INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN 2278 – 0211 (Online)

Decision Making System for Comparative Question Using Entity Mining

J. V. Ramteke
Lecturer, University of Pune, S.V.I.T., Chincholi Nasik, Maharashtra, India
J. S. Jondhale
Lecturer, Pravara Education Society Loni, Maharashtra, India

Abstract:
Comparing one thing with another is a typical part of human decision making process; especially during an online purchase scheme making Comparisons between things is a typical part of human decision making process. But however, it is difficult to know what are to be compared and what can be the alternatives. For example, if someone is interested in certain products such as digital cameras, then he /she would want to know what the alternatives are and compare different cameras before making any purchase. To get rid of this difficulty, my paper presents an ideal way for automatically mine comparable entities from comparative questions that users posted online. It gives an opportunity to improve the search experience by automatically offering comparisons to user. To ensure high precision and high recall, we are developing a weakly-supervised bootstrapping method for comparative question identification and comparable entity extraction by leveraging a large online question archive. The results will be very useful in helping users' exploration of alternative choices by suggesting comparable entities based on other users' prior requests.

Keywords: *Weakly- supervised bootstrapping method, high recall, precision, pattern mining*

1. Introduction

Decision making comprises comparing alternative options .For example, if someone is interested in buying certain products such as cars, phones etc he or she would want to know what the alternatives are and compare different items with each other before making a purchase. This type of comparison activity is very common in our daily life but requires high knowledge skill for effective comparison. Many sites as well as media strive in providing editorial comparison content and surveys to satisfy the need of user. 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, rating products on basis of comparison and identify pros and cons.

In this paper focus is on one variant of Mining of comparable entity from comparative questions. Finding set of comparable entities given a user's input entity. It is difficult to decide if two entities are comparable or not since people do compare 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, Flipcart recommends products to its customers based on their own purchase histories, similar customers" purchase histories, and similarity between products. However, recommending an item is totally different from finding comparable element. In the case of Flipcart, the purpose of recommendation is to entice their customers to add more items to their shopping carts by suggesting similar or related items. Which have more features than the comparable items. While in the case of comparison, we want users to explore alternatives, i.e. helping them make a decision among comparable items.

Our work on pattern generation 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).They developed WHISK to overcome this knowledge-engineering gridlock by learning text extraction rules automatically. WHISK is designed to handle text styles that are highly structured, free text, text that is neither rigidly formatted nor composed of grammatical sentences. Such semi-structured text has largely been beyond the scope of previous systems. When used in conjunction with a syntactic analyzer and semantic tagging, WHISK can also efficiently do extraction from free text. The most pertinent work is done by Jindal and Liu on mining comparative sentences and relations. Their methods applied class sequential rules (CSR) and label sequential rules (LSR) 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. However, ensuring high recall is crucial in intended application scenario where users can issue arbitrary queries. To overcome this problem, developing a weakly-supervised bootstrapping pattern learning method by effectively leveraging unlabeled questions and large amount of dataset. 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). My work is similar to the methodology using bootstrapping technique to extract entities with a specific relation. However, my working is different from theirs in that it requires not only 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. Details of Dissertation Work

Proposing a novel pattern generation & pattern evaluation method to identify comparative questions and extract comparator pairs simultaneously. By 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.

