled questions (called as SET-R) were used for training our weakly supervised method.

Annotators are asked to annotate comparable comparators in the comparative questions in SET-B. Those comparators can be nouns/noun phrases, verb/verb phrases, pronouns, etc. The distribution of comparator in different part-of-speech types are shown in Table 3.

3. There are 26 top level categories in Yahoo! Answers.

TABLE 3
Distributions of Comparators of Different Types

Comparator Type	Number	Percentage
Nouns/Noun phrases	1,471	84.78%
Pronouns	3	0.17%
Verbs/verb phrases	78	4.50%
Adjectives/adjective phrases	60	3.45%
None of above	123	8.36%
Total	**1,735**	**100%**

As a baseline method, we carefully implemented J&L's method. Specifically, CSRs for comparative question identification were learned from the labeled questions, and then a statistical classifier was built by using CSR rules as features. We examined both SVM and Naïve Bayes (NB) models as reported in their experiments. For the comparator extraction, LSRs were learned from SET-B and applied for comparator extraction.

To start the bootstrapping procedure, we applied the IEP "<#start nn/$c vs/cc nn/$c ?/. #end>" to all the questions in SET-R and gathered 12,194 comparator pairs as the initial seeds. For our weakly supervised method, there are four parameters, i.e., α, β, γ, and λ, need to be determined empirically. We first mined all possible candidate patterns from the suffix tree using the initial seeds. From these candidate patterns, we applied them to SET-R and got a new set of 59,410 candidate comparator pairs. Among these new candidate comparator pairs, we randomly selected 100 comparator pairs and manually classified them into reliable or nonreliable comparators. Then we found α that maximized precision without hurting recall by investigating frequencies of pairs in the labeled set. By this method, α was set to 3 in our experiments. Similarly, the threshold parameters β and γ for pattern evaluation were set to 10 and 0.8, respectively. For the interpolation parameter λ in (3), we simply set the value to 0.5 by assuming that two reliability scores are equally important.

As evaluation measures for comparative question identification and comparator extraction, we used precision, recall, and F1-measure. All results were obtained from fivefold cross validation. Note that J&L's method needs a training data but ours use the unlabeled data (SET-R) with weakly supervised method to find parameter setting. This fivefold evaluation data is not in the unlabeled data. Both methods were tested on the same test split in the fivefold cross validation. All evaluation scores are averaged across all five folds.

For question processing, we used our own statistical POS tagger developed in-house.[4]

5.1.3 Evaluation Data for Comparator Ranking

By applying learned IEPs to the source data, a database consisting of 328,364 unique comparator pairs were constructed from 679,909 automatically identified comparative questions using Max Length Strategy (which is proved to be the most efficient in our experiments). For each comparator, average three comparable comparators are extracted. Specifically, the numbers of their comparable comparators even come up to 500 for some popular comparators such as "windows," "apple," etc. Based on these comparator pairs, our comparator ranking method was examined.

We built three different sets of user's input entities from our comparator database, named FREQ-QUERY, MID-QUERY, and RARE-QUERY. The FREQ-QUERY consists of 200 entities that have more than 50 comparators in the comparator database, and the MID-QUERY consists of 200 entities that have more than 20 but less than 30 comparators. The RARE-QUERY consists of 200 entities which have less than 10 comparators. All entities in the query sets were randomly selected.

5.2 Experiment Results

5.2.1 Comparative Question Identification and Comparator Extraction

Table 4 shows our experimental results. In the table, "Identification only" indicates the performances in comparative question identification, "Extraction only" denotes the performances of comparator extraction when only comparative questions are used as input, and "All" indicates the end-to-end performances when question identification results were used in comparator extraction. Note that the results of J&L's method on our collections are very comparable to what is reported in their paper.

In terms of precision, the J&L's method is competitive to our method in comparative question identification. However, the recall is significantly lower than ours. In terms of recall, our method outperforms J&L's method by 35 and 22 percent in comparative question identification and comparator extraction respectively. In our analysis, the low recall of J&L's method is mainly caused by low coverage of learned CSR patterns over the test set.

In the end-to-end experiments, our weakly supervised method performs significantly better than J&L's method. Our method is about 55 percent better in F1-measure. This result also highlights another advantage of our method that identifies comparative questions and extracts comparators simultaneously using one single pattern. J&L's method uses two kinds of pattern rules, i.e., CSRs and LSRs. Its performance drops significantly due to error propagations. F1-measure of J&L's method in "All" is about 30 percent and 32 percent worse than the scores of "Identification only" and "Extraction" only respectively, our method only shows small amount of performance decrease (approximately 7-8 percent).

