Entity Mining Extraction Using Sequential Rules

Thota Srilatha, Gorantala.Sindhuri, HimaBindu Reesu

AssistantProfessor(M.TechCSE), Balaji Institute of Engineering and sciences. Andhrapradesh

Abstract: A comparative sentence expresses an ordering relation between two sets of entities with respect to some common features. For example, the comparative sentence "*Canon's optics are better than those of Sony and Nikon*" expresses the comparative relation 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 new way for automatically extracting comparable entities from comparative questions based on the pattern. We propose new techniques based on these two types of sequential rules to perform the tasks.

Introduction

Comparing alternative options is one of the essential things in decision-making that we carry out every day. Example, if someone is interested in certain products such as digital cameras, he or she would want to know what the different alternatives we have 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. And 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 World Wide Web, 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 users 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.

Comparisons can be subjective or objective. For example, a typical opinion sentence is "*The picture quality of camera x is great*" A subjective comparison is "*the picture quality of camera x is better than that of camera y*" An objective comparison is "*car x is 2 feet longer than car y*". We can see that comparative sentences use different language constructs from typical opinion sentences (although the first comparative sentence above is also an opinion). In this paper, we study the problem of comparative sentence mining. It has two tasks:

1. Given a set of evaluative texts, identify comparative sentences from them, and classify the identified comparative sentences into different types (or classes).
2. Extract comparative relations from the identified sentences. This involves the extraction of entities and their features that are being compared, and comparative keywords. The relation is expressed with (<relationWord>, <features>, <entityS1>, <entityS2>)

For example, we have the comparative sentence "*Canon's optics is better than those of Sony and Nikon.*" The extracted relation is: (better, {optics}, {Canon}, {Sony, Nikon})

Both tasks are very challenging. Although we see that the above sentences all contain some indicators i.e., "better", "longer", many sentences that contain such words are not comparatives, e.g., "*I cannot agree with you more*". The second step is a difficult information extraction problem. For the first task, we present an approach that integrates *class sequential rules* (CSR) and *naïve Bayesian classification* to perform the task. This task is studied in detail in (Jindal & Liu 2006). We include it for completeness. For the second task, a new type of rules called *label sequential rules* (LSR) is proposed for extraction. Our results show that LSRs outperform Conditional Random Fields (CRF) (Lafferty, McCallum & Pereira 2001), which is perhaps the most effective extraction method so far (Mooney & Bunescu 2005).

Types of Sequential Rules

We now start to present the proposed techniques, which are based on two types of sequential rules. Mining of such rules is related to mining of sequential patterns (SPM) (Agrawal and Srikant 1994). Given a set of input sequences, SPM finds all subsequences (called *sequential patterns*) that satisfy a user-specified minimum support threshold. Below, we first explain some notations, and then define the two new types of rules, *Class sequential rules* (CSR) used in classification of sentences, and *label sequential rules* (LSR) used in relation item extraction. For more details about these types of rules and their mining algorithms, please see (Liu 2006).

Let $I = \{i1, i2, ..., in\}$ be a set of items. A *sequence* is an ordered list of itemsets. An *itemset* X is a non-empty set of items. We denote a sequence s by $(a_1a_2...a_r)$, where a_i is an itemset, also called an *element* of s. We denote an element of a sequence by $\{x_1, x_2, ..., x_k\}$, where x_i is an item. An item can occur only once in an element of a sequence, but can occur multiple times in different elements. A sequence $s1 = (a_1a_2...a_r)$ is a *subsequence* of another sequence $s2 = (b_1b_2...b_m)$ or s_2 is a *supersequence* of s_1, if there exist integers $1 \leq j_1 < j_2 < ... < j_{r-1} \leq j_r$ *such that* $a_1 \subseteq b_{j1}, a_2 \subseteq b_{j2}, ..., a_r \subseteq b_{jr}$. We also say that s_2 contains s_1.

Class Sequential Rules

Let S be a set of data sequences. Each sequence is labeled with a class y. Let Y be the set of all classes, $I \cap Y = \emptyset$. Thus, the input data D for mining is represented with $D = \{(s_1, y_1), (s_2, y_2), ..., (s_n, y_n)\}$, where s_i is a sequence and $y_i \in Y$ is its class. A *class sequential rule* (CSR) is an implication of the form $X \rightarrow y$, where X is a sequence, and $y \in Y$. A data instance (s_i, y_i) in D is said to *cover* the CSR if X is a subsequence of s_i. A data instance (s_i, y_i) is said to *satisfy* a CSR if X is a subsequence of s_i and $y_i = y$. The *support* (sup) of the rule is the fraction of total instances in D that satisfies the rule. The *confidence* (conf) of the rule is the proportion of instances in D that covers the rule also satisfies the rule. Given a labeled sequence data set D, a minimum support (*minsup*) and a minimum confidence (*minconf*) threshold, CSR mining finds all class sequential rules in D.

Label Sequential Rules

A *label sequential rule* (LSR) is of the following form,

$X \rightarrow Y$,

where Y is a sequence and X is a sequence produced from Y by replacing some of its items with wildcards. A wildcard, denoted by a '*', matches any item. The definitions of support and confidence are similar to those above. The input data is a set of sequences, called *data sequences*.

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.

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.

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 sentence 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. $2n - 1$ 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.

Conclusion

This paper studied the new problem of identifying comparative sentences in evaluative texts, and extracting comparative relations from them. Two techniques were proposed to perform the tasks, based on class sequential rules and label sequential rules, which give us syntactic clues of comparative relations. Experimental results show that these methods are quite promising.

References

1. Mary Elaine Califf and Raymond J. Mooney. 1999.Relational learning of pattern-match rules for information extraction. In *Proceedings of AAAI'99 /IAAI'99*.
2. Claire Cardie. 1997. Empirical methods in information extraction. *AI magazine*, 18:65–79.
3. Nitin Jindal and Bing Liu. 2006a. Identifying comparative sentences in text documents. In *Proceedings of SIGIR '06*, pages 244–251.
4. Nitin Jindal and Bing Liu. 2006b. Mining comparative sentences and relations. In *Proceedings of AAAI '06*.
5. Greg Linden, Brent Smith and Jeremy York. 2003.Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing*, pages 76-80.
6. Raymond J. Mooney and Razvan Bunescu. 2005. Mining knowledge from text using information extraction. *ACM SIGKDD Exploration Newsletter*, 7(1):3–10.
7. 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.
8. Deepak Ravichandran and Eduard Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of ACL '02*, pages 41–47.
9. 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.
10. Ellen Riloff. 1996. Automatically generating extraction patterns from untagged text. In *Proceedings of the 13th National Conference on Artificial Intelligence*, pages 1044–1049.
11. Stephen Soderland. 1999. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3):233–272.