12-2018

Comparison mining from text

Maksim TKACHENKO
Singapore Management University, mtkachenko.2015@phdis.smu.edu.sg

COMPARISON MINING FROM TEXT

MAKSIM TKACHENKO

SINGAPORE MANAGEMENT UNIVERSITY

2018

Comparison Mining from Text

by

Maksim Tkachenko

Submitted to School of Information Systems in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Information Systems

Dissertation Committee:

Hady W. Lauw (Supervisor/Chair)
Associate Professor of Information Systems
Singapore Management University

Jiang Jing
Associate Professor of Information Systems
Singapore Management University

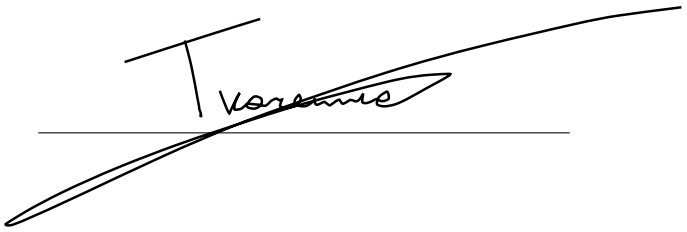
Zheng Baihua
Associate Professor of Information Systems
Singapore Management University

Xiaoli Li
Department Head (Data Analytics)
Institute for Infocomm Research

Singapore Management University
2018

I hereby declare that this PhD dissertation is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in this dissertation.

This PhD dissertation has also not been submitted for any degree in any university previously.



Maksim Tkachenko

14 November 2018

COMPARISON MINING FROM TEXT

Maksim Tkachenko

ABSTRACT

Online product reviews are important factors of consumers' purchase decisions. They invade more and more spheres of our life, we have reviews on books, electronics, groceries, entertainments, restaurants, travel experiences, etc. More than 90 percent of consumers read online reviews before they purchase products as reported by various consumers surveys. This observation suggests that product review information enhances consumer experience and helps them to make better-informed purchase decisions. There is an enormous amount of online reviews posted on e-commerce platforms, such as Amazon, Apple, Yelp, TripAdvisor. They vary in information and may be written with different experiences and preferences.

If online opinions are indeed important in many spheres of our lives, then their systematic analysis is a real-life problem. Due to an enormous amount of opinions scattered across the Web, a handcrafted analysis seems to carry an inadmissible cost of time and efforts. An alternative to consider is an automated or, more appropriately, semi-automated analysis conducted by computers as an assistance to human analysts. Text processing applications have received much attention in the past three decades and have been shown successful for language understanding.

Comparison mining aims at understanding opinion mining problems when multiple entities are present simultaneously. This includes, but not limited to deriving similarities and differences between entities and discovering information about the entity relations. The entities may be products, individuals, issues, etc. The notion of comparison tangles in in a form of joint evaluative statements, such as *'I think A is better than B'*, *'I think A is a good alternative to B'*, and introduces new research

questions, similar and yet different from traditional opinion mining. How do we find these statements in a review? How do we interpret these statements? How do we make sense of thousands of such comparisons? In this study, we seek to answer these questions and propose a set of related computational solutions.

First, we investigate a comparison identification problem and cast it as a relation extraction problem. Within the relation extraction setup, we develop a new approach for identifying comparative relations. The formal investigation of the syntactic structure of comparative statements leads us to a kernel-based approach, which relies on the dependency structure of sentences. The proposed method shows state-of-the-art results for the comparison identification problem.

Second, we explore intrinsic properties of a comparative corpus to derive a joint model for comparison interpretation and aggregation. At the level of comparisons, the model seeks to derive the comparison outcome of a statement, i.e., which entity is preferred by the writer. At the aggregated level, it seeks to understand the overall ranking of the entities in a corpus of comparisons. The proposed model is shown to be superior to the approaches that tackle each level separately. An empirical evaluation demonstrates its effectiveness on real-world datasets.

Third, we look at the phenomenon of comparison disagreement, i.e., different users may have different preferences over the same set of entities. To capture this diversity, we propose a model for preference clustering and demonstrate its effectiveness and utility.

Fourth, we propose a method for explaining entity comparisons, when entities are identified by their textual representations. CompareLDA, a supervised topic model, is employed to align topics, distributions of co-occurring words, with comparisons, so that the topics are indicative of the "better" and "worse" entities. Through an empirical evaluation, we show that the proposed model is more effective for capturing comparisons than alternative supervised topic models.

All the proposed methods form substantial contribution within the comparison mining research and facilitate a better understanding of the opinion language.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENT

I would like to express my deep gratitude to Hady Lauw, whose support and trust have been keeping me on track towards making this work possible. I believe he has given me the best advice one could get through all the years we have been working together. Thank you!

I am grateful to have terrific committee members Jing Jiang, Baihua Zheng, and Xiaoli Li who have provided me with their valuable and prompt feedback.

I would like to thank Boris Novikov who has set me on the research track during my university days and who has been very supportive of me. Thank you Andrey Simanovsky, Alexander Ulanov, and Natalia Vassilieva for your guidance during my early research career. I believe you could find a touch of your ideas in this work.

It has been a great pleasure and enjoyment to spend time with my SMU colleagues. I am grateful to have you by my side. I want to thank all my friends for their support and understanding.

I wish to thank my parents and my brother for their love and encouragement in pursuing my dreams.

CHAPTER 1

INTRODUCTION

1.1 Comparison Mining

Nowadays, whenever I want to buy a new electronic device like laptop, camera, or MP3 player, I most definitely browse the Web in search of related product information. Online reviews, a form of digital word-of-mouth, have become one of the most common and important sources of such information for me and, I believe, for the modern consumer. E-commerce platforms, turning into a type of social networks, facilitate interaction between consumers, exchange of product-related information, and help to make informed purchases.

Online reviewing increasingly affects more and more spheres of consumers' life, reviewing is available for books, electronics, music, entertainment, groceries (e.g., Amazon, iTunes Store) as well as for various services that provide restaurant, travel or other kind of experiences (e.g., Yelp, TripAdvisor, Booking.com). Consumers' experiences and opinions shared online draw a considerable attention from companies, because the ability to quickly react and address consumers' concerns and requests is crucial for success in a competitive environment. Online platforms make communication between consumers and businesses unmediated, which in turn can be analysed and put into good service.

Public opinions on controversial matters have always been of interest in politics. The rise of social media (e.g., Twitter, Facebook, Reddit) essentially has changed

the way political communication works. Online users are able to connect directly to politicians and express their approval or disapproval by simply pressing "like button" or posting a comment. These "likes" and comments can then be used to approximate election results.

If online opinions are indeed important in many spheres of our lives and help to make informed decisions, then their systematic analysis is a real-life problem. Due to an enormous amount of opinions scattered across the Web, their handcrafted analysis seems to carry an inadmissible cost of time and efforts. An alternative to consider is an automated or, more appropriately, semi-automated analysis conducted by computers as an assistance to human analysts.

Text processing applications have received much attention in the past three decades and have been shown successful in a number of language understanding tasks: semantic parsing [175], machine translation [189], information extraction [23], etc. Among all of them, opinion mining [102], which deals with written opinions, is the most relevant study to the topic of this work (see Chapter 2).

Comparison mining, a part of opinion mining, aims at understanding opinion analysis problems, when multiple entities are present simultaneously. For example, one may look for the difference between two entities in terms of their description and relative appraisal. Written opinions tangle in in a form of evaluative statements (*'I think A is better than B'* or *'I think A is a good alternative to B'*) and introduce new research questions, similar and yet different from traditional opinion mining: How do we interpret these statements? How do we make sense of thousands of such comparisons? Could we pinpoint the difference between entities? Traditional research on opinion mining focuses contrarily on separate entity evaluations like *'I think A is good'* or *'I think build quality of B is excellent'*. For comparison mining, the notion of relations between entities becomes central.

In this study, we develop components of an end-to-end system, which aim to answer some of the aforementioned questions in a fully computational manner. First, we formally investigate the syntactic structure of comparative statements,

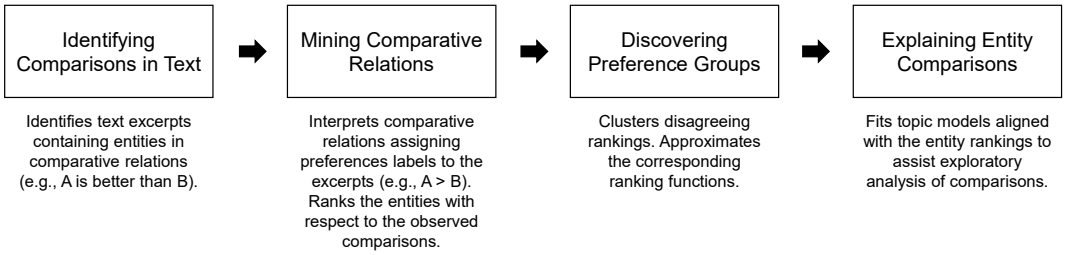


Figure 1.1: Component-based Overview.

which leads to a kernel-based approach for extracting comparative sentences from text (Chapter 3). Second, we explore intrinsic properties of a comparative corpus to derive a joint generative model for comparison interpretation and aggregation (Chapter 4). Third, we look at the phenomenon of comparison disagreement among preferences and develop a clustering technique for discovering preference groups within a population of users (Chapter 5). Fourth, with the aim to provide an intuitive interpretation of entity comparisons, we develop a supervised topic model for entity comparisons (Chapter 6). Figure 1.1 illustrates the structure of this work in a component-based manner, each box corresponds to a chapter.

Comparisons form about 10% of sentences in a typical review [86]. Thus, to analyse comparison at the large scale, we first need to be able to locate them in a text collection. A handcrafted approach does not scale, as it requires considerable time and is labor-intensive. We turn to machine learning methods. The problem of comparison identification, which was often cast as sentence classification (*comparative sentence* vs. *non comparative sentence*), turned out to be more interesting. For example, a comparative sentence can have three entity mentions, but only two comparisons. Sentence *'A is better than B and C'* compares A and B, A and C, but does not compare B and C. Therefore, comparison identification can be casted as a relation extraction problem, where given an excerpt and a couple of entities within it, a method is required to determine if a comparison between entities is present. We analysed the syntactic structure of comparisons and developed a new kernel approach to measure sentence similarity. Comparisons impose heavy constraints on the sentence structure, even if the words in a sentence vary a lot, their dependency

structure often remains the same. The novel dependency-based kernel helps us to identify comparisons within a text collection with a higher performance and greater granularity.

Having a corpus of comparisons, we want to interpret information it contains. Each comparison can be interpreted individually, one can induce an entity ranking between entity mentions based on the words a sentence contains. On the surface, for a pair of entities, the problem is a binary classification problem, we want to know whether the first or the second-mentioned entity of a comparison is preferred [77]. Then all the individual comparison predictions can be combined to induce preferences at the corpus level. Ranking aggregation models can be used for this task [91]. We systematically explore these two levels of interpretation within a joint probabilistic graphical model and show that combined interpretation allows to achieving better interpretation performance at both.

If we pursue the line of the corpus interpretation, then we may find that different users have different opinions of how their preferences should be distributed: one may state *'A is better than B'* and the other *'A is worse than B'*. Consumers looking for a new laptop may value different features: a gamer would look for a high performing laptop, a businessman would look for something more ergonomic and light at the expense of the computing power. To make sound conclusions about how people compare things, we need to consider different preference groups of users and be able to differentiate one from another. Armed with this observation, we propose a clustering method for deriving and describing such preference groups.

A comparison choice often needs to be accompanied with a rationale if we seek to gain some insight about the decision. When a formal set of entity features is observed (e.g., price, compactness), this task can be reduced to the interpretation of regression parameters or other intuitive models. However, dealing with text usually requires additional inference. We need to make sense of the word distributions. Assuming that entity comparisons come from the same preference group (which can be achieved by preference group clustering), we aim to find an explanation

for the comparisons through the topics. We propose a supervised topic model for document comparisons, which is based on intuition that we can align topics to be predictive of comparison decisions. Thus, reviewing the derived topics, one can form a better understanding of the subjects and objects of the comparisons.

1.2 Contributions

The contributions of this thesis are:

- A new method for tackling comparison identification problem. We propose a different problem formulation shifting focus from comparative sentence identification to *comparison identification*. Given a pair of entities within a sentence, we are interested to answer if there is any comparative relation between two entities. This point of view frames the identification as a form of relation extraction. Due to the specificity of the problem domain, which imposes well-defined syntactic structure on comparative sentences, we propose a novel relation extraction method. The method operates on sentence dependency trees and measures sentence similarity via Skip-node, a kernel function. Skip-node captures both the syntax and lexicon of a sentence. An extensive evaluation on manually annotated data shows effectiveness and efficiency of the proposed approach.

  - **Maksim Tkachenko** and Hady W. Lauw, *A Convolution Kernel Approach to Identifying Comparisons in Text*, Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, ACL, 2015.

- **Compa**rative **Re**lation **Ge**nerative **M**odel (CompareGem), a model for comparison interpretation. CompareGem connects two levels of interpretations: sentence level and corpus level. At the sentence level it induces meaning of each sentence or each comparison within a sentence. At the corpus level, it summarizes individual comparison interpretations to build a complete ranking of compared entities. The ranking is induced with respect to entity aspect,

e.g., product functionality, portability, etc. The latent factors of the model (i.e., comparison outcomes and aspects) can be inferred automatically in unsupervised or supervised fashion. Through experiments on real datasets, we demonstrate effectiveness of the proposed approach and show that the model reflects innate properties of comparison corpora.

- **Maksim Tkachenko** and Hady W. Lauw, *Generative Modeling of Entity Comparisons in Text*, Proceedings of the 23rd ACM International Conference on Information and Knowledge Management, CIKM, 2014;
- **Maksim Tkachenko** and Hady W. Lauw, *Comparative Relation Generative Model*, IEEE Transactions on Knowledge and Data Engineering, TKDE, 2017.

- An approach to model preference groups within a population of rankers. The model is called Plackett-Luce regressions mixture, a graphical model, which aims to discover latent preference groups and approximate their preference ranking functions. A comparison is a special case of a ranking involving only a pair of entities. We propose an efficient algorithm to fit the model parameters and to induce preferences for a set of previously unobserved entities. The model permits exploration and exploitation use. Its parameters can be easily interpreted. We demonstrate effectiveness of the model through a thoughtful empirical evaluation.

  - **Maksim Tkachenko** and Hady W. Lauw, *Plackett-Luce Regression Mixture Model for Heterogeneous Rankings*, Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM, 2016.

- CompareLDA, a supervised topic model for document comparisons. We want to infer latent semantics associated with comparison signals of entities. For example, for a set of graded essays, we want to know what makes a better essay; for products, we want to know what makes a better product based on their reviews. Given documents describing these entities along with their pairwise comparisons, the topic modeling approach is used to derive topics that comply with a comparison supervision. A topic is a distribution of words that

often co-occur with each other. These co-occurrence regularities may shed some light on the comparisons decisions. We derive a variational inference procedure for CompareLDA. Our experimental results show benefits of using the proposed model over alternative supervised topic models.

- **Maksim Tkachenko** and Hady W. Lauw, *CompareLDA: A Topic Model for Document Comparison*, Proceedings of the 33rd AAAI Conference on Artificial Intelligence, AAAI, 2019.

CHAPTER 2

RELATED WORK

2.1 Opinion Mining

Often, we want to understand others' opinions to make a good decision. This might be a decision on which doctor to visit, which product to buy, or which hotel to stay in. Critical decisions carry significant failure costs, which we all want to avoid. Since our personal experiences are limited, others' opinions may help us to anticipate the consequences of our choices. Especially, when these opinions expressed by the people that are somehow similar to us.

Opinion mining is a computational study of people's opinions, sentiments, emotions, evaluations, etc. [18]. For example, a general task of opinion mining is to derive algorithms and methods for understanding the attitude of a writer towards some entities [102]. The attitude may be an appraisal, a positive or negative sentiment, affective state, such as anger, sadness, excitement, happiness, etc. The entities may be products, services, companies, individuals, issues, events, and so on.

Opinion mining is usually a study of natural language processing, computational linguistics, and text mining, because the largest proportion of opinions is expressed in a written form. These are tweets, blog posts on social media (e.g., Twitter, Facebook) and reviews on e-commerce platforms (e.g., Amazon, Alibaba). Taking advantage of these services, online users constantly contribute opinion content on a variety of topics, including politics, entertainment, technology, society,

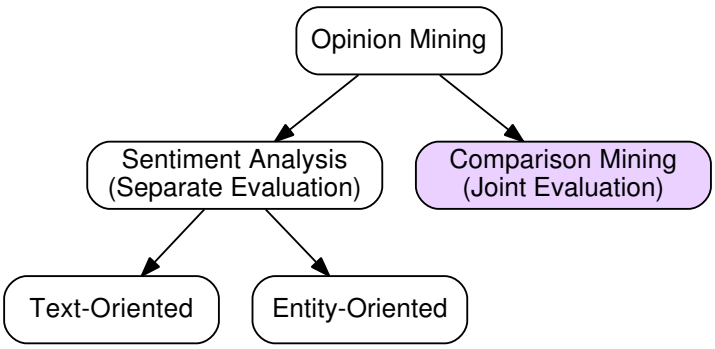


Figure 2.1: Opinion Mining

etc. The scale and diversity of the written opinions make them both attractive and challenging to study. Besides the written opinions, the other forms are receiving a considerable attention of the research community in recent years, the new modes of data include audio recordings, video streams, and images [83, 123, 173]. It is possible that these new forms of data in future will steer away the research focus from text. Nevertheless, this study mainly focuses on written opinions and builds up a foundation for comparison opinions, which can be exploited in any future study.

The following discussion can be reflected by the diagram shown in Figure 2.1. We explain the rationale behind this categorisation and expand discussion of each sub-field of opinion mining. This section mainly focuses on sentiment analysis, discussion of comparison mining is presented in Section 2.3.

2.1.1 Joint and Separate Evaluations

Opinion mining is tightly related to decision theory and social psychology, since it investigates how people evaluate entities. The psychology studies differentiate two entity evaluation modes: *separate evaluation* and *joint evaluation* [65, 66]. Evaluation mode itself is a feature in the decision-making process and according to research may significantly affect decisions. In separate evaluation, entities are evaluated in isolation and independently; in joint evaluation, options are examined simultaneously. The same categories may be adopted in opinion mining. In this case, the study of *separate evaluation* deals with statements that assess entities independently from one another. For example, in sentences *'I like A'* and *'I like B'*,

entities *A* and *B* are apprised independently, and no connection can be derived between them. In fact, the majority of the works in opinion mining focus on separate evaluation, which we attribute to *sentiment analysis*, and which we discuss in the next subsection. *Joint evaluation*, in turn, aims to assess entities simultaneously, these are expressions like *'A is better than B'*, the sentence states relation between *A* and *B* and apprises them relatively. Drawing insights into how certain entities are connected is a task of *comparison mining*.

Sentiment analysis and opinion mining are often considered as synonyms [101, 154]. A minor difference sometimes is drawn to distinguish the task domain for these two studies, i.e., opinion mining focuses on polarity detection, whereas sentiment analysis is about emotion recognition [19]. In this work, we adopt psychology-inspired categories of entity evaluation [66], and assume that sentiment analysis deals only with the statements of separate evaluations. This choice serves a couple of purposes: the evaluation-related categorisation does not disturb the conventional view of sentiment analysis, and it allows us to talk about comparison mining as a sub-study of opinion mining.

Hereafter, we discuss sentiment analysis in Section 2.1.2 and come back to comparison mining in Section 2.3.

2.1.2 Sentiment Analysis

Sentiment analysis, a computational study of people's opinions expressed in separate evaluations (see Section 2.1.1), is essentially the main focus of opinion mining at the moment. It spreads across various domains, such as social media, financial microblogs, news articles, and reviews. Annually, over the past years, International Workshop on Semantic Evaluation (SemEval) organizes a number of sentiment analysis shared tasks [142, 157, 31, 121]. The shared tasks attract considerable attention of researchers as well as industry.

Text-oriented

A basic task of sentiment analysis is polarity classification [154]. Polarity classification occurs when an excerpt of text stating an opinion is classified as one of two opposing sentiments, i.e., negative and positive. To avoid any ambiguity, we assume that excerpts represent only one entity at a time. Thus, our main concern of analysis is text itself. Classification of reviews, such as 'one star' vs 'five stars', classification of comment 'likes' and 'dislikes' are common incarnations of polarity classification. One may assign different degrees of sentiment to an excerpt [128], for example, strongly positive vs. positive, including as well the neutral assessment of the text, i.e., containing no sentiment. The latter is related to the task of subjectivity classification, which aims to determine if an excerpt contains a subjective evaluation or an objective statement [27].

Over the decades, there have been proposed variety of approaches to deal with text-oriented sentiment analysis. The basis ground lies around understanding opinion words (e.g., good, bad, like, hate) that usually trigger opinions and their syntactic interactions. Thus, researchers propose to compile opinion lexicons and use them to identify polarities of excerpts [6, 67]. However, such lexicons must be domain-specific to capture domain-specific connotations, for example, compare *'low price'* vs. *'low blood pressure'*, *low* is positive in the context of *price*, but may be negative in medical evaluation. Merely a presence of opinion word may not imply any sentiment. To overcome these problems and capture ambiguity of the language, researchers often turn to machine learning methods or handcrafted patterns along with sentiment lexicons [167, 165, 147, 144, 37]. The difference between methods is defined by the difference in patterns, if they work on word sequences, on dependency tree representations, etc. As the result, having training data and domain expertise is an essential part of modern sentiment analysis research.

Entity-oriented

Entity-oriented sentiment analysis prioritises entities over their textual representations. Thereby, text becomes a proxy for opinions expressed about entities, that are scattered around sentences, documents, or even various online resources. A basic task of entity-oriented sentiment analysis is to identify a particular entity or a part of an entity, which opinions are expressed about. The particular part is usually called *aspect* or *feature* and the study is *aspect-based* or *feature-based* sentiment analysis [101]. The aspect may be compactness of camera, its build quality, price, etc. The aspects are entity related and defined by a particular domain. For example, sentence *'I like that A is super portable but at the same time it is hard to hold'* comments on two aspects of *A*, namely its portability and usability, where the portability is endorsed (positive opinion), but the usability is considered as a flaw (negative opinion).

The aspect set may be defined in advance or extracted from text [68, 145]. Basic approaches to aspect extraction employ frequency analysis in order to locate the salient nouns [67, 143]. Supervised labeling methods [73, 69, 15, 146] are widely adapted for aspect extraction. Due to diversity in the ways one can introduce aspects in text, topic modeling approaches are proposed to deal with the aspect extraction problem [138, 14, 126]. There are also works focusing on modeling the correlation between topics and user ratings [179].

Once the aspects are identified, we may produce a summary for every entity in a text collection [68], which presents detailed table about aspects endorsed by the users, proportion of negative and positive opinions, etc.

2.2 Comparison Studies

Comparison mining does not exist only as a part of opinion mining, but also as a part of *comparison studies* (see Figure 2.2), which is a collective term for the research involving joint evaluation of entities. For example, one may explore the problem of

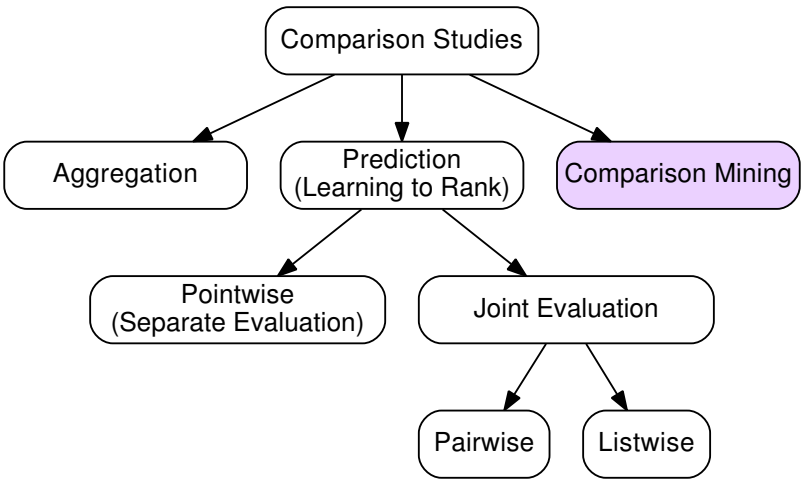


Figure 2.2: Comparison Studies

finding similarities and differences between two documents. Another kind of joint evaluation is a ranking problem, which requires a set of entities to be arranged with respect to each other.

In this section, we discuss in details two sub-fields of comparison studies: *aggregation* (Section 2.2.1) and *prediction* (Section 2.2.2) models. Comparison mining, the third part of comparison studies, is discussed later in Section 2.3.

2.2.1 Aggregation

Aggregation models aim to combine multiple usually short orderings into a unified ranking. For example, if we have three entities A, B, C, and we know that A is better than B in some sense and B is better than C in the same sense, then we may arrive at the complete ranking where A is at the first position, B is at the second, and C is at the third. Note that this conclusion requires a transitivity assumption, because comparison between A and C was never stated, but implied.

We may want to aggregate rankings, for example, to assess relative skill levels of players in a sports tournament or to place relative customer preferences on the scale, which indicates how likely a customer is to buy a certain product. A more technical example is meta-search, which aims to combine several relevance rankings of different search engines.

Comparisons or pairwise orderings are often aggregated to assess relative player

skills in competitions like chess [40] or computer games [63]. Considerable amount of studies were driven by psychometric research that were focused on studying preference scaling [13, 106, 166]. Despite being different in purpose, competition and psychometric models are grounded on the same probability modeling principle: to assess the probability that one entity wins over the other, where the notion winning is defined for a particular task. Comparison aggregations requirements may vary with respect to the domain and explicitly model situations like draw or team tournaments [63]. The probability models are usually defined over the same set of latent 'ability' scores which can be compared across all the entities, and, thus, support prediction via direct score comparison, although limited to the entities that already 'participated' in the competition.

It is natural to extend comparison aggregation models to accommodate aggregation of ranked lists [141, 152, 35], when the latter makes sense. For example, Plackett considers the problem of estimating the probability of a horse race outcome [141]. Such extensions require specialized estimation procedures [71, 57].

Aggregation methods find their applications in meta-search [39, 176], where different entity rankings coming from various sources need to be merged into a single ranking. A number of methods were explored for this problem, for example, Mallows' model [149], Borda's count [39, 42], graphical models [177], supervised preference aggregation [176].

A set of methods explore the related problem of ranking clustering. Given a set of entity rankings, we seek to determine the ranking groups with high inter-agreement, e.g., to have high probability to rank entities in the same way. For instance, [21] describes a nonparametric extension to model an infinite number of entities, and clustered rankings via Dirichlet process mixtures. Others apply mixture models for profiling Irish electorates, including [54, 53]. In turn, [55] explores a mixture of Benter's models, which are generalized forms of the Plackett-Luce. [194] addresses the question of identifiability of Plackett-Luce mixture and proposes an efficient method to learn mixture of two Plackett-Luce models.

The common basis for this category of methods is entity representation. Each entity is unique and is encoded via one-hot vector when required. Therefore, it limits aggregation only to the entities that models can subsume during the training phase. The main purpose is information aggregation. Generalized prediction models require feature-based entity representation or are able to compute these representations on-the-fly.

2.2.2 Prediction

Learning to rank is an application of machine learning techniques for entity ranking [46]. Usually, we want to induce a ranking function, given some ranking observations. For example, typical learning to rank application is to learn a ranking function, that assesses how relevant each document in a collection to a given query. In this case, documents and queries may be represented as bag-of-word vectors, and the learned ranking function may assess word similarity between query and document. Besides, learning to rank has been successfully applied in computational linguistics [159], computational biology [100], and recommender systems [82].

By the input representation and form of a loss function that is minimized during the training, learning to rank approaches are usually divided into three groups: pointwise, pairwise, and listwise [105]. I discuss each in the following subsections.

Pointwise

Pointwise approach assumes that the entities to be ranked have numerical or ordinal scores. Thereby, learning methods are used to approximate these scores, essentially casting learning to rank problem as regression learning problem. Thus, a number of standard machine learning methods can be applied for this purpose [62, 195]. At the learning and prediction phase entities are examined independently and, therefore, pointwise approach falls under **separate evaluation** category (see Section 2.1.1 for details).

Pairwise

In contrast to pointwise approach, entities can be evaluated in the **joint evaluation** mode. One example of joint evaluation is to simultaneously consider pairs of entities for the optimization problem. For the pairwise approach, learning to rank is usually approximated via classification problem. The ranking outcome for pair (A, B) is encoded by a class label: whether A goes before B or otherwise. A binary classifier then can tell which entity should be given the priority. For the learning to rank objective, we want to minimize the average number of ranking inversions [80, 17].

Listwise

Another approach of **joint evaluation** is listwise approach. Listwise approach directly optimizes ranking quality measures, for instance, mean average precision, Kendall's tau, or discounted cumulative gain. The idea of optimizing target quality measure is challenging, because most of the evaluation measures are not continuous. Viable approximations are required for standard approximation techniques [174]. Aggregation-inspired loss functions (see Section 2.2.1) are successfully used for this purpose [183, 170]. Empirically, the listwise approach has been shown to outperform the pointwise and pairwise approaches [20]. Examples of listwise approaches are Coordinate Ascent [119] and ListNet [20].

2.3 Comparison Mining

Comparison mining aims to discover, extract, analyse and summarise comparison-related information. The focus of this study is on text, so the following discussion is about comparison mining from text. Comparison mining emerges on the intersection of opinion mining and comparison studies. It handles opinion-related information, yet it requires to solve satellite problems, that may be applied outside of opinion analysis. For example, analysing a set of product reviews, we may want to highlight similarities and differences of consumers' opinions with respect to their

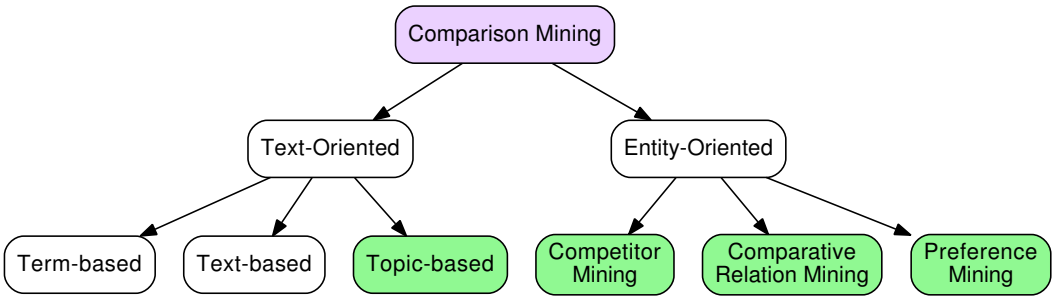


Figure 2.3: Comparison Mining. The green marks the main contributions.

ratings in terms of their words, sentences, topics, aspects, etc. This particular analysis example is neither aggregative, it does not merge entity rankings, nor predictive, it does not focus on extrapolating rankings to unseen entities; however, it might use them as a part of the analysis process.

In contrast to sentiment analysis (Section 2.1.2), comparison mining is an approach of *joint evaluation* (see Section 2.1.1 for details). All the methods of comparison mining must jointly analyse a set of entities to draw a conclusion. Independent analysis is not applicable, since a comparison involves at least two entities, and mining information about this comparison is the goal.

In a way similar to sentiment analysis, comparison mining may be divided into two categories: *text-oriented* and *entity-oriented* mining (Section 2.1.2). Text-oriented mining seeks to discover and summarise comparison information for the entities represented in their textual form, such as product descriptions, political speeches, etc. These methods are discussed in Section 2.3.1. Entity-oriented mining seeks to extract information about the entities from text. For example, processing online reviews, we may want to discover which entities are good substitutes for each other. Entity-oriented mining is discussed in Section 2.3.2. The categories of comparison mining are outlined in Figure 2.3.

2.3.1 Text-oriented

Text-oriented comparison mining aims to find and summarise differences between text collections. The collections are usually selected to have a certain property, for

example, to have different star ratings (e.g., one-star vs. five-star reviews), speeches from different political parities, reviews of competitive products, etc. The goal often is to provide a short summary of the opposing view and to explore hypotheses about their difference.

Term-based

Term-based comparison mining seeks to extract terms that are significantly represented in one collection but not the other. For example, we may expect that positive sentiment words occur more often in five-stars reviews than in one-star reviews. These methods allow us to quantify this difference. The idea is to assign the association score for pair term-document, which is often defined via relative term frequencies. To a certain extent, term-based comparison mining resembles filtering-based feature selection techniques [22], however, their purpose is different. Whereas feature selection methods are generally used to reduce computation time and improve prediction performance, comparison mining aims to support data understanding process. As a research tool, it often plays a supportive role in linguistic investigations and has been successfully applied across domains. Term-based methods have been used to explore narratives of positive and negative opinions in restaurant reviews [81], as well as to analyse U.S. Senate speeches [122].

