Electronic Commerce Research and Applications 30 (2018) 62–71

Contents lists available at ScienceDirect

Electronic Commerce Research and Applications

journal homepage: www.elsevier.com/locate/elerap

Aspect ontology based review exploration

Anand Konjengbam*, Neelesh Dewangan, Nagendra Kumar, Manish Singh

Indian Institute of Technology Hyderabad, 502285, India

A R T I C L E I N F O

Keywords:
Electronic commerce
Review exploration
Opinion mining
Aspect ontology

A B S T R A C T

User feedback in the form of customer reviews, blogs, and forum posts is an essential feature of e-commerce. Users often read online product reviews to get an insight into the quality of various aspects of a product. Besides, users have different aspect preferences, and they look for reviews that contain relevant information regarding their preferred aspect(s). However, as reviews are unstructured and voluminous, it becomes exhaustive and laborious for users to find relevant reviews. Lack of domain knowledge about various aspects and sub-aspects of a product, and how they are related to each other, also add to the problem. Although this information could be there in product reviews, it is not easy for users to spot it instantly from the reviews. This paper seeks to address the above problems and presents two novel algorithms that summarize product reviews, and provides an interactive search interface, similar to popular faceted navigation. We solve the problem by creating an aspect ontology tree with high aspect extraction precision.

1. Introduction

The proliferation of the Internet and Internet devices over the last decade has changed people's way of living. It has become routine to see users making extensive use of online portals and e-commerce sites to purchase goods and services. After using the products, customers often write reviews and give feedback expressing their opinions on various product aspects. This results in an enormous number of reviews left on the web by customers. Mining and extracting opinions from these reviews could help customers in making informed purchase decisions.

Opinion mining is the process of extracting user opinions from textual data and determining the polarity of the extracted opinions. It is also commonly known as *sentiment analysis*. Opinion mining is a popular research area (Fang and Zhan, 2015; Liu, 2012). Most of the existing work is focused on sentiment classification Liu (2012), which could be for a whole review, each sentence of a review, or for each aspect mentioned in a review. Aspects are primarily the product features on which users express their opinions. Part-of-speech tags with other NLP grammar rules are used to extract aspects and opinion words. However, the main limitation of these techniques is that the relationship between different aspects remain neglected. For example, finding synonym aspects, finding hierarchical relationship between the extracted aspects, and so on.

Many of the existing opinion mining systems do not focus on high-precision aspect extraction as they do not explain about the extracted aspects to users. In this paper, we propose an exploratory search interface where users can view aspects mined from reviews with high precision, along with the semantic relationship between the aspects. By using our interface, users can easily access all snippets of the reviews that talk about a preferred aspect and all the sub-aspects thereof. To build the system, we mine the ontological relationship between aspects using product reviews and semantic knowledge from external sources.

Our main focus is on the automatic creation of aspect ontology using semantic knowledge between the aspects. The whole process may be divided into the following subtasks: (1) extract aspects from product reviews; (2) construct an ontology tree of extracted aspects; and (3) summarize reviews based on the aspect ontology tree.

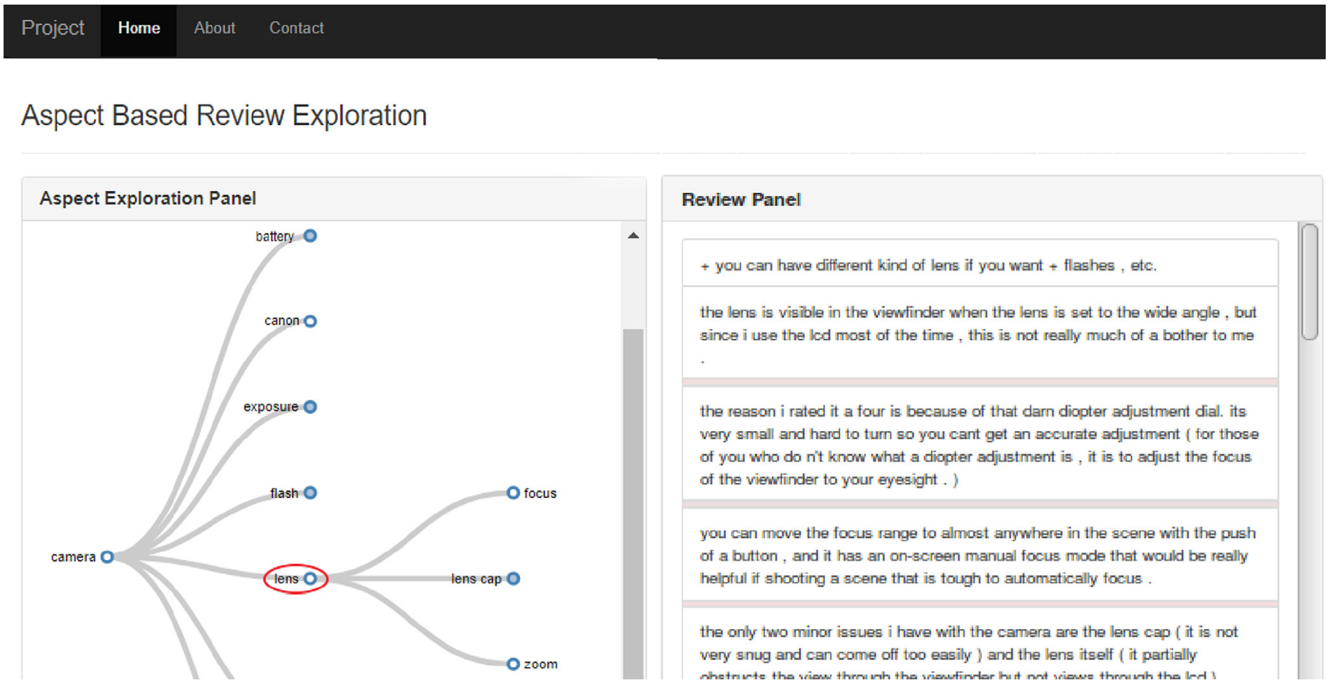
Fig. 1 shows a screenshot of our proposed ontology-based review exploration system. Here, we take the example of browsing through digital camera reviews. The interface consists of two panels, namely the aspect exploration panel and the review snippet panel. When the user clicks on the aspect *lens* in the aspect panel, the sub-aspects, such as *focus*, *lens cap* and *zoom* pop out. Also, review snippets related to *lens* and all its sub-aspects are shown in the review snippet panel.

Such a summary is useful to users in many ways. First, the user does not need to go through all the reviews while looking for information on some specific aspects. Using this interface, the user can click on the aspects of interest, and focus only on the aspect relevant review snippets that are shown on the review panel. Second, and more importantly, if the user is new to the product, the relationship between various aspects and sub-aspects of the product can be understood using our proposed review exploration system. Our system integrates the aspects with the reviews.

Unlike any other review summarization methods (Hu and Liu,

* Corresponding author.
E-mail addresses: cs14resch11004@iith.ac.in (A. Konjengbam), cs14mtech11009@iith.ac.in (N. Dewangan), cs14resch11005@iith.ac.in (N. Kumar), msingh@iith.ac.in (M. Singh).

https://doi.org/10.1016/j.elerap.2018.05.006
Received 21 September 2017; Received in revised form 14 May 2018; Accepted 14 May 2018
Available online 17 May 2018



+ you can have different kind of lens if you want + flashes , etc.

the lens is visible in the viewfinder when the lens is set to the wide angle , but since i use the lcd most of the time , this is not really much of a bother to me .

the reason i rated it a four is because of that darn diopter adjustment dial. its very small and hard to turn so you cant get an accurate adjustment (for those of you who do n't know what a diopter adjustment is , it is to adjust the focus of the viewfinder to your eyesight .)

you can move the focus range to almost anywhere in the scene with the push of a button , and it has an on-screen manual focus mode that would be really helpful if shooting a scene that is tough to automatically focus .

the only two minor issues i have with the camera are the lens cap (it is not very snug and can come off too easily) and the lens itself (it partially obstructs the view through the viewfinder but not views through the lcd)

Fig. 1. Screenshot of ontology-based review exploration system.

