ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 8, August 2015

Product selection from comparative question using cosine similarity analysis and weight feature analysis

Miss. Pranali Patil[1], Mr. Sudip Tembhurne[2], Mr. Rugved Deolekar[3]

M. E. Student, Computer Engineering, VIT Mumbai, India[1,2]

Assistant Professor, Department of Computer Engineering, VIT Mumbai, India[3]

Abstract: Comparing one product with another product when anyone want to purchase products is very typical decision making process. However it is not always easy to know what product compare and what are the alternatives are available so to address such difficulty we propose a novel way that can help user to get best product by calculating feature weight of respective products. The feature weight of product is calculated by using weight analysis. When any new product is added or any new user given rank then the table where total rank is going to calculate for every product and for every feature is updated as per weighted feature algorithm .On that basis when user gives query to application then it will check first is that is a comparable question or not by using cosine similarity analysis and show result the best from those products respectively.

Keywords:Data mining, compare, weight feature analysis, cosine similarity algorithm.

I. INTRODUCTION

Comparing alternative options while purchasing something is one essential step in decision-making that we carry out every day. For example, if someone is interested in certain products such as Laptops or Mobile phones , the person would want to know what the alternatives are and compare different Laptops or Mobiles 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. Search for relevant web pages containing information about the targeted products, find competing products, read reviews, and identify pros and cons.

Here we are finding a set of comparable entities given a user's input entity. For example, given an entity, Dell L124 (a Laptop), we want to find comparable entities such as Dell S5, HP N123 and so on. In general, it is difficult to decide if two entities are comparable or not since people are 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 sometimes we rarely see people comparing car models "Ford Focus" and "BMW 68i". Things get more complicated when an entity has several functionalities or we can say in case of laptop and mobile as they are having several features. For example, one might compare "iPhone" and "PSP" as "portable game player" wharere compare "iPhone" and "Nokia N95" as "mobile phone". So 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 so In this paper, we define comparative questions and 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.
"Inspiron 5110 15r" and "Vostro 14 V3456 Notebook" are comparators.
Q1: "Which one is better?"
Q2: "Is Inspiron 5110 15r the best camera?"
Q3: "What‟s the difference between Inspiron 5110 15r Touch and Vostro 14 V3456 Notebook?"

The goal of this work is mining comparators from comparative questions. 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 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.

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 8, August 2015

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.

II. RELATED WORK

[1] Nitin jindal and Bingliu introduced the problem of identifying comparative sentences in text documents. The problem is related to but quite different from sentiment/opinion sentence identification or classification. Sentiment classification studies the problem of classifying a document or a sentence based on the subjective opinion of the author. An important application area of sentiment/opinion identification is business intelligence as a product manufacturer always wants to know consumers' opinions on its products. Comparisons on the other hand can be subjective or objective. Furthermore, a comparison is not concerned with an object in isolation. Instead, it compares the object with others. An example opinion sentence is "the sound quality of CD player X is poor". An example comparative sentence is "the sound quality of CD player X is not as good as that of CD player Y". Clearly, these two sentences give different information.

A subjective comparative sentence may be
"Car X is much better than Car Y"
An objective comparative sentence may be
"Car X is 2 feet longer than Car Y"

Such sentences are useful in many applications, e.g., marketing intelligence, product benchmarking, and e-commerce. We first analyzed different types of comparative sentences from both the linguistic point of view and the practical usage point of view, and showed that existing linguistic studies have some limitations.

[2] Greg Linden, Brent Smith, and Jeremy York At Amazon.com, we use recommendation algorithms to personalize the online store for each customer. The store radically changes based on customer interests, showing programming titles to a software engineer and baby toys to a new mother. The click-through and conversion rates — two important measures of Web-based and email advertising effectiveness — vastly exceed those of untargeted content such as banner advertisements and top-seller lists.

Most recommendation algorithms start by finding a set of customers whose purchased and rated items overlap the user's purchased and rated items.2 The algorithm aggregates items from these similar customers, eliminates items the user has already purchased or rated, and recommends the remaining items to the user. Two popular versions of these algorithms are collaborative filtering and cluster models.

[3] Nitin jindal and Bing liu introduced a text mining problem, comparative sentence mining. 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: (better, {optics}, {Canon}, {Sony, Nikon}). Given a set of evaluative texts on the Web, e.g., reviews, forum postings, and news articles, the task of comparative sentence mining is (1) to identify comparative sentences from the texts and (2) to extract comparative relations from the identified comparative sentences. This problem has many applications. For example, a product manufacturer wants to know customer opinions of its products in comparison with those of its competitors.

[4] Claire Cardie explained the use of empirical, machine learning methods for a particular natural language–understanding task—information extraction. The author presents a generic architecture for information-extraction systems and then surveys the learning algorithms that have been developed to address the problems of accuracy, portability, and knowledge acquisition for each component of the architecture.

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

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 8, August 2015

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 $S(s1s2 \ldots sn)$ 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 $S(s1s2 \ldots si \ldots sn)$ to a labelled sequence $S'(s1s2 \ldots li \ldots sn)$ by replacing one token (si) in the input sequence with a designated label (li). This token is referred as the anchor. The anchor in the input sequence could be extracted if its corresponding label in the labelled 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 relationextraction 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 *li* & *rj 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 over fitting, and allow rules to include discriminative lexical tokens to retain precision.

