

# Exploration of Product Reviews

Konjengbam Anand

A Thesis Submitted to

Indian Institute of Technology Hyderabad

In Partial Fulfillment of the Requirements for

The Degree of Doctor of Philosophy



भारतीय प्रौद्योगिकी संस्थान हैदराबाद  
Indian Institute of Technology Hyderabad

Department of Computer Science and Engineering

September 2019

## Declaration

I declare that this written submission represents my ideas in my own words, and where ideas or words of others have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources that have thus not been properly cited, or from whom proper permission has not been taken when needed.

*K. Anand*

---

(Signature)

**Konjengbam Anand**

---

(Name)

**CS14RESCH11004**

---

(Roll No.)

## Approval Sheet

This thesis entitled **Exploration of Product Reviews** by Konjengbam Anand is approved for the degree of Doctor of Philosophy from IIT Hyderabad.

Mamata Jenamani

Prof. Mamata Jenamani, IIT Kharagpur

Examiner 1



Prof. Durga Toshniwal, IIT Roorkee

Examiner 2

Maunendra

Dr. Maunendra Sankar Desarkar, IIT Hyderabad

Internal Examiner



Dr. Manish Singh, IIT Hyderabad

Adviser/Guide

K. Badarinath

Dr. Karri Badarinath, IIT Hyderabad

Chairman

# Acknowledgements

At the outset, I would like to express my heartfelt gratitude to my thesis advisor Dr. Manish Singh, whose constant guidance, support and encouragement proved to be pivotal during my Ph.D. years here. His invaluable insights and knowledge have contributed immensely to my research work and will help me further in my career ahead. I am very thankful for his time and patience, and for his mentorship, which helped make my Ph.D. journey a smooth and fruitful one. I could not have wished for a better advisor for my doctoral study.

I would also like to extend my gratitude to my doctoral committee members, Dr. Vineeth N Balasubramanian, Dr. Suryakumar S. and Dr. Maunendra Sankar Desarkar, for their thoughtful comments and encouragement. Their advice motivated me in broadening my research analysis from multiple aspects.

I would like to thank Dr. Kotaro Kataoka for his mentorship and support in carrying out research here at IITH. I am also grateful to other professors at IITH for their valuable support and wisdom: Prof. U.B. Desai, Prof. C. Krishna Mohan, Dr. Bheemarjuna Reddy Tamma, and Dr. M. V. Panduranga Rao.

I would like to express my heartfelt gratitude to Nagendra Kumar, who provided technical support throughout my doctoral study and was also a reliable friend. I am also thankful to Dileep Reddy for helping me to set up and maintain our CSE IITH website.

This journey would have been difficult without the love and care of my friends. I would like to thank my labmates for being very supportive and caring: Utpal Bora, Sailaja Rajanala, Dinesh Singh, Samujjwal Ghosh, Nazil Perveen, Sakshi Varshney, Shiraj Arora, and Pragati Srivastava, among others. I am also thankful to my friends at IITH for their care and emotional support: Amol Bendkule, Dilin Sharaf, S Sandeep Kumar, Khunjamayum Reshikanta, Jayaprakash Mishra, Praveen Jangam, Deep-sikha Changmai, Monika Pebam, Musukha Brahma, Ankana Sen, Anindita Laha,

Neethi Alexander, Nilanjana Ghosal, and countless others. I would also like to thank my friends who have been there to support me through the thick and thin: Hatamori Kazuki, Tadahiro Inoue, Muto Motoki, Nameirakpam Burnishwar, Laitonjam Vipin, Sonia Longjam, Mitsuki Kagiya, Misa Shindo, and Ayuka Maeda.

A special note of thanks to my family. I would like to offer my deepest gratitude to my parents, my brother and my relatives for having faith in me during my work. I would like to dedicate this success to my late grandmother, who passed away last year, before the completion of my degree.

Last but not least, I would like to thank the Divine Force that led me to the path of pursuing my Ph.D.

# Dedication

*To my Parents, Teachers, Family, Friends and Well-wishers.*

# Abstract

E-commerce is a popular platform for trade of products and services through the Internet. Product reviews play a vital role in e-commerce by influencing the purchasing decision of customers. As many products have thousands of reviews, it is difficult to explore and extract useful information from them. We need review exploration systems to help users quickly explore and comprehend huge volumes of reviews. The current review exploration systems lack effective means of finding relevant review and summarizing them. In this thesis, we present three limitations of existing review exploration systems and propose solutions to address each of the three limitations.

Existing aspect based review summarization systems do not show the semantic relations that exist between aspects, which is required for proper exploration of reviews. We address this limitation by showing users an aspect ontology tree, which is created in an unsupervised manner from reviews, to show the relationship between aspects and sub-aspects. We then allow users to navigate reviews according to the aspects of the ontology.

Many of the existing review exploration systems summarize reviews by giving an average star rating of the reviews, or by finding the number of positive and negative reviews for each of the aspects. Such summarization methods do not show the actual opinion words of users, which is crucial to understand what other users like or don't like. We address the problem of summarizing reviews by creating informative and readable tags. We present a novel unsupervised method to generate the top-k opinionated tags. We also address the problem of tag generation for cold products, which have only a limited number of reviews and that too, with very limited content.

Finally, we study the problem of stance detection in comparative reviews using word-embeddings. Online debate sites are popular platforms for users to express and form opinions. Comparative reviews are very popular as they give a comparison of different aspects of two competing products. Standard aspect-based summarization

approaches cannot be used for comparative reviews as we need to figure out the target preference of each of the aspects, which is often not explicitly available in the review text. We propose an unsupervised approach to summarize comparative reviews by detecting the stance of users from comparative reviews.



# Contents

Declaration . . . . .	ii
Approval Sheet . . . . .	iii
Acknowledgements . . . . .	iv
Abstract . . . . .	vii
List of Abbreviations . . . . .	xvii
List of Symbols . . . . .	1
<b>1 Introduction</b>	<b>1</b>
1.1 Big Data in E-commerce . . . . .	2
1.2 Review Exploration . . . . .	3
1.3 Challenges in Review Exploration . . . . .	6
1.3.1 Lack of Semantic Relationship between Aspects . . . . .	6
1.3.2 Generating Meaningful Summary . . . . .	6
1.3.3 Mining Comparative Reviews . . . . .	6
1.4 Approaches . . . . .	7
1.4.1 Exploration of Reviews using Aspect Ontology . . . . .	7
1.4.2 Exploration of Reviews using Opinionated Tags . . . . .	8
1.4.3 Unsupervised Stance Detection in Comparative Reviews . . . . .	10
1.5 Thesis Contributions . . . . .	10
1.6 Research Overview . . . . .	11
1.7 Organization of the Thesis . . . . .	12

<b>2</b>	<b>Existing Systems and Background</b>	<b>14</b>
2.1	Data Exploration . . . . .	14
2.2	Preliminaries . . . . .	17
2.2.1	Review Types . . . . .	17
2.2.2	Pre-processing . . . . .	18
2.2.3	Part-Of-Speech (POS) Tagging . . . . .	20
<b>3</b>	<b>Related Work</b>	<b>21</b>
3.1	Sentiment Analysis . . . . .	21
3.2	Opinion Words in Sentiment Analysis . . . . .	22
3.3	Sentiment Lexicon in Sentiment Analysis . . . . .	23
3.4	Aspect-based Sentiment Analysis . . . . .	23
3.5	Aspect Extraction . . . . .	24
3.5.1	NLP based Aspect Extraction . . . . .	24
3.5.2	Frequent Term-based Aspect Extraction . . . . .	25
3.5.3	Limitation of Aspect-based Summarization . . . . .	25
3.6	Review visualization . . . . .	26
3.7	Tag Generation . . . . .	28
3.7.1	Supervised Tag Recommendation . . . . .	28
3.7.2	Unsupervised Tag Generation . . . . .	30
3.8	Mining Comparative Reviews . . . . .	31
<b>4</b>	<b>Exploration of Reviews using Aspect Ontology</b>	<b>33</b>
4.1	Introduction . . . . .	33
4.2	Proposed Approach . . . . .	35
4.2.1	Candidate Aspect Set Generation . . . . .	38
4.2.2	Schematic Interface . . . . .	39
4.2.3	Aspect Ontology Tree Generation . . . . .	40

4.3	Evaluation . . . . .	53
4.3.1	Experimental Setup . . . . .	53
4.3.2	Ontology Evaluation Metric . . . . .	54
4.3.3	Ontology Evaluation . . . . .	57
4.4	Conclusion . . . . .	62
<b>5</b>	<b>Exploration of Reviews using Opinionated Tags</b>	<b>63</b>
5.1	Introduction . . . . .	63
5.2	Tagging Product Reviews . . . . .	68
5.2.1	Problem Definition . . . . .	68
5.2.2	System Architecture . . . . .	70
5.2.3	Generating Candidate Tags . . . . .	71
5.2.4	Ranking Candidate Tags . . . . .	74
5.2.5	Refining Top-K Tags using Syntactic Rules . . . . .	77
5.3	Evaluation . . . . .	80
5.3.1	Experimental Setup . . . . .	81
5.3.2	Evaluation Metric . . . . .	84
5.3.3	Results . . . . .	85
5.4	Conclusion . . . . .	91
<b>6</b>	<b>Unsupervised Stance Detection in Comparative Reviews</b>	<b>92</b>
6.1	Introduction . . . . .	92
6.2	Proposed Approach . . . . .	96
6.2.1	Data Cleansing . . . . .	98
6.2.2	Opinion Word Extraction . . . . .	98
6.2.3	Aspect-Opinion Pairing . . . . .	99
6.2.4	Aspect Pruning . . . . .	99
6.2.5	Aspect-Polarity Mapping . . . . .	105

6.2.6	Aspect-Target Mapping . . . . .	106
6.2.7	Detect Stance . . . . .	108
6.3	Evaluation . . . . .	110
6.3.1	Dataset . . . . .	111
6.3.2	Stance Prediction Evaluation . . . . .	111
6.3.3	Results . . . . .	113
6.3.4	Aspect Pruning Evaluation . . . . .	116
6.4	Conclusion . . . . .	118
<b>7</b>	<b>Conclusion</b>	<b>119</b>
7.1	Summary of Contribution . . . . .	119
7.1.1	Review Exploration using Aspect Ontology . . . . .	120
7.1.2	Review Exploration using Opinionated Tags . . . . .	120
7.1.3	Stance Detection in Comparative Reviews . . . . .	121
7.2	Future Work . . . . .	122
	<b>References</b>	<b>124</b>

# List of Figures

1.1	Some popular e-commerce sites. . . . .	2
1.2	Big Data in Amazon.com. . . . .	3
1.3	Aspect-based exploration in Google Products. . . . .	4
1.4	Aspect-based exploration in Bing Shopping. . . . .	5
1.5	Aspect arrangement in the form of an ontology. . . . .	8
1.6	An example of a comparative review: Windows vs Mac. . . . .	10
1.7	Research Overview. . . . .	12
2.1	Category navigation in Amazon.com. . . . .	15
2.2	Faceted navigation in Amazon.com. . . . .	16
2.3	Examples of review summarization systems. . . . .	16
2.4	An example of a single entity review. . . . .	18
2.5	An example of a POS tagged sentence. . . . .	20
4.1	Screenshot of the ontology-based review exploration system. . . . .	34
4.2	System architecture of the aspect-based review exploration system. . . . .	36
4.3	A snippet from ConceptNet’s semantic network, showing various concepts related to the entity <i>camera</i> . . . . .	41
4.4	An instance of topic drift. . . . .	42
4.5	Using relation class to resolve topic drift. . . . .	43
4.6	Computing LSA score between words. . . . .	48

4.7	Number of topic drift observed by varying the values of $\alpha$ and $\delta$ . . . .	51
4.8	Number of nodes in the ontology with varying values $\alpha$ and $\delta$ . . . . .	52
4.9	Topic drift examples. . . . .	52
4.10	Gold-standard aspect ontology tree for camera. . . . .	55
4.11	Aspect ontology tree created using the SemR approach. . . . .	59
4.12	Aspect ontology tree created using the SemS approach. . . . .	59
4.13	Aspect ontology tree created using SemR approach on automatically extracted aspects. . . . .	61
4.14	Aspect ontology tree created using SemS approach on automatically extracted aspects. . . . .	61
5.1	Some popular platforms that use tags for information identification and categorization. . . . .	64
5.2	Amazon generated tags for a camera from reviews (accessed on 25.12.2018). . . . .	65
5.3	System Architecture. . . . .	70
5.4	Evaluation of top-k tags for popular products using the metric NDCG@k. . . . .	86
5.5	Evaluation of top-k tags for cold products using the metric NDCG@k. . . . .	88
6.1	Workflow of Debate Stance Classification using Word Embeddings. . . . .	96
6.2	Precision comparison for various debate classification methods. . . . .	114
6.3	Recall comparison for various debate classification methods. . . . .	114
6.4	F1 score comparison for various debate classification methods. . . . .	115

# List of Tables

1.1	Comparisons of tags generated from Amazon.com and our proposed approach. . . . .	9
4.1	Notations used. . . . .	37
4.2	ConceptNet relations used for building the ontology tree along with their relational class and priority order. ‘>’ shows inner class priority order. . . . .	44
4.3	Edge types and relation list. . . . .	44
4.4	Values of threshold $T_{LSA}$ for each tree level with varying values of threshold ( $\alpha$ ) and damping factor ( $\delta$ ). . . . .	51
4.5	Statistics of the product review dataset. . . . .	53
4.6	Precision and recall of the proposed approaches w.r.t. gold-standard aspect ontology. . . . .	58
4.7	Similarity between created ontology trees and gold-standard ontology. . . . .	58
5.1	POS rules used to enhance the meaning of tags. . . . .	78
5.2	Dataset Statistics. . . . .	82
5.3	Evaluation of tag refinement using the precision of top 50 tags. . . . .	90
6.1	Classes of irrelevant aspects obtained after applying the syntactic rules. . . . .	103
6.2	Aspect pruning accuracy across all four debate datasets. . . . .	105
6.3	Aspect-Target (AT) preference dictionary examples . . . . .	107

6.4	ILP implementation example . . . . .	110
6.5	Details of debate post dataset. . . . .	111
6.6	Precision, Recall, and $F1$ measure of various debate classification meth- ods. . . . .	113
6.7	Aspect pruning accuracy across all four debate datasets. . . . .	117
6.8	Word embedding similarity score of aspects. . . . .	118



## List of Abbreviations

$A_C$	Candidate Aspects
$A_P$	Popular Aspects
$A_T$	Subset of $A_P$ closely related to the product
ARExplorer	Aspect ontology based Review Explorer
BDR	Bidirectional Relation
BUR	Bottom Up Relation
CQA	Community Question Answering
DT	Determinant
FCA	Formal Concept Analysis
$F_R$	Functional Relation
$H_R$	Hierarchical Relation
IE	Internet Explorer
JJ	Adjective
$K_S$	Semantic knowledge base obtained using LSA
LDA	Latent Dirichlet Allocation
LSA	Latent Semantic Analysis
NDCG	Normalized Discounted Cumulative Gain
NLP	Natural Language Programming
NLTK	Natural Language Toolkit
NNP	Noun Phrase
NN	Noun
OS	Operating System
OWL	Web Ontology Language
PMI	Pointwise Mutual Information
POS	Part-of-Speech
R	Relation list obtained from ConceptNet

RST	Rhetorical Structure Theory
RB	Adverb
$S_R$	Synonym Relation
TBR	Top-to-bottom Relation
TF	Term Frequency
TNG	Topical n-gram
TPR	Tagging Product Reviews
TUR	Tag Usefulness Ranking
$T_O$	Aspect ontology tree
VBZ	Verb

# Chapter 1

## Introduction

The rapid expansion of the Internet and internet devices have transformed business, markets and shopping experience. It has paved the way for e-commerce by facilitating global trade of products and services. E-commerce includes all businesses that empower sales and purchase of products or services through the Internet. It provides a quick, convenient and low-cost marketplace, where customers can buy products and services just by pressing a few clicks instead of physically going to the market. Advancements in technologies such as online shopping websites, social network services, online banking, shopping cart, product tracking, etc., have aided in the success of e-commerce.

Over the past decade, e-commerce has grown at an average pace of 19% per year. E-commerce accounted for 11.9% of global retail sales in 2018 [145]. Figure 1.1 shows some popular e-commerce sites such as Amazon.com, Alibaba.com, Ebay.com, Flipkart.com, Tripadvisor.com, Netflix.com, etc., each one having millions of users transacting on a daily basis.



Figure 1.1: Some popular e-commerce sites.

## 1.1 Big Data in E-commerce

Due to the massive volume of information available in e-commerce sites, it is difficult for users to explore and find the right product. Figure 1.2 shows the big data associated with Amazon.com, one of the most popular e-commerce sites. It has millions of users, sellers, products and accounts for over \$232 billions worth of net sales. As there are millions of products, e-commerce sites use product navigation systems to help users find the item of their choice. Product navigation has become quite easy [138] due to various search interfaces, such as keyword search, auto-completion, faceted navigation and navigation by category hierarchy.

Considering the wide variety of available products and sellers, users take help of customer reviews to make an informed decision. Customer reviews generally consist of several lines of text and a numerical rating. Opinion rich reviews provide a subjective evaluation of the product by acting as word-of-mouth marketing on the web. Reviews play a vital role in influencing the purchasing decision of customers.

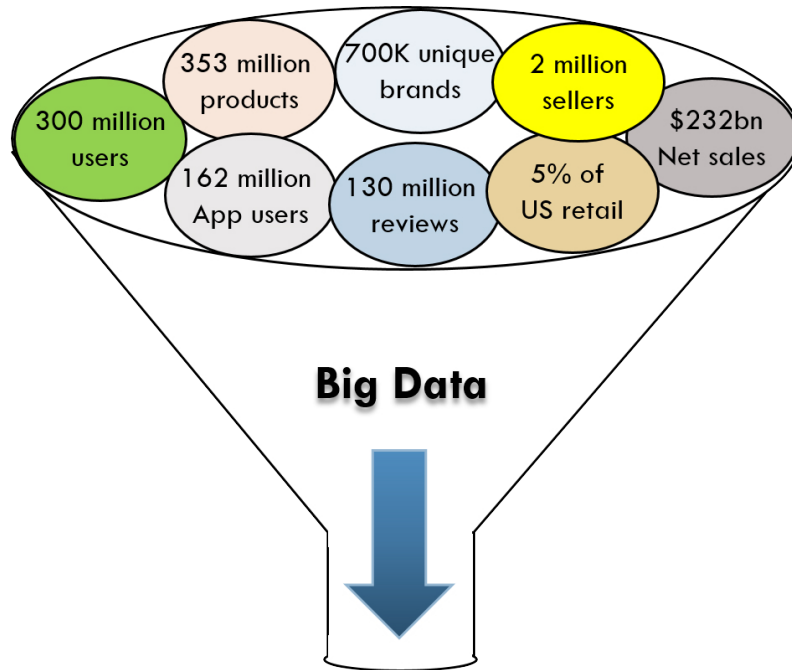


Figure 1.2: Big Data in Amazon.com.

According to a study conducted by Forbes in 2016, over 88% of online customers consider product review contents as significantly paramount [41]. Customer reviews also provide market intelligence to manufacturers. It acts as valuable feedback to manufacturers about their products and tells which components they need to pay attention to.

## 1.2 Review Exploration

Although product exploratory systems are well developed, little work has been done on review exploration. E-commerce sites provide basic review navigation mechanisms for exploring reviews [117, 179]. For example, they may allow users to sort reviews based on rating, social interaction such as most helpful votes, or time such as most recent reviews, etc. Since many products have thousands of reviews, it is hard to explore all of them. We thus need review exploration systems that can summarize enormous unstructured reviews in a succinct format and ease the process of reading

reviews.



Figure 1.3: Aspect-based exploration in Google Products.

Recent works have been carried out on aspect-based review summarization [125] where aspects and polarity information are extracted from reviews. Aspects are primarily the product features on which users express their opinions. Figures 1.3 and 1.4 show the aspect-based review summarization systems developed by Google and Microsoft Bing respectively. For a given product, these systems show some prominent aspects and the number of positive and negative reviews associated with each aspect. A few sample sentences from reviews may also be shown for each aspect. Although aspect-based review exploration systems exhibit product aspects and overall polarity, they do not present the semantic relationship between the aspects.

The image shows a Bing Shopping search result for an HP LaserJet 1020 printer. On the left, a sidebar titled 'POPULAR FEATURES' lists various attributes with progress bars: Affordability, Speed (highlighted in blue), Print Quality, Reliability, Ease Of Use, Brand, Installation, Size, and Compatibility. The main content area shows the product title 'HP LaserJet 1020 - printer - B/W - laser, 15ppm, USB', a price of '\$179 (2 stores)', a 'Bing cashback - 3%' offer, and a 4.5-star rating from 177 user reviews. Below this, there are tabs for 'user reviews', 'product details', 'expert reviews', and 'compare prices'. The 'user reviews' tab is active, showing a 'speed' filter with a 96% green bar and several positive comments from users like 'Love Reading' and 'Arthur L. Taylor'.

Figure 1.4: Aspect-based exploration in Bing Shopping.

## 1.3 Challenges in Review Exploration

### 1.3.1 Lack of Semantic Relationship between Aspects

Existing systems [21, 34] assume products to have a flat entity-aspect relationship. They do not consider the relationship between aspects and sub-aspects while creating aspect-based review exploration system.

Users usually have diverse aspect preferences and each aspect in turn may have multiple related aspects. For instance, a user who wants to buy a camera may be interested in good lens quality while another user may be interested in a low-priced camera. The first user would be interested in reviews containing the aspect *lens* while the second user would like to read reviews related to *price*. Also, it would be helpful for the first user to read reviews of other aspects related to *lens*, such as *focus*, *zoom*, *shutter* and *aperture*. However, users may be unaware about these related aspects and further, existing systems do not show such relationship between aspects.

### 1.3.2 Generating Meaningful Summary

Existing review summarization systems focus on extracting aspects and sentiments from reviews. However, such systems fail to capture the actual opinion left by users towards various aspects. Standard topic labeling algorithms [14, 161] do not perform well in summarizing product reviews. Although they perform well in content identification and categorization, they are not appropriate for summarizing reviews as many products have few reviews, many of which are short. Tags obtained using topic labeling algorithms are not easily understandable.

### 1.3.3 Mining Comparative Reviews

Many of the top helpful reviews are comparative in nature [2, 143]. Users express comparison between two opposing targets and opinions are expressed on either of the



targets. In such settings, it is hard to relate opinions and targets. Most of the existing works [125, 160] focus on mining simple reviews that describe a single product or a target entity. They do not work well for comparative reviews due to the presence of multiple targets.

## 1.4 Approaches

This thesis focuses on building user-friendly review exploration systems. In particular, three types of review exploration are discussed, each addressing the challenges in the preceding section.

### 1.4.1 Exploration of Reviews using Aspect Ontology

Opinion mining is the process of extracting user opinions from textual data and determining the polarity of extracted opinions. Existing works on opinion mining focus on sentiment classification [86], which could either focus on an entire review, on each sentence, or each aspect mentioned in the review. Part-of-speech (POS) tags with other NLP grammar rules [117, 160] are used to extract aspects and opinion words.

A limitation of existing approaches is that the relationship between different aspects remains neglected, such as synonym aspects, hierarchical relationships between aspects, etc. Instead of the flat aspect structure assumed by extant approaches, it would be better to present aspects in the form of an ontology. Ontology represents the hierarchical organization of concepts.

Figure 1.5 shows an illustration of how aspects from reviews can be made understandable by representing them in the form of an ontology. We propose to build aspect ontology based on the semantic relationship between the aspects. This would help to comprehend the relationship between related aspects and sub-aspects. It

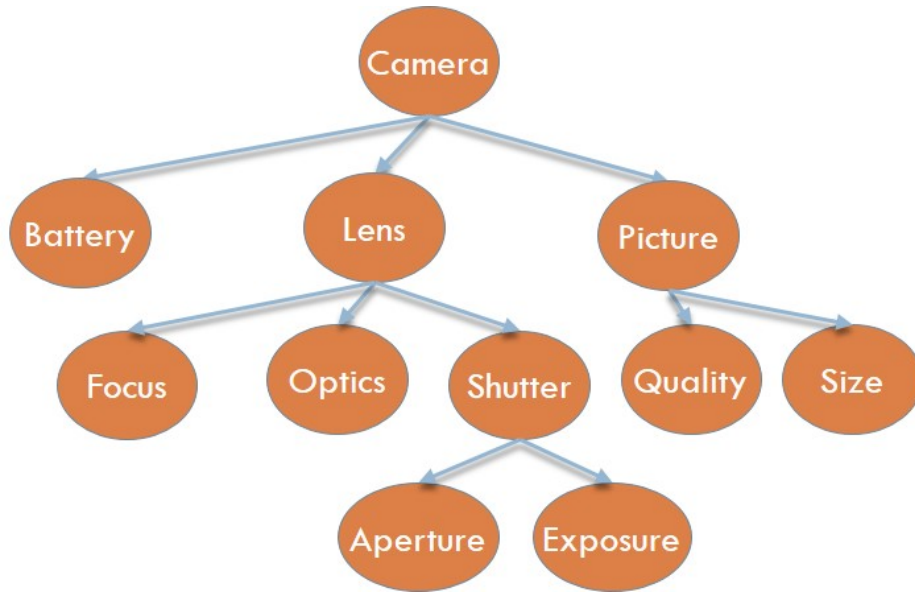


Figure 1.5: Aspect arrangement in the form of an ontology.

would also narrow down review search for desired aspects and users can find specific reviews based on a selected aspect. Such representation would also help users gain a better understanding of aspects and sub-aspects of a product.

In this thesis, we propose a novel unsupervised approach to create aspect ontology from reviews. We then use this ontology to create an exploratory search interface where users can access reviews using the product aspects. Users can view reviews that talk about an aspect, including its sub-aspects by clicking on any aspect shown in the Aspect Exploration panel.

### 1.4.2 Exploration of Reviews using Opinionated Tags

Current review summarization methods, such as star rating, sentiment prediction, and aspect summary give a quantitative evaluation of the product, but do not show the fine-grained relationships between aspects and relevant opinion words. Aspects and opinion words are the key ingredients of a review content.

In this thesis, we propose to generate opinionated tags to summarize product reviews. User-created tags are popularly used for organizing and summarizing user-

generated contents due to their simplicity in indexing and ease of user participation. Tags are used in social networking sites, such as Facebook [33], Twitter [153], YouTube [174], etc., for identifying the content topic, summarizing content, increasing visibility [23], and connecting people with a common interest. However, users do not provide tags when they write reviews.

Tags generated by existing topic labeling approaches [14,161] are not appropriate for summarizing reviews as many products have few reviews, a number of which are short. Tags generated using topic labeling algorithms are not easily understandable. The generated tags do not show aspect and opinion relationship. We extend conventional topic labeling approaches to generate opinionated tags from reviews and use the tags to summarize customer reviews.

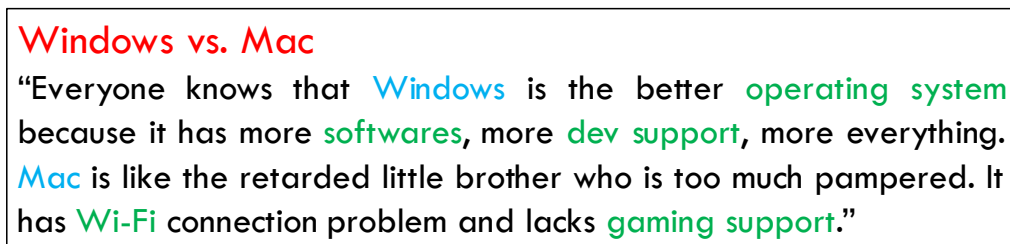
<b>Existing tags from Amazon.com</b>	<b>Proposed approach</b>
Lens	Cheap quality lens
Photo	Great photo
Camera	Small camera
Slow	Slow focus
Battery life	Long battery life
Level DSLR	Entry level DSLR
WiFi	No WiFi

Table 1.1: Comparisons of tags generated from Amazon.com and our proposed approach.

Table 1.1 shows the comparisons of tags shown by Amazon.com and our proposed approach for a camera. We can see that the tags generated by Amazon.com are mostly aspects (e.g., lens, photo) and do not provide meaningful information. On the other hand, the tags generated by our proposed approach (e.g., cheap quality lens, slow focus) are more informative and readable.

### 1.4.3 Unsupervised Stance Detection in Comparative Reviews

Many of the top helpful reviews are comparative in nature, where a product is compared with another rival product. Comparative reviews have two opposing targets and opinions expressed on the targets. Knowing the stance of users can influence people's opinions and play an important role in forming public opinion, discourse analysis and text summarization. However, in such settings, it is hard to relate opinions and targets. Figure 1.6 shows a comparative review comparing two rival products: Windows vs. Mac.



**Windows vs. Mac**  
“Everyone knows that Windows is the better operating system because it has more softwares, more dev support, more everything. Mac is like the retarded little brother who is too much pampered. It has Wi-Fi connection problem and lacks gaming support.”

Figure 1.6: An example of a comparative review: Windows vs Mac.

Here, the user expresses her support for Windows and talks about the flaws of Mac. Since the user writes about both the targets, it is difficult to figure out which target each aspect supports. Aspects such as *operating system*, *software*, *dev support* are associated with *Windows*, while aspects such as *Wi-Fi connection* and *gaming support* are associated with *Mac*.

It would be useful to summarize such comparative reviews by determining the stance of each user. In this thesis, we propose an unsupervised approach to find the stance of users in comparative reviews through the use of word embeddings [99].