2004a; Zhuang et al., 2006), the summarization method described is unique in a number of ways. First, we arrange the aspects using a semantic ontology tree rather than presenting them in random order. This aids in better understanding of the product. Second, we allow users to view review snippets at different degree of details, starting from generalized to fine-grained levels by exploring the various levels of the aspect ontology tree.

The remaining paper is organized as follows. Section 2 contains the related work on sentiment analysis and review summarization. Section 3 includes a detailed description of our review exploration system. The experimental evaluations and results are presented in Section 4. Section 5 contains the conclusion and future scope of the study.

2. Related works

Opinion mining or sentiment analysis on product reviews has been studied extensively. Sentiment analysis can be done either for the whole review (Dave et al., 2003; Turney, 2002) or for each aspect of the product (Mukherjee and Liu, 2012; Pontiki et al., 2016). Although sentiment classification of whole review is an interesting problem, the research done on aspect level sentiment analysis are considered more important. The reason is that one is more interested in knowing what specifically the users liked or disliked compared to just knowing whether their overall opinion is positive, negative or neutral. There are many supervised and unsupervised methods to solve the above two problems. Supervised methods (Zhang and Lin, 2018; Zhou and Duan, 2012) rely on training data for sentiment classification, while unsupervised approaches (Das and Chen, 2007; Hu and Liu, 2004a) use syntactic rules and lexical resources for polarity mining.

Our work is closely related to aspect based opinion mining and opinion summarization. To solve this problem, one needs to precisely find the targets of opinions (Liu, 2012). This problem is slightly easier for product reviews compared to other types of opinion data, such as debates, blogs, etc. (Somasundaran and Wiebe, 2009). The reason is that most of the product reviews are about one entity, and the opinions are expressed on different aspects of that one entity. However, due to the presence of multiple entities and other noisy information in other data, it is difficult to precisely map the opinions to the respective entity and aspect.

The main limitation of existing aspect-based opinion mining algorithms is that they do not consider the sub-entities within an entity. For example, *lens* is a sub-entity of camera. Whereas the aspects, such as *focus*, *lens cap*, and *zoom*, are aspects of lens and not of camera directly. Existing works (Pontiki et al., 2016; Hu and Liu, 2004b) do not consider this ontology of entities and sub-entities and their corresponding aspects while doing opinion mining and summarization. For simplicity they treat all sub-entities and aspects within a broader frame 'aspects'. This often leads to erroneous opinion mining. For example, consider the sentence: 'The casing of this camera is very ugly'. Here the opinion *ugly* is for the appearance aspect of casing and not the appearance aspect of the camera.

Aspects can be extracted using: semi-automatic approach (Kobayashi et al., 2004), association rule mining (Hu and Liu, 2004b), and pointwise mutual information (Popescu et al., 2005). Scaffidi et al. (2007) developed an aspect-based product search interface that extracts product aspects and assigns scores to each product aspect. Huang and Cheng (2015) proposed an aspect based summarization approach for summarizing car and movie reviews, which is based on pattern mining and association rule mining. Aspect summarization from short comments was studied by Lu et al. (2009). In all these works, the relationship among the aspects while summarizing the reviews was not considered. Another set of related works focus on aspect aggregation (Liu et al., 2005; Bollegala et al., 2007), which means uncovering aspects that are synonymous. But, such methods of aspect aggregation fail to extract the hierarchical relationships between aspects. Use of textual clues to extract aspect terms and relations among them was explored by Kobayashi et al. (2007).

Next we present related work on constructing ontologies. Shein (2009) used Formal Concept Analysis (FCA) for constructing movie ontology. The work by Lee et al. (2006) focused on building an operational ontology system to represent product information. They followed a three-level meta modelling approach and implemented the ontology using Web Ontology Language (OWL), which is hard to train and implement. Lee and Goodwin (2006) worked on developing an enterprise-scale ontology management system and proposed an approach to represent ontologies in relational database tables. Faria et al.

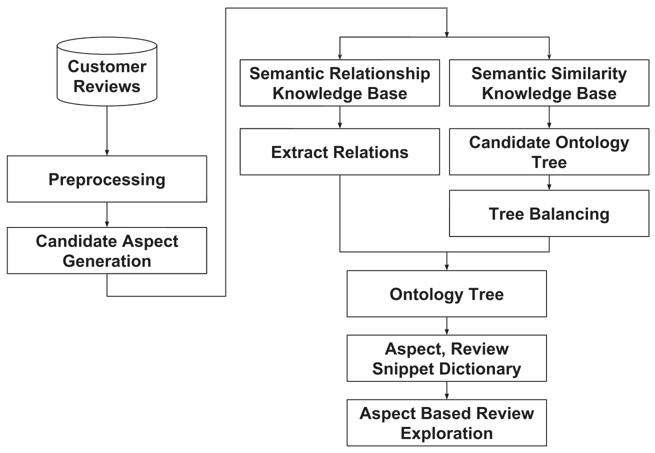

Customer Reviews
Preprocessing
Candidate Aspect Generation
Semantic Relationship Knowledge Base
Semantic Similarity Knowledge Base
Extract Relations
Candidate Ontology Tree
Tree Balancing
Ontology Tree
Aspect, Review Snippet Dictionary
Aspect Based Review Exploration

Fig. 2. System architecture of aspect-based review exploration system.

(2014) used natural language processing and information extraction techniques to acquire and classify ontology instances. Kang et al. (2014) focused on extracting key concepts that can be used to create the ontology. The use of ontology to help in solving the heterogeneity problem of e-commerce was explored by Malucelli et al. (2006). However, extant ontology construction methods require manual input in some form for the creation of ontology tree. Some of the methods are domain specific, while others require a pre-defined ontology rules or structure. In contrast to the aforementioned works, we propose an automatic approach to generate the aspect ontology tree using similarity techniques that work across domains.

3. Proposed approach

In this section, we formally introduce our aspect-based review exploration system. The architectural overview of the system is given in Fig. 2.

The system takes product reviews as input and generates output as an aspect-based review summary, where the aspects are presented in the form of an ontology tree. We primarily focus on the automatic creation of aspect ontology, negating the need for any prior domain knowledge. Table 1 contains the list of notations used.

Algorithm 1 gives an overview of the Aspect-based Review Explorer (ARExplorer) system. ARExplorer has four main steps: (1) extract candidate aspects from reviews; (2) extract popular aspects; (3) create aspect ontology tree using semantic knowledge base; and (4) create aspect-based review exploration system.

Table 1
Notations used.