Text-based

Text-based mining seeks to accomplish the same task as term-based mining. Given two collections of texts, a method is to select a small number of excerpts that characterize their similarities and differences. For example, one problem of text-based comparison mining is *contrastive summarisation*, which focuses on highlighting differences between two entities [95], Lerman and McDonald experimented with customer reviews. *Comparative summarisation* constructs short summaries by aligning excerpts of reviewers' opinions for a pair of products [139]. The excerpts are aligned on the basis of aspects, which is a common objective of research in

entity-oriented sentiment analysis (see Section 2.1.2). Information to be selected varies with the input domain and intended application [89, 161]. The methods are often formulated as the optimization problems, constrained to look for similarities and differences of two entities.

This area is tightly related to text summarisation, which is a process of shortening a text to create a summary of the original document [50] or a collection of related documents [116]. The key difference between such kind of automatic text summarisation and text-based comparison mining is the requirement to analyse a set of different entities simultaneously.

Topic-based

Topic-based comparison mining takes the middle ground between term-based and text-based comparison mining. The output of such methods is defined in terms of topics, sets of co-occurring words [10], which often are used to discover hidden semantic structures in text. The topics may be aligned with the prediction tasks like regression or classification, such methods emerge as a family of supervised topic models [150, 197]. A comparison-based supervision is a new line of study introduced in this work [172] (see Chapter 6). As a part of decision-making process, it is important to explain the ranking decisions that reviewers postulate. For example, having entity rankings induced from text (see Section 2.3.2), we wish to uncover the hidden semantics behind them. The closest to our study is the pointwise sLDA [114]. There are yet others that pursue pointwise supervision, but explore other angles that are not directly comparable. [92] introduces class-specific linear transformation to modify the topic distribution of a document, which would be applicable only to categorical labels but not continuous numerical responses. [197] explores max-margin learning. [151, 150] associate a document with multiple labels (e.g., tags).

Regarding modeling of document connections, there are works that leverage document-pair supervision, such as two documents being similar or being linked in

a network [24, 117, 41, 187, 25].

Other topic models focus on idiosyncratic notions of "comparison" different from ours. For instance, [191] compares two or more corpora, by finding shared topics and distinct topics between the corpora. Similarly, [88] employs different unsupervised procedure to satisfy the same goal. Also focusing on the corpora-level comparison, [44] seeks to identify contrasting opinions on specific topics.

2.3.2 Entity-oriented

Entity-oriented comparison mining aims to acquire relations between entities. One may be interested to know if two products are comparable [74], and if they are how often consumers compare them, and what their preferences are [77]. The entities are no longer represented by their textual form, but rather their interconnections that must be extracted from text. The contributions of this study are scattered around this category, which is indicated by the green in Figure 2.3.

Competitor Mining

Several studies explore mining of *comparable entities* or *competitors* [96, 72]. These are entities, that are likely to share the same utility for a buyer, for example, laptops with similar specifications but different brandings, different digital cameras in the same product line, etc. When a customer wants to buy a product, she is likely to choose it among the comparable entities. The study of discovering comparable entities is called *competitor mining*.

One way to find competitors is to look for the similar entities, where similarity can be defined over a set of entity attributes [93].

Other common approaches suggest to mine competitors from texts, such as reviews. The idea is to use textual patterns to match the explicit mentions of comparability, like *'A is a good alternative to B'*. In the example, *A* and *B* are indicated to be comparable by the writer. The methods vary in the way the patterns are represented and learned. A pattern is usually a 'regular expression', extended to match part-

of-speech tags [76, 96]. Machine learning algorithms are successfully employed [85]. Often similar patterns and learning algorithms are used to identify 'components' within comparative sentences, i.e., entities, aspects, comparative predicates [77, 64, 87, 85, 45], however this is not the focus of competitor mining.

In this study, we propose a machine learning method based on dependency tree matching for extracting comparisons via kernel-based optimization [169] (see Chapter 3), which empirically outperforms its competitors. The task is to derive if a pair of entity mentions is in comparative relation. If two entities are compared within text, they can be considered comparable. In this case, we assume that the entity mentions are marked within sentences [77, 84] and different mentions linked to each other when they represent the same entity [36]. The previous state of the art for extracting comparisons is the baseline CSR approach [76]. For scientific text, [137] explored handcrafted syntactic rules. Comparison identification is also studied in other languages via similar approaches [70, 186, 91, 186, 192].

Other than comparison identification, dependency grammar has also found applications in natural language-related tasks, such as sentiment classification [127], question answering [148, 99], as well as relation extraction [33, 16]. Often kernel methods based on trees are applied to relation extraction. [28] applied convolution kernels [61, 180] to natural language objects, which evolved into tree kernels, e.g., sub-tree [162], subset tree [29], descending-path kernel [98], partial tree [124]. Our proposed kernel, Skip-node kernel, joins this list.

An alternative approach to mine comparable entities is to try to predict them [188]. For example, Myungha at al. [74] employ graph-based clustering to complete the graph of comparable entities. The nodes in such graph represent entities and the edges are comparability relations, if we have initial information about entity comparability, which can be obtained via text mining as mentioned above, then the problem can be cast as the edge prediction problem.

Comparative Relation Mining

Comparative relation mining expands the notion of being comparable and seeks to interpret comparative expressions if possible. Comparative expressions imply similarity or differences between entities, e.g., sentence *'A beats B and C'* induces partial ordering relation among entities: A is better than B, A is better than C, but B and C are not compared. Comparative relation mining can be divided into a couple of tasks: comparison interpretation and aggregation. Interpretation seeks to infer the meaning of comparative excerpts, stating which entity should be ranked first. It is often done upon extraction of comparable entities (see Section 2.3.2), where patterns are *labelled* and immediately attach the preference class [91]. Such patterns can be combined with a binary classifier to perform interpretation as classification [77, 51, 184]. External informations, such as "pros" and "cons" sections in reviews is also utilized in for comparison interpretation [51].

The aggregation part studies how to combine individual comparison relations into ranked list. This can be accomplished by the *aggregation models*, see Section 2.2.1 for details. A number of them were adopted in previous studies [91, 193, 97]. Another way to look at the problem of aggregation is to use PageRank [133] on a graph, where the nodes are entities and edges encode information about individual comparisons. However, we have shown that this is inferior to another Bradley-Terry-Luce model [168].

In this study, we show that the combination of the interpretation and aggregation parts of comparative relation mining in a unified graphical model outperforms the other approaches [171], which deal with the problem in a pipeline manner, solving the two problems independently. The unified approach is discussed in Chapter 4.

Preference Mining

Preference mining arises when a number of users have differing opinions about which entities they prefer, and, as the result, naïve aggregation models cannot be applied. In this case, an aggregation model has to recognize that a set of comparisons

or rankings is not consistent and to be able to extract and express this inconsistency. We propose to treat preferential differences at the level of groups, so that inside every group the preferences are homogeneous [170]. Each group is then attached to a ranking function that can be used to induce preference for a set of unseen entities on behalf of the group member. Preference mining extends learning to rank research (see Section 2.2.2) by assuming the existence of multiple ranking functions used within the same set of entities [105]. The problem of ranking aggregation is different to ours, as it seeks to combine multiple rankings instead [39]. Another related work [56] relies on clustering instances in the feature space to obtain rankings. This is a distinct problem, because it clusters instances by similarity in features, rather than similarity in ranking functions. In contrast to collaborative filtering and other recommender systems, preference mining does not seek to predict preferences of an individual user and does not model user-specific parameters [1, 90]. In the same way, personalized ranking approaches [156, 181, 104] are different from preference mining. The extensive discussion on preference mining is presented in Chapter 5.

CHAPTER 3

IDENTIFYING COMPARISONS IN TEXT

When weighing various alternatives, users increasingly turn to the social media, by scouring online reviews, discussion forums, etc. Our goal is to extract from such corpora those text snippets where users make direct comparisons of entities. While sentiment analysis [135] may be helpful in evaluating individual entities, comparison by the same author within a sentence provides an unambiguous and more equitable basis for the relative positions of two entities on some aspects. For example, sentence s_1 in Table 3.1, taken from an Amazon review about a digital camera, makes two distinct comparisons: #1) between "A630" and "A-series cameras" and #2) between "A630" and "its competition", with a clear sense of which entity mention is the *greater* on some aspect ("larger"). Moreover, comparisons may be objective (e.g., larger) or subjective (e.g., better), while sentiments are primarily subjective.

ID	Sentence	Remarks
s_1	The A630 is slightly larger than previous generation A-series cameras, and also larger than much of its competition.	Contains two comparisons: (A630, A-series cameras) and (A630, its competition).
s_2	I got 30D for my wife because she wanted a better camera.	Includes comparative predicate "better", but contains no comparison.
s_3	I had D3100 and it was nice but the D5100 is truly amazing.	No comparative predicate, but has a comparison: (D3100, D5100).
s_4	D7000 and D7100 do better at high ISO than D300s.	Contains two comparisons: (D7000, D300s) and (D7100, D300s).

Table 3.1: Example Sentences with ≥ 2 Entity Mentions from Amazon.com Digital Cameras Reviews.

Problem Given a sentence and a specific pair of entity mentions, we seek to determine if a comparison exists between those two mentions. In previous work, the problem was formulated as identifying *comparative sentences*, i.e., those containing at least one comparison [76]. This is not ideal because a sentence may contain more than two entity mentions, and may be comparing only some of them. For instance, s_1 is comparative with respect to pair (A630, A-series cameras) and pair (A630, its competition), but not pair (A-series cameras, its competition).

We therefore postulate that the more appropriate formulation is *comparisons within sentences*. If a sentence compares two entities (A, B) with respect to aspect Z, it should be possible to reformulate it into another sentence such as: "A is better than B with respect to Z" [86]. Based on this definition, there is no comparison between (A-series cameras, its competition) in s_1. Here, we adopt this apt definition with a slight restriction to make it more practical, and seek to identify such comparisons automatically. We consider only sentences with at least two entity mentions involved in gradable comparisons, i.e., a clear sense of scaling in the comparison (e.g., A is better than B.). Such comparisons are more useful in investigating the pros and cons of entities, as opposed to equative comparisons expressing parity between two mentions (e.g., A is as good as B.), or superlative comparisons expressing the primacy of an entity with respect to unknown reference entities (e.g., A is the best.).

Approach For English, there usually is a comparative predicate that anchors a comparison, such as "better" or "worse". However, many sentences with such predicate words are not comparisons. Sentence s_2 in Table 3.1 has the word "better", but does not contain any comparison between the entity mentions. Yet, other words (e.g., "amazing"), though not a comparative predicate, could signify a comparison, e.g., in s_3 in Table 3.1.

[76] considers the "context" around a predicate. A sentence is transformed into a sequence involving the predicate and the part of speech (POS) within a text window around the predicate (usually three words before and after). For instance, s_2

in Table 3.1 would be transformed into the sequence ⟨*PRP VBD DT better NN*⟩, where *PRP* stands for personal pronoun, *VBD* for verb in the past tense, *DT* for determiner, and *NN* for singular noun[1]. Such sequences are labeled comparative or non-comparative, upon which [76] applies sequential pattern mining [2, 5, 140] to learn *class sequential rule* (CSR). These CSRs are then used as features in classifying comparative sentences.

While [76] makes some progress by considering context, its performance may be affected by several factors. First, CSRs are not sensitive to entity mentions. It may classify s_1 as comparative generally, missing the nuance that s_1 is not comparing the pair (A-series cameras, its competition). Second, as CSRs require a list of comparative predicates, the quality and the completeness of the list are crucial. For instance, "amazing" is not in their list, and thus the comparison in s_3 may not be identifiable. Third, due to the windowing effect, CSRs has a limited ability to model long-range dependencies. For s_4, a window of three words around the predicate "better" excludes the word "than" that would have been very informative. Yet, enlarging the window might then bring in irrelevant associations.

What is important then is not so much whether a sentence is comparative as whether two entity mentions are related by a comparative relation. One insight we draw is how comparison identification is effectively a form of *relation extraction*. While there are diverse relation extraction formulations [33, 16, 130], our distinct relation type is comparison of two entity mentions.

Armed with this insight, we propose a kernel-based approach based on a dependency tree representation [132], with significant innovations motivated by the comparative identification task. This proposed approach has several advantages over CSR. Most importantly, it models dependencies between any pair of words (including entity mentions), whereas CSR only relates a comparative predicate to nearby POS tags. For other advantages, unlike CSR, this approach is contingent on neither a pre-specified list of comparative predicates, nor a specific window length.

[1]For the comprehensive list of part-of-speech tags refer to [158].

In Section 3.1, we give a formal overview of the comparison identification problem. In Section 3.2, we discuss the kernel approaches that have been used for relation extraction, and develop Skip-node kernel, which is suitable for comparison identification. In Section 3.3, we derive a formal computational framework for the kernel. Section 3.4 discusses effectiveness and efficiency of the method through experiments on real-life datasets. Section 3.5 concludes this chapter with the discussion.

3.1 Problem

The input is a corpus of sentences $\mathcal{S}$ concerning entities within a certain domain (e.g., digital cameras). Every sentence $s \in \mathcal{S}$ contains at least two entity mentions. The set of entity mentions in s is denoted M_s. For instance, sentence s_4 in Table 3.1 contains three entity mentions: D7000, D7100, and D300s. The same entity may be mentioned more than once in a sentence, in which case each mention is a distinct instance.

As output, we seek to determine, for each pair of entity mentions $(m_i, m_j) \in M_s$ in a sentence $s \in \mathcal{S}$, a binary class label of whether s contains a comparison between m_i and m_j. For the pair (D7000, D7100) in s_4, the correct class is 0 (no comparison). For the other two pairs (D7000, D300s) and (D7100, D300s), the correct class is 1 (comparisons). We do not seek to identify the aspect of comparison, which is a different problem of independent research interest (see Section 3.5).

Dependency Tree In order to represent both the lexical units (words) as well their structural dependencies seamlessly, we represent each sentence s as a dependency tree T. For example, Figure 3.1(a) shows the dependency tree of s_4 in Table 3.1. The tree is rooted at the main verb ("do"), and each dependency relation associates a head word and a dependent word. To describe a tree or any of its substructures, we use the bracket notation. Figure 3.1(a) in this notation is

```
[do [D7000 [and] [D7100]] [better [at [ISO [high]]]
[than [D300s]]]].
```

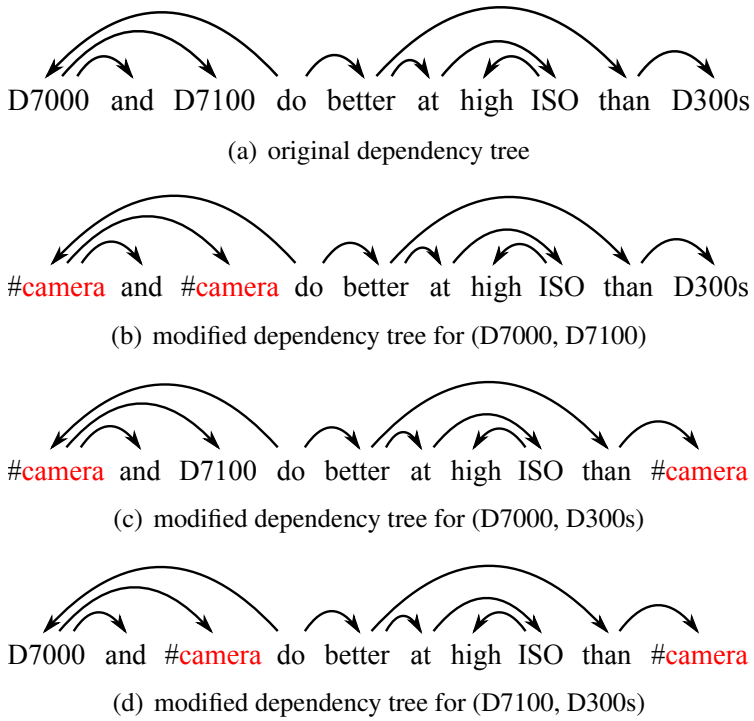


Figure 3.1: Modified Dependency Trees.

Here, we make two observations. First, there is one tree even for a sentence with multiple pairs of entity mentions. Second, the information signalling a comparison is borne by the structures around the mentions (e.g., `[better [than]]`), rather than the actual mentions (e.g., "D7000"). These lead us to introduce a *modified* dependency tree that is distinct for every pair of mentions, achieved by replacing each entity mention of interest by a placeholder token. Here, we use the token "#camera" for illustration. Figure 3.1(b) shows the modified tree for the pair (D7000, D7100). This enables learning in an entity-agnostic way, because the token ensures that sentences about different cameras are interpreted similarly.

Convolution Kernel Observe how the trees of the pair (D7000, D300s) in Figure 3.1(c) and the pair (D7100, D300s) in Figure 3.1(d), which are both comparisons, share certain substructures, such as `[do [better [than [#camera]]]]`. In contrast, the tree in Figure 3.1(b) for the pair (D7000, D7100), which is not a comparison, does not contain this substructure. What we need is a way to systematically examine tree substructures to determine the similarity between two trees.

Kernel methods offer a way to measure the similarity by exploring an implicit

feature space without enumerating all substructures explicitly. Suppose that $\mathbf{T}$ denotes the space of all possible instances. A kernel function K is a symmetric and positive semidefinite function that maps the instance space $\mathbf{T} \times \mathbf{T}$ to a real value in the range of $[0, \infty)$ [61]. A tree kernel function can be reformulated into a *convolution kernel* [28], shown in Equation 3.1.

$$K(T_1, T_2) = \sum_{n_i \in T_1} \sum_{n_j \in T_2} D(n_i, n_j) \tag{3.1}$$

Here, n_i and n_j denote each node in their respective tree instances T_1 and T_2. $D(n_i, n_j)$ is the number of common substructure instances between the two subtrees rooted in n_i and n_j respectively. The exact form of $D(n_i, n_j)$ depends on the specific definition of the tree kernel space. In Section 3.2, we systematically explore the applicability of various tree kernel spaces, leading to the introduction of the new *Skip-node Kernel*.

The appropriate kernel function can be embedded seamlessly in kernel methods for classification. In this work, we use the Support Vector Machines (SVM) [164].

3.2 Tree Kernel Spaces

Tree kernels count substructures of a tree in some high-dimensional feature space. Different tree kernel spaces vary in the amount and the type of information they can capture, and thus may suit different purposes. To find a suitable tree kernel for the comparison identification task, we first systematically explore a progression of known tree kernel spaces, including Sub-tree, Subset Tree, and Partial Tree. Through the use of appropriate examples, we show how these existing tree kernel spaces may not be appropriate for certain instances. This section culminates in the introduction of a new feature space that we call Skip-node.

Sub-tree (ST) Space In this space, the basic substructure is a subgraph formed by a node along with all its descendants. Applying this kernel to two dependency trees of similar sentences may not be appropriate due to, for example, modifier

words that change the dependency structure. To illustrate this, let us examine the two dependency parses in Figure 3.2. Both support comparisons, and ideally we can detect some level of similarity. However, if we consider only sub-trees, the two dependency trees share in common only two fragments: `[#camera]` and `[is]`. Neither of these fragments is indicative of a comparison.

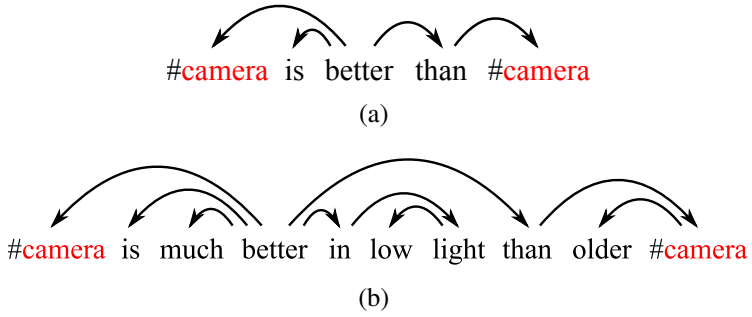


Figure 3.2: Dependency Parses. The working example for the Sub-tree, Subset Tree, Partial Tree kernels.

Subset Tree (SST) Space We next consider the SST kernel, which computes similarity in a more general space of substructures than ST. Any subgraph of a tree that preserves production rules is counted. This definition suggests SST is intended more for a constituency parse [124]. In this feature space, the parses in Figure 3.2 now have in common the following fragments: `[#camera]`, `[is]`, `[than [#camera]]`. This representation is better than ST's, e.g., the fragment `[than [#camera]]` is informative. However, as a whole, the set of features are still insufficient to identify a comparison.

Partial Tree (PT) Space In turn, the PT space allows breaking of production rules, making it a better choice than SST for dependency parses. PT kernel would find that the parse in Figure 3.2(a) with all its subgraphs can be matched as a whole within the parse in Figure 3.2(b), identifying a close match.

However, PT kernel is prone to two drawbacks. By generating an exponential feature space, it may overfit and degrade generalization [34]. More importantly, PT considers tree fragments independently from their contexts, resulting in features involving non-related parts of a sentence. This is particularly apparent when we consider multiple entities within a sentence.

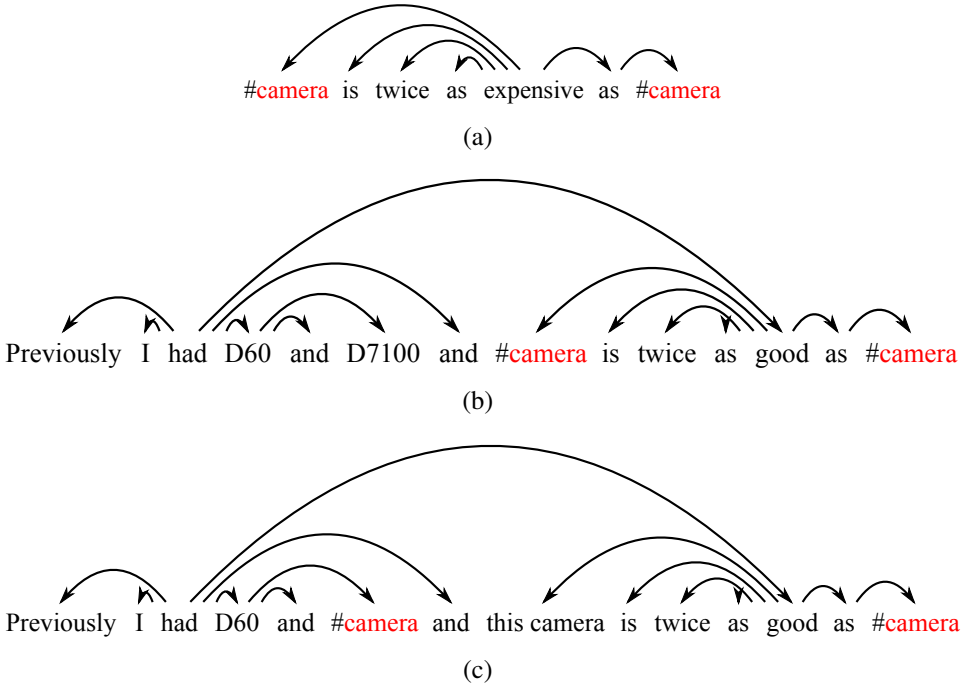


Figure 3.3: Dependency Parses. The working example for the Partial Tree, Skip-node kernels.

Suppose that Figure 3.3(a) is in our training set, and we have the sentence below in the testing set:

> Previously, I had D60 and D7100, and this camera is twice as good as D60.

Figure 3.3(b) shows the parse for (this camera, D60), and Figure 3.3(c) for (D7100, D60). The former is a comparison, and should match Figure 3.3(a). The latter is not and should not match. PT kernel cannot resolve this ambiguity, computing the same similarity value to Figure 3.3(a) for both. The common features are: `[#camera]`, `[is]`, `[twice]`, `[as]`, and `[as [#camera]]`.

Skip-node (SN) Space Figures 3.3(a) and 3.3(b) share a similar substructure "twice as ... as", but because they use different words to express the comparisons ("expensive" vs. "good"), previous kernels treat their features disjointly, missing out on their similarity. To reduce this over-reliance on exact word similarity, we seek a feature space that would allow some degree of relaxation in determining the *structural* similarity between trees.

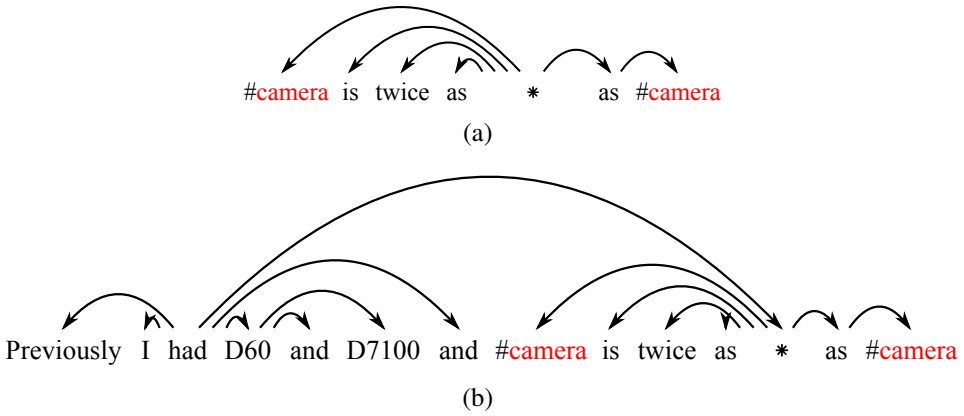


Figure 3.4: Dependency Parses with Skipped Nodes.

We therefore propose the Skip-node (SN) space, which represents a generalized space of tree fragments, where some nodes can be "skipped" or re-labeled to a special symbol '$\star$' that would match nodes of any label. A restriction on this space is that each skip symbol must connect two non-skip (regular) nodes. The implication is that skips code for some notion of connecting distance between non-skip nodes. Moreover, the space would not include features such as `[* [* [#camera]]]` that serve only to indicate the presence of ancestors, and not any relationship of non-skip nodes.

Figure 3.4 resolves the ambiguity in Figure 3.3 by skipping the words "expensive" and "good", introducing a new set of features: `[* [#camera] [is] [twice] [as] [as [#camera]]]`. Note how in this case the skip symbol effectively serves as a "context" that pulls together the previously disjoint features identified by the PT kernel. These new context-sensitive features would allow a match between the earlier Figures 3.3(a) and 3.3(b), but not Figure 3.3(c).

Thus, SN space effectively generalizes over the PT space, and enriches it with context-sensitive features. To avoid overfitting, in addition to decay parameter λ used in PT kernel, we associate SN kernel with two other parameters. The SN space consists of rooted ordered trees where some nodes are labeled with a special skip symbol '$\star$', such that the number of regular nodes (not marked with '$\star$') is at most S, and each skip node is within a distance of L from a non-skip node. This engenders a graceful gradation of similarity as the number of skip nodes in a

substructure grows, yet imposes a limit to the extent of relaxation.

3.3 Skip-node Kernel Computation

We now discuss the computation of Skip-node Kernel, first exactly, and thereafter approximately.

3.3.1 Exact Computation

We define the alignment of common fragments between two trees in the Skip-node space. When $S = 1$, only singleton nodes with the same labels contribute to the kernel, and alignment is straightforward. When aligning fragments with two regular nodes ($S > 1$), we consider their connection structure and the order of the child nodes to prevent over-counting substructures with the same labels (e.g., `[*[as][as]]` in Figure 3.4). To preserve the natural order of words in a sentence, we enumerate the tree nodes according to preorder, left-to-right depth-first search (DFS) traversal.

In turn, the connection structure is defined by the skip-node path connecting two regular nodes. This can be expressed as a sequence of upward (towards the root) and downward (towards the leaves) steps we need to perform to get from the leftmost to the rightmost regular node. Due to the natural ordering of regular nodes, upward steps are followed by downward steps. The sequence can be expressed as a pair of numbers: $\langle \rho(n_l, u), \rho(n_r, u) \rangle$, where n_l is the leftmost regular node of a fragment, n_r is the rightmost one, $u = \sigma(n_l, n_r)$ is the lowest common ancestor of nodes n_l, n_r, and ρ returns the number of edges in the shortest path connecting two nodes.

Suppose rooted tree $T = (N, E)$ has corresponding preorder DFS enumeration $N = (n_1, n_2, ..., n_{|N|})$. For $i < j$, we define a function $\pi(n_i, n_j)$, which canonically

represents the way two nodes are connected in a tree, as follows:

$$\pi(n_i, n_j) = \langle \rho(n_i, \sigma(n_i, n_j)), \rho(n_j, \sigma(n_i, n_j)) \rangle.$$

DEFINITION 1 (STRUCTURAL ISOMORPHISM): Given trees $T_1 = (N_1, E_1)$, $T_2 = (N_2, E_2)$, we say that pairs of nodes $(v_i, u_{i'}), (v_j, u_{j'}) \in N_1 \times N_2$ are *structurally isomorphic* and write $(v_i, u_{i'}) \leftrightsquigarrow (v_j, u_{j'})$ when $\pi(v_i, v_j) = \pi(u_{i'}, u_{j'})$ on the valid domain.

It can be shown that structural isomorphism is a transitive relation. This property allows us to grow aligned fragments by adding one node at a time:

$$(v_i, u_{i'}) \leftrightsquigarrow (v_j, u_{j'}) \wedge (v_j, v_{j'}) \leftrightsquigarrow (v_k, u_{k'}) \Rightarrow (v_i, u_{i'}) \leftrightsquigarrow (v_k, u_{k'}).$$

To compute the kernel, we use a graph-based approach to enumerate all the common substructures in the Skip-node space. Given two trees T_1 and T_2, we begin by aligning their nodes. The sets of nodes in T_1 and T_2 are N_1 and N_2 respectively. Let N_G be a set of pairs $(n_i, n_j) \in N_1 \times N_2$, where n_i and n_j have the same label. On top of N_G, we build a graph $G = (N_G, E_G)$. We draw an edge between two vertices $(v_i, v_k), (u_j, u_l) \in N_G$, if $(v_i, u_j) \leftrightsquigarrow (v_k, u_l)$ and $\rho(v_i, v_k) \leq L$.

Any connected subgraph of G represents a feature in the Skip-node space common to both T_1 and T_2. The kernel then needs to count the number of connected subgraphs of sizes not more than S. To see that this procedure is correct, we simply need to trace back the construction of graph G, and build a bijection from a subgraph of G to the corresponding fragments of T_1 and T_2.

Enumerating all the connected subgraphs of a given graph requires exponential time. The algorithm described above requires $\mathcal{O}(|N_1||N_2| + \sum_{i=1}^{S} \binom{|N_G|}{i})$ time, assuming that the distance between two nodes in a tree can be computed in $\mathcal{O}(1)$ with appropriate linear preprocessing. See [8] for insight. The exact computation is still tractable on the condition that S and L are not very large. This condition would probably hold in most realistic scenarios. Yet, to improve the practicality of

the kernel, we propose a couple of approximations as follows.

3.3.2 Approximate Computation

One reason for the complexity of the Skip-node kernel is that although the graph G is formed by aligning two trees, by allowing connections through skips, G itself may not necessarily be in the form of a tree. In deriving an approximation, our strategy is to form G through alignment of linear substructures of the original two trees. A Skip-node space over linear structures can be computed in polynomial time using dynamic programming.

Linear Skip-node One approximation is to consider linear substructures in the form of root-paths. A root-path is a path from the root of a tree to a leaf. Given two trees T_1 and T_2, with DFS enumerated nodes $N_1 = (v_1, v_2, ..., v_{m_1})$ and $N_2 = (u_1, u_2, ..., u_{m_2})$ respectively. Here, v_1 and u_1 are roots, and v_{m_1} and u_{m_2} are the leaves. Starting with common fragments at the leaves, we grow them into larger common fragments towards the root. We call this approximation *Linear Skip-node*. Figure 3.5(a) shows examples of features considered by Linear Skip-node for the illustrated tree T in skip-node space $(S = 3, L = 2)$.

The kernel function can be decomposed into:

$$K(T_1, T_2) = \sum_{v_i \in N_1} \sum_{u_j \in N_2} \sum_{s=1}^{S} \lambda^s D(v_i, u_j, s),$$

where $D(v_i, u_j, s)$ is the number of common substructures of size s with the leftmost regular nodes v_i and u_j. λ is a decay factor for substructure size.

The recursive definition of the kernel is:

$$D(v_i, u_j, s) = \sum_{i<k\leq m_1} \sum_{j<l\leq m_2} I(v_i, v_k, u_j, u_l) D(v_k, u_l, s-1),$$

$$D(v_i, u_j, 1) = \begin{cases} 1 & \text{if } label(v_i) = label(u_j), \\ 0 & \text{otherwise;} \end{cases}$$

$$I(v_i, v_k, u_j, u_l) = \mathbb{1}_{(v_i,u_j)\rightsquigarrow(v_k,u_l)} \cdot \mathbb{1}_{\rho(v_i,v_k)\leq L} \cdot \mathbb{1}_{(v_i \text{ is an ancestor of } v_k)},$$

where $\mathbb{1}_c$ equals 1 when constraint c is satisfied and 0 otherwise. Note that the first two factors of indicator function I just represent the general Skip-node space constraints, the last factor ensures that features are computed along the root-paths.