2.1. Implementation Detail

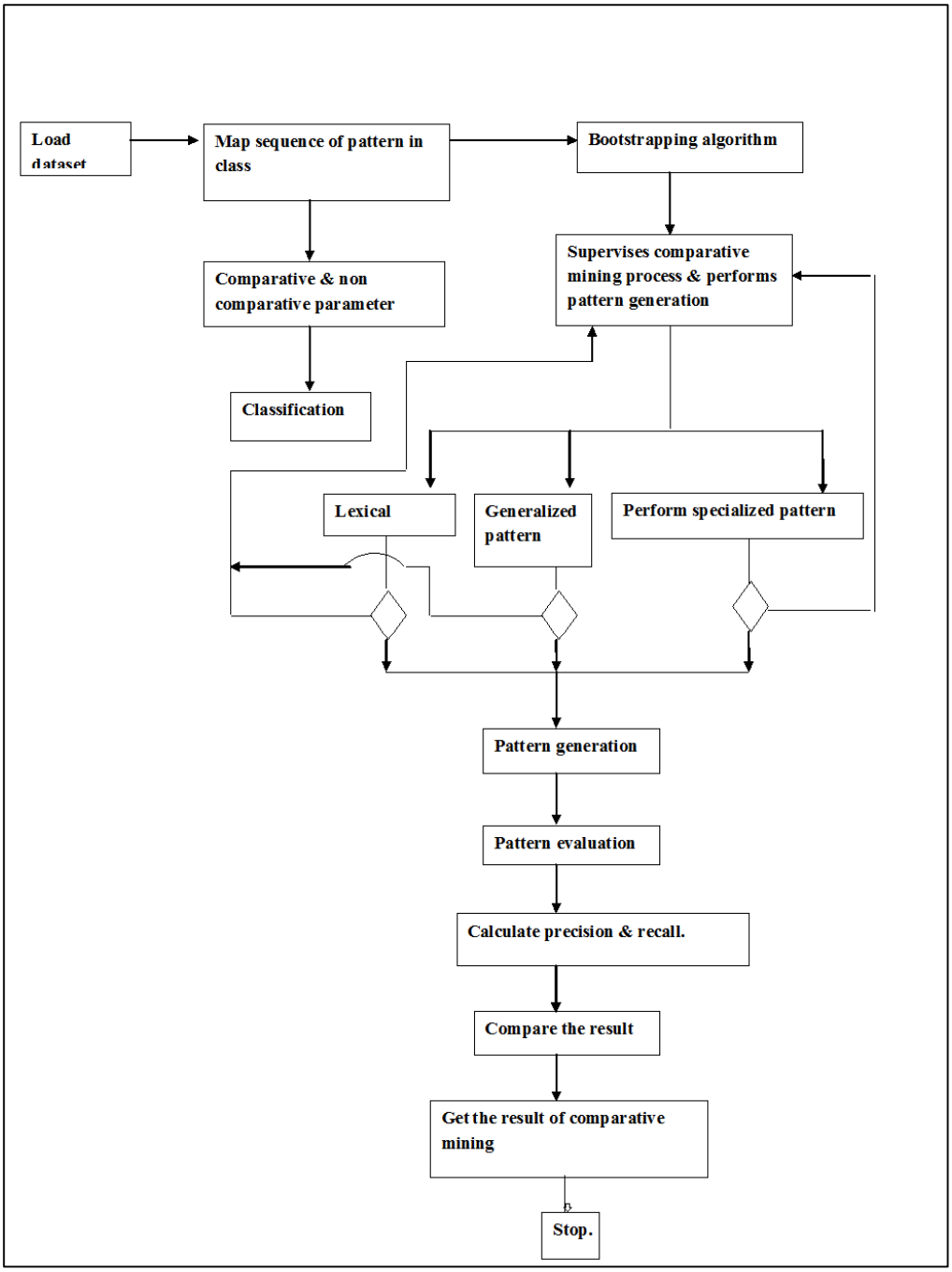
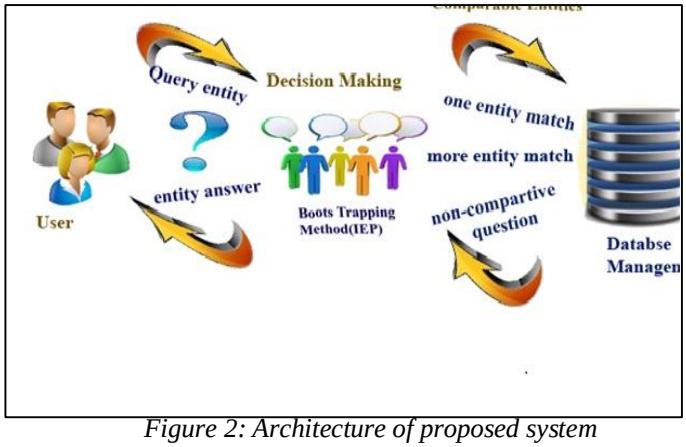


Figure 1: Implementation Flow architecture of proposed system



\

Figure 2: Architecture of proposed system

3. Main Modules

Pattern Generation (comparable Entity): Generates lexical, generalized and specialized patterns from identified comparative questions.

- Lexical patterns
- Generalized patterns
- Specialized patterns

3.1. Lexical Patterns

Lexical patterns indicate sequential patterns consisting of only words and symbols ($C, #start, and #end). They are generated by suffix tree algorithm with two constraints: A pattern should contain more than one $C, and its frequency in collection should be more than an empirically determined number.

