Feature based Sentiment Analysis using a Domain Ontology

Neha Yadav
Department of Computer Science and Engineering
Indian Institute of Technology (BHU)
Varanasi, India 221005
neharao.rao8@gmail.com

C Ravindranath Chowdary
Department of Computer Science and Engineering
Indian Institute of Technology (BHU)
Varanasi, India 221005
rchowdary.cse@iitbhu.ac.in

Abstract

With the increase of unstructured social media data, sentiment analysis can be applied to infer useful information to assist organizations and their customers. We propose a model for feature-based sentiment analysis using ontology to address queries like:*"which car is more comfortable?, which car has better performance and interior?"*. Feature based sentiment analysis is done using SentiWordNet with word sense disambiguation and an ontology that is developed by us. Data Tables are prepared from the RDF triples of parsed ontology and the sentiment ranks of car attributes. To relate the RDBM data to the built ontology of car, mapping axioms are proposed to connect them using OBDA model. Using SPARQL query, the results of the proposed model are compared with a dictionary-based method with respect to different car attributes. The performance of our model is better than dictionary based method.

1 Introduction

With the development of Web 2.0, the measure of individual feeling (reviews, ratings, recommendations, feedbacks, comments) in online social networking has been seriously studied by many groups. The challenge is to decipher enormous amount of data for analyzing sentiments, feelings or emotions from the web. There are two types of approaches for sentiment analysis (SA): corpus-based and dictionary-based. The classification of customers' reviews at either sentence or complete document level is not adequate for many applications as these do not recognize right conclusion or sentiment targets. In this work, we emphasize on word-phrase and word-level-based sentiment classification also called feature-based opinion mining which covers both entities and aspects. In our approach, there are two major tasks: feature extraction and feature sentiment classification to determine sentiments on targets, for example, "*The speed of bmw 3.0 is great but the navigation is not good.*". Here there are two attributes of car (*bmw3.0*); *speed* and *navigation*.

According to Liu (2012), there are four principle approaches to recognize every assessment expression and its objective from the opinion: *"extraction based on frequent nouns and noun phrases, extraction by exploiting opinion and target relations, extraction using supervised learning and topic modeling"*. We used the second approach using nltk[1] lexical resources. Feature-based sentiment analysis (FBSA) is done on car customers' reviews in which features are extracted using our ontology on car domain. According to Gruber (1993), an ontology is an *"explicit and formal specification of a conceptualization"*. We proposed FBSA on car reviews using SentiWordNet with word sense disambiguation (WSD) and an ontology associated with mappings.

2 Related Work

Guzman and Maalej, (2014) suggested an automated technique in three phases to analyze relevant features: 1) collocation finding algorithm to extract fine-grained (two keywords) features, 2) SentiStrength for SA on sentence level 3) Latent Dirichlet Allocation (LDA) topic modeling technique to group into high-level or coarse-grained features and used weighted average sentiment score to each topic. The coherence of ten most popular features in app reviews are measured on a five-point scale, which represents a logical and consistent level of shared theme. The authors compared the results with the manually created

[1]http://www.nltk.org/

truth sets by content analysis and reported 91% precision and 73% recall.

Virmani et al., (2014) proposed an algorithm for aspect-level sentiment analysis on reviews used in Letter of recommendation (LOR) system. The authors made an aspect-tree of different aspect levels and sentiment weightage is assigned to each branch. Their idea is based on Sentiment Ontology Tree (SOT) and hierarchical classification algorithm (Wei and Gulla, 2010). After extracting aspects, each aspect value is computed by multiplying score of branches from the aspect location to root while traversing the aspect tree. The sentiment values of aspects are calculated using a dictionary-based method.