Lookahead Skip-node The second approximation, *Lookahead Skip-node*, is related to the observation that when growing a substructure, we do not have to confine the growth only towards ancestors, as DFS traversal already ensures iterative manner of computation. In other words, the constraint v_i is an ancestor of v_k can be dropped:

$$I(v_i, v_k, u_j, u_l) = \mathbb{1}_{(v_i,u_j)\rightsquigarrow(v_k,u_l)} \cdot \mathbb{1}_{\rho(v_i,v_k)\leq L}.$$

In addition to those features generated by Linear Skip-node in Figure 3.5(a), Lookahead Skip-node can generate additional tree substructures, shown in Figure 3.5(b). The approximation can be computed using different DFS enumerations, which may result in different feature sets. In our experiments, we used pre-order left-to-right enumeration. Given the enumeration of tree T as in Figure 3.5, we start to grow feature fragments from node n_4. According to the Skip-node space constraints, the growth can only proceed to nodes n_1 or n_2. Once any of these nodes is attached to n_4, we lose tree fragments containing n_3, as the procedure allows us to grow substructures only towards nodes with smaller (earlier) DFS enumeration numbers. Figure 3.5(c) shows the fragments that Lookahead Skip-node cannot capture[2].

[2]In this particular case, all features could have been computed by Lookahead Skip-node using preorder right-to-left DFS enumeration, although it may not be true in general.

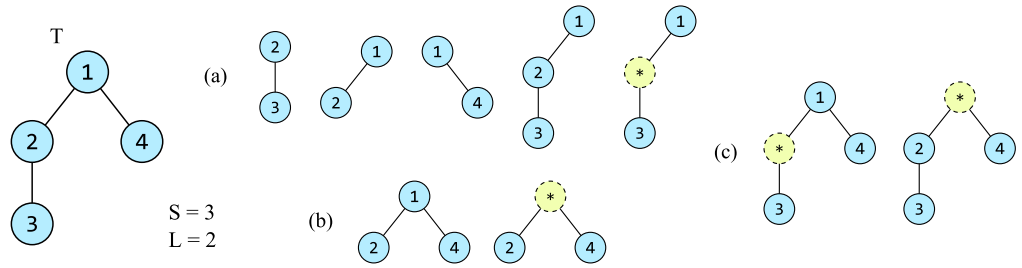


Figure 3.5: Features of T in skip-node space $(S = 3, L = 2)$. The numbers indicate pre-order left-to-right DFS enumeration of T. The dashed circles represent skip nodes. Subfigures: (a) - modeled by all; (b) - modeled by Lookahead Skip-node, but not by Linear Skip-node; (c) - modeled only by Exact Skip-node.

Domain	# sentences	% comp.	# pairs	% comp.
Camera	1716	59.4%	2170	49.9%
Cell	821	35.2%	1110	30.5%

Table 3.2: Dataset Statistics for Each Domain.

The computation procedure is similar for both approximations and requires $\mathcal{O}(S|N_1|^2|N_2|^2)$.

3.4 Experiments

Data For experiments, we compiled two annotated datasets in two domains: Digital Camera and Cell Phone from online review sentences. The reviews were collected from *Amazon* and *Epinions*[3].

We identified the entity mentions through dictionary matching, followed by manual annotation to weed out false positives. Each dictionary entry is a product name (e.g., *Canon PowerShot D20*, *D7100*) or a common product reference (e.g., *this camera*, *that phone*). The dataset includes only sentences that contain at least two entity mentions. Every pair of entities within a sentence was annotated with a comparative label according to the definition given in Section 3.1. A sentence is comparative if at least one pair of entities within it is in a comparative relation. Table 3.2 shows the dataset properties, in terms of the number of sentences and the

[3]We used already available snapshots for *Epinions* dataset: http://groups.csail.mit.edu/rbg/code/precis/.

	Camera			Cell		
	P	R	F1	P	R	F1
CSR	74.3	52.3	61.3	48.9	**61.5***	54.3
BoW	76.9	**76.3**	76.6	62.2	58.0	59.8
BoW†	77.3	71.9	74.4	69.0	56.3	61.8
SNK	**80.5***	75.2	**77.7****	**77.2***	55.1	**64.1***

Table 3.3: Comparison Identification.

percentage that are comparative sentences, as well as the number of pairs of entity mentions and the percentage that are comparative relations. There are more pairs than sentences, i.e., many sentences mention more than two entities.

This dataset subsumes the annotated gradable comparisons of [86] derived from *Epinions* reviews on Digital Cameras. [76]'s dataset is inapplicable, due to its lack of entity-centric comparison.

Evaluation The experiments were carried out with SVM-light-TK framework[4] [79, 125], into which we built Skip-node Kernel. We further release a separate standalone library that we built, called Tree-SVM, which does SVM optimization using the tree kernels described in this study. The sentences were parsed and lemmatized with the use of the Stanford NLP software [26].

The experiments were done on 10 random data splits in $80{:}20$ proportion of training vs. testing. Performance is measured by using F_1, which is the harmonic mean of precision P and recall R: $F_1 = \frac{2PR}{P+R}$. The average results are reported. The statistical significance[5] is measured by randomization test [190]. The hyper-parameters, including the baselines', were optimized for F_1 through grid-search.

3.4.1 Comparison Identification

Our first and primary objective is to investigate the effectiveness of the proposed approach on the task of identifying comparisons between a pair of entity mentions.

[4] http://disi.unitn.it/moschitti/Tree-Kernel.htm

[5] When presenting the results, an asterisk indicates that the outperformance over the second-best result is significant at 0.05 level. Two asterisks indicate the same at 0.1 level.

	Camera			Cell		
	P	R	F1	P	R	F1
CSR	74.6	51.7	60.9	50.9	**61.2***	55.3
BoW	77.5	**76.3**	76.8	63.4	57.7	60.2
BoW†	77.6	72.4	74.9	70.9	57.3	63.2
SNK	**81.0***	75.2	**78.0****	**77.9***	54.8	**64.2**

Table 3.4: Comparative Sentence Identification.

Previous work focused on identifying comparative sentences. We compare to three baselines. One is CSR, implemented following the description in [76]. Another is BoW, classification using bag-of-words as features. For the baselines, if a comparative sentence contains more than one pair of entities, we assume that every pair is in comparative relation. The third baseline, BoW†, considers only the words in between of the two target entities.

Table 3.3 shows the performance on the comparison identification task (best results are in bold). In terms of F_1, it is evident that SNK outperforms the baselines. This is achieved through significant gains in precision. It is expected that the baselines tend to have a high recall. CSR benefits from the human-constructed predefined list of comparative keywords and key phrases that a kernel-based method is unable to learn from a training split. BoW† tends to have a higher precision than the other baselines, as it is able to distinguish between different pairs of entities within one sentence.

While SNK may have an inherent advantage over CSR or BoW due to its entity orientation, to investigate the effectiveness of the method itself, we now compare them on the previous task of comparative sentence identification. Table 3.4 shows that even in this task, SNK still performs better than the baselines. Comparing Table 3.3 and Table 3.4, the results also concur with the intuition: once we fold up multiple entity pairs in a sentence into a comparative sentence, we observe a drop in recall and an increase in precision.

	Camera			Cell		
	P	R	F1	P	R	F1
STK	67.5	64.0	64.9	43.7	41.9	42.6
SSTK	72.1	72.6	71.8	**79.6**	42.4	54.9
PTK	79.2	74.9	76.9	72.3	**56.0****	62.7
SNK	**80.5***	**75.2**	**77.7****	77.2	55.1	**64.1***

Table 3.5: Comparison Identification: Tree Kernels.

	Camera			Cell		
	P	R	F1	P	R	F1
STK_{BoW}	79.9	65.1	71.7	**77.5**	45.3	56.8
$SSTK_{BoW}$	78.0	73.5	75.6	71.8	54.5	61.6
PTK_{BoW}	78.6	74.1	76.2	71.0	53.8	60.8
SNK	**80.5**	**75.2****	**77.7***	77.2	**55.1**	**64.1****

Table 3.6: Tree Kernels Combined with Bag-of-Words.

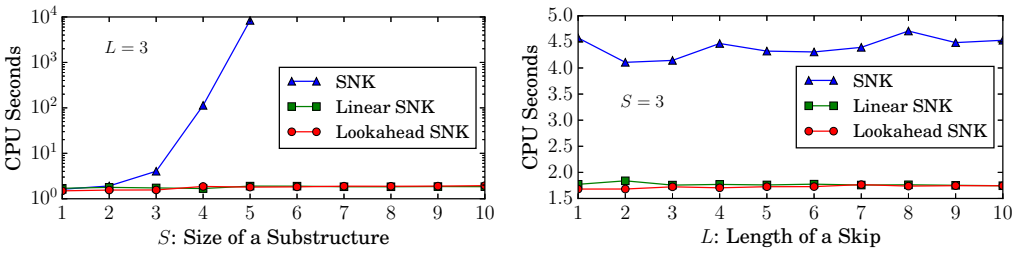
3.4.2 Tree Kernel Spaces

Our second objective is to explore the progression of feature spaces discussed in Section 3.2. Table 3.5 reports the results on comparison identification task. The F_1 columns show that the performance gradually increases from STK to SNK along with the increase in the complexity of feature space. PTK and SNK can be considered high-variance estimators due to the power of their feature spaces. The data is such that these kernels may not have fully modeled the feature space completely enough to show even sharper differences.

SNK's parameters were optimized to non-trivial cases ($S > 1$ and $L > 1$) by the grid-search, i.e., $S = 3$ and $L = 2$ for Digital Camera and $S = 2$ and $L = 3$ for Cell Phone. The trivial case $S = 1$ represents a standard bag-of-words feature space, i.e., this space is embedded into Skip-node space whenever $S > 1$. To show that SNK does not merely take advantage of this simple space to compete with structural kernels, we carried out another experiment where we combined STK, SSTK, and PTK with bag-of-word representation of a sentence. Table 3.6 shows that surpris-

	Camera			Cell		
	P	R	F1	P	R	F1
Linear SNK	78.9	**77.1***	**77.9**	71.8	**55.3**	62.2
Lookahead SNK	**80.5**	75.2	77.7	71.8	**55.3**	62.2
SNK	**80.5**	75.2	77.7	**77.2***	55.1	**64.1**

Table 3.7: Effectiveness: SNK vs. Approximations.



(a) $L = 3, S \in 1..10$ (b) $S = 3, L \in 1..10$

Figure 3.6: Efficiency: SNK vs. Approximations.

ingly this combination harms the quality of PTK. STK and SSTK gain more from bag-of-words features. Nevertheless, the overall outperformance by SNK remains.

3.4.3 Skip-node Kernel Approximations

Our third objective is to study the utility of the approximations of SNK described in Section 3.3. Table 3.7 reports the performance of the approximations. For Camera, the performance of Lookahead SNK and SNK are the same. In turn, Linear SNK represents more restricted features, yielding a drop in precision and a gain in recall, resulting in the best F_1. For Cell Phone, the approximations are close, but the original SNK has the best F_1.

To study the running time, we randomly select 500 sentences. Figure 3.6 shows the time for applying a kernel function to 250k pairs of sentences when we vary two parameters: S and L. When S varies, SNK running time has exponential behaviour, whereas the approximations show fairly linear curves. L seems to influence the computation time linearly for SNK and and its approximations. The experiments were carried out on a PC with Intel Core i5 CPU 3.2 GHz and 4Gb RAM.

This experiment shows that the original SNK is still tractable for small S and L, which turn out to be the case for optimal effectiveness. If efficiency is of paramount importance, the two approximations are significantly faster, without much degradation (none in some cases) of effectiveness.

3.5 Discussion

Skip-node kernel gives another perspective on sparsity, employing structural alignment of the tree fragments, when labels cannot be matched exactly. This approach differs from [32, 163], which allow soft label matching via lexical similarity over distributional word representation, yet lexical similarity can be incorporated into Skip-node kernel. Skip-node kernel may also be applied to other types of trees (e.g., constituency trees [196]) or directed acyclic graphs.

Once the well-rolled methodology for identifying comparison is designed, we must proceed with the comparison interpretation. Having a large corpus of comparisons, we would like to explore if it is possible to induce the user preferences for each individual sentence (which entities is preferred in a comparison) as well as to induce the overall ranking of the entities with respect to the observed corpus. Chapter 4 investigates this question.

CHAPTER 4

MINING COMPARATIVE RELATIONS

Given the abundance of text reviews on the Web, we seek to mine these reviews to assist consumers in making well-informed comparisons of entities. That would allow consumers to benefit from the wisdom of the crowd to determine the relative quality of entities, from users' vantage point.

For *comparison mining*, the basis for comparison is a comparative sentence about two entities. The following example compares CANON EOS 50D vs. CANON EOS 40D in terms of image quality: *"The 50D is sharper than my 40D and the images are not soft."* One user provides a common benchmark and context in comparing two entities. Table 4.1 shows several more examples for two pairs of digital cameras. To maintain focus, we deal with sentences involving two entities. From sentences s_1 to s_3, we observe some variance in terms of which entity is considered better, and the words used to express the comparison. Sentences s_4 and s_5 give examples for different entities and aspects.

Problem. Given a corpus of comparative sentences, relating pairs of entities in a particular domain (e.g., digital cameras), we seek to derive the comparative relations among the entities, i.e., between any two comparable entities, which one is better with respect to each aspect. The input corpus of comparative sentences may be obtained from user-generated content expressing user preferences, such as reviews, through comparative sentence identification (see Section 4.5).

Naturally, comparative relations ought to be modeled at two levels. First, at the

Entities	ID	Aspect	Example Comparative Sentences
CANON EOS 40D CANON EOS 50D	s_1	Image Quality	I am surprised to see that the images on the 40D are better than the 50D.
	s_2	Image Quality	And from the research I did it appears the 50D's images can be sharpened and still have more detail than the 40D.
	s_3	Image Quality	The 50D is sharper than my 40D and the images are not soft.
CANON REBEL XSI CANON EOS 40D	s_4	Functionality	After visiting Best Buy and actually trying the cameras out the XSi felt like a toy compared to its big brother the 40D.
	s_5	Form Factor	I picked the XSi over the 40D primarily because of weight (I like to hang cameras off telescopes , weight is an issue).

Table 4.1: Comparative Sentences about Digital Cameras from Amazon.com.

level of a sentence, e.g., s_1 in Table 4.1 favors CANON EOS 40D, while s_3 favors CANON EOS 50D. Second, at the level of entity pairs, whereby sentence-level relations are aggregated into the comparative relation, e.g., whether CANON EOS 50D is better than CANON EOS 40D. Both are important and useful. The former provides supporting evidence, the latter provides a summative view.

In addition, comparative relations also need to be studied in the context of each aspect. For instance, if one camera is lighter than another, it does not necessarily imply that it would also have a better image quality. Moreover, the words used to express superiority or mediocrity vary across aspects. While *"higher"* may connote positively for functionality or image quality, it may connote negatively for price.

Approach. The previous approach to deal with the afore-mentioned two levels of comparison is to solve them as a pipeline, by first determining sentence-level comparisons, and then aggregating them into entity-level comparisons. Not only is this fragmentation unnecessary, but it could also be detrimental when errors from one level propagate to the next.

In this study, we propose an *integrated* approach to exploit the synergy owing to the inherent relation between sentence-level and entity-level comparisons. Intuitively, if one entity is indeed better than another, we would expect that many comparative sentences would compare the former favorably to the latter. Thus, knowing which entity is better helps to determine the comparison in a sentence, and vice

ID	Training Sentence	ID	Testing Sentence
d_1	e_1 is smaller than e_2	d_5	e_1 is thinner than e_3
d_2	e_2 is smaller than e_3	d_6	e_4 is thinner than e_5
d_3	e_3 is smaller than e_4		
d_4	e_3 is smaller than e_5		

Table 4.2: Illustrative Corpus.

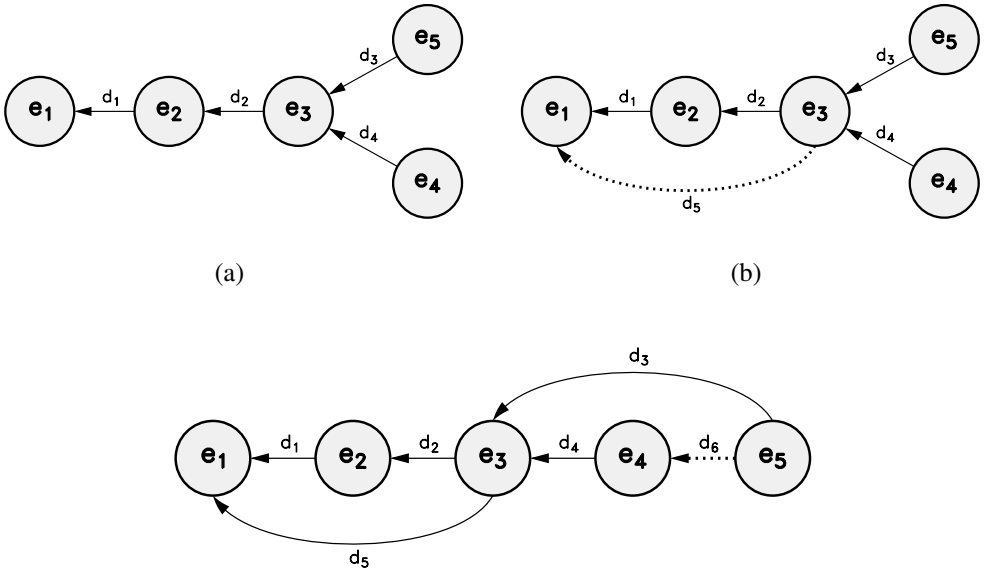


Figure 4.1: Comparison Graph Based on Table 4.2

versa.

We now illustrate this important intuition with a mock-up example in Figure 4.1 involving the 6 sentences shown in Table 4.2, concerning 5 entities $\{e_1, e_2, e_3, e_4, e_5\}$. Let us suppose the meaning of the first four sentences $\{d_1, d_2, d_3, d_4\}$ with the word *"smaller"* is already known. For form factor, *"smaller"* is better. That will allow us to confidently rank some pairs, by drawing a bold directed edge from the worse entity to the better entity, e.g., from e_2 to e_1, since *"e_1 is smaller than e_2"*. Figure 4.1(a) show the comparison graph constructed from these "known" sentences.

From here, we could make further inferences to answer another couple of questions. One is which of e_4 or e_5 is better, since there is no clue from the bold edges alone. Another is the meaning of the last two sentences, since we have not yet understood the meaning of *"thinner"*. Considering these two questions separately

does not offer an answer. However, jointly they allow us to arrive at an answer to both.

Since $e_1 \leftarrow e_2 \leftarrow e_3$, by transitivity, we can infer that $e_1 \leftarrow e_3$, and update the comparison graph with the dotted arrow as in Figure 4.1(b). In turn, if $e_1 \leftarrow e_3$, the interpretation of *"e_1 is thinner than e_3"* can be inferred, i.e., *"thinner"* implies the first-mentioned entity is better. This allows us to parse the last sentence to infer that $e_4 \leftarrow e_5$ (dotted). We thus can recover the correct rank order $e_1 \leftarrow e_2 \leftarrow e_3 \leftarrow e_4 \leftarrow e_5$ (see Figure 4.1(c)).

We leverage on the above intuition to build a joint model for learning the comparative relations among entities, both at the sentence level and the entity level. We propose an integrated approach for comparative relation mining, which is novel compared to the previous pipelined approaches. Our integrated formulation is presented in Section 4.1. We design a generative model (see Section 4.2), called *CompareGem*, which stands for COMPArative RElation GEnerative Model. We turn to *generative modeling* because it offers significant advantages in connecting sentence-level and entity-level comparisons seamlessly. It has greater flexibility in accommodating supervised and unsupervised settings. While aspect identification is not our main focus, it is a necessary component for studying comparisons, and *CompareGem* has the flexibility of both working with observed aspects, as well as learning latent aspects. In order to accommodate different formats of ranking entities, either via a discrete or a continuous range of rank scores, we develop two inference algorithms for *CompareGem*, based on Gibbs sampling and Variational method respectively (see Section 4.3). In Section 4.4, through experiments on real datasets, we show that *CompareGem* outperforms the pipelined baselines, underlining the utility of integrated approach for comparative relation mining. Section 4.5 conclude this chapter with a discussion.

4.1 Problem

As input, we consider set of entities E (e.g., digital cameras). For each pair of entities $e_i, e_j \in E$, S_{ij} denotes the set of comparative sentences involving e_i and e_j. For instance, in Table 4.1, the pair of entities CANON EOS 50D and CANON EOS 40D are associated with a set of three comparative sentences $\{s_1, s_2, s_3\}$ on image quality. Some pairs may not have any comparative sentence, if they are never compared by any user, i.e., $S_{ij} = \emptyset$. The union is denoted $\mathcal{S} = \bigcup_{e_i,e_j \in E} S_{ij}$. We will describe how $\mathcal{S}$ can be obtained from a corpus of reviews in Section 4.4.1.

Our objective is to learn the comparative relation between any two entities e_i or e_j. Using the example of CANON EOS 50D and CANON EOS 40D in Table 4.1, we see that s_1 favors CANON EOS 40D, whereas s_2 and s_3 favor CANON EOS 50D. In this case, there is slightly more evidence that CANON EOS 50D is better in image quality. The more evidence there is in favour of one direction, the more confident we would be. The aspect $a \in A$ of a comparison (e.g.., image quality) is to be derived, where A denotes a set of possible aspects. We assume a sentence belongs to only one aspect. S_{ija} denotes a set of comparative sentences on aspect a involving entities e_i and e_j.

To capture the notion of aggregative "quality", we associate each entity with an aspect-specific rank score $r_{ia} \in \mathbb{R}$. e_i is "better" than e_j on aspect a if $r_{ia} > r_{ja}$. This rank score is latent, and needs to be learnt. The sentences are not always unanimous in terms of which entity is favored. Even when there is a consensus, there may also be some variance. It is important not just to capture the relation at the entity level, but also the comparative direction at the sentence level.

With the notations in place, we are now ready to state our problem formally, as follows.

Problem 1 (Comparative Relation Mining) *Given a set of entities E and the associated corpus of comparative sentences $\mathcal{S}$, find:*

- *For every sentence $s \in \mathcal{S}$, its aspect a,*

- *For every sentence $s \in \mathcal{S}$ about a pair of entities e_i and e_j, the comparative direction (or comparison outcome), i.e., whether e_i or e_j is favored by s,*
- *For every entity $e_i \in E$ and every aspect $a \in A$, the rank score r_{ia} of the entity.*

4.2 Model

We discuss the modeling of features in comparative sentences, before describing our *CompareGem* model.

4.2.1 Bag of Features

The convention in modeling text, either for classification [115] or topic modeling [11], is to model a document as a bag of words, due to the assumption of exchangeability of words within a document. Only the frequencies of words, and not the sequence, matter.

In our case, the unit of interest is a sentence. A bag-of-words model is not appropriate for modeling a *comparative* sentence. Recognizing the favored entity in a comparative sentence is challenging due to complex sentence structure, whereby word order now becomes important. Let us consider the following comparative sentence: *"The 50D is sharper than my 40D"*. In terms of the bag-of-words model, the order between *50D* and *40D* could be swapped interchangeably. In fact, swapping those two words would change the meaning of the comparison completely.

We observe that the entity position is important. We distinguish whether a word appears before the first-mentioned entity, in between, or after the second-mentioned entity. For example, the word "sharper" may translate to a feature $\langle \#1\ sharper\ \#2 \rangle$, where #1 and #2 refer to first- and second-mentioned entities.

We model each comparative sentence s as a bag of *features*, where each feature w is drawn from a vocabulary of features W. The bag representation maintains the feature frequencies within each sentence. The complete representation

for the considered comparative sentence follows: $\{\langle the\ \#1\ \#2\rangle, \langle \#1\ is\ \#2\rangle, \langle \#1\ sharper\ \#2\rangle, \langle \#1\ than\ \#2\rangle, \langle \#1\ my\ \#2\rangle\}$.

4.2.2 Generative Model

To help illustrate *CompareGem* in terms of feature generation, we may refer to comparative sentences related to various aspects of digital cameras. Sentences $s_1 - s_3$ from Table 4.1 may be used for reference.

Generating Features We first observe that within a corpus, there are sentences that belong to different aspects (e.g., image quality, functionality), and that frequently each sentence focuses on one aspect. Hence, each sentence s is associated with one of $|A|$ aspects. This is done with the use of a categorical distribution π over A. Furthermore, some features in a sentence provide information on background words or words that encode the relevant aspect (e.g., *"surprised"*, *"images"*, *"detailed"*). We therefore introduce for each aspect a background distribution θ_{ba}, which defines a distribution over common features.

Yet other features are helpful in discovering the comparison outcome: whether a sentence favors the first-mentioned entity (e.g., *"sharper"*, *"more"*) or the second (e.g., *"heavier"*). We introduce two more feature distributions. $\theta_{\succ a}$ is a distribution over features when the first-mentioned entity is favored, in which case features involving words such as *"better"*, *"sharper"* have higher probabilities. $\theta_{\prec a}$ is the same for when the second-mentioned entity is favored.

Every feature in a sentence is associated with binary variable ν indicating whether the feature is drawn from the background distribution θ_{ba}, or from one of $\theta_{\succ a}$ or $\theta_{\prec a}$. Every ν is a sample of the Bernoulli distribution with parameter γ, which can be understood as the expected proportion of common features.

Comparison Outcome. Each sentence s expresses a comparison outcome involving two entities (say e_i and e_j). Which of $\theta_{\succ a}$ or $\theta_{\prec a}$ is used to draw the comparative features in a sentence is indicated by a variable $c_s \in \{\prec, \succ\}$. The event $c_s = \succ$ means the first-mentioned entity is favored, whereas $c_s = \prec$ means the

second-mentioned entity is favored. For simplicity, we do not model a draw, which would not influence the ranking between the two entities. We now can specify the distribution over sentence feature, as follows:

$$\mathrm{P}(w|\theta, c_s, a_s, \nu_{sw}) = \big(\mathrm{P}(w|\theta_{ba_s})\big)^{\mathrm{I}_{[\nu_{sw}=0]}} \big(\mathrm{P}(w|\theta_{\succ a_s})\big)^{\mathrm{I}_{[\nu_{sw}=1 \wedge c_s=\succ]}} \big(\mathrm{P}(w|\theta_{\prec a_s})\big)^{\mathrm{I}_{[\nu_{sw}=1 \wedge c_s=\prec]}} \tag{4.1}$$

where $\mathrm{I}_{[\cdot]}$, an indicator function, equals 1 when its condition is true, and 0 otherwise.

The outcome of c_s depends on an underlying distribution. We associate each entity $e_i \in E$ with a rank score r_{ia} that reflects the quality of e_i with respect to aspect a. Intuitively, the higher r_{ia} is than r_{ja}, the higher is the probability that a comparative sentence favors e_i. One suitable probability function is sigmoid, as in (4.2), supposing the first-mentioned entity is e_i and the second-mentioned entity is e_j.

$$\begin{aligned} \mathrm{P}(c_s = 0|r_{ia}, r_{ja}) &= \mathrm{P}(e_i \text{ is better than } e_j|r_{ia}, r_{ja}) \\ &= \sigma_\upsilon(r_{ia} - r_{ja}) = \frac{1}{1 + e^{-\upsilon(r_{ia} - r_{ja})}} \end{aligned} \tag{4.2}$$

If r_{ia} is significantly higher than r_{ja}, the probability would tend towards 1, reflecting e_i's much higher quality. If $r_{ia} = r_{ja}$, the probability is 0.5, reflecting the uncertain outcome between two evenly matched entities. Conversely, if r_{ia} is significantly lower than r_{ja}, the probability would tend towards 0. The parameter υ models the sensitivity to the difference between two rank scores. If υ is large, small differences in scores would have a high impact on the probabilities.

Generative Process. *CompareGem*'s plate notation is shown in Figure 4.2, the notation is explained in Table 4.3. First, the model assigns each sentence $s \in \mathcal{S}$ to one of the $|A|$ aspects. Once the aspect is assigned, two entities mentioned in the sentence can compete along the comparison dimension specified by the aspect, and we generate the comparison outcome (which entity is favored in s). Thereafter,

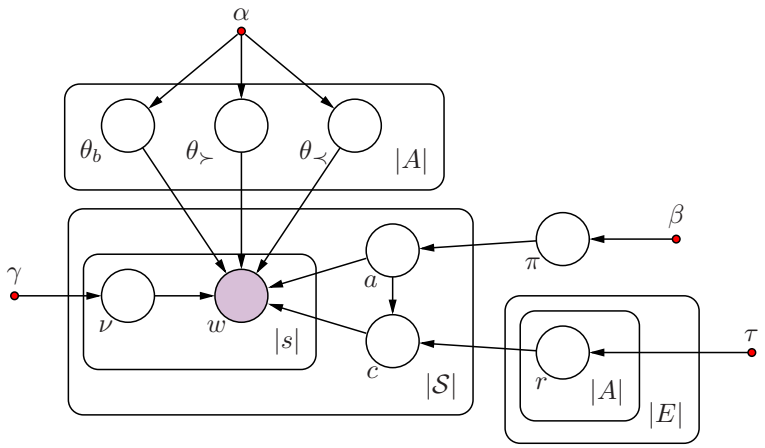


Figure 4.2: *CompareGem* in Plate Notation.

Notation	Description
α	feature-related Dirichlet distribution parameter
β	aspect-related Dirichlet distribution parameter
γ	Bernoulli distribution parameter
τ	ranking score distribution parameters
θ_{ba}	background feature distribution for aspect a
$\theta_{\succ a}, \theta_{\prec a}$	feature distributions for aspect a when the first-mentioned and the second-mentioned entities are favored respectively
π	topic proportion
w	feature
a_s	aspect of sentence s
c_s	comparative direction of sentence c
r_{ia}	ranking score for entity e_i with respect to aspect a
ν_{sw}	background indicator for feature w in sentence s

Table 4.3: Notations.

based on the comparison outcome, we generate each feature $w \in s$.

The full generative process is as follows:

1. For a given corpus, we sample π, an aspect proportion from the Dirichlet distribution[1]:

$$\pi \sim Dirichlet(\beta)$$

2. For every aspect, all $\theta_{\succ a}$, $\theta_{\prec a}$ and θ_{ba} are sampled from the Dirichlet distribution with α prior:

$$\theta_{\succ a}, \theta_{\prec a}, \theta_{ba} \sim Dirichlet(\alpha)$$

3. For each entity $e_i \in E$ and for each aspect $a \in A$, we sample rank score r_{ia} from distribution F with some parameter set τ, we will discuss the form of the distribution in Section 4.3:

$$r_{ia} \sim F(\tau)$$

4. For every sentence $s \in \mathcal{S}$ involving two entities e_i (first-mentioned) and e_j (second-mentioned):

   (a) Sample sentence aspect a_s:

$$a_s \sim Categorical(\pi)$$

   (b) Sample comparison outcome c_s:

$$c_s \sim Bernoulli(\sigma_v(r_{ia_s} - r_{ja_s}))$$

   (c) Sample ν_{sw} for each feature w in s:

$$\nu_{sw} \sim Bernoulli(\gamma)$$

[1]For detailed information on the distributions used in this study: Dirichlet, Categorical, Bernoulli, please refer to [7].

(d) Sample each w in sentence s using appropriate feature distribution, see (4.1):

$$w \sim \mathrm{P}(w|\theta, c_s, a_s, \nu_{sw})$$

As in Figure 4.2, the only observed (shaded) variables are the features w's within each sentence s. All others are latent. The likelihood function of an assignment of scores $R = \{r_{ia}\}_{e_i \in E, a \in A}$, comparison outcomes $C = \{c_s\}_{s \in \mathcal{S}}$, aspects $A = \{a_s\}_{s \in \mathcal{S}}$, latent distributions over features $\theta = \{\theta_{\succ a}, \theta_{\prec a}, \theta_{ba}\}_{a \in A}$ and aspects π, and background indicators $H = \{\nu_{sw}\}_{s \in \mathcal{S}, w \in s}$, is shown in (4.3).

$$\begin{aligned}
&\mathcal{L}(\pi, \theta, R, A, C, H|\alpha, \beta, \gamma, \tau) = \\
&\mathrm{P}(\pi|\beta) \times \prod_{s \in \mathcal{S}} \mathrm{P}(a_s|\pi) \times \prod_{a \in A} \mathrm{P}(\theta_{ba}|\alpha)\mathrm{P}(\theta_{\succ a}|\alpha)\mathrm{P}(\theta_{\prec a}|\alpha) \times \\
&\prod_{a \in A} \prod_{e_i \in E} \mathrm{P}(r_{ia}|\tau) \times \prod_{e_i, e_j \in E} \prod_{s \in S_{ij}} \mathrm{P}(c_s|r_{ia_s}, r_{ja_s}) \times \\
&\prod_{s \in \mathcal{S}} \prod_{w \in s} \mathrm{P}(\nu_{sw}|\gamma)\mathrm{P}(w|\theta, c_s, a_s, \nu_{sw})
\end{aligned} \tag{4.3}$$

Once the model parameters are learned, we obtain the solution to the Problem 1 defined in Section 4.1.