3.2. Generalized Patterns

A lexical pattern can be too specific. Thus, we generalize lexical patterns by replacing one or more words with their POS tags. $2n - 1$ generalized patterns can be produced from a lexical pattern containing N words excluding *$Cs.*

3.3. Specialized Patterns

In some cases, a pattern can be too general. For example, a question *"samsung or lumia?"* is comparative, the pattern "<$C or $C>" is too general, and there can be many non-comparative questions matching the pattern, for instance, *"true or false?"*. For this reason, performing pattern specialization by adding POS tags to all comparator slots. For example ,from the lexical pattern "<$C or $C>"and the question "*samsung or lumia?*", "<$C/NNor $C/NN?>" will be produced as a specialized pattern .

3.4. Pattern Evaluation (Comparable Questions)

Evaluates patterns to generate a reliability score. If a patterns reliability score is greater than a threshold value, it is considered to be reliable and the pattern is kept for the next iteration. For This work, dataset from the Yahoo Research AllianceWebscope program consisting of more than 6 million questions gathered from Yahoo Answers. As the complexity of this application is very high, it's only realistic to run with a set of about 10,000 to 100,000 questions. It has complete 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 might be underestimated which could affect the effectiveness of on distinguishing IEPs from non-reliable patterns. By mitigate this problem by a look ahead procedure. Now denote the set of candidate patterns at the iteration k by. Defining the support S for comparator pair c which can be extracted by P k and does not exist in the current reliable set.

3.4.1. The Part-of-Speech Tagger

The implementation in the paper uses NLC-PosTagger which is developed by NLC group of Microsoft Research Asia.

3.4.2. Phrase Chunking

The project having in a short table a method they call phrase chunking where they change the POS tag of a word or merge words and mark them as one through a POS tag (example More+ADJ -> JJR). The paper lists six rules that the phrase chunker adheres to, however this list is non-exhaustive as the list ends with ellipsis.

3.4.3. Suffix Tree

Allows for fast storage and fast(er) retrieval by creating a tree-based index out of a set of strings. Unlike common suffix trees, which are generally used to build an index out of one (very) long string, a *Generalized Suffix Tree* can be used to build an index over many strings.

Its main operations are put and search:

- Put adds the given key to the index, allowing for later retrieval of the given value.
- Search can be used to retrieve the set of all the values that were put in the index with keys that contain a given input.

In particular, after put (K, V), search (H) will return a set containing V for any string H that is substring of K. The overall complexity of the retrieval operation (search) is *O (m)* where *m* is the length of the string to search within the index.

3.4.4. Differences from the Original Suffix Tree

Although the implementation is based on the original design by Ukkonen, there are a few aspects where it differs significantly [12].
The tree is composed of a set of nodes and labeled edges. The labels on the edges can have any length as long as it's greater than 0. The only constraint is that no two edges going out from the same node start with the same character.
Because of this, a given *(startNode, stringSuffix)* pair can denote a unique path within the tree, and it is the path (if any) that can be composed by sequentially traversing all the edges *(e1, e2 …)* starting from *startNode* such that *(e1.label + e2.label + …)* is equal to the *stringSuffix*. See the GeneralizedSuffixTree#search method for details.The union of all the edge labels from the root to a given leaf node denotes the set of the strings explicitly contained within the GST. In addition to those Strings, there are a set of different strings that are implicitly contained within the GST, and it is composed of the strings built by concatenating *e1.label + e2.label + ... + $end*, where *e1, e2* … is a proper path and *$end* is prefix of any of the labels of the edges starting from the last node of the path.

4. Mathematical Module

D: Dataset of questions.
C: Classification.
SCM: Supervised Comparative Mining process.
PG: Pattern Generator.
PE: Pattern Evaluation.
R: Result.
Let S: {D, C, SCM, PG, PE, R}.
Let D: {d1, d2, d3…….dn} where D consist of number of questions.
Let C: {c1, c2, c3…….cn} where C consist of number of classification values.
Let SCM :{ scm1, scm2, sm3…….scmn} where SCM consist of comparative results.
Number equations consecutively. Equation numbers, within Let PG: {pg1, pg2, pg3……pgn} where PG consist of pattern generation.
Let PE: {pe1, pe2, pe3……pen} where PE consist of pattern evaluation values.
Let R: {r1, r2, r3……..rn} where R consist of comparative results.
Function F1 returns the classification of dataset inputs
F1 (D) → C for input dataset
For example F1 (D) → {c1, c2, c3……cn} € C
Function F2 returns the comparative results
F2 (C) → SCM
For example F2 (C) → {scm1, scm2, scm3……scmn} € SCM.
Function F3 returns the generated patterns
F3 (SCM) → PG
For example F3 (SCM) → {pg1, pg2, pg3……pgn} € PG.
Function F4 returns the evaluate pattern results
F4 (PG) → PE
For example F4 (PG) →{ pe1, pe2 ,pe3……pen} € PE.
Function F5 returns the comparative results
F5 (PE) → R
For example F5 (PE) →{ r1, r2 ,r3……rn} € R.

