Comparable Entity Mining from Comparative Questions

Shasha Li[1] , Chin-Yew Lin[2] , Young-In Song[2] , Zhoujun Li[3]
[1]National University of Defense Technology, Changsha, China
[2]Microsoft Research Asia, Beijing, China
[3]Beihang University, Beijing, China
shashali@nudt.edu.cn[1], {cyl,yosong}@microsoft.com[2],
lizj@buaa.edu.cn[3]

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. To address this difficulty, we present a novel way to automatically mine comparable entities from comparative questions that users posted online. To ensure high precision and high recall, we develop a weakly-supervised bootstrapping method for comparative question identification and comparable entity extraction by leveraging a large online question archive. The experimental results show our method achieves F1-measure of 82.5% in comparative question identification and 83.3% in comparable entity extraction. Both significantly outperform an existing state-of-the-art method.

1 Introduction

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

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

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

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 650–658,
Uppsala, Sweden, 11-16 July 2010.

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 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 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% F1-measure in comparative question identification, 83.3% in comparator extraction, and 76.8% in end-to-end comparative question identification and comparator extraction which outperform the most relevant state-of-the-art method by Jindal & Liu (2006b) 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 reports the evaluations of our techniques, and we conclude the paper and discuss future work in Section 5.

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 (Linden et al., 2003). 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 (Cardie, 1997; Califf and Mooney, 1999; Soderland, 1999; Radev et al., 2002; Carreras et al., 2003). Specifically, the most relevant work is by Jindal and Liu (2006a and 2006b) on mining comparative sentences and relations. Their methods applied class sequential rules (CSR) (Chapter 2, Liu 2006) and label sequential rules (LSR) (Chapter 2, Liu 2006) 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 (Jindal and Liu, 2006b) (J&L). 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 (Riloff, 1996; Riloff and Jones, 1999; Ravichandran and Hovy, 2002; Mooney and Bunescu, 2005; Kozareva et al., 2008). 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 & Liu 2006

In this subsection, we provide a brief summary of the comparative mining method proposed by Jindal and Liu (2006a and 2006b), 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 J&L's original papers for more details.

CSR and LSR

CSR is a classification rule. It maps a sequence pattern $S(s_1 s_2 \ldots s_n)$ to a class *C*. In our problem, *C* is either *comparative* or *non-comparative*. 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.

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 *non-comparative*, 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 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. non-entity-feature. These labels were used as pivots together with special tokens l_i & r_j[1] (token position), #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% 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 & Liu (2006b) 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 se-

[1] l_i marks a token is at the i^{th} position to the left of the pivot and r_j marks a token is at j^{th} position to the right of the pivot where i and j are between 1 and 4 in J&L (2006b).

quential patterns which can be used to identify

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 ?>
...

Table 1: Candidate indicative extraction pattern (IEP) examples of the question "*which city is better, NYC or Paris?*"

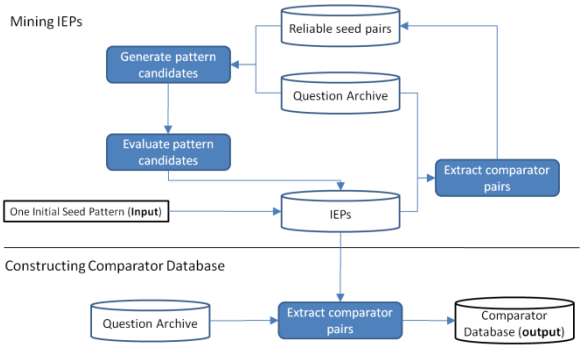

Figure 1: Overview of the bootstrapping alogorithm

comparative question and extract comparators simultaneously.

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 subsections. The evaluations shown in section 4 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 (2002). 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 a 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 Figure 1. The bootstrapping process starts with a single IEP. From it, we extract a set of 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.

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

Pattern Generation

To generate sequential patterns, we adapt the surface text pattern mining method introduced in (Ravichandran and Hovy, 2002). 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 a sen-

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

tence in the question. 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 (Gusfield, 1997) 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 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 $C>*"* is too general, and there can be many non-comparative questions matching the pattern, for instance, *"true or false?"*. For this reason, we perform pattern specialization by adding POS tags to all comparator slots. For example, from the lexical pattern "<$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.

Pattern Evaluation

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

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

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

However, Equation (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 Equation (1) might be underestimated which could affect the effectiveness of equation (1) on distinguishing IEPs from non-reliable 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 Equation (1) and (3), the final reliability score $R(p_i)^k_{final}$ for a pattern is defined as follows:

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

Using Equation (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.

4 Experiments

4.1 Experiment Setup

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.

Evaluation Data

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*, *non-comparative*, or *unknown*. Among them, 139 (2.67%) questions were classified as comparative, 4,934 (94.88%) as non-comparative, and 127 (2.44%) 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% 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 non-comparative 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.

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 non-reliable 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 Equation (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 5-fold 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 5-fold evaluation data is not in the unlabeled data. Both methods were tested on the same test split in the 5-fold cross validation. All evaluation scores are averaged across all 5 folds.

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

4.2 Experiment Results

Comparative Question Identification and Comparator Extraction

Table 2 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 ques-

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

[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, 3rd person singular), JJ (adjective), RB(adverb), and so on.