We also analyzed the effect of pattern generalization and specialization. Table 5 shows the results. Despite of the simplicity of our methods, they significantly contribute to performance improvements. This result shows the importance of learning patterns flexibly to capture various comparative question expressions. Among the 6,127 learned IEPs in our database, 5,930 patterns are generalized ones, 171 are specialized ones, and only 26 patterns are non-generalized and specialized ones.

4. We used NLC-PosTagger which is developed by NLC group of Microsoft Research Asia. It uses the modified Penn Treebank POS set for its output; for example, NNS (plural nouns), NN (nouns), NP (noun phrases), NPS (plural noun phrases), VBZ (verb, present tense, third person singular), JJ (adjective), RB(adverb), and so on.

TABLE 4
Performance Comparison between Our Method and Jindal and Bing's Method (Denoted as J&L)

	Identification only (SET-A+SET-B)			Extraction only (SET-B)		All (SET-B)		
	J&L (CSR)		**Our**	J&L	**Our**	J&L		**Our**
	SVM	NB	**Method**	(LSR)	**Method**	SVM	NB	**Method**
Recall	0.601	0.537	**0.817***	0.621	**0.760***	0.373	0.363	**0.760***
Precision	0.847	**0.851**	0.833	0.861	**0.916***	0.729	0.703	**0.776***
F-score	0.704	0.659	**0.825***	0.722	**0.833***	0.493	0.479	**0.768***

*The values with * indicate statistically significant improvements over J&L (CSR) SVM or J&L (LSR) according to t-test at $p < 0.01$ level.*

Table 6 shows the efficiency of different comparator extraction strategies. The results show Max Reliablility Strategy has the similar performance to Random Strategy. Max Length Strategy works better. That's because applicable IEPs usually have the same reliability scores. It's hard to distinguish different IEPs with reliability scores.

To investigate the robustness of our bootstrapping algorithm for different seed configurations, we compare the performances between two different seed IEPs. The results are shown in Table 7. As shown in the table, the performance of our bootstrapping algorithm is stable regardless of significantly different number of seed pairs generated by the two IEPs. This result implies that our bootstrapping algorithm is not sensitive to the choice of IEP.

Table 8 also shows the robustness of our bootstrapping algorithm. In Table 8, "All" indicates the performances that all comparator pairs from a single seed IEP is used for the bootstrapping, and "Partial" indicate the performances using only 1,000 randomly sampled pairs from "All." As shown in the table, there is no significant performance difference.

In addition, we conducted error analysis for the cases where our method fails to extract correct comparator pairs.

- 8 percent of the unidentified comparative questions are due to rare forms of comparators. For example, pattern "#start $c or $c ? #end" is not precise enough. When we constrain the pattern, only "#start NN/$c or NN/$c ? #end" are generated. Then, "Saving a dog or breaking an entering ?" cannot be identified.
- 3.3 percent of the unidentified comparative questions are due to their wrongly postag labeling. For example, "survey: ps2 versus wii?" can be easily identified by the pattern ": NN/$C CC NN/$C ?" if it's tagged into "VB : NN CC NN.". However, it's wrongly tagged into "VB: CD NN NN." which makes it looks noncomparative.
- 88.7 percent of the unidentified comparative questions are due to their rare patterns of comparable comparator contexts. These kind of unidentified comparative questions can be further categorized into three categories shown as follows:
  - Contexts of comparable comparators contain low frequency words (24.6 percent). For example, question "What's your preference air cooled or water cooled 911's?" is difficult to be identified because pattern containing "preference" is difficult to be mined due to low frequency of preference" and "What's your NN $C or $C ?" is not accurate enough.
  - Words in contexts of comparable comparators are frequent, while contexts are organized in a rare form (61.9 percent). This case is especially serious when there are description sentences in the questions. For example, for "I have the option of going to either Japan or Argentina; both are a month long in January. Where should I go?", it is difficult to derive a common pattern from the question.