We now illustrate how *CompareGem* captures the intuition of the integrated approach with only one aspect. We use the same corpus as before (see Table 4.2). The rank scores of entities are samples of some $F(\tau)$ distribution. A priori, we assume no difference among the entities. When sentences $d_1 - d_4$ are given for training, the scores of the entities should be shifted to satisfy the corpus observation: e_1 should be better than e_2 ($r_1 > r_2$), e_2 is better than e_3 ($r_2 > r_3$), etc.

There are two interpretations for the test sentences. In one interpretation, we infer the wrong meaning of sentences d_5 and d_6, so the second-mentioned entity is considered better. This places e_5 before e_4 ($r_5 > r_4$) and e_3 before e_1 ($r_3 > r_1$). However, the last placement makes this ranking less probable, since it is in conflict with $r_1 > r_2 > r_3$ according to training sentences d_1 and d_2. In the other

interpretation, when d_5 and d_6 are correctly parsed, this contradiction is resolved. As d_5 is consistent with d_1 and d_2, the scores satisfying $r_1 > r_2 > r_3 > r_4 > r_5$ give higher likelihood.

4.3 Inference

There are two options in modeling the rank score distribution. One is to model it along a continuous spectrum, with a Gaussian prior for the distribution of r_{ia}'s, which encode the prior belief that most entities are of "average" rank scores, while some are very high or low. F can be a Gaussian specified by its mean and standard deviation $\tau = (\mu, \sigma)$. The mean is assumed to be zero. σ acts as a regularization parameter.

Another option is to have a discretized model, with n ranking steps in the scale of 0 to $n-1$. The prior $F(\tau)$ can thus be simulated by a binomial distribution $Binomial(n-1, p_0)$, where p_0 is the probability of success in a Bernoulli trial ($p_0 = 0.5$ for our model). This prior encodes the same information as a Gaussian, shrinking the rank scores towards the mean.

Both approaches for rank score modeling are acceptable. The use of a particular model can come from the specific tasks and needs. Due to the difference in mathematical formulations, these two models can take advantage of different optimization methods. Variational method is employed to fit the model with continuous rank scores. Gibbs sampling is used to maximize a posteriori distribution over the hidden variables when discrete model is assumed.

4.3.1 Continuous Model via Variational Method

Variational approximation can be used to solve complex Bayesian models. To make the posterior distribution tractable for computation, one can assume a family of distributions over the hidden variable with its own parameters. The approximate distributions are denoted $q(\cdot)$. The lower bound optimization of the likelihood can

be performed. For Bayesian model, the factorized form of a distribution stemming from the mean field theory has been used with great success. We assume that every hidden variable has its own distribution, which is independent from the others:

$$\mathcal{L}(\pi, \theta, R, A, C, H|\alpha, \beta, \gamma, \tau) \approx q(R)q(\pi)$$
$$\prod_{a \in A} q(\theta_{ba})q(\theta_{\succ a})q(\theta_{\prec a}) \times \prod_{s \in \mathcal{S}} q(c_s)q(a_s) \prod_{w \in s} q(\nu_{sw}) \tag{4.4}$$

Since the priors for the rank scores R are not conjugate, direct computation may not be tractable. We assume a parametric form of $q(R) = q(R|\hat{R})$:

$$q(R|\hat{R}) = \prod_{a \in A} q(R_a|\hat{R}_a) = \prod_{a \in A} \prod_{e_i \in E} \mathrm{I}_{[r_{ia} = \hat{r}_{ia}]} \tag{4.5}$$

$R_a = \{r_{ia}\}_{e_i \in E}$ denotes the set of aspect-specific rank scores. Though aspect-specific factorization is not necessary, rank scores are assumed independent. This factorization is employed for parameter optimization. An indicator probability function put the whole probability mass to the value specified by parameter set $\hat{R}$.

Let Z be the set of the hidden variables. $q(z_i)$ denotes a probability density function for variable z_i. Once approximation distributions are specified, we can run Variational Method to estimate $\{q(z_i)\}_{z_i \in Z}$ and optimize model parameters. To find $q(z_i)$ that maximizes the lower bound given $\{q(z_j)\}_{z_j \neq z_i}$ fixed, the following equation has to be solved:

$$q(z_i) = \frac{1}{L} e^{\mathrm{E}_{z_j \neq z_i}[\ln \mathcal{L}(Z)]}, \text{ where} L = \int e^{\mathrm{E}_{z_j \neq z_i}[\ln \mathcal{L}(Z)]} \, dz_i. \tag{4.6}$$

We work with (4.6) in logarithmic form:

$$\ln q(z_i) = \ln \mathrm{E}_{z_j \neq z_i}[\ln \mathcal{L}(Z)] - \ln L \tag{4.7}$$

Taking into account the fact that constant $\ln L$ can be obtained through normaliza-

tion, we can drop it in our notation. Instead of Equation 4.7 we write:

$$\ln q(z_i) \cong \ln \mathrm{E}_{z_j \neq z_i}[\ln \mathcal{L}(Z)]. \tag{4.8}$$

We iteratively estimate required distributions in a round robin manner. The update procedures are shown below. $\psi(x)$ denotes the digamma function. $[s]_1$ is the index of the first-mentioned entity in sentence s, $[s]_2$ is the index of the second-mentioned entity.

Estimating $q(\pi)$.

$$\ln q(\pi) \cong \ln \prod_{a \in A} \pi_a^{\beta_a - 1}, \text{ where} \beta_a = \sum_{s \in \mathcal{S}} q(a_s = a) + \beta \tag{4.9}$$

Estimating $q(\theta_{ba})$.

$$\ln q(\theta_{ba}) \cong \ln \prod_{w \in V} (\theta_{ba})_w^{\alpha_{ba}^w + 1}, \text{ where} \tag{4.10}$$

$$\alpha_{ba}^w = \sum_{s \in \mathcal{S}} \sum_{w' \in s} \mathrm{I}_{[w'=w]} q(\nu_{sw} = 0) q(a_s = a) + \alpha \tag{4.11}$$

Estimating $q(\theta_{\prec a})$ and $q(\theta_{\succ a})$.

$$\ln q(\theta_{ca}) \cong \ln \prod_{w \in V} (\theta_{ca})_w^{\alpha_{ca}^w + 1}, \text{ where} \tag{4.12}$$

$$\alpha_{ca}^w = \sum_{s \in \mathcal{S}} \sum_{w' \in s} \mathrm{I}_{[w'=w]} q(\nu_{sw} = 1) q(a_s = a) q(c_s = c) + \alpha \tag{4.13}$$

Estimating $q(\nu_{ws})$.

$$\ln q(\nu_{ws}) \cong \ln \mathrm{P}(\nu_{ws}|\gamma) + \mathrm{I}_{[\nu_{ws}=0]} \sum_{a \in A} q(a_s = a) \big(\psi(\alpha_{ba}^w) - \psi(\sum_{w \in V} \alpha_{ba}^w)\big) + \\ \mathrm{I}_{[\nu_{ws}=1]} \sum_{c \in \{\prec, \succ\}} \sum_{a \in A} q(a_s = a) \big(\psi(\alpha_{ca}^w) - \psi(\sum_{w \in V} \alpha_{ca}^w)\big), \tag{4.14}$$

Estimating $q(a_s)$.

$$\ln q(a_s) \cong \psi(\beta_{a_s}) - \psi(\sum_{a\in A}\beta_a) + \sum_{c\in\{\prec,\succ\}} q(c_s=c)\ln \mathrm{P}(c|\hat{r}_{[s]_1a_s},\hat{r}_{[s]_2a_s}) + \sum_{w\in s} q(\nu_{sw}=0)\big(\psi(\alpha^w_{ba_s}) - \psi(\sum_{w\in V}\alpha^w_{ba_s})\big) + \sum_{c\in\{\prec,\succ\}} q(c_s=c)\sum_{w\in s} q(\nu_{sw}=1)\big(\psi(\alpha^w_{ca_s}) - \psi(\sum_{w\in V}\alpha^w_{ca_s})\big) \tag{4.15}$$

Estimating $q(c_s)$.

$$\ln q(c_s) \cong \sum_{a\in A} q(a_s=a)\Big(\ln \mathrm{P}(c_s|\hat{r}_{[s]_1a},\hat{r}_{[s]_2a}) + \sum_{w\in s} q(\nu_{sw}=1)\big(\psi(\alpha^w_{c_sa}) - \psi(\sum_{w\in V}\alpha^w_{c_sa})\big)\Big) \tag{4.16}$$

Estimating $q(R|\hat{R})$. To update $\hat{R}$, we compute evidence lower bound $\mathcal{F}$ of the log likelihood and maximize it via gradient ascent. The form of $q(R|\hat{R})$ allows us to update all the parameters at a time. The form of $\mathcal{F}$ makes it possible to update the parameters independently for every aspect $a \in A$, i.e., $\hat{R}_a$. The lower bound is computed up to an additive constant, which can be ignored for optimization purposes:

$$\mathcal{F}(\hat{R}_a) \cong \sum_{e_i\in E}\ln \mathrm{P}(\hat{r}_{ia}|\tau) + \sum_{s\in\mathcal{S}} q(a_s=a)\sum_{c\in\{\prec,\succ\}} q(c_s=c)\ln \mathrm{P}(c|\hat{r}_{[s]_1a},\hat{r}_{[s]_2a}) = -\sum_{e_i\in E}\frac{\hat{r}^2_{ia}}{2\sigma^2} - \sum_{s\in\mathcal{S}} q(a_s=a)\sum_{c\in\{\prec,\succ\}} q(c_s=c)\ln\left(1+e^{-\upsilon k(c)(\hat{r}_{[s]_1a}-\hat{r}_{[s]_2a})}\right) \tag{4.17}$$

$k(c)$ equals to 1, when $c =\succ$, and to -1 otherwise. The derivative w.r.t. $\hat{r}_{ia}$ for $e_i \in E$ and $a \in A$ is:

$$\mathcal{F}'_{\hat{r}_{ia}}(\hat{R}_a) = -\frac{\hat{r}_{ia}}{\sigma^2} + \sum_{s\in\mathcal{S}} q(a_s=a)\sum_{c\in\{\prec,\succ\}} q(c_s=c)\,\upsilon k(c)\left(\frac{\mathrm{I}_{[i=[s]_1]}}{1+e^{-\upsilon k(c)(\hat{r}_{[s]_2a}-\hat{r}_{ia})}} - \frac{\mathrm{I}_{[i=[s]_2]}}{1+e^{-\upsilon k(c)(\hat{r}_{ia}-\hat{r}_{[s]_1a})}}\right) \tag{4.18}$$

Equations (4.17) and (4.18) may seem similar to the Bradley-Terry-Luce model,

but with significant differences due to the uncertainties for aspect and comparison outcome, as well as a Gaussian prior on rank score.

The time required by each iteration scales linearly with the corpus size, $\mathcal{O}(|\mathcal{S}|)$. The gradient descent step requires $\mathcal{O}(\xi|\mathcal{S}|)$, where ξ is the number of iteration until convergence. ξ depends on the properties of an individual dataset and optimization parameters, in practice the procedure converges fast, and ξ reduces from one iteration to another.

4.3.2 Discrete Model via Gibbs Sampling

Gibbs sampling [52] provides a mechanism to infer hidden variables of a graphical model. It is a special case of Monte Carlo algorithm that defines a Markov chain in the space of possible variable assignments. We sample one variable at a time from the conditional distribution of that variable, conditioned on all the others. The stationary distribution of the Markov chain is the joint distribution over the variables and samples drawn in a such way are guaranteed from the joint distribution. In comparison to variational approximation, Gibbs sampling does not impose any constraint on the form of the distribution to be approximated. When the number of samples is large, we can arrive at a good approximation of the true posterior probability distribution over parameters.

We use the collapsed version of Gibbs sampling, by integrating out continuous

variables $\theta_{\prec a}, \theta_{\succ a}$, θ_{ba}, and π. The derivation is provided below.

$$\mathcal{L}(R, A, C, H|\alpha, \beta, \gamma, \tau) =$$
$$\int_{\pi}\int_{\theta} \mathcal{L}(\pi, \theta, R, A, C, H|\alpha, \beta, \gamma, \tau)\, d\theta\, d\pi =$$
$$\int_{\pi} \prod_{s\in\mathcal{S}} \mathrm{P}(a_s|\pi) \times \mathrm{P}(\pi|\beta)\, d\pi \times$$
$$\prod_{a\in A}\prod_{e_i\in E} \mathrm{P}(r_{ia}|\tau) \times \prod_{e_i,e_j\in E}\prod_{s\in S_{ij}} \mathrm{P}(c_s|r_{ia_s}, r_{ja_s}) \times$$
$$\prod_{s\in\mathcal{S}}\prod_{w\in s} \mathrm{P}(\nu_{sw}|\gamma) \times \int_{\theta} \prod_{a\in A} \mathrm{P}(\theta_{ba}|\alpha)\mathrm{P}(\theta_{\prec a}|\alpha)\mathrm{P}(\theta_{\succ a}|\alpha) \times$$
$$\prod_{a\in A}\prod_{s\in\mathcal{S}}\prod_{w\in s} \Big(\big(\mathrm{P}(w|\theta_{ba})\big)^{\mathrm{I}_{[\nu_{sw}=0]}} \big(\mathrm{P}(w|\theta_{\succ a})\big)^{\mathrm{I}_{[\nu_{sw}=1\wedge c_s=\succ]}}$$
$$\big(\mathrm{P}(w|\theta_{\prec a})\big)^{\mathrm{I}_{[\nu_{sw}=1\wedge c_s=\prec]}} \Big)^{\mathrm{I}_{[a_s=a]}}\, d\theta \tag{4.19}$$

The integral for π is the Dirichlet-multinomial distribution over aspects. If $K = |A|$ and n_a denotes the number of sentences assigned to aspect a, we have:

$$\int_{\pi} \prod_{s\in\mathcal{S}} \mathrm{P}(a_s|\pi) \times \mathrm{P}(\pi|\beta)\, d\pi = \frac{\Gamma(\beta K)}{\Gamma^K(\beta)} \frac{\prod_{a\in A}\Gamma(n_a + \beta)}{\Gamma(\beta K + \sum_{a\in A} n_a)} \tag{4.20}$$

We separately integrate the all θ-expressions out. We can regroup factors as follows:

$$\prod_{a\in A} \int_{\theta_{ba}} \mathrm{P}(\theta_{ba}|\alpha) \prod_{s\in\mathcal{S}}\prod_{w\in s} \big(\mathrm{P}(w|\theta_{ba})\big)^{\mathrm{I}_{[a_s=a\wedge\nu_{sw}=0]}}\, d\theta_{ba} \tag{4.21}$$
$$\int_{\theta_{\prec a}} \mathrm{P}(\theta_{\prec a}|\alpha) \prod_{s\in\mathcal{S}}\prod_{w\in s} \big(\mathrm{P}(w|\theta_{\prec a})\big)^{\mathrm{I}_{[a_s=a\wedge\nu_{sw}=1\wedge c_s=\prec]}}\, d\theta_{\prec a} \tag{4.22}$$
$$\int_{\theta_{\succ a}} \mathrm{P}(\theta_{\succ a}|\alpha) \prod_{s\in\mathcal{S}}\prod_{w\in s} \big(\mathrm{P}(w|\theta_{\succ a})\big)^{\mathrm{I}_{[a_s=a\wedge\nu_{sw}=1\wedge c_s=\succ]}}\, d\theta_{\succ a} \tag{4.23}$$

The integral expressions in (4.21), (4.22), and (4.23) are Dirichlet-multinomial distributions over features. Let $n(w, \nu, a, c)$ be the number of times feature w sampled from the background distribution ($\nu = 0$) or the comparative distributions ($\nu = 1$) occurs in a given corpus within a sentence assigned to aspect a with com-

parison preference c. Assume that the corpus feature vocabulary is V, $M = |V|$ is the vocabulary size. We can rewrite, for instance, the integral in (4.23) for every aspect a as follows:

$$\frac{\Gamma(\alpha M)}{\Gamma^M(\alpha)} \frac{\prod_{w \in V} \Gamma\big(n(w,1,a,\succ) + \alpha\big)}{\Gamma\big(\alpha M + \sum_{w \in V} n(w,1,a,\succ)\big)}. \tag{4.24}$$

Similarly with the integrals in (4.21) and (4.22).

$\mathrm{P}(r_{ia}|\tau)$ and $\mathrm{P}(c_s|r_{ia_s}, r_{ja_s})$ are defined below.

$$\mathrm{P}\big(r_{ia} = r|\tau = (n, p_0)\big) = \binom{n-1}{r} p_0^r (1-p_0)^{(n-1)-r} \tag{4.25}$$

$$\mathrm{P}(c_s = c|r_{ia}, r_{ja}) = \big(1 - \sigma_\upsilon(r_{ia} - r_{ja})\big)^c \big(\sigma_\upsilon(r_{ia} - r_{ja})\big)^{1-c} \tag{4.26}$$

We iteratively sample background indicator ν_{sw} for every feature w in each sentence $s \in \mathcal{S}$, comparison outcome c_s and aspect a_s for each sentence, and rank scores r_{ia} for each entity $e_i \in E$ and aspect $a \in A$.

Sampling Background Indicators H. We want to sample ν_{sw} for each word w in sentence $s \in \mathcal{S}$, keeping the rest variables fixed. Let H_{-sw} be the set of background indicator variables without ν_{sw}, then:

$$\mathcal{L}(\nu_{sw}|H_{-sw}, R, A, C) \propto \mathrm{P}(\nu_{sw}|\gamma) \times \left(\mathrm{I}_{[\nu_{sw}=0]} \frac{\alpha + \bar{n}(w,0,a_s,*)}{\alpha M + \sum_{w \in V} \bar{n}(w,0,a_s,*)} + \mathrm{I}_{[\nu_{sw}=1]} \frac{\alpha + \bar{n}(w,1,a_s,c_s)}{\alpha M + \sum_{w \in V} \bar{n}(w,1,a_s,c_s)} \right). \tag{4.27}$$

We assume $n(w,0,a,*) = n(w,0,a,\prec) + n(w,0,a,\succ)$. $\bar{n}(w,\nu,a,c)$ is defined the same way as $n(w,\nu,a,c)$, but without the count for feature position w in s.

Sampling Aspects A. The re-sampling of the aspect variable of sentence s affects the feature distributions and entity rankings. A_{-s} denotes the set of aspect

variables excluding a_s.

$$\begin{aligned}\mathcal{L}(a_s|A_{-s},R,C,H) \propto (\bar{n}_{a_s}+\beta)\mathrm{P}(c_s|r_{[s]_1a_s},r_{[s]_2a_s})\\ \frac{\prod_{w\in V}\prod_{i=1}^{l_s(w,0)}\left(\bar{n}(w,0,a_s,*)+\alpha+i-1\right)}{\prod_{i=1}^{\sum_{w\in V}l_s(w,0)}\left(\alpha M+\sum_{w\in V}\bar{n}(w,0,a_s,*)+i-1\right)}\\ \frac{\prod_{w\in V}\prod_{i=1}^{l_s(w,1)}\left(\bar{n}(w,1,a_s,c_s)+\alpha+i-1\right)}{\prod_{i=1}^{\sum_{w\in V}l_s(w,1)}\left(\alpha M+\sum_{w\in V}\bar{n}(w,1,a,c_s)+i-1\right)}\end{aligned} \tag{4.28}$$

$l_s(w,\nu)$ returns the count of feature w with background indicator ν in sentence s.

Sampling Comparison Outcomes C. We independently sample comparison outcome c_s for each sentence $s \in S$. C_{-s} is the comparison outcome variable set without c_s

$$\begin{aligned}\mathcal{L}(c_s|C_{-s},R,A,H) \propto \mathrm{P}(c_s|r_{[s]_1a_s},r_{[s]_2a_s})\\ \frac{\prod_{w\in V}\prod_{i=1}^{l_s(w,1)}\left(\bar{n}(w,1,a_s,c_s)+\alpha+i-1\right)}{\prod_{i=1}^{\sum_{w\in V}l_s(w,1)}\left(\alpha M+\sum_{w\in V}\bar{n}(w,1,a_s,c_s)+i-1\right)}\end{aligned} \tag{4.29}$$

Sampling Rank Scores R. We sample rank score r_{ia} for each entity e_i independently from each other rather than simultaneously. This allows us dramatically reduce computational complexity of the algorithm. In comparison to Gibbs sampling, Variational method makes it possible to optimize rank scores simultaneously (see Section 4.3.1). R_{-ia} denotes the rank score variables excluding r_{ia}

$$\mathcal{L}(r_{ia}|R_{-ia},A,C,H) \propto \mathrm{P}(r_{ia}|\tau)\prod_{s\in\mathcal{S}}\Big(\mathrm{P}(c_s|r_{[s]_1a},r_{[s]_2a})\Big)^{\mathrm{I}_{[[s]_1=i\vee[s]_2=i]}} \tag{4.30}$$

Gibbs Sampling with Simulated Annealing. Although Gibbs sampling allows estimating the shape of a probability distribution, one can modify this process to maximize the model likelihood. We used *simulated annealing*, the technique used in optimization to find global optimum of a given (non-convex) function. We sample each variable from the modified distribution:

$$\mathrm{P}(z_j=z|...) \rightarrow \frac{\mathrm{P}(z_j=z|...)^{1/t_j}}{\sum_z \mathrm{P}(z_j=z|...)^{1/t_j}}, \tag{4.31}$$

where the sequence $T = (t_j)_{j=1}^{n}$ defines the cooling schedule and particular value t_j is called the temperature. As $t_j \to 0$ the distribution becomes sharper (setting $t_j = 1$ for every j recovers standard Gibbs sampling procedure) and the modified distribution concentrates all the mass on the maximal outcome.

A single iteration of the Gibbs sampler scales linearly with respect to the corpus size, and takes $\mathcal{O}(|\mathcal{S}|)$ time. Each sampling subroutine requires only one sentence at a time. The rest of the parameters are considered fixed and bounded by some constant. However, the number of aspects and score ranks, when large, can substantially increase the computational time.

4.3.3 Discussion: Unsupervised vs. Supervised

Thus far, we have assumed unsupervised setting. We will explore both unsupervised and supervised settings in Section 4.4. To introduce "light" supervision, we label a subset of sentences in terms of their comparison outcomes and aspect assignments. Where in the unsupervised setting, only the w's are observed, in the supervised setting, we consider some c_s and a_s variables (corresponding to a subset of labeled sentences) to also be observed (having known outcomes). This has the effect of grouping together sentences of the same label, which would then influence the respective feature distributions $\theta_{\succ a}$, $\theta_{\prec a}$, and θ_{ba}.

An alternative is to consider the rank score r_{ia} of some entities to be observed. This is too heavy-handed, and runs counter to learning the crowdsourced ranking based on user-generated content.

4.4 Experiments

Our focus here is on effectiveness, rather than efficiency. All experiments were conducted on a PC with Intel Core i5 CPU 3.3 GHz and 12GB of RAM.

4.4.1 Datasets

The corpus of comparative sentences $\mathcal{S}$ can be obtained from user evaluation of pairs of products. We crawled reviews from the Digital Camera and Cell Phone categories of *Amazon*. The latter was augmented with data constructed by [169], which in turn was based on corpus presented in [86]. We describe a methodology we used for extracting comparative sentences from reviews. For practical purposes, off-the-shelf approaches are available (e.g., [169, 76]). There are four key information to determine: whether a sentence is comparative, the entities being compared, the comparison direction, and the aspect of interest.

Comparative Sentence & Aspect Identification. Our scope covers sentences containing two entities. For Digital Camera, we pick the four most frequent aspects: *functionality*, *form factor*, *image quality*, and *price*. Cell Phone is represented by the general aspect of overall quality, due to the lack of a meaningful volume of comparative sentences for more specialized aspects. We take a random sample of sentences and manually label them for comparative sentence identification and aspect identification, and train the respective classifiers. We apply these classifiers to the remaining sentences, followed by manual inspection to remove false positives to ensure a high quality of the dataset.

Entity Recognition & Linking. There is no ready-made named entity recognition (NER) system for the domain we are considering. Therefore, we employ a dictionary matching approach[2] that works well in tying the mentions of an object together. We construct the dictionary of entities from product titles, which we employ to perform token-based partial matching search. Then, sentences are manually reviewed. This works well for cameras, but not for cell phones due to many generic references (e.g., *it*, *this phone*, etc.). As co-reference resolution is manually time-consuming and difficult to automate, we will use the Cell Phone dataset in a pseudo-synthetic scenario replacing mentions with artificial entity tokens (see Section 4.4.3).

[2]The dictionaries are manually crafted from the Amazon and Epinions product pages corresponding to the related domains.

Domain/Aspect	# sentences	#1 is favored (%)	#2 is favored (%)
DIGITAL CAMERA			
Functionality	457	38.5	61.5
Form Factor	78	61.3	38.7
Image Quality	129	58.1	41.9
Price	165	52.1	47.9
CELL PHONE	544	67.1	32.9

Table 4.4: Dataset Statistics.

Aspect	Specification		Crowdsourced	
	# entities	# pairs	# entities	# pairs
Functionality	65	171	69	87
Form Factor	40	110	21	14
Image Quality	-	-	28	17
Price	34	103	37	27

Table 4.5: Ranking Benchmarks for Digital Camera Dataset.

Table 4.4 shows the dataset properties. The number of products being compared for Digital Camera is 180. The four aspects respectively have 457, 78, 129, and 165 comparative sentences. Cell Phone is represented by 544 sentences. The distributions between the two classes (whether the first-mentioned (#1) or second-mentioned (#2) entity is favored) are relatively well-balanced. These data sizes are significant, in light of the need to carefully annotate the data, not just with labels, but also with ranking benchmarks (see Section 4.4.1). These datasets are also larger than that used in the previous work on entity ranking [91].

Entity Ranking Benchmarks

Because there is no definitive ranking ground truth, we use two benchmarks that together provide a more complete picture. Their sizes are presented in Table 4.5.

Specification Benchmark. The intuition is that users' preferences can be traced to some specific attribute of the entities. We collect product specification information from dpreview.com[3] and wikipedia.com. For *form factor*, we say that entity e_i is better than e_j if both the volume and weight of e_i are smaller than those of

[3] *Digital Photography Review* has a large database with detailed information about individual digital cameras.

e_j. For *functionality*, the entity with the later release date is better, assuming that the newer model is more functional (comparison is only within product lines). To ensure that the functionality has indeed changed, we only consider differences of more than one year. For *price*, we consider the lower price to be better. To be conservative against price fluctuations, we only consider differences of more than 1000USD. There are 291 entity pairs for *functionality*, 5836 pairs for *form factor*, and 1479 pairs for *price*. After pruning the pairs whose ranking cannot be inferred from data, it contains 171, 110, and 103 pairs from *functionality*, *from factor* and *price* respectively. It is not defined for *image quality* and Cell Phone, for a lack of corresponding knowledge base.

Crowdsourced Benchmark. This benchmark is created from the set of labels used for comparative direction classification. For each pair of entities, we consider each sentence to vote based on its label. The entity with the majority votes is considered better. This benchmark reflects how users in general rank these entities, which may not always be consistent with the specifications. There are 175 entity pairs for *functionality*, 53 pairs for *form factor*, 102 pairs for *price*, and 90 pairs for *image quality*.

For an evaluation pair, we refer to the difference between the majority votes and the minority votes as "support". For greater confidence in the evaluation pairs, we only include such pairs with support of at least two. This leave us 87, 14, 17, and 28 evaluation pairs for *functionality*, *form factor*, *image quality*, and *price* respectively. The average support per entity pair is at least 2.5, and goes up to 3.9 for *functionality*, reducing the probability of choosing a comparison direction by chance. This benchmark is smaller than the specification one because it is defined only for pairs that have been explicitly compared within the data. However the variety of entities is comparable to the specification benchmark (see Table 4.5). We did not use transitive extension for the benchmark generation.

Evaluation Tasks and Metrics

We evaluate the performance of *CompareGem* along three dimensions, as follows.

Comparative Direction Classification. All the competing algorithms are given a set of labeled (training) and a set of unlabeled (test) data. Each algorithm identifies the favored entity for each comparative sentence in the test data. One can see this essentially as a binary classification problem. To measure the performance of an algorithm, we calculate its *classification accuracy*, i.e., the fraction of correctly classified sentences (over the total number of sentences in the test set).

Entity Ranking. We also want to assess the quality of ranking scores produced by the competing algorithms. It is not always feasible to have a ground truth in the form of a rank list, because some pairs may not be comparable. We assume that the ground truth (see Section 4.4.1) has the form of a set of entity pairs X, where the favored (higher-ranked) entity for each pair in X is known. We transform the output ranking scores into a set of ordered pairs Y, which we compare in terms of its agreement with the ground truth X.

As metric, *ranking accuracy* is the agreement between the ground truth X and the output Y in terms of the fraction of concordant pairs over all pairs in the intersection, expressed as a percentage. It is closely related to Kendall's tau [43]. Whereas Kendall's tau is defined for the totally ordered sets, the proposed metric accepts partially ordered sets, and, thus is more suitable here, as comparison makes sense only for comparable entities. Two entities are comparable if there is at least one comparative sentence of aspect a involving them. Comparability is also transitive.

Aspect Identification. We first investigate the previous two primary tasks in the scenario where individual aspects are known. As *CompareGem*'s flexibility allows for latent aspects, we then investigate the second scenario where aspects are pooled and latent, and assess the aspect identification using balanced F-measure. This gives a better assessment, as the aspect distribution in the dataset is very skewed, and simple majority vote alone already attains 55% accuracy.

4.4.2 Parameter Setting

CompareGem involves a number of hyperparameters. To tune them, we perform two-step grid search. The first step optimizes the parameters (α, β, γ) when the ranking component is switched off ($\upsilon = 0$). Once these hyperparameters are fixed, the ranking-related parameters (υ, σ, n) are optimized in the second step.

The number of comparative features (e.g., *"more"*, *"better"*) in a sentence is usually fewer than the number of the background words (e.g., emph"tried", *"actually"*, *"'pixel"*). This suggests a reasonable set of possible values for γ, which should lie within $(0.5, 1)$. We use the same non-informative hyperparameters for the feature distributions over all aspects.

For the grid search, the measures (e.g., accuracy, ranking accuracy) were combined into the harmonic mean, $H(M) = k/\left(\sum_{i=1}^{k} m_i^{-1}\right)$, where $M = \{m_i\}_{i=1}^{k}$ are the appropriate evaluation measures.

The Gibbs sampling algorithm uses simulated annealing and requires specification of initial temperature and cooling schedule. We analyzed the exponential cooling and linear cooling schedules. The linear schedule managed to produce better result for the same fixed number of iteration and was adopted. The number of optimization steps were set to 250 for Variational Approximation and 500 for Gibbs sampling. Increases did not show substantial improvement.

4.4.3 Supervised Evaluation

The aim is to understand how well *CompareGem* tackles the classification and ranking tasks in the presence of training data. We lemmatize words before turning them into features. Rare features are discarded. We show results for the 50:50 training and test data splits. Similar results are observed on the 40:60 and 60:40 data splits, but are not discussed here due to space consideration. We repeat every run 10 times on different data shuffles, and report the averages. We conduct randomization test [12] at 5% statistical significance level for the differences between methods. The

best result among methods is in bold. * indicates the presence of a significant difference between the best and second best methods. Lower results with statistically insignificant differences are shown in italics.

Evaluation on Individual Aspects

First, in this section, we focus on the evaluation of the two primary tasks of comparative direction classification and entity ranking to study their synergy. Therefore, we fully supervise the aspect assignment, and run these experiments on each aspect separately.

Methods. At the sentence level, the aim is to determine which entity being mentioned is better. For this classification task, we compare to two popular classifiers: Support Vector Machine (*SVM*) [30] and Naive Bayes (*NB*) [115] as implemented in Weka [59]. We used SVM with linear kernel, and tuned the regularization parameter C via grid search, $C \in \{10^{-1}, 10^{0}, ..., 10^{3}\}$. For the ranking task, our baseline is Bradley-Terry-Luce model (*BTL*). Because *BTL* assumes the comparison outcomes of sentences are known, we use the classification output from the first task, together with the training sentences as inputs to the ranking model. For this reason, BTL is not a complete baseline, because it cannot operate independently from a source of comparative directions. For ranking, we create a composite baseline from pipelining the two steps discussed in this section (i.e., *SVM+BTL*).

In contrast to the baseline, as *CompareGem* is a generative model, we simply learn the two tasks simultaneously. There are two versions of *CompareGem*: with *Continuous* rank scores learnt via Variational Method, and *Discrete* rank scores learnt via Gibbs Sampling.

Comparative Direction Classification. We first validate the hypothesis that joint modeling improves the comparative direction classification. Table 4.6 shows results for the different configurations of *CompareGem*, continuous and discrete, with or without modeling the entity ranking. The ranking component is put out by setting the sigmoid scaling parameter to zero ($\upsilon = 0$) in (4.2). The versions of

Aspect	Continuous		Discrete	
	With Ranking	Without Ranking	With Ranking	Without Ranking
Functionality	**85.0***	74.2	83.1	65.5
Form Factor	**74.5***	62.5	70.0	55.5
Image Quality	**76.7**	*76.3*	62.5	61.1
Price	*68.6*	60.5	**68.9**	57.1
Overall	**75.7**	67.7	70.3	59.6

Table 4.6: Supervised: CompareGem Comparative Direction Classification.