## 1.5 Thesis Contributions

The main contributions of the dissertation is summarized as follows:

- We propose an unsupervised approach to extract aspects from reviews and organize the aspects in the form of an ontology tree based on semantic relation between the aspects. A review exploration system is then built by aggregating the reviews based on the aspects of the ontology tree. By using this aspect ontology-based review exploration system, users can smoothly go through the desired reviews without getting overwhelmed by the volume of reviews. The aspect ontology also helps users gain domain knowledge in case they are new to the product.
- We propose a novel approach to generate a meaningful summary of reviews by means of opinionated tags and rank them based on relevance. We choose top-k tags that cover a maximum of the reviews and that are easily readable by users. The top-k tags present a bird’s eye view of the reviews and help users get an instant feel about the product. We further refine the generated tags using NLP syntactic rules to make them more informative. We also address the problem of tag generation for cold products, which have only a limited number of reviews and that too with very short content.
- We summarize comparative reviews by determining the stance of users in their review posts. We propose a novel unsupervised approach to find the target preference of each aspect in comparative reviews, which in turn helps to determine the stance of the users. We propose the use of word embeddings to address the problem of identifying the preferred target of each aspect. Our approach does not depend on any external corpus and hence is domain independent.

## 1.6 Research Overview

The research overview of this dissertation is shown in Figure 1.7. We broadly divide product reviews into single entity reviews and comparative reviews. For single entity

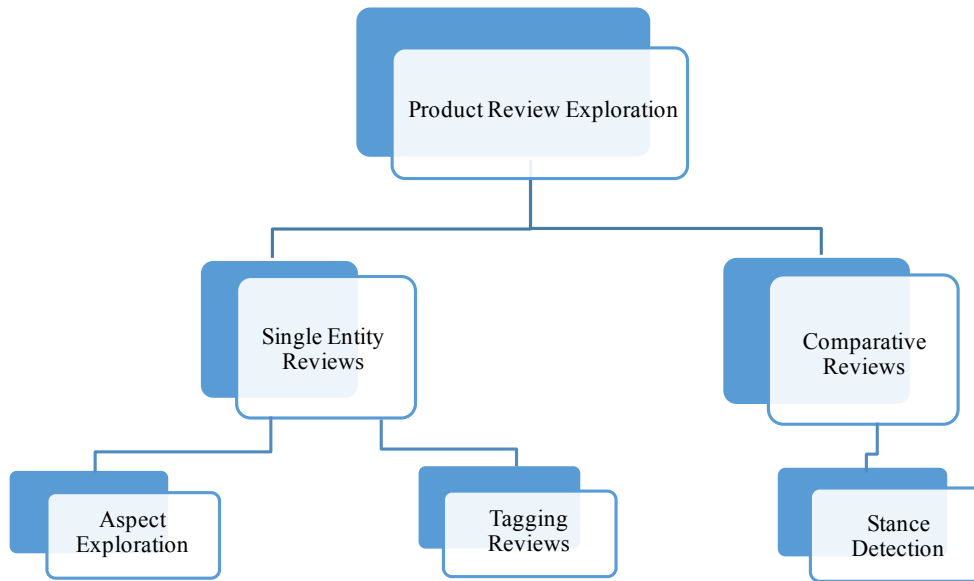


Figure 1.7: Research Overview.

reviews, we perform reviews exploration based on aspect ontology and opinionated tags. For comparative reviews, we summarize the reviews by determining the stance of users.

## 1.7 Organization of the Thesis

The rest of the thesis is organized as follows:

- Chapter 2 presents the background of data exploration in e-commerce and Natural Language Processing (NLP).
- Chapter 3 presents the related work on review exploration and summarization.
- Chapter 4 discusses a novel aspect ontology based review exploration system that allows users to view reviews based on desired aspects.
- Chapter 5 addresses the problem of summarizing reviews by means of informative and readable tags.

- Chapter 6 addresses the problem of summarizing comparative reviews by determining the stance of the users.
- Chapter 7 concludes the thesis and discusses scope for future work.

# Chapter 2

## Existing Systems and Background

In this chapter, we explain some background concepts used in all our works. We start with the basic concept of data exploration in e-commerce followed by basic Natural Language Processing (NLP) we use for opinion mining.

### 2.1 Data Exploration

Two important components of data exploration in e-commerce are product exploration and review exploration. The goal of product exploration is to enable users to access the products easily. Most users have limited knowledge of the product database, and thus, they often end up browsing endlessly through the database without getting what they want. To help users with product exploration, e-commerce sites provide easy to use query and navigation techniques, such as keyword search, auto-suggestion, faceted navigation, category search, result ranking, etc.

Figure 2.1 is a screenshot of standard product navigation by category from Amazon.com. It organizes the products in the form of a category hierarchy. Users can browse through the category hierarchy to find products. Category search helps find related products and products that are frequently bought together.



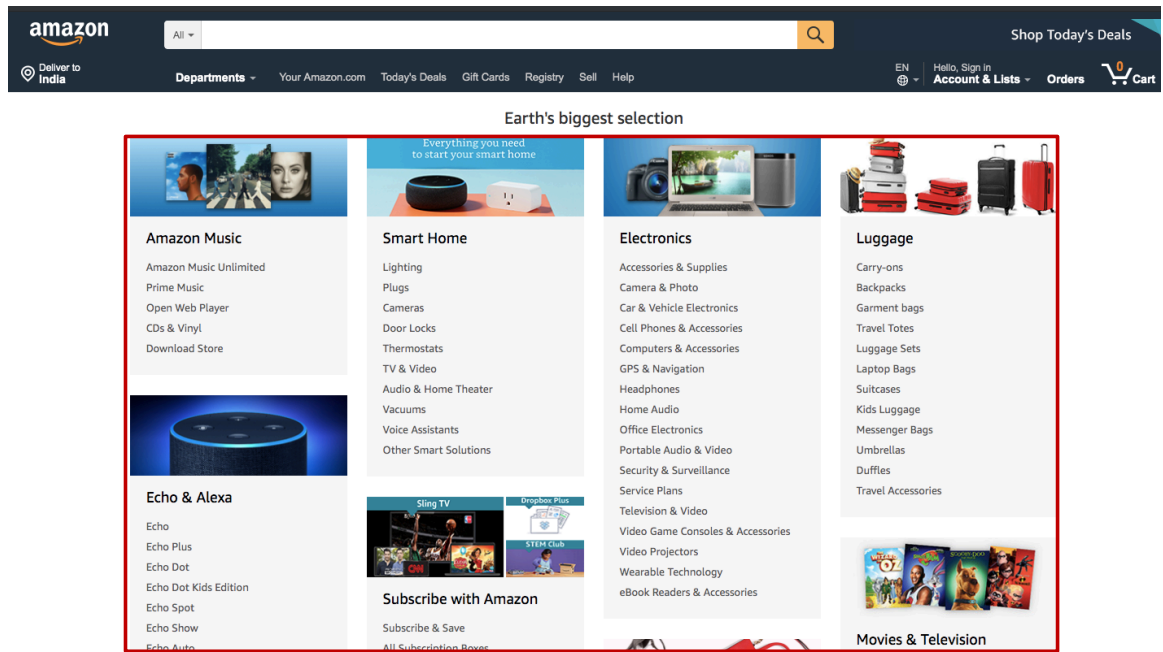


Figure 2.1: Category navigation in Amazon.com.

Figure 2.2 shows another product exploration interface from Amazon.com for browsing a database of cameras. It provides users with options for keyword search as well as faceted navigation to find their desired items. Keyword search is mainly useful when users have specific knowledge of the product they want to buy.

Faceted navigation is a cataloging technique that is used to arrange items in a database based on certain facets or keys. Facets are characteristics of products such as category, relevance, review ratings, etc. When users lack detailed knowledge of the products they desire, they can choose facet values using faceted navigation interface to find products. Facet values are used as preference criteria. Users have the option to select facet values through options such as checkboxes, sliders and input values. Once the facet values are set, products having the facet values are shown to the user. Faceted navigation has the advantage that users can make additional queries through facets to probe further when a search query gives a broad range of products.

Unlike product exploration, review exploration systems are not well-developed. Although reviews play an important role in influencing the purchasing behavior of

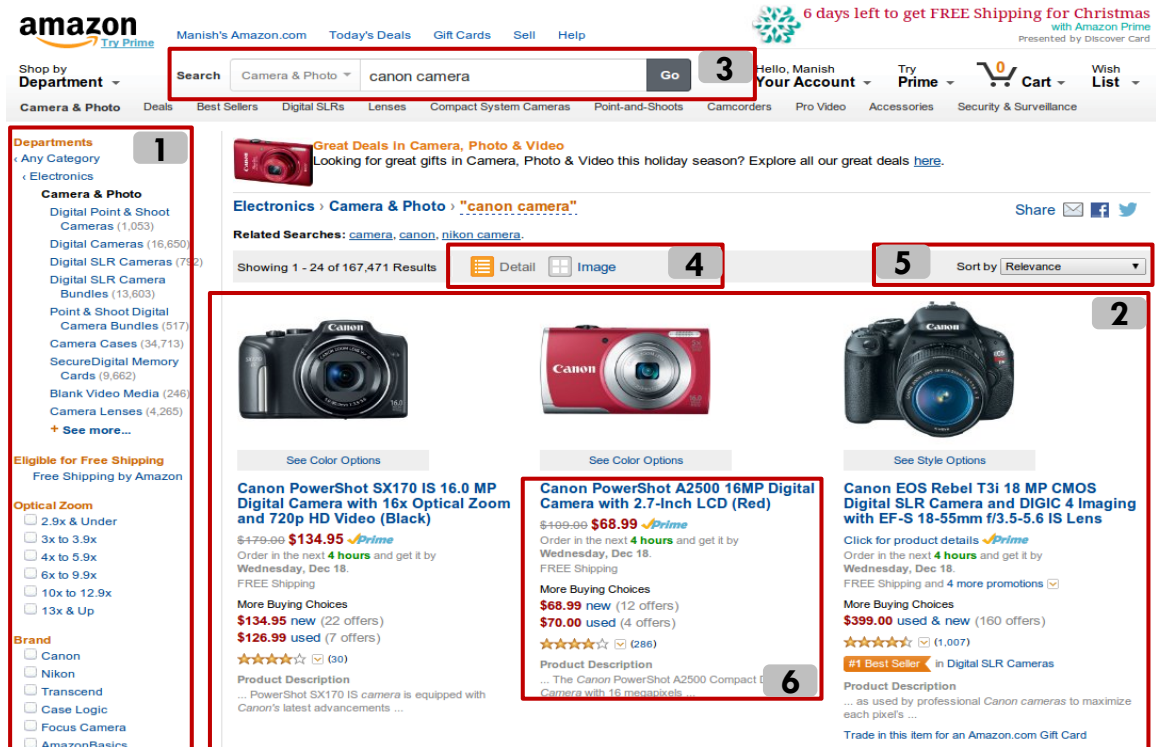
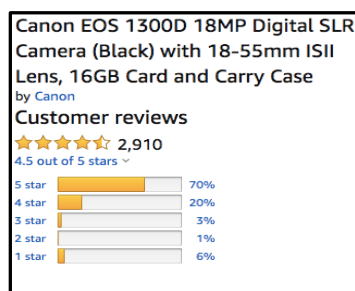
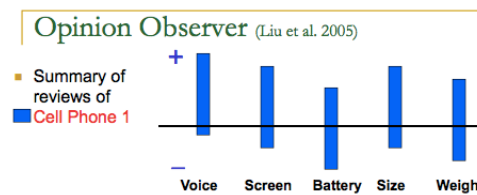


Figure 2.2: Faceted navigation in Amazon.com.

users, many of the existing review exploration systems struggle to present easy to digest information to users. Existing systems summarize reviews by giving an average star rating of the reviews, or by finding the frequency of positive and negative reviews for individual aspects.



(a) Star rating.



(b) Aspect polarity.

Digital Camera 1:	
Aspect: <b>GENERAL</b>	
Positive:	105
Negative:	12
Aspect: <b>Picture quality</b>	
Positive:	95
Negative:	10
Aspect: <b>Battery life</b>	
Positive:	50
Negative:	9

(c) Aspect frequency.

Figure 2.3: Examples of review summarization systems.

Figure 2.3 shows some existing review summarization systems. Figure 2.3a uses

average star rating to summarize reviews. It shows the number of users who have provided a rating score and the percentage of users that belong to each of the rating score. By clicking at a rating score, users can read reviews that have a particular star rating. Figure 2.3b is a screenshot from Opinion Observer [87], where aspects are extracted from reviews and the polarity scales of each aspect are presented. Such representation helps in aspect-level polarity analysis of products. Figure 2.3c presents aspect-frequency based review summarization that shows the number of positive and negative reviews associated with each aspect. For example, the aspect *Battery life* is present in 59 reviews, out of which 50 reviews talk positive about the aspect. This is particularly useful to find popular aspects among users. Such summarization methods do not show the actual opinion of users, which is crucial to understand what other users like or do not like.

It is hard for users to find reviews related to a particular aspect. To help understand the problem, consider a user looking to buy a camera with a good lens. Camera has many aspects and sub-aspects such as *lens quality*, *battery life*, *size* and *price*. Although the user is interested in reading reviews that talk about *lens quality*, there is no review search mechanism to narrow down the reviews.

## 2.2 Preliminaries

### 2.2.1 Review Types

To make the task of opinion mining easier, reviews are divided into single entity reviews and comparative reviews based on the number of entities associated with the review. A single entity review talks about single target/item (e.g., camera product) and opinions are expressed on its aspects (e.g., lens, screen). On the other hand, a comparative review contains a comparison of two targets/items (e.g., Windows vs. Mac) and the aspects are associated with either of the targets (e.g., game → Windows,

OS → Mac).

## 2.2.2 Pre-processing

Unlike structured data or formal text, reviews are hard to deal with as they are mostly free text. As grammatical rules are generally not followed while writing reviews, reviews contain many noisy texts such as stopwords, repeated sequence characters, acronyms, slangs and spelling mistakes.

**Canon camera**  
“This is the **besttt** camera for me. The **bater**y is **sg**. Canon is **knwn** for its long **battery** life.”

Figure 2.4: An example of a single entity review.

One such example can be seen from a Canon camera review shown in Figure 2.4. The review has typos (besttt, bater, knwn) and text-speaks (sg). Such faulty words make the opinion mining process difficult. We clean the reviews using the following pre-processing steps:

**Stop word removal:** Stop words are non-semantic words such as articles, prepositions, pronouns, conjunctions, etc. that are used to while writing sentences. They occur frequently in text (e.g., *what* and *an*), but they are of little value in determining opinion about the targets. We remove stop words by using NLTK’s English stop word corpus.

**Removal of repeated character sequence:** Users write repeated sequence of a character to emphasize sentiment (e.g., Nikon is the *besttttt*). In order to capture such opinion words, we discard any extra letters that occur more than twice in a row, as suggested in Go et al. [43].

**URL filtering:** Users frequently provide URLs (e.g., [www.nikon.com](http://www.nikon.com)) to some other pages for reference or more detailed information. Such URLs are only pointers

and do not contribute to the sentiment of posts. So, URLs are identified and removed from reviews.

**Acronym expansion:** Users frequently use acronyms while writing reviews to save keystroke. Understanding the meaning of acronyms is important as they contain sentiments about the targets. We identify and replace the acronyms with their actual words using an acronym dictionary [62].

**Contraction substitution:** Contractions are commonly occurring short form of words formed by combining a noun or a pronoun with a verb; or a verb and *not* (e.g., *would've*, *I'm*, and *hasn't*). We create a list of contraction dictionary and replace contractions with the expanded words. For instance, *would've* is replaced by *would have*.

**Emoticon substitution:** Emoticons are often used at the end of a sentence to express sentiments towards a target. For example, ‘:)’ indicates happiness, while ‘:(’ indicates sadness. We employ an emoticon dictionary [51] to identify and replace each emoticon with one of the polarity scores: -1, 0 or +1 (e.g., ‘:)’ → +1; ‘:(’ → -1).

**Spelling correction:** Spelling mistake is one of the most common problems in customer reviews. For instance, a customer may misspell the word *excellent* as *excellent*. This may result in different variations of the same tag. In order to correct the spelling mistakes, we employ an NLP based spelling corrector [110].

**Lemmatization:** Lemmatization is performed to substitute the inflectional form of words with corresponding lemmas (e.g. *cars* → *car*, *saw* → *see*, etc.). This prevents multiple occurrences of the same aspect.

### 2.2.3 Part-Of-Speech (POS) Tagging

Part-Of-Speech (POS) plays an important role in detecting aspects and opinions from reviews. Aspects are generally nouns or noun phrases while opinions are mostly adjectives or adverbs.

Canon|NNP is|VBZ a|DT very|RB good|JJ camera|NN

Figure 2.5: An example of a POS tagged sentence.

For instance, consider the review text shown in Figure 2.5. After applying POS tagging, we find that the aspects *camera* and *Canon* are a noun (NN) and a noun phrase (NNP) respectively. The opinion *good* is an adjective (JJ) and the adverb (RB) *very* act as an intensifier of the opinion. POS tags are also used to identify patterns and rules in identifying text structures. We use the Stanford POS parser [70] for POS tagging and to extract aspects, opinions as well as relations between them.

# Chapter 3

## Related Work

This chapter gives a background of existing approaches of opinion mining and discusses the related work in addressing the challenges of review exploration.

### 3.1 Sentiment Analysis

Sentiment analysis or opinion mining focuses on determining the polarity of text content such as tweets, product reviews, and social media post [79, 109, 121, 139]. Sentiment analysis of text documents has been studied extensively [15, 18, 34, 86, 169, 176]. Existing works on sentiment analysis focus on classifying the sentiment of a content [86], which could be carried out at the document level [12, 26, 104, 111, 151], sentence level [4, 5, 77, 163], or aspect level [54, 106, 115, 131, 158, 166]. There are many supervised and unsupervised methods to solve the above problems. Supervised methods [179, 182] rely on training data for sentiment classification, while unsupervised approaches [25, 54] use syntactic rules and lexical resources for polarity mining.

Pang et al. [111] explore various machine learning methods to analyze movie reviews. [12] uses discourse structure and Rhetorical Structure Theory (RST) to improve sentiment classification of documents. Document-level sentiment analysis assumes that all the opinions are associated with a single target. They fail to find sentiments

when the document contains more than one target. Sentence level sentiment analysis is performed by separating subjective sentences from objective ones and then determining the polarity of subjective sentence only [77]. However, objective sentences also may express an opinion. For example, the sentence ‘The camera battery lasts for 10 hours’ is objective, but suggests that the camera has a long battery life. The above two approaches do not give fine-grained information about which features the user likes or does not like.

Although sentiment classification of whole review is an interesting problem, the research done on aspect level sentiment analysis is 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. Aspects are the objects associated with a sentiment. Aspect level sentiment analysis find users opinion on each aspect of a target [54]. It is useful for contents where the user talks about pros and cons of the target product in terms of its aspects.

## 3.2 Opinion Words in Sentiment Analysis

Opinion words or sentiment words are the most valuable component of a sentence that indicates users’ sentiment. They indicate either a positive or negative sentiment towards the target. For instance, the opinion words excellent, speedy, easy to use, etc., indicate positive sentiment, while the opinion words slow, worst, flaw, etc., show negative sentiment. Researchers [49, 64] have worked on extracting opinion words from documents to identify sentiment polarity. Semantic orientation of adjectives is used to determine polarity in [49]. [64] uses WordNet [37] to compute the semantic distance of an opinion word from *bad* and *good*, and polarity classification is made based on the semantic distance. Turney [151] generalize this method by using Pointwise Mutual Information (PMI) to compute the semantic distance between words.



Later, Turney and Littman [152] showed that applying cosine distance in the Latent Semantic Analysis (LSA) space can capture the degree of semantic orientation with better accuracy.

### **3.3 Sentiment Lexicon in Sentiment Analysis**

Considering the significance of opinion words in sentiment analysis, investigators [28, 57, 103, 105] have proposed many methods for creating opinion lexicons. Apart from manually compiled lexicons, there are two other approaches for compiling opinion lexicons: dictionary-based approach and corpus-based approach. Dictionary-based approach [13, 69, 101, 154, 164] uses a collection of opinion words with known polarity as labeled information. The collection is then used to bootstrap and expand by searching for their synonyms and antonyms in a dictionary, such as WordNet. Dictionary-based approaches do not work well in finding the sentiment of context-dependent words. For example, for cameras, if it has a small size, it is usually positive as it is portable. However, for a car, if it is small, it is negative as it is less spacious. The corpus-based approaches [29, 63, 65] overcome this ambiguity by finding the sentiment orientation of words based on the context, which is derived from the corpus.

### **3.4 Aspect-based Sentiment Analysis**

Aspect-based summarization approaches deal with finding the opinions of aspects present in reviews and summarizing reviews based on the aspects. Aspect-based summarization process may be divided into three parts: aspect extraction, polarity identification, and aspect summary generation. Aspect extraction is used to mine relevant aspect from documents, polarity identification finds the sentiment associated with the extracted aspects, and aspect summary generation presents a summary of the aspects and their overall orientation.

## 3.5 Aspect Extraction

Aspect extraction finds relevant aspects from text documents. For instance, some of the prominent aspects of camera are *resolution*, *picture quality* and *flash*. Much work has been done in the area of aspect extraction for opinion mining [54, 87, 130, 166]. Most of the works fall into two categories, namely NLP based and frequent term-based approaches.

### 3.5.1 NLP based Aspect Extraction

NLP-based methods [54, 95, 116] are popularly used to extract aspects from texts. As aspects are mostly noun or noun phrases, POS tagging along with syntax tree parsing is used to identify aspects. NLP syntactic rules are applied to extract aspects and sentiments. Aspect summarization from short comments based on NLP rules was studied by [95]. Wu et al. [166] utilized dependency parser to extract noun phrases and verb phrases from reviews as candidate aspects. Scaffidi et al. [130] applied a language model method to develop an aspect-based product search interface that extracts product aspects and assigns scores to each product aspect.

Liu et al. [87] propose a language pattern mining method to extract aspects from pros and cons reviews. KnowItAll information extraction system [32] is used in [116] to detect aspects from product reviews. The system recursively identifies aspects from noun phrases extracted from reviews by separating aspect and sub-aspect from the noun phrases. For example the noun phrase *camera size* contains the aspect *camera* and the sub-aspect *size*. NLP based approaches fail to identify aspects that are not nouns.

### 3.5.2 Frequent Term-based Aspect Extraction

Term-based approaches use frequency to extract aspects. Using frequent term mining for aspect extraction helps to overcome the weakness of NLP-based approaches. By introducing measures like support, confidence and frequency patterns to determine aspects, term-based approaches are not restricted to specific word patterns that occur in NLP-based approaches. Such approaches also help to prune out irrelevant aspects by means of redundancy and compactness pruning.

Most of the term-based approaches analyze contents present in documents and parts of documents for aspect extraction. [54] is one of the pioneering work that used association rule mining of documents to extract frequent nouns and noun phrases as aspects. [56] proposed a similar work to extract aspects from car and movie reviews. [183] used aspect lexicon and regular expressions to detect candidate aspects from movie reviews. Among the candidate aspects, the terms with high frequency are considered as aspects. [71, 76] consider term occurrence in documents as well as paragraphs for aspect extraction. Unlike traditional approaches, they consider word frequency within paragraphs as well as across paragraphs to give more refined results. However, term-based approaches do not work well on a small corpus and their performance may vary across different domains.

### 3.5.3 Limitation of Aspect-based Summarization

The main limitation of existing aspect-based opinion mining approaches is that they fail to extract the hierarchical relationships between aspects. For example, *lens* is a sub-entity of camera and the aspects, such as *focus*, *lens cap*, and *zoom* are sub-aspects of lens and not of camera directly. Existing works [55, 115] do not consider this relationship of entities and sub-entities while performing 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.

### 3.6 Review visualization

In this section, we present related work on review visualization. As information visualization plays a complementary role in understanding data, researchers [6, 27, 162] have proposed various techniques for enhancing review summary by means of information graphics. The use of content visualization as an effective means of summarizing reviews is studied in [20, 36, 45, 175]. The work in [20] uses Treemaps [135] for visualizing aspect hierarchy. Using Treemap, they represent aspect nodes as rectangles and uses nested rectangles to show sub-aspects. The rectangles are color-coded to represent the scale of polarity and size of the rectangles vary based on the importance of the aspect. However, the nested rectangles make the structure complicated.

Ontology is popularly used an effective means of visualizing topical information of documents, enabling interoperability of data and enhancing information retrieval [9, 39, 68, 132, 180]. The use of ontology for knowledge representation [35, 66, 72, 82, 83] has gained popularity in the last decade. The use of ontology to help in solving the heterogeneity problem of representations and terminologies in e-commerce is explored in [96]. The work in [83] focuses on building an operational ontology system to represent product information. They follow a three-level metamodeling approach and implement ontology using Web Ontology Language (OWL). However, this approach is hard to train and implement. [82] work on developing an enterprise-scale ontology management system and propose an approach to represent ontologies in the form of relational database tables. [150] use concept hierarchies of ontology to discover keywords in patent documents. These approaches make use of an already existing ontology to perform their respective tasks.

Due to the usefulness of ontology, a lot of work has been done for efficient construction of ontology. [35] uses NLP and information extraction techniques to acquire and classify ontology instances. [66] create ontology by extracting key concepts from text using NLP, statistical knowledge and domain knowledge. Lu and Zhai [94] use semi-supervised topic models to build ontology with integrated opinion. They assume the documents to be structured and well-written with explicit and implicit aspect information, which is not the case with reviews and feedback. Sauper and Barzilay [129] follow a supervised approach to construct topic ontology from Wikipedia articles using titles and section names of documents. These methods require a large number of training documents to create an ontology.

Ontology has been popularly used in visualizing and summarizing movie reviews. Formal Concept Analysis (FCA) with manual input is used in [133] for constructing aspect ontology for movie reviews. [181] use ontology to tackle the sentiment analysis problem. They start with a top-down approach of ontology construction based on the IMDb [58] movie metadata taxonomy such as title, cast, crew, locations, etc. Texts from movie reviews are divided into segments and each segment gets assigned to a node of the ontology. They aggregate the sentiment of text segments to determine the polarity of each node.

Extant ontology construction methods require manual input in some form for the creation of an ontology tree. Some of the methods are domain-specific, while others require pre-defined ontology rules or structures. In contrast to the aforementioned works, we propose an automatic approach to generate the aspect ontology tree using similarity techniques that work across domains. In this thesis, we work on automatic creation of aspect ontology to complement sentiment analysis.

## 3.7 Tag Generation

Because of the nature of review data, opinion summarization differs from general text summarization. Text summarization [52] deals with extracting prominent topics from content and combining them to produce a summary of the entire content. On the other hand, opinion words and their polarity form a crucial component of opinion summary.

Researchers have studied the problem of tag recommendation for various applications. Existing approaches of tag recommendation can be broadly classified into two categories: supervised [100, 136, 165] and unsupervised [124, 141, 159]. Majority of the work falls into the supervised approach where the given object or item already has some tags, and the task is to expand the tags or predict tags of similar items. Supervised approaches may be further classified into three sub-categories based on how they work: (1) co-occurrence based approaches, (2) Matrix-based approaches, and (3) graph-based approaches.

On the other hand, unsupervised approaches do not require any tag to work. They extract information from the content and its associated features to recommend tags. In this thesis, we work on extracting tags from contents that do not have tags associated with them.

### 3.7.1 Supervised Tag Recommendation

Supervised approaches can be used where some tagged data is available in the form of training data. For recommendation systems, such data form ternary relations between users, items and tags. For news posts, blog posts, etc., it is in the form of binary relation between items and tags. Related work on various supervised approaches are discussed below:

**Co-occurrence based approaches** utilize tags assigned to a collection of objects

to recommend tags of a new object using association rules. Given some existing tags associated with a target object, this approach expands the tag collection based on the co-occurrence pattern of the candidate tag with the initial tags [42,50,75,97,137,165]. [42] propose an interactive way of recommending relevant tags whenever a user selects a new tag by considering the user’s tag usage history. The co-occurrence based approaches are dependent on the availability of some initial tags assigned to the target objects. These approaches require a lot of tagged data as they use tag co-occurrence to recommend tags for the target object.

**Matrix factorization approaches** work well for data that have information on users, tags and items. They represent the tag assignment to items in the form of a matrix and factorize the matrix into smaller ones to discover latent features between different entities. Reducing the dimension of the matrix makes it easier to find the relationships between users, objects and tags. [126] used matrix factorization and TensorFlow [1] to model the pairwise relationship between users, tags and items. Although this exploits the benefits of dimensionality reduction, matrix factorization work well only for dense data and involves expensive operations that are not so scalable [11].

**Graph-based approaches** follow a graph representation of the system, where the nodes are the objects, and the edges denote the similarity relationship between the objects. They exploit the textual features and neighborhood similarity information to find candidate tags [53,60,92]. The social connection between users is exploited in [89] to build a graph-based tag recommendation system. Tag propagation between objects with similar textual content is proposed in [92,177]. [38] modeled the folksonomy as a heterogeneous graph containing tags, users, and objects as nodes. Graph-based approaches do not perform well when the data has noisy content like user reviews.

### 3.7.2 Unsupervised Tag Generation

Unsupervised methods are suitable for items where only the content is available without any tags. They are generally used to find relevant topics from documents such as social media posts [124], community question answering sites [78] and scientific publications [80, 127]. They leverage contextual information such as patterns, similarities and syntactic rules to generate topical tags. Topic modeling has been used to extract topical words from texts and to suggest relevant tags to characterize the contents [124]. [141] followed a graph-partitioning method to find the most representative topics from a document. The performance of the aforementioned methods highly relies on the assumption of a dense set of data upon which the model is built. [127] use content-based unsupervised tag recommendations as a mechanism for expert profiling in the scientific domain. They showed that keywords are the most effective content to generate tags compared to titles and abstracts.

Our approach differs from the existing methods in that (1) We focus on tagging opinionated documents, such as product reviews and debates, (2) We focus on generating tags that capture the semantic objects from reviews. Existing works on unsupervised tag recommendation focus on extracting topical tags that has objective information. Extracting only topical tags are not helpful to understand product reviews. Reviews contain lots of subjective information from the reviewer. We incorporate NLP rules to associate semantic words to the tags. (3) Product reviews are harder to deal with compared to the scientific documents. Reviews are relatively short compared to other documents and do not have well-defined components, such as title, abstract and keywords, that complement tag generation. While scientific documents are well structured, reviews are generally unstructured, which makes it hard to process.