Thakor and Sasi, (2015) proposed partially automated ontology-based SA method (OSAPS) on social media data. Their aim is to identify the problem area on the customers' feedback on delivery issues of postal services and to generate automated online reply for those issues. They build ontology model from extracted data and use it to determine issues from the negative sentiments with SentiStrength. Their process includes data cleaning, extract only combination of nouns and verbs tags for query building and retrieves information from SPARQL Query from ontology model. Their ontology model with SPARQL query extract combinations of various nouns and verbs from negative tweets.

Tran and Phan, (2015) suggested a model for construction of sentiment ontology based Vietnamese reviews. This model comprised three phases: conceptual relation identification to determine the relationship among newly extracted features, feature extraction using word relationship in dependency tree and sentiment polarity assignment using corpus. Extraction of new features and sentiments is done by double propagation algorithm which proposed rules of syntactic relationship of sentiment words with feature, feature with feature and among sentiment words. They proposed six conceptual rules based on Vietnamese grammar to identify the semantic relationship of sentiment words with features present in opinion and reported an accuracy of 58.15% in terms of F-measure.

Ankit Ramteke et al., (2013) proposed rule based and machine learning (ML) approach for detection of thwarting and sarcasm using ontology in camera domain. They find document polarity by determining weights for the nodes of camera domain ontology from sentiment lexicons and used polarity reversal in opinion on different parts of product. Their ML based approach (SVM) outperforms rule based approach.

The topic modeling technique used in Guzman and Maalej, (2014) does not involve detection of infrequent features, lexical SA on sarcasm level, negation and conditionals. By measuring coherency of extracted features or using aspect tree, more significant features in the opinion can be obtained. Building unsupervised sentiment lexicon with SentiWordNet resulted in better accuracy. The ontology provides hierarchical semantic data to identify and extract features in an efficient way. Ramanathan and Ramnath, (2010) got the best accuracy at 80% for software dataset using ontology-domain mapping sentences on the objects contained in the ontology. Recupero et al.,(2014) suggested SA scoring algorithm to build a framework called Sentilo for extraction of sentiment relations in a sentence. Their framework identified topics in different context and target holders which are stored in RDF (Resource Description Framework) representation using Linked data and an opinion ontology. Thakor and Sasi, (2015) approach to identify negative sentiments for query data isn't efficient and needs updating and extraction of more logical data to optimize SPARQL query. Tran and Phan, (2015) work did not identify features in phrases.

3 FBSA using Dictionaries

Customer car reviews are extracted from *edmund*[2] site.

3.1 Dictionary-based sentiment analysis

According to Liu (2012), a sentiment is defined as 5-tuple (quintuple) in a given document d as follows:

$$S = (e_i, a_{ij}, s_{ijkl}, h_k, t_l) \tag{1}$$

where e_i is the i^{th} target under consideration, a_{ij} is a j^{th} feature of the target i, h_k is the user who expresses the opinion, t_l is the time at which the review is conveyed, s_{ijkl} is the sentiment of j^{th} feature a_{ij} of i^{th} target e_i by h_k at time t_l.

For example, *"Last Sunday, David found out that the bmw 3.0 navigation is very poor"*. In this sentence, e_i = *bmw 3.0*, a_{ij} = *navigation*, h_k =

[2]http://www.edmunds.com/car-reviews/

David, t_l = *last Sunday*, s_{ijkl} = *poor*, which should be negative on sentiment analysis.

For dictionary-based SA, two dictionaries of text files containing positive and negative sentiment word list is taken from the Opinion Lexicon [3] (English) (Hu and Liu, 2004) for sentiment analysis.