Notation	Description
A_C	Candidate aspects extracted from the reviews
A_P	Frequent product aspects present in the reviews
A_T	Subset of A_P closely related to the product
D	Dictionary containing counts of aspects in A_P
R	Relation list obtained from ConceptNet
BDR	Bidirectional relations obtained from ConceptNet
BUR	Buttom-Up relations obtained from ConceptNet
TBR	Top-to-Bottom relations obtained from ConceptNet
F	Functional relations obtained from ConceptNet
H	Hierarchical relations obtained from ConceptNet
S	Synonym relations obtained from ConceptNet
K_S	Semantic knowledge base obtained using LSA
$SemR$	Algorithm for ontology creation using semantic relationship
$SemS$	Algorithm for ontology creation using semantic similarity
T_O	Aspect ontology tree

Algorithm 1 ARExplorer(D_R, P, K_S)

```
Input:   $D_R$: Review dataset
         $P$: Product domain
         $K_S$: Semantic knowledge base
Output:  $T$: Aspect-based review exploration system
Method:
1: $A_C \leftarrow ExtractCandidateAspect(D_R, P)$
2: $A_P \leftarrow ExtractPopularAspect(D_R, A_C)$
3: $T_O, A_T \leftarrow CreateOntologyTree(A_P, K_S)$
4: $D_{AR} \leftarrow AspectReviewDictionary(A_T, D_R)$
5: $T \leftarrow ARESys(T_O, D_{AR})$
```

Given a review corpus D_R and a product domain P, Step 1 parses and extracts all the candidate aspects A_C from the reviews. From the list of extracted aspects, only popular aspects A_P are selected, using Step 2, for further processing. The system then uses Step 3 to build the aspect ontology tree T_O from A_P using semantic knowledge base. In this step, some more aspects which might be popular but have little semantic relationship with the product also get pruned. Two approaches are examined for creating the ontology tree, one using a semantic relatedness knowledge base and the other using a semantic similarity knowledge base. In Step 4, the aspects A_T in the ontology tree are mapped to related review snippets. After that, Step 5 creates an interactive aspect-based review search interface. Users can explore product reviews by clicking on the aspects. In addition, users also have the option to view the sub-aspects and their related reviews. New reviews can be easily added to the system. With the addition of new reviews, we only need to index those reviews using the aspects present in the ontology. However, only if a new aspect becomes popular in the new reviews and was not popular in the previous reviews, such aspect might be added to the ontology tree. Our main contribution is the automatic creation of the ontology tree for better aspect-based review exploration. We next discuss the steps in detail.

3.1. Candidate aspect set generation

The purpose of this step is to extract the candidate aspects from reviews. Aspects are usually represented by nouns or noun phrases in reviews. Part-of-speech (POS) tagging and dependency tree parsing are the commonly used techniques for extracting aspects. Many other methods for aspect extraction have been explored (Eirinaki et al., 2012; Liu et al., 2016). These techniques may also give a broad range of aspects and sub-aspects as some users comment on specific aspects that might fall outside common interest of other users. For our aspect ontology tree, too many aspects might cause an aspect overload and confuse users during aspect exploration. To tackle this problem, we attempt to filter the aspects to get only the more popular ones. We define popular aspects as those aspects which appear in more than a threshold (T_P) number of review. This approach also helps in pruning out the irrelevant nouns, which are wrongly detected as aspects. Typos, text-speaks, or Martian languages also gets filtered in this step as they occur few times, compared to popular aspects. However, there may be popular aspects which have low semantic relationship with the products like *camera* and *software*. We prune out such unrelated aspects using semantic knowledge during aspect ontology tree generation.

3.2. User interface

Our aspect-based review exploratory system consists of a web-based interface (as shown in Fig. 1) that enables the user to browse through various product aspects and go through the related review snippets. The user interface supports multiple levels of review abstraction with the help of the aspect exploration panel on the left side and the review panel on the right side of the screen. The aspect exploration panel displays aspect ontology of the product, and the review panel shows review snippets. Initially, the review panel consists of overall reviews of the product if no product aspect is selected. Whenever a user clicks on an aspect, related sub-aspects pop out and the review panel displays review snippets related to the aspect. The sub-aspects are kept hidden until the user clicks on the related parent aspect. This way, the user does not have to go through distracting irrelevant aspects. The added feature of this interface is that it is interactive. It makes easy for the end user to identify product aspects of interest, and focus on reviews that contain relevant aspect information.

3.3. Aspect ontology tree generation

Ontologies represent the concepts in a domain as well as the relations among them in the form of a hierarchical tree structure. As the aspects and sub-aspects have hierarchical relationship, we use a tree structure to represent the aspect ontology. Aspect ontology tree illustrates the relations among aspects of a product and helps in conceptualizing product specific information by adding semantic relationship to various aspects. In an aspect ontology tree, aspects are represented as nodes while edges outline the semantic relationship between the aspects.

An aspect ontology tree exhibits aspect information at different granularity levels. For example, *focus* and *zoom* are kept as a more fine-grained sub-aspects of *lens*. This helps resolve the aspect overload problem by displaying a group of aspects at various fine-grained levels. The sub-aspects can be viewed at a finer level by following the path of higher level aspects. This way, users are able to comprehend the relationship among aspects and sub-aspects better. Here, we propose two approaches for building the aspect ontology tree using semantic relationship and semantic similarity. We further explore generating the aspect ontology from automatically extracted aspects, as manually labeled aspects are not always readily available. The semantic relationship method misses out some of the aspects as the semantic relationship knowledge base is too big to build manually. On the other hand, the semantic similarity method is able to cover more aspects as the aspects usually co-occur with the product in documents.

3.3.1. Ontology using semantic relationship (SemR)

A semantic relationship between two aspects can be used to convey the association between the two. For example, the sentence 'Camera has a lens' shows the connection that *lens* is-part-of *camera*. We use ConceptNet (Liu and Singh, 2004) to find the semantic relationship between the extracted aspects.

ConceptNet consists of a wide-range of common sense knowledge, which are optimized for making practical context-based inferences over real-world texts. The common sense knowledge contains basic facts and understanding between concepts. The concepts are real world objects. For example, consider the following sentences: (1) 'Camera has a lens'; (2) 'Camera is used for taking pictures'. The above sentences provide a basic understanding about a *camera*. The sentences contain relationship of concepts such as *has a* and *used for*. ConceptNet can be used to find semantic relationships between concept pairs. Such knowledge contains relationship between concepts, for instance, '*has a*' and '*used for*' in the above example.

Fig. 3 shows an instance of actual knowledge in ConceptNet concerning the concept *camera*. Each node represents a concept, and each directed edge represents a semantic relationship between the concepts. We leverage these relations between the aspects to build the aspect ontology tree.

ConceptNet has 24 common relations such as *PartOf*, *MadeOf*, *AtLocation*, etc., to describe how concepts are related through common sense knowledge. These relations can be leveraged to find a connection between concepts that are not directly connected. For example, 'camera is *RelatedTo* lens', 'lens is *UsedFor* focus'. Here, it is evident that *camera* and *focus* are related. However, there may be one-to-many relations between various concepts. Linking multiple concepts may lead to topic drift. For example, consider the ConceptNet statements 'camera is *RelatedTo* lens', 'lens is *MadeOf* glass' and 'glass is *RelatedTo* window '. Here *camera* and *window* are related as per ConceptNet but *window* doesn't belong to camera domain. To control topic drift, as in Mukherjee and Joshi (2013), we use three classes of relations: Hierarchical relations (H), Synonym relations (S), and Functional relations (F), with assigned priority order of $H > S > F$. We use these relation classes and the priority order to choose the proper relation in case multiple possible relationships exist between two aspects. After manual inspection, we consider only 12 types of relations that highlight the aspects related to a concept (e.g. HasA, PartOf, etc.) and discard others which are not useful for building the ontology tree (such as *MotivatedByGoal*, *ObstructedBy*, etc.). The relations along with relation class and priority order are shown in Table 2.