We give more focus on extracting tags that also have sentiment. Unlike existing works [8, 24, 67, 149] that focus on extracting aspects and polarity, we use sentiment



analysis to improve the tag generation by adding prominent sentiment words related to the tags. Adding dominant sentiment words make the tags more informative. We combine the contextual and syntactic information from the reviews to generate tags that help understand product quality.

### 3.8 Mining Comparative Reviews

Although finding users' sentiment in comparative reviews is closely related to aspect-based opinion mining and opinion summarization, it represents a more complex problem. To solve this problem, one needs to precisely find the targets of opinions [86]. This problem is slightly easier for product reviews compared to other types of opinion data, such as debates, blogs, etc. [140]. 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.

Opinion mining in comparative reviews is closely related to stance detection of online debates. Existing work on debate stance classification can be broadly categorized into two settings, viz. (1) congressional floor debates [2, 157, 172], and (2) social debates [46–48, 73, 108, 143]. The congressional floor debates are the more professional debates, whereas the social debates are informal. Stance classification in social debates is more challenging compared to congressional floor debates due to the expressive language, such as slangs, lack of grammar, abbreviations, etc., used in the social discussions [3, 122, 156]. Stance classification of congressional floor debates investigates the political orientation of users on issues such as political reforms, controversial policies, and candidate preference [7, 17, 147]. Stance detection of social debates aim at detecting the users' stance towards topics in informal settings such as

online public forums and debate websites<sup>1</sup> [17, 140, 147]. In this thesis, we perform stance detection for social debates, where the post length is long compared to tweets.

Our work is an extension of the work of Somasundaran et al. [140]. We employ word embedding techniques in various steps of stance detection. This approach speeds up the stance classification process and gives more accurate result compared to the earlier approach that requires crawling of external information related to the targets to find aspect-target preference information. Researchers have used word embeddings for opinion mining due to its ability to capture semantic similarities [91, 112, 168]. Liu et al. [91] used word vectors trained from a large corpus of product reviews for opinion aspect extraction. Xu et al. [168] used word embeddings as feature representation for a joint opinion relation detection. Pavlopoulos and Androutsopoulos [112] used word vectors trained on English Wikipedia to extract aspect terms for sentiment analysis. Mikolov et al. [99] developed *word2vec* model for creating neural-embedded word representations. Various researches have shown that word2vec model captures relational and semantic similarities better than other methods [84, 85].

Our work is also related to the aspect extraction step of opinion mining. Recent works have used word embeddings for aspect extraction [118, 173]. Nevertheless, identifying the target-related aspects from a comparative review post is a more challenging task as there are more than one targets involved in a debate post. Somasundaran and Wiebe [140] used syntactic rules to extract aspect from debate posts. However, many of the extracted aspects are irrelevant to the debate targets. In this thesis, we propose a novel word embedding based method to train a supervised aspect classifier to prune aspects irrelevant to the targets. The classifier can be used for transfer learning to find target related aspects across multiple domains.

---

<sup>1</sup><http://www.convinceme.net>, <http://www.createdebate.com>, and <http://www.debate.org> are some of the popular social debate sites

# Chapter 4

## Exploration of Reviews using Aspect Ontology

### 4.1 Introduction

Users make extensive use of online portals and e-commerce sites to purchase goods and services. After using 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.

Many of the existing opinion mining systems do not focus on high-precision aspect extraction as they do not explain the extracted aspects to users. In this chapter, we present 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 primary 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.

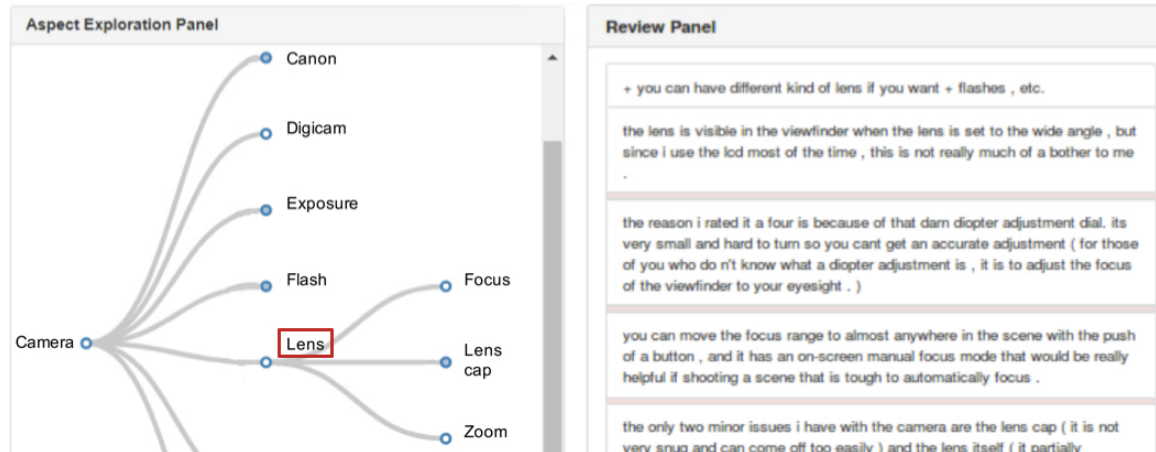


Figure 4.1: Screenshot of the ontology-based review exploration system.

Figure 4.1 shows a screenshot of our proposed ontology-based review exploration system. Here, we take the example of browsing through a 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 aspects with the reviews.

Unlike any other review summarization methods [54, 183], 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 randomly in a flat aspect structure. 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 main contributions of this chapter are as follow:

- We identify popular aspects that are mentioned by many users and use these aspects for creating aspect ontology.
- Based on aspect relationship, we propose to arrange the aspects present in reviews in the form of an ontology.
- We aggregate reviews at different granularity levels based on aspects present in the ontology.
- We propose two unsupervised approaches to generate aspect ontology.
- Our best approach can generate ontology, which is closely similar to the manually created gold standard ontology.

The remaining chapter is organized as follows. Section 4.2 includes a detailed description of our review exploration system. The experimental evaluations and results are presented in Section 4.3. Section 4.4 contains the conclusion and future scope of the study.

## 4.2 Proposed Approach

In this section, we formally introduce our aspect-based review exploration system. The architectural overview of the system is given in Figure 4.2.

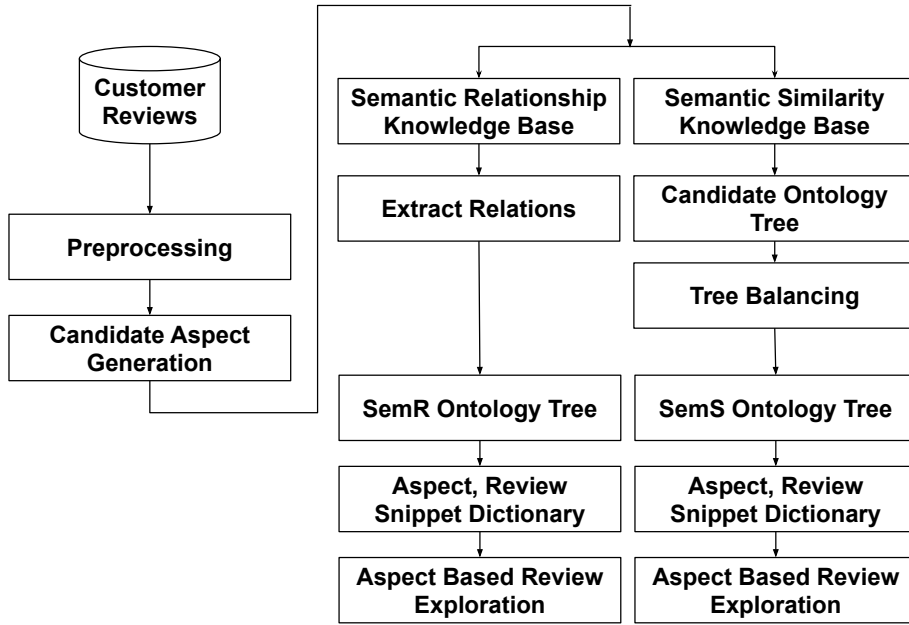


Figure 4.2: System architecture of the aspect-based review exploration system.

The system takes product reviews as input and generates output as an aspect-based review summary, where 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 4.1 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.

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

---

<b>Notation</b>	<b>Description</b>
$A_C$	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

---

Table 4.1: Notations used.

---

**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:  $A_C \leftarrow \text{EXTRACTCANDIDATEASPECT}(D_R, P)$
  - 2:  $A_P \leftarrow \text{EXTRACTPOPULARASPECT}(D_R, A_C)$
  - 3:  $T_O, A_T \leftarrow \text{CREATEONTOLOGYTREE}(A_P, K_S)$
  - 4:  $D_{AR} \leftarrow \text{ASPECTREVIEWDICTIONARY}(A_T, D_R)$
  - 5:  $T \leftarrow \text{ARESAYS}(T_O, D_{AR})$
- 

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.

### 4.2.1 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 [21, 26, 30, 54, 91, 111, 119]. 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 aspect overload and confuse users during aspect exploration. To tackle this problem, we attempt to filter 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 a few times, compared to popular aspects. However, there may be popular aspects which have a low semantic relationship with the product like *camera* and *software*. We prune out such unrelated aspects using semantic knowledge during aspect ontology tree generation.

## 4.2.2 Schematic Interface

The schematic design of our aspect-based review exploratory system consists of an interface (as shown in Figure 4.1) that enables the user to browse through various product aspects and go through the related review snippets. The 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 it easy for the end-user to identify product aspects of interest, and focus on reviews that contain relevant aspect information.

### 4.2.3 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 a 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 aspects usually co-occur with the product in documents.

#### **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 find semantically related aspects using knowledge graph. Knowledge graph contains a huge database of common sense knowledge that links concepts (nodes) with semantic relations (edges). Concepts are real-world ob-

jects. We use ConceptNet [88] knowledge graph 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. For example, consider the following sentences:

**Example 1** ‘*Camera has a lens*’

**Example 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.

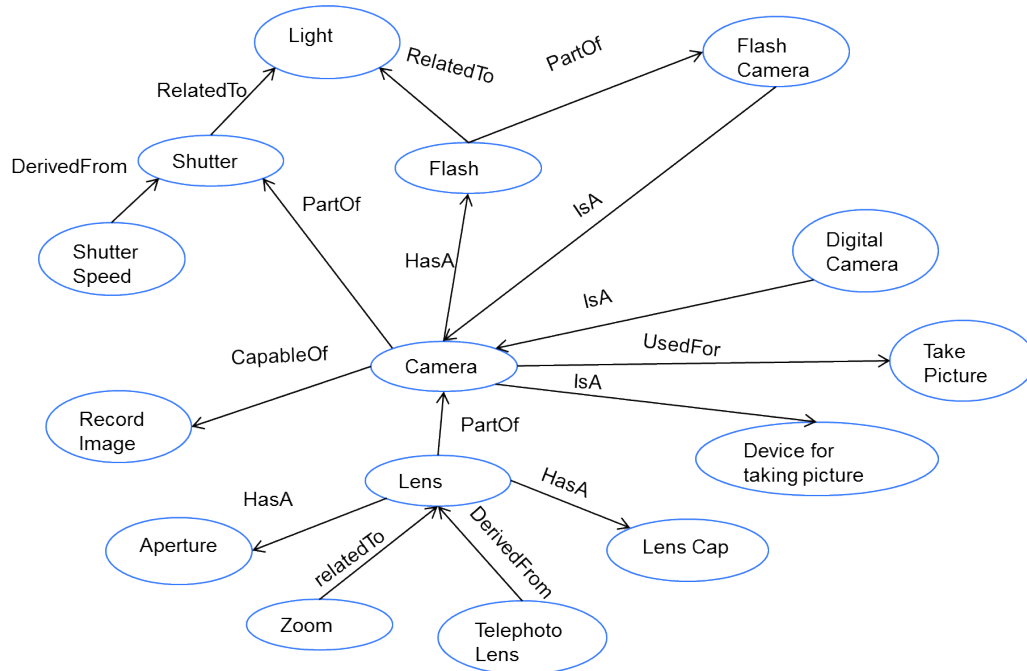


Figure 4.3: A snippet from ConceptNet’s semantic network, showing various concepts related to the entity *camera*.

Figure 4.3 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 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. To understand consider the following ConceptNet instances:

**Example 3** ‘*camera is RelatedTo lens*’

**Example 4** ‘*lens is UsedFor focus*’

From Examples 3 and 4, we find that the concept *camera* has relation with the concept *lens*, and *lens* has relation with *focus*. Hence, it is evident that *camera* and *focus* are related through transitive relation. However, there may be one-to-many relations between various concepts. Linking multiple concepts may lead to topic drift. To illustrate, consider the ConceptNet relations given in Figure 4.4.

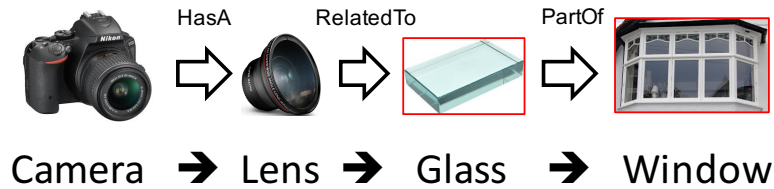


Figure 4.4: An instance of topic drift.

From Figure 4.4, we can see that the concepts *camera* and *window* share a transitive relationship as per ConceptNet but *window* doesn't belong to the camera domain, resulting causes topic drift. To control topic drift, as in [107], we use three classes of relations: Hierarchical relations (H), Synonym relations (S), and Functional relations (F), with the 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. When there are two or more possible relations between concepts, the relation classes help to choose the relation that has a higher priority order.  $H > S > F$  priority order is followed to resolve the conflict.

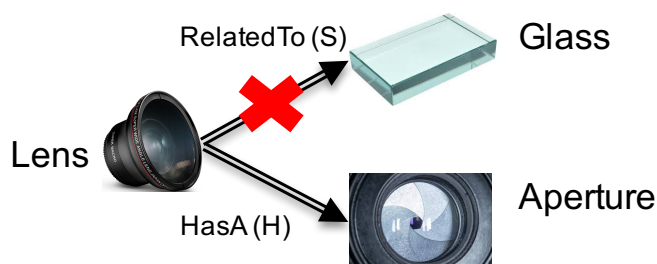


Figure 4.5: Using relation class to resolve topic drift.

For example, Figure 4.5 shows a case where there are two possible relations from the concept *lens* to *glass* and *aperture*. Using the relation class, *lens* and *aperture* has a Hierarchical relation (H), which has higher priority over the Synonym relation (S) that exist between *lens* and *glass*. Based on this precedence, we conclude that there is a relation between *lens* and *aperture* and ignore the relationship between *lens* and *glass*, which would have caused topic drift. Accordingly, we add an edge from lens to aperture ( $lens \rightarrow aperture$ ). Also, to cluster all the relevant synonyms at a single respective aspect, we set inner priority order within the synonym relation class, as mentioned in Table 4.2. In case of conflicting relations, an edge is added to 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.

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 ExternalURL, MotivatedByGoal, ObstructedBy, etc.). The relations along with relation class and priority order are shown in Table 4.2.

Relational Class	Relation	Priority
Hierarchical Relations ( <i>H</i> )	<i>HasA, PartOf, MadeOf, LocatedNear</i>	1
Synonym Relations ( <i>S</i> )	<i>Synonym, DefinedAs &gt; DerivedFrom &gt; IsA, RelatedTo</i>	2
Functional Relations ( <i>F</i> )	<i>UsedFor, CapableOf, HasProperty</i>	3

Table 4.2: ConceptNet relations used for building the ontology tree along with their relational class and priority order. ‘>’ shows inner class priority order.

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. Directionality of the edge is required to differentiate aspects and sub-aspects. 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 introduce the concept of *edge-type* to identify parent and child aspects, which is in turn used to determine the direction of relations. We classify relations into three edge types: Top-to-Bottom Relation (TBR), Bottom Up Relation (BUR) and Bidirectional Relation (BDR). Relations belonging to these edge types are given in Table 4.3.

Edge Type	Relations
Top to Bottom Relations (TBR)	<i>CapableOf, HasA, HasProperty, LocatedNear, MadeOf, UsedFor</i>
Bottom Up Relation (BUR)	<i>DerivedFrom, PartOf</i>
Bidirectional Relation (BDR)	<i>DefinedAs, IsA, RelatedTo, Synonym</i>

Table 4.3: Edge types and relation list.

TBR represents a 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:

**Example 5** *'lens is PartOf camera'*

**Example 6** *'camera is CapableOf taking pictures'*

**Example 7** *'focus is RelatedTo lens'*

In the Example 5, *lens* and *camera* has a BUR, indicating that *lens* is a child of *camera*. Hence, an edge is added from *camera* to *lens* ( $camera \rightarrow lens$ ) in the ontology tree. Similarly, for Examples 6 and 7, edges ( $camera \rightarrow picture$ ) and ( $focus \leftrightarrow lens$ ) are added to the ontology tree as they have TBR and BDR respectively. For bidirectional 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. To maintain tree consistency, we direct edges only from top to bottom.

In addition to edge types, we follow 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. The benefits of using BFS exploration is that it provides the shortest path to relevant concepts from product domain at minimum depth. Our SemR approach is described in Algorithm 2.

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

---

**Algorithm 2** Semantic Knowledge Ontology Tree (SemR).

---

**Input:**  $D$ : Popular aspects dictionary =  $\langle A_P, f \rangle$ , where  
 $A_P$ : Popular aspect set =  $\{A_{P_1}, A_{P_2}, \dots, A_{P_n}\}$   
 $f$ : Frequency of each candidate aspect =  $\{f_1, f_2, \dots, f_n\}$ ,  
 $R$ : 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:  $visited[P] \leftarrow true$ 
5: for all relation  $\in R$  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:               $parent(v_i) \leftarrow c$ 
22:            end if
23:          end if
24:           $Q.enqueue(v_i)$ 
25:           $visited[v_i] \leftarrow true$ 
26:        end if
27:      end for
28:    end while
29:  end for
30:  $T_O, A_T \leftarrow MERGESYNONYMS(G)$ 
```

---



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 [37] to avoid duplicates.

### **Ontology using Semantic Similarity (SemS)**

Since the knowledge graph is manually created, the relations present in the knowledge graph limits the semantic relationship-based approach. Aspects which are not present in the knowledge base are not considered while creating the aspect ontology. Moreover, strength of relation is not defined in SemR approach. 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 often 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 sec’, the related aspects *lens* and *shutter speed* appear together. Similarly, the two related aspects co-occur together in other documents also. Semantic similarity is measured by considering a huge corpus of documents and finding the co-occurrence pattern of the aspects in those documents.

We choose Latent Semantic Analysis (LSA: [81]) as a metric for measuring semantic similarity. 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 applies Singular Value Decomposition (SVD) for dimensionality reduction. The term-topic matrix obtained after SVD decomposition is used to represent terms as topic vectors. 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. Figure 4.6 shows how the LSA score is computed between words.

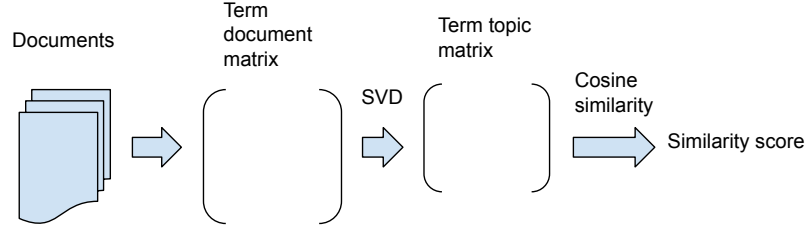


Figure 4.6: Computing LSA score between words.

To compute the LSA score between aspects, we use SEMILAR semantic similarity toolkit [128]. SEMILAR provides a framework for a 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 use of a huge corpus helps to overcome the limitations of SemR approach. 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 the semantic knowledge base approach. The details of ontology tree creation using similarity score is given in Algorithm 3.

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,

---

**Algorithm 3** Semantic Similarity Ontology Tree (SemS).

---

**Input:**  $D$ : Aspects dictionary =  $\langle A_P, f \rangle$ , where  
 $A_P$ : Popular aspect set =  $\{A_{P_1}, A_{P_2}, \dots, A_{P_n}\}$ ,  
 $f$ : Frequency count of each candidate aspect =  $\{f_1, f_2, \dots, f_n\}$ ,  
 $T_{LSA}$ : 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:             $E.ADDEDGE(c, v_i)$ 
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:  $T_O, A_T \leftarrow MERGESYNONYMS(G)$ 
```

---

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*  $\rightarrow$  *flash*  $\rightarrow$  *shot*  $\rightarrow$  *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:

$$T_{LSA} = \alpha + \delta(l - 1) \tag{4.1}$$

where,  $\alpha$  is the initial value of similarity threshold used to find the first level nodes.  $l$  represents the level of the ontology tree, and  $\delta$  act as a damping factor for topic drift. We increase its value by a factor  $\delta$  at each level down the tree. This makes the relationship constraint more stringent and ensures that only closely related aspects

are preserved when we go down the tree.

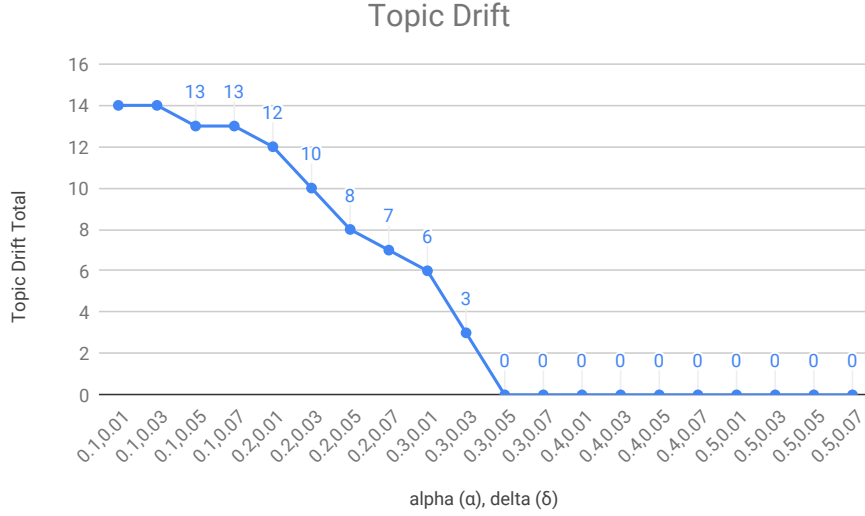


Figure 4.7: Number of topic drift observed by varying the values of  $\alpha$  and  $\delta$ .

$\alpha$	$\delta$	$T_{LSA}$ for level $L_1$	Topic drift in level $L_1$	$T_{LSA}$ for level $L_2$	Topic drift in level $L_2$	$T_{LSA}$ for level $L_3$	Topic drift in level $L_3$	Topic drift Total
0.1	0.01	0.1	2	0.11	5	0.12	7	14
0.1	0.03	0.1	2	0.13	5	0.16	7	14
0.1	0.05	0.1	2	0.15	5	0.2	6	13
0.1	0.07	0.1	2	0.17	5	0.24	6	13
0.2	0.01	0.2	1	0.21	5	0.22	6	12
0.2	0.03	0.2	1	0.23	3	0.26	6	10
0.2	0.05	0.2	1	0.25	3	0.3	4	8
0.2	0.07	0.2	1	0.27	2	0.34	4	7
0.3	0.01	0.3	0	0.31	2	0.32	4	6
0.3	0.03	0.3	0	0.33	0	0.36	2	2
0.3	0.05	0.3	0	0.35	0	0.4	0	0
0.3	0.07	0.3	0	0.37	0	0.44	0	0
0.4	0.01	0.4	0	0.41	0	0.42	0	0
0.4	0.03	0.4	0	0.43	0	0.46	0	0
0.4	0.05	0.4	0	0.45	0	0.5	0	0
0.4	0.07	0.4	0	0.47	0	0.54	0	0
0.5	0.01	0.5	0	0.51	0	0.52	0	0
0.5	0.03	0.5	0	0.53	0	0.56	0	0
0.5	0.05	0.5	0	0.55	0	0.6	0	0
0.5	0.07	0.5	0	0.57	0	0.64	0	0

Table 4.4: Values of threshold  $T_{LSA}$  for each tree level with varying values of threshold ( $\alpha$ ) and damping factor ( $\delta$ ).

The values of  $\alpha$  and  $\delta$  are determined empirically. To tune the values of  $\alpha$  and  $\delta$ , we tested Eq. 4.1 using different values of  $\alpha$  ( $\alpha = \{0.10, 0.20, 0.30, 0.40, 0.50\}$ ) and  $\delta$  ( $\delta = \{0.01, 0.03, 0.05, 0.07\}$ ). Figure 4.7 and Table 4.4 respectively show the various

values of topic drift and threshold LSA ( $T_{LSA}$ ) with varying values of  $\alpha$  and  $\delta$ . From Figure 4.7, we observe that the values  $\alpha = 0.30$ ,  $\delta = 0.05$  give the best result.

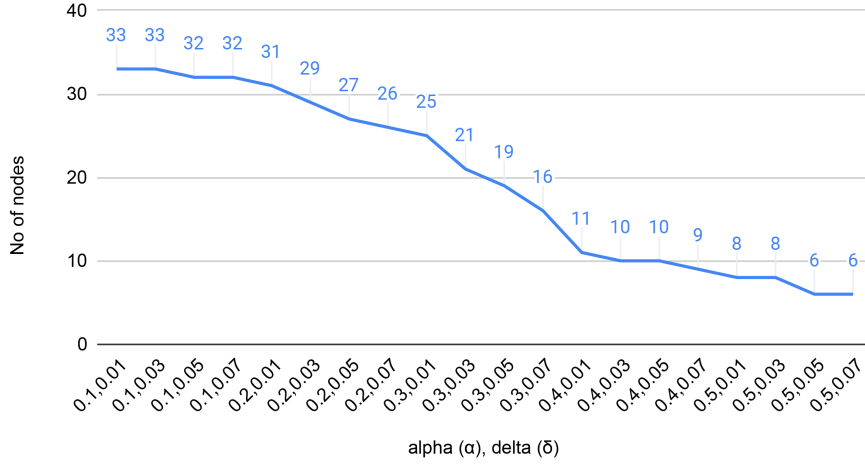


Figure 4.8: Number of nodes in the ontology with varying values  $\alpha$  and  $\delta$ .

Increasing  $T_{LSA}$  further may affect the ontology by eliminating actual aspects. Figure 4.8 shows the number of nodes present in the ontology with varying values of  $\alpha$  and  $\delta$ . We observe that initially as the values of  $\alpha$  and  $\delta$  increase, irrelevant aspects get removed one after another. But, when we increase the values of  $\alpha$  and  $\delta$  beyond the minimal topic drift values ( $\alpha = 0.3$ ,  $\delta = 0.05$ ), correct aspects start getting removed, thereby decreasing the number of nodes.

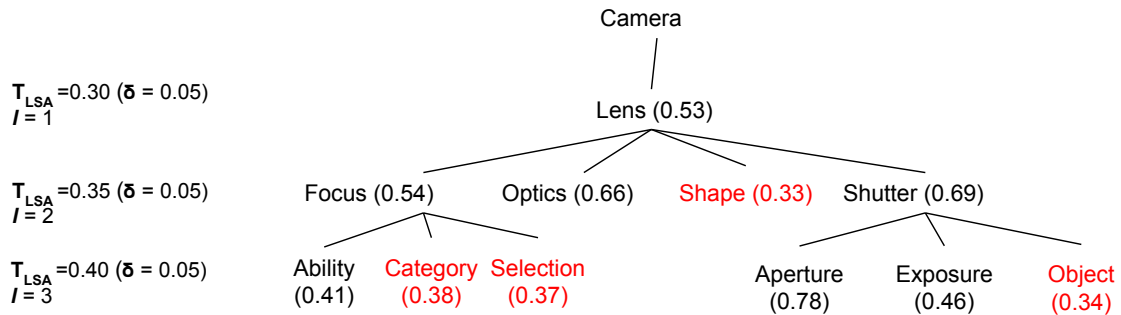


Figure 4.9: Topic drift examples.

Some examples of topic drift removed using modified  $T_{LSA}$  are given in Figure 4.9.

As shown in Figure 4.9, using Eq 4.1 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.

## 4.3 Evaluation

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

### 4.3.1 Experimental Setup

We perform our experiments on the dataset created by [54]. This dataset is commonly used for the evaluation of different aspect extraction methods [90, 119]. The dataset consists of customer reviews of four electronic products, namely digital camera, mobile phone, DVD player, and mp3 player. Table 4.5 presents a summary of the dataset used for evaluation.

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

Table 4.5: Statistics of the product review dataset.

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 pre-processing, 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 [110]. 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 [37]. 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.

### 4.3.2 Ontology Evaluation Metric

The primary contribution of this chapter 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 made 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.

#### 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 these 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 Figure 4.10.

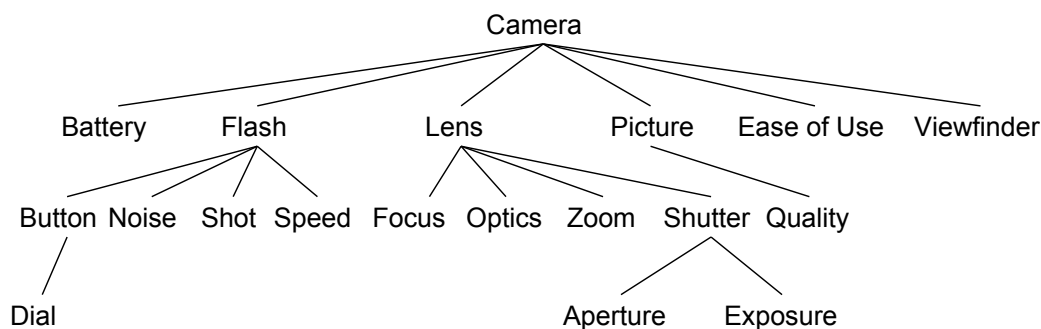


Figure 4.10: Gold-standard aspect ontology tree for camera.

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 frequent 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 [54] 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.