3.2 Algorithm for FBSA using dictionaries

This is a naïve method for SA. $stSum$ is the overall sentiment score for the review and sc is the sentiment score of a feature. Table 1 and 2 give labels to the ranges of $stSum$, sc (second and third column respectively) of the four functions $rR1(stSum)$, $rR2(stSum)$, $rF1(sc)$, $rF2(sc)$. These ranges are decided on the basis of selection of better distribution of scores of reviews. The $rR1(stSum)$, $rR2(stSum)$ functions give labels to sentiment polarity ($stSum$) for each review and $rF1(sc)$, $rF2(sc)$ functions add labels to sentiment polarity (sc) for each key in $carD$.

labels	$rR1(stSum)$	$rF1(sc)$
VERY_BAD	≤ -5	≤ -2
BAD	(-5,-2]	(-2,-1]
NOT_GOOD	(-2,0)	(-1,0)
NEUTRAL	$\{0\}$	$\{0\}$
GOOD	(0,8]	(0,5]
VERY_GOOD	(8,14]	(5,7]
EXCELLENT	> 14	> 7

Table 1: sentiment labels of reviews and features for Algorithm 1

In Algorithm 1 and 3, $carD$ is a dictionary having car attributes or features as keys and their values are relevant sentiment words in the user review, sentiment score and labels (ranks) of the features. The words (attributes) in $carJ$ is a jargon or list of features extracted from the RDF triples obtained from built ontology of car ($Car\_Ontology$) as described in Section 5.

Generating $stList$ of sentence s_i : $stList$ is a list of tuples of form (word, part-of-speech tag, polarity, position) denoted as (w, pos, pol, p). First, sentence s_i is tokenized into words. Each word is represented as a tuple (w, pos, pol, p), where p is position of w in sentence s_i, pos are pos-tag using *nltk.pos_tag* and pol (default value is 0) is computed by checking, if w is in positive dictionary, pol is 1 and if w is in negative dictionary, pol is -1.

Breaking into segments: The sentence segments are formed by finding positions of the conjunction words ('but', 'and','or', ',' and 'because'). For e.g, sentence s_i: *"The mileage is not good but engine is smooth, sound system is good"*. The segments of this example are: *"The mileage is not good"*,*"engine is smooth"*, *"sound system is good"* which are formed by breaking sentence at positions of conjunction words. Within each segment S_{s_i}, each word in $carD$ (having car attributes as keys) is tagged with tuples of $stList$ of words of S_{s_i} except the tuples having words as keys.

Algorithm 1 FBSA using dictionaries

```
1: Input: reviews, $carJ$, sentiment dictionaries
2: Output : files of FB sentiment ranks of cars.
3: for each review R in review-list of f do:
4:     Break R into sentences;
5:     stSum ← 0;                ▷ total review polarity
6:     stList ← [];
7:     $carD \leftarrow \{\}$;
8:     make car attributes of $carJ$ as $carD$ keys;
9:     for each sentence $s_i$ in $R$ do:
10:        generate stList of $s_i$ as explained in
           Section 3.2;
11:        $stSum \leftarrow stSum + \sum pol$ of each $w$
           in $stList$;
12:        make segments of $s_i$ as explained in
           Section 3.2;
13:        for each segment $S_{s_i} \in s_i$ do:
14:            add tuples of $S_{s_i}$ to $carD$ keys as
               explained in Section 3.2;
15:    ▷ add ranks and score of each $carD$ key;
16:    for each key $k_j \in carD$ do:
17:        $sc = \sum pol$ of tuples of $carD[k_j]$;
18:        $carD[k_j].append(rF1(sc))$;
19:        ▷ ranks to sentiment of each review;
20:    rR1{stSum};
21:            ▷ overall polarity of each review;
22:    $stType \leftarrow$ "$NEUTRAL$";
23:    if $stSum > 0$ then:
24:        $stType \leftarrow$ "$POSITIVE$";
25:    if $stSum < 0$ then:
26:        $stType \leftarrow$ "$NEGATIVE$";
```

[3]https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html

4 FBSA using lexical resources

4.1 SentiWordNet

SentiWordNet 3.0[4] is an explicit lexical resource document for assignment of sentiment weightage for opinion mining. It contains English words, derived from WordNet[5] which carry semantic word relations, attached with a score. The POS in SentiWordNet contains sentiment subjectivity of adverb (r), noun (n), verb (v), adjective (a) synsets i.e. according to the meaning of each word used in a sentence. Its output contains five elements: pos tag, offset-id (identify synset), positive score, negative score and list of synsets. The sentiment score of each synset in SentiWordNet varies from 0 to 1.