The relations between concepts are represented using undirected edges between concepts. For example, 'lens is PartOf camera' will result in an undirected edge between *lens* and *camera* in the ontology tree. Due to the undirected edge, it is hard to find out the parent and child aspect. However, the correct parent-child relationship is necessary to identify aspects and sub-aspects. To resolve this, we propose the concept of *edge-type* to differentiate aspects and sub-aspects in the commonsense knowledge, based on semantic relationship. We classify relations into three edge types: Top to Bottom Relation (TBR), Bottom to Up Relation (BUR) and Bidirectional Relation (BDR). Relations belonging to these edge types are given in Table 3.

TBR represents parent-to-child relationship while BUR exhibit child-to-parent relationship. BDR is a particular case that exhibits the properties of both TBR and BUR. Depending on these edge types, parent and child aspects are decided. For example, let us consider the following statements: (1) 'lens is *PartOf* camera'; (2) 'camera is *CapableOf* taking pictures'; (3) 'focus is *RelatedTo* lens'.

In the first statement, *lens* and *camera* has a BUR, hence an edge is added from *camera* to *lens* ($camera \rightarrow lens$) to the ontology tree. Similarly, for the other two examples, edges ($camera \rightarrow picture$) and ($focus \leftrightarrow lens$) are added to the ontology tree as they have TBR and BDR respectively. For bi-directional edges, we have two possibilities of adding edge (e.g. $A_1 \rightarrow A_2$ or $A_2 \rightarrow A_1$). A_2 can become sub-aspect of A_1 or vice versa. Hence to maintain tree consistency, we direct edges only from top to bottom.

In addition to edge types, we use Breadth First Search (BFS) exploration for our ontology expansion. In other words, we extract relevant concepts from ConceptNet by providing the product domain initially and connect the aspect edges based on edge type in BFS manner. The process is repeated recursively as long as the child concept belongs to the set of frequently occurring aspects. Benefits of using BFS exploration is that it provides the shortest path to relevant concepts from product domain at minimum depth.

For many-to-many relation edge conflict, $H > S > F$ order evaluation is followed. For instance, if we consider the relations (1) *shutter* is *RelatedTo camera* and (2) *shutter* is *PartOf lens*. Here, *shutter* has Hierarchical (H) relation with lens and Synonymous (S) relation with camera. Accordingly, we add an edge from lens to shutter ($lens \rightarrow shutter$). Also, to cluster all the relevant synonyms at single respective aspect, we set inner priority order within synonym relation class as mentioned in Table 2. In case of conflicting relations, an edge is added in the ontology tree for a higher priority relation, and we ignore the lower priority relation edge. Finally, we merge all the synonyms to remove the redundant information from the tree. Our SemR approach is described in Algorithm 2.

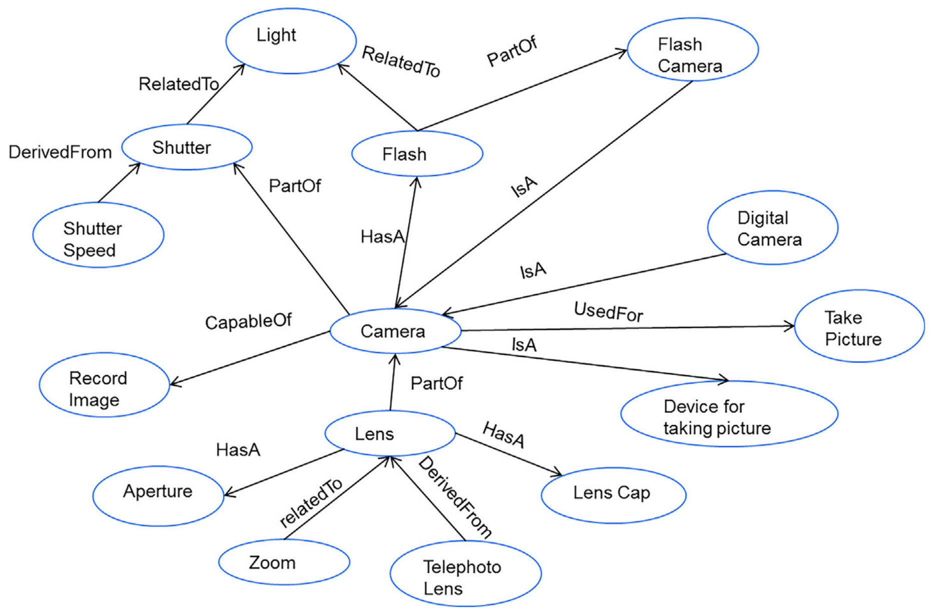


Fig. 3. A snippet from ConceptNet's semantic network, showing various concepts related to the entity *camera*.

Table 2
ConceptNet relations used for building the ontology tree along with their relational class and priority order. '>' shows inner class priority order.

Relational class	Relation	Priority
Hierarchical Relations (H)	*HasA, PartOf, MadeOf, LocatedNear*	1
Synonym Relations (S)	*Synonym, DefinedAs > DerivedFrom > IsA, RelatedTo*	2
Functional Relations (F)	*UsedFor, CapableOf, HasProperty*	3

Table 3
Edge types and relation list.

Edge type	Relations
Top to Buttom Relations (TBR)	*CapableOf, HasA, HasProperty, LocatedNear, MadeOf, UsedFor*
Bottom Up Relation (BUR)	*DerivedFrom, PartOf*
Bidirectional Relation (BDR)	*DefinedAs, IsA, RelatedTo, Synonym*

Algorithm 2 Semantic Knowledge Ontology Tree (SemR)

```
Input:   D: Popular aspects dictionary = <$A_P$,$f$>, where
         $A_P$: Popular aspect set = {$A_{P_1}$,$A_{P_2}$,...$A_{P_n}$}
         $f$: Frequency of each candidate aspect = {$f_1$,$f_2$,...$f_n$}
         $R$: Relation List = [$H$,$S$,$F$]
         $P$: Product domain
Output:  $T_O$: Aspect ontology tree
         $A_T$: Ontology aspects
Method:
1: Vertex set, $V \leftarrow \Phi$, Edge set, $E \leftarrow \Phi$, Queue, $Q \leftarrow \Phi$
2: Graph $G \leftarrow (V,E)$
3: $Q$.enqueue($P$)
4: visited[$P$]← true
5: for all relation ∈ $R$ do
6:   while $Q \neq \Phi$ do
7:     Concept, $c \leftarrow Q$.dequeue()
8:     $V' \leftarrow$ ExtractConnectedNodes(c) if $r_{new} \in R$
9:     for all $v_i \in V'$ do
10:      if $v_i \in A_P$ and $v_i \notin$ visited then
11:        if $v_i \notin G$ then
12:          $V.add(v_i)$
13:          if $r_{new} \in TBR$ or $r_{new} \in BDR$ then
14:            $E.AddEdge(c, v_i)$
15:          else
16:            $E.AddEdge(v_i, c)$
17:          end if
18:        else
19:          $r_{old} \leftarrow OldRelation(parent, c)$
20:          if priority($r_{new}$)> priority($r_{old}$) then
21:            parent($v_i$) ← $c$
22:          end if
23:        end if
24:        $Q$.enqueue($v_i$)
25:        visited[$v_i$]←true
26:      end if
27:    end for
28:  end while
29: end for
30: $T_O, A_T \leftarrow MergeSynonyms(G)$
```

SemR takes the product domain as the root of the tree. It follows Breadth First Search (BFS) for creating the ontology. For this, a queue is maintained to store the nodes. Steps 1–2 initialize the tree. For processing, the product domain is put in a queue (Q) and is marked as visited (Steps 3–4). For each of the relations, BFS traversal is applied to create the ontology tree (Steps 5–29). A node (c) gets dequeued from Q in Step 7, and in Step 8, we extract the connected nodes if the relationship type is in R. For all the extracted aspects, Step 10 checks if it is popular and is not already present in the tree. If so, it is added as a node of the tree (Step 12). An edge is added between parent and the new node according to the edge type (Steps 13–18). If the aspect is already present in the ontology, we compare whether the new relation r_{new} has a higher priority than the older relation (Steps 20–22). If so, c is made the parent of the new aspect. After adding the new aspect as a node, it is enqueued and marked as visited. This process is repeated until Q is empty. Finally, in Step 25, we merge similar aspects using Wordnet to avoid duplicates.