### 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  $|T \setminus C|$ ,  $FN$  (false negative) is  $|C \setminus T|$ . We can formulate precision and recall as follows:

$$Precision, P = \frac{TP}{TP + FP} \quad (4.2)$$

$$Recall, R = \frac{TP}{TP + FN} \quad (4.3)$$

$$F_1 = \frac{2 * P * R}{P + R} \quad (4.4)$$

### 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 [113]. 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:

$$E = \min\{C_j\}; C_j = \{d_j + i_j\}, j = 1, 2, 3, \dots, n \quad (4.5)$$

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

### 4.3.3 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 [55] 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 Table 4.6 and Table 4.7.

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 4.6: Precision and recall of the proposed approaches w.r.t. gold-standard aspect 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

Table 4.7: Similarity between created ontology trees and gold-standard ontology.

### Ontology Using Manually Labeled Aspects

As can be seen in Table 4.6, SemS has higher precision, recall, and F1 score than SemR. Compared to SemR, precision, 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. Figures 4.11 and 4.12 show the ontology obtained using SemR and SemS approach with manually labeled aspects.

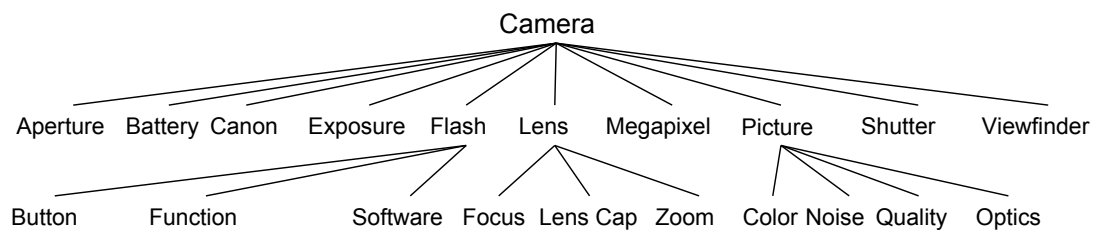


Figure 4.11: Aspect ontology tree created using the SemR approach.

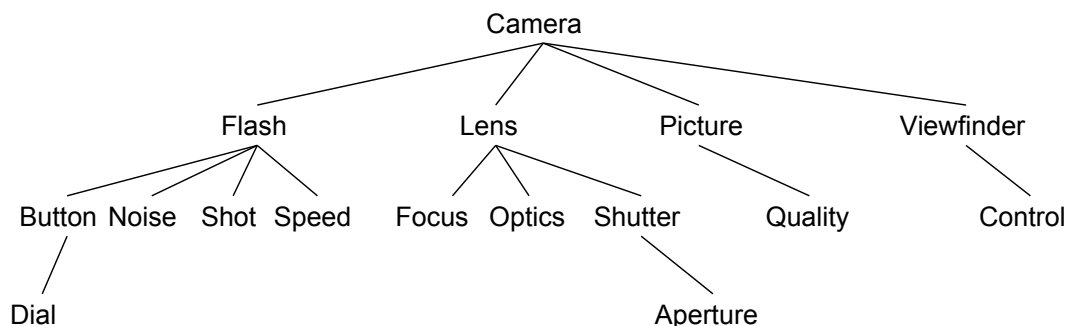


Figure 4.12: Aspect ontology tree created using the SemS approach.

In terms of tree edit distance, from Table 4.7, 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 4.7, 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 [107]. To give an illustration, the concepts *aperture* and *crack* are related. Increasing the aspect ontology height by putting *crack* as a sub-aspect of *aperture* in Figure 4.12 would lead to topic drift as *crack* is not a sub-aspect of *camera*.

### **Ontology using Automatically Extracted Aspects**

Figure 4.13 and 4.14 shows the aspect ontology generated by SemR and SemS respectively on the automatically extracted aspects (AEA). From Table 4.6, 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 is 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 results in a slight increase of tree edit distance for AEA methods.

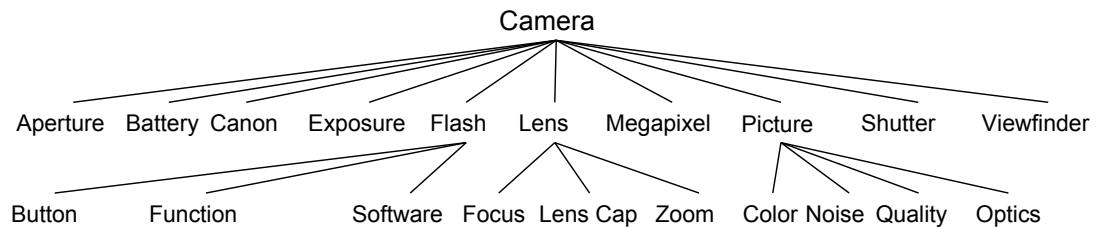


Figure 4.13: Aspect ontology tree created using SemR approach on automatically extracted aspects.

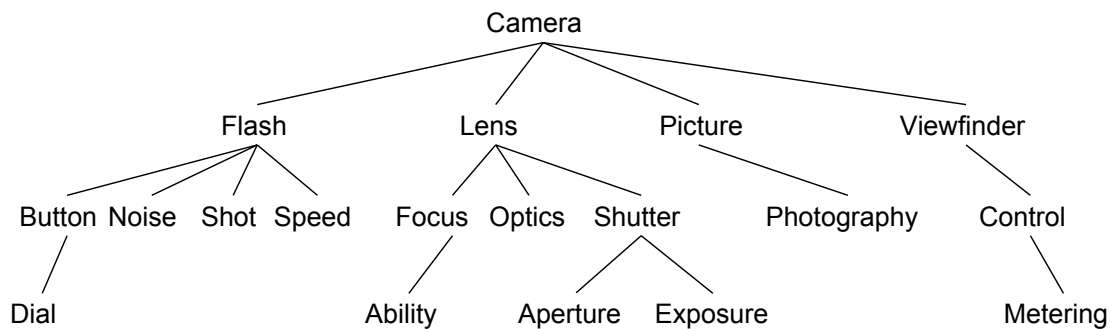


Figure 4.14: Aspect ontology tree created using SemS approach on automatically extracted aspects.

## 4.4 Conclusion

In this chapter, we introduce 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 construct 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 find 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.



# Chapter 5

## Exploration of Reviews using Opinionated Tags

### 5.1 Introduction

Web 2.0 is marked by a tremendous growth of social media and user-generated content, such as texts, images and videos. Tags have lately surfaced as a convenient way of organizing and summarizing such user-generated contents due to their simplicity in indexing and ease of user participation. A tag is a keyword that is used to describe the object of content, and it facilitates keyword-based classification and information search. The unique ability of tags to group and share information has changed how people consume information. Recently researchers [16, 22] have shown that tags are one of the most reliable sources for many Information Retrieval (IR) services, such as content classification, searching and ranking of posts. Tags have also been used for expert profiling [127] and document summarization [155].

Tags are employed in social networking sites and Community Question Answering (CQA) sites, such as Yahoo! Answers [170], Stack Exchange [144] and Quora [120] for indexing question-answers [171] and for routing questions [167] to get answers

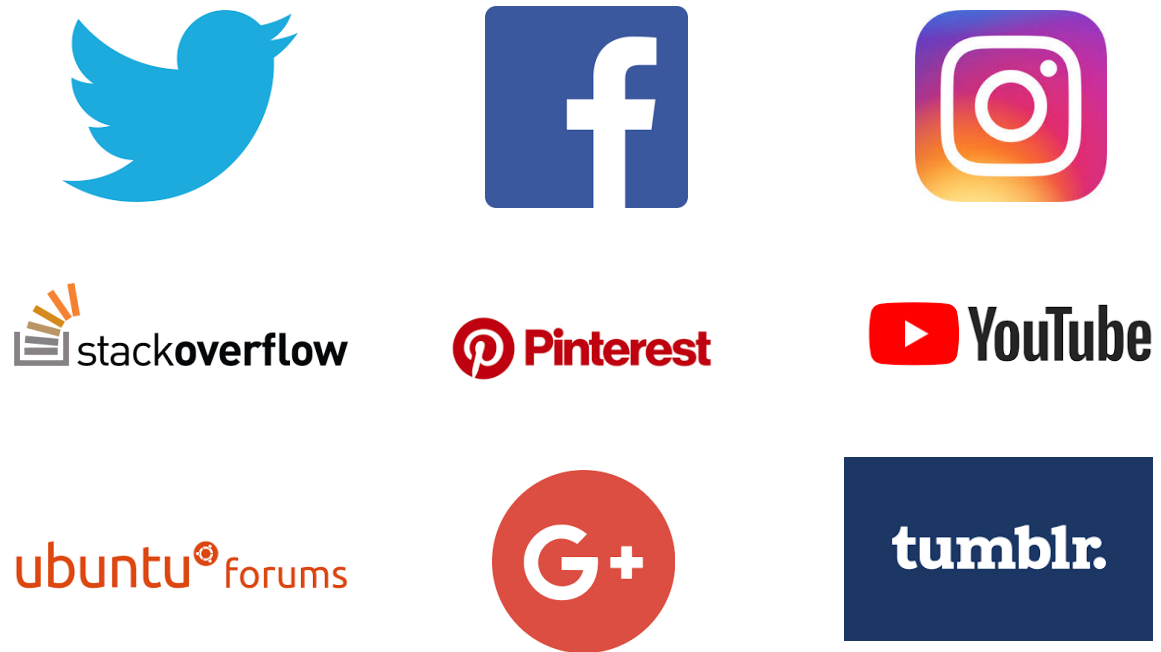


Figure 5.1: Some popular platforms that use tags for information identification and categorization.

from experts. Some of the popular user-generated sites that use tags for organizing information are shown in Figure 5.1.

Although social networking sites allow users to create their own tags while posting content, CQA sites require users to select a set of tags from the collection of tags that are created by privileged users. For quality control, CQA sites allow only users with at least some minimum reputation score to create new tags. The visibility of content depends primarily on the post content and the tags associated with it. Most of the existing tagging schemes rely mainly on their users to manually annotate content with tags.

Manually assigning tags is laborious and is often difficult due to the enormous number of possible tags. For example, the CQA site StackOverflow has around 50K tags, and a user has to select between 1-5 tags that best describe their question. Few of these sites use tag recommendation systems [40, 44, 148] to help the user select good tags. Tag recommendation systems can only help with tag annotation, but not with tag creation. In this chapter, we present a tag generation system for opinionated

contents, such as reviews and debates. Although the problem of sentiment analysis has been widely studied for opinionated contents, to the extent of our knowledge, there is no existing work that has addressed the problem of automatically tagging such contents. We propose the use of tags as a mechanism to summarize product reviews.

A recent trend of research on tag generation [127,134,142] has focused on providing useful and relevant tag suggestion to users. Most of the existing approaches result in unigram tags, which are sometimes not sufficiently informative. In the case of product reviews, the generated tags should be rich enough to eliminate the need for the customer's prior insight into the product. To illustrate, Fig. 5.2 shows the tags generated from reviews of a camera product by Amazon.com.

### Read reviews that mention

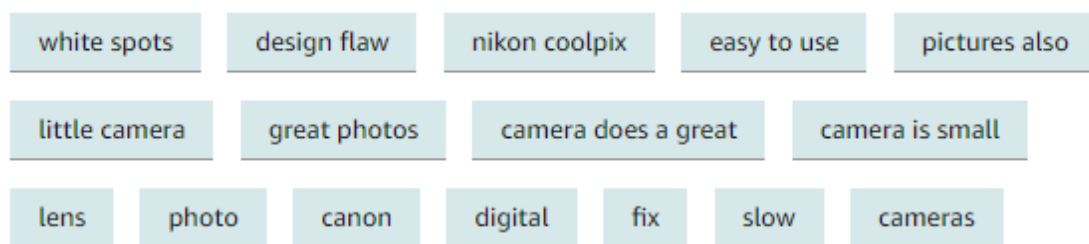


Figure 5.2: Amazon generated tags for a camera from reviews (accessed on 25.12.2018).

It can be noticed that almost half of the tags such as *canon* and *digital* do not provide useful information about the product. Also, tags such as *fix* and *pictures also* provide noisy information about the product. Instead, providing phrases conveys deeper understanding than individual words. To better understand the problem, consider the following example of a camera review:

**Example 8** *'Best picture quality for any point and shoot I have used. The lens is excellent and the camera case is good. It has an amazing flash for the price! If you're just starting out, I would recommend starting out with this flash so that you have*

*something good while saving up for top-notch lighting equipment. It is easy to use even for an amateur such as myself.'*

Here, summarizing the review with the unigram tags such as *flash* is not useful to understand the quality of the product or its aspect *flash*. However, summarizing it with the phrase tag like *amazing flash* makes it easy to understand the customer's sentiment associated with the product aspect *flash*. We extend conventional tag generation approaches to generate meaningful tag phrases from the opinionated text and use the tag phrases to summarize customer reviews.

As products generally have a considerable number of customer reviews, computing a set of meaningful tags for review summarization is a challenging task. A product may have many aspects and some aspects are more popular than others, thereby having a significant impact on consumers' decision. In Example 8, the aspect *picture quality* would have more impact than the aspect *camera case*. *Picture quality* is a primary aspect for buying a camera while *camera case* is an accessory and it is not related to the performance of the camera. We address this task as a tag ranking problem.

Often, there are cases where the grammatical structure of the reviews does not give sufficient information for generating useful tags. In Example 8, the opinion *excellent* is associated with the aspect *lens* although they do not appear adjacent. Forming the tag phrase *excellent lens* makes more sense than the tag *lens*. We introduce five Natural Language Processing (NLP) rules based on Part-of-Speech (POS) structure that helps in making the tags meaningful. Given a product, we build a system that assists users by providing a set of meaningful tags from the product reviews. Our system achieves promising results when evaluated on 12 real-world product review datasets from Amazon.com.

Following are some of the significant observations:

- The tag phrases generated by our unsupervised approach are related to the

products and are found helpful to understand the product reviews.

- Popular (high-rated) products have a higher number of reviews than cold (low-rated) products, suggesting that people are mainly interested in popular products.
- The average length of a review is much shorter for cold products compared to popular products. This is because people tend to talk about their good experiences for popular products and write longer reviews whereas they do not write in length about their bad experiences of using cold products. So, cold products suffer from the problem of generating noisy tags.

We show how tags from popular product reviews can be leveraged to find relevant tags in cold product reviews. Our key contributions of this chapter are as follows:

- We address the problem of customer reviews summarization. We present the problem as the tag generation problem.
- We propose a novel approach to find useful tags by ranking and refining the generated tags.
- We investigate the performance of tag generation on reviews of 12 products.
- We categorize products into popular products and cold products and investigate the tag generation performance.

The rest of the chapter is structured as follows. Section 5.2 details the problem definition and explains the proposed approach for unsupervised tag generation from reviews. The dataset used, experimental setup, and evaluation is discussed in Section 5.3. We conclude this chapter in Section 5.4.

## 5.2 Tagging Product Reviews

In this section, we present our unsupervised approach to summarize product reviews by generating top- $k$  tags. We give the problem definition in Section 3.1 and our solution in the remaining three subsections.

### 5.2.1 Problem Definition

There are many factors that determine the nature of reviews that a product gets. For example, a popular product, which is available since a significant time, may have many reviews, some of which could also be very detailed. On the other hand, a new product or a low rated product may have only few reviews, most of which will be very short. Although for popular products, the reviews may explain the pros-and-cons of different aspects of the product; for low rated products, most of the reviews will be short with an overall negative sentiment. To tag these different classes of product reviews, we propose the following two tag generation problems.

**Problem 1 Tagging Popular Products:** Let  $P$  be a product and  $R_P = \{r_1, r_2, r_3, \dots, r_n\}$  be the set of customer reviews of the product  $P$ , where each review  $r_i$  consists of a customer’s feedback and an overall rating on the product  $P$ . Our task is to generate top- $k$  tags  $T = \{t_1, t_2, \dots, t_k\}$  for the review set  $R_P$ .

Popular products have many reviews. Many of these reviews have sizable content and has a high rating for the product. They contain experience of users about the product and its comparison with other products. In general, a popular product has hundreds of reviews and the average length of each review is around 1500 characters.

Since popular products have many reviews and lot of content, if we use existing topic labeling methods, such as LDA [14], TNG [161], etc., to generate the tags, then we will get many topic labels, all of which we cannot show to users. Moreover, many of these labels may not be suitable for human reading. For example, they

would generate tags such as *years ago*, *level camera*, *camera takes*, etc. To reduce the number of tags, we select top-k tags that summarize all the reviews. To make it more presentable and informative to users, we process the top-k tags using NLP syntactic rules, as described in Problem 3.

**Problem 2 Tagging Cold Products:** Let  $P$  be a product that has very few reviews or has mostly low ratings, and  $R_P = \{r_1, r_2, r_3, \dots, r_n\}$  be the set of customer reviews of the product  $P$ , where each review  $r_i$  consists of a customer’s feedback and an overall rating on the product  $P$ . Our task is to generate top- $k$  tags  $T = \{t_1, t_2, \dots, t_k\}$  for the review set  $R_P$ .

In contrast to popular products, there are many cold products, such as low rated products and new products, which have less than 15 reviews with an average length of 500 characters per review. These products suffer from the problem of *cold start* due to lack of review content. In this chapter, we are not targeting new products as many of them do not have reviews or rating information. We are targeting low rated products that have few reviews, which are mostly negative. Low rated products are not good quality product; thus most of the reviews contain customers’ discontent with the product. Such reviews are generally short and may not even contain much information about the product. Due to these reasons, it is hard to generate tags for them.

Existing topic-generation methods do not work well when we have such a small amount of data. For example, consider a low-rated product review: ‘It is a complete waste of time and money’. Since the review is short and does not even contain much of product-related information, it is hard to generate candidate tags. We propose a tag generation method for such products by using tags from popular products.

**Problem 3 Tag Refinement:** Given the top-k generated tags for a product  $P$ , parse the reviews and use NLP syntactic rules based on the POS patterns to make the tags more readable and informative through word transformation and by including popular

customer sentiment.

Topic modeling algorithms are effective in finding noun forms of tags. However, such tags may not capture the sentiment terms associated the tags and thus may not give sufficient semantic information. For example, the tag *lens cap* represents an aspect only, but does not provide the opinion of customers associated with the aspect. Providing the associated sentiment *loose* to form the tag *loose lens cap* makes the tag more meaningful. Sometimes, the grammatical structure of the tags makes it difficult to understand. For example, the tag *focusing low light* does not make much sense. We can make it meaningful by rewriting as *low light focusing*. Occasionally, the tags may fail to capture the right context. For instance, the tags *good picture* and *big lens* are entirely different from the tags *not good picture* and *overly big lens* respectively. We make the tags more readable by adding popular sentiment words from the reviews and changing the grammatical structure of the tags based on NLP syntactic rules.

## 5.2.2 System Architecture

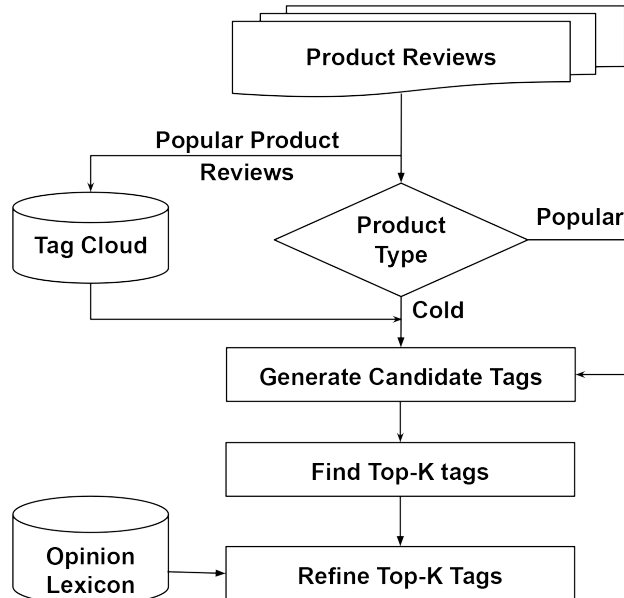


Figure 5.3: System Architecture.



Figure. 5.3 shows the architectural overview of our proposed review tagging system. Given a product, the system takes user reviews as input and generates top-k product relevant tags as output. The reviews undergo the following steps: (1) generating candidate tags, (2) ranking the candidate tags, and (3) refining the top-k tags using syntactic rules. Unlike existing supervised tag recommendation systems [10,19], this chapter presents an unsupervised method to generate tags for customer reviews. The details are explained in the following subsections.

### 5.2.3 Generating Candidate Tags

User-generated reviews are generally noisy in nature with many grammatical and spelling mistakes. Such faulty texts may generate unwanted tags and we may also miss good tags. Since we do not want such distorted texts to hinder the tag generation process, we clean review texts using the pre-processing steps mentioned in Section 2.2.2.

After refining the reviews of noisy text, the next step is to generate candidate tags. Customers usually do not provide any topic marker while writing reviews. Topic modeling methods based on bag-of-words representations are commonly used to identify key terms from text documents [31,44,155,161]. We consider each review as a document and adopt a bag-of-words representation of the reviews by extracting n-grams with varying sizes ( $n=1, 2, 3$ ) from the corpus. Using the n-grams, we extract candidate tags by applying three popular topic modeling techniques, namely popularity, TF-IDF and Topical n-gram (TNG) [161]. The techniques are explained as follows:

**Popularity:** Popular tags are the n-grams that frequently appear in customer reviews. We use term frequency based popularity algorithm [155] to find candidate

tags. Popularity is defined as follows:

$$Pop(q) = \frac{f_q}{max_k f_k} \quad (5.1)$$

where  $q$  is the given tag,  $f_q$  is the number of times the tag  $q$  appears in all reviews, and  $max_k f_k$  is the count of the most popular tag across all reviews. A high popularity score indicates that the tag is highly relevant as many times it has been mentioned in the reviews.

**TF-IDF:** To reduce the significance of commonly appearing tags in favor of more unique ones, we use TF-IDF score of tags, which is defined as:

$$TF-IDF(q) = Pop(q) * \log \left( \frac{N}{n_q} \right) \quad (5.2)$$

where  $N$  is the total number of reviews of a product,  $n_q$  is the number of reviews with tag  $q$  in it. TF-IDF score has two terms: the first term represents the term frequency (TF) and the second term represents the Inverse Term Frequency (IDF).

**Topical n-gram (TNG):** TNG [161] is a probabilistic topic modelling algorithm that extends Latent Dirichlet Allocation (LDA) [14] to model topics from documents. Unlike LDA, TNG can model words as well as phrases. TNG model is able to decide whether a bigram is a phrase or not according to context. For example, the bigram *good picture* carries a special meaning with respect to the topic *camera*, whereas the phrase *good news* does not. TNG does this by adopting a joint distribution  $P(t,b|x)$  and has the distributional presumptions:

$$x_i | x_{i-1}, t_i, b_i = 1, \theta \sim Discrete(\theta_{t_i}) \quad (5.3)$$

$$x_i | x_{i-1}, t_i, b_i = 0, \phi \sim Discrete(\phi_{t_i, x_{i-1}}) \quad (5.4)$$

$$b_i|x_{i-1}, t_{i-1}, \gamma \sim \text{Bernoulli}(\pi_{t_{i-1}}, x_{i-1}) \quad (5.5)$$

where each document  $d$  is expressed as a mixture of latent topics  $t_i$ 's. Each topic  $t$  is expressed as a mixture of terms  $x$ 's determined by a multinomial distribution  $\theta$ .  $b_i$  is a binary variable that indicates whether the term  $x$  is affected by the preceding term or is the beginning of a new n-gram starting at the  $i^{\text{th}}$  position. If  $b_i = 1$ , then the term  $x_i$  is not affected by the previous word and is considered to be the start of a new n-gram.  $b_i$  divides a collection into sequential non-intersecting n-grams of different sizes.  $\gamma_{t,x}$  and  $\phi_{t,x}$  are distributions equipped with conjugate prior distributions:

$$\gamma_{t,x} \sim \text{Beta}(\lambda) \quad (5.6)$$

$$\phi_{t,x} \sim \text{Dirichlet}(\delta) \quad (5.7)$$

where  $\gamma$  and  $\delta$  are some hyperparameters.

The above tag generation algorithms favor frequently occurring n-grams as candidate tags. Candidate tags generated by popularity and TF-IDF are mostly unigrams as they occur more frequently than phrases. Unlike popularity and TF-IDF, TNG takes the sequential nature of text into consideration to generate words and phrases as topics. As phrases bear more meaningful information than words, TNG gives better tags compared to popularity and TF-IDF. We consider the tags generated from TNG as candidate tags for further processing.

Due to the limited availability of reviews in cold products, TNG is not able to generate good key terms as tags. Many of the irrelevant words get generated as tags. We do not want to show irrelevant words as tags. To find the useful tags from cold product reviews, we leverage popular product reviews having a similar domain. We approach this as a transfer learning problem.

Candidate tags of popular products contain useful tags that are commonly used by customers while writing reviews such as *picture quality*, *shutter speed*, *battery life*,

and *touch screen*. So, for products having a similar domain, the same set of frequent aspects would be present in popular as well as cold products. Although such useful tags are also present in cold products, TNG is not able to detect such tags due to the sparsity of reviews. We leverage the candidate tags from reviews of popular products to enhance the tag generation of cold products by discovering useful tags.

After applying TNG, we collect the top 50 candidate tags containing aspect information from each popular product belonging to similar domain and use it to form a tag cloud ( $C$ ). From  $C$ , we use the aspect information to find tags information for the cold products. For a cold product, we parse through each review and find the tags that are present in  $C$ .  $C$  is also used to find the review-tags dictionary ( $RT$ ) for each review, which stores the tags present in each review.  $RT$  is in turn used to obtain the top-k most useful tag set.

#### 5.2.4 Ranking Candidate Tags

Some reviewers write long reviews, while others write short reviews. TNG gives more importance to long reviews as they often have repeated key terms. But, in short reviews, key terms generally appear once only and related terms are also absent. Although useful tags may be present in short reviews, frequent tags from long reviews tend to dominate the result of TNG. To overcome this bias, we take into consideration the number of reviews where a tag is present and focus on finding the useful tags instead of frequent ones. A tag that is present in many reviews is considered more useful than a tag that is present in few reviews.

We propose a greedy approach to rank the candidate tags obtained using TNG to find the tags that are preferred by many users. We use coverage to find useful tags, i.e., tags that cover a maximum number of reviews. Coverage of a tag  $t$  is defined as:

$$Cov(t) = \frac{n_t}{N} \quad (5.8)$$

where  $n_t$  is the number of reviews that contain the tag  $t$  and  $N$  is the total number of reviews.

---

**Algorithm 4** Tag Usefulness Ranking TUR( $P, P_p, R_p$ )

---

**Input:**  $P$ : Product  
 $P_p$ : Set of popular products belonging to domain  $D = \{p_1, p_2, \dots, p_l\}$   
 $R_p$ :  $\{r_1, r_2, \dots, a_n\}$  = Customer reviews of product  $P$   
 $T_C$ :  $\{tc_1, tc_2, \dots, tc_n\}$  = Candidate tags for popular products  $P_p$

**Output:**  $C = \{c_1, c_2, \dots, c_m\}$  = Tag cloud = set of candidate tags of  $P_p$   
 $T$ :  $\{t_1, t_2, \dots, t_k\}$  = Top-k relevant tags generated from the reviews

**Method:**

- 1:  $C \leftarrow \{\}, T_C \leftarrow \[],$  Max coverage tags,  $T \leftarrow \phi,$  Review-tags dictionary,  $RT \leftarrow \[]$
- 2: **for all**  $p \in P_p$  **do**
- 3:      $T_C \leftarrow \text{RANKCANDIDATETAGS}(p)$
- 4:     **for all**  $c_i \in T_C$  **do**
- 5:         **if**  $c_i \notin C$  **then**
- 6:              $C.\text{ADD}(c_i)$
- 7:         **end if**
- 8:     **end for**
- 9: **end for**
- 10: **if**  $P \in P_p$  **then**
- 11:     **for all**  $r_i \in R_p$  **do**
- 12:          $RT[r_i] \leftarrow \text{EXTRACTTAGS}(r_i, T_C)$
- 13:     **end for**
- 14: **else**
- 15:     **for all**  $r_i \in R_p$  **do**
- 16:          $RT[r_i] \leftarrow \text{EXTRACTTAGS}(r_i, C)$
- 17:     **end for**
- 18: **end if**
- 19: **while**  $\text{len}(T) \leq k$  or  $R_p \neq \Phi$  **do**
- 20:      $t \leftarrow \arg \max_{c_i} \{|\text{cov}(c_i, RT)|\}$
- 21:      $T.\text{APPEND}(t)$
- 22:      $R_t \leftarrow \text{ASSOCIATIONSET}(t, RT)$
- 23:      $T_C \leftarrow T_C - t$
- 24:      $R_p \leftarrow R_p - R_t$
- 25: **end while**
- 26: **return**  $T$

---

Due to the limited availability of reviews in cold products, TNG is not able to generate good key terms as tags. Many of the irrelevant words get generated as tags. We do not want to show irrelevant words as tags. To find the useful tags from cold

product reviews, we leverage popular product reviews having a similar domain. We approach this as a transfer learning problem. Candidate tags of popular products contain useful tags that are commonly used by customers while writing reviews such as *picture quality*, *shutter speed*, *battery life*, and *touch screen*. Although such useful tags are also present in cold products, TNG is not able to detect such tags due to the sparsity of reviews. We leverage the candidate tags from reviews of popular products to enhance the tag generation of cold products by discovering useful tags. After applying TNG, we form a tag cloud ( $C$ ) by collecting the top 50 candidate tags from each popular product belonging to the same domain.  $C$  is used to find useful tags for cold products. For a cold product, we parse through each review and find the tags that are present in  $C$ .  $C$  is also used to find the review-tags dictionary ( $RT$ ) for each review, which stores the tags present in each review.  $RT$  is in turn used to obtain the top-k most useful tag set. The tag ranking algorithm is given in Algorithm 4.