TABLE 5
Effect of Pattern Specialization and Generalization in the End-to-End Experiments

	Recall	Precision	F-score
Original Patterns	0.689	0. 449	0.544
+ Specialized	0.731	0.602	0.665
+ Generalized	0.760	0.776	0.768

TABLE 6
Comparison between Different Extraction Strategies

Extraction Strategies	Recall	Precision	F-score
Random Extraction	0.744	0.891	0.81
Max Length Extraction	0.76	0.916	0.833
Max Reliability Extraction	0.747	0.891	0.813

TABLE 7
Performance Variation over Different Initial Seed IEPs in the End-to-End Experiments

Seed patterns	# of resulted seed pairs	F-score
<#start nn/$c vs/cc nn/$c ?/. #end>	12,194	0.768
<#start which/wdt is/vb better/jjr , nn/$c or/cc nn/$c ?/. #end>	1,478	0.760

TABLE 8
Performance Variation over Different Sizes of Seed Pairs Generated from a Single Initial Seed IEP
"<#start nn/$c vs/cc nn/$c ?/. #end>"

Set (# of seed pairs)	Recall	Precision	F-score
All (12,194)	0.760	0.774	0.768
Partial (1,000)	0.724	0.763	0.743

- Some words in cQA services are wrongly spelled (2 percent). Content in cQA services is generated by users and is expressed more freely. On one hand, there are many abberations such as "thx" ("thanks"), "u" ("you"), etc. These "new words" cannot be correctly identified resulting in wrongly postag labeling of the whole question. On the other hand, many key words for identification are also wrongly spelled, e.g., "different," "wich," etc. A question containing those wrongly spelled words cannot match an existing pattern containing these wrongly spelled keywords.

5.2.2 Comparator Extraction and Ranking Comparison between cQA Questions and Other Sources

The coverage of mined comparators for an arbitrary user's comparison intent is obviously one of the important factors in the evaluation. The coverage can be implied by comparing comparators extracted from cQA questions and that from other sources. In this paper, we compare comparator extracted from cQA questions and query logs.

According to our observation, "versus" is a strong indicator for comparison intents in query logs. We extract a form of "X versus Y" queries from about 13M unique queries which frequency is more than 10 in the 6 month query log of a commercial web search engine. Then we extract strings corresponding to "X" and "Y" and regard them as a comparator pair, if their length is more than 3. Then we calculate overlap statistics between comparator pairs from query log and from cQA questions. The total number of comparator pairs extracted from the query log is about 4,200.

In the results, our comparator database covers about 64 percent of comparator pairs from query log on the basis of frequencies (about 45 percent of unique pairs), while the coverage of comparator pairs from query log over our database is only 0.6 percent. For the queries which frequency in query log is more than 100, the coverage of our database is raised up to 76 percent. When considering that there can be lots of noisy comparator pairs from the "X versus Y" pattern (e.g., "1 versus 300" is a name of TV show program), these numbers can indicate that cQA questions are rich sources for comparator extraction and our comparator pair database built from cQA questions could cover many comparator pairs which can be mined from query logs.

Comparator ranking aims to rank comparable comparators given a user's input entity according to some reasonable criteria. Due to the subjective nature of comparators, to determine if a comparator is good for a user's input entity is not easy. In this work, we just calculate comparator ranking correlation between cQA questions and webpages to validate that our comparator ranking is biased on cQA questions. Frequency of comparators compared in webpages is obtained by the number of webpages returned by Google search engine using a comparator pair connected by "versus"as search query. Details are as follows.

First, for an input entity Q in each set, we rank all its comparators Es in the database with our ranking method. Then, we investigate the number of webpages returned by Google using phrasal queries, "Q versus E" or "E versus Q" by combining Q and E with quotation marks. If there are more interests in comparing E with Q in webpages, there would be many Webpages containing the phrase "Q versus E" or "E versus Q."

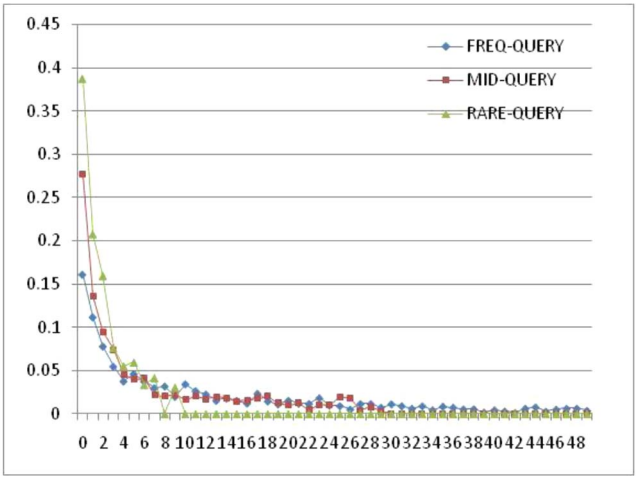

Fig. 4. Changes of average number of retrieved webpages according to comparator ranks. X-axis indicates the rank position of E given a query Q in a query set. Y-axis indicates the number of webpages retrieved by phrasal queries containing E and Q. The values are normalized by the total number of webpages retrieved by phrasal queries containing Q.