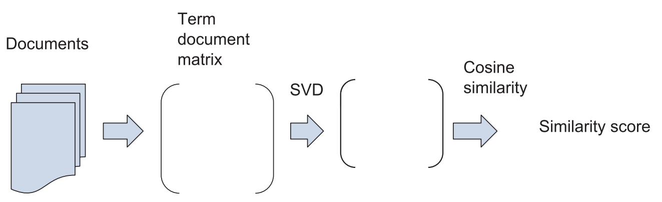


Fig. 4. Computing LSA score between words.

3.3.2. *Ontology using semantic similarity (SemS)*

The relations present in the knowledge base limits the semantic relationship-based approach. Aspects which are not present in the knowledge base are not considered while creating the aspect ontology. To overcome this constraint, we propose a method to identify the semantic similarity between aspects and use the similarity score to construct the aspect ontology. This is based on the idea that related aspects usually appear together in documents. For example, in the review: 'The camera has a maximum lens opening of f2.8 and a maximum shutter speed of 1/1000 s', the related aspects *lens* and *shutter speed* appear together. We chose Latent Semantic Analysis (LSA: Landauer, 2006) as a metric for measuring semantic similarity as LSA works on the notion that related words occur in similar documents. LSA uses a term-document matrix to describe the occurrence of terms in documents and then uses Singular Value Decomposition (SVD) for dimensionality reduction. Finally, terms are compared using cosine similarity between the term vectors. Values close to 1 represent closely related aspects while values close to 0 represent non-related aspects. LSA represents words as vectors in high-dimensional space. The similarity of two texts can be computed using the cosine similarity of their LSA vectors. Fig. 4 shows how the LSA score is computed between words.

To compute LSA score between aspects, we use SEMILAR semantic similarity toolkit (Rus et al., 2013). SEMILAR provides a framework for systematic comparison of various semantic similarity methods. It uses LSA models, developed employing the whole of Wikipedia articles. The benefit of using this method is that we do not need any prior relationship knowledge between the aspects. The similarity score between the terms can be used to establish the relationship between aspects. This way, we can find more relationship between the aspects as compared to semantic knowledge base approach. The details of ontology tree creation using similarity score is given in Algorithm 3.

Algorithm 3 Semantic Similarity Ontology Tree (SemS)

```
Input:  $D$: Aspects dictionary $= <A_P, f>$, where
        $A_P$: Popular aspect set $= \{A_{P_1}, A_{P_2}, ... A_{P_n}\}$
        $f$: Frequency count of each candidate aspect $= \{f_1, f_2, ... f_n\}$
        $T_{LSA}$: Threshold LSA score
        $P$: Product domain
Output: $T_O$: Aspect ontology tree
Method:
1: Vertex set, $V \leftarrow \phi$, Edge set, $E \leftarrow \phi$, Queue, $Q \leftarrow \phi$
2: Graph, $G \leftarrow (V, E)$
3: $Q$.enqueue($P$)
4: visited[$P$]$\leftarrow$ true
5: while $Q \neq \Phi$ do
6:     $V' \leftarrow []$
7:     Concept $c = Q$.dequeue()
8:     function ExtractSimilarNodes($c, A_P$)
9:         for all $v_i \in A_P$ do
10:            if $LSA(c, v_i) > T_{LSA}$ then
11:                $V' \leftarrow append(v_i)$
12:            end if
13:        end for
14:        $V' \leftarrow sort(V')$
15:        return $V'$
16:    end function
17:    for all $v_i \in V'$ do
18:        if$v_i \notin$ visited then
19:            if $v_i \notin G$ then
20:                $V.\ add(v_i)$
21:                for all $v_j \in$ children($c$) do
22:                    if $LSA(c, v_i) > LSA(v_i, v_j)$ then
23:                        $E.\ AddEdgec, v_i$
24:                    else
25:                        $E.\ AddEdge(v_j, v_i)$
26:                    end if
27:                end for
28:            end if
29:            $Q$.enqueue($V_i$)
30:            visited[$v_i$]$\leftarrow$true
31:        end if
32:    end for
33: end while
34: $T_O, A_T \leftarrow MergeSynonyms(G)$
```

SemS uses semantic similarity to create aspect ontology. Similar to SemR, SemS also uses BFS exploration. The initial Steps 1–7 of SemS are similar to that of SemR. The main difference between the two is in the method used to find aspect relationship for ontology construction. SemS uses LSA score to create aspect ontology instead of using ConceptNet relations. We start the algorithm by computing LSA score (T_{LSA}) between candidate aspects (Steps 8–16). Aspects with LSA score above a threshold (T_{LSA}) are considered as closely related aspects. For a particular aspect, ExtractSimilarNodes (c, A_P) finds a list of related aspects (V'), whose LSA score is above T_{LSA}. This list (V') contains new children nodes of the tree. These nodes are added as children of the parent such that the children aspects are sorted based on dictionary order from left to right. Step 14 is used to maintain consistency while creating the aspect ontology. For adding an edge between parent and a new node, we consider all existing children nodes of the parent c and compare the LSA score. In Steps 21–26, LSA scores are compared to add an edge between the nodes. Step 22 checks if the new aspect v_i is more related to the parent aspect c compared to any of the children aspects. If so, an edge is added from c to v_i (Step 23). Otherwise, an edge is added from the more similar node v_j to v_i (Step 25). This ensures that the paths in the ontology tree are of comparable lengths. After adding a new node, it is enqueued and marked as visited for further processing (Steps 29–30).

We observe that in lower levels, the sub-aspects become less relevant to the product domain even though they are related to their respective parents. The aspects' relatedness tends to deviate away from the original product domain as we go down the ontology tree. For instance, consider the branch of the ontology *camera* → *flash* → *shot* → *noise*.

Here, the aspect *noise* deviates from the *camera* domain even though it is related to the aspect *shot*. This is due to topic drift as we go down the tree levels. To prevent topic drift from happening, we propose to increase T_{LSA} at every level down the tree so that the relationship constraint becomes more stringent. We formulate the new T_{LSA} as follows:

$$T_{LSA} = \alpha + \delta(l-1) \tag{1}$$

where, α is the initial value of similarity threshold used to find the first level nodes. l represents the level of the ontology tree. δ act as a damping factor for topic drift. Instead of using the same T_{LSA} at all levels, we increase its value by a factor δ at each level down the tree. This makes the relationship constraint become more stringent and ensures that only closely related aspects are preserved when we go down the tree. The values of α and δ are determined empirically. We tested Eq. (1) using different values of α ($\alpha = \{0.10, 0.20, 0.30, 0.40, 0.50\}$) and δ ($\delta = \{0.01, 0.03, 0.05, 0.07\}$)

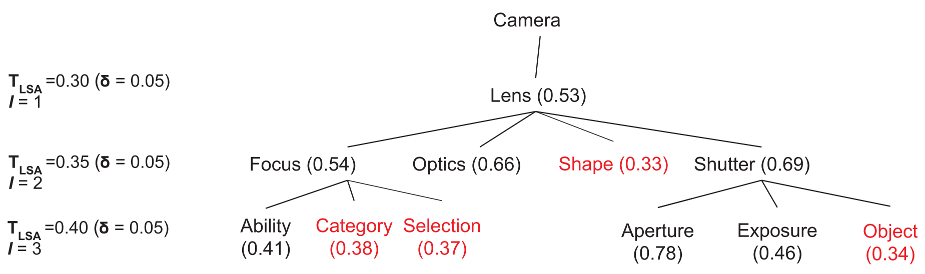


Fig. 5. Topic drift examples.