	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***

Table 2: Performance comparison between our method and Jindal and Bing's Method (denoted as J&L). 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.

tion identification. However, the recall is significantly lower than ours. In terms of recall, our method outperforms J&L's method by 35% and 22% 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% 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% and 32% worse than the scores of "Identification only" and "Extraction" only respectively, our method only shows small amount of performance decrease (approximately 7-8%).

We also analyzed the effect of pattern generalization and specialization. Table 3 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.

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 4. 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 5 also shows the robustness of our bootstrapping algorithm. In Table 5, '*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:

- 23.75% of errors on comparator extraction are due to wrong pattern selection by our simple maximum IEP length strategy.
- The remaining 67.63% of errors come from comparative questions which cannot be covered by the learned IEPs.

	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 3: Effect of pattern specialization and Generalization 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 4: Performance variation over different initial seed IEPs in the end-to-end experiments

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

Table 5: Performance variation over different sizes of seed pairs generated from a single initial seed IEP "*<#start nn/$c vs/cc nn/$c ?/. #end>*".

	Chanel	Gap	iPod	Kobe	Canon
1	Dior	Old Navy	Zune	Lebron	Nikon
2	Louis Vuitton	American Eagle	mp3 player	Jordan	Sony
3	Coach	Banana Republic	PSP	MJ	Kodak
4	Gucci	Guess by Marciano	cell phone	Shaq	Panasonic
5	Prada	ACP Ammunition	iPhone	Wade	Casio
6	Lancome	Old Navy brand	Creative Zen	T-mac	Olympus
7	Versace	Hollister	Zen	Lebron James	Hp
8	LV	Aeropostal	iPod nano	Nash	Lexmark
9	Mac	American Eagle outfitters	iPod touch	KG	Pentax
10	Dooney	Guess	iRiver	Bonds	Xerox

Table 6: Examples of comparators for different entities

Chanel	Gap	iPod	Kobe	Canon
Chanel handbag	Gap coupons	***iPod nano***	Kobe Bryant stats	Canon t2i
Chanel sunglass	Gap outlet	***iPod touch***	Lakers Kobe	Canon printers
Chanel earrings	Gap card	iPod best buy	Kobe espn	Canon printer drivers
Chanel watches	Gap careers	iTunes	Kobe Dallas Mavericks	Canon downloads
Chanel shoes	Gap casting call	Apple	Kobe NBA	Canon copiers
Chanel jewelry	Gap adventures	iPod shuffle	Kobe 2009	Canon scanner
Chanel clothing	***Old navy***	iPod support	Kobe san Antonio	Canon lenses
Dior	***Banana republic***	iPod classic	Kobe Bryant 24	***Nikon***

Table 7: Related queries returned by Google related searches for the same target entities in Table 6. The bold ones indicate overlapped queries to the comparators in Table 6.

Examples of Comparator Extraction

By applying our bootstrapping method to the entire source data (60M questions), 328,364 unique comparator pairs were extracted from 679,909 automatically identified comparative questions.

Table 6 lists top 10 frequently compared entities for a target item, such as *Chanel*, *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 7 can show the difference between our comparator mining and query/item recommendation. As shown in 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.

5 Conclusion

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

The experimental results show that our method is effective in both comparative question identification and comparator extraction. It sig-

nificantly 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 vs. London*" as location and "*Paris vs. Nicole*" as celebrity are all interesting research topics. We also plan to develop methods to summarize answers pooled by a given comparator pair.

6 Acknowledgement

This work was done when the first author worked as an intern at Microsoft Research Asia.

References

Mary Elaine Califf and Raymond J. Mooney. 1999. Relational learning of pattern-match rules for information extraction. In *Proceedings of AAAI'99 /IAAI'99*.

Claire Cardie. 1997. Empirical methods in information extraction. *AI magazine*, 18:65–79.

Dan Gusfield. 1997. *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge University Press, New York, NY, USA

Taher H. Haveliwala. 2002. Topic-sensitive pagerank. In *Proceedings of WWW '02*, pages 517–526.

Glen Jeh and Jennifer Widom. 2003. Scaling personalized web search. In *Proceedings of WWW '03*, pages 271–279.

Nitin Jindal and Bing Liu. 2006a. Identifying comparative sentences in text documents. In *Proceedings of SIGIR '06*, pages 244–251.

Nitin Jindal and Bing Liu. 2006b. Mining comparative sentences and relations. In *Proceedings of AAAI '06*.

Zornitsa Kozareva, Ellen Riloff, and Eduard Hovy. 2008. Semantic class learning from the web with hyponym pattern linkage graphs. In *Proceedings of ACL-08: HLT*, pages 1048–1056.

Greg Linden, Brent Smith and Jeremy York. 2003. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing*, pages 76-80.

Raymond J. Mooney and Razvan Bunescu. 2005. Mining knowledge from text using information extraction. *ACM SIGKDD Exploration Newsletter*, 7(1):3–10.

Dragomir Radev, Weiguo Fan, Hong Qi, and Harris Wu and Amardeep Grewal. 2002. Probabilistic question answering on the web. *Journal of the American Society for Information Science and Technology*, pages 408–419.

Deepak Ravichandran and Eduard Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of ACL '02*, pages 41–47.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of AAAI '99 /IAAI '99*, pages 474–479.

Ellen Riloff. 1996. Automatically generating extraction patterns from untagged text. In *Proceedings of the 13th National Conference on Artificial Intelligence*, pages 1044–1049.

Stephen Soderland. 1999. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3):233–272.