Aspect	CompareGem (Continuous)	SVM	NB
Functionality	**85.0***	79.3	74.7
Form Factor	**74.5***	66.8	62.5
Image Quality	**76.7***	71.6	69.8
Price	**68.6***	60.9	60.1
Overall	**75.7**	69.0	66.2

Table 4.7: Supervised: Comparative Direction Classification

CompareGem that take advantage of the entity ranking information perform better than their non-ranking counterparts. The Overall row[4], derived as the harmonic mean across the four aspects, indicates that *CompareGem* (Continuous) performs substantially better that its competitors, thus we use it latter to compare with the baseline methods.

Table 4.7 reports the accuracy results of the baseline methods. For all four aspects, the best performing method is Continuous *CompareGem*. The baselines, SVM and NB, perform worse (statistically significant).

This outperformance validates our hypothesis that jointly modeling ranking and classification helps the model do better at classifying sentences.

Between the two baselines, SVM is noticeably better than NB. It also has better results than non-ranking versions of *CompareGem* (Table 4.6). We keep SVM as the primary baseline in subsequent experiments.

Entity Ranking. For ranking, we rely on two benchmarks. Table 4.8 shows the ranking accuracies for the crowdsourced benchmark. *CompareGem* (Continuous)

[4]Overall, as harmonic mean, does not lend itself to randomization test for significance, and thus does not admit the * indicator.

Aspect	CompareGem (Continuous)	CompareGem (Discrete)	SVM+BTL
Functionality	**94.9***	89.7	93.8
Form Factor	**94.3**	*92.9*	90.7
Image Quality	**94.1***	90.0	89.4
Price	**89.6**	*88.1*	86.7
Overall	**93.1**	90.1	90.0

Table 4.8: Supervised: Entity Ranking (Crowdsourced).

Aspect	CompareGem (Continuous)	CompareGem (Discrete)	SVM+BTL
Functionality	75.8	76.6	**80.1***
Form Factor	**58.7***	53.4	51.0
Price	**75.8**	*75.4*	70.0
Overall	**69.0**	66.6	64.6

Table 4.9: Supervised: Entity Ranking (Specification).

has the highest ranking accuracies. Though *CompareGem* outperforms *SVM+BTL* significantly, the magnitude of the difference is less impressive than for classification task. We hypothesize that ranking is an "easier" task than classification. Though *SVM* performs significantly worse in classification at the sentence level (Table 4.7), at the level of entity pairs, there could still be sufficient number of correctly classified sentences to get the ranking right.

Table 4.9 shows the ranking accuracies for the specification benchmark. Against this benchmark, *CompareGem* still performs well for *form factor* and *price*. For *functionality*, it is worse than *SVM+BTL*.

Though the absolute numbers are different, the main conclusions that can be derived from the two benchmarks are similar. Indeed, the evaluation pairs that exist in both benchmarks are quite consistent. Only one disagreement is indicated within *functionality* between them. The difference in the results can be explained in part by the difference in benchmark sizes (see Table 4.5). The specification benchmark imposes more constraints on the entity placement within a ranking, making it more 'difficult' for the methods.

For these datasets, between the two versions of *CompareGem*, Continuous is

noticeably better across Tables 4.6 to 4.9. Subsequently, we report the results of *CompareGem* (Continuous) as a representative.

Evaluation on Combined Aspects

In this section, we pull all aspects together, and explore the scenario when only partial supervision on aspect assignment is available. The aim here is to understand how well *CompareGem* tackles the major tasks of comparative direction classification and entity ranking, while also pursuing aspect identification.

The natural baseline is to use a pipeline of classifiers and ranking model. The first classifier is trained to identify the aspect of a sentence. The second one is to identify the comparative direction. The third model is to build entity ranking for every aspect based on the classification outcomes. We report the results for a *SVM+BTL* pipeline, i.e., *SVM* classifiers for aspect and comparative direction and *BTL* model for ranking.

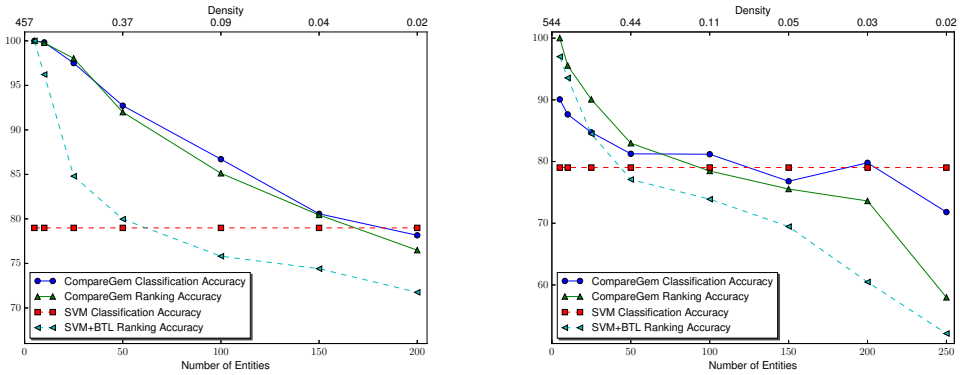
Table 4.10 summarizes the evaluation results of *CompareGem* (Continuous) vs. the *SVM+BTL* pipeline on the Digital Camera dataset. Because we are dealing with a single pool, we show "overall" figures derived as harmonic mean across results for individual aspects. For ranking, we use the crowdsourced ranking benchmark, which is applicable to all four aspects. Evidently, *CompareGem* still outperforms the baseline on the two primary tasks of comparative direction classification and entity ranking. This is despite a marginally lower performance in aspect identification, which is still sufficiently accurate to support the primary tasks. This speaks of the flexibility of *CompareGem*, in its higher capacity for comparative direction classification and entity ranking, in both scenarios of operating with known aspects or with partial information on aspects.

Effect of Sentence Density

To better understand *CompareGem*, we conduct further analysis to investigate the effect of the density of entities in a corpus. The density here is defined as the ex-

Measure	CompareGem (Continuous)	SVM+BTL
Comparative Direction Classification	**68.2**	67.4
Entity Ranking	**91.4**	89.8
Aspect Identification	68.4	**69.0**

Table 4.10: Supervised: Combined Aspects.



(a) Digital Camera (b) Cell Phone

Figure 4.3: Classification and Ranking Accuracies Against Density.

pected number of sentences per comparable pair of entities. The working hypothesis is that *CompareGem* produces better quality output for denser corpora.

Data. This experiment requires fitting the model on several corpora of similar contents, but with varying density. It is infeasible to find such corpora naturally, and therefore we rely on a pseudo-synthetic scenario. Starting with an original corpus, suppose that we need to come up with another corpus of comparative sentences involving a desired number of n entities. Given a corpus of comparative sentences $\mathcal{S}$, we replace the original entity tokens within these sentences with artificial entity tokens from a predefined set $E^n = \{e_i\}_{i=1}^n$. As synthetic ground truth, we set the entity rank scores beforehand. For each sentence in $\mathcal{S}$, we randomly pick two entities form E^n and place them in the sentence in the order consistent with the comparative outcome based on the rank scores.

The density of a corpus $\mathcal{S}$ is defined as:

$$\text{density}(\mathcal{S}) = |\mathcal{S}| \binom{n}{2}^{-1}. \tag{4.32}$$

Figure 4.3 tracks how the performance of *CompareGem* and the baselines on comparative direction classification and entity ranking (y-axis) varies with the density (upper x-axis) or the number of entities (lower x-axis) in the corpus. For the pseudo-synthetic datasets, we use two original corpora, namely: the *functionality* aspect of the Digital Camera dataset and the Cell Phone dataset. For both datasets, the results are consistent.

The higher the density, the higher is the ranking accuracy performance of both *CompareGem* and the *SVM+BTL* baseline. This is expected as the more comparative sentences we have for each a pair of entities, the easier and the more robust it is to determine the ranking of entities. Importantly, though the trend is similar, *CompareGem*'s ranking accuracy (green line) is consistently higher than that of *SVM+BTL* (cyan line).

Interestingly, increasing ranking accuracy with density lifts *CompareGem*'s performance in comparative direction classification (navy blue line) correspondingly. In contrast, the baseline *SVM* has a flat accuracy (red line), unaffected by the ranking accuracy due to its pipeline design. Even at low densities (around 0.02 onwards for Digital Camera, and 0.5 for Cell Phone), *CompareGem* outperforms *SVM*.

Notably, the ranking accuracy of *CompareGem* is always better than the ranking accuracy of the baseline even if the classification accuracy drops. This may be the effect of model regularization and the probabilistic nature of the comparison outcomes. Uncertain predictions should not affect the ranking of the corresponding entities much, while the pipeline setting does not deal with these situations. That joint modeling outperforms the pipeline setting validates our original motivation for developing a joint model to solve both problems simultaneously.

4.4.4 Unsupervised Evaluation

CompareGem can also run in unsupervised setting, when no labeled data is used as input. The goal is to explore how suitable *CompareGem* is for modeling a corpus of comparative sentences. If a model can find a good fit to a dataset even without any

Measure	CompareGem (Continuous)	NB-EM+BTL	Majority
Purity	**62.4***	52.9	53.6
Ranking Accuracy	**64.7***	57.0	47.7
Overall	**63.5**	54.8	50.4

Table 4.11: Unsupervised: Purity and Ranking Accuracy.

labels, arguably it encodes some essential parts of the corpus. In this experiment, we observe the aspect labels, and the task becomes a binary clustering problem, i.e., finding comparison clusters for each aspect. The entire corpus is observable, not only individual aspects.

As a baseline clustering, we use Naive Bayes with an Expectation Maximization algorithm (*NB-EM*) [131]. This choice is supported by the argument that *CompareGem* reduces to a variation of Naive Bayes when the ranking component is dropped. The baseline clusters comparison outcomes for each aspect separately. We also include a *Majority* baseline, which simply puts all the available sentences into one cluster.

We can still use the labels to evaluate this clustering. We measure the comparison outcome identification quality via *purity*. To compute purity, each cluster is assigned to a class that is most represented within the cluster. Once the clusters are mapped to the labels, we measure the accuracy of the assignment. A low quality clustering has low purity, while a perfect clustering has the purity of 1. The ranking accuracy is calculated for each possible mapping of the obtained clusters, and the maximum value is reported.

Table 4.11 shows the results of this experiment. Comparing the results of supervised vs. unsupervised configurations, we see that the unsupervised results are indeed lower, as expected. Interestingly, the comparison outcome clustering purity is still relatively good and significantly better that the baselines. This supports our intuition that the *CompareGem* captures properties of comparative sentence corpora.

$\theta_{\succ}$: #1 is favored	$\theta_{\prec}$: #2 is favored	θ_b: background
#1 from #2	from #1 #2	#1 model #2
#1 improvement #2	#1 #2 improvement	amateur #1 #2
recommend #1 #2	#1 recommend #2	nikon #1 #2
#1 much #2	much #1 #2	hobbyist #1 #2
pleased #1 #2	#1 #2 blow	old #1 #2

Table 4.12: Supervised: Top Features in Functionality.

4.4.5 Feature Analysis

To gain further insight into the workings of *CompareGem*, here we investigate the features that play an important role in the supervised settings. Since there are two comparison outcomes ($c =\succ$ indicating the first-mentioned entity #1 in a sentence is favored, as well as $c =\prec$ indicating the second-mentioned entity #2 is favored), we focus on features that are most discriminative between the two classes. In the same way, we report the top discriminative features among the aspects. A *discriminative* feature w is one that yields top conditional probabilities $\mathrm{P}(c|w)$.

The feature distributions of *CompareGem* are somewhat related to "topics" in topic modeling [11]. In our case, the "topics" correspond to the comparison outcomes (and not an arbitrary number of topics), the distribution is over features (and not over words), and the primary mechanism for learning is the comparison model in addition to feature co-occurrences (and not word co-occurrences alone as in topic modeling).

In Tables 4.12 and 4.13, we show the top features satisfying the discriminative condition for *functionality* and *image quality* respectively. The first two columns report features relate to comparison outcomes, while the last column shows aspect-related features. For each feature, #1 and #2 refer to the relative positions of the first- and second-mentioned entities, with respect to a word. The relative word positions are important in discriminating comparison outcome. For *functionality*, the features *"#1 improvement #2"* and *"#1 #2 improvement"* relate to the different classes. The background features can give a clue about the aspect. For example, *image quality* emphasizes background words such as *"image"*, *"pic"*, *"capture"*.

$\theta_{\succ}$: #1 is favored	$\theta_{\prec}$: #2 is favored	θ_b: background
#1 give #2	#1 #2 give	#1 image #2
#1 sharper #2	#1 #2 accurate	#1 pic #2
#1 significantly #2	#1 #2 photo	#1 capture #2
#1 detail #2	#1 #2 perform	shooting #1 #2
#1 upgrade #2	#1 #2 upgrade	noise #1 #2

Table 4.13: Supervised: Top Features in Image Quality.

4.5 Discussion

Since more and more users have become accustomed to consulting online reviews, some merchants driven by greater commercial benefits write fake reviews to promote their products. The reviews and opinions deliberately created to mislead consumers are known as deceptive opinion spam [58]. The deceptive spam can be used to undeservingly promote or demote target entities, which, in turn, may mislead CompareGem, if spam reviews contain fake comparisons. The investigation in this chapter is done under assumption and holds when we work with clean data, free of deceptive opinion spam. To satisfy this assumption, one may filter spam from reviews, using methods and models proposed in the related studies [78, 155, 129].

CompareGem shares similar sigmoid-based probability as Bradley-Terry-Luce (BTL) model, which is a form of aggregation model [153, 40, 63]. For the detailed discussion of aggregation models, see Section 2.2.1. In the problem settings, the comparison outcomes being aggregated are latent and unknown and need to be learned from text. Due to synergy between aggregation modeling and generative modeling of text, CompareGem achieves better performance than its pipeline competitors.

Being empirically successful, however, at the entity pair level, our proposed model assumes a homogeneous expression of preferences. Users may express different preferences depending on their requirements, consider a laptop purchase: a gamer would look for a high performing laptop, whereas a businessman would look for something more ergonomic and light at the expense of computing power. This

difference in preferences has to be captured to provide more realistic analysis of user preferences. In Chapter 5, we explore the problem of preference mining in details.

CHAPTER 5

DISCOVERING PREFERENCE GROUPS

Learning to rank is a machine learning approach to rank objects based on their features [20]. It has found applications in many areas. In information retrieval [105], we would like to know which search result is more relevant to a query, and thus should be ranked higher. In recommender system [160, 109], it is important to determine which item is preferred by a user, and thus should be recommended to the user. Several natural language processing tasks may also involve ranking, such as text summarization [120] or keyphrase extraction [75].

The key idea behind learning to rank is to learn a ranking function that maps feature vectors to rank scores or rank orders. This function is learned from data consisting of rankings or ranked list of objects. An implicit assumption in many scenarios is that these rankings come from a homogeneous population. In other words, there is one way to rank objects based on their features, which is represented by a central ranking function. This assumption of a central ranking function may very well be applicable to some scenarios, such as homepage finding or named page finding, where most users would practically agree on the rankings.

Problem Yet there are other scenarios where there may be more than one way to rank objects based on their features. In this chapter, we consider the problem of modeling rankings for a heterogeneous population. In such a population, there may be several sub-populations that rank objects differently. We call such sub-populations "preference groups". For instance, when shopping for cameras, con-

sumers may have diverse preferences with respect to the attributes of a camera, and therefore varied ways for ranking cameras. Professionals may rank DSLRs highly for its customizability, while casual users may prefer point-and-shoot cameras for its portability. In a voting electorate [54, 53], there may be several preference groups that rank electoral candidates differently based on where they stand on issues. Thus a single ranking function would not be able to represent diverse preference groups in a heterogeneous population.

If only these preference groups were identifiable or known in advance, then the problem would devolve into employing learning to rank separately within each preference group independently. On the contrary, in many cases we merely observe the diverse rankings within a population. Discovering the preference groups is inherently part of the problem.

The problem can thus be informally stated as follows. Given a set of objects and their feature vectors, as well as a set of ranked lists defined over these objects, we seek to learn K latent preference groups and correspondingly K ranking functions, one for each preference group. The population of ranked lists is heterogeneous, i.e., there may be different permutations of the same set of objects in the data.

Approach One way to think about the problem is to consider it as an amalgamation of two requisite components: discovering the preference groups, and employing learning to rank within each group.

To discover the preference groups, it is not appropriate to employ clustering in the feature space. The reason is that heterogeneity in our context concerns the variance in rankings over objects with similar features. Therefore, it is more relevant to consider clustering in the ranking space. For this, we turn to mixture models for ranking distributions.

While there are several models for estimating the distribution over rankings [111], as reviewed in Section 5.5, we build on the Plackett-Luce model [141, 106], which is widely applicable and lends itself to maximum likelihood estimation. It is based on Luce's Choice Axiom [107], which states that the probability of choosing

one item over another is not affected by the presence or absence of other items in the pool. This axiom is frequently cited in economics for modeling consumer behavior when choosing one product over another [9]. Plackett-Luce model is characterized by a set of item-specific parameters, as described in Section 5.2. In this case, each preference group is associated with a set of Plackett-Luce parameters. The K preference groups could therefore be modeled as a mixture of K Plackett-Luce models [54, 53].

One limitation of a ranking model such as Plackett-Luce is that it is defined over a finite set of objects. Therefore, it does not generalize well to items not seen, or rarely seen, in the training data. This is where the learning to rank component comes in. Instead of learning item-specific parameters in each preference group (defined over items), we seek to learn a ranking function defined over features. There are at least two advantages to this modeling. For one, we would obtain better generalization from greater applicability to unseen items with similar features. For another, we would obtain better interpretability, as it may allow inspection of which features are important to each preference group.

While it is possible to think of the two components identified above as a pipeline, and we will explore this as well in the experiments, it is much more natural to consider them as two inherent components of a *unified joint model*. For one thing, the two components are mutually beneficial. Better clustering leads to better ranking functions due to more accurate reflection of preferences. Meanwhile, better ranking functions lead to better clustering, allowing better alignment of each ranked list to the closest preference group. Moreover, in a joint model, there is no need for two sets of parameters, one for clustering and another for learning to rank, and the parameters can be unified.

This chapter is structured as follows. We define the problem of heterogeneous ranking populations in Section 5.1. In Section 5.2 we introduce Plackett-Luce Regression Mixture or PLRM, probabilistic graphical model that discovers latent preference groups and their corresponding ranking functions. In Section 5.3, we de-

scribe its inference algorithm based on Expectation-Maximization. In Section 5.4, through comprehensive experiments on several public datasets with varying heterogeneity, we show the effectiveness of the joint PLRM model vis-á-vis a pipeline model, as well as learning to rank models designed for homogeneous populations. Section 5.5 concludes the chapter with a discussion.

5.1 Problem

We consider a set of M items of the same type. For instance, in the context of consumer choice, these may be products of a given category, e.g., digital cameras. For multimedia retrieval, these may be images to be ranked.

An item i is associated with a feature vector x_i in the D-dimensional space, $x_i \in \mathbb{R}^D$. For instance, cameras may have features such as sensor size, the presence of flash, weight and physical dimensions, etc. For images, the features may be gist descriptors and color histograms [136]. The collection of feature vectors of various entities is denoted $X = \{x_i\}_{i=1}^{M}$. For ease of reference, we list our notations in Table 5.1.

In addition to X, we are also given N ranked lists $R = \{r^{(n)}\}_{n=1}^{N}$, corresponding to N "judges". A judge n may rank a subset of items denoted $\bar{X}_n \subseteq X$. The corresponding ranking induced on $\bar{X}_n$ in the form of a permutation, without ties, is denoted $r^{(n)}$. When item i (with feature vector x_i) is placed in position j among items in $\bar{X}_n$, we have $r_i^{(n)} = j$. Position $j = 1$ is the highest, followed by position 2, etc. Equivalently, we write $r^{(n)}[j] = i$.

We further assume that these judges can be clustered into K preference groups. Each group is relatively homogeneous, whereby the ranking behaviors of judges within a group do not vary too much. In contrast, ranking behaviors across groups are heterogeneous. Two individual judges from different groups are likely to have different rankings $r^{(n)} \neq r^{(n')}$ over the same set of items $\bar{X}_n = \bar{X}_{n'}$. These preference groups are latent, and need to be discovered from the data.

Problem Statement Our problem can thus be stated as follows. Given the feature vectors X and the ranked lists R, as well as an integer K, we seek to identify:

- K latent preference groups among the N judges in R,
- a ranking function within each latent preference group.

5.2 Model

In this section, we discuss the formal definition of the proposed Plackett-Luce Regression Mixture (PLRM) model. The plate representation of PLRM is shown in Figure 5.1.

Overview PLRM is a probabilistic graphical model for representing different latent preference groups within a population of judges. Each judge arranges a given set of items into a ranked list (a permutation) based on the features of the item. In the conventional Plackett-Luce model, the ranking is based on item-specific parameters, which may connote for item quality. In contrast PLRM assumes that the ranking is based on a ranking function on item features.

We further assume that K groups exist within the population, and each group is associated with a ranking function. Each judge's ranking is based on the ranking function of the group it belongs to, while still allowing for some variance among group members. Accounting for this variance is best done through probabilistic modeling.

To generate the observed ranked lists R, we consider N experiments as follows. At each random trial, we ask a new judge to select a group. The group is chosen stochastically with a categorical variable $z_n \in \{1, 2, ..., K\}$ indicating the choice. We then ask the judge to rank a subset of items, defined by their feature vectors $\bar{X}_n$. The judge relies on the group's ranking parameter $w_{z_n} \in \mathbb{R}^d$. This parameter is a vector in D-dimensional space, so that each component of w_{z_n} corresponds to a particular feature of $x_i \in \bar{X}_n$.

To produce the ranking $r^{(n)}$ over items in $\bar{X}_n$, the judge may apply the group

Notation	Description
i	index of an item
M	total number of items
d	index of a feature
D	number of item features
x_i	feature vector of an item i
X	collection of items/feature vectors
n	index of a judge or a ranked list
N	total number of judges
$\bar{X}_n$	subset of items ranked by judge n
$r^{(n)}$	permutation over $\bar{X}_n$ given by judge n
r_i	position of item i in the ranked list r
$r[j]$	index of the item occupying position j in r
R	collection of ranked lists by N judges
K	number of preference groups
k	index of a preference group
w_k	preference vector for group k
W	collection of preference vectors
v_i	ranking parameter for item i, equivalent to $\exp\left(x_i w^T\right)$ for PLRM
V	collection of ranking parameters
π	mixture proportion among preference groups
z_n	group assignment for judge n
Z	collection of group assignments

Table 5.1: Notations.

parameter w_{z_k} via regression over the items in $\bar{X}_n$, i.e., $Y = \bar{X}_n {w_{z_n}}^T$. Relying on exact regression values may be unrealistic, given the likely variance among group members. Therefore, to account for the trial uncertainties and ranking deviation among the group members, the regression values serve as conditional parameters to a ranking probability model, as described in the following.

Ranking Probability Model We first describe a ranking model based on the basic Plackett-Luce, after which we introduce the regression-based Plackett-Luce in PLRM.

Let r be a ranking of M items, i.e., permutation of M indices. Plackett-Luce (PL) model defines a probability distribution over all possible rankings of M items.

It is expressed in terms of item-specific parameters $V = \{v_i\}_{i=1}^{M}$; $v_i \geq 0$.

$$\mathrm{PL}(r|V) = \prod_{j=1}^{M} p_j(r|V), \tag{5.1}$$

where

$$p_j(r|V) = \frac{v_{r[j]}}{v_{r[j]} + v_{r[j+1]} + \cdots + v_{r[M]}} = \frac{v_{r[j]}}{\sum_{l=j}^{M} v_{r[l]}}. \tag{5.2}$$

The probability distribution yields an intuitive interpretation in the form of a ranking procedure. A judge generates a ranked list from the first position to the last position. $p_1(r|V)$ indicates the probability of placing an item $r[1] = i$, parameterized by v_i, in the first place. Having selected the item to occupy the first position, we repeat this procedure with the subsequent positions. $p_2(r|V)$ is the probability of placing another element $r[2] = i'$, parameterized by $v_{i'}$ in the second place, and so on. This procedure continues for all the items within a ranked list r. The joint probability of this process for a ranked list r is presented in Eq. 5.1.

The PL model defined above has a couple of important properties. The first one is the intuitive property that an item i is more likely to be placed higher than another item i', if $v_i > v_{i'}$. The second property flows from the afore-mentioned Luce's Choice Axiom. Items that have already been placed into r would not influence the choice probability of the remaining items. This property, also known as "independence from irrelevant alternatives" [107], allows ranked lists of varying sizes to be induced for subsets of items.

One limitation of the conventional PL model, in the context of learning to rank, is the reliance on the item-specific parameter v_i. This requires all items not just to have been seen, but also to have had sufficient representation in the training data. To address this limitation, we therefore seek to bring the ranking parameter into the feature space of items. This is accomplished by expressing the parameter v_i in terms of a regression of the feature vector x_i with weight parameter or "preference vector" w, as expressed in Eq. 5.3. In this work, we use the exponential transformation to satisfy non-negativity constraint. In practice, there could be other possible choices

such as sigmoid.

$$v_i = \exp\left(x_i w^T\right) \tag{5.3}$$

We call this approach *Plackett-Luce Regression* or PLR. However, because of the heterogeneity of the population, there may not be only a single preference vector w for all ranked lists. Instead, we postulate that there are K sub-populations, or preference groups, each of which is associated with its own preference vector. This gives rise to a mixture of PLR models, which we term the *Plackett-Luce Regression Mixture* or PLRM, as described in the following.

Generative Process The PLRM model can effectively be described by the following generative process.

1. π, a K-dimensional mixture proportion, is sampled from Dirichlet distribution with symmetric prior α:

$$\pi \sim Dirichlet(\alpha)$$

2. For each of the K preference groups, its preference vector w_k is sampled from a D-dimensional Gaussian with zero mean and σ^2 variance:

$$w_k \sim \mathcal{N}(0, \sigma^2)$$

3. For each ranked list $r^{(n)}$ defined over the subset of items $\bar{X}_n \subset X$, where $n = 1, \ldots, N$:

   (a) Select a preference group z_n from a choice of K groups according to the mixture proportion π:

$$z_n \sim Categorical(\pi)$$

   (b) Sample a ranking $r^{(n)}$ from the Plackett-Luce model parameterized by

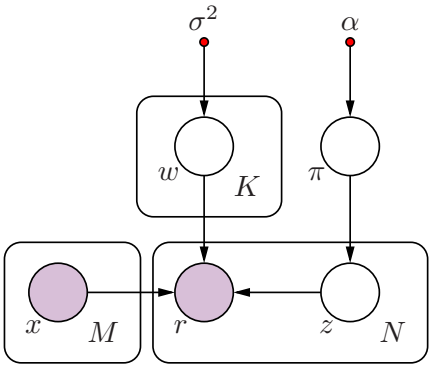


Figure 5.1: Plackett-Luce Regression Mixture Model in Plate Notation.

the regressed values over the set of feature vectors in $\bar{X}_n$:

$$r^{(n)} \sim \mathrm{PL}\left(\exp\left(\bar{X}_n {w_{z_k}}^T\right)\right)$$

The likelihood of this generative process is as follows:

$$\mathcal{L}(R, Z, W, \pi | X) = \mathrm{P}(\pi|\alpha) \times \prod_{k=1}^{K} \mathrm{P}(w_k|0, \sigma^2) \times \prod_{n=1}^{N} \mathrm{P}(z_n|\pi) \mathrm{PL}\left(r^{(n)} | \exp\left(\bar{X}_n {w_{z_n}}^T\right)\right), \tag{5.4}$$

where $R = \{r^{(n)}\}_{n=1}^{N}$ are the set of ranked lists, $Z = \{z_i\}_{n=1}^{N}$ are the corresponding group assignments for each ranked list, and $W = \{w_k\}_{k=1}^{K}$ are the groups' preference vectors.

Discussion The above generative process defines the probabilistic generative model that we call PLRM, with a mixture modeling component representing the latent preference groups as well as a regression component representing the learning to rank based on features. This represents the *joint* modeling approach. With appropriate settings, we can decouple the two components, yielding simpler models.

First, we can turn the model into a purely clustering model based on rankings, without features. In this case, an item i is represented by a feature vector x_i, whose dimensionality is the same as the number of items M. Rather than representing

features, x_i becomes a one-hot "identity" vector, with a value of 1 in the i-th dimension, and 0 in all other dimensions. Effectively, the regression $x_i w^T$ yields an item-specific ranking parameter, just as in the original PL model. Given that this results in a mixture of K Plackett-Luce models, we call this *Plackett-Luce Mixture* or PLM, which is capable of clustering but not ranking by features. Later, we will compare PLRM to PLM, to verify that regression on the features does help the clustering function.

Second, we can turn the model into a purely learning to rank model, by simply setting $K = 1$. In this case, there is no mixture. There is only a single regression model, embedded within a probabilistic ranking model. We thus call this *Plackett-Luce Regression* or PLR, which is capable of learning to rank, but not clustering. Later, we will compare PLRM to PLR, to verify that modeling a mixture does help for a heterogeneous ranking population.

Third, the above two simpler models essentially decouple the two components that are joined together by PLRM. Therefore, they could be employed in a disjoint pipeline. This pipeline of PLM+PLR would first cluster the ranked lists in the population R into K preference groups using PLM, without the help of features. Thereafter, we run PLR within each preference group to learn a ranking function based on features. Later, we will compare PLRM to PLM+PLR to see how the joint approach compares in the effectiveness of both the clustering and ranking objectives.

5.3 Inference

In this section, we derive an Expectation-Maximization (EM) algorithm for fitting the Plackett-Luce Regression Mixture (PLRM) model parameters, as well as discuss how the model could be used for ranking prediction.

5.3.1 Optimization

EM is an iterative algorithm that is commonly used for finding maximum likelihood estimate of a model involving unobserved parameters. In the case of PLRM, we consider the group assignments $Z = \{z_n\}_{n=1}^{N}$ as latent variables that guide the estimation procedure. The groups' preference vectors $W = \{w_k\}_{k=1}^{K}$, as well as the mixture proportion π, are unknown parameters to be maximized during the maximization step. The initial estimates are chosen randomly.

Expectation Step In the expectation step, we estimate the latent variables (Z), and calculate the expected value of the log likelihood function with respect to their a posteriori distribution. We denote the expected value of the log likelihood as follows:

$$Q(W, \pi|W', \pi') = E_{Z|R,W',\pi',X}\left[\log \mathcal{L}(R, Z, W, \pi|X)\right], \tag{5.5}$$

where W' and π' are the current parameter estimates.

Let T_{nk} be an auxiliary function defined as follows:

$$T_{nk} = \frac{\mathrm{P}(z_n = k|\pi')\mathrm{PL}\left(r^{(n)}|\exp\left(\bar{X}_n {w'_k}^T\right)\right)}{\sum_{l=1}^{K}\mathrm{P}(z_n = l|\pi')\mathrm{PL}\left(r^{(n)}|\exp\left(\bar{X}_n {w'_l}^T\right)\right)}. \tag{5.6}$$

Then, we can rewrite Eq. 5.5 into Eq. 5.7 below:

$$Q(W, \pi|W', \pi') = \log \mathrm{P}(\pi|\alpha) + \sum_{k=1}^{K}\log \mathrm{P}(w_k|0, \sigma^2) + \sum_{n=1}^{N}\sum_{k=1}^{K} T_{nk}\left(\log \mathrm{P}(z_n = k|\pi) + \log \mathrm{P}\left(r^{(n)}|\exp\left(\bar{X}_n {w_k}^T\right)\right)\right) \tag{5.7}$$

Maximization Step Eq. 5.7 is used to maximize the model parameters π and W.

Updating π: An update step can be written for π:

$$\pi_k = \frac{1}{\lambda}\left(\sum_{n=1}^{N} T_{nk} + \alpha - 1\right), \tag{5.8}$$

$$\text{where } \lambda = \sum_{n=1}^{N}\left(\sum_{k=1}^{K} T_{nk} + \alpha - 1\right) \tag{5.9}$$

To make sure that every π_k is positive, we accept only $\alpha > 1$, so that $\alpha - 1 = \beta > 0$ serves as a smoothing pseudo-count for each group.