4.2 Word Sense Disambiguation

Word Sense Disambiguation (WSD) (Navigli, 2009) recognizes which feeling of a word (which means) is utilized as a part of a sentence, when there are polysemy words or phrases. WSD needs dictionary for word reference to determine the possible meanings or glossary and a corpus database for disambiguated words identification. The WSD based on corpus method is not much practical because for word target labeling, a user has to refer to the word-phrase glossary repeatedly for the same word. The proposed method used fine-grained computational lexicon of WordNet (Fellbaum and Christiane, 1998) dictionary for WSD by finding semantic word relations to find the sentiment polarity furthermore.

4.3 Feature based Sentiment Analysis

4.3.1 Building Lexicon:

Following types of sentiment lexicons are created manually (Palanisamy et al., 2013) for FBSA:

1. *Regular Lexicons:*
   This sort of lexicon comprises of sentiment information having consistent semantics or connotation crosswise over various classes and the opinion words.
   (a) *Default Sentiment Words:* Common list of words (adjective, adverbs) having consistent sentiment semantic value (positive or negative weightage) across different fields.
   (b) *TNgList words:* These reverse the sentiment subjectivity ($\pm$) sign. For e.g., *"The mileage is not good"* has sentiment word *"good"* but it is negative subject.
   (c) *HdNg Words:* the words (*'absence', 'deals', 'needs', 'presence' etc.*), which remain undetected by common sentiment words, have hidden sentiment in statement. In the sentence, *"The car needs a better wheel"*, word *"needs"* have a negative sense.
   (d) *Conjunctive Words:* the words which connect sub-statements like commas, conjunctions like *'and'* etc. The sentence is segmented for feature extraction. For e.g., *"Speed is good but the mileage is bad"* contains two fragments *"Speed is good"* and *"mileage is bad"*.

2. *Feature-based Lexicon:*
   For an entity: cars, the entity names are car brands e.g. *bmw3.0* and aspects/attributes like *performance, mileage* etc. This lexicon type includes: entity or product list which identifies users' target entities, a list of car properties for features, entity-based word-list of sentiments related to subjectivity.

4.3.2 Sentiment Calculation:

We apply sentiment analysis based on sentence-level and entity-level. FBSA is done by feature extraction using ontology. The sentiment labels are also assigned based on the overall review and feature-based sentiment polarity value mentioned in *rR2(), rF2()* respectively. After pre-processing, the sentiment is calculated as given in Algorithm 3:

1. WSD on the tokenized word list is performed to identify correct word sense in a review.

2. Compute sentiment list ($stList$) having (w, pos, pol, p) tuples. $stList$ contains nouns, verbs, adjectives and adverbs.

3. To calculate polarity, we check (Algorithm 2) if the word is not present in SentiWordNet but present in $NgList$ or $HdNg$ list, then the word tuple is added to the $stList$ along with the corresponding sentiment polarity. For e.g., in review, *"The car needs a better wheel"*, word *"needs"* has negative aspect for *"speed"* feature.

[4]http://sentiwordnet.isti.cnr.it/
[5]http://wordnet.princeton.edu

labels	$rR2(stSum)$	$rF2(sc)$
VERY_BAD	≤ -2	≤ -1
BAD	(-2,-1]	(-1,-0.5]
NOT_GOOD	(-1,0)	(-0.5,0)
NEUTRAL	{0}	{0}
GOOD	(0,1]	(0,1]
VERY_GOOD	(1,2]	(1,1.5]
EXCELLENT	> 2	> 1.5

Table 2: sentiment labels of reviews and features for Algorithm 3

4. In Algorithm 2, if the word is in $TNgList$, then the polarity of all the words present within word length of 2 in both the directions (4 units proximity by checking distance) will be reversed.