III. SYSTEM ARCHITECHURE

3.1 Proposed Architecture is developed for to get the best product while user purchasing things as comparing one thing with another is typical human decision making process. In this architecture we are giving various options to user to select the product by calculating best feature. Here when user type the comparative question the system will identify that question typed by user is comparable or not, If question is comparable then it will extract the products from the given question. Once we get the product we can apply weighted analysis algorithm and by calculating total feature of every product final result will get.Proposed system mainly consists of 2 modules i.e.Admin and User Module. Admin will add product, query, and feature rate.

3.2 Steps

Step1) Store Sequential patterns [indicative extraction pattern (IEP)] and comparator data also to use as pivots to create sequence data.

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 8, August 2015

Step2) Take user input as comparative Question.

Step3) Compare with Sequential patterns and find closely related pattern format by using cosine similarity algorithm and which are passing to the next step.

Step4) If (Sequential patterns >1)
{
Select one of them by comparing other comparators like max cosine similarity and max ranking value from ranking
}

Step5) on the basis of those i.e. sequential patterns and comparator data we get the comparable attributes of both the products which will compare in all terms and gives us result as in required pattern.

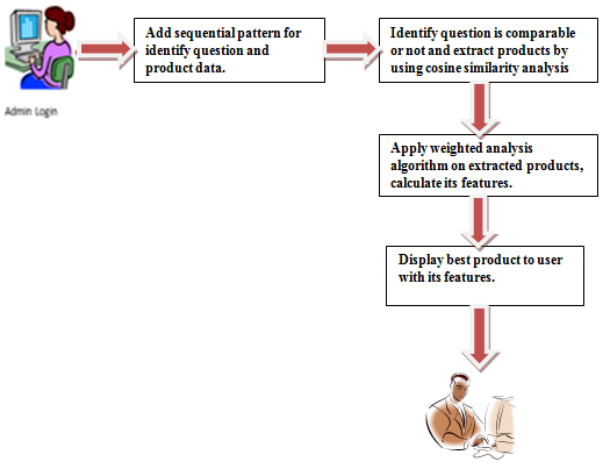

Fig.3.1 System Architecture

3.3 Cosine similarity algorithm used for finding closely related patterns

1) Take all sequential patterns in one String array take the input from user. i.e. comparator question
2) Calculate frequency of all strings and user input string also on the basis of count of how many words comes in sequence patterns repeatedly.
3) By using these frequencies we can calculate cosine similarity of all patterns

cosine similarity = a.b/((a^2)* (b^2))

Where,

a = user input comparator question's cosine similarity
b = sequential patterns cosine similarity

4) So when we get cosine similarity we can easily find closely related pattern among all.

3.4 Weight Analysis Algorithm

Total Weight of each product will get calculate by adding weight of every attributes of single item. Some attributes whose values will be number, their values will get calculated using Step 1 and weight of some attributes will get calculated using step 2. Then it will sum these weights which will return weight of single item.

➢Step 1:

➔Find Threshold = sum of columns/ total rows(or items).

➔Find max value among all.

➔assign x= weight(feature rank) to max value
= Threshold + (total rows-count).

➔Again find max among remaining items and assign weight.

➔Follow this steps until all finish.

Note :=> this weight calculation will be for that features or attributes whose value will contain integer values.
like memory, camera(3 MP or 4 MP etc).

➔Admin will assign some weight(w) to all attributes.
for example:(w1) Camera: 3, (w2)RAM: 5, (w3)HDD: 4
Then total rank(of each item)= w1x1+w2x2+w3x3

➢Step 2: Some Features like OS, Processor System will take rating from users and then it will find its average as weight.

IV. CONCLUSION

In this paper, we present cosine similarity algorithm to identify comparative questions and extract comparator pairs simultaneously, that means we can extract the products. 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 cosine similarity system. By leveraging large amount of unlabeled data and the cosine similarity analysis
The experimental results show that our method is effective in both comparative question identification and comparator extraction. Now once we get products in between user want to compare we can apply weight analysis algorithm in which the features of the products are calculated and depend on highest rating on product the best answer return to user .

REFERENCES

[1] Nitin Jindal and Bing Liu. 2006a. Identifying comparative sentences in text documents. In *Proceedings of SIGIR '06*, pages 244–251.
[2] Greg Linden, Brent Smith and Jeremy York. 2003. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing*, pages 76-80.
[3] Nitin Jindal and Bing Liu. 2006b. Mining comparative sentences and relations. In *Proceedings of AAAI '06*.
[4] Claire Cardie. 1997. Empirical methods in information extraction. *AI magazine*, 18:65–79.
[5]Deepak Ravichandran and Eduard Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of ACL '02*, pages 41–47.
[6]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.
[7]Ellen Riloff. 1996. Automatically generating extraction patterns from untagged text. In *Proceedings o fthe 13th National Conference on Artificial Intelligence*,pages 1044–1049.
[8]Stephen Soderland. 1999. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3):233–272.