Fig. 4 shows the relation between a rank position of E in our ranking results from cQA questions and the number of web pages returned by phrasal queries containing E. As shown in the figure, the average number of webpages generally decreases as the rank of E get lower. Specifically, the number of webpages in top rank areas is much higher than other areas in every query set. Fig. 3 proves that the comparison interests in cQA questions are highly correlated with that in webpages.

5.2.3 Examples of Comparator Extraction and Ranking

Table 9 lists top 10 frequently compared entities for a target item, such as "Channel," "Gap," in our question archive. As shown in the table, our comparator mining method successfully discovers realistic comparators. For example, for "Chanel," most results are high-end fashion brands such as "Dior" or "Louis Vuitton," while the ranking results for "Gap" usually contains similar apparel brands for young people, such as "Old Navy" or "Banana Republic." For the basketball player "Kobe," most of the top ranked comparators are also famous basketball players. Some interesting comparators are shown for "Canon" (the company name). It is famous for different kinds of its products, for example, digital cameras and printers, so it can be compared to different kinds of companies. For example, it is compared to "HP," "Lexmark," or "Xerox," the printer manufacturers, and also compared to "Nikon," "Sony," or "Kodak," the digital camera manufactures. Besides general entities such as a brand or company name, our method also found an interesting comparable entity for a specific item in the experiments. For example, our method recommends "Nikon d40i," "Canon rebel xti," "Canon rebel xt," "Nikon d3000," "Pentax k100d," "Canon eos 1000d" as comparators for the specific camera product "Nikon 40d."

Table 10 can show the difference between our comparator mining and query/item recommendation. As shown in

TABLE 9
Examples of Comparators for Different Entities

	Chanel	Gap	Nikon d40	Kobe	Canon
1	Dior	Old Navy	Nikon d40i	Lebron	Nikon
2	Louis Vuitton	American Eagle	Canon rebel xti	Jordan	Sony
3	Coach	Banana Republic	Canon rebel xt	MJ	Kodak
4	Gucci	Guess by Marciano	Nikon d3000	Shaq	Panasonic
5	Prada	ACP Ammunition	Pentax k100d	Wade	Casio
6	Lancome	Old Navy brand	Canon eos 1000d	T-mac	Olympus
7	Versace	Hollister	-	Lebron James	Hp
8	LV	Aeropostal	-	Nash	Lexmark
9	Mac	American Eagle outfitters	-	KG	Pentax
10	Dooney	Guess	-	Bonds	Xerox

TABLE 10
Related Queries Returned by Google Related Searches for the Same Target Entities in Table 9

Chanel	Gap	Nikon d40	Kobe	Canon
Chanel handbag	Gap coupons	Nikon d60	Kobe Bryant stats	Canon t2i
Chanel sunglass	Gap outlet	Nikon d40x	Lakers Kobe	Canon printers
Chanel earrings	Gap card	Nikon d40 review	Kobe espn	Canon printer drivers
Chanel watches	Gap careers	Ritz camera	Kobe Dallas Mavericks	Canon downloads
Chanel shoes	Gap casting call	Nikon d80	Kobe NBA	Canon copiers
Chanel jewelry	Gap adventures	Nikon d50	Kobe 2009	Canon scanner
Chanel clothing	*Old navy*	Nikon d40 kit	Kobe san Antonio	Canon lenses
Dior	*Banana republic*	b&h photo	Kobe Bryant 24	*Nikon*

the table, "Google related searches" generally suggests a mixed set of two kinds of related queries for a target entity: 1) queries specified with subtopics for an original query (e.g., "Chanel handbag" for "Chanel") and 2) its comparable entities (e.g., "Dior" for "Chanel"). It confirms one of our claims that comparator mining and query/item recommendation are related but not the same.

Table 11 compares ranking results of comparability-based and graph-based ranking methods. For some queries whose comparators' frequency differs significantly, such as "Obama," "iphone," and "xbox 360," the ranking results of two methods do not make many differences. That's because frequency plays the main role in the ranking process for these queries in graph-base ranking methods. However, for queries whose comparators share similar frequency, such as "BMW 328i," "Nokia N75," and "Nikon D200," the differences between two methods are obvious. For example, "Cadillac Cts" has more comparison relations with other comparable comparators of "BMW 328i" and is ranked much higher by graph-based algorithm though it is not compared with "BMW 328i" more frequent than "Toyota Avalon."