	F1	F2	F3	F4	F5
F1	1	0	0	0	0
F2	1	1	0	0	0
F3	0	1	1	0	0
F4	0	0	1	1	1
F5	0	0	0	1	1

Table 1: Functional Dependency.

5. Algorithm

Construction of Suffix Tree

r ←top;
while g(r, ti) is undefined do
create new state r' and new transition g(r, ti) = r';

if r ≠ top then create new suffix link f(oldr') = r';
oldr' ← r';
r ← f(r);
create new suffix link f(oldr') = g(r, ti);
top ← g(top, ti).

Starting from STrie(€), which consists only of root and ⊥ and the links between them, and repeating Algorithm 1 for ti = t1, t2, . . . , tn, we obviously get STrie(T). The algorithm is optimal in the sense that it takes time proportional to the size of its end result STrie(T). This in turn is proportional to |Q|, that is, to the number of different substrings of T. Unfortunately,this can be quadratic in |T|, as is the case for example if $T = a^n b^n$

6. Results

To evaluate the performance of implementation, manually labeled a random 100 questions from each Yahoo Answers category except the Yahoo Products category. This gave me a total of 2500 questions of which 1037 questions where comparative. For the comparative questions I also tagged the comparative pairs for each sentence. The dataset used for testing contained 10,000 questions including the 2500 tagged questions. I measured the precision, recall and f-score of comparative question identification .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.

	Identification only (SET-A +SETB)			**Extraction only (SET-B)**			**ALL(SET-B)**	
	J&L(CSR)		Our	J&L	Our	J&L		Our
	SVM	NB	Method	(L&R)	Method	SVM	NB	Method
Recall	0.601	0.537	0.820*	0.621	0.762*	0.373	0.365	0.762*
Precision	0.847	0.851	0.835	0.861	0.917*	0.729	0.705	0.778*
F-score	0.704	0.659	0.830*	0.722	0.835*	0.479	0.480	0.770*

Table 2: Performance comparison between our method and J&L method

7. Conclusion

In the future, proposed system can extraction pattern application and mine rare extraction patterns. For example How to identify comparator aliases such as"*MCOERC"* and "*Matoshri college of engineering and research center*" and how to separate ambiguous entities such "*Bombay to Goa*" as location and "*Bombay to Goa*" as movie title are all interesting research topics. Additionally experiments with different datasets from same domains for evaluating our work with data from specific application domains being implemented. Investigating other techniques to improved result. This paper suggests efficient implementation of Pattern Mining.

8. References

1. Mary Elaine Califf and Raymond J. Mooney. 1999. Relational learning of pattern-match rules for information extration. In Proceedings of AAAI'99/IAAI'99.
2. Cardie. 1997. Empirical methods in information extraction. AI magazine, 18:65–79.
3. Dan Gusfield. 1997. Algorithms on strings, trees, and sequences: computer science and computational biology. Cambridge University Press, New York, NY, USA
4. H. Haveliwala. 2002. Topic-sensitive pagerank. In Proceedings of WWW '02, pages 517–526.
5. Glen Jeh and Jennifer Widom. 2003. Scaling personalized web search. In Proceedings of WWW '03, pages 271–279.
6. Nitin Jindal and Bing Liu. 2006a. Identifying comparative sentences in text documents. In Proceedings of SIGIR '06, pages 244–251.
7. Nitin Jindal and Bing Liu. 2006b. Mining comparative sentences and relations. In Proceedings of AAAI '06.
8. 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.
9. Greg Linden, Brent Smith and Jeremy York. 2003. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. IEEE Internet Computing, pages 76-80.
10. Raymond J. Mooney and Razvan Bunescu. 2005. Mining knowledge from text using information extraction. ACM IGKDD Exploration Newsletter, 7(1):3–10.
11. 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.
12. Deepak Ravichandran and Eduard Hovy. 2002. Learning surface text patterns for a question answering system. In Proceedings of ACL '02, pages 41–47.
13. 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.
14. Ellen Riloff. 1996. Automatically generating extraction patterns from untagged text. In Proceedings of the 13th N tional Conference on Artificial Intelligence, pages 1044–1049.
15. Stephen Soderland. 1999. Learning information extration rules for semi-structured and free text. Machine Learning, 34(1-3):233–272.