Updating W: The update for groups' preference vectors w_k can be done via iterative optimization, using, for example, BFGS (Broyden-Fletcher-Goldfarb-Shanno) algorithm [103]. The function to be optimized for every $k \in \{1, 2, ..., K\}$, $w \equiv w_k$ is:

$$F(w) = -\frac{ww^T}{2\sigma^2} + \sum_{n=1}^{N}\sum_{k=1}^{K} T_{nk} \log \mathrm{PL}\left(r^{(n)} | \exp\left(\bar{X}_n w^T\right)\right) \tag{5.10}$$

The derivative for the d-th element $w[d]$ of the vector w can be computed as follows:

$$\frac{dF(w)}{dw[d]} = -\frac{w[d]}{\sigma^2} + \sum_{n=1}^{N}\sum_{k=1}^{K} T_{nk} \sum_{i=1}^{|\bar{X}_n|}\left(x_i[d] - \frac{\sum_{l=i}^{|\bar{X}_n|} x_l[d] e^{x_l w^T}}{\sum_{l=i}^{|\bar{X}_n|} e^{x_l w^T}}\right) \tag{5.11}$$

5.3.2 Prediction

Once the model parameters are learned, we can use the model for predictions. Here, we discuss two prediction tasks.

Group Assignment For the first prediction task, given a ranked list, predict the latent preference group that this ranked list belongs to. This task allows us to align a new ranked list to one of the learnt preference groups. To address this task, we pick the $z \in \{1, 2, ..., K\}$ that maximizes the a posteriori distribution of this assignment. Let $\bar{X}$ be an items set, and r its ranking. Given the trained model parameters, we

want to maximize the following probability:

$$\begin{aligned} &\mathrm{P}(z|r,\pi,W,\bar{X}) \propto \mathrm{P}(z,r,\pi,W,\bar{X}) \\ &= \mathrm{P}(\pi|\alpha)\prod_{k=1}^{K}\mathrm{P}(w_k|0,\sigma^2) \times Prob(z|\pi)\mathrm{PL}\left(r|\exp\left(\bar{X}{w_z}^T\right)\right) \\ &\propto \mathrm{P}(z|\pi)\mathrm{PL}\left(r|\exp\left(\bar{X}{w_z}^T\right)\right) \end{aligned} \tag{5.12}$$

Ranking Prediction For the second prediction task, given a set of items, where a ranking for some subset of the items is known, predict the ranks of the other items. This allows us to extend the rankings to other items beyond the known ranking. To address this task, we first predict the group assignment to which the set of items belongs, based on the known subset ranking (as in the first task). Once the group assignment z^* is identified, the remaining items of $\tilde{X}$ whose rankings are not yet known are arranged into a ranked list, using the group's Plackett-Luce Regression parameter, i.e., $Y = \tilde{X}{w_{z^*}}^T$. Taking into account the Plackett-Luce model properties, greater values yield higher rank positions.

5.4 Experiments

The objectives of the following experiments are two-fold. First, as PLRM both discovers the latent preference groups, as well as learns a ranking function for each group, we would like to investigate the relationship between these two objectives, particularly comparing the joint modeling approach vs. the disjoint pipeline approach. Second, since PLRM is designed for a heterogeneous ranking population, we would like to verify its applicability, particularly when compared to a baseline that assumes a homogeneous ranking population.

5.4.1 Datasets

We describe four datasets used in the experiments. The first two: *PubFig* and *OSR* will be our main datasets that appear in all experiments, because they have known

cluster labels, which are necessary as ground truth for validating the accuracy of identifying the preference groups. In addition, we include another two datasets: *Comp* and *DCam*, with rankings but without known cluster labels, which we would use only in the second half of the experiments to evaluate ranking accuracies for heterogeneous populations.

Public Figures (PubFig). This dataset[1], described by [136], consists of 772 facial images (items) of 8 public figures ($\sim$ 100 images per person). The 8 public figures are ranked with respect to 11 physical attributes (e.g., masculine-looking, pointy nose, big lips), as listed in Table 5.2. Each public figure is identified by a letter[2]. Expression $A \prec B$ means that item A precedes item B in the permutation. Some items share the same rank position. The third column shows the permutation lengths possible for each attribute.

These 11 attributes are considered the ground truth preference groups, because each induces a different ranking over the 8 identities. For experiments, we construct 300 ranked lists for each attribute, for a total of 3300 ranked lists. Each list is constructed by sampling an image for each identity. The feature vector of each image is a concatenation of 512-dimensional gist descriptor and a 45-dimensional Lab color histogram. The resulting collection of ranked lists and their feature vectors (but without the ground truth labels) are pooled together. For learning, we create ten random splits, such that 90% of the ranked lists for each attribute are used for training vs. 10% for testing, and average the accuracies.

Outdoor Scene Recognition (OSR) This dataset 1, described by [136], contains 2688 scenes with different spatial envelopes from 8 categories[3]. A scene (item) is represented by its 512-dimensional gist descriptor (feature vector). The categories are organized into rankings with respect to 6 attributes (e.g., natural, open, perspective), as shown in Table 5.2. These attributes are considered the ground truth pref-

[1] https://filebox.ece.vt.edu/~parikh/relative.html

[2] The 8 identities in *PubFig* are: Alex Rodriguez (A), Clive Owen (C), Hugh Laurie (H), Jared Leto (J), Miley Cyrus (M), Scarlett Johansson (S), Viggo Mortensen (V) and Zac Efron (Z).

[3] The 8 categories in *OSR* are: coast (C), forest (F), highway (H), inside-city (I), mountain (M), open-country (O), street (S) and tall-building (T).

Attribute	Permutation	Length
PubFig		
Masculine-looking	S≺M≺Z≺V≺J≺A≺H≺C	8
White	A≺C≺H≺Z≺J≺S≺M≺V	8
Young	V≺H≺C≺J≺A≺S≺Z≺M	8
Smiling	J≺V≺H≺{A,C}≺{S,Z}≺M	6
Chubby	V≺J≺H≺C≺Z≺M≺S≺A	8
Visible Forehead	J≺Z≺M≺S≺{A,C,H,V}	5
Bushy Eyebrows	M≺S≺Z≺V≺H≺A≺C≺J	8
Narrow Eyes	M≺J≺S≺A≺H≺C≺V≺Z	8
Pointy Nose	A≺C≺{J,M,V}≺S≺Z≺H	6
Big Lips	H≺J≺V≺Z≺C≺M≺A≺S	8
Round Face	H≺V≺J≺C≺Z≺A≺S≺M	8
OSR		
Natural	T≺{I,S}≺H≺{C,O,M,F}	4
Open	{T,F}≺{I,S}≺M≺{H,C,O}	4
Perspective	O≺C≺{M,F}≺H≺I≺S≺T	7
Large Objects	F≺{O,M}≺{I,S}≺{H,C}≺T	5
Diagonal Plane	F≺{O,M}≺C≺{I,S}≺H≺T	6
Close Depth	C≺M≺O≺{T,I,S,H,F}	4

Table 5.2: Permutations on Attributes [136] (Ground Truth Rankings).

erence groups. As in *PubFig*, we construct 300 ranked lists for each attribute (for a total of 1800 ranked lists), and create ten random splits of 90:10 for training:testing.

Computer Survey (Comp) This marketing-related dataset[4] is in the form of surveys [185]. The subjects were asked to rate 20 personal computers (items) based on their likelihood of purchasing each computer (on a scale from 0 to 10). A computer is described by its feature vector, which indicates intrinsic characteristics of the computer (e.g., amount of RAM, CPU speed) as well as extrinsic features (e.g., hotline service availability, warranty), resulting in a total of 13 binary features. We excluded subjects with missing responses and with fewer than 5 distinct likelihood values; these were 33 out of 201 subjects. Therefore, 168 subjects were used in experiments. We induce a ranked list of computers for each subject based on the likelihood ratings.

Digital Cameras (DCam) The last dataset concerns digital cameras (items).

[4] `https://github.com/probml/pmtkdata/tree/master/conjointAnalysisComputerBuyers`

We collected the specifications of 876 digital cameras from *www.dpreview.com* and formed feature vectors according to their specifications. These include weight, number of pixels, sensor size, body type, resulting in a total of 32 features. These cameras were manually linked to Amazon product pages (*www.amazon.com*). We used the public Amazon dataset[5] described in [113, 112]. Ranked lists among the cameras were induced from ratings given by Amazon reviewers (on a scale from 1 to 5). We retain only reviewers with at least 3 distinct rating values within the linked data, resulting in 880 ranked lists.

5.4.2 Evaluation Tasks and Metrics

In the experiments, we evaluate the methods based on two prediction tasks that we have outlined earlier in Section 5.3.2.

Group Assignment The first task is to assign a ranked list to the correct preference group. This can only be evaluated on *PubFig* and *OSR*, with known preference groups.

To measure the group assignment accuracy, we compare the preference groups arrived at by a model to the ground truth. For evaluation metric, we use the Rand Index (RI), a widely used statistical measure for data clustering. This metric is defined on the space of object pairs. We want to assign two objects (ranked lists) to the same latent preference group, if and only if they belong to the same ground truth grouping. Otherwise, we want to assign them to two different latent preference groups. The former is known as true positive (TP), while the latter is known as true negative (TN). For N objects, the total number of object pairs is $N(N-1)/2$. Therefore, the Rand Index is defined as follows:

$$RI = \frac{2(TP + TN)}{N(N-1)} \tag{5.13}$$

Rand Index or RI ranges from 0 (worst) to 1 (best). We will express them as

[5] http://jmcauley.ucsd.edu/data/amazon/

percentages.

Ranking The second evaluation task is to predict the ranking of items based on their features. To measure the ranking accuracy, we employ Kendall's Tau correlation coefficient. It measures how similar two ranked lists are in terms of the difference between two probabilities, namely: the probability that the observed ranked lists are in the same order versus the probability that they are not.

Given two ranked lists $A = (a_i)_{i=1}^{M}$ and $B = (b_i)_{i=1}^{M}$ in the form of permutations, we say that for $i \neq j$, a pair (a_i, b_i) is concordant with another pair (a_j, b_j) if either both $a_i \succ a_j$ and $b_i \succ b_j$, or both $a_i \prec a_j$ and $b_i \prec b_j$. Otherwise we say that the pairs are discordant. Kendall's Tau is defined as follows:

$$\tau = \frac{\text{\# concordant pairs} - \text{\# discordant pairs}}{\frac{1}{2}M(M-1)}. \tag{5.14}$$

τ can take the values between minus one and plus one. For evaluation purposes, we re-normalize the coefficient so that it yields a value from zero to one, as follows:

$$\tau^* = \frac{\tau + 1}{2} = \frac{\text{\# concordant pairs}}{\frac{1}{2}M(M-1)}. \tag{5.15}$$

We use Kendall's Tau to compare the ranking produced by a method with the ground truth. Thus, higher Kendall's Tau is better. We will express the value in terms of percentages, averaging across the ranked lists in the testing set.

Where perfect rankings are known, Kendall's Tau better reflects how close an output ranking is to the perfect ranking [46]. In our datasets, all rank positions are important, and not just the top positions. For instance, in *PubFig* when ranking facial images based on a certain physical attribute, we wish to get the ranking right across the full length of the list. For that reason, Kendall's Tau is more appropriate than those favoring the top-ranked elements such as DCG.

Method	PubFig			
	{Masculine, Pointy Nose, Round Face}			All
	Random	Exclusive	All-for-One	Random
PLRM	**99.8**	**100.**	**99.2**	**89.4**
PLM	**99.8**	51.1	55.7	76.3
Method	OSR			
	{Natural, Large Objects, Close Depth}			All
	Random	Exclusive	All-for-One	Random
PLRM	**95.1**	**98.7**	**99.6**	**83.4**
PLM	56.1	60.5	55.7	57.9

Table 5.3: Group Assignment (Rand Index).

5.4.3 Compare to Pipeline Approach

Here, we seek to evaluate the efficacy of the PLRM model, which joins together the tasks of discovering the preference groups as well as learning a ranking function for each group. As we look into validating both preference groups and ranking, we can use only *PubFig* and *OSR* in these experiments.

Group Assignment We first explore how well PLRM can recover the ground truth clustering structure within the data (i.e., the attributes in *PubFig* and *OSR*). The most appropriate baseline is Plackett-Luce Mixture or PLM, which is a mixture model based on Plackett-Luce that does not use the feature space representation to generalize elements beyond their identity (see Section 5.2). That way, we can see how PLRM's modeling of regression-based parameters based on features helps in the clustering objective. The number of latent preference groups K in both PLRM and PLM is set to the actual number of attributes in the respective datasets.

The clustering results for PLRM and PLM are shown in Table 5.3. Since the effect of heterogeneous rankings can most clearly be studied when the attributes are really diverse and distinct, we start with an experiment involving three such attributes. For *PubFig*, we use {Masculine-looking, Pointy Nose, Round Face}. For *OSR*, we use {Natural, Large Objects, Close Depth}. In each case, the three attributes are diverse, with the lowest cumulative Kendall's Tau-b statistics (adjusted for ties), indicating stronger disagreement in terms of the permutation among the

three attributes.

Furthermore, for greater insight into results, we consider three different ways of sampling for generating ranked lists.

- In the *Random* experiment, we sample items of each identity at random for each considered attribute. For *PubFig*, both PLRM and PLM do well, achieving close to 99.8% in terms of Rand Index. In this case, the number of samples is enough to learn an appropriate ranking value for each element in the dataset with respect to the attributes (each person in *PubFig* has only about 100 images). However, for *OSR*, PLRM with 95.1% outperforms PLM with 56.1% significantly. Because it relies on identities, but not features, PLM performs worse in the case where there is insufficient ranking information for specific items, such as in *OSR*.

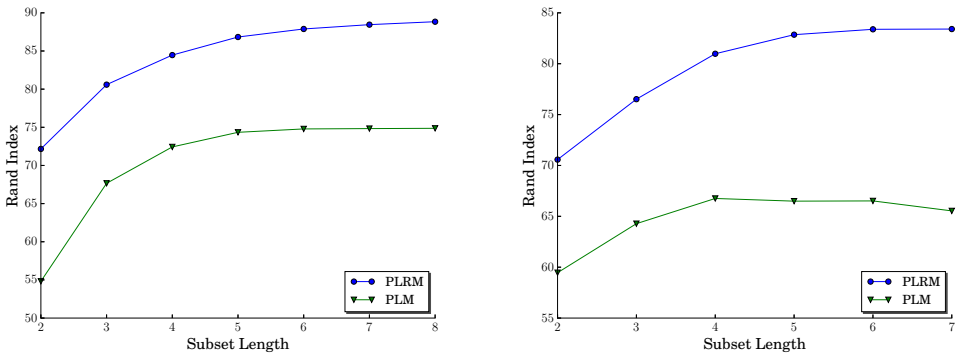
- In the *Exclusive* experiment, we consider three non-overlapping partitions of items, one for each attribute, from which the ranked lists are generated. In this scenario, PLRM could still learn through the feature space, getting 100% for *PubFig* and 98.7% for *OSR*. In contrast, PLM cannot learn how the same items may be ranked differently, and thus gets lower Rand Indices of 51.1% for *PubFig* and 60.5% for *OSR*. This shows the limitation of PLM when an item has not been seen across all the preference groups, which is overcome by PLRM that does not need to see the exact item if other items with similar features have been seen.

- In the *All-for-One* experiment, we first select a subset of items to rank, and then generate the ranked lists for all attributes. Next, we select a different subset of items to rank. Therefore, two ranked lists from the same attribute do not share items. Although we always see all rankings from all attributes, there is not enough information to connect different ranked lists of the same attribute. This showcases the weakness of PLM that requires to have seen cooccurrences of items, whereas PLRM that works through the feature space

can still solve it, attaining 99% accuracies for both datasets, as compared to PLM's 55.7%.

Finally, we consider *all* the attributes (11 for *PubFig* and 6 for *OSR*), and show the results under the *All* columns. Overall, PLRM does a better job in clustering than PLM. For *PubFig*, PLRM's 89.4% on *PubFig* and 83.4% on *OSR* are better than PLM's results (bold indicates best results). PLM is unable to generalize from item id, while PLRM seeks ranked lists that are consistent with the ranking function.

Predicting Group Assignment with Subset Length In the previous experiments, we have assumed that we have ranked lists of sufficient length, and seek to identify the group. In some predictive scenarios, we may have a new ranked list with very few rankings for which we would like to know what its ranking function would be, in order to predict unseen rankings. We first need to identify its group assignment, in order to identify its ranking function.



(a) PubFig (b) OSR

Figure 5.2: Predicting Group Assignments Based on Subset Length.

Figure 5.2 shows the clustering results when only a subset of the ranked list (of a specified length) is used for group assignment. This experiment is for *All* attributes with *Random* sampling. The figure shows that for both PLRM and PLM, the longer the subset length used, the more accurate is the group assignment, which is reasonable because there is more information to identify the group. In relative terms, PLRM considerably outperforms the PLM, due to the former's feature-based nature. PLM may not result in reasonable predictions for unseen items, in which

Method	PubFig			
	{Masculine, Pointy Nose, Round Face}			All
	Random	Exclusive	All-for-One	Random
PLRM	*89.8*	**91.6**	**89.7**	85.1
PLM+PLR	**91.3**	89.2	80.6	**86.6**
Method	OSR			
	{Natural, Large Objects, Close Depth}			All
	Random	Exclusive	All-for-One	Random
PLRM	**71.7**	**74.7**	**89.5**	**76.9**
PLM+PLR	63.4	66.9	86.3	66.5

Table 5.4: Ranking (Normalized Kendall's Tau).

case the most probable cluster according to π is chosen.

Ranking Prediction It is possible to predict unseen ranking of items if the preference group of a judge (ranked list) is known (see Section 5.3.2). Given a set of items, we consider the scenario when a judge is first asked to rank some subset of these items. After the initial ranking, we then predict the preference group for the judge, and determine the ranking for the rest of the items on the judge's behalf.

The previously identified baseline PLM can only perform clustering, but not ranking prediction of unseen items because it does not consider features. To investigate the effects of both clustering and ranking, we consider a pipeline baseline, involving first clustering using PLM followed by learning-to-rank using PLR for each cluster (see Section 5.2).

Table 5.4 shows the results of ranking prediction when the different sampling strategies are applied, corresponding to the clustering experiments in Table 5.3. We reserve a subset length of 3 and 2 for *PubFig* and *OSR* respectively for first predicting the group assignment, which still leaves sufficient remaining rankings to be predicted for all attributes. Thereafter, we use the assigned group's ranking function.

In general, the ranking prediction results are consistent with the clustering results. Most of the time, when PLRM has better clustering performance, it also has better ranking performance. This is most notable for *OSR*, whereby PLRM consistently has better ranking performance than PLM+PLR across different sampling

strategies. For *PubFig*, that is mostly true, with a couple of reasonable exceptions. For the three distinct attributes, in the *Random* experiment, PLRM and PLM have very similar clustering performances for reasons cited above. Therefore it is reasonable that PLRM and PLM+PLR also have very similar ranking performances (italics indicates that the difference is not statistically significant). For *All* attributes, PLRM has slightly lower ranking performance. This may be due to the fact that not all the 11 attributes in *PubFig* are distinct. As we will see shortly, the intrinsic number of preference groups is around 3 in *PubFig*, implying that some of the 11 attributes may be correlated. For *OSR*, PLRM is better.

5.4.4 Compare to Non-Heterogeneous Approach

We now look into the utility of PLRM in ranking scenarios, particularly comparing to methods that do not assume a heterogeneous ranking population and thus rely on a central ranking function. In addition to *PubFig* and *OSR*, in these experiments we use two additional datasets containing user opinion responses: Computer Survey (*Comp*) and Digital Cameras (*DCam*). Since we know the users who rate the products in *DCam*, we assign each user to a particular preference group. These datasets were not studied in the previous section because they lacked ground truth for clustering. However, they still allow for validation of rankings.

Number of Clusters For this comparison, we first need to determine the number of preference groups for PLRM. It is not advisable to rely on the known number of attributes. For one reason, the intrinsic number of preference groups may be different than the number of attributes. For another reason, some datasets such as *Comp* and *DCam* do not have known preference groups. Therefore, we first determine the intrinsic number of preference groups by varying K for each dataset, and measure the ranking prediction using the assigned group's ranking function. For each dataset, we try to accommodate as long a subset length for group assignment as possible, while still allowing sufficient remaining items to rank. The subset lengths are 3 for *PubFig*, 2 for *OSR*, 3 for *Comp*, and 3 to 5 for *DCam* (varying

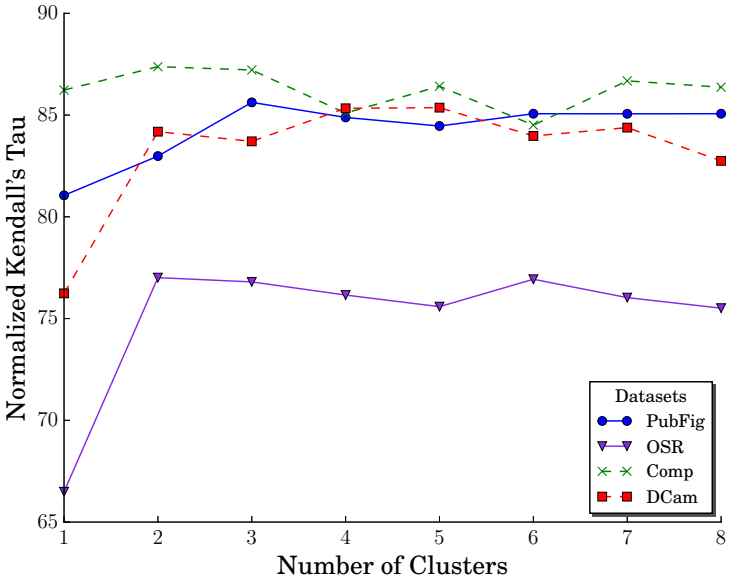


Figure 5.3: Ranking Prediction. PLRM with varying number of clusters K.

because users have rated different numbers of items).

Figure 5.3 shows ranking prediction quality plotted against the number of clusters for each dataset. It shows that the greatest gains come from increasing the number of clusters from 1 to 2, thereafter the performance increases slower or converges. The numbers of clusters or preference groups maximizing the ranking performance are 3 for *PubFig*, 2 for *OSR*, and 5 for *DCam*. For *Comp*, there is not much difference among different numbers of clusters, but 2 clusters are slightly better than 1; this may be an indicator that there is less heterogeneity overall for *Comp*. Subsequently, we will use these numbers to compare to the baseline.

Ranking Prediction As our focus is on validating the applicability to heterogeneous ranking population, the most appropriate baseline to PLRM in this respect is PLR (see Section 5.2), which is based on the same underlying Plackett-Luce regression modeling, but does not model a mixture.

Table 5.5 compares PLRM with the specified numbers of clusters to PLR, which effectively only has one cluster. The results show that PLRM outperforms PLR on all datasets. This outperformance is quite considerable and statistically significant for *PubFig*, *OSR*, and *DCam*. This outperformance helps to support the case that

Method	PubFig	OSR	Comp	DCam
PLRM	**85.6**	**77.0**	**87.3**	**85.4**
PLR	81.1	66.5	*86.2*	76.2

Table 5.5: Ranking Prediction: PLRM vs. PLR.

Method	PubFig	OSR	Comp	DCam
PLR	81.1	66.5	86.2	76.2
Coordinate Ascent	75.0	58.0	81.9	75.6
SVM-Rank	78.4	67.0	86.6	71.0
RankNet	76.2	63.7	79.6	67.2
ListNet	76.6	64.4	86.8	67.9
RankBoost	78.6	61.7	83.6	58.6

Table 5.6: Comparison of Learning to Rank Methods.

when the population has a high level of heterogeneity, a method that considers multiple latent preference groups such as PLRM have the potential to do significantly better. For a dataset that does not have a high level of heterogeneity in the first place, such as *Comp*, the improvement is rather modest.

Though PLR is a simplified version of PLRM without mixture modeling, we point out that PLR is not a weak baseline, and is actually a competitive learning to rank method in its own right. Table 5.6 benchmarks PLR to popular learning to rank methods, such as Coordinate Ascent [119], SVM-Rank [80], RankNet [17], ListNet [20], and RankBoost [49]. We use their implementations in RankLib[6] and SVM^{rank}[7]. Because these methods are based on very different algorithms, these are provided as a point of reference, rather than as a direct comparison. Nevertheless, Table 5.6 shows that PLR gets good results on the datasets. In many cases, PLR is comparable or even better. This underlines the relative strength of PLR, which in turn lends greater support to PLRM's outperformance.

Brief Comment on Running Time Our focus in this work is on effectiveness and accuracy, and not on computational efficiency. The training times are reasonable. For instance, among the learning to rank methods, PLR's training takes less than a minute. This is comparable to SVM-Rank, and faster than other learning to

[6]https://sourceforge.net/p/lemur/wiki/RankLib/
[7]https://www.cs.cornell.edu/people/tj/svm_light/

rank methods. In turn, the training of PLRM requires optimization of latent variables with EM algorithm. Hence, it takes more time, which increases with the required number of clusters. For instance, it takes under 30 iterations till convergence on *PubFig*, with each iteration taking a minute on average. These time measurements were conducted on a PC with Intel Core i5 CPU 3.3 GHz and 12GB of RAM running Windows OS.

Case Study To gain a sense of the nature of the clusters that PLRM learns, we show the top five features for each of the five clusters or preference groups learnt from *DCam*:

1. Pentaprism VF (viewfinder), mid-size, CMOS sensor, CCD sensor, mirrorless-style;
2. Pentaprism VF, mid-size, screen size, CMOS sensor, BSI-CMOS sensor, pentamirror VF;
3. Pentaprism VF, mid-size, Foveon X3 sensor, pentamirror VF, rangefinder-style;
4. Tunnel VF, compact, mirrorless, pentaprism VF, Foveon X3 sensor;
5. Max ISO, electronic VF, pentamirror VF, BSI-CMOS sensor, compact.

The first three groups favor mid-size cameras with pentaprism viewfinders having CMOS-like sensors. The last two preference groups give more credit to compact cameras than to mid-size cameras. The last preference group also favors cameras that can work in low-light conditions. The top features *Max ISO* (indicating maximal light sensitivity) and *BSI-CMOS Sensor* (a specific type of sensor that increases amount of light can be captured) support this observation.

5.5 Discussion

In this work, we build on the Plackett-Luce model, first introduced by Plackett [141] and Luce [106] independently. It expresses the probability of a permutation in terms of element-specific parameters and is related to the aggregation model discussed in Section 2.2.1. Aside from Plackett-Luce, there are other paradigms for expressing distribution over rankings. Some are based on the notion of distances [47]. For instance, Mallows model [110] expresses the probability of a permutation in terms of its distance to a reference permutation. [4, 94, 118] consider a mixture of distance-based models. Yet another paradigm is Bradley-Terry [13, 48], based on pairwise

comparisons.

Our notion of a regression based on Plackett-Luce model may be confused with the one introduced in [3], however, their formulation aims to deal with categorical data and significantly differs from Plackett-Luce regression discussed here.

Plackett-Luce regression mixture falls under preference mining category, which is discussed in Section 2.3.2. In the next chapter, we briefly discuss the contribution of this study and outline future directions.

CHAPTER 6

EXPLAINING ENTITY COMPARISONS

The ability to quickly analyse large collections of text data becomes more and more critical. Since manual analysis is time-consuming and expensive even for relatively small collections of texts, computer-aided processing is necessary to achieve preliminary exploratory analysis in short time. Topic model is a class of probabilistic models that "reduce" an input corpus into a manageable number of "topics", where each topic congeals words that tend to co-occur with one another in documents, thus signifying some hidden semantics in the corpus. By identifying the topics that essentialize the corpus, and discerning which ones predominate in a specific document, a topic model is a crucial tool for sensemaking.

Increasingly, there are real-world scenarios where the purpose of analyzing a corpus is to compare entities based on their textual representations (documents). For example, analysts may seek to explore why one country could achieve a better healthcare (alternatively economic, educational, etc.) outcome than another based on certain documents such as country reports. Funding agencies or scientists may seek a better understanding of what may get a grant proposal funded over another based on proposal contents. Among the products browsed by consumers, some are purchased while others are not. Among the purchases, some satisfy customers more than others. Thus, delving into product descriptions or reviews could reveal insights on consumer preferences.

An unsupervised topic model, such as Latent Dirichlet Allocation (LDA) [10],

is oriented towards capturing topics that could reflect the word co-occurrences in the corpus well. In doing so, its topics tend to capture general semantics. For instance, a topic model based on country reports may well discover topics aligned to geographical or linguistic commonalities, which however may or may not bear direct relevance to the question at hand (e.g., healthcare outcomes). Topics based on grant proposals may describe various scientific foci, though such topics may group competing proposals but may not be indicative of their likelihood of acceptance. In turn, product reviews may yield topics focused on brands or features, but such topics may coalesce opposing sentiment polarities as words with positive (e.g., "good") or negative (e.g., "bad") connotations tend to co-occur with similar words, e.g., "battery life".

6.1 Problem

We postulate that introducing supervision that signals how one entity (document) compares to another into topic modeling would better align the topics discovered from a corpus to the comparison dimension of interest. Suppose that in addition to a corpus of documents, we are also given some pairwise comparisons among the documents. Each pairwise comparison indicates which of two documents is considered "higher" or "better" according to some desired dimension (e.g., a country healthier than another, a pair of accepted and rejected proposals, a product preferred to another). Constraining the topic model to "comply" with the pairwise comparison observations may yield topics that differentiate entities along the dimension of interest (e.g., why one product is preferred), rather than simply discovering commonality in words (e.g., products with similar features).

There are inherent advantages to modeling pairwise comparisons as opposed to pointwise ratings. The latter may not even be available in some scenarios. In the implicit feedback settings, comparisons are naturally relative, when it may be known that one entity is better, but not necessarily clear by how much in absolute

terms. For instance, when a consumer browses but skips a product, and purchases another, the latter is probably preferred to the former. Even when pointwise ratings are available, fitting the absolute ratings may not always be appropriate. They may have been assigned by different human subjects (with varying biases and scales), rendering direct comparison across human subjects inequitable.

Approach In a nutshell, our proposed model CompareLDA associates each topic with a distribution over words, and each document with a distribution over topics, as in a conventional topic model. In addition, a document topic proportion maps to a merit value that ranks documents. These entity merit values probabilistically determine the observed pairwise comparison outcomes. As a generative model, CompareLDA has generative capacity over unobserved pairwise comparisons. This enables the model to learn even with relatively few observed comparisons, as we will see in the experiments. It also generalizes to out-of-training documents whose topic distributions and entity merit values could be inferred accordingly.

6.2 Model

In this section, we describe the development of our approach CompareLDA, as well as the methodology to fit the model parameters through variational inference.

CompareLDA is a supervised topic model with non-linear response. A response variable is associated with a pair of documents (each concerning an entity), and indicates the comparison result: which of the two entities is "better" or ranked higher than the other. The notion of comparison is latent, and may vary from application to application.

CompareLDA extends Latent Dirichlet Allocation (LDA). It has the same basis assumption regarding the association of topics and words, but also a significant distinction in its incorporation of pairwise comparisons (as we will see shortly). Each document is generated from a set of latent topics. A topic is an unknown distribution over the corpus vocabulary, which has to be inferred from the data. The documents

in a corpus share the same set of topics, but mix them in different proportions. The topics are associated with the words and essentially defined as distributions over the vocabulary. Each word is a sample from only one topic distribution.

We are given a set of entities $D = \{d_i\}_{i=1}^{N}$. An entity is represented by its textual form, a document. The notation d_i is used to refer to either entity or document. Furthermore, we assume that an oracle takes a pair of entities at a time, d_i and d_j, 'glances' at their documents, and makes a comparison decision: which of the two entities is better according to some definition of merit, e.g., healthier, more likely to get funded, preferred by consumers. The decision would be based on the topics discussed in the text rather than on individual words. The oracle makes M pairwise comparison decisions, providing the training data. CompareLDA seeks to reproduce this judging process by learning the topics, and inferring these topics for unseen documents.

6.2.1 Definition

CompareLDA unfolds the process in the following way. Each entity is imbued with a latent merit value, inducing a pairwise comparison with another entity. Suppose m_i and m_j are the respective merit values for a pair of entities d_i and d_j. If $m_i > m_j$, then d_i is more likely, though not certainly, to come out the winner in a comparison with d_j.

To define the probability of winning in a comparison, we use the sigmoid function:

$$\mathrm{P}(d_i \succ d_j) = \sigma(m_i - m_j) = \frac{1}{1 + e^{-(m_i - m_j)}}. \tag{6.1}$$

The greater is m_i than m_j, the higher the probability that d_i would be favored by the oracle, as the probability in Eq. 6.1 tends towards 1. When the merit values are similar $m_i \approx m_j$, the probability reflects uncertainty in the outcome, i.e., $\mathrm{P}(d_i \succ d_j) \approx 0.5$.

Presumably, the oracle obtains the comparison information from the topics. For instance, preferred products or healthier countries may be associated with special

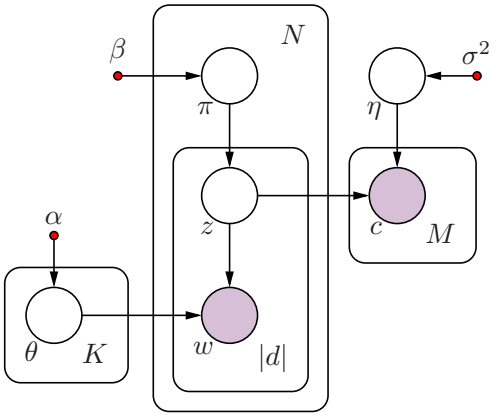


Figure 6.1: CompareLDA in Plate Notation.

qualities whose description manifests as topics. CompareLDA uses the empirical topic distributions of the texts, and transforms them into the entity merit values via a linear regression. Given a text d_i where each word w_{ij} is assigned to topic z_{ij}, we calculate its empirical topic distribution $\bar{z}_i$ as follows:

$$\bar{z}_i = \frac{1}{|d_i|} \sum_{j=1}^{|d_i|} z_{ij}. \tag{6.2}$$

For some regression parameters $\bar{\eta}$, we assume:

$$m_i = \bar{\eta} \cdot \bar{z}_i \tag{6.3}$$

Note that for such merit values as defined above, the bias term is effectively redundant, as it vanishes when comparison is concerned (Eq. 6.1).