Table 4
Statistics of the product review dataset.

Product	#Sentences	#Aspects
Digital Camera	597	85
Mobile Phone	346	100
DVD Player	740	97
Jukebox	1716	162

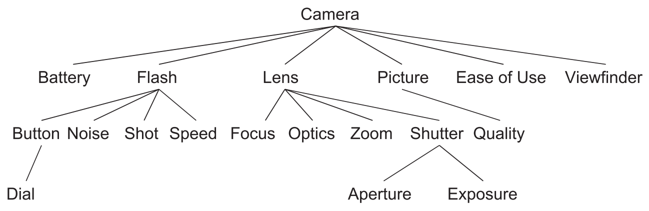


Fig. 6. Gold-standard aspect ontology tree for camera.

Table 5
Precision and recall of the proposed approaches w.r.t. gold-standard aspect ontology.

Algorithms	Manually Labeled Aspects (MLA)			Automatically Extracted Aspects (AEA)		
	Precision	Recall	F1	Precision	Recall	F1
SemR	66.66	73.68	69.99	56	73.68	63.63
SemS	93.75	78.94	85.71	78.94	78.94	78.94

Table 6
Similarity between created ontology trees and gold-standard ontology.

Ontology	#Aspects	Height	Edit Distance	#Leaf Nodes
Gold Standard	19	4	0	14
SemR on MLA	21	3	16	17
SemS on MLA	16	4	5	9
SemR on AEA	25	3	16	20
SemS on AEA	19	4	7	10

and found that $\alpha = 0.30$, $\delta = 0.05$ give the best result. Some examples of topic drift removed using modified T_{LSA} are given in Fig. 5.

As shown in Fig. 5, using Eq. (1) preserves closely related aspects and discards irrelevant aspects such as *category*, *selection*, *shape*, and *object*. We found this new definition of T_{LSA} to be quite effective in avoiding topic drift.

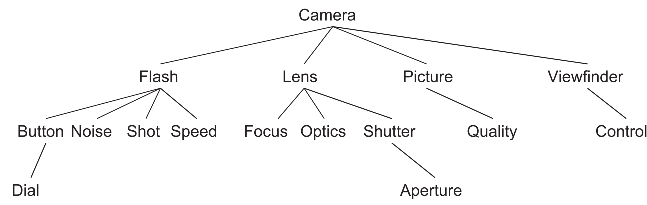


Fig. 8. Aspect ontology tree created using SemS approach.

4. Evaluation

In this section, we present the detailed experimental evaluation of the proposed approach.

4.1. Experimental Setup

We perform our experiments on the dataset created by Hu and Liu (2004a). This dataset is commonly used for the evaluation of different aspect extraction methods (Liu et al., 2013; Qiu et al., 2011). The dataset consists of customer reviews of four electronic products, namely digital camera, mobile phone, DVD player, and mp3 player. Table 4 presents a summary of the dataset used for evaluation.

Due to brevity of space, we present results for the camera product only. Similar results are achieved with other products. In the camera product review, there are 597 sentences and 285 manually annotated aspects.

In order to prepare a clean dataset, we perform two types of preprocessing, namely spelling correction and aspect grouping. Spelling error is the most common problem in user reviews. For example, a user may misspell words, such as *camera* as *canera*, *aperture* as *aperature*, etc. Due to such spelling errors, we get an ontology tree that has wrong aspect names. Thus our first pre-processing step is the spelling correction. To perform spelling correction, we use NLP based auto correction technique developed by Norvig, 2007. In the reviews, aspects may be mentioned in many synonymous forms. For example, *photo*, *picture*, *photograph*, and *image* are all synonymous aspects. In this case, we identify and merge all such synonyms using the WordNet synsets (Fellbaum, 1998). From the synonyms, we select the most frequently used aspect in the reviews, for the ontology tree creation. This step reduces the number of manually annotated aspects, and also gives the actual frequency count of aspects.

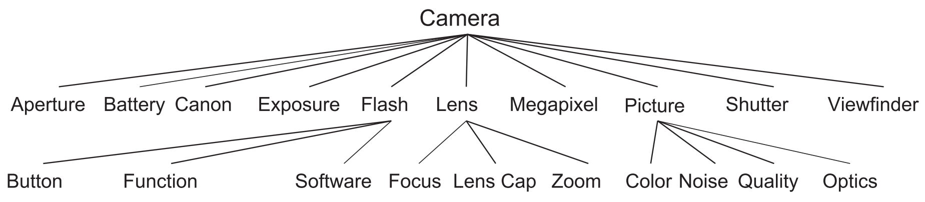


Fig. 7. Aspect ontology tree created using the proposed SemR approach.

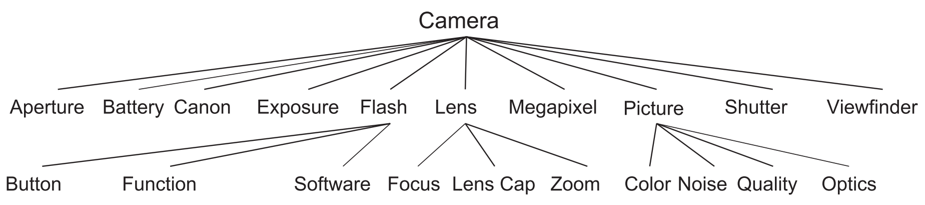


Fig. 9. Aspect ontology tree created using SemR approach on automatically extracted aspects.

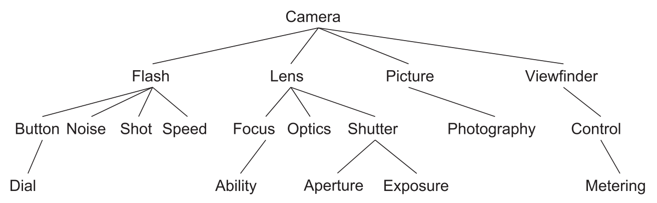


Fig. 10. Aspect ontology tree created using SemS approach on automatically extracted aspects.

4.2. Ontology evaluation metric

The primary contribution of this paper is the automatic construction of aspect ontology. The evaluation of an ontology tree is done by comparing it with a gold-standard ontology. In existing works, often the comparison is done manually by domain experts. We use two metrics, namely accuracy measure and edit distance, to evaluate the created ontologies. The accuracy metric involves standard precision and recall. We also use a tree edit distance based measure to compare the aspects arrangement in the constructed ontology with that of the gold-standard ontology.

4.2.1. Gold-standard ontology

To create the reference gold-standard ontology, we asked five Ph.D. students from Computer Science background to construct aspect ontology. All the five Ph.D. students work in the area of data mining, and are aware of sentiment analysis and opinion mining of product reviews. Specifically, three of them have in-depth knowledge about camera specifications and reviews, while the remaining two have a reasonable idea about *camera* domain. The students were given a list of all manually marked aspects extracted from the dataset and the reviews. Each student had to create an ontology utilizing this two information. Since our algorithm uses reviews to mine the relationship between aspects, we asked the students to utilize the aspect relationship information present in reviews. After each student came up with their own ontology, they resolved their ontology disagreement together. To maintain consistency, we required the sub-aspects to be placed alphabetically from left to right at each level of the ontology. The obtained gold-standard ontology is shown in Fig. 6.