Given a product  $P$  and a set of reviews  $R_P$ ,  $TUR$  follows a greedy approach to find the most useful tags. Step 1 initializes the tag cloud  $C$ , the ranked list of candidate tags  $T_C$ , the useful tag set  $T$  and the review-tags dictionary  $RT$ . Steps 2–9 uses the candidate tags of popular products to get the tag cloud  $C$ . For each product  $p$ , step 3 finds the top-50 tags that have the highest TNG topic relevance score and store it in  $T_C$ . Then, for each tag  $c_i$  in  $T_C$ , steps 4–8 adds  $c_i$  to the tag cloud  $C$  if it is already not present. We leverage  $C$  to find tags from cold product reviews. Step 10 checks if the product  $P$  is present in the popular product set  $P_p$ .  $P_p$  contains a collection of popular products obtained after analyzing the average product rating of each product. For each review  $r_i$ , steps 11–13 use the candidate tags  $T_C$  to extract the relevant tags present in  $r_i$  and store the tags in the Review-Tag dictionary  $RT$ . While for a cold product review, step 15–17 uses the tag cloud  $C$  to extract review specific tags  $RT$ . Using the relevant tags present in  $RT$ , steps 19–25 iterate until we get either the top-k useful tags or all the reviews are covered by the tags. At each iteration, it finds

and appends the most helpful tag from  $T_C$  to  $T$ , which covers the maximum number of uncovered reviews at that moment. Step 20 finds the tag  $t$  that has maximum coverage and adds it to the tag set  $T$  in step 21. Step 22 computes the association set ( $R_t$ ) of  $t$  to find the reviews that are covered by it. Steps 23–24 remove  $t$  and the associated reviews ( $R_t$ ) from  $T_C$  and  $R_P$  respectively. Finally, step 26 returns the list of top-k most useful tags ( $T$ ) after greedily finding the tags.

### 5.2.5 Refining Top-K Tags using Syntactic Rules

The previous step generates a list of core phrase tags from the reviews. Although the generated top-k tags are useful, some of the generated tags do not give sufficient semantic information due to missing sentiment terms. In general, popular products are associated with positive sentiment words, while many of the cold products are associated with neutral or negative sentiment words. To find the sentiment terms, we analyze the Part-of-Speech (POS) patterns that are generally associated with phrases of opinionated text using the Stanford NLP dependency parser and develop five syntactic rules to enhance the usefulness of the tags. We apply these syntactic rules after parsing the reviews and also finding the top-k useful tags  $T$ . The tag phrases obtained after applying the syntactic rules are found to be more readable and meaningful. Table 5.1 presents the syntactic rules.

#### Opinion Rule

Opinion words are generally adjectives and they provide useful subjective information. So, whenever an adjective is present before a noun tag phrase (NP), we check if it belongs to an opinion word by using an opinion lexicon [73] and note its polarity. When an opinion word (JJ) is present before a noun phrase tag (NP) in reviews, the opinion word is considered as a part of the tag phrase. However, different customers may use different opinion words of varying polarity for the same tag. Consider an

Rule	POS Pattern	Description	Example
Opinion rule	JJ+NP	Opinion word (JJ) is present before tag phrase (NP) in reviews	amazing picture quality, good battery life, loose lens cap, better picture quality, best camera, great optical zoom, powerful features, fantastic raw image mode, small display
Negation rule	NG+NP	Negation word (NG) is present before tag phrase (TP)	not good picture, not auto focus, no tiff format, never fails, no better camera
Intensifier rule	RB+JP	Adverb intensifier (RB) is present before adjective tag phrase (JP)	extremely flexible lens, easily operable, extremely satisfied, really long zoom, very pleased, too small display, quite bulky, very long zoom, moderately bright flash
Gerund transformation rule	VBG+JP → JP+ VBG	If a verb (VBG) is present before an adjective tag phrase(JP), then put it at the end of the tag	processing raw image → raw image processing
			focusing low light → low light focusing
			zooming lens → lens zooming
			transferring image → image transferring
			shooting portrait → portrait shooting
			enlarging easily → easily enlarging
			charging battery → battery charging
			shaking less → less shaking
looking ugly → ugly looking			
Opinion inclusion rule	NP/JP+VBZ+JJ → JJ+NP	If an opinion word (JJ) is present at the end of a noun/adjective phrase tag with a verb (VBZ) in between, place the opinion word (JJ) at the beginning of the tag and remove the verb (VBZ)	optical zoom works great → great optical zoom
			lens cap is very loose → loose lens cap
			camera is sharp → sharp camera
			optical zoom works great → great optical zoom
			strap is horrible → horrible strap
			quality is amazing → quality
			settings provides excellent → excellent settings

Table 5.1: POS rules used to enhance the meaning of tags.



example, where multiple customers associate the tag *picture quality* with different opinion words such as *amazing*, *excellent*, *good*, *poor*, *bad*, etc. For a given tag, generating multiple tag phrases with varying opinion words would lead to several ambiguous or redundant tags such as *amazing picture quality*, *good picture quality*, and *poor picture quality*. We need to find the right polarity and the accurate opinion word associated with the tag. To this end, we count the number of positive and negative opinion words associated with the tag and select the polarity that has maximum count as the polarity of the tag. Among the same polarity opinion words, we choose the most frequent word as the opinion word associated with the tag. In the above example, the tag *picture quality* is associated with more positive opinions. Among the positive opinion words, considering *amazing* appears in more number of reviews compared to *excellent* and *good*, we refine the tag as *amazing picture quality*.

### **Negation Rule**

Sometimes, tags are associated with negation words, thereby reversing the meaning of the tags. For instance, the meaning of the tag phrase *good picture* is reversed when the negation word *not* is present before it, forming the tag phrase *not good picture*. Negation rule is used to handle such cases. We add negation word to a tag phrase only if the majority of the customer talking about the tag phrase are negative about it. If a negation word appears before a tag phrase, we count the number of times the tag phrase is associated with negation word in the reviews. If it is more than half of the number of occurrence of the tag phrase, we associate a negation word *not* with the corresponding tag phrase.

### **Intensifier Rule**

Adverbs (RB) intensifies an adjective phrase (JP) when present before the phrase. So, whenever an adverb is present before an adjective phrase tag, intensifier rule is used to

add adverbs (RB) present before an adjective phrase tag (JP). Doing so complements the sentiment associated with the phrase tag. For example, the tags *overly big lens*, and *fast shutter speed* conveys more information than the tags *big lens*, and *shutter speed* respectively.

### **Gerund Transformation Rule**

Gerunds in some of the tag phrases appear in distorted forms such as *processing raw image*, *focusing low light*, etc. These phrases have a gerund (VBG) followed by an adjective phrase (JP). We observe that if the gerund appears after the adjective phrase (JP + VBG), the phrases are more readable. Gerund transformation rule is used to find the tag phrases that contain gerunds followed by adjective phrase (VBG + JP). Whenever such a phrase occurs, the gerund is shuffled at the end of the phrase (JP + VBG) as shown in Table 5.1.

### **Opinion Inclusion Rule**

In some cases, we observe that the opinion words (JJ) appear after tag phrases (NP/JP) with a verb (VBZ) in between (NP/JP + VBZ + JJ) such as ‘lens cap is loose’, ‘optical zoom works great’, etc. Opinion inclusion rule adds opinions words having such a pattern to the beginning of the phrase tag and discard the verb (VBZ). For example, the phrase ‘lens cap is loose’ is transformed into ‘loose lens cap’.

## **5.3 Evaluation**

In this section, we present the evaluation of our proposed approach on tag recommendation for product reviews. We first describe the dataset and experimental setup. Then, we present the evaluations and interpretations of our observations.

### 5.3.1 Experimental Setup

#### Dataset

To perform our experiments, we consider product reviews from Amazon. We use the Amazon review dataset<sup>1</sup>, which is one of the most popular datasets in review analysis research [61, 98, 178]. The dataset comprises of consumer experiences, such as review, rating, and helpfulness votes on products along with product information, such as descriptions, brand, price, etc., on 24 product domains for the period May 1996 - July 2014. Every product domain has multiple sub-domains. We choose the ‘Camera and its accessories’ domain and conduct experiments on the consumer reviews of 12 electronics products from 5 sub-domains, namely 8 digital cameras, 1 digital frame, 1 router, 1 speedlight and 1 wireless trigger.

#### Product Review Selection

We observe that the nature of reviews varies from popular products to cold products. Popular products have significantly more reviews, which are also quite detailed, compared to the cold products. From the dataset, we identify popular products and cold products based on the average overall rating of the product and the number of reviews. For the experiment, a product with an average overall rating of four or five and a huge number of reviews is considered as a popular product. On the other hand, a product with an average overall rating of one or two and with less number of reviews is considered a cold product. To ensure that the reviews are useful, we select only those products that have reviews with average sentence length greater than 5 and at least two helpfulness votes. We choose the top six popular products and the top six cold products with the highest number of reviews. The entire dataset contains 13218 sentences. The statistics of the dataset is shown in Table 5.2.

From Table 5.2, we can infer that the number of reviews for popular products is

---

<sup>1</sup><http://jmcauley.ucsd.edu/data/amazon/>

Product name	No. of reviews	Product type
Digital Camera 1	137	P
Digital Camera 2	137	P
Digital Camera 3	134	P
Camera Flash	403	P
Digital Frame	110	P
Router	205	P
Digital Camera 4	17	C
Digital Camera 5	15	C
Digital Camera 6	18	C
Digital Camera 7	8	C
Digital Camera 8	10	C
Wireless Trigger	14	C

Table 5.2: Dataset Statistics.

much more compared to cold products. This is because customers do not prefer to buy low rated products and also do not spend much time to describe their bad experiences. Also, the average number of sentences for high-rated products is much more than the low-rated products. This indicates that while writing reviews, customers tend to describe more about their likings of high-rated products and very little about low-rated products.

### Tag Relevance Assessment

To determine the relevance of the generated tags, we asked five Ph.D. students to label the generated tags. All ten students have proper knowledge of electronic products. We select a set of reviews that cover the generated tags and are provided to the students for reference. Then, the scholars are asked to grade each of the generated tag based on the following five-point relevance scale:

- **non-relevant (score 0)**: The tag is distorted and has no association with the product (e.g., view, stars, ghost hunting, awkward place).
- **ordinary (score 1)**: Although the tag is well-formed, the association with the product is ambiguous (e.g., accessory genie, canon line) or partial (e.g., frame, shutter).
- **marginally relevant (score 2)**: The tag is well-formed and is a secondary product aspect without sentiment (e.g., camera settings, extra battery, wireless feature).
- **relevant (score 3)**: The tag is well-formed and is a primary product aspect without sentiment (e.g., white balance, viewfinder, power button, touch screen).
- **highly relevant (score 4)**: The tag is well-formed and is either a primary or secondary feature with a sentiment (e.g., awesome video, compact design, fast focus, flash works great).

After the students came up with their own relevance scale, any ambiguity in tag scoring was resolved together through discussion and mutual agreement. We observe that around 31% of the tags are non-relevant or ordinary, 25% of the tags are either relevant or highly relevant, and 44% of the tags are marginally relevant. The high percentage of non-relevant and ordinary tags are expected as the baseline algorithms generate mostly bad tags and tags at the lower position of the rankings are not much relevant to the products. The scores of the tag relevance assessment are considered as ground truth labels for evaluation of tag relevance and tag ranking. We observe that, for all the products, the proportion of non-relevant and ordinary tags is lower than that of relevant and highly relevant tags. This supports the suitability of the chosen approaches for generating useful tags.

## Baseline Methods

We compare the performance of the proposed Tag Usefulness Ranking (TUR) algorithm used in Tagging Product Review (TPR) system with the three topic modeling techniques described in Section 5.2, namely, popularity, TF-IDF, and TNG, taken as baselines. TPR uses Tag Usefulness Ranking (TUR) algorithm on the review datasets to find useful tags from the product reviews. The tags are selected such so as to maximize the reach of reviews. TUR follows an iterative and greedy approach for selecting the top  $k$  tags by choosing the tags that are present in most yet uncovered reviews in each iteration.

### 5.3.2 Evaluation Metric

#### Effectiveness of Tag Ranking

We evaluate the effectiveness of tag ranking using the evaluation metric Normalized Discounted Cumulative Gain at top- $k$  (NDCG@ $k$ ) [59]. NDCG@ $k$  of a ranked list of  $k$  tags is defined as follows:

$$NDCG@k = \frac{1}{iDCG} \sum_{i=1}^k \frac{2^{t(i)} - 1}{\log(1 + i)} \quad (5.9)$$

where  $t(i)$  is the scoring function that indicates the score assigned to the tag at the  $i^{th}$  location,  $iDCG$  is the DCG value of the top- $k$  tags obtained from an ideal ranking. The use of  $iDCG$  normalizes the value of NDCG@ $k$  within a range of  $[0, 1]$ . The value of the scoring function  $t(i)$  of a tag  $i$  is obtained from the tag relevance assessment score of the corresponding tag. NDCG@ $k$  favors the ranking that ranks the most relevant tags at the top. In our experiments, we consider different values of  $k$  ranging between 1 to 50.

## Accuracy of Tag Generation

The accuracy of tag refinement is measured using precision. Precision gives the percentage of correct tags present in the generated tag list after refinement. Precision is defined as follows:

$$P = \frac{\# \text{ correctly generated tag}}{\# \text{ generated tags}} \quad (5.10)$$

We use the tag relevance scores for ground truth labels. Tags with a relevance score of 4 or 5 are considered as correct tags while the rest are considered as incorrect tags.

### 5.3.3 Results

#### Evaluation of Tagging Popular Products

This section we present the evaluation of popular products. For cold products, it is presented in the following subsection.

Fig. 5.4 shows the comparison between tag ranking algorithms for popular products using NDCG. The NDCG score of popularity and TF-IDF is comparable, with each having an average NDCG score of 46.16% and 50.42% respectively. Compared to them, TNG gives a higher NDCG score, with an average of 64.62%, improving the other two baselines by over 18% and 14% respectively.

For *On Camera Flash* and *Router*, the NDCG score of popularity and TF-IDF methods are comparatively lower than the rest of the products. As these two products have significantly more number of reviews compared to the other four products, the presence of noisy tags gets amplified. However, the presence of a large corpus complements our TUR approach in finding better coverage, giving a relatively high NDCG score compared to other products. Digital frame has the least number of reviews among popular products. Hence the performance of popularity, TF-IDF and TNG are comparable due to the lesser amount of noisy text.

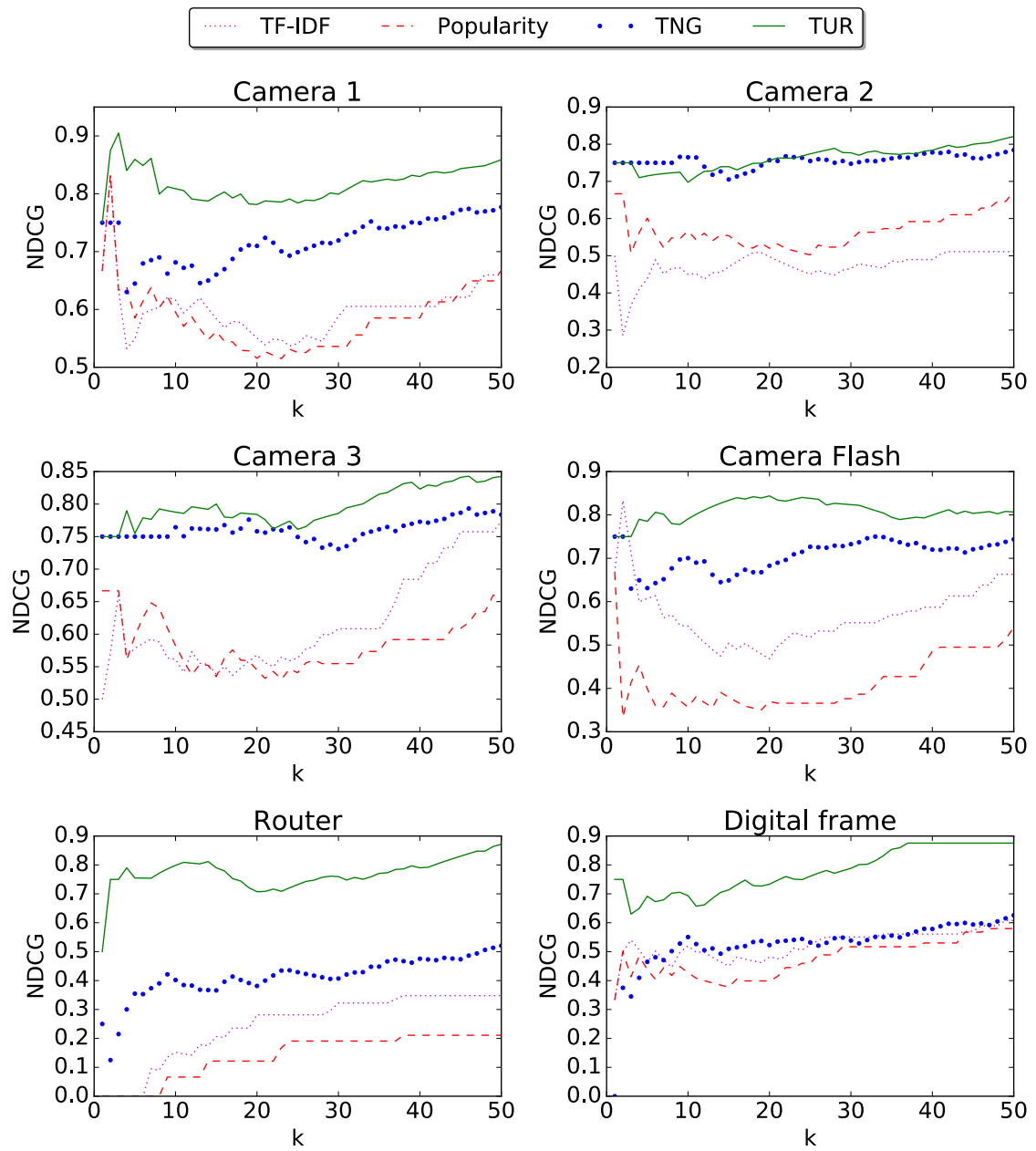


Figure 5.4: Evaluation of top-k tags for popular products using the metric  $NDCG@k$ .



For two of the products, namely *Router* and *Digital Frame*, there is a sharp rise in NDCG score of popularity and TF-IDF methods when the value of ‘k’ increases initially. This is because they largely rely on the frequency of tags in the corpus. As words occur more frequently than phrases, most of the top-k tags for popularity and TF-IDF methods are trivial words. Moreover, they segregate the corpus into small independent components, such as unigrams, bigrams, and trigrams, and in the process, the contextual information of the text is lost. This causes frequently occurring unrelated words like *time*, and *money* to appear as top tags. As a result, these two methods give very low NDCG score when the value of ‘k’ is small. As ‘k’ increases, relevant tags appear in the list, and hence NDCG score rises. Among the three baselines, TNG gives the highest average NDCG because it uses the contextual information of the text in the corpus to generate meaningful words and phrases as tags. TNG’s ability to generate phrases as topics also improves its NDCG score.

Although TNG gives meaningful tags, many of the tags are irrelevant to the product, such as *poor weather*, *tech savvy*, *bottom line*, *bright light*, etc. The presence of such tags at the top lowers the relevance score. TPR improves the ranking of TNG by filtering out such irrelevant tags. TPR follows an iterative and greedy approach to find tags that cover the most number of reviews instead of depending solely on the tag frequency in the corpus. TPR improves the relevance of generated tags by finding tags that are most talked about by customers. Since most of the tags by TPR are relevant, the value of NDCG does not fluctuate much with varying values of ‘k’. Applying syntactic rules to the generated tags further improves the readability of the tags. This, in turn, increases the tag relevance score, thereby improving the NDCG score. For all popular products, on average TUR outperforms the popularity, TF-IDF and TNG based approaches by over 32%, 28%, and 14% respectively. Thus from the results, we can infer that our proposed TUR algorithm can better identify important opinionated aspects from customer reviews compared to standard topic

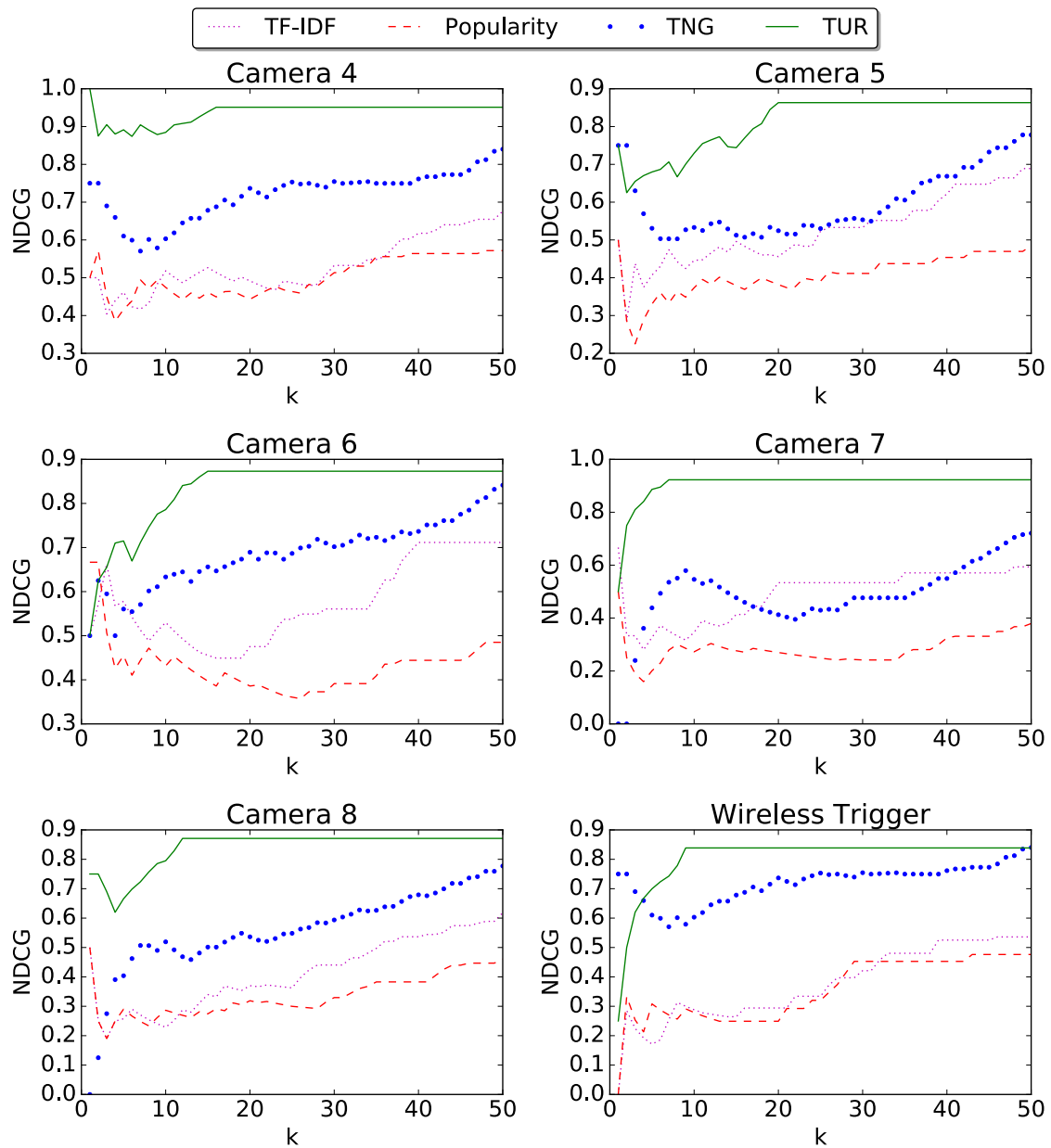


Figure 5.5: Evaluation of top-k tags for cold products using the metric NDCG@k.

modeling algorithms.

## Evaluation of Tagging Cold Products

Fig. 5.5 shows the result of tag relevance evaluation for cold products. For cold products, our proposed TUR approach outperforms the three baseline approaches by a huge margin. For the six cold products, the average NDCG of TUR is 85.56% compared to the average NDCG of 38.57%, 48.57%, and 62.90% for popularity, TF-IDF and TNG respectively. In other words, our approach outperforms the baselines by over 47%, 37%, and 23% respectively. Our approach is very helpful compared to baseline methods because majority of the e-commerce products have few reviews, comparable to cold products.

The big difference in NDCG for cold products is due to the use of tag cloud-based transfer learning. Since cold products have small review corpus, the frequency of key terms is very less. Hence popularity and TF-IDF methods perform poorly by picking up frequently occurring irrelevant terms such as *buy*, *design*, and *money* as tags. The lack of review text also affects the performance of TNG in generating key topics. Due to the absence of sufficient contextual information, the tags generated by TNG are less relevant (e.g., *worth noting*, *bottom line*, etc.). Compared to popular products, the relevance score of TNG drops considerably for low values of ‘k’.

Our proposed approach overcomes the lack of contextual information by leveraging transfer learning to identify good tags. The use of tag cloud from popular products to identify tags from cold product reviews compensates the lack of contextual information. Due to the lack of review text, only a few tags are present in cold product reviews. Therefore, the relevance curve of TUR increases at the beginning and continues as a horizontal line with increasing k.

Product name	Product type	Popularity	TF-IDF	TNG	TUR
Camera 1	P	0.32	0.36	0.84	0.96
Camera 2	P	0.34	0.32	0.84	0.90
Camera 3	P	0.34	0.42	0.82	0.90
Camera flash	P	0.24	0.30	0.82	0.90
Digital frame	P	0.18	0.24	0.66	0.84
Router	P	0.06	0.14	0.46	0.82
Camera 4	C	0.34	0.40	0.40	0.88
Camera 5	C	0.30	0.38	0.50	0.88
Camera 6	C	0.26	0.28	0.74	0.98
Camera 7	C	0.28	0.22	0.50	1.00
Camera 8	C	0.36	0.38	0.66	0.88
Wireless trigger	C	0.20	0.22	0.50	0.88

Table 5.3: Evaluation of tag refinement using the precision of top 50 tags.

## Evaluation of Tag Refinement

We use precision metric to evaluate the effect of tag refinement. Due to information overload and display constraints, e-commerce sites always show less than 10-15 tags per product. However, to evaluate the accuracy of tag generation algorithms for different values of  $k$ , we considered maximum  $k=50$  tags. The effect of tag refinement is shown in Table 5.3. From Table 5.3, it can be observed that our proposed approach significantly outperforms the other baseline approaches both for popular and cold products. For both the type of products, popularity and TF-IDF methods give low accuracy of 26.83% and 30.50% respectively. TNG gives an average precision of 64.50%, which is better than that of popularity and TF-IDF. TNG performs better for popular products compared to cold products. On average, TNG gives 25% higher precision for popular products than cold products. This is because TNG depends on the size of the corpus and the number of reviews for popular products is much

more than cold products. On the other hand, our proposed approach (TUR) gives a consistently good result with an average precision of 90.16%, which is 25.66% higher than TNG. By ranking the tags based on usefulness and refining the tags based on syntactic rules, the average precision improves for both popular and cold products. In the case of cold products, we observe that TUR gives very high precision. This is because only a limited number of tags are present in cold product reviews. For instance, even though *Camera 7* has only 8 reviews, our approach can find 7 relevant tags giving a precision of 100%.

## 5.4 Conclusion

This chapter presents a novel unsupervised approach for automatic tag generation from product reviews. We construct a Tagging Product Review (TPR) system using topic modeling and NLP syntactic rules. We use TNG to find and rank tag phrases and then apply our *TUR* algorithm to get the important tags that cover most of the reviews. The use of topic modeling enables us to generate tags without using external data. We also introduce five syntactic rules to enhance the usefulness of the tags. Generating tags for cold products is challenging due to the lack of contextual information. We overcome this shortcoming by using tag clouds that are generated from popular product reviews. Our evaluation of twelve datasets shows that our approach can generate and rank tags significantly better than existing approaches, whose tags give either partial or no information about the product. Also, we develop a greedy approach to rank and refine candidate tags. Our approach is applicable to any domain which has textual content such as movie reviews, community forums, social media sites, and debate posts as it does not require any pre-trained information. In our future work, we plan to apply our approach to social media sites for summarizing user discussions.

# Chapter 6

## Unsupervised Stance Detection in Comparative Reviews

### 6.1 Introduction

Comparative reviews are becoming more and more popular with the advancement of e-commerce technology. These reviews contain rich information that can be exploited in various services or decision-making processes. Many users turn to a variety of online platforms such as forums, blogs, social media sites, etc. to discuss and compare various competing products in the form of debates. Specially, understanding the users' opinion and their stance from these debates can provide valuable information in the decision-making process of governments, companies, and can help shape public opinion towards issues.

Stance detection is the process of discovering the standpoint of users in a comparative post. A debate consists of a topic of discussion, targets, and posts from various users. A target may be a product, technical topic, political side, ideological belief, social issue, etc. For example, consider the debate post:

*“Windows supports most of the games. Mac has a lot of software compatibility issue”.*

Here, the topic of discussion is “*Is Mac better than Windows?*”, and the targets are *Windows* and *Mac*. One can deduce that the author is writing favourably about Windows and drawbacks of Mac.

Stance detection incorporates a new dimension to other related areas like information retrieval, discourse analysis, fake news detection [114], and recommendation system. For example, by applying stance detection on a product debate, we can find out the product preference of users. Advertisements of related product can be displayed to users according to their stance. In this chapter, we study two-sided debate posts, where there are two opposing targets. The user may write about the good points of her preferred target or bad points about why she is against the other.

There are two main approaches for stance detection, namely: supervised and unsupervised. Most of the existing stance detection methods follow supervised approach [3, 74, 123, 184]. These methods are not scalable as for each domain, they require a large dataset to be collected and annotated for training stance classifiers. The advantage of unsupervised approaches is that they do not require any manual data labeling [93, 140, 146]. These unsupervised methods can easily adapt to a variety of domains.

Unsupervised methods typically use linguistic and syntactic dependency relations between targets and opinions. Most of the existing unsupervised methods focus on stance detection on tweets [74, 102]. Since tweets are very short (140 characters or less) and they contain useful target and opinion information in the form of hashtags and emoticons, it is relatively easy to identify targets and opinions compared to online debate posts, which are typically long and do not have markers such as hashtags. In this chapter, we focus on unsupervised stance detection on online debate posts, which are on average 600 characters long and have no tags. We extend the work of Somasundaran and Weibe [140], which performs unsupervised stance detection on long debate posts and does not require any marker information.