5. The sentiment of an entity is given by sc which is computed by summing the sentiment scores i.e. pol's of the entity in the review. The overall sentiment of the review is given by $stSum$.

4.3.3 Feature Extraction:

1. Only those features that are present in both the reviews and $CarJ$ are the candidate for features in this phase.

2. These candidate features and their sentiment word list is appended to $CarD$ dictionary.

4.4 Algorithm for FBSA Using Lexical Resources

In Algorithm 2, *HdNgList* is the list of sentiment words (not present in SentiWordNet): 'needs', 'absence', 'deficient', 'lack', 'incomplete', 'partial', 'fragmental'. *TNgList* is the list of words to reverse polarity.

Data preprocessing of sentence s_i:

This aims at normalizing the text into an appropriate form for sentiments extraction. Data preprocessing includes:

- *Tokenizing*: We break each review R into sentences and each sentence s_i is tokenized into words with part-of-speech tagging (identifying n, v, a, r) by *nltk pos_tokenizer*.

- *Reduction to root words*: We reduce words to derived form or stem, which helps in building lexicons. For e.g., words: 'hates', 'hating' reduced to 'hate'. It is done by *nltk stemmer*.

- *Expansion of abbreviated words*: After removing stopwords, we expand abbreviated words in s_i manually. We identify words, having some repetition of letters, and convert them to proper form like "wooowwww" to "wow".

- *Slang words or spelling correction*: We correct spellings and convert slang words to standard English by replacing the words with their expansion or correct form.

- *Appending sentiments of emoticons*: We add sentiments of the emoticons by replacing the smileys with appropriate words : happy, very happy, winky, sad, crying, angry.

- *Word Sense Disambiguation*: We generate $wsdList$ which is a list of word-synsets (having correct word sense, i.e. pos tag, in the sentence) after applying WSD. In this step, we apply WSD using maximum path similarity method, which is based on the idea that path length of more similar word sense of the word w is lesser than other less similar words, i.e. computes the shortest word sense edge from all edges of w.

Generating $stList$ of sentence s_i:

- $stList$ is a list of tuples of form (w, pos, pol, p) as mentioned in Section 3.2. But pol value is computed using SentiWordNet method which is the difference of positive and negative SentiWordNet score considering WSD

- *capitalization weightage*: Add extra $\pm\alpha$ sentiment weightage for capital emotion words

- *exclamation mark weightage:* Add extra sentiment score $\pm\beta/(\#(\text{'!'}))$ to the words with attached exclamation mark

- *start sentence weightage:* if s_i is start sentence and has more positive or negative score, then add $\pm\gamma$ to $stSum$

- *last sentence weightage:* if s_i is last sentence and has more positive or negative score, then add $\pm\delta$ to $stSum$

The values of α, β, γ and δ are fixed empirically.

Algorithm 2 reverse polarity of negation words

```
1:  function RVP($stList$, $stNgList$)
2:      if $stList \neq \emptyset$ and $stNgList \neq \emptyset$ then:
3:          for each word $w_1 \in$ stList do:
4:              for each word $w_2 \in$ stNgList do:
5:                  if proximity distance $|w_1 - w_2| \leq 2$ then:      ▷ both side
6:                      Reverse polarity of $w_1$;
7:      else
8:          if $stNgList \neq \emptyset$ then:
9:              for each word $w_i \in stNgList$ do:
10:                 if $w_i \notin stList$ then:
11:                     $stList.append((w_i, pos_i, pol_i, p_i))$;
12:     $HdNg \leftarrow$ tuples of $stList$ having words from $HdNgList$;
13:     for each word $w_i \in HdNg$ do:
14:         if $w_i \notin$ SentiWordNet then:
15:             $stList.append((w_i, pos_i, pol_i, p_i))$;
16:     return $stList$;
```

5 Building ontology in car domain

The car ontology ($Car\_Ontology$) about car properties shown in Figure 1 is built using Protégé[6] (protege-5.0.0-beta-15).