Although there were 85 manually annotated aspects in the dataset, the volunteers agreed on using only 19 aspects for creating the ontology. This means that only 22% of the annotated aspects are popular and unique. The unused manually annotated aspects were either synonymous or very less related to the product. For example, *picture* and *image* are synonymous, and the aspects such as *feel*, *casing*, *strap*, *tiff format*, *import*, *depth*, *lever* have weak relationship with *camera*.

There were some challenges in creating the gold-standard ontology. When users provide reviews, they do not follow a standard naming convention for aspects, which leads to creation of synonymous aspects. Another challenge is to deal with redundant aspects. For example, the aspects *battery*, *battery life*, and *battery charging* are three different aspects related to one aspect *battery*. To overcome this problem, redundancy pruning (Hu and Liu, 2004a) is applied. In redundancy pruning, we find the support for each aspect, where the support of an aspect is the number of sentences that includes the particular aspect. The aspects having support lower than a minimum support value (T_{min}) are pruned out as redundant aspects.

4.2.2. Accuracy measures

The aspect accuracy of the ontologies generated by applying our proposed approaches is measured using precision, recall, and $F1$ score. Precision gives the percentage of correct aspects that are present in the generated aspect ontology tree while recall gives the percentage of correct aspects out of the total number of correct aspects present in the gold standard aspect ontology. Precision is defined as the ratio of correct aspects that are present out of the total number of aspects in the generated aspect ontology while recall is defined as the ratio of the number of correct aspects in the generated ontology and the number of gold standard aspects. Let T be the number of aspects present in the generated ontology tree and C be the total number of correct aspects present in the gold standard aspect ontology. Then, TP (true positive) is $|T \cap C|$, FP (false positive) is $|T\backslash C|$, FN (false negative) is $|C\backslash T|$. We can formulate precision and recall as follows:

$$Precision,\ P = \frac{TP}{TP+FP};\quad Recall,\ R = \frac{TP}{TP+FN};\quad F_1 = \frac{2*P*R}{P+R} \tag{2}$$

4.2.3. Tree edit distance

We evaluate the similarity of the created ontology compared to the gold-standard ontology by using tree edit distance as a metric (Pawlik and Augsten, 2016). Tree edit distance between two ordered labeled trees is defined as the minimal-cost sequence of node edit operations that transforms one tree into another using three node edit operations: *delete*, *insert* and *rename*. Each node edit operation has an associated cost. The cost of an edit sequence is the sum of the costs of its edit operations. If the created ontology tree and the gold-standard ontology tree have two aspects that are synonymous, then we rename the aspect in the created ontology using the gold-standard ontology aspect name. Thus to compute edit distance, we only count the delete and insert operations. Let d and i be the number of delete and insert operations respectively, and n be the number of possible ways to transform the created ontology to gold standard aspect ontology. Then, tree edit distance, E is defined as follows:

$$E = min\{C_j\};\quad C_j = \{d_j + i_j\},\quad j = 1,2,3,...,n \tag{3}$$

where C_j is the cost of j^{th} way to transform to the gold standard ontology.

4.3. Ontology evaluation

In this section, we compare the ontology created by our two proposed ontology creation algorithms, namely SemR and SemS, with the gold-standard ontology. SemR uses semantic relationships, whereas SemS uses semantic similarity to construct ontology. The process of constructing ontology is independent of how the aspects are extracted. While creating ontology, we only need to consider how to place the aspects in the ontology tree. In our dataset, the aspects are manually labeled. We apply our SemR and SemS algorithms on these aspects and compare them with the gold-standard ontology. Since it is difficult to get a manually labeled aspect review dataset, we use the algorithm proposed by Hu and Liu (2004b) to automatically extract the aspects. We then apply SemR and SemS algorithms on these automatically extracted aspects to create ontologies and compare them with the gold-standard ontology. In both cases, we find that SemS gives significantly better ontology compared to SemR. The result of the comparison is given in Tables 5 and 6.

4.3.1. Ontology using manually labeled aspects

As can be seen in Table 5, SemS has higher precision, recall, and F1 score than SemR. Compared to SemR, prcision, recall, and F1 score of SemS are higher by 27.09%, 5.26%, 15.72% respectively. SemR is based on manually assigned relationship between the concepts. Since it is hard to define all possible relations between concepts, SemR is limited by the number of relations present in the knowledge base. In SemR, some of the relations are less relevant to a given ontology concept. Since SemR does not consider intensity of association between concepts, the ontology generated by SemR contains some unrelated aspects having a weak association with *camera*, such as *function*, *software*, and *color*. On the contrary, SemS is based on the principle that related aspects tend to occur together in reviews. SemS uses LSA to assign a qualitative similarity score to aspect associations, and so is not limited by the relatedness knowledge base. As a result, it can achieve higher accuracy and efficiently prune out aspects that have weak association with the product. Figs. 7 and 8 show the ontology obtained using SemR and SemS approach with manually labeled aspects.

In terms of tree edit distance, from Table 6, we observe that SemS approach gives an edit distance of 5 and has 71% fewer edit operations compared to SemR approach. This shows that the resulting aspect ontology is able to capture most of the popular aspects present in the gold standard aspect ontology. SemR ontology has some aspects that are present at different locations compared to the gold standard aspect ontology. This is due to the limitations of its knowledge base. For example, *aperture* and *shutter* appear as children of *camera* using SemR approach, while they appear as children of *lens* in the gold standard aspect ontology. In addition to this, some more unrelated aspects are present such as *software*, *function*, and *color*. The lower precision of SemR also affects the tree edit distance as more delete operations are required to remove the unrelated aspects.

Another key observation is the height of the generated aspect ontology trees. From Table 6, we find that the height of the ontology trees obtained using SemR and SemS approaches are 3 and 4 respectively, which is very close to that of the gold standard ontology. The shallow height implies that the nodes of the aspect ontology are distributed equally at all levels. This results in a well-balanced structure of the aspect ontology. Increasing the ontology height accelerate the chance of topic drift and domain concept dilution (Mukherjee and Joshi, 2013). To give an illustration, the concepts *dial* and *phone* are related. Increasing the aspect ontology height by putting *phone* as a sub-aspect of *dial* in Fig. 8 would lead to topic drift as *phone* is not a sub-aspect of *camera*.

4.3.2. Ontology using automatically extracted aspects

Fig. 9 and 10 shows the aspect ontology generated by SemR and SemS respectively on the automatically extracted aspects (AEA). From Table 5, we observe that the automatic methods give comparable results as that of the ontology trees obtained using manually labeled aspects (MLA). The recall of the AEA methods are same as that of MLA methods for both SemR and SemS approaches. This is because once the aspects are extracted, our algorithms accurately capture associations between the aspects. However, there is a slight decrease in precision of AEA methods compared to MLA methods. The precision of SemR with AEA drops by 10%, while SemS with AEA drops by 15% when compared to the corresponding MLA methods. This slight decrease in precision is due to the inability of AEA methods to detect implicit aspects such as *ease-of-use* as well as due to the presence of aspects that are not related to the ontology domain. Some of the unrelated aspects include *ability*, *photography*, and *metering*. The presence of such unrelated aspects also result in a slight increase of tree edit distance for AEA methods.