Stance detection is often misunderstood as sentiment classification. Nonetheless, stance detection is a more complex problem compared to sentiment classification, where the task is to classify the sentiment of a piece of text as positive, negative, or neutral. The main challenges in stance detection is to identify the referred target of each aspect. The sentiment of the text or aspects alone is not enough to determine the stance as debate posts do not contain sufficient information to determine the aspect-target preference. Sometimes, a text may not contain any target, and opinions may be expressed towards unrelated targets. To better understand the problem, consider the following two examples, comparing a review and a debate:

**Example 9** “*Canon is a very good camera. It has good picture quality, clear display, and long battery life.*”

**Example 10** “*Everyone knows that Windows is the better operating system because it has more softwares, more dev support, more everything. Also, Windows gives more freedom to users. Mac is like the retarded little brother who is too much pampered. It has Wi-Fi connection problem and lacks gaming support. You can use a Hackintosh if you really want Mac OS, and it will be much cheaper than buying from Apple.*”

Example 9 is an extract from a review of a camera product. Here, the user expresses her opinion about some of the aspects such as *picture quality*, *display*, and *battery life*, and all the aspects refer to a single target *camera*. The sentiment of the author can be summarized by aggregating the opinions about the aspects. On the other hand, Example 10 is of a debate post on the topic *Windows vs. Mac*. Here, the user writes about why she supports one target (Windows) over the other (Mac). Since the user writes about both the targets, it is hard to figure out which target each aspect (e.g., operating system, software, gaming, etc.) is supporting. Also, the aspect *brother* is not related to the debate targets. Normal sentiment analysis approach would only detect the sentiment of each aspects. We need to rely on some external sources, such



as a web corpus to extract aspect-target relation. Existing approaches need to crawl web pages related to a given debate topic to get the aspect-target relationship [140]. These approaches are computationally expensive and time-consuming. Besides, the accuracy of such approaches is limited by the availability of domain information from the web corpus. Even when using web search, we may have access to a limited number of pages, many of which may not be even relevant.

In this chapter, we propose the use of word embeddings to perform unsupervised stance detection. This enables us to do unsupervised stance detection without using a web corpus and helps us to extract relevant aspect-target pairs with a much higher precision compared to existing methods that use only syntactic dependency relations between targets and opinions.

Word embedding is a neural-network language modeling method used to represent text as dense vectors. It has been shown to encode precise semantic and syntactic word relationships and provide useful information about words co-occurrence [99]. We utilize this property of word embeddings to find aspects that are relevant to the debate targets. We also use it to find aspect-target preference. For example, using word embedding, one can find that the aspect *software* is more closely related to *Microsoft* than to *Mac*. This is true as Microsoft Operating Systems (OS) support more softwares than Mac OS.

After applying existing unsupervised methods for aspect extraction on our dataset, we found that almost 80% of the extracted aspects are irrelevant to the debate targets. This is because there are generic aspects which appear frequently in most of the debate posts such as *anyone*, *anything*, and *example*, or are not relevant to the debate targets such as *freedom*, *Hackintosh*, and *brother*. We use word embeddings to prune the aspects obtained using syntactic dependency relations. For this, we train a supervised classifier that uses both the similarity and difference score of the aspects and the target topics. The classifier gives an average accuracy of 84% on multiple datasets.



takes a two-sided debate post as input. The output is the standpoint of the user in the debate post. It performs the following steps to generate the output: (1) data cleansing, (2) extracting opinion words, (3) creating aspect-opinion pairs, (4) pruning irrelevant aspects using word embeddings, (5) creating aspect-polarity pairs, (6) creating aspect-target preference using word embeddings, and (7) detecting debate post stance using Integer Linear Programming (*ILP*). As it can be seen in Figure 6.1, unlike the work proposed by Somasundaran and Wiebe [140], our focus is on using word embeddings for finding aspect-target preference instead of creating a probability preference table, which requires an external web corpus. We also introduce three additional steps, namely, data cleansing, lexicon expansion, and aspect pruning, which differentiate our proposed approach from the existing one.

---

**Algorithm 5** DSCW( $D, T, O_L$ )

---

**Input:**  $D: \{d_1, d_2, d_3, \dots, d_n\}$  = Set of debate posts  
 $T: t_1, t_2$  = Two targets of the post  
 $O_L$ : Opinion lexicon  
**Output:**  $S = \{s_1, s_2, s_3, \dots, s_n\}$  = Stance of the debate posts  
**Method:**

- 1:  $CD \leftarrow \text{CLEANDATA}(D)$
  - 2:  $O \leftarrow \text{EXTRACTOPINIONWORDS}(CD, O_L)$
  - 3:  $AOP \leftarrow \text{CREATEASPECTOPINIONPAIRS}(CD, O)$
  - 4:  $A \leftarrow \text{PRUNEASPECTS}(WE(AOP), WE(T))$
  - 5:  $AP \leftarrow \text{CREATEASPECTPOLARITYPAIRS}(A, O_L)$
  - 6:  $AT \leftarrow \text{CLASSIFYASPECTTARGETPREFERENCE}(WE(A), WE(T))$
  - 7:  $S \leftarrow \text{DETECTSTANCE}(AP, AT)$
  - 8: return  $S$
- 

Algorithm 5 shows the step by step process to detect the stance of a debate post. Given a set of debate posts  $D$  on a topic, and debate targets  $T=\{t1, t2\}$ , Step 1 performs pre-processing to correct incomplete, noisy and inconsistent data such as contractions, URLs, stop words, and repeated letters. After cleansing the data, Step 2 extracts the opinion words ( $O$ ) present in the posts using an opinion lexicon ( $O_L$ ). For each opinion word in  $O$ , the associated aspects are extracted using Step 3. This

step generates many aspects that have little or no relevance to the debate targets. In Step 4, word embeddings ( $WE$ ) of the aspects and the targets are used to prune unrelated aspects and find a list of relevant aspects  $A$ . After getting the relevant aspects, Step 5 computes the aspect-polarity pairs  $AP$  from  $A$  and  $O_L$ . In Step 6, word embeddings are used to find the preferred target for each aspect ( $AT$ ). The use of word embeddings eliminates the need for collecting web corpus for aspect-target preference. Finally, the system uses Step 7 to compute a weighted score  $S_{ij}$  for each aspect-target pair. The algorithm uses  $ILP$  and  $S_{ij}$  to formulate the objective function. In Step 8, the stance of the user is assigned to the side that maximizes the objective function. These steps are further explained in the following subsections:

### 6.2.1 Data Cleansing

Users often do not follow proper grammatical rules and use short-forms while writing online comparative reviews. The syntactic rules used in unsupervised methods will not perform well if the sentences are not grammatically correct. Therefore, we pre-process the comparative review posts using steps mentioned in Section 2.2.2 to make it grammatically correct, and also clean noisy and inconsistent data.

### 6.2.2 Opinion Word Extraction

After cleaning and refining the post, the next step is to extract opinion words from the cleaned post. To address this issue, researchers turn to opinion lexicons [54, 57, 140], which is a compilation of opinion words along with the associated semantic polarity (e.g. *good* : +1; *bad* : -1; *really* : 0). If the opinion lexicon is small, we could lose out opinion words present in debate posts, which are in turn, used to detect aspects and find the stance of the post. In order to increase the coverage of opinion words across multiple domains, we combine multiple publicly available opinion lexicons [54, 103]. In addition, we also identify comparative adjectives (e.g., *better*) and superlative

adjectives (e.g., *best*) from posts using Stanford Part Of Speech (POS) tagger<sup>1</sup> and add them to our opinion lexicon ( $O_L$ ). These comparative words appear frequently in debate posts and express sentiments. The polarity associated with these words are obtained using SentiWordNet<sup>2</sup>. After merging all these lexicons, the size of our  $O_L$  is almost 1.7 times that of the opinion lexicon size used in [140].

### 6.2.3 Aspect-Opinion Pairing

This step identifies the aspects associated with opinion words extracted from debate posts. We use the unsupervised rule-based method proposed by Somasundaran and Wiebe [140] to extract aspect-opinion pairs (AOP). They leverage the property that opinions are expressed on aspects of the targets. Their method identifies dependency relations between aspects and opinions using syntactic dependency trees generated by the Stanford parser<sup>3</sup>. Based on the dependency relations, they proposed five syntactic rules (DIRECT OBJECT rule, NOMINAL SUBJECT rule, ADJECTIVAL MODIFIER rule, PREPOSITIONAL OBJECT rule, and RECURSIVE MODIFIER rule) to pair opinions with aspects.

### 6.2.4 Aspect Pruning

We observe that more than 80% of the aspects extracted using the syntactic rules are not related to the debate topics. For instance, in Example 2, the sentence ‘*Windows gives more freedom to users*’, has the aspect *freedom*, which is neither related to *Windows* nor *Mac*. Similarly, when we manually inspect the aspects extracted from the ‘*Firefox Vs Internet Explorer (IE)*’ debate post, we observe that out of 552 aspects obtained using the syntactic rules, only 113 are found to be relevant to either of the debate topics. We identify five types of irrelevant aspects that need to be pruned out

---

<sup>1</sup><https://nlp.stanford.edu/software/tagger.shtml>

<sup>2</sup><http://sentiwordnet.isti.cnr.it/>

<sup>3</sup><https://nlp.stanford.edu/software/lex-parser.shtml>

based on the POS patterns as such aspects do not contribute in deciding the stance of the debate post. They are described in Table 6.1.

	Sample	
Class	irrelevant aspects	Example debate posts
Pronouns		Firefox provides options to import and export bookmarks.
	anyone, everyone, everything, someone, something,	So, <b>anyone</b> can transfer all their bookmarks from one browser to another by performing a few simple steps.
	who, which, us, they, anyway	Why does Microsoft continue developing internet explorer if <b>everyone</b> thinks of this browser as a joke compared to other browsers. This argument is futile <b>anyway</b> since Xbox is the best next-gen console out there. Firefox has a vast team of volunteers <b>who</b> are willing to donate their time to fight new types of ads.
Proper nouns	ALT, AOL, W3C, Apple, IIS, IM, MS, Vista, Warner, Norton, Yahoo, ads, Warner, alt, del, Ballmer	<b>Norton</b> blocks off IE from any updates as IE is filled with many ads and pop-ups. Windows does not have any uptime problem as control <b>alt del</b> is usually more than enough to deal with any problem. <b>Ballmer</b> and the Windows engineering team spent almost two days trying to rid of worms, viruses, spyware, malware. IE is filled with many <b>ads</b> and popups.

Continued on next page

Table 6.1 – continued from previous page

Class	Sample irrelevant aspects	Example debate posts
Common nouns related sentences	in article, availability, base, beholder, example	<p>IE's bloated <b>approach</b> to software have frequently prevented it from rolling out much sought after changes and improvements.</p> <p>Forgot to add this in my last <b>argument</b>, but Windows also has games, whereas Macs do not have nearly as many.</p> <p>I think that over time the Wii gameplay is going to get stale.</p> <p>For <b>example</b>, if you look at Wii sports, there are limited possibilities of actual different motions you actually have to execute to play the game.</p> <p>Wow, you are writing an <b>article</b> in support of Windows and yet you have never even had it crash on you?</p>

Continued on next page

Table 6.1 – continued from previous page

Class	Sample irrelevant aspects	Example debate posts
Common nouns in unrelated sentences	bank, avail- ability, bat- tle, beast, brother, business, economics, cars, case	<p>Just because IE has 70% of market share doesn't mean it's the best in terms of quality. Honda Civics out-sell BMWs and other luxury <b>cars</b> 4 or 5 to 1, but I think if you drive both <b>cars</b> you may find that a Civic is not a higher quality <b>car</b> than BMW.</p> <p>My family computer was locked down as well as it could be - firewall, antivirus, antispysware once a week, latest patches. But it got infected while using IE to browse a <b>bank</b> website, whose web servers were already infected.</p> <p>I am not so sure why lowering prices to increase sales is a sign of failure. I am pretty sure almost every <b>business</b> does that at some point, and in fact, it is a natural process of <b>economics</b>.</p> <p>I would settle for a Linux distro any day, but Windows will always win the <b>battle</b> over Mac.</p>

Continued on next page



Table 6.1 – continued from previous page

Class	Sample irrelevant aspects	Example debate posts
Misspelt or unknown words	addage, desigenrs, wewill, youhave, haveto	<p>IE does not support the well known <b>addage</b>: time is money.</p> <p>IE’s security holes create an opening for spyware that slows down my computer, lost time and productivity.</p> <p>Macs are often touted as being good for <b>desigenrs</b>, producers and artists.</p> <p>Everyone that does not like windows talks about how unstable it is, how you <b>haveto</b> restart from viruses all the time.</p> <p>Saying <b>youhave</b> never had a virus does not mean they do not exist and that the issue is a myth.</p>

Table 6.1: Classes of irrelevant aspects obtained after applying the syntactic rules.

It is observed that the irrelevant aspects are either pronouns, proper nouns, common nouns, or unknown words. Pronouns do not represent aspects, and most of the proper nouns are not related to debate targets. The syntactic rules cannot distinguish common nouns that are actual aspects from the irrelevant ones. Also, there are cases of common nouns present in unrelated sentences. We perform aspect pruning using word embeddings, to filter out such irrelevant aspects before further processing. Step 4 of Algorithm 5 uses word2vec embeddings to build word vectors for aspects and targets. The distance between the word vectors is used to prune the aspects that are unrelated to the targets.

## Word2vec Embedding

Word2vec model uses continuous bag-of-words and skip-gram methods to generate word vectors. Word vectors represent text as dense vectors in low dimensional space. Word vectors are learned from Google News dataset<sup>4</sup>, which consists of almost 100 billion words. Word2vec representation incorporates accurate word relatedness information. Word meaning and relationship between the words are encoded spatially. The spatial distance between the word vectors correspond to word similarity. We therefore generate word vectors for aspects and targets using word2vec and compute word vector similarity score. Since there are two targets in a debate, each aspect generates a two-dimensional feature vector from the similarity scores computed using the two targets. Once we have the feature vectors, we use them as features to build Gaussian Naive Bayes (GNB) and Support Vector Machine (SVM) aspect classifiers. Upon comparison, GNB is found to predict related aspects better than SVM since the feature vectors are only two dimensional and GNB works well with continuous values. Hence, we use GNB for aspect pruning. The details of GNB implementation are given in the next section.

## Gaussian Naive Bayes (GNB)

Initially, we consider *Firefox vs. IE* debate posts to build the GNB aspect-target classifier. There are 552 aspects present in the AOP of *Firefox vs. IE* debate posts. Three research scholars manually annotate the aspects as related or not related to either of the targets. The disagreements are resolved through mutual discussion. After manual annotation, it is found that 114 aspects are related to either of the targets and 438 aspects not related. Then, we perform a random split of the aspects into 370 (66.66%) training and 182 (33.33%) testing data. The 2-D similarity score feature vector is used to train the GNB classifier. After cross validation, the trained

---

<sup>4</sup><https://goo.gl/4Hzd8L>

model gives an average accuracy of 81% in classifying the aspects. This trained model is used to classify whether a new aspect from the post is related to either of the debate targets or not.

In general, manual labelling of aspects for each debate takes effort and is time consuming. We, therefore, test if transfer learning can be applied on the GNB classifier for cross-domain classification of irrelevant aspects. For this purpose, the research scholars perform manual annotation of related aspects on the remaining three datasets. The above trained GNB classifier is applied to the remaining three datasets to remove irrelevant aspects. The results of the classifier are shown in Table 6.7, where the classifier is trained on only *Firefox vs. IE* debate dataset. We observe that the

Measure	Firefox vs. IE	Windows vs. Mac	Sony Ps3 vs. Wii	Opera vs. Firefox
Classification accuracy (%)	81	84	87	84.20

Table 6.2: Aspect pruning accuracy across all four debate datasets.

GNB classifier is able to detect target-related aspects across multiple domains with high accuracy. This indicates that the classifier learns a function whether the aspect is relevant or irrelevant based on the aspects similarity with the target topics. Since this function is independent of the targets or the aspects, it can be used for transfer learning across multiple debate topics, without lowering classification accuracy.

### 6.2.5 Aspect-Polarity Mapping

After getting the aspects related to debate targets, the opinion words in aspect-opinion pairs (AOP) are substituted with the respective polarities from  $O_L$  to form aspect-polarity pairs (AP) (e.g.,  $\{Windows, bad\} \rightarrow \{Windows, -1\}$ ). There is a special case where negation words (e.g., not, no, and nor) are present before the opinion words. In such cases, the polarity of the opinion word present in  $O_L$  is reversed.

Thus, we get the sentiment expressed on aspects of the post. Aspect-polarity pairs ( $AP$ ) are used to find the opinions expressed on the targets.

### 6.2.6 Aspect-Target Mapping

In this step, we find the target preference of each aspect. To find the target preference, we use the word vector similarity score generated using Google word2vec. For each aspect, we compute the word2vec similarity with both the targets. Algorithm 6 gives the details of aspect-target mapping that determines each aspect’s preference towards the two targets.

---

#### Algorithm 6 $ATP(A, T)$

---

**Input:**  $A: \{a_1, a_2, a_3, \dots, a_m\}$  = Set of relevant aspects  
 $T: t_1, t_2$  = Two targets of the post  
 $th$ : Threshold difference  
**Output:**  $AT = aspect : target$  = Aspect-target preference dictionary  
**Method:**

```

1:  $AT \leftarrow []$ 
2: for all  $a_i \in A$  do
3:    $sim1 \leftarrow \text{WORD2VEC\_SIMILARITY}(a_i, t_1)$ 
4:    $sim2 \leftarrow \text{WORD2VEC\_SIMILARITY}(a_i, t_2)$ 
5:    $diff \leftarrow |SIM1 - SIM2|$ 
6:   if  $diff < th$  then
7:      $AT[a_i] \leftarrow (t_1, t_2)$ 
8:   else if  $sim1 > sim2$  then
9:      $AT[a_i] \leftarrow (t_1)$ 
10:  else
11:     $AT[a_i] \leftarrow (t_2)$ 
12:  end if
13: end for
14: return  $AT$ 

```

---

Given the debate targets  $T$  and a set of relevant aspects  $A$  obtained after aspect pruning, Step 1 initializes the aspect-target preference dictionary  $AT$ . For each aspect  $a_i$ , Steps 3–4 computes word2vec similarity between each relevant aspect  $a_i$  and the targets  $t_1$  and  $t_2$ . Step 5 computes the difference between the similarity scores.

Steps 4–6 determines the aspect-target preference based on the similarity and difference scores. If the difference in the similarity score is less than a threshold  $th$ , we consider that the aspect is related to both the targets. Otherwise, the aspect prefers the target with the higher similarity score. The value of  $th$  is determined empirically using the four datasets described in Section 6.3. We tried different values of  $th$  (0.05, 0.10, 0.15, 0.20, 0.25, 0.30) and found that  $th=0.1$  gives the best result. After the comparisons have been made for all the relevant aspects, we get the aspect-target preference dictionary ( $AT$ ).

To illustrate, consider the debate post comparing Windows and Mac.

**Example 11** “*Windows supports most of the games. Mac has a lot of software compatibility issue*”.

Here the targets are *Windows* and *Mac* and the aspects are *games* and *software*. To find the target preference of the aspects, let us consider the difference threshold ( $th$ ) as 0.10. First, we compute the similarity ( $sim1$ ,  $sim2$ ) of the aspect *games* and both the targets. Next, we compute the difference ( $diff$ ) in similarity score. We see that the difference in sim score is more than the threshold  $th$ . Also, the similarity of *games* with *Windows* is more than that of *games* with *Mac*. So, the aspect *games* prefers the target *Windows*. Similarly, we find that the aspect *software* prefers the target *Windows*. Table 6.3 shows a few examples of target preference of some of the aspects.

Aspect	Target preference
Games	Windows
Software	Windows
Bookmark	Firefox
Mozilla	Firefox

Table 6.3: Aspect-Target (AT) preference dictionary examples

### 6.2.7 Detect Stance

The next step is to deduce the stance of the user. We articulate this problem as an Integer Linear Programming (ILP) optimization problem. ILP is a powerful optimization technique, which optimizes a function based on some given constraints. We use the  $AT$  preference dictionary and the similarity information to calculate a weighted score. This score is used to formulate the ILP equations. For calculating the weighted score, we collect all the aspect-polarity pairs  $AP$  and the aspect-target preference dictionary  $AT$ . Let  $m$  be the number of aspect-polarity pairs present in a post. The weighted score for each target-aspect pair ( $S_{ij}$ ) is computed as follows:

$$S_{ij} = (AP(j) * sim(i, j) * NF(j) + diff(sim(i, j), sim(1 - i, j)) + T_i) * PT_j \quad (6.1)$$

where  $i = (0, 1)$ , represents the debate targets,  $j = \{a_1, a_2, a_3, \dots, a_m\}$  represents the relevant aspects present in the post,  $AP(j) = \{-1, 0, 1\}$  is the polarity of aspect  $j$  in the post,  $sim(i, j)$  is a continuous valued word2vec similarity score between the target  $i$  and the aspect  $j$ , and  $diff(sim(i, j), sim(1 - i, j))$  is the absolute difference of the similarity scores of aspect  $j$  with each of the targets. The difference score is used to penalize those aspects that are common to both the targets and reward those aspects that are unique towards a target.  $T_i$  is a preference value that rewards when the aspect is also the target. This happens when a the post contains a target along with an associated opinion, such as “*Windows is obviously better than Mac*”.  $NF(j)$  is the normalized frequency of the aspect  $j$ . It is used to give more weightage to those aspects that are present more frequently in the post corpus. This is because frequent aspects are generally more popular and well known.  $NF(j)$  is computed as the ratio of the frequency of aspect  $j$  in the corpus to the frequency of all the aspects in the

corpus. It is defined as follows:

$$NF(j) = \frac{|j|}{\sum_{p=1}^m |j_p|} \quad (6.2)$$

$PT_j$  is the value of aspect preference over the target. If the aspect does not prefer the target, it gives a score of 0, thus ignoring the weighted score when the aspect prefers the opposite target. If the aspect prefers the target, then  $PT_j$  gets a value of 1. The preference information is obtained from *AT* dictionary. It is formulated as follows:

$$PT_j = \begin{cases} 1, & \text{if } (j \in AT[j]) \\ 0, & \text{otherwise} \end{cases} \quad (6.3)$$

### ILP formulation

The stance of the post is determined by the side that maximizes the *ILP* objective function. Given all the aspect-target instances of a post, the objective function is formulated as follows:

$$\sum_{j=1}^m (S_{0j}p_j + S_{1j}q_j) \quad (6.4)$$

and it's constraints are,

$$p_j, q_j \in 0, 1, \forall j \quad (6.5)$$

$$p_j + q_j = 1, \forall j \quad (6.6)$$

$$p_j - p_{j-1} = 0, j \in \{2 \dots m\} \quad (6.7)$$

$$q_j - q_{j-1} = 0, j \in \{2 \dots m\} \quad (6.8)$$

where,  $p_j, q_j$  are boolean variables.  $p_j$  supports the first target when its value is 1 while  $q_j$  supports the second target when its value is 1. Equation 6.6 makes sure that the variables are mutually exclusive so that each aspect instance can prefer only one

target. Equations 6.7 and 6.8 make sure that the user supports only one side and writes in support of that side for all aspects. This is based on the assumption that in a debate, the user is consistent about her stance of supporting single target and its aspects. So, for a particular post, all the values of  $p_j$  (or  $q_j$ ) are made identical. This is a unique condition for debate posts unlike others such as forum posts, reviews, and tweets, where the user writes both good and bad points about a target.

To make the constraints understand better, reconsider Example 11, from *Windows* vs *Mac* debate post “Windows supports most of the games. Mac has a lot of software compatibility issues”.

Aspect	Target	Polarity (+/-)	p	q
Games	Windows	+	1	0
Software	Mac	-	1	0

Table 6.4: ILP implementation example

As can be seen from Table 6.4, the targets are *Windows* and *Mac*. The aspect *games* has a positive opinion about *Windows* while the aspect *software* has negative opinion about *Mac*. In other words, a negative opinion about mac means this aspect *software* is also preferring *Windows*. As the author is writing in favor of *Windows*, the aspects are supporting the target *Windows*. Equations 6.5 and 6.6 states that both  $p$  and  $q$  are boolean and mutually exclusive. This is reflected in the table where the values of p’s are all 1 and q’s are all 0. Equations 6.7 and 6.8 are reflected in the fact that all the aspects support only the target *Windows*.

## 6.3 Evaluation

In this section, we describe our dataset and evaluation metrics. We compare the performance of our proposed system, Debate Stance Classification using Word Embeddings (*DSCW*) with existing unsupervised debate classification approaches, and



show that our method can detect stance side more accurately.

### 6.3.1 Dataset

We consider the debate side annotated corpora used in [140]. This corpora consists of posts from four debate topics *Firefox vs. Internet Explorer (IE)*, *Opera vs. Firefox*, *SonyPs3 vs. Wii*, and *Windows vs. Mac*. Each of the datasets has manually annotated debate side preference. The datasets are summarized in Table 6.5.

Sl. No.	Debate topic	Number of posts
1	Firefox vs. Internet Explorer (IE)	169
2	Opera vs. Firefox	16
3	SonyPs3 vs. Wii	68
4	Windows vs. Mac	27

Table 6.5: Details of debate post dataset.

### 6.3.2 Stance Prediction Evaluation

#### Evaluation Metrics

We evaluate the stance prediction performance of our proposed method in terms of the accuracy of debate stance classification. We consider all the posts as relevant for debate classification. The accuracy of our debate classification method in classifying debate stance is measured in terms of precision( $P$ ), recall( $R$ ) and  $F1$  score. The metrics are defined as follows:

$$P = \frac{\# \text{ correctly classified post}}{\# \text{ classified post}} \quad (6.9)$$

$$R = \frac{\# \text{ correctly classified posts}}{\# \text{ relevant post}} \quad (6.10)$$

$$F1 = \frac{2 * P * R}{P + R} \quad (6.11)$$

## Baseline Methods

In order to evaluate the effectiveness of the proposed method, we conducted accuracy comparison with three baseline methods. We applied OpTopic, OpPMI, and OpProbability methods described in [140] on the four datasets and determine their stance prediction accuracy.

OpTopic finds target-polarity pairs from debate posts and counts the number of positive opinion words associated with each of the debate topics. A negative opinion word or a topic is counted as a positive opinion word for the opposing topic. Debate stance is assigned to the topic with more positive opinion word count.

OpPMI extracts terms (nouns) from debate posts and computes Pointwise Mutual Information (PMI) of the terms with each target. Then, the target-polarity pairs are found and the count of positive opinion words is computed. Debate stance is assigned to the side with more positive opinion word count as in OpTopic. Each term is then assigned to the target with higher PMI value. The polarity-target pair for each noun is then computed, to find terms which are closely related to the debate topics. In both methods, the post is assigned to the side with higher cumulative polarity-topic pair score.

In OpProbability, aspect-opinion pairs are extracted from debate posts and are converted to aspect-polarity pairs. Weblogs and forums related to the topics are mined to find aspect-target preference. An aspect-target preference probability table is then constructed from the mined corpus and a classifier is built using ILP to classify debate posts.

We compare the performance of the three baseline approaches with our two proposed methods: debate Opinion detection using Word Embedding (OpWE), and debate Opinion detection using Word Embedding and Aspect Pruning (OpWEAP). As mentioned in Section 6.2, OpWE uses word embeddings to build the aspect-target preference mapping ( $AT$ ) by comparing the word embedding similarity scores be-

tween the aspects and each target. Then stance classification is done by applying ILP to weighted similarity score ( $S_{ij}$ ) obtained from  $AT$ . Additionally, OpWEAP applies a Gaussian Naive Bayes aspect pruning classification to  $AOP$  (as described in Section 6.2) before computing  $AT$  to filter aspects, which are not related to either of the targets.

### 6.3.3 Results

The result of the accuracy comparison of the five algorithms, namely OpTopic, OpPMI, OpProbability, OpWE and OpWEAP is presented in Table 6.6 and also, in the form of plots in Figures 6.2, 6.3 and 6.4 respectively.

Metric	Debate topic	OpTopic	OpPMI	OpProb	OpWE	OpWEAP
<b>Precision</b>	Firefox vs. IE	67.74	60.00	66.27	75.65	<b>79.58</b>
	Windows vs Mac	40.00	53.84	66.67	69.56	<b>77.27</b>
	SonyPs3 vs. Wii	<b>80.00</b>	46.15	61.11	68.96	70.90
	Opera vs. Firefox	33.33	<b>100.00</b>	<b>100.00</b>	75.00	81.82
<b>Recall</b>	Firefox vs.. IE	17.16	27.22	33.72	<b>68.04</b>	66.86
	Windows vs. Mac	7.04	25.93	37.03	59.25	<b>62.96</b>
	SonyPs3 vs. Wii	17.65	17.65	32.35	<b>58.82</b>	57.35
	Opera vs. Firefox	12.50	25.00	43.75	56.25	<b>56.25</b>
<b>F1</b>	Firefox vs. IE	27.38	37.45	44.70	71.64	<b>75.41</b>
	Windows vs. Mac	11.98	35.00	47.61	64.00	<b>69.38</b>
	SonyPs3 vs. Wii	28.92	25.53	42.30	<b>63.48</b>	63.41
	Opera vs. Firefox	18.18	40.00	60.86	64.29	<b>66.67</b>

Table 6.6: Precision, Recall, and  $F1$  measure of various debate classification methods.