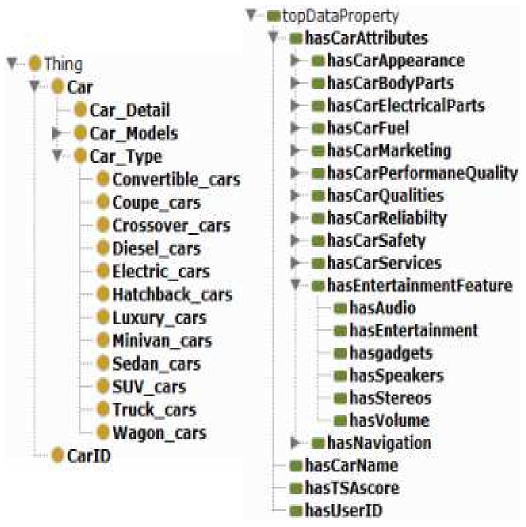

Figure 1: $Car\_Ontology$: classes, subclasses, data properties of Car attributes.

6 Data Parsing

The list of features of car is collected from parsing RDF syntax of built $Car\_Ontology$. RDF triples

[6]http://protege.stanford.edu/

Algorithm 3 FBSA using Lexical Resources

```
1:  Input: $HdNgList$, $TNgList$, reviews, $carJ$;
2:  Output: files of FB sentiment ranks of cars
3:  for each review R in review-list of f do:
4:      Break R into sentences;
5:      $stSum \leftarrow 0$;      ▷ total review polarity;
6:      $stList \leftarrow []$;
7:      $carD \leftarrow \{\}$;
8:      make car attributes of $carJ$ as $carD$ keys;
9:      for each sentence $s_i$ in R do:
10:         $wsdList \leftarrow []$;
11:         data preprocessing of $s_i$ and make $wsdList$ as explained in Section 4.4;
12:         generate $stList$ of $s_i$ as explained in Section 4.4;
13:                 ▷ reverse $pol$ of $TNgList$ words;
14:         $stNgList \leftarrow$ list of tuples of $stList$ having words present in $TNgList$;
15:         $stList \leftarrow$ RVP$(stList, stNgList)$;
16:         $stSum \leftarrow stSum + \sum pol$ of each $w$ in $stList$;
17:         make segments of $s_i$ as explained in Section 3.2;
18:         for each segment $S_{s_i} \in s_i$ do:
19:             add tuples of $S_{s_i}$ to $carD$ keys as explained in Section 3.2;
20:       ▷ add ranks and score of each $carD$ key;
21:     for each key $k_j \in carD$ do:
22:         $sc = \sum pol$ of tuples of $carD[k_j]$;
23:         $carD[k_j].append(rF1(sc))$;
24:             ▷ ranks to sentiment of each review;
25:     rR2{stSum};
26:             ▷ overall polarity of each review;
27:     $stType \leftarrow$ "$NEUTRAL$";
28:     if $stSum > 0$ then:
29:         $stType \leftarrow$ "$POSITIVE$";
30:     if $stSum < 0$ then:
31:         $stType \leftarrow$ "$NEGATIVE$";
```

(subject-predicate-object) between tags are stored by a graph (tree form) method which is used by previous Algorithms 1 and 3 for FBSA in collecting sentiment words related to defined category. We made a car database with thirteen data tables using the parsed and extracted data from Algorithm 1 and 3 which represents the customers' opinion about different car attributes.

7 Modeling in OBDA

7.1 Experimentation using ontop

An OBDA model (obdaModel, 2009) includes only single data source which is the database that constitutes information about the system and from which the queries would be executed in the framework. This model contains a set of mapping axioms in the ontology.