Table 12 shows parameter impacts on graph-based ranking method for "Obama" for which little difference exists between two proposed ranking methods. When λ equals 0, only representative is considered. When λ equals 1, only comparability is considered. We can see that only the orders of "Hilary" and "Palin" changed because "Hilary" is more often compared with other comparators. The reason for the consistence of two ranking results is that a comparator is usually a good baseline which is representative when it's much more frequently compared with the query entity.

Although the results from our ranking methods reflect the people's interests on comparison reasonably in the examples, there are also several problems.

- The redundancies in suggestion results, such as "Hillary" and "Hillary," must be removed by using an automatic method. Most of them are caused by alias of entities or misspelling words. Also, filtering noisy comparators such as "camera" can be also an important issue.
- Sometimes, comparison relationships for one entity can be multidimensional. For example, the ranking result for the query "London" might be a mixed set of its comparable cities in England, such as "Manchester," and other popular travel locations, such as "Paris." In this case, it would be necessary to distinguish comparators in the context of comparison purpose.

6 Conclusion

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

TABLE 11
Comparison between Ranking Results of Comparability-Based and Graph-Based Ranking Methods

	Obama		iphone		Xbox 360	
	Comparability	PageRank	Comparability	PageRank	Comparability	PageRank
1	Mccain	Mccain	ipod touch	ipod touch	Playstation 3	Playstation 3
2	Clinton	Clinton	Blackberry	Blackberry	Wii	Wii
3	Hillary	Hillary	itouch	itouch	Ps3	Ps3
4	Palin	Hilary	Blackberry storm	Blackberry storm	Nintendo Wii	Nintendo Wii
5	Hilary	Palin	Voyager	Voyager	ipod touch	Psp
6	Bush	Bush	Sidekick lx	Sidekick lx	Psp	Xbox
7	Biden	Biden	Sidekick	Sidekick	Xbox	ipod touch
8	Hillary Clinton	Hillary Clinton	Blackberry Curve	Blackberry Curve	Xbox 360 elite	Xbox 360 elite
9	Osama	Osama	ipod	ipod	pc	pc
10	John Mccain	John Mccain	Instinct	Blackberry Pearl	Playstation 2	Playstation 2
	BMW 328i		Nokia N75		Nikon D200	
	Comparability	PageRank	Comparability	PageRank	Comparability	PageRank
1	Toyota Avalon	**Cadillac Cts**	LG Shine	LG Shine	Nikon D80	**Canon 30D**
2	BMW 335i	Toyota Avalon	Samsung Sync	Blackberry Pearl	Canon 40D	Nikon D80
3	Mercedes c300 sport	Integra Gsr	Black Berry Curve	Samsung Sync	**Canon 30D**	Canon 40D
4	Audi A3	Acura TL	Motorola k1 Krzr	**Sony Ericsson w580i walkman**	Canon EOS 40D	Canon EOS 40D
5	Honda Accord 08	Honda Accord 08	Samsung D807	Nokia 6555	Canon EOS 30D	Nikon D40x
6	Acura TL	Audi A3	ipod nano	Samsung D807	Canon EOS 5D	Canon EOS 400D
7	Integra Gsr	Lexus 350	Nokia 6555	ipod nano	Nikon D40x	camera
8	**Cadillac Cts**	BMW 335i	Motorola Razr2	Motorola Razr2	Canon EOS Rebel xsi	Canon EOS 30D
9	Mercedes c300	Mercedes c300	Motorola A1200 Ming	Motorola A1200 Ming	Sigma SD14	Canon EOS 5D
10	Lexus 350	Mercedes c300 sport	**Sony Ericsson w580i walkman**	Motorola k1 Krzr	Canon EOS 400D	Canon EOS Rebel xsi

The experimental results show that our method is effective in both comparative question identification and comparator extraction. It significantly improves recall in both tasks while maintains high precision. Our examples show that these comparator pairs reflect what users are really interested in comparing.

Our comparator mining results can be used for a commerce search or product recommendation system. For example, automatic suggestion of comparable entities can assist users in their comparison activities before making their purchase decisions. Also, our results can provide useful information to companies which want to identify their competitors.

In the future, we would like to improve extraction pattern application and mine rare extraction patterns. How to identify comparator aliases such as "LV" and "Louis Vuitton" and how to separate ambiguous entities such "Paris versus London" as location and "Paris versus Nicole" as celebrity are all interesting research topics. We also plan to develop methods to summarize answers pooled by a given comparator pair.