From Table 6.6, we can see that among the three baselines, the OpProbability algorithm gives the highest precision and recall. The reason for this higher precision and recall compared to OpTopic and OpPMI is that it uses external knowledge to

## Precision

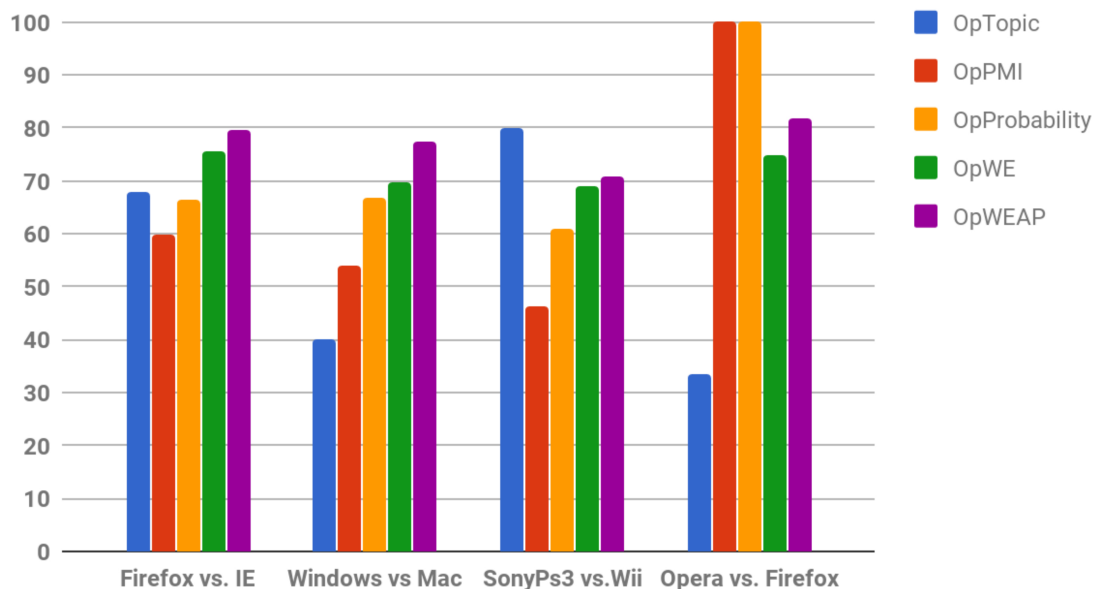


Figure 6.2: Precision comparison for various debate classification methods.

## Recall

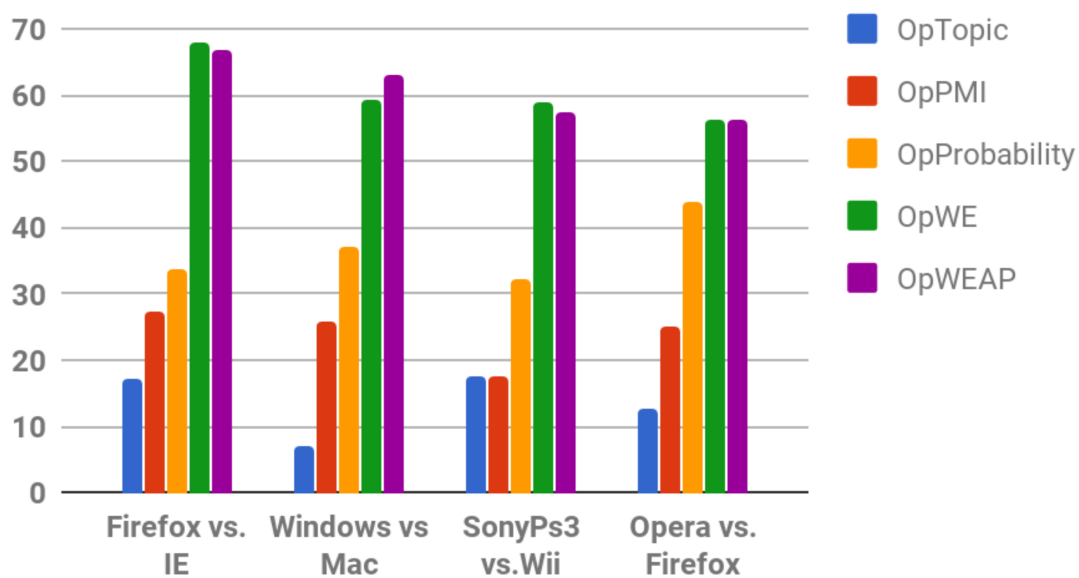


Figure 6.3: Recall comparison for various debate classification methods.

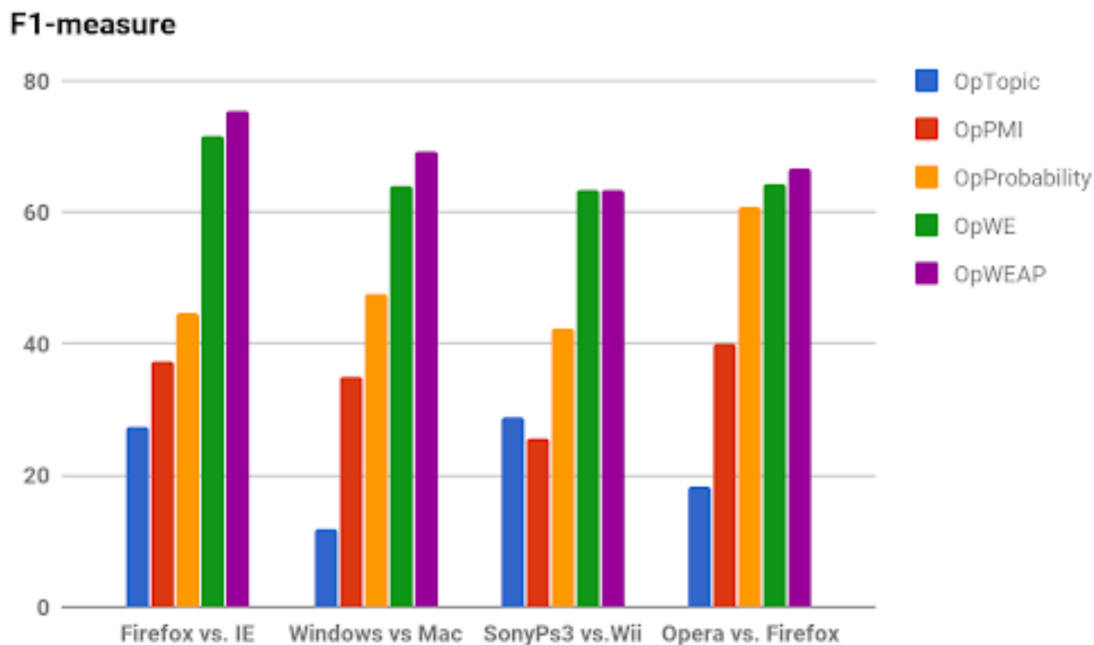


Figure 6.4: F1 score comparison for various debate classification methods.

determine the aspect-target preference. OpProbability is able to learn aspect preference from the web corpus. Even this approach is limited by the availability of proper domain information from the web corpus. The web corpus may have limited number of pages or irrelevant pages.

As can be noted, OpTopic has very low recall in detecting debate stance. This is because it is a naive approach that relies only on opinions associated with the debate topic. The opinions of the aspects related to the topic are discarded. OpPMI is able to detect stance with higher recall as it associates each term with either of the target and opinion of each term is considered. However, in this case, the precision drops as not all the terms are aspects and a term be associated with both targets.

Compared to OpProbability our proposed two algorithms give 10% higher precision and 30% higher recall. The reason for this very high recall is that we are able to detect stance in more number of posts because of better pre-processing, more external knowledge in the form of word embedding, and a bigger opinion lexicon.

In the *Opera vs Firefox* dataset, two of the baseline algorithms give 100% precision, whereas our best method gives only 82%. The reason is the size of the dataset. This dataset has only 16 posts, and the algorithm is able to detect stance in only 4 out of the 16 posts. Since it detects correct stance in all the 4 posts, it has 100% precision. However, the recall of these algorithms is 25% and 43%, which is 13% lower compared to our methods.

OpWE works well in multiple domains since the word embeddings are obtained from a large text corpus and no domain-specific corpus is required. Compared to OpWE, OpWEAP further improves the average precision by 5% and  $F_1$  score by 2.85%. The result suggests that pruning aspects irrelevant to the targets increases the classification accuracy. The reason for the increase is that it removes their influence in detecting stance in the ILP formulation.

### 6.3.4 Aspect Pruning Evaluation

In this section, we describe the GNB classifier used for pruning irrelevant aspects, and evaluate its effectiveness in determining topic-related aspects. We also evaluate the use of the GNB classifier, trained on a domain, for transfer learning for pruning irrelevant aspects in other domains.

#### Gaussian Naive Bayes (GNB)

Initially, we consider *Firefox vs. IE* debate posts to build the GNB aspect-target classifier. There are 552 aspects present in the AOP of *Firefox vs. IE* debate posts. Three research scholars manually annotate the aspects as related or not related to either of the targets. The disagreements are resolved through mutual discussion. After manual annotation, it is found that 114 aspects are related to either of the targets and 438 aspects not related. Then, we perform a random split of the aspects into 370 (66.66%) training and 182 (33.33%) testing data. The 2-D similarity score

feature vector is used to train the GNB classifier. After cross validation, the trained model gives an average accuracy of 81% in classifying the aspects. This trained model is used to classify whether a new aspect from the post is related to either of the debate targets or not.

In general, manual labelling of aspects for each debate takes effort and is time consuming. We, therefore, test if transfer learning can be applied on the GNB classifier for cross-domain classification of irrelevant aspects. For this purpose, the research scholars perform manual annotation of related aspects on the remaining three datasets. The above trained GNB classifier is applied to the remaining three datasets to remove irrelevant aspects. The results of the classifier are shown in Table 6.7, where the classifier is trained on only *Firefox vs. IE* debate dataset. We observe that the

<b>Measure</b>	<b>Firefox vs. IE</b>	<b>Windows vs. Mac</b>	<b>Sony Ps3 vs. Wii</b>	<b>Opera vs. Firefox</b>
Classification accuracy (%)	81	84	87	84.20

Table 6.7: Aspect pruning accuracy across all four debate datasets.

GNB classifier is able to detect target-related aspects across multiple domains with high accuracy. This indicates that the classifier learns a function whether the aspect is relevant or irrelevant based on the aspects similarity with the target topics. Since this function is independent of the targets or the aspects, it can be used for transfer learning across multiple debate topics, without lowering classification accuracy.

Table 6.8 shows the word embedding similarity score of some aspects with respect to the targets *Firefox* and *Internet Explorer (IE)*. We observe that the aspects *bookmark*, *browser*, *Firefox*, and *Linux* that have high similarity scores with both the targets are relevant to the targets *Firefox* and *IE*, while the aspects *address*, *architecture*, *color*, and *videos* that have low similarity scores with the targets are found to be irrelevant. Also, aspects with low difference in similarity score are either common

Aspect	Similarity (Firefox)	Similarity (IE)	Relevant(1)/Irrelevant(0)	Difference
Bookmark	.43	.36	1	.07
Browser	.72	.86	1	.04
Firefox	1.00	.52	1	.11
Linux	.56	.46	1	.10
Address	.04	.12	0	.08
Architecture	.19	.16	0	.08
Color	.12	.10	0	.02
Video	.16	.14	0	.02

Table 6.8: Word embedding similarity score of aspects.

aspects for both the targets (*bookmark*, *browser*) or irrelevant to the targets (*color*, *video*).

## 6.4 Conclusion

This chapter presents an unsupervised approach to perform stance classification of comparative reviews using word embeddings. The use of word embeddings enables us to perform stance detection without using a web corpus. Our results show that our approach can detect user stance significantly better than existing methods. In existing techniques, many of the aspects obtained are irrelevant to the debate topics. We introduce data cleaning, lexicon expansion, and aspect pruning for better opinion and aspect detection. We also train a supervised aspect classifier that can be used as a transfer learning method to detect target-related aspects across multiple domains. Our approach is scalable to two-sided debate posts across multiple domains as it does not require any domain specific information to perform stance detection.



# Chapter 7

## Conclusion

In this chapter, we conclude this thesis by summarizing the contribution and then propose some directions for future work.

### 7.1 Summary of Contribution

In this thesis, we study the problem of review exploration and summarization in e-commerce products. We highlight three challenges in review exploration, namely, finding the semantic relationship between aspects, providing a meaningful summary of reviews and summarizing comparative reviews. To address each of the challenges, we propose three different ways of review exploration, namely exploration using aspect ontology, exploration using opinionated tags, and stance detection in comparative reviews. To find the semantic relationship between aspects, we present a novel method to construct aspect ontology. We next look at summarizing reviews by generating useful and understandable tags from reviews. Finally, we work on detecting the stance of users in comparative review through the use of word embeddings. The significant findings of the research work are summarized in the following subsections.

### **7.1.1 Review Exploration using Aspect Ontology**

Knowing the semantic relation between aspects and sub-aspects is an interesting, yet challenging problem for aspect-based review summarization systems. Existing aspect-based review exploration systems assume a flat relationship between aspects. In Chapter 4, we propose a solution to this problem by arranging the aspects in the form of an ontology to show hierarchical aspect relations. We propose two novel unsupervised approaches to create aspect ontology. Aspects extracted from reviews are used to create ontology based on semantic relationship (SemR) and semantic relatedness knowledgebase (SemS). SemR uses manually created knowledge graph, which limits the number of aspects and relations that can be identified. On the other hand, SemS uses co-occurrence pattern of words in a large corpus to compute semantic similarity. This helps in overcoming the limitations of knowledge graph, which requires prior knowledge between aspects. After creating the ontology, reviews are aggregated according to the aspects present in the ontology. Summarizing reviews using aspect ontology provides a user-friendly interface to browse through unstructured and voluminous reviews. Aspect specific reviews can be viewed by choosing a particular aspect of interest. It also helps to overcome the lack of domain knowledge about the products by visualizing the relations between various aspects and sub-aspects of a product.

### **7.1.2 Review Exploration using Opinionated Tags**

Existing review exploration systems mainly focus on providing an overall sentiment or aspect sentiment from a review. In Chapter 5, we propose a novel approach to provide meaningful insight into product aspects from reviews by generating opinionated tags. In particular, we choose the tags that have both aspect and opinion words, and cover a majority of reviews. Based on the nature of reviews, we classify products as popular and cold products. We employ three topic modeling techniques, namely popularity,

TF-TDF and TNG to generate meaningful tags. We show that TNG gives better tags compared to popularity and TF-IDF as TNG considers the sequential nature of text to generate coherent words and phrases as tags. For cold products, the tag generation techniques generate many noisy tags due to lack of review content. So, we leverage a tag cloud obtained from the tags of popular products to discover the tags in cold products. Once the tags are generated from reviews, we find the top-k most useful tags based on a greedy set coverage algorithm. Then, we apply NLP syntactic rules to make the tags more understandable. Our proposed approach outperforms three baselines, namely popularity, TF-IDF and TNG as it can identify important opinionated aspects better than baselines.

### **7.1.3 Stance Detection in Comparative Reviews**

Comparative reviews contain detailed comparisons of competing products and are more useful than single reviews as they provide an insight into which of the competing product is better. Knowing the stance of comparative reviews can influence people’s opinion. In Chapter 6, we propose to summarize comparative reviews by determining the stance of users from the review posts. Stance detection is harder than standard sentiment analysis due to the presence of multiple targets. To determine the stance of a user, we need to find the target preference of each aspect present in the debate post. We propose a domain-independent approach of stance detection using word embeddings. Word embedding is employed to find the relevant aspects as well as the aspect-target preference. The use of word embedding in our approach helps to prune irrelevant aspects that are present in existing approaches. Word embedding also helps in finding target preference of each aspect, which is essential to determine the stance of users. Our word embedding-based approach is able to detect the stance of most of the comparative posts and outperforms existing corpus-based approaches in detecting the correct stance.

## 7.2 Future Work

The thesis explores various ways of review exploration and summarization. Following are some possible directions for future work:

- Semantic relationship (SemR) and semantic similarity (SemS) methods have their own merits and demerits. As a part of future work, SemS and SemR methods may be combined using ensemble learning to improve aspect ontology. Both knowledge graph and semantic similarity can complement one another in finding strong aspect and sub-aspect relations. Strong aspect relations would help in constructing a more accurate aspect ontology and would also reduce topic drift.
- The product descriptions provided by sellers may be incorporated so as to improve aspect ontology. Product description specified on online shopping sites contains useful objective information about various aspects and sub-aspects of the product. Such information is accurate and can be used to enhance finding relations between aspects and sub-aspects.
- Aspect-level stance detection in political discourse may be performed. Political discourse has a countable number of aspects. Instead of detecting the overall stance of users, we can identify the various aspects present in political discourse and then find: (a) Which discourse aspects the user is concerned about and has an opinion on? (b) What is the stance of the user on the aspects?
- Opinionated tag summary may be generated from pros and cons reviews. Pros and cons reviews contain a vast treasure trove of aspects and sentiments and provide a more in-depth understanding of product quality. Tag generation and stance detection methods can be combined to provide opinionated tags along with the stance of pros and cons reviews.

- The proposed TPR tag summarization system provides an overall summary of the product reviews. We may customize the system to make more sense by providing a tag summary depending on the context of who would receive the tag recommendation. For example, tags may benefit the manufacturers by providing customer feedback about their products.
- Tag recommendation can be personalized for each customer. Different users have different taste and preference of aspects and products. User browsing and purchase history may be used to find their aspect preference, and personalized relevant tags may be shown to the users.
- In this thesis, discrete values are used to test threshold parameters (Algorithm 3 and Algorithm 6). A good optimization problem can help to optimize the threshold by considering continuous values.
- ILP solvers are computing expensive. When we consider debate topics having a large number of debate posts, it may become computationally expensive to use general ILP solving techniques. Another direction for future research may be to work on developing a fast and robust ILP solver that may scale to big data.
- The proposed stance detection technique is suitable for two-sided debates posts with only two opposing targets. The technique is not applicable to debate posts where multiple targets are compared. Also, the technique would perform poorly on debate posts that compare multiple models of the same brand. Addressing these limitations would increase the robustness of our technique.
- Semantic Similarity (SemS) uses Latent Semantic Analysis (LSA) of Wikipedia data for ontology creation. As we create the ontology for a product belonging to one domain, training the LSA model on product reviews belonging to a domain may provide good insight and may increase the accuracy of the similarity

measure. It would be good to evaluate the ontology we get using Wikipedia vs using the review corpus.

# Bibliography

- [1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, “Tensorflow: A system for large-scale machine learning,” in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI’16. Berkeley, CA, USA: USENIX Association, 2016, pp. 265–283. [Online]. Available: <http://dl.acm.org/citation.cfm?id=3026877.3026899>
- [2] S. Alashri, S. Alzahrani, L. Bustikova, D. Siroky, and H. Davulcu, “What animates political debates? analyzing ideological perspectives in online debates between opposing parties,” in *Proceedings of SocialCom ’15*. ASE/IEEE, 2015.
- [3] P. Anand, M. Walker, R. Abbott, J. E. F. Tree, R. Bowmani, and M. Minor, “Cats rule and dogs drool!: Classifying stance in online debate,” in *Proceedings of the workshop on computational approaches to subjectivity and sentiment analysis*. ACL, 2011, pp. 1–9.
- [4] O. Appel, F. Chiclana, J. Carter, and H. Fujita, “A hybrid approach to the sentiment analysis problem at the sentence level,” *Knowledge-Based Systems*, vol. 108, pp. 110–124, 2016.

- [5] M. Araujo, J. Reis, A. Pereira, and F. Benevenuto, “An evaluation of machine translation for multilingual sentence-level sentiment analysis,” in *Proceedings of the 31st Annual Association for Computing Machinery Symposium on Applied Computing*. ACM, 2016, pp. 1140–1145.
- [6] D. Bačić and A. Fadlalla, “Business information visualization intellectual contributions: An integrative framework of visualization capabilities and dimensions of visual intelligence,” *Decision Support Systems*, vol. 89, pp. 77–86, 2016.
- [7] A. Balahur, Z. Kozareva, and A. Montoyo, “Determining the polarity and source of opinions expressed in political debates,” *Computational Linguistics and Intelligent Text Processing*, vol. 5449, pp. 468–480, 2009.
- [8] P. Barnaghi, G. Kontonatsios, N. Bessis, and Y. Korkontzelos, “Aspect extraction from reviews using convolutional neural networks and embeddings,” in *International Conference on Applications of Natural Language to Information Systems*. Springer, 2019, pp. 409–415.
- [9] M. Baziz, M. Boughanem, N. Aussenac-Gilles, and C. Chrisment, “Semantic cores for representing documents in ir,” in *Proceedings of the 2005 ACM Symposium on Applied Computing*, ser. SAC ’05. New York, NY, USA: ACM, 2005, pp. 1011–1017. [Online]. Available: <http://doi.acm.org/10.1145/1066677.1066911>
- [10] F. Belém, R. Santos, J. Almeida, and M. Gonçalves, “Topic diversity in tag recommendation,” in *Proceedings of the 7th Association for Computing Machinery conference on Recommender systems*. ACM, 2013, pp. 141–148.
- [11] F. M. Belém, E. F. Martins, J. M. Almeida, and M. A. Gonçalves, “Personalized and object-centered tag recommendation methods for web 2.0 applications,” *Information Processing & Management*, vol. 50, no. 4, pp. 524–553, 2014.



- [12] P. Bhatia, Y. Ji, and J. Eisenstein, “Better document-level sentiment analysis from RST discourse parsing,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, 2015, pp. 2212–2218. [Online]. Available: <http://aclweb.org/anthology/D/D15/D15-1263.pdf>
- [13] S. Blair-Goldensohn, K. Hannan, R. McDonald, T. Neylon, G. Reis, and J. Reynar, “Building a sentiment summarizer for local service reviews,” in *WWW Workshop on NLP Challenges in the Information Explosion Era (NLPIX)*, 2008.
- [14] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent dirichlet allocation,” *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.
- [15] E. Breck and C. Cardie, “Opinion mining and sentiment analysis,” in *The Oxford Handbook of Computational Linguistics 2nd edition*, 2017.
- [16] B. N. Bullock, A. Hotho, and G. Stumme, “Accessing information with tags: Search and ranking,” in *Social Information Access*. Springer, 2018, pp. 310–343.
- [17] C. Burfoot, S. Bird, and T. Baldwin, “Collective classification of congressional floor-debate transcripts,” in *Proceedings of ACL HLT ’11*. ACL, 2011, pp. 1506–1515.
- [18] E. Cambria, “Affective computing and sentiment analysis,” *IEEE Intelligent Systems*, vol. 31, no. 2, pp. 102–107, 2016.
- [19] H. Cao, M. Xie, L. Xue, C. Liu, F. Teng, and Y. Huang, “Social tag prediction base on supervised ranking model,” in *Proceeding of ECML/PKDD 2009 Discovery Challenge Workshop*, 2009, pp. 35–48.

- [20] G. Carenini, R. T. Ng, and A. Pauls, “Interactive multimedia summaries of evaluative text,” in *Proceedings of the 11th International Conference on Intelligent User Interfaces*, ser. IUI '06. New York, NY, USA: ACM, 2006, pp. 124–131. [Online]. Available: <http://doi.acm.org/10.1145/1111449.1111480>
- [21] L. Chen, L. Qi, and F. Wang, “Comparison of feature-level learning methods for mining online consumer reviews,” *Expert Systems with Applications*, vol. 39, no. 10, pp. 9588–9601, 2012.
- [22] Y.-L. Chen, C.-L. Chang, and C.-S. Yeh, “Emotion classification of youtube videos,” *Decision Support Systems*, vol. 101, pp. 40–50, 2017.
- [23] J. Cook, K. Kenthapadi, and N. Mishra, “Group chats on twitter,” in *Proceedings of the 22nd international conference on World Wide Web*. ACM, 2013, pp. 225–236.
- [24] H. Dai and Y. Song, “Neural aspect and opinion term extraction with mined rules as weak supervision,” *arXiv preprint arXiv:1907.03750*, 2019.
- [25] S. R. Das and M. Y. Chen, “Yahoo! for amazon: Sentiment extraction from small talk on the web,” *Management science*, vol. 53, no. 9, pp. 1375–1388, 2007.
- [26] K. Dave, S. Lawrence, and D. M. Pennock, “Mining the peanut gallery: opinion extraction and semantic classification of product reviews,” in *Proceedings of the Twelfth International World Wide Web Conference, WWW 2003, Budapest, Hungary, May 20-24, 2003*, 2003, pp. 519–528. [Online]. Available: <https://doi.org/10.1145/775152.775226>
- [27] E. Dimara, A. Bezerianos, and P. Dragicevic, “The attraction effect in information visualization,” *IEEE transactions on visualization and computer graphics*, vol. 23, no. 1, pp. 471–480, 2016.

- [28] X. Ding, B. Liu, and P. S. Yu, “A holistic lexicon-based approach to opinion mining,” in *Proceedings of the International Conference on Web Search and Data Mining*. ACM, 2008, pp. 231–240.
- [29] W. Du and S. Tan, “Building domain-oriented sentiment lexicon by improved information bottleneck,” in *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, ser. CIKM '09. New York, NY, USA: ACM, 2009, pp. 1749–1752. [Online]. Available: <http://doi.acm.org/10.1145/1645953.1646221>
- [30] M. Eirinaki, S. Pisal, and J. Singh, “Feature-based opinion mining and ranking,” *Journal of Computer and System Sciences*, vol. 78, no. 4, pp. 1175–1184, 2012.
- [31] A. El-Kishky, Y. Song, C. Wang, C. R. Voss, and J. Han, “Scalable topical phrase mining from text corpora,” *Proceedings of the VLDB Endowment*, vol. 8, no. 3, pp. 305–316, 2014.
- [32] O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates, “Methods for domain-independent information extraction from the web: An experimental comparison,” in *Proceedings of the 19th National Conference on Artificial Intelligence*, ser. AAAI'04. AAAI Press, 2004, pp. 391–398. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1597148.1597213>
- [33] “Facebook.com,” 2019. [Online]. Available: <https://www.facebook.com>
- [34] X. Fang and J. Zhan, “Sentiment analysis using product review data,” *Journal of Big Data*, vol. 1, no. 2, pp. 1–14, 2015.
- [35] C. Faria, I. Serra, and R. Girardi, “A domain-independent process for automatic ontology population from text,” *Science of Computer Programming*, vol. 95, pp. 26–43, 2014.

- [36] M. Fasciano and G. Lapalme, “Intentions in the coordinated generation of graphics and text from tabular data,” *Knowl. Inf. Syst.*, vol. 2, no. 3, pp. 310–339, Aug. 2000. [Online]. Available: <http://dx.doi.org/10.1007/PL00011645>
- [37] C. Fellbaum, *WordNet: An Electronic Lexical Database*. MIT Press, 1998.
- [38] W. Feng and J. Wang, “Incorporating heterogeneous information for personalized tag recommendation in social tagging systems,” in *Proceedings of the 18th Association for Computing Machinery SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 1276–1284.
- [39] D. Fensel, “Ontology-based knowledge management,” *Computer*, vol. 35, no. 11, pp. 56–59, Nov 2002.
- [40] P. Ferragina, F. Piccinno, and R. Santoro, “On analyzing hashtags in twitter,” in *International Conference on Web and Social Media (ICWSM)*. AAAI Press, 2015, pp. 110–119.
- [41] Forbes, *These 12 Astonishing Shopping Facts Perfectly Sum Up E-Commerce For 2016*, 2016, <https://www.forbes.com/sites/tompopomaronis/2016/12/19/these-12-astonishing-shopping-facts-perfectly-sum-up-e-commerce-for-2016/#37deb8f038be>.
- [42] N. Garg and I. Weber, “Personalized, interactive tag recommendation for flickr,” in *Proceedings of the 2008 Association for Computing Machinery conference on Recommender systems*. ACM, 2008, pp. 67–74.
- [43] A. Go, R. Bhayani, and L. Huang, “Twitter sentiment classification using distant supervision,” *CS224N Project Report, Stanford*, vol. 1, no. 2009, p. 12, 2009.

- [44] F. Godin, V. Slavkovikj, W. De Neve, B. Schrauwen, and R. Van de Walle, “Using topic models for twitter hashtag recommendation,” in *Proceedings of the 22nd International Conference on World Wide Web*. ACM, 2013, pp. 593–596.
- [45] N. L. Green, G. Carenini, S. Kerpedjiev, J. Mattis, J. D. Moore, and S. F. Roth, “Autobrief: An experimental system for the automatic generation of briefings in integrated text and information graphics,” *Int. J. Hum.-Comput. Stud.*, vol. 61, no. 1, pp. 32–70, Jul. 2004. [Online]. Available: <http://dx.doi.org/10.1016/j.ijhcs.2003.10.007>
- [46] K. S. Hasan and V. Ng, “Stance classification of ideological debates: Data, models, features, and constraints,” in *Proceedings of IJCNLP '13*, 2013, pp. 1348–1356.
- [47] —, “Predicting stance in ideological debate with rich linguistic knowledge,” in *Proceedings of the 24th International Conference on Computational Linguistics, Proceedings of the Conference: Posters, 8-15 December 2012, Mumbai, India*, 2012, pp. 451–460. [Online]. Available: <http://aclweb.org/anthology/C/C12/C12-2045.pdf>
- [48] —, “Why are you taking this stance? identifying and classifying reasons in ideological debates.” in *In proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing '14*, vol. 14, 2014, pp. 751–762.
- [49] V. Hatzivassiloglou and K. R. McKeown, “Predicting the semantic orientation of adjectives,” in *Proceedings of the 35th annual meeting of the association for computational linguistics and eighth conference of the european chapter of*

*the association for computational linguistics.* Association for Computational Linguistics, 1997, pp. 174–181.