7.1.1 Mapping axioms

A mapping axiom in this model consists of the following three elements: mappingId (name: any string which recognizes the axiom), source (randomly selected SQL query processed over the car database) and target (RDF triple template that contains placeholders to refer the column names described in source query). For e.g., a valid ontop mapping in $Car\_Ontology$ is given below:

mappingId: Car_Attrbiutes
source : SELECT userid, appearance FROM carAppearanceTable
target : <*http://www.semanticweb.org/ont25#CarID_*$\{$*id*$\}$> *rdf:type :Car; :title* $\{$*appearance*$\}$

In target mapping, following placeholders are present in this model: a literal template, i.e., *appearance* describing one of the car attributes, and a URI template, i.e., *http://www.semanticweb.org/ont25#CarID_*$\{$*id*$\}$. The literal template placeholder is used to generate literal values, while URI template is used to create object URIs from the data in car database.

id	$name$
22	bmw3.0
23	acura_mdx

Table 3: query example

Meaning of a mapping axiom: The main objective of the mapping axioms in this model is to convert data in the specified data sources into a set of ABox assertions/RDF triples. Consider the following mapping about Appearance attribute:

mappingId: Appearance
source : SELECT id, name FROM carAppearanceTable
target: <*http://www.semanticweb.org/ont25#CarID_*$\{$*id*$\}$> *rdf:type :Car; :name* $\{$*name*$\}$ *:hasCarAttriubtes "appearance"*

The source query takes the mapping axioms and $Car\_Ontology$ as input and associate RDF triples set for each resultant target row. By substituting the placeholders in target ($\{$id$\}$ and $\{$name$\}$) with different values from the row, more triples can be generated. For e.g., if the answer to the source query is shown in Table 3, then the given mapping would generate the following six RDF triples:
<*http://www.semanticweb.org/ont25#CarID_*$\{$*22*$\}$>
rdf:type :Car;
:name "bmw3.0"
:hasCarAttriubtes "appearance".
<*http://www.semanticweb.org/ont25#CarID_*$\{$*23*$\}$>
rdf:type :Car;
:name "acura_mdx"
:hasCarAttriubtes "appearance".

Here, each target row generates three triples, since the target in the mapping had three triple templates. Here, by replacing $\{$*id*$\}$ in the URI template <*http://www.semanticweb.org/ont25#CarID_*$\{$*22*$\}$> by the value 22, we get an actual URI <*http://www.semanticweb.org/ont25#CarID_*$\{$*23*$\}$>, and by replacing $\{$*name*$\}$ in the literal template $\{$*name*$\}$*: string* by the value *bmw3.0*, we obtain an actual literal value *"bmw3.0"*.

8 Comparison of FBSA for cars

The numbers of sentiment ranks of car attributes are obtained by querying in SPARQL. Here, five-star rating is calculated based on weighted mean or average using the following formula:

$$\frac{\sum_{i=1}^{5} wt_i \times r_i}{R} \tag{2}$$

where wt_i is the weight of the i^{th}-star rating, r_i is the number of reviews at that weight wt_i, R is the total number of reviews. In Table 4, 5, 6, 7, 5-star rating is available at edmund site. The FBSA using dictionary-based (DbFBSA) method is compared with the FBSA using lexical resources and ontology (LoFBSA) on the error-rate basis. Error (%) is computed by the absolute difference of experimental five-star rating from the observed five-star rating available at edmund site.

As we observe here that error (%) of FBSA using lexical resources and ontology is less than the dictionary-based method.