The intuition behind regressing on topics is that some topics help to gain merit values (e.g., newly introduced product features), while others may decrease the merit values (e.g., discovered flaws). Considering the difference in the topic proportions of two products, $\bar{z}_i - \bar{z}_j$, we would be able to draw the conclusion on which entity is likely the winner.

6.2.2 Generative Process

Here we summarize the generative process of CompareLDA, whose plate notation is given in Figure 6.1.

1. We sample K topic distributions $\{\theta_i\}_{i=1}^{K}$ from Dirichlet distribution with α prior:

$$\theta_i \sim Dirichlet(\alpha).$$

2. We sample $\bar{\eta}$, the transformation weights from K-dimensional Gaussian with zero mean and σ^2 variance:

$$\bar{\eta} \sim \mathcal{N}(0, \sigma^2).$$

3. For each document:

   (a) We sample its topic proportion $\{\pi_i\}_{i=1}^{N}$ from Dirichlet distribution with β prior:

$$\pi_i \sim Dirichlet(\beta).$$

   (b) For each word w_{ij} in document d_i we sample its topic assignment variable z_{ij}:

$$z_{ij} \sim Categorical(\pi_i);$$

   and based on the topic assignment the observed word:

$$w_{ij} \sim Categorical(\theta_{z_{ij}}).$$

   (c) We calculate empirical topic proportion $\bar{z}_i$ and transform it to the entity merit value m_i:

$$m_i = \bar{\eta} \cdot \bar{z}_i = \bar{\eta} \cdot \left(\frac{1}{|d_i|} \sum_{i=1}^{|d_i|} z_{ij} \right).$$

4. For each pairwise comparison trial r_i, we sample the winner c_i:

$$c_i \sim Bernoulli(\sigma(m_{r_i[1]} - m_{r_i[2]})).$$

r_i is a pair of indices indicating which documents are compared in the trial, and c_i indicates the winner for the pair. If $c_i = 1$, then item $d_{r_i[1]}$ is the winner, otherwise $d_{r_i[2]}$ is.

The complete data likelihood for a set of entities/documents D and their corresponding pairwise comparison observations (R, C) is as follows:

$$\mathrm{P}(D, Z, \Theta, \Pi, R, C, \eta) = \mathrm{P}(\eta|\sigma^2) \prod_{i=1}^{K} \mathrm{P}(\theta_i|\alpha)$$
$$\times \prod_{i=1}^{N} \mathrm{P}(\pi_i|\beta) \times \prod_{i=1}^{N} \prod_{j=1}^{|d_i|} \mathrm{P}(z_{ij}|\pi_i)\mathrm{P}(w_{ij}|\theta_{z_{ij}}) \times \prod_{i=1}^{M} \mathrm{P}(c_i|m_{r_i[1]}, m_{r_i[2]}) \tag{6.4}$$

We consider the collapsed version of the likelihood by integrating out the multinomial parameters.

$$\mathrm{P}(D, Z, R, C, \eta) = \int_{\Theta} \int_{\Pi} \mathrm{P}(D, Z, \Theta, \Pi, R, \eta)$$
$$= \mathrm{P}(\bar{\eta}|\sigma^2) \times \prod_{i=1}^{N} \mathrm{P}(z_{i\cdot}|\beta) \times \mathrm{P}(W|Z, \alpha) \times \prod_{i=1}^{M} \mathrm{P}(c_i|m_{r_i[1]}, m_{r_i[2]}), \tag{6.5}$$

where $\mathrm{P}(z_{i\cdot}|\beta)$ and $\mathrm{P}(W|Z, \alpha)$ are Dirichlet-multinomial distributions over topics and words.

6.2.3 Model Fitting

To find the maximizing latent parameters $\bar{\eta}$ and Z for the posteriori distribution, we use variational approximation algorithm to optimize the evidence lower bound L.

$$\log \mathrm{P}(D, Z, R, C, \eta) \geq L(Z, R, C, \eta) = \log \mathrm{P}(\eta|\sigma^2) + \sum_{i=1}^{N} \langle \log \mathrm{P}(Z_i|\alpha) \rangle + \langle \log \mathrm{P}(W|Z, \beta) \rangle + \sum_{i=1}^{M} \left\langle \log \mathrm{P}(c_i|m_{r_i[1]}, m_{r_i[2]}) \right\rangle + \mathrm{H}(q), \tag{6.6}$$

where $\langle \cdot \rangle$ indicates expectation taken with respect to the variational distribution $q(Z)$, and $\mathrm{H}(\cdot)$ is the entropy operator. We treat $\bar{\eta}$ as a model parameter.

We factorize the variational distribution into independent factors, one for each z_{ij}. In most of the cases, assuming the fully factorized distribution is enough to obtain the tractable closed-form update equations, where each factorized distribution will have the same form as their conjugate priors. However, due to non-linear interaction term between the entities, the update formulas are intractable. We additionally assume that each $q(z_{ij})$ is an indicator probability distribution, which places all the probability mass on the one topic $q(z_{ij}) = q(z_{ij}|v_{ij}) = \mathrm{I}_{[z_{ij}=v_{ij}]}$. Essentially each v_{ij} represents empirical topic assignment for word w_{ij}.

$$q(Z|V) = \prod_{i=1}^{N} \prod_{j=1}^{|d_i|} q(z_{ij}|v_{ij}) = \prod_{i=1}^{N} \prod_{j=1}^{|d_i|} \mathrm{I}_{[z_{ij}=v_{ij}]} \tag{6.7}$$

Note that under this assumption, expectation operator does not change any $f(V)$: $\langle f(V) \rangle = f(V)$.

We use coordinate-ascent variational approximation, and maximize the evidence lower bound with respect to $\bar{\eta}$ and V, optimizing each parameter in turn. Further we assume that the document comparisons are configured in such a way that for any i, $c_i = 1$, to simplify the description of equations.

Optimizing η: With V fixed, we want to optimize the following objective:

$$f(\bar{\eta}) = \log \mathrm{P}(\bar{\eta}|\sigma^2) + \sum_{i=1}^{M} \left\langle \log \mathrm{P}(c_i|m_{r_i[1]}, m_{r_i[2]}) \right\rangle$$
$$= -\sum_{i=1}^{K} \frac{\eta_i^2}{2\sigma^2} - \sum_{i=1}^{M} \log\left(1 + e^{-\bar{\eta}\cdot\left(\bar{v}_{r_i[1]} - \bar{v}_{r_i[2]}\right)}\right) \tag{6.8}$$

where $\bar{v}_i = \left(\sum_{j=1}^{|d_i|} v_{ij}\right) / |d_i|$. We develop a basic gradient ascent algorithm, taking derivative of $f(\bar{\eta})$ with respect to η_j:

$$f'_{\eta_j}(\bar{\eta}) = -\frac{\eta_j}{\sigma^2} - \sum_{i=1}^{M} \frac{\left(\bar{v}_{r_i[1]}\right)_j - \left(\bar{v}_{r_i[2]}\right)_j}{1 + e^{-\bar{\eta}\cdot\left(\bar{v}_{r_i[1]} - \bar{v}_{r_i[2]}\right)}}. \tag{6.9}$$

Optimizing V: With $\bar{\eta}$ fixed, we seek the empirical topic assignments that maximize (6.6). As exhaustive search for the optimal solution has exponential complexity and, therefore, is infeasible for any reasonable datasets, we exploit the probabilistic nature of the model and develop Metropolis-Hastings [60] procedure for approximate optimization. Metropolis-Hasting is a method for obtaining a sequence of random samples from a probability distribution for which direct sampling is difficult. In case of CompareLDA, the procedure changes one word-topic assignment v_j as a time, eventually approximating the probability distribution of word-topic assignments for the whole corpus. We work with probability distribution induced by the empirical lower bound. Here we compute the difference between two word-topic assignments, that are different only in one assignment, current assignment $v_{ij} = a$ and evaluated assignment $v_{ij} = b$ (chosen at random).

$$
\begin{aligned}
E_{a\to b}^{v_{ij}} &= \log\left(\beta + n_i(a) - 1\right) - \log\left(\beta + n_i(b)\right) \\
&+ \log\left(\alpha + n(a, w_{ij}) - 1\right) - \log\left(\alpha + n(b, w_{ij})\right) \\
&- \log\left(\alpha|X| + n(a, \cdot)\right) - 1) + \log\left(\alpha|X| + n(b, \cdot)\right) \\
&+ \sum_{k=1}^{M} \log\left(1 + e^{\bar{\eta}\cdot\left(\bar{v}_{r_k[1]} - \bar{v}_{r_k[2]}\right) + \delta_i\left(\mathrm{I}_{[r_k[1]=i]} - \mathrm{I}_{[r_k[2]=i]}\right)}\right) \\
&- \sum_{k=1}^{M} \log\left(1 + e^{\bar{\eta}\cdot\left(\bar{v}_{r_k[1]} - \bar{v}_{r_k[2]}\right)}\right)
\end{aligned} \tag{6.10}
$$

where X is vocabulary, $n_i(z)$ is document-topic count, $n(z, w)$ is term-topic count, $n(z, \cdot) = \sum_{w \in X} n(z, w)$, and $\delta_i = \left(\eta_b - \eta_a\right) / |d_i|$. The acceptance probability $\gamma = \exp\left(-E_{a\to b}^{v_{ij}}\right)$ then indicates how probable the evaluated sample is with respect to the current assignment, according to the approximated sample. If we attempt to move to an assignment which is more probable than the current one w.r.t the evidence lower bound, we always accept the move. If the move is taken towards the less probable assignment, it will be accepted with γ probability.

For the purpose of optimization, the Metropolis-Hasting algorithm can be converted to simulated annealing procedure, where acceptance probability γ is reduced over time, to prevent the moves towards less probable states. We use $\gamma^{\frac{1}{T}}$ as the probability of accepting a new assignment, where T, a temperature, approaches 0 as the iteration count increases.

6.3 Experiments

Our experimental objective is to validate the efficacy of CompareLDA in deriving topics that are well-aligned to document comparisons. First, we investigate the utility of modeling pairwise comparisons as supervision on topic models, vis-à-vis a baseline with pointwise supervision. Thereafter, we move to additional experiments and discussions, which shed light upon various aspects of the model.

6.3.1 Datasets

For experiments, we rely on public text corpora, whereby not only it is meaningful to attach the notion of comparisons to entities within a corpus, but the comparisons also define a part of the core semantics of the corpus. We identify three such corpora that yield five experimental datasets as follows.

Wikipedia The first is a set of three datasets constructed from Wikipedia[1] pages with country infobox and category. The corpus contains 467 entities (countries and associations, e.g., BRICS, NATO). The page content is the document. As supervision, we induce three sets of pairwise comparisons from Wikipedia's lists of countries: by alcohol consumption (AC), by cigarette consumption (CC), and by life expectancy (LE). Each list results in a different number of pairwise comparisons: 17,955 for AC, 16,290 for CC, and 16,653 for LE. Coupled with the text corpus, each set of pairwise comparisons constitutes a dataset. Our intention is to study if CompareLDA could derive different topics from the same corpus, but with different pairwise comparisons.

Product Reviews The second dataset is from Amazon as described in [112, 113]. Here, an entity is a review from the Electronics category. We assume that the reviews mention various features and qualities to illustrate the product's intrinsic merit. As supervision, we induce pairwise comparisons based on the number of stars indicated by the reviews. The "positive" reviews (5 and 4 stars) are compared to the "negative" reviews (2 and 1 stars), i.e., positive "win" over negative. We sample 10,000 reviews at random to assemble the corpus. For this dataset, out of all the induced comparisons, we randomly sample 0.25% to simulate a realistic scenario of where only partial comparisons have been observed. The dataset contains 43,881 pairwise comparisons.

Movie Reviews The third dataset contains movie reviews [134]. We used the 4-star scale as described in [134] to induce pairwise comparisons, i.e., a review with more stars "wins". The intuition here is to discover the topics that are aligned with

[1] We used the Wikipedia dump dated 30 July 2018.

what makes a good movie. As before for the Product Reviews corpus, we retain only 0.25% of comparisons as supervision. The corpus contains 5,006 documents along with 21,965 comparisons.

Each dataset is split into training and testing folds in $80{:}20$ proportion respectively. Conservatively, comparisons that cross folds are ignored during training and evaluation. The corpora undergo the same preprocessing steps, i.e., removing short documents, punctuation, stop-words; the tokens converted to their lemmas. For each dataset, we retain top 5000-term vocabulary selected by tf-idf.

6.3.2 Evaluation

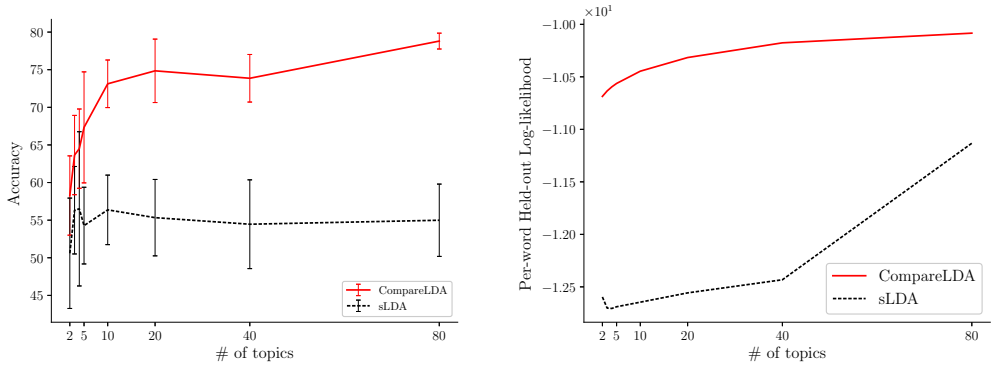
To jointly model topics and pairwise comparisons, a method should be adept at both assigning topics to words and assessing the ranking among documents. We explore these respective dimensions of evaluation.

Ranking To assess ranking quality, we report the ranking accuracy. For two entities d_i and d_j, we define a function f, where $f(d_i, d_j)$ returns 1 if d_i is preferred over d_j in comparison, 0 when the preference between d_i and d_j is not assumed, and -1 when d_j is preferred over d_i. Given a set of entities $D = \{d_i\}_{i=1}^{M}$ and reference comparison function f (ground-truth) and its approximation g (prediction), we define the ranking accuracy (or accuracy) as follows:

$$A = \frac{\sum_{i=1}^{M} \sum_{j=1}^{M} \mathrm{I}_{[f(d_i,d_j)=1]} \mathrm{I}_{[g(d_i,d_j)=1]}}{\sum_{i=1}^{M} \sum_{j=1}^{M} \mathrm{I}_{[f(d_i,d_j)=1]}} \tag{6.11}$$

When the approximation and reference functions are identical, then approximation is good and $A = 1$. In case of complete disagreement, $g(d_i, d_j) \neq 1$ for every d_i and d_j such that $f(d_i, d_j) = 1$, then $A = 0$. The ranking accuracy is closely related to Kendall's Tau. While Kendall's Tau is suitable for totally ordered sets, the proposed metric considers only items for which relative comparison makes sense, and thus it is more appropriate in our study.

Topics Topic models are commonly evaluated by estimating probability of



(a) Accuracy (b) Log-likelihood

Figure 6.2: Wikipedia Dataset Ranked by Alcohol Consumption.

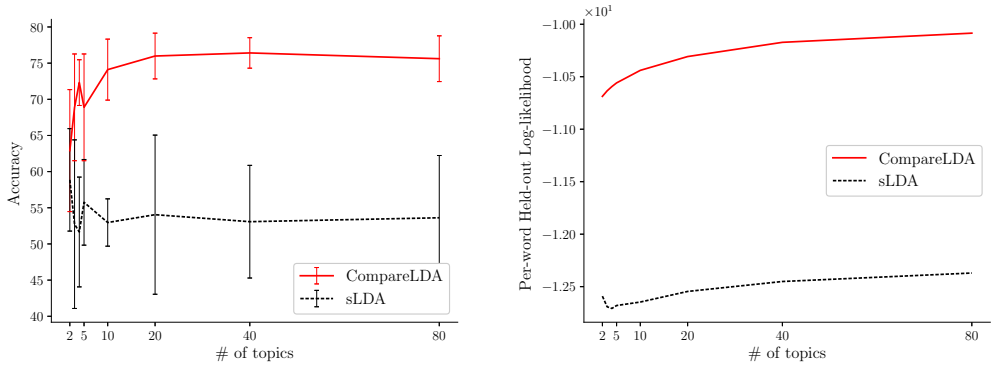
held-out documents. The intuition is that a better model will give rise to the likelihood of held-out documents D. $L = \frac{\log \mathrm{P}(D|\mathcal{M})}{\sum_{d \in D} |d|}$ is per-word log-likelihood for an LDA model with parameters $\mathcal{M}$. To approximate L marginalized over all possible topic assignments, we use Chib-style estimator [178].

6.3.3 Comparison to Baseline

As our proposed CompareLDA[2] is the first topic model supervised by pairwise comparisons, our main baseline is the previous topic model supervised by pointwise response variables. Among such models (see Related Work), sLDA[3] bases the prediction on empirical topic assignments, which makes the former an ideal baseline to CompareLDA that also uses empirical topic assignments. sLDA predicts merit values of documents directly via regression. When supervision is supplied in terms of pairwise comparisons, this model is not immediately applicable. Instead, it requires preprocessing to convert the pairwise comparisons into pointwise merit values for each document, which are then supplied to sLDA. For conversion, we employ the Bradley-Terry-Luce (BTL) model [13, 108] due to its similarity to the comparison component of CompareLDA.

We evaluate both models by varying the number of topics (default is 80). The

[2]We plan to eventually release our implementation over a public repository upon publication.
[3]We used the following implementation: `https://github.com/vietansegan/segan/`



(a) Accuracy (b) Log-likelihood

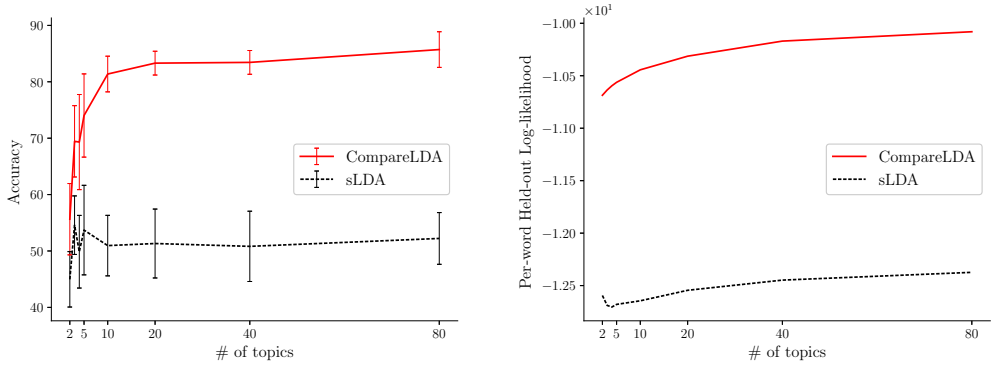
Figure 6.3: Wikipedia Dataset Ranked by Cigarette Consumption.

experiments are repeated 10 times with different random initializations. Figures 6.2 to 6.6 show the results for the five datasets for both accuracy and likelihood.

CompareLDA consistently outperforms sLDA on each dataset with respect to both evaluation dimensions. For instance, Figure 6.2(a) shows the ranking accuracy for the Wikipedia dataset ranked by alcohol consumption. In general CompareLDA achieves better results as the number of topics increases. The accuracy gap over sLDA increases significantly, when the number of topics hits 10 and beyond. Bars denote the standard deviation. The deviation tends to reduce as the number of topics increase. This reflects well on CompareLDA, suggesting that supervision in the form of pairwise comparisons helps to uncover the ranking structure. CompareLDA demonstrates better alignment of topics with rankings, reaching higher than 75% accuracy. sLDA shows lower performance, hovering around 55%, which is close to random; this suggests that the regression objective does not fit the problem, when pairwise supervision is concerned.

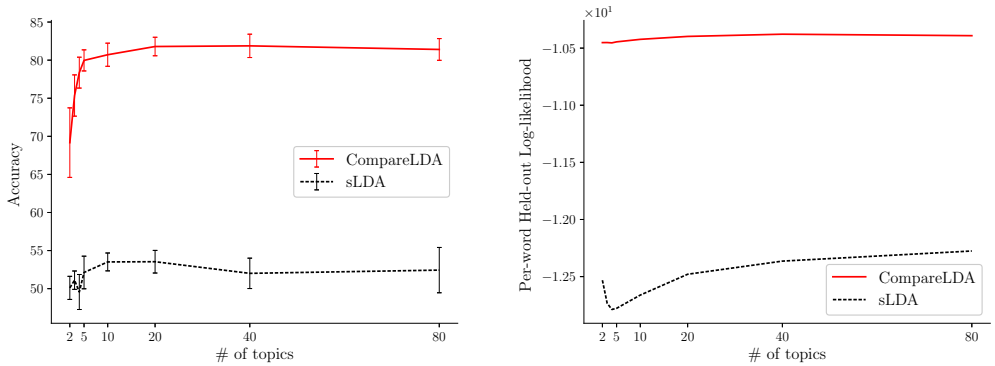
In turn, Figure 6.2(b) shows that CompareLDA reaches significantly higher log-likelihood than sLDA as well. The log-likelihood plots show significant outperformance by CompareLDA over sLDA even when the number of topics is small. It seems that the regression objective interferes with the objective to infer "good" predictable topics.

The other Wikipedia datasets ranked by cigarette consumption (Figure 6.3) and



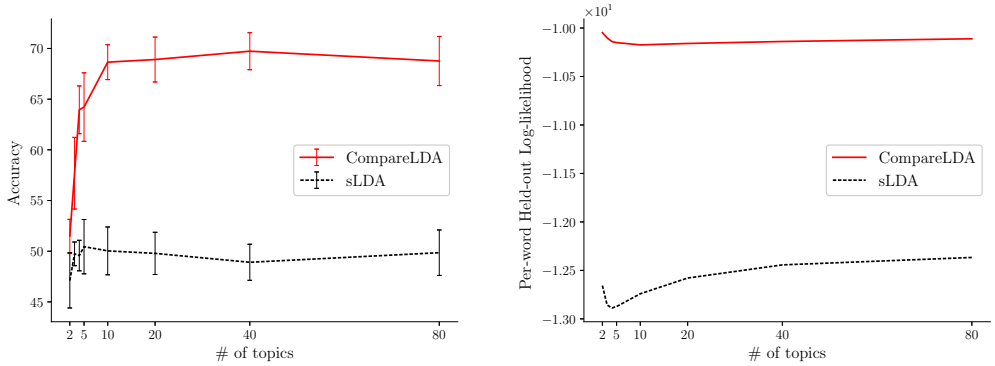
(a) Accuracy (b) Log-likelihood

Figure 6.4: Wikipedia Dataset Ranked by Life Expectancy.



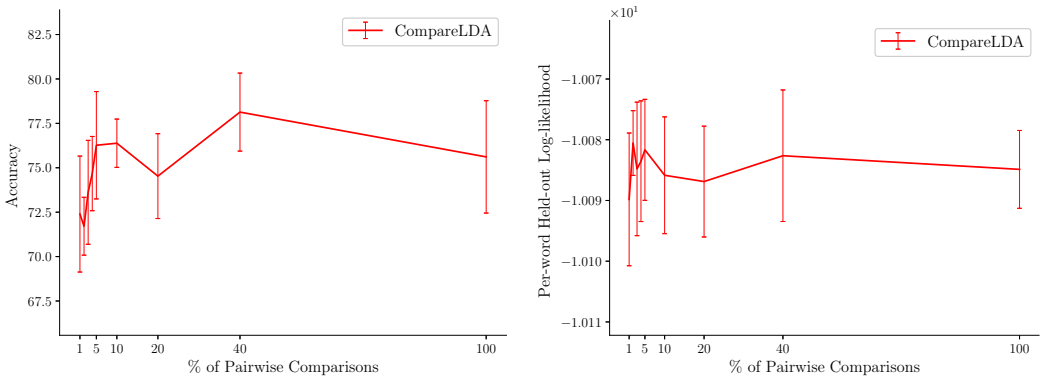
(a) Accuracy (b) Log-likelihood

Figure 6.5: Product Reviews.



(a) Accuracy (b) Log-likelihood

Figure 6.6: Movie Reviews.



(a) Accuracy (b) Log-likelihood

Figure 6.7: Varying Number of Pairwise Comparisons. Wikipedia dataset ranked by cigarette consumption. The other rankings show similar behavior.

life expectancy (Figure 6.4) show similar trends, evidence that CompareLDA could derive different topic models from the same corpus by fitting different supervisions. For the review datasets (Figures 6.5 and 6.6), the outperformance is more vivid and starts with a few topics.

6.3.4 Amount of Supervision

We study the amount of supervision, as the number of comparisons for the fully ordered set of N elements is quadratic, $\mathcal{O}(N^2)$, i.e., harder to obtain than independently labeling each document. Figure 6.7 shows the performance when the amount of supervision gradually increases from 1% to 100% on the Wikipedia dataset ranked by cigarette consumption for 80 topics (other rankings and topic counts show similar results). Figure 6.7(a) shows that initially ranking accuracy grows fast as the amount of supervision increases. When 5% of supervision is supplied, rankings accuracy remains stable. Figure 6.7(b) shows that log-likelihood remains stable regardless of the amount of supervision. These results indicate that CompareLDA does not require fully ordered set to fit the model and, therefore, a small subset of comparisons may be used to achieve high ranking performance and topic quality.

Data	CompareLDA	LDA+BTL-R
Wikipedia (AC)	$\mathbf{78.8} \pm 0.8$	75.4 ± 1.6
Wikipedia (CC)	75.6 ± 2.3	74.9 ± 1.8
Wikipedia (LE)	85.7 ± 2.3	84.0 ± 1.8
Product Reviews	$\mathbf{81.4} \pm 1.0$	76.7 ± 1.0
Movie Reviews	$\mathbf{68.8} \pm 1.7$	63.8 ± 1.3

Table 6.1: Accuracy. AC - Alcohol Consumption, CC - Cigarette Consumption, LE - Life Expectancy.

6.3.5 Joint vs. Pipeline Models

One may improbably surmise that LDA may naturally align with document comparisons anyway, even without supervision. To debunk this, we consider a decoupled form of CompareLDA, which first discovers topics with LDA, and then solves the comparison problem using the empirical topic assignments. To tackle the pairwise comparison, we introduce Bradley-Terry-Luce regression (BTL-R), which is similar to CompareLDA's comparison modeling but done as a separate step. We refer to this pipeline as LDA+BTL-R.

Table 6.1 shows the ranking accuracy (and 95% confidence intervals). Bold typeface indicates statistically significant difference. CompareLDA shows better results than its pipeline equivalent on every dataset, with significant improvement on Wikipedia ranked by alcohol consumption and the review datasets. In turn, for the held-out log-likelihood, one may expect some decrease in performance due to additional objective to satisfy the comparison supervision, whereas LDA (BTL-R part does not influence topic inference in this case) cares only about getting the topics right. Gratifyingly, Table 6.2 shows that in fact there is no significant difference between the topics derived by CompareLDA and LDA, supporting that CompareLDA could align topics to comparisons well without hurting the likelihood.

6.3.6 sLDA Supervision

As mentioned earlier, sLDA requires pointwise supervision. When the input is pairwise, we need a preprocessing step. In a scenario where some form of pointwise

Data	CompareLDA	LDA(+BTL-R)
Wikipedia (AC)	-10.084 ± 0.007	-10.084 ± 0.010
Wikipedia (CC)	-10.085 ± 0.005	-10.084 ± 0.010
Wikipedia (LE)	-10.081 ± 0.010	-10.084 ± 0.010
Product Reviews	-10.391 ± 0.003	-10.389 ± 0.003
Movie Reviews	-10.111 ± 0.004	-10.111 ± 0.002

Table 6.2: Log-likelihood. AC - Alcohol Consumption, CC - Cigarette Consumption, LE - Life Expectancy.

Data	sLDA	sLDA*
Wikipedia (AC)	55.0 ± 3.4	55.6 ± 4.2
Wikipedia (CC)	53.6 ± 6.2	52.6 ± 5.2
Wikipedia (LE)	52.2 ± 3.3	50.8 ± 3.5

Table 6.3: sLDA Accuracy. AC - Alcohol Consumption, CC - Cigarette Consumption, LE - Life Expectancy.

response exists, we could alternatively use that directly, e.g., the rank position in the list for the Wikipedia dataset. We look into whether the two forms of supervision affect the results much. sLDA* is supervised with the ranked list, whereas sLDA is supervised with comparisons. Table 6.3 shows that there is no significant difference for ranking accuracy between the two. The BTL transformation matters when we explore held-out log-likelihood on Wikipedia ranked by alcohol consumption (see Figure 6.4), where it helps to achieve significantly better performance. However, the differences for the other Wikipedia rankings are not significant. In any case, the form of sLDA supervision would not affect the earlier conclusions on the relative comparisons with CompareLDA.

Data	sLDA	sLDA*
Wikipedia (AC)	$\mathbf{-11.132} \pm 0.008$	-12.367 ± 0.007
Wikipedia (CC)	-12.370 ± 0.006	-12.368 ± 0.011
Wikipedia (LE)	-12.374 ± 0.009	-12.373 ± 0.016

Table 6.4: sLDA Log-likelihood. AC - Alcohol Consumption, CC - Cigarette Consumption, LE - Life Expectancy.

Top $\bar{\eta}$-positive Topics	Top $\bar{\eta}$-negative Topics
work great well phone use also everything since need easy good set recommend issue clear	product would one back new month work buy get warranty worked return issue problem year
picture great tv price good quality love amazing feature best get really recommend got better	one would tried time review product work bought got money new first returned different try
color great little look came still perfect get easy could love easily really would want	device work adapter even connection get computer use unit time product well network car ca
one fan great two really also work air new room put purchased right got connector	year bought still first week working month warranty another one since would two completely last
one use bought price year well good model work still frame know used wanted made	one thing like money buy get even worth really could got make review cheap going

Table 6.5: CompareLDA Topics for Product Reviews.

6.3.7 Topics

To get a sense of the semantics reflected by the topics, we show 5 topics associated with top positive and top negative $\bar{\eta}$ parameters. For Product Reviews (Table 6.5), the $\bar{\eta}$-positive topics tend to associate with words of positive connotations, e.g., great, well, good, love, etc. $\bar{\eta}$-negative topics tend to talk about issues, money, returns, and warranty. For Wikipedia by alcohol consumption (Table 6.6), some of $\bar{\eta}$-positive topics are associated with the country and region names, e.g., Lithuania, Estonia, Baltic. These places are, in fact, reported to have higher level of alcohol consumption. The country name clusters are present in $\bar{\eta}$-negative topics as well. For instance, Islamic countries may have lower level of alcohol consumption due to religious prohibition.

6.4 Discussion

In this chapter, we describe CompareLDA, a topic model for document comparison. It is novel in its incorporation of pairwise comparison to align the topics learnt to the comparison dimension of interest. Qualitative analysis shows that the topics achieve semantic coherence and are useful in understanding and explaining comparisons.

Top $\bar{\eta}$-positive Topics	Top $\bar{\eta}$-negative Topics
world country europe century largest national european language first hungary central union law film modern	region war people city group population state force press east army official political eastern refugee
state world population united country economy nation census press million largest development economic new tax	country world national development population education party per election rate nation trade health year specie
world war university british pp united new great country history london time million britain oxford	bangladesh libya morocco algeria country arab libyan moroccan bengal africa algerian berber bengali government sahara
lithuania estonia lithuanian german germany estonian baltic soviet vilnius state lietuvos independence europe county eesti	pakistan muslim islamic country islam maldives muhammad persian pakistani sultanate military india filipino ali power
population language needed people many year percent capital economy force main minister small export mi	indian government state united india time first samoa south court national journal language people pacific

Table 6.6: CompareLDA Topics for Wikipedia Ranked by Alcohol Consumption.

Experiments show that it helps to uncover more conducive topics for assessing the relative merits between entities than baseline with pointwise supervision.

CHAPTER 7

CONCLUSION

In this study, we explored some components of a comparison mining system and expanded frontiers of opinion mining (Section 2.1) and comparison studies (Section 2.2). Their intersection shapes the comparison mining research (Section 2.3).