ACKNOWLEDGMENTS

This work was done when the first author worked as an intern at Microsoft Research Asia. It was supported by the National Natural Science Foundation of China under Grant Nos. 61170189, 61202239, and 60973105, the Research Fund for the Doctoral Program of Higher Education under Grant No. 20111102130003, and the State Key Laboratory of Software Development Environment under Grant No.SKLSDE-2011ZX-03. This paper was

TABLE 12
Parameter Tuning on Graph-Based Comparator Ranking Method for "Obama"

	0	0.2	0.4	0.6	0.8	1
1	Mccain	Mccain	Mccain	Mccain	Mccain	Mccain
2	Clinton	Clinton	Clinton	Clinton	Clinton	Clinton
3	Hillary	Hillary	Hillary	Hillary	Hillary	Hillary
4	Hilary	Hilary	Hilary	Hilary	Hilary	Palin
5	Palin	Palin	Palin	Palin	Palin	Hilary
6	Bush	Bush	Bush	Bush	Bush	Bush
7	Biden	Biden	Biden	Biden	Biden	Biden
8	Hillary Clinton	Hillary Clinton	Hillary Clinton	Hillary Clinton	Hillary Clinton	Hillary Clinton
9	Osama	Osama	Osama	Osama	Osama	Osama
10	John Mccain	John Mccain	John Mccain	John Mccain	John Mccain	John Mccain

presented in part at the 48th Annual Meeting of the Association for Computational Linguistics (ACL'10), Uppsala, Sweden, July 11-16, 2010 [9].

REFERENCES

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

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

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

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

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

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

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

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

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

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

[11] R.J. Mooney and R. Bunescu, "Mining Knowledge from Text Using Information Extraction," *ACM SIGKDD Exploration Newsletter*, vol. 7, no. 1, pp. 3-10, 2005.

[12] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PagRank Citation Ranking: Bringing Order to the Web," *Stanford Digital Libraries Working Paper*, 1998.

[13] D. Radev, W. Fan, H. Qi, H. Wu, and A. Grewal, "Probabilistic Question Answering on the Web," *J. Am. Soc. for Information Science and Technology*, pp. 408-419, 2002.

[14] D. Ravichandran and E. Hovy, "Learning Surface Text Patterns for a Question Answering System," *Proc. 40th Ann. Meeting on Assoc. for Computational Linguistics (ACL '02)*, pp. 41-47, 2002.

[15] E. Riloff and R. Jones, "Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping," *Proc. 16th Nat'l Conf. Artificial Intelligence and the 11th Innovative Applications of Artificial Intelligence Conf. (AAAI '99/IAAI '99)*, pp. 474-479, 1999.

[16] E. Riloff, "Automatically Generating Extraction Patterns from Untagged Text," *Proc. 13th Nat'l Conf. Artificial Intelligence*, pp. 1044-1049, 1996.

[17] S. Soderland, "Learning Information Extraction Rules for Semi-Structured and Free Text," *Machine Learning*, vol. 34, nos. 1-3, pp. 233-272, 1999.

Shasha Li received the BS degree in the School of Computer Science from National University of Defense Technology in 2005. Currently, she is working toward the PHD degree at the School of Computer Science, National University of Defense Technology. Her research interests include information extraction, text summarization, and text mining.

Young-In Song received the BS, MS, and PhD degrees in computer science from Korea University, Seoul, Korea, in 2001, 2003, and 2008, respectively. Currently, he is working as an associate researcher of WIT Group in Microsoft Research Asia (MSRA). His research interests include automatic summarization, QA applications, intelligent IR systems based on language understanding, and so on.

Chin-Yew Lin received the BS degree in electrical and control engineering from National Chiao Tung University in 1987, and the MS and PhD degrees in computer engineering from the University of Southern California, in 1991 and 1997, respectively. Currently, he is the group manager of the Web Intelligence (WIT) group at Microsoft Research Asia. His research interests include automated summarization, opinion analysis, question answering (QA), social computing, community intelligence, machine translation (MT), and machine learning. He is a member of the IEEE.

Zhoujun Li received the BS degree in the School of Computer Science from Wuhan University in 1984, and the MS and PhD degrees in the School of Computer Science from National University of Defense Technology. Currently, he is working as a professor of Beihang University. His research interests include data mining, information retrieval and Bioinformatics. He is a member of the IEEE.