CarAttribute	5-star rating	LoFBSA	DbFBSA
Comfort	4.73	8.03%	15.22%
Interior	4.73	7.82%	15.22%
Performance	4.72	9.95%	15.04%
Reliability	4.77	9.22%	16%

Table 4: Comparison table for Acura_mdx

CarAttribute	5-star rating	LoFBSA	DbFBSA
Comfort	4.68	8.33%	15%
Interior	4.75	9.05%	15.37%
Performance	4.76	10.08%	15.54%
Reliability	4.78	10.46%	16.32%

Table 5: Comparison table for Acura_rdx

CarAttribute	5-star rating	LoFBSA	DbFBSA
Comfort	4.67	9.2%	14%
Interior	4.67	8.13%	13.9%
Performance	4.60	8.2%	13%
Reliability	4.67	8.1%	14.34%

Table 6: Comparison table for Acura_tl

CarAttribute	5-star rating	LoFBSA	DbFBSA
Comfort	4.53	6.84%	12%
Interior	4.57	6.56%	12.03%
Performance	4.71	10.10%	15%
Reliability	4.37	5.49%	8%

Table 7: Comparison table for Bmw_3series

9 Conclusions

The main objective of this work is to provide a better recommendation to a user based on the reviews given by the customers. The proposed system performs FBSA on car reviews using SentiWordNet with WSD and a domain ontology. The contributions of this paper are 1) FBSA using lexical resources and ontology of car, 2) the mappings of car attributes for query analysis to determine the five-star ratings. Performance of LoFBSA is better than DbFBSA (without WSD) since LoFBSA considers data preprocessing with WSD, slangs, reversing polarity in case of negation and conjunctions for feature extraction.

10 Acknowledgement

This project was supported by DST-SERB No. YSS/2015/000906.

References

Fellbaum and Christiane. 1998. Wordnet: An electronic lexical database mit press. *Cambridge MA*.

Thomas R Gruber et al. 1993. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220.

Emitza Guzman and Walid Maalej. 2014. How do users like this feature? a fine grained sentiment analysis of app reviews. In *2014 IEEE 22nd international requirements engineering conference (RE)*, pages 153–162, Karlskrona, Sweden. IEEE.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA. ACM.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2):10:1–10:69, February.

obdaModel. 2009. ontop. `http://ontop.inf.unibz.it/`. [Online; accessed 23-June-2016].

Prabu Palanisamy, Vineet Yadav, and Harsha Elchuri. 2013. Serendio: Simple and practical lexicon based approach to sentiment analysis. In *proceedings of Second Joint Conference on Lexical and Computational Semantics*, pages 543–548, Atlanta, Georgia. Citeseer.

J Ramanathan and R Ramnath. 2010. Context-assisted sentiment analysis. In *The 25th Annual ACM Symposium on Applied Computing*, pages 404–413, Sierre, Switzerland.

Ankit Ramteke, Akshat Malu, Pushpak Bhattacharyya, and J Saketha Nath. 2013. Detecting turnarounds in sentiment analysis: Thwarting. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 860–865, Sofia, Bulgaria. Association for Computational Linguistics.

Diego Reforgiato Recupero, Sergio Consoli, Aldo Gangemi, Andrea Giovanni Nuzzolese, and Daria Spampinato. 2014. A semantic web based core engine to efficiently perform sentiment analysis. In *European Semantic Web Conference*, pages 245–248. Springer.

Pratik Thakor and Sreela Sasi. 2015. Ontology-based sentiment analysis process for social media content. *Procedia Computer Science*, 53:199–207.

Thien Khai Tran and Tuoi Thi Phan. 2015. Constructing sentiment ontology for vietnamese reviews. In *Proceedings of the 17th International Conference on Information Integration and Web-based Applications & Services*, iiWAS ’15, pages 36:1–36:5, Brussels, Belgium. ACM.

Deepali Virmani, Vikrant Malhotra, and Ridhi Tyagi. 2014. Aspect based sentiment analysis to extract meticulous opinion value. *arXiv preprint arXiv:1405.7519*.

Wei Wei and Jon Atle Gulla. 2010. Sentiment learning on product reviews via sentiment ontology tree. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 404–413, Uppsala, Sweden. Association for Computational Linguistics.