We discuss a new method for tackling the comparison identification problem in Chapter 3. We explore different ways to formulate this problem and determine relation extraction as the most suitable formulation. Within the relation extraction setup, we develop a new kernel-based approach for identifying comparative relations between entities, which relies on the syntactic structure of sentences. This method, called Skip-node, measures similarity of two sentences via their dependency tree representations. The novel kernel is shown to be empirically effective on the real-life datasets.

We propose Comparative Relation Generative Model (CompareGem) for the problem of comparison interpretation in Chapter 4. CompareGem connects two levels of comparison interpretation within a graphical model framework. At the individual comparison level it induces the meaning of each sentence or every comparison within a sentence. It also summarizes individual comparison interpretations to come up with an entity ranking for a whole input corpus. The interpretations are induced with respect to entity aspects and can be inferred in unsupervised or supervised fashion. Through the experiments, we demonstrate effectiveness of the proposed approach and show that the model reflects innate properties of comparison

corpora.

We propose an approach to discover preference groups within a population of rankers. We show that it is possible and productive to extract groups with disagreeing preferences. The approach is to use a graphical model called Plackett-Luce regressions mixture to cluster such latent preference groups (see Chapter 5). We demonstrate effectiveness of the model through the empirical evaluation.

To explain the comparisons made within a preference group, we use a topic modeling approach. If an entity is represented by its textual representation, then we show in Chapter 6, that for a corpus of entity documents, topics aligned with comparisons can be successfully induced. These topics then can be used to explain the comparison decisions. The approach we present is called CompareLDA, a specialized supervised topic model with pairwise comparison labels.

These four components form a pipeline for processing written opinions with comparisons. The pipeline starts from identifying comparisons in texts via the Skip-node kernel. Then it proceeds with interpreting these comparisons. The interpretation process is conducted at the level of individual comparisons and at the aggregated level. CompareGem is used to construct the both interpretations. When the comparisons of different users presumably are in disagreement, it is resolved and captured via the Plackett-Luce regression mixture model. Within one preference group, the comparison decisions made by users can be explained by CompareLDA.

Future Work This work outlines a backbone of comparison mining from text and provides initial research on the comparison mining components. It rather serves as the beginning of a long journey, than the conclusion. The doors are opened to explore different paradigms of solutions, for example, deep and representation learning. As the deep learning replaced a substantial amount of the language processing research, it is still limited in applications, where the data are scarce and clear interpretation of decisions is required. Comparison mining is one of these fields due to the rarity of comparisons and rankings in general. As the more and more data are available, these paradigms may appear to be plausible.

Another angle, which seems attractive for exploration is joint analysis of the comparison mining components. We refer to in-vivo evaluation of comparison mining components that solve a particular task, for instance, ranking electronic products based on their reviews. The concern here is the system architecture as a whole: how to design and select a system that is more suitable for this task.

Comparisons attract attention of various science fields, studies in linguistics focus on their grammatical representation [38, 182], studies in psychology address questions of choice evaluation [66]. Could computational ideas and techniques alter and assist the multidisciplinary research across the field? Formal linguistics theories of grammar may help to shape the mathematical model for identifying and interpreting comparisons. The result of computational comparison mining, in turn, can provide data to study affective and decision-making aspects of reviewers.

BIBLIOGRAPHY

[1] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749, 2005.

[2] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 3–14, 1995.

[3] Cédric Archambeau and Francois Caron. Plackett-Luce regression: A new Bayesian model for polychotomous data. In *Conference on Uncertainty in Artificial Intelligence (UAI'2012)*, 2012.

[4] Pranjal Awasthi, Avrim Blum, Or Sheffet, and Aravindan Vijayaraghavan. Learning mixtures of ranking models. In *Advances in Neural Information Processing Systems*, pages 2609–2617, 2014.

[5] Jay Ayres, Jason Flannick, Johannes Gehrke, and Tomi Yiu. Sequential pattern mining using a bitmap representation. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 429–435, 2002.

[6] Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, pages 2200–2204, 2010.

[7] D. Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[8] Michael A. Bender and Martin Farach-Colton. The lca problem revisited. In *Proceedings of the Latin American Symposium on Theoretical Informatics*, pages 88–94, 2000.

[9] James R. Bettman, Mary Frances Luce, and John W. Payne. Constructive consumer choice processes. *Journal of Consumer Research*, 25(3):187–217, 1998.

[10] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022, 2003.

[11] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[12] G. E. P. Box, J. S. Hunter, and W. G. Hunter. *Statistics for Experimenters*. Wiley, 2005.

[13] Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3):324–345, 1952.

[14] Samuel Brody and Noemie Elhadad. An unsupervised aspect-sentiment model for online reviews. In *NAACL HLT*, pages 804–812, 2010.

[15] Caroline Brun, Diana Nicoleta Popa, and Claude Roux. Xrce: Hybrid classification for aspect-based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 838–842, 2014.

[16] Razvan C. Bunescu and Raymond J. Mooney. A shortest path dependency kernel for relation extraction. In *Proceedings of the Conference on Human*

Language Technology and Empirical Methods in Natural Language Processing (HLT), pages 724–731, 2005.

[17] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *ICML*, pages 89–96, 2005.

[18] Erik Cambria. Affective computing and sentiment analysis. *IEEE Intelligent Systems*, 31(2):102–107, 2016.

[19] Erik Cambria, Björn Schuller, Yunqing Xia, and Catherine Havasi. New avenues in opinion mining and sentiment analysis. *IEEE Intelligent Systems*, 28(2):15–21, 2013.

[20] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *ICML*, pages 129–136, 2007.

[21] Francois Caron, Yee Whye Teh, Thomas Brendan Murphy, et al. Bayesian nonparametric Plackett-Luce models for the analysis of preferences for college degree programmes. *The Annals of Applied Statistics*, 8(2):1145–1181, 2014.

[22] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

[23] Chia-Hui Chang, Mohammed Kayed, Moheb R. Girgis, and Khaled F. Shaalan. A survey of web information extraction systems. *IEEE transactions on knowledge and data engineering*, 18(10):1411–1428, 2006.

[24] Jonathan Chang and David M. Blei. Relational topic models for document networks. In *AIStats*, pages 81–88, 2009.

[25] Jonathan Chang, Jordan Boyd-Graber, and David M. Blei. Connections between the lines: augmenting social networks with text. In *Proceedings of the*

15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 169–178, 2009.

[26] Danqi Chen and Christopher D. Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, 2014.

[27] Jose M. Chenlo and David E. Losada. An empirical study of sentence features for subjectivity and polarity classification. *Information Sciences*, 280(1):275–288, 2014.

[28] Michael Collins and Nigel Duffy. Convolution kernels for natural language. In *Advances in Neural Information Processing Systems (NIPS)*, pages 625–632, 2001.

[29] Michael Collins and Nigel Duffy. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (COLING)*, pages 263–270, 2002.

[30] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[31] Keith Cortis, André Freitas, Tobias Daudert, Manuela Huerlimann, Manel Zarrouk, Siegfried Handschuh, and Brian Davis. Semeval-2017 task 5: Fine-grained sentiment analysis on financial microblogs and news. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 519–535, 2017.

[32] Danilo Croce, Alessandro Moschitti, and Roberto Basili. Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1034–1046, 2011.

[33] Aron Culotta and Jeffrey Sorensen. Dependency tree kernels for relation extraction. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (COLING)*, page 423, 2004.

[34] Chad Cumby and Dan Roth. On kernel methods for relational learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 107–114, 2003.

[35] Roger R. Davidson. On extending the bradley-terry model to accommodate ties in paired comparison experiments. *Journal of the American Statistical Association*, 65(329):317–328, 1970.

[36] Xiaowen Ding, Bing Liu, and Lei Zhang. Entity discovery and assignment for opinion mining applications. In *KDD*, pages 1125–1134, 2009.

[37] Cicero dos Santos and Maira Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78, 2014.

[38] Ewan Dunbar and Alexis Wellwood. Addressing the'two interface'problem: Comparatives and superlatives. *Glossa: a journal of general linguistics*, 1(1):1, 2016.

[39] Cynthia Dwork, Ravi Kumar, Moni Naor, and Dandapani Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th international conference on World Wide Web*, pages 613–622, 2001.

[40] Arpad E. Elo. *The rating of chessplayers, past and present*. Arco Pub., 1978.

[41] Elena Erosheva, Stephen Fienberg, and John Lafferty. Mixed-membership models of scientific publications. *Proceedings of the National Academy of Sciences*, 101(1):5220–5227, 2004.

[42] Merijn Van Erp and Lambert Schomaker. Variants of the borda count method for combining ranked classifier hypotheses. In *Proceedings 7th International Workshop on frontiers in handwriting recognition (7th IWFHR)*, pages 443–452, 2000.

[43] Ronald Fagin, Ravi Kumar, and D. Sivakumar. Comparing top K lists. In *SIDMA*, pages 134–160, 2004.

[44] Yi Fang, Luo Si, Naveen Somasundaram, and Zhengtao Yu. Mining contrastive opinions on political texts using cross-perspective topic model. In *Proceedings of the 5th ACM International Conference on Web Search and Data Mining*, pages 63–72, 2012.

[45] Ronen Feldman, Moshe Fresko, Jacob Goldenberg, Oded Netzer, and Lyle Ungar. Extracting product comparisons from discussion boards. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 469–474, 2007.

[46] Basura Fernando, Efstratios Gavves, Damien Muselet, and Tinne Tuytelaars. Learning to rank based on subsequences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2785–2793, 2015.

[47] Michael A. Fligner and Joseph S. Verducci. Distance based ranking models. *Journal of the Royal Statistical Society. Series B (Methodological)*, 48(3):359–369, 1986.

[48] Brian Francis, Regina Dittrich, and Reinhold Hatzinger. Modeling heterogeneity in ranked responses by nonparametric maximum likelihood: How do Europeans get their scientific knowledge? *The Annals of Applied Statistics*, 4(4):2181–2202, 2010.

[49] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

[50] Mahak Gambhir and Vishal Gupta. Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review*, 47(1):1–66, 2017.

[51] Murthy Ganapathibhotla and Bing Liu. Mining opinions in comparative sentences. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 241–248, 2008.

[52] Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *TPAMI*, 6(6):721–741, 1984.

[53] Isobel Claire Gormley and Thomas Brendan Murphy. Exploring voting blocs within the Irish electorate: A mixture modeling approach. *Journal of the American Statistical Association*, 103(483):1014–1027, 2008.

[54] Isobel Claire Gormley and Thomas Brendan Murphy. Mixed membership models for rank data: Investigating structure in irish voting data. In *Airoldi, EM, Blei, DM, Erosheva, EA and Fienberg, SE (eds.). Handbook of Mixed Membership Models and Their Applications*, pages 441–460. 2014.

[55] Isobel Claire Gormley, Thomas Brendan Murphy, et al. A mixture of experts model for rank data with applications in election studies. *The Annals of Applied Statistics*, 2(4):1452–1477, 2008.

[56] Mihajlo Grbovic, Nemanja Djuric, Shengbo Guo, and Slobodan Vucetic. Supervised clustering of label ranking data using label preference information. *Machine Learning*, 93(2-3):191–225, 2013.

[57] John Guiver and Edward Snelson. Bayesian inference for Plackett-Luce ranking models. In *proceedings of the 26th annual international conference on machine learning*, pages 377–384, 2009.

[58] Zhen Hai, Peilin Zhao, Peng Cheng, Peng Yang, Xiaoli Li, and Guangxia Li. Deceptive review spam detection via exploiting task relatedness and unlabeled data. In *Proceedings of the 2016 Conference on Empirical Methods in*

Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016, pages 1817–1826, 2016.

[59] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, 2009.

[60] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[61] David Haussler. Convolution kernels on discrete structures. Technical report, Department of Computer Science, University of California at Santa Cruz, 1999.

[62] R. Herbrich, T. Graepel, and K. Obermayer. Support vector learning for ordinal regression. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, pages 97–102 vol.1, 1999.

[63] Ralf Herbrich, Tom Minka, and Thore Graepel. Trueskill: A bayesian skill rating system. In *NIPS*, pages 569–576, 2006.

[64] Feng Hou and Guo-Hui Li. Mining Chinese comparative sentences by semantic role labeling. In *International Conference on Machine Learning and Cybernetics*, pages 2563–2568, 2008.

[65] Chrisopher K. Hsee, George F. Loewenstein, Sally Blount, and Max H. Bazerman. Preference reversals between joint and separate evaluations of options: a review and theoretical analysis. *Psychological bulletin*, 125(5):576, 1999.

[66] Christopher K. Hsee and Jiao Zhang. Distinction bias: Misprediction and mischoice due to joint evaluation. *Journal of personality and social psychology*, 86(5):680, 2004.

[67] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 168–177, 2004.

[68] Minqing Hu and Bing Liu. Mining opinion features in customer reviews. In *AAAI*, pages 755–760, 2004.

[69] Sheng Huang, Xinlan Liu, Xueping Peng, and Zhendong Niu. Fine-grained product features extraction and categorization in reviews opinion mining. In *IEEE 12th Int'l Conf. Data Mining Workshops*, pages 680–686, 2012.

[70] Xiaojiang Huang, Xiaojun Wan, Jianwu Yang, and Jianguo Xiao. Learning to identify comparative sentences in Chinese text. In *Pacific Rim International Conference on Artificial Intelligence*, pages 187–198, 2008.

[71] David R. Hunter et al. Mm algorithms for generalized bradley-terry models. *The annals of statistics*, 32(1):384–406, 2004.

[72] Alpa Jain and Patrick Pantel. How do they compare? automatic identification of comparable entities on the web. In *Information Reuse and Integration (IRI), 2011 IEEE International Conference on*, pages 228–233, 2011.

[73] Niklas Jakob and Iryna Gurevych. Extracting opinion targets in a single- and cross-domain setting with conditional random fields. In *EMNLP*, pages 1035–1045, 2010.

[74] Myungha Jang, Jin woo Park, and Seung won Hwang. Predictive mining of comparable entities from the web. In *AAAI Conf. Artificial Intelligence (AAAI 12)*, pages 66–72, 2012.

[75] Xin Jiang, Yunhua Hu, and Hang Li. A ranking approach to keyphrase extraction. In *SIGIR*, pages 756–757, 2009.

[76] Nitin Jindal and Bing Liu. Identifying comparative sentences in text documents. In *Proceedings of the International ACM SIGIR Conference on Re-*

search and Development in Information Retrieval (SIGIR), pages 244–251, 2006.

[77] Nitin Jindal and Bing Liu. Mining comparative sentences and relations. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 1331–1336, 2006.

[78] Nitin Jindal and Bing Liu. Opinion spam and analysis. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 219–230. ACM, 2008.

[79] Thorsten Joachims. Making large-scale support vector machine learning practical. In *Advances in Kernel Methods: Support Vector Learning*, pages 169–184. 1999.

[80] Thorsten Joachims. Optimizing search engines using clickthrough data. In *KDD*, pages 133–142, 2002.

[81] Dan Jurafsky, Victor Chahuneau, Bryan R. Routledge, and Noah A. Smith. Narrative framing of consumer sentiment in online restaurant reviews. *First Monday*, 19(4), 2014.

[82] Alexandros Karatzoglou, Linas Baltrunas, and Yue Shi. Learning to rank for recommender systems. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 493–494, 2013.

[83] L. Kaushik, A. Sangwan, and J. H. L. Hansen. Sentiment extraction from natural audio streams. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8485–8489, 2013.

[84] Wiltrud Kessler and Jonas Kuhn. Detection of product comparisons - how far does an out-of-the-box semantic role labeling system take you? In *EMNLP*, pages 1892–1897, 2013.

[85] Wiltrud Kessler and Jonas Kuhn. Detection of product comparisons-how far does an out-of-the-box semantic role labeling system take you? In *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*, pages 1892–1897, 2013.

[86] Wiltrud Kessler and Jonas Kuhn. A corpus of comparisons in product reviews. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 2242–2248, 2014.

[87] Wiltrud Kessler and Jonas Kuhn. Detecting comparative sentiment expressions – a case study in annotation design decisions. In *Proceedings of Konferenz zur Verarbeitung Nat?rlicher Sprache (KONVENS)*, pages 165–170, 2014.

[88] Hannah Kim, Jaegul Choo, Jingu Kim, Chandan K. Reddy, and Haesun Park. Simultaneous discovery of common and discriminative topics via joint nonnegative matrix factorization. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 567–576, 2015.

[89] Hyun Duk Kim and ChengXiang Zhai. Generating comparative summaries of contradictory opinions in text. In *CIKM*, pages 385–394, 2009.

[90] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[91] Takeshi Kurashima, Katsuji Bessho, Hiroyuki Toda, Toshio Uchiyama, and Ryoji Kataoka. Ranking entities using comparative relations. In *Database and Expert Systems Applications (DEXA)*, pages 124–133, 2008.

[92] Simon Lacoste-Julien, Fei Sha, and Michael I. Jordan. DiscLDA: Discriminative learning for dimensionality reduction and classification. In *Advances in neural information processing systems*, pages 897–904, 2009.

[93] Theodoros Lappas, George Valkanas, and Dimitrios Gunopulos. Efficient and domain-invariant competitor mining. In *KDD*, pages 408–416, 2012.

[94] Paul H. Lee and LH Philip. Mixtures of weighted distance-based models for ranking data with applications in political studies. *Computational Statistics & Data Analysis*, 56(8):2486–2500, 2012.

[95] Kevin Lerman and Ryan McDonald. Contrastive summarization: an experiment with consumer reviews. In *Proceedings of human language technologies: The 2009 annual conference of the North American chapter of the association for computational linguistics, companion volume: Short papers*, pages 113–116, 2009.

[96] Shasha Li, Chin-Yew Lin, Young-In Song, and Zhoujun Li. Comparable entity mining from comparative questions. *IEEE transactions on knowledge and data engineering*, 25(7):1498–1509, 2013.

[97] Si Li, Zheng-Jun Zha, Zhaoyan Ming, Meng Wang, Tat-Seng Chua, Jun Guo, and Weiran Xu. Product comparison using comparative relations. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 1151–1152, 2011.

[98] Chen Lin, Timothy Miller, Alvin Kho, Steven Bethard, Dmitriy Dligach, Sameer Pradhan, and Guergana Savova. Descending-path convolution kernel for syntactic structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 81–86, 2014.

[99] Dekang Lin and Patrick Pantel. Discovery of inference rules for question-answering. *Natural Language Engineering*, 7(04):343–360, 2001.

[100] Bin Liu, Junjie Chen, and Xiaolong Wang. Application of learning to rank to protein remote homology detection. *Bioinformatics*, 31(21):3492–3498, 2015.

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

[102] Bing Liu and Lei Zhang. A survey of opinion mining and sentiment analysis. In *Mining Text Data*, pages 415–463. 2012.

[103] Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1-3):503–528, 1989.

[104] Nathan N. Liu, Min Zhao, and Qiang Yang. Probabilistic latent preference analysis for collaborative filtering. In *CIKM*, pages 759–766, 2009.

[105] Tie-Yan Liu et al. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, 2009.

[106] R. Ducan Luce. *Individual Choice Behavior a Theoretical Analysis*. John Wiley and Sons, 1959.

[107] R Duncan Luce. The choice axiom after twenty years. *Journal of Mathematical Psychology*, 15(3):215–233, 1977.

[108] R Duncan Luce. *Individual choice behavior: A theoretical analysis*. Courier Dover Publications, 2012.

[109] Yuanhua Lv, Taesup Moon, Pranam Kolari, Zhaohui Zheng, Xuanhui Wang, and Yi Chang. Learning to model relatedness for news recommendation. In *WWW*, pages 57–66, 2011.

[110] Colin L. Mallows. Non-null ranking models. i. *Biometrika*, 44(1/2):114–130, 1957.

[111] John I. Marden. *Analyzing and modeling rank data*. CRC Press, 1996.

[112] Julian McAuley, Rahul Pandey, and Jure Leskovec. Inferring networks of substitutable and complementary products. In *KDD*, pages 785–794, 2015.

[113] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes. In *SIGIR*, pages 43–52, 2015.

[114] Jon D. McAuliffe and David M. Blei. Supervised topic models. In *Advances in Neural Information Processing Systems*, pages 121–128, 2008.

[115] Andrew McCallum, Kamal Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, pages 41–48, 1998.

[116] Kathleen McKeown and Dragomir R. Radev. Generating summaries of multiple news articles. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 74–82, 1995.

[117] Qiaozhu Mei, Deng Cai, Duo Zhang, and ChengXiang Zhai. Topic modeling with network regularization. In *Proceedings of the 17th International Conference on World Wide Web*, pages 101–110, 2008.

[118] Marina Meilă and Harr Chen. Dirichlet process mixtures of generalized mallows models. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, pages 358–367, 2010.

[119] Donald Metzler and W Bruce Croft. Linear feature-based models for information retrieval. *Information Retrieval*, 10(3):257–274, 2007.

[120] Donald Metzler and Tapas Kanungo. Machine learned sentence selection strategies for query-biased summarization. In *SIGIR Learning to Rank Workshop*, pages 40–47, 2008.

[121] Saif M. Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. Semeval-2018 Task 1: Affect in tweets. In *Proceedings of International Workshop on Semantic Evaluation (SemEval-2018)*, pages 1–17, 2018.

[122] Burt L. Monroe, Michael P. Colaresi, and Kevin M. Quinn. Fightin' words: Lexical feature selection and evaluation for identifying the content of political conflict. *Political Analysis*, 16(4):372–403, 2008.

[123] Louis-Philippe Morency, Rada Mihalcea, and Payal Doshi. Towards multimodal sentiment analysis: Harvesting opinions from the web. In *Proceedings of the 13th International Conference on Multimodal Interfaces*, pages 169–176, 2011.

[124] Alessandro Moschitti. Efficient convolution kernels for dependency and constituent syntactic trees. In *European Conference on Machine Learning (ECML)*, pages 318–329, 2006.

[125] Alessandro Moschitti. Making tree kernels practical for natural language learning. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 113–120, 2006.

[126] Arjun Mukherjee and Bing Liu. Aspect extraction through semi-supervised modeling. In *ACL*, pages 339–348, 2012.

[127] Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. Dependency tree-based sentiment classification using crfs with hidden variables. In *Human Language Technologies: Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 786–794, 2010.

[128] Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. Semeval-2016 task 4: Sentiment analysis in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1–18, 2016.

[129] Rohit Narayan, Jitendra Kumar Rout, and Sanjay Kumar Jena. Review spam detection using semi-supervised technique. In *Progress in Intelligent*

Computing Techniques: Theory, Practice, and Applications, pages 281–286. Springer, 2018.

[130] Truc-Vien T. Nguyen, Alessandro Moschitti, and Giuseppe Riccardi. Convolution kernels on constituent, dependency and sequential structures for relation extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1378–1387, 2009.

[131] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2-3):103–134, 2000.

[132] Joakim Nivre. Dependency grammar and dependency parsing. *MSI report*, 5133(1959):1–32, 2005.

[133] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[134] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124, 2005.

[135] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135, 2008.

[136] Devi Parikh and Kristen Grauman. Relative attributes. In *ICCV*, pages 503–510, 2011.

[137] Dae Hoon Park and Catherine Blake. Identifying comparative claim sentences in full-text scientific articles. In *Proceedings of the Workshop on Detecting Structure in Scholarly Discourse*, pages 1–9, 2012.

[138] Michael Paul and Roxana Girju. A two-dimensional topic-aspect model for discovering multi-faceted topics. In *AAAI*, pages 545–550, 2010.

[139] Michael J. Paul, ChengXiang Zhai, and Roxana Girju. Summarizing contrastive viewpoints in opinionated text. In *EMNLP*, pages 66–76, 2010.

[140] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 0215–0215, 2001.

[141] Robin L. Plackett. The analysis of permutations. *Applied Statistics*, 24(2):193–202, 1975.

[142] Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, AL-Smadi Mohammad, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, et al. Semeval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, pages 19–30, 2016.

[143] Ana-Maria Popescu and Orena Etzioni. Extracting product features and opinions from reviews. In *Natural Language Processing and Text Mining*, pages 9–28. 2007.

[144] S. Poria, E. Cambria, A. Gelbukh, F. Bisio, and A. Hussain. Sentiment data flow analysis by means of dynamic linguistic patterns. *IEEE Computational Intelligence Magazine*, 10(4):26–36, 2015.

[145] Soujanya Poria, Erik Cambria, and Alexander Gelbukh. Aspect extraction for opinion mining with a deep convolutional neural network. *Know.-Based Syst.*, 108(C):42–49, 2016.

[146] Soujanya Poria, Erik Cambria, and Alexander Gelbukh. Aspect extraction for opinion mining with a deep convolutional neural network. *Knowledge-Based Systems*, 108(1):42–49, 2016.

[147] Soujanya Poria, Erik Cambria, Gregoire Winterstein, and Guang-Bin Huang. Sentic patterns: Dependency-based rules for concept-level sentiment analysis. *Knowledge-Based Systems*, 69(1):45–63, 2014.

[148] Vasin Punyakanok, Dan Roth, and Wen tau Yih. Natural language inference via dependency tree mapping: An application to question answering. Technical report, University of Illinois at Urbana-Champaign, 2004.

[149] Tao Qin, Xiubo Geng, and Tie-Yan Liu. A new probabilistic model for rank aggregation. In *Advances in neural information processing systems*, pages 1948–1956, 2010.

[150] Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D. Manning. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 248–256, 2009.

[151] Daniel Ramage, Paul Heymann, Christopher D. Manning, and Hector Garcia-Molina. Clustering the tagged web. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 54–63, 2009.

[152] PV Rao and Lawrence L. Kupper. Ties in paired-comparison experiments: A generalization of the bradley-terry model. *Journal of the American Statistical Association*, 62(317):194–204, 1967.

[153] George Rasch. *Probabilistic Models for Some Intelligence and Attainment Tests*. University of Chicago Press, 1981.

[154] Kumar Ravi and Vadlamani Ravi. A survey on opinion mining and sentiment analysis: tasks, approaches and applications. *Knowledge-Based Systems*, 89(1):14–46, 2015.

[155] Yafeng Ren and Donghong Ji. Neural networks for deceptive opinion spam detection: An empirical study. *Information Sciences*, 385:213–224, 2017.

[156] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.

[157] Sara Rosenthal, Noura Farra, and Preslav Nakov. Semeval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 502–518, 2017.

[158] Beatrice Santorini. Part-of-speech tagging guidelines for the penn treebank project (3rd revision). Technical report, University of Pennsylvania, 1990.

[159] Libin Shen, Anoop Sarkar, and Franz Josef Och. Discriminative reranking for machine translation. In *HLT-NAACL 2004: Main Proceedings*, pages 177–184, 2004.

[160] Yue Shi, Martha Larson, and Alan Hanjalic. List-wise learning to rank with matrix factorization for collaborative filtering. In *RecSys*, pages 269–272, 2010.

[161] Ruben Sipos and Thorsten Joachims. Generating comparative summaries from reviews. In *CIKM*, pages 1853–1856, 2013.

[162] Alex J. Smola and SVN Vishwanathan. Fast kernels for string and tree matching. In *Advances in neural information processing systems*, pages 585–592, 2003.

[163] Shashank Srivastava, Dirk Hovy, and Eduard Hovy. A walk-based semantically enriched tree kernel over distributed word representations. In *Pro-*

ceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1411–1416, 2013.

[164] Ingo Steinwart and Andreas Christmann. *Support vector machines*. Springer, 2008.

[165] L. I. Tan, W. S. Phang, K. O. Chin, and P. Anthony. Rule-based sentiment analysis for financial news. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1601–1606, 2015.

[166] Louis L. Thurstone. The method of paired comparisons for social values. *The Journal of Abnormal and Social Psychology*, 21(4):384, 1927.

[167] Ivan Titov and Ryan McDonald. Modeling online reviews with multi-grain topic models. In *Proceedings of the 17th international conference on World Wide Web*, pages 111–120, 2008.

[168] Maksim Tkachenko and Hady W. Lauw. Generative modeling of entity comparisons in text. In *CIKM*, pages 859–868, 2014.

[169] Maksim Tkachenko and Hady W. Lauw. A convolution kernel approach to identifying comparisons in text. In *ACL-IJCNLP*, pages 376–386, 2015.

[170] Maksim Tkachenko and Hady W. Lauw. Plackett-luce regression mixture model for heterogeneous rankings. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 237–246, 2016.

[171] Maksim Tkachenko and Hady W. Lauw. Comparative relation generative model. *IEEE Transactions on Knowledge and Data Engineering*, 29(4):771–783, 2017.

[172] Maksim Tkachenko and Hady W. Lauw. Comparelda: A topic model for document comparison. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, 2019.

[173] Quoc-Tuan Truong and Hady W. Lauw. Visual sentiment analysis for review images with item-oriented and user-oriented cnn. In *Proceedings of the 2017 ACM on Multimedia Conference*, pages 1274–1282, 2017.

[174] Hamed Valizadegan, Rong Jin, Ruofei Zhang, and Jianchang Mao. Learning to rank by optimizing ndcg measure. In *Advances in neural information processing systems*, pages 1883–1891, 2009.

[175] Rik van Noord and Johan Bos. Dealing with co-reference in neural semantic parsing. In *Proceedings of the 2nd Workshop on Semantic Deep Learning (SemDeep-2)*, pages 41–49, 2017.

[176] Maksims N. Volkovs, Hugo Larochelle, and Richard S. Zemel. Learning to rank by aggregating expert preferences. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 843–851, 2012.

[177] Maksims N. Volkovs and Richard S. Zemel. A flexible generative model for preference aggregation. In *Proceedings of the 21st international conference on World Wide Web*, pages 479–488, 2012.

[178] Hanna M. Wallach, Iain Murray, Ruslan Salakhutdinov, and David Mimno. Evaluation methods for topic models. In *Proceedings of the 26th annual international conference on machine learning*, pages 1105–1112, 2009.

[179] Chong Wang and David M. Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 448–456, 2011.

[180] Chris Watkins. Dynamic alignment kernels. In *Advances in Large-Margin Classifiers*, pages 39–50. 1999.

[181] Markus Weimer, Alexandros Karatzoglou, Quoc V. Le, and Alex J. Smola. Cofi rank - maximum margin matrix factorization for collaborative ranking.

In *Advances in Neural Information Processing Systems 20*, pages 1593–1600. 2008.

[182] Alexis Wellwood. On the semantics of comparison across categories. *Linguistics and Philosophy*, 38(1):67–101, 2015.

[183] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pages 1192–1199, 2008.

[184] Kaiquan Xu, Stephen Shaoyi Liao, Raymond Y. K. Lau, Heng Tang, and Shanshan Wang. Building comparative product relation maps by mining consumer opinions on the web. In *AMCIS*, page 179, 2009.

[185] Linli Xu, Aiqing Huang, Jianhui Chen, and Enhong Chen. Exploiting task-feature co-clusters in multi-task learning. In *AAAI*, pages 1931–1937, 2015.

[186] Seon Yang and Youngjoong Ko. Extracting comparative sentences from Korean text documents using comparative lexical patterns and machine learning techniques. In *Proceedings of the ACL-IJCNLP Conference Short Papers*, pages 153–156, 2009.

[187] Weiwei Yang, Jordan Boyd-Graber, and Philip Resnik. A discriminative topic model using document network structure. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 686–696, 2016.

[188] Yang Yang, Jie Tang, Jacklyne Keomany, Yanting Zhao, Juanzi Li, Ying Ding, Tian Li, and Liangwei Wang. Mining competitive relationships by learning across heterogeneous networks. In *CIKM*, pages 1432–1441, 2012.

[189] Zichao Yang, Zhiting Hu, Yuntian Deng, Chris Dyer, and Alex Smola. Neural machine translation with recurrent attention modeling. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 383–387, 2017.

[190] Alexander Yeh. More accurate tests for the statistical significance of result differences. In *Proceedings of the Conference on Computational Linguistics (COLING)*, pages 947–953, 2000.

[191] ChengXiang Zhai, Atulya Velivelli, and Bei Yu. A cross-collection mixture model for comparative text mining. In *KDD*, pages 743–748, 2004.

[192] Runxiang Zhang and Yaohong Jin. Identification and transformation of comparative sentences in patent Chinese-English machine translation. In *International Conference on Asian Language Processing (IALP)*, pages 217–220, 2012.

[193] Zhu Zhang, Chenhui Guo, and Paulo Goes. Product comparison networks for competitive analysis of online word-of-mouth. *ACM Transactions on Management Information Systems (TMIS)*, 3(4):20, 2013.

[194] Zhibing Zhao, Peter Piech, and Lirong Xia. Learning mixtures of Plackett-Luce models. In *International Conference on Machine Learning*, pages 2906–2914, 2016.

[195] Zhaohui Zheng, Keke Chen, Gordon Sun, and Hongyuan Zha. A regression framework for learning ranking functions using relative relevance judgments. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 287–294, 2007.

[196] GuoDong Zhou, Min Zhang, Dong Hong Ji, and Qiaoming Zhu. Tree kernel-based relation extraction with context-sensitive structured parse tree information. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 728–736, 2007.

[197] Jun Zhu, Amr Ahmed, and Eric P. Xing. MedLDA: maximum margin supervised topic models. *Journal of Machine Learning Research*, 13(Aug):2237–2278, 2012.