- [50] P. Heymann, D. Ramage, and H. Garcia-Molina, “Social tag prediction,” in *Proceedings of the 31st annual international Association for Computing Machinery SIGIR conference on Research and development in information retrieval*. ACM, 2008, pp. 531–538.
- [51] A. Hogenboom, D. Bal, F. Frasincar, M. Bal, F. de Jong, and U. Kaymak, “Exploiting Emoticons in Polarity Classification of Text,” *Journal of Web Engineering*, vol. 14, no. 1, pp. 22–40, 2015.
- [52] E. Hovy, C.-Y. Lin *et al.*, “Automated text summarization in summarist,” *Advances in automatic text summarization*, vol. 14, 1999.
- [53] M. Hu, E.-P. Lim, and J. Jiang, “A probabilistic approach to personalized tag recommendation,” in *2010 IEEE Second International Conference on Social Computing*. IEEE, 2010, pp. 33–40.
- [54] M. Hu and B. Liu, “Mining and summarizing customer reviews,” in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '04. New York, NY, USA: ACM, 2004, pp. 168–177. [Online]. Available: <http://doi.acm.org/10.1145/1014052.1014073>
- [55] ———, “Mining opinion features in customer reviews,” in *Proceedings of the National Conference on Artificial Intelligence*. AAAI Press, 2004, pp. 755–760.
- [56] S. Huang and W. Cheng, “Discovering chinese sentence patterns for feature-based opinion summarization,” *Electronic Commerce Research and Applications*, vol. 14, no. 6, pp. 582–591, 2015.

- [57] C. Hung, “Word of mouth quality classification based on contextual sentiment lexicons,” *Information Processing & Management*, vol. 53, no. 4, pp. 751–763, 2017.
- [58] “imdb.com,” 2019. [Online]. Available: <http://www.imdb.com>
- [59] K. Järvelin and J. Kekäläinen, “Cumulated gain-based evaluation of ir techniques,” *Association for Computing Machinery Trans. Inf. Syst.*, vol. 20, no. 4, pp. 422–446, Oct. 2002. [Online]. Available: <http://doi.acm.org/10.1145/582415.582418>
- [60] R. Jäschke, L. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme, “Tag recommendations in folksonomies,” in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2007, pp. 506–514.
- [61] R. Johnson and T. Zhang, “Effective use of word order for text categorization with convolutional neural networks,” *arXiv preprint arXiv:1412.1058*, 2014.
- [62] R. Jones, *Internet and Text Slang Dictionary*, 2005–, <https://www.noslang.com/dictionary/>.
- [63] N. Kaji and M. Kitsuregawa, “Building lexicon for sentiment analysis from massive collection of HTML documents,” in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Prague, Czech Republic: Association for Computational Linguistics, Jun. 2007, pp. 1075–1083. [Online]. Available: <https://www.aclweb.org/anthology/D07-1115>
- [64] J. Kamps and M. Marx, “Words with attitude,” in *Proceedings of the 1st International Conference on Global WordNet*. Central Institute of Indian Languages, Mysore, India, 2002, pp. 332–341.

- [65] H. Kanayama and T. Nasukawa, “Fully automatic lexicon expansion for domain-oriented sentiment analysis,” in *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '06. Stroudsburg, PA, USA: Association for Computational Linguistics, 2006, pp. 355–363. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1610075.1610125>
- [66] Y.-B. Kang, P. D. Haghghi, and F. Burstein, “Cfinder: An intelligent key concept finder from text for ontology development,” *Expert Systems with Applications*, vol. 41, no. 9, pp. 4494–4504, 2014.
- [67] P. Karagoz, B. Kama, M. Ozturk, I. H. Toroslu, and D. Canturk, “A framework for aspect based sentiment analysis on turkish informal texts,” *Journal of Intelligent Information Systems*, Jun 2019. [Online]. Available: <https://doi.org/10.1007/s10844-019-00565-w>
- [68] L. Khan, D. McLeod, and E. Hovy, “Retrieval effectiveness of an ontology-based model for information selection,” *The VLDB Journal*, vol. 13, no. 1, pp. 71–85, Jan 2004. [Online]. Available: <https://doi.org/10.1007/s00778-003-0105-1>
- [69] S.-M. Kim and E. H. Hovy, “Crystal: Analyzing predictive opinions on the web.” in *In Proceedings of EMNLP-CoNLL '07*, 2007, pp. 1056–1064.
- [70] D. Klein and C. D. Manning, “Fast exact inference with a factored model for natural language parsing,” in *Advances in neural information processing systems*, 2003, pp. 3–10.
- [71] N. Kobayashi, K. Inui, and Y. Matsumoto, “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, 2007, pp. 1065–1074.



- [72] A. Konjengbam, N. Dewangan, N. Kumar, and M. Singh, “Aspect ontology based review exploration,” *Electronic Commerce Research and Applications*, vol. 30, pp. 62–71, 2018. [Online]. Available: <https://doi.org/10.1016/j.elerap.2018.05.006>
- [73] A. Konjengbam, S. Ghosh, N. Kumar, and M. Singh, “Debate stance classification using word embeddings,” in *International Conference on Big Data Analytics and Knowledge Discovery*. Springer, 2018, pp. 382–395.
- [74] P. Krejzl and J. Steinberger, “Uwb at semeval-2016 task 6: Stance detection.” in *Proceedings of SemEval@ NAACL-HLT*, 2016, pp. 408–412.
- [75] R. Krestel, P. Fankhauser, and W. Nejdl, “Latent dirichlet allocation for tag recommendation,” in *Proceedings of the third Association for Computing Machinery conference on Recommender systems*. ACM, 2009, pp. 61–68.
- [76] L.-W. Ku, Y.-T. Liang, and H.-H. Chen, “Opinion extraction, summarization and tracking in news and blog corpora,” in *AAAI Symposium on Computational Approaches to Analysing Weblogs (AAAI-CAAW)*, 2006, pp. 100–107.
- [77] A. Kumar, T. M. Sebastian *et al.*, “Sentiment analysis: A perspective on its past, present and future,” *International Journal of Intelligent Systems and Applications*, vol. 4, no. 10, pp. 1–14, 2012.
- [78] N. Kumar, K. Y. Mudda, G. Trishal, A. Konjengbam, M. Singh *et al.*, “Where to post: Routing questions to right community in community question answering systems,” in *Proceedings of the Association for Computing Machinery India Joint International Conference on Data Science and Management of Data*. ACM, 2019, pp. 136–142.

- [79] N. Kumar, R. Nagalla, T. Marwah, and M. Singh, “Sentiment dynamics in social media news channels,” *Online Social Networks and Media*, vol. 8, pp. 42–54, 2018.
- [80] N. Kumar, R. Utkoor, B. K. Appareddy, and M. Singh, “Generating topics of interests for research communities,” in *International Conference on Advanced Data Mining and Applications*. Springer, 2017, pp. 488–501.
- [81] T. K. Landauer, *Latent semantic analysis*. Wiley Online Library, 2006.
- [82] J. Lee and R. Goodwin, “Ontology management for large-scale enterprise systems,” *Electronic Commerce Research and Applications*, vol. 5, no. 1, pp. 2–15, 2006.
- [83] T. Lee, I.-h. Lee, S. Lee, S.-g. Lee, D. Kim, J. Chun, H. Lee, and J. Shim, “Building an operational product ontology system,” *Electronic Commerce Research and Applications*, vol. 5, no. 1, pp. 16–28, 2006.
- [84] O. Levy and Y. Goldberg, “Dependency-based word embeddings,” in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Baltimore, Maryland: Association for Computational Linguistics, Jun. 2014, pp. 302–308. [Online]. Available: <https://www.aclweb.org/anthology/P14-2050>
- [85] —, “Linguistic regularities in sparse and explicit word representations,” in *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*. Ann Arbor, Michigan: Association for Computational Linguistics, June 2014, pp. 171–180. [Online]. Available: <https://www.aclweb.org/anthology/W14-1618>
- [86] B. Liu, “Sentiment analysis and opinion mining,” *Synthesis lectures on human language technologies*, vol. 5, no. 1, pp. 1–167, 2012.

- [87] B. Liu, M. Hu, and J. Cheng, “Opinion observer: analyzing and comparing opinions on the web,” in *Proceedings of the International Conference on World Wide Web*. ACM, 2005, pp. 342–351.
- [88] H. Liu and P. Singh, “Conceptnet—a practical commonsense reasoning toolkit,” *BT technology journal*, vol. 22, no. 4, pp. 211–226, 2004.
- [89] K. Liu, B. Fang, and W. Zhang, “Speak the same language with your friends: augmenting tag recommenders with social relations,” in *Proceedings of the 21st Association for Computing Machinery conference on Hypertext and hypermedia*. ACM, 2010, pp. 45–50.
- [90] K. Liu, H. L. Xu, Y. Liu, and J. Zhao, “Opinion target extraction using partially-supervised word alignment model,” in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*. AAAI, 2013, pp. 2134–2140.
- [91] Q. Liu, B. Liu, Y. Zhang, D. S. Kim, and Z. Gao, “Improving opinion aspect extraction using semantic similarity and aspect associations,” in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI, 2016, pp. 2986–2992.
- [92] Y.-T. Lu, S.-I. Yu, T.-C. Chang, and J. Y.-j. Hsu, “A content-based method to enhance tag recommendation,” in *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.
- [93] Y. Lu, H. Wang, C. Zhai, and D. Roth, “Unsupervised discovery of opposing opinion networks from forum discussions,” in *Proceedings of CIKM '12*. ACM, 2012, pp. 1642–1646.
- [94] Y. Lu and C. Zhai, “Opinion integration through semi-supervised topic modeling,” in *Proceedings of the 17th International Conference on World Wide*

- Web*, ser. WWW '08. New York, NY, USA: ACM, 2008, pp. 121–130. [Online]. Available: <http://doi.acm.org/10.1145/1367497.1367514>
- [95] Y. Lu, C. Zhai, and N. Sundaresan, “Rated aspect summarization of short comments,” in *Proceedings of the International Conference on World Wide Web*. New York, USA: ACM, 2009, pp. 131–140.
- [96] A. Malucelli, D. Palzer, and E. Oliveira, “Ontology-based services to help solving the heterogeneity problem in e-commerce negotiations,” *Electronic Commerce Research and Applications*, vol. 5, no. 1, pp. 29–43, 2006.
- [97] G. V. Menezes, J. M. Almeida, F. Belém, M. A. Gonçalves, A. Lacerda, E. S. De Moura, G. L. Pappa, A. Veloso, and N. Ziviani, “Demand-driven tag recommendation,” in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2010, pp. 402–417.
- [98] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen *et al.*, “Mllib: Machine learning in apache spark,” *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1235–1241, 2016.
- [99] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Proceedings of the 26th Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2013, pp. 3111–3119. [Online]. Available: <http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf>
- [100] G. Mishne, “Autotag: a collaborative approach to automated tag assignment for weblog posts,” in *Proceedings of the 15th international conference on World Wide Web*. ACM, 2006, pp. 953–954.

- [101] S. Mohammad, C. Dunne, and B. Dorr, “Generating high-coverage semantic orientation lexicons from overtly marked words and a thesaurus,” in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. Singapore: Association for Computational Linguistics, Aug. 2009, pp. 599–608. [Online]. Available: <https://www.aclweb.org/anthology/D09-1063>
- [102] S. M. Mohammad, “# emotional tweets,” in *Proceedings of SemEval ’12*. ACL, 2012, pp. 246–255.
- [103] S. M. Mohammad and P. D. Turney, “Crowdsourcing a word-emotion association lexicon,” *Computational Intelligence*, vol. 29, no. 3, pp. 436–465, 2013.
- [104] R. Moraes, J. F. Valiati, and W. P. G. Neto, “Document-level sentiment classification: An empirical comparison between svm and ann,” *Expert Systems with Applications*, vol. 40, no. 2, pp. 621–633, 2013.
- [105] A. Moreo, M. Romero, J. Castro, and J. M. Zurita, “Lexicon-based comments-oriented news sentiment analyzer system,” *Expert Systems with Applications*, vol. 39, no. 10, pp. 9166–9180, 2012.
- [106] A. Mukherjee and B. Liu, “Aspect extraction through semi-supervised modeling,” in *Proceedings of the Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Stroudsburg, USA: ACL, 2012, pp. 339–348.
- [107] S. Mukherjee and S. Joshi, “Sentiment aggregation using conceptnet ontology,” in *Proceedings of the Sixth International Joint Conference on Natural Language Processing*. Asian Federation of Natural Language Processing, 2013, pp. 570–578.
- [108] A. Murakami and R. Raymond, “Support or oppose?: classifying positions in online debates from reply activities and opinion expressions,” in *Proceedings of*

*the International Conference on Computational Linguistics '10: Posters*. ACL, 2010, pp. 869–875.

- [109] P. Nakov, A. Ritter, S. Rosenthal, F. Sebastiani, and V. Stoyanov, “Semeval-2016 task 4: Sentiment analysis in twitter,” in *Proceedings of the 10th international workshop on semantic evaluation (semeval-2016)*, 2016, pp. 1–18.
- [110] P. Norvig, “How to write a spelling corrector,” *De: <http://norvig.com/spell-correct.html>*, 2007.
- [111] B. Pang, L. Lee, and S. Vaithyanathan, “Thumbs up?: Sentiment classification using machine learning techniques,” in *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing, EMNLP 2002, Philadelphia, PA, USA, July 6-7, 2002*, ser. EMNLP '02. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 79–86. [Online]. Available: <https://doi.org/10.3115/1118693.1118704>
- [112] J. Pavlopoulos and I. Androutsopoulos, “Aspect term extraction for sentiment analysis: New datasets, new evaluation measures and an improved unsupervised method,” *Proceedings of LASMEACL '14*, pp. 44–52, 2014.
- [113] M. Pawlik and N. Augsten, “Tree edit distance: Robust and memory-efficient,” *Information Systems*, vol. 56, no. C, pp. 157–173, 2016.
- [114] S. R. Pfohl and O. Triebe, “Stance detection for the fake news challenge with attention and conditional encoding,” in *Stanford CS224d Deep Learning for NLP final project*, 2017.
- [115] M. Pontiki, D. Galanis, H. Papageorgiou, I. Androutsopoulos, S. Manandhar, A.-S. Mohammad, M. Al-Ayyoub, Y. Zhao, B. Qin, O. De Clercq *et al.*, “Semeval-2016 task 5: Aspect based sentiment analysis,” in *Proceedings of the*

- 10th International Workshop on Semantic Evaluation (SemEval-2016)*. ACL, 2016, pp. 19–30.
- [116] A.-M. Popescu and O. Etzioni, “Extracting product features and opinions from reviews,” in *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, ser. HLT ’05. Stroudsburg, PA, USA: Association for Computational Linguistics, 2005, pp. 339–346. [Online]. Available: <https://doi.org/10.3115/1220575.1220618>
- [117] —, “Extracting product features and opinions from reviews,” in *Proceeding of the Natural language processing and text mining*. Springer, 2007, pp. 9–28.
- [118] S. Poria, E. Cambria, and A. Gelbukh, “Aspect extraction for opinion mining with a deep convolutional neural network,” *Knowledge-Based Systems*, vol. 108, pp. 42–49, 2016.
- [119] G. Qiu, B. Liu, J. Bu, and C. Chen, “Opinion word expansion and target extraction through double propagation,” *Computational linguistics*, vol. 37, no. 1, pp. 9–27, 2011.
- [120] “quora.com,” 2019. [Online]. Available: <https://www.quora.com>
- [121] A. Radford, R. Jozefowicz, and I. Sutskever, “Learning to generate reviews and discovering sentiment,” *arXiv preprint arXiv:1704.01444*, 2017.
- [122] A. Rajadesingan and H. Liu, “Identifying users with opposing opinions in twitter debates,” in *Proceedings of SBP-BRiMS ’14*. Springer, 2014, pp. 153–160.
- [123] P. Rajendran, D. Bollegala, and S. Parsons, “Contextual stance classification of opinions: A step towards enthymeme reconstruction in online reviews,” *ACL 2016*, p. 31, 2016.

- [124] D. Ramage, S. Dumais, and D. Liebling, “Characterizing microblogs with topic models,” in *Fourth International AAAI Conference on Weblogs and Social Media*, 2010.
- [125] K. Ravi and V. Ravi, “A survey on opinion mining and sentiment analysis: tasks, approaches and applications,” *Knowledge-Based Systems*, vol. 89, pp. 14–46, 2015.
- [126] S. Rendle and L. Schmidt-Thieme, “Pairwise interaction tensor factorization for personalized tag recommendation,” in *Proceedings of the third Association for Computing Machinery international conference on Web search and data mining*. ACM, 2010, pp. 81–90.
- [127] I. S. Ribeiro, R. L. Santos, M. A. Gonçalves, and A. H. Laender, “On tag recommendation for expertise profiling: A case study in the scientific domain,” in *Proceedings of the Eighth Association for Computing Machinery International Conference on Web Search and Data Mining*. ACM, 2015, pp. 189–198.
- [128] V. Rus, M. C. Lintean, R. Banjade, N. B. Niraula, and D. Stefanescu, “Semilar: The semantic similarity toolkit,” in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. ACL, 2013, pp. 163–168.
- [129] C. Sauper and R. Barzilay, “Automatically generating wikipedia articles: A structure-aware approach,” in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ser. ACL ’09. Stroudsburg, PA, USA: Association for Computational Linguistics, 2009, pp. 208–216. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1687878.1687909>



- [130] C. Scaffidi, K. Bierhoff, E. Chang, M. Felker, H. Ng, and C. Jin, “Red opal: product-feature scoring from reviews,” in *Proceedings of the 8th Association for Computing Machinery conference on Electronic commerce*. New York, USA: ACM, 2007, pp. 182–191.
- [131] K. Schouten and F. Frasincar, “Survey on aspect-level sentiment analysis,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 3, pp. 813–830, 2015.
- [132] A. T. G. Schreiber, B. Dubbeldam, J. Wielemaker, and B. Wielinga, “Ontology-based photo annotation,” *IEEE Intelligent Systems*, vol. 16, no. 3, pp. 66–74, May 2001. [Online]. Available: <https://doi.org/10.1109/5254.940028>
- [133] K. P. P. Shein, “Ontology based combined approach for sentiment classification,” in *Proceedings of the International Conference on Communications and information technology*. Stevens Point, USA: WSEAS, 2009, pp. 112–115.
- [134] Z. Shen, S. Arslan Ay, S. H. Kim, and R. Zimmermann, “Automatic tag generation and ranking for sensor-rich outdoor videos,” in *Proceedings of the 19th Association for Computing Machinery international conference on Multimedia*. ACM, 2011, pp. 93–102.
- [135] B. Shneiderman, “Tree visualization with tree-maps: 2-d space-filling approach,” *ACM Trans. Graph.*, vol. 11, no. 1, pp. 92–99, Jan. 1992. [Online]. Available: <http://doi.acm.org/10.1145/102377.115768>
- [136] X. Si and M. Sun, “Tag-lda for scalable real-time tag recommendation,” *JOURNAL OF INFORMATION & COMPUTATIONAL SCIENCE*, vol. 6, no. 2, pp. 1009–1016, 2008.

- [137] B. Sigurbjörnsson and R. Van Zwol, “Flickr tag recommendation based on collective knowledge,” in *Proceedings of the 17th international conference on World Wide Web*. ACM, 2008, pp. 327–336.
- [138] M. Singh and J. HV, *Effective faceted browsing*. Indian Institute of Technology Hyderabad, 2014.
- [139] R. Singh and D. Toshniwal, “Location prediction using sentiments of twitter users,” in *International Conference on Big Data Analytics and Knowledge Discovery*. Springer, 2018, pp. 98–108.
- [140] S. Somasundaran and J. Wiebe, “Recognizing stances in online debates,” in *Proceedings of ACL-IJCNLP ’09*. ACL, 2009, pp. 226–234.
- [141] Y. Song, L. Zhang, and C. L. Giles, “Automatic tag recommendation algorithms for social recommender systems,” *Association for Computing Machinery Transactions on the Web (TWEB)*, vol. 5, no. 1, p. 4, 2011.
- [142] S. Sood, S. Owsley, K. J. Hammond, and L. Birnbaum, “Tagassist: Automatic tag suggestion for blog posts.” in *ICWSM*, 2007.
- [143] D. Sridhar, J. Foulds, B. Huang, L. Getoor, and M. Walker, “Joint models of disagreement and stance in online debate,” in *Proceedings of IJCNLP ’15*, vol. 1, 2015, pp. 116–125.
- [144] “stackexchange.com,” 2019. [Online]. Available: <https://eee.stackexchange.com>
- [145] Statista, *Number of Digital Buyers Worldwide from 2014 to 2021 (in billions)*, 2018, <https://www.statista.com/statistics/251666/number-of-digital-buyers-worldwide/>.

- [146] S. Sun, G. Kong, and C. Zhao, “Polarity words distance-weight count for opinion analysis of online news comments,” *Procedia Engineering*, vol. 15, pp. 1916–1920, 2011.
- [147] M. Thomas, B. Pang, and L. Lee, “Get out the vote: Determining support or opposition from congressional floor-debate transcripts,” in *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing '06*. ACL, 2006, pp. 327–335.
- [148] H. Toba, Z.-Y. Ming, M. Adriani, and T.-S. Chua, “Discovering high quality answers in community question answering archives using a hierarchy of classifiers,” *Information Sciences*, vol. 261, pp. 101–115, 2014.
- [149] T. U. Tran, H. T. T. Hoang, and H. X. Huynh, “Aspect extraction with bidirectional gru and crf,” in *2019 IEEE-RIVF International Conference on Computing and Communication Technologies (RIVF)*. IEEE, 2019, pp. 1–5.
- [150] A. J. Trappey, C. V. Trappey, and C.-Y. Wu, “Automatic patent document summarization for collaborative knowledge systems and services,” *Journal of Systems Science and Systems Engineering*, vol. 18, no. 1, pp. 71–94, Mar 2009. [Online]. Available: <https://doi.org/10.1007/s11518-009-5100-7>
- [151] P. D. Turney, “Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews,” in *Proceedings of the annual meeting on Association for Computational Linguistics*. Stroudsburg, USA: ACL, 2002, pp. 417–424.
- [152] P. D. Turney and M. L. Littman, “Measuring praise and criticism: Inference of semantic orientation from association,” *Association for Computing Machinery Transactions on Information Systems (TOIS)*, vol. 21, no. 4, pp. 315–346, 2003.
- [153] “twitter.com,” 2019. [Online]. Available: <https://www.twitter.com>

- [154] A. Valitutti, C. Strapparava, and O. Stock, “Developing affective lexical resources,” *PsychNology Journal*, vol. 2, pp. 61–83, January 2004. [Online]. Available: <https://www.microsoft.com/en-us/research/publication/developing-affective-lexical-resources/>
- [155] P. Venetis, G. Koutrika, and H. Garcia-Molina, “On the selection of tags for tag clouds,” in *Proceedings of the fourth Association for Computing Machinery international conference on Web search and data mining*. ACM, 2011, pp. 835–844.
- [156] M. A. Walker, P. Anand, R. Abbott, and R. Grant, “Stance classification using dialogic properties of persuasion,” in *Proceedings of ACL ’12*. ACL, 2012, pp. 592–596.
- [157] M. A. Walker, P. Anand, R. Abbott, J. E. F. Tree, C. Martell, and J. King, “That is your evidence?: Classifying stance in online political debate,” *Decision Support Systems*, vol. 53, no. 4, pp. 719–729, 2012.
- [158] H. Wang, Y. Lu, and C. Zhai, “Latent aspect rating analysis on review text data: a rating regression approach,” in *Proceedings of the 16th Association for Computing Machinery SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2010, pp. 783–792.
- [159] J. Wang, L. Hong, and B. D. Davison, “Rsd09: Tag recommendation using keywords and association rules.” in *DC@ PKDD/ECML*, 2009.
- [160] R. Wang, D. Zhou, M. Jiang, J. Si, and Y. Yang, “A survey on opinion mining: From stance to product aspect,” *IEEE Access*, vol. 7, pp. 41 101–41 124, 2019.
- [161] X. Wang, A. McCallum, and X. Wei, “Topical n-grams: Phrase and topic discovery, with an application to information retrieval,” in *icdm*. IEEE, 2007, pp. 697–702.

- [162] C. Ware, *Information visualization: perception for design*. Elsevier, 2012.
- [163] J. M. Wiebe, R. F. Bruce, and T. P. O’Hara, “Development and use of a gold-standard data set for subjectivity classifications,” in *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, ser. ACL ’99. Stroudsburg, PA, USA: Association for Computational Linguistics, 1999, pp. 246–253. [Online]. Available: <https://doi.org/10.3115/1034678.1034721>
- [164] G. Williams and S. Anand, “Predicting the polarity strength of adjectives using wordnet,” in *Third International AAAI Conference on Weblogs and Social Media*, 2009. [Online]. Available: <https://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/214>
- [165] L. Wu, L. Yang, N. Yu, and X.-S. Hua, “Learning to tag,” in *Proceedings of the 18th international conference on World wide web*. ACM, 2009, pp. 361–370.
- [166] Y. Wu, Q. Zhang, X. Huang, and L. Wu, “Phrase dependency parsing for opinion mining,” in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*. Association for Computational Linguistics, 2009, pp. 1533–1541.
- [167] X. Xia, D. Lo, X. Wang, and B. Zhou, “Tag recommendation in software information sites,” in *Proceedings of the 10th Working Conference on Mining Software Repositories*. IEEE Press, 2013, pp. 287–296.
- [168] L. Xu, K. Liu, J. Zhao *et al.*, “Joint opinion relation detection using one-class deep neural network.” in *Proceedings of the the International Conference on Computational Linguistics ’14*, vol. 14, 2014, pp. 677–687.

- [169] A. Yadollahi, A. G. Shahraki, and O. R. Zaiane, “Current state of text sentiment analysis from opinion to emotion mining,” *Association for Computing Machinery Computing Surveys (CSUR)*, vol. 50, no. 2, p. 25, 2017.
- [170] “answers.yahoo.com,” 2019. [Online]. Available: <https://www.answers.yahoo.com>
- [171] B. Yang and S. Manandhar, “Tag-based expert recommendation in community question answering,” in *2014 IEEE/Association for Computing Machinery International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)*. IEEE, 2014, pp. 960–963.
- [172] T. Yano, N. A. Smith, and J. D. Wilkerson, “Textual predictors of bill survival in congressional committees,” in *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2012, pp. 793–802.
- [173] Y. Yin, F. Wei, L. Dong, K. Xu, M. Zhang, and M. Zhou, “Unsupervised word and dependency path embeddings for aspect term extraction,” in *Proceedings of IJCAI-ECAI ’16*. AAAI Press, 2016, pp. 2979–2985.
- [174] “Youtube.com,” 2019. [Online]. Available: <https://www.youtube.com>
- [175] J. YU, E. REITER, J. HUNTER, and C. MELLISH, “Choosing the content of textual summaries of large time-series data sets,” *Natural Language Engineering*, vol. 13, no. 1, p. 25–49, 2007.
- [176] L. Zhang, S. Wang, and B. Liu, “Deep learning for sentiment analysis: A survey,” *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1253, 2018.

- [177] N. Zhang, Y. Zhang, and J. Tang, “A tag recommendation system based on contents,” *ECML PKDD Discovery Challenge 2009 (DC09)*, p. 285, 2009.
- [178] X. Zhang, J. Zhao, and Y. LeCun, “Character-level convolutional networks for text classification,” in *Advances in neural information processing systems*, 2015, pp. 649–657.
- [179] Y. Zhang and Z. Lin, “Predicting the helpfulness of online product reviews: A multilingual approach,” *Electronic Commerce Research and Applications*, vol. 27, pp. 1–10, 2018.
- [180] Zhang Duo, Li Juan-Zi, and Xu Bin, “Web service annotation using ontology mapping,” in *IEEE International Workshop on Service-Oriented System Engineering (SOSE’05)*, Oct 2005, pp. 235–242.
- [181] L. Zhou and P. Chaovalit, “Ontology-supported polarity mining,” *Journal of the Association for Information Science and Technology*, vol. 59, no. 1, pp. 98–110, 2008.
- [182] W. Zhou and W. Duan, “Online user reviews, product variety, and the long tail: An empirical investigation on online software downloads,” *Electronic Commerce Research and Applications*, vol. 11, no. 3, pp. 275–289, 2012.
- [183] L. Zhuang, F. Jing, and X.-Y. Zhu, “Movie review mining and summarization,” in *Proceedings of the 15th Association for Computing Machinery International Conference on Information and Knowledge Management*. New York, USA: ACM, 2006, pp. 43–50.
- [184] A. Zubiaga, E. Kochkina, M. Liakata, R. Procter, M. Lukasik, K. Bontcheva, T. Cohn, and I. Augenstein, “Discourse-aware rumour stance classification in social media using sequential classifiers,” *Information Processing & Management*, vol. 54, no. 2, pp. 273–290, 2018.

# List of Publications

## Journals:

- **Konjengbam Anand**, Nagendra Kumar, and Manish Singh. *Unsupervised Tag Recommendation for Popular and Cold Products*, *Journal of Intelligent Information Systems*. 53 (2019): 1-22.
- **Konjengbam Anand**, Neelesh Dewangan, Nagendra Kumar, and Manish Singh. *Aspect ontology based review exploration*, *Electronic Commerce Research and Applications* 30 (2018): 62-71.

## Conferences:

- **Konjengbam Anand**, Subrata Ghosh, Nagendra Kumar, and Manish Singh. *Debate Stance Classification Using Word Embeddings*, In *International Conference on Big Data Analytics and Knowledge Discovery*, pp. 382-395. Springer, Cham, 2018.
- Nagendra Kumar, Karthik Yadav Mudda, Gayam Trishal, **Konjengbam Anand**, and Manish Singh. *Where to Post: Routing Questions to Right Community in Community Question Answering Systems*, In *ACM India Joint International Conference on Data Science and Management of Data*, pp. 136-142. ACM, 2019.
- Subrata Ghosh, **Konjengbam Anand**, Sailaja Rajanala, A. Bharath Reddy, and Manish Singh. *Unsupervised stance classification in online debates*, In *ACM India Joint International Conference on Data Science and Management of Data*, pp. 30-36. ACM, 2018.
- Nagendra Kumar, Yash Chandarana, **Konjengbam Anand**, and Manish Singh. *Using social media for word-of-mouth marketing*, In *International Con-*



*ference on Big Data Analytics and Knowledge Discovery*, pp. 391-406. Springer, Cham, 2017.

- Prathyusha Thammineni, Vipul Jindal, Saurabh Gangwar, **Konjengbam Anand**, and Kotaro Kataoka. *SPACE: An Empirical Approach Towards a User-Centric Smart Campus*, In *International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, pp. 35-41. IARIA, 2018