5. Conclusion

In this paper, we introduced two novel approaches to create an aspect ontology tree by finding the relationship between aspects of a product. One can use this aspect ontology tree to explore reviews. We constructed the aspect ontology tree using semantic knowledge base and semantic similarity. Our results are quite promising, showing that our approach can create aspect ontology with high $F1$ score. We found that SemS approach gives higher $F1$ score compared to SemR approach, both for manually labeled aspects and automatically labeled aspects. By incorporating ontology trees, we can make review exploration simpler and provide insight about aspects relationship while summarizing product reviews. Our method is ideally suited for product review summarization. Our approach is also extendable to summarize blogs, debates and online forum posts.

References

Bollegala, D., Matsuo, Y., Ishizuka, M., 2007. Measuring semantic similarity between words using web search engines. In: Proceedings of the International Conference on World Wide Web. ACM, New York, USA, pp. 757–766.

Das, S.R., Chen, M.Y., 2007. Yahoo! for amazon: sentiment extraction from small talk on the web. Manage. Sci. 53 (9), 1375–1388.

Dave, K., Lawrence, S., Pennock, D.M., 2003. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In: Proceedings of the International Conference on World Wide Web. ACM, pp. 519–528.

Eirinaki, M., Pisal, S., Singh, J., 2012. Feature-based opinion mining and ranking. J. Comput. Syst. Sci. 78 (4), 1175–1184.

Fang, X., Zhan, J., 2015. Sentiment analysis using product review data. J. Big Data 1 (2), 1–14.

Faria, C., Serra, I., Girardi, R., 2014. A domain-independent process for automatic ontology population from text. Sci. Comput. Program. 95, 26–43.

Fellbaum, C., 1998. WordNet: An Electronic Lexical Database. MIT Press.

Hu, M., Liu, B., 2004a. Mining and summarizing customer reviews. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, New York, USA, pp. 168–177.

Hu, M., Liu, B., 2004b. Mining opinion features in customer reviews. In: Proceedings of the National Conference on Artificial Intellgience. AAAI Press, pp. 755–760.

Huang, S., Cheng, W., 2015. Discovering chinese sentence patterns for feature-based opinion summarization. Electron. Commer. Res. Appl. 14 (6), 582–591.

Kang, Y.-B., Haghighi, P.D., Burstein, F., 2014. Cfinder: an intelligent key concept finder from text for ontology development. Expert Syst. Appl. 41 (9), 4494–4504.

Kobayashi, N., Inui, K., Matsumoto, Y., Tateishi, K., Fukushima, T., 2004. Collecting evaluative expressions for opinion extraction. In: Proceeding of the International Conference on Natural Language Processing. Springer, pp. 596–605.

Kobayashi, N., Inui, K., Matsumoto, Y., 2007. Extracting aspect-evaluation and aspect-of relations in opinion mining. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. ACL, pp. 1065–1074.

Landauer, T.K., 2006. Latent Semantic Analysis. Wiley Online Library.

Lee, J., Goodwin, R., 2006. Ontology management for large-scale enterprise systems. Electron. Commer. Res. Appl. 5 (1), 2–15.

Lee, T., Lee, I.-H., Lee, S., Lee, S.-G., Kim, D., Chun, J., Lee, H., Shim, J., 2006. Building an operational product ontology system. Electron. Commer. Res. Appl. 5 (1), 16–28.

Liu, B., 2012. Sentiment analysis and opinion mining. Synthesis Lectures Human Lang. Technol. 5 (1), 1–167.

Liu, H., Singh, P., 2004. Conceptnet – a practical commonsense reasoning tool-kit. BT Technol. J. 22 (4), 211–226.

Liu, B., Hu, M., Cheng, J., 2005. Opinion observer: analyzing and comparing opinions on the web. In: Proceedings of the International Conference on World Wide Web. ACM, pp. 342–351.

Liu, K., Xu, H.L., Liu, Y., Zhao, J., 2013. Opinion target extraction using partially-supervised word alignment model. In: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence. AAAI, pp. 2134–2140.

Liu, Q., Liu, B., Zhang, Y., Kim, D.S., Gao, Z., 2016. Improving opinion aspect extraction using semantic similarity and aspect associations. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. AAAI, pp. 2986–2992.

Lu, Y., Zhai, C., Sundaresan, N., 2009. Rated aspect summarization of short comments. In: Proceedings of the International Conference on World Wide Web. ACM, New York, USA, pp. 131–140.

Malucelli, A., Palzer, D., Oliveira, E., 2006. Ontology-based services to help solving the heterogeneity problem in e-commerce negotiations. Electron. Commer. Res. Appl. 5 (1), 29–43.

Mukherjee, S., Joshi, S., 2013. Sentiment aggregation using conceptnet ontology. In: Proceedings of the Sixth International Joint Conference on Natural Language Processing. Asian Federation of Natural Language Processing, pp. 570–578.

Mukherjee, A., Liu, B., 2012. Aspect extraction through semi-supervised modeling. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1. ACL, Stroudsburg, USA, pp. 339–348.

Norvig, P., 2007. How to write a spelling corrector. De:http://norvig.com/spell-correct.html.

Pawlik, M., Augsten, N., 2016. Tree edit distance: robust and memory-efficient. Inf. Syst. 56 (C), 157–173.

Pontiki, M., Galanis, D., Papageorgiou, H., Androutsopoulos, I., Manandhar, S., Mohammad, A.-S., Al-Ayyoub, M., Zhao, Y., Qin, B., De Clercq, O., et al., 2016. Semeval-2016 task 5: aspect based sentiment analysis. In: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016). ACL, pp. 19–30.

Popescu, A.-M., Nguyen, B., Etzioni, O., 2005. Opine: extracting product features and opinions from reviews. In: Proceedings of the HLT/EMNLP on interactive demonstrations. Association for Computational Linguistics, Stroudsburg, USA, pp. 32–33.

Qiu, G., Liu, B., Bu, J., Chen, C., 2011. Opinion word expansion and target extraction through double propagation. Comput. Linguist. 37 (1), 9–27.

Rus, V., Lintean, M.C., Banjade, R., Niraula, N.B., Stefanescu, D., 2013. Semilar: the semantic similarity toolkit. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations. ACL, pp. 163–168.

Scaffidi, C., Bierhoff, K., Chang, E., Felker, M., Ng, H., Jin, C., 2007. Red opal: product-feature scoring from reviews. In: Proceedings of the 8th ACM conference on Electronic Commerce. ACM, New York, USA, pp. 182–191.

Shein, K.P.P., 2009. Ontology based combined approach for sentiment classification. In: Proceedings of the International Conference on Communications and Information Technology. WSEAS, Stevens Point, USA, pp. 112–115.

Somasundaran, S., Wiebe, J., 2009. Recognizing stances in online debates. In: Proceedings of ACL-IJCNLP '09. ACL, Stroudsburg, USA, pp. 226–234.

Turney, P.D., 2002. Thumbs up or thumbs down?: Semantic orientation applied to unsupervised classification of reviews. In: Proceedings of the Annual Meeting on Association for Computational Linguistics. ACL, Stroudsburg, USA, pp. 417–424.

Zhang, Y., Lin, Z., 2018. Predicting the helpfulness of online product reviews: a multilingual approach. Electron. Commer. Res. Appl. 27, 1–10.

Zhou, W., Duan, W., 2012. Online user reviews, product variety, and the long tail: an empirical investigation on online software downloads. Electron. Commer. Res. Appl. 11 (3), 275–289.

Zhuang, L., Jing, F., Zhu, X.-Y., 2006. Movie review mining and summarization. In: Proceedings of the 15th ACM International Conference on Information and Knowledge Management. ACM, New York, USA, pp. 43–50.