Unsupervised Context Extraction for Review-Based Recommendations

by

Padipat SITKRONGWONG

Dissertation

submitted to the Department of Informatics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

The Graduate University for Advanced Studies, SOKENDAI

September 2020

Committee

Advisor	Dr. Atsuhiro TAKASU
	Professor of National Institute of Informatics/SOKENDAI
Examiner	Dr. Akiko AIZAWA
	Professor of National Institute of Informatics/SOKENDAI
Examiner	Dr. Seiji YAMADA
	Professor of National Institute of Informatics/SOKENDAI
Examiner	Dr. Kenro AIHARA
	Associate Professor of National Institute of Informatics/SOKENDAI
Examiner	Dr. Saranya MANEEROJ
	Associate Professor of Chulalongkorn University, Bangkok, Thailand

Acknowledgements

I would like to thank the people who helped and supported me during my Ph.D. studies.

Firstly, I would like to express my sincere gratitude to my advisor Professor Takasu Atsuhiro for the continuous support of my Ph.D study and related research, for his patience, motivation, and immense knowledge.

I want to thank all the members and internship students in Takasu Laboratory, especially Takenaka Akiko, for providing a motivating, fun, and positive working environment. Also, thanks to the people in NII for all the support and fruitful discussions about both research and life in Japan.

Abstract

Recommender systems were devised to provide personalized recommendations on suitable items that would match the individual users' interests. The standard recommendation approach relies on user preferences on items, such as the numeric ratings, for building a predictive model for future rating prediction. Utilizing only the rating data, however, might not be sufficient to overcome two common challenges in recommender systems: the prediction accuracy, and the rating sparsity. Whereas contextual information has often been utilized to improve the prediction accuracy, user-generated reviews have been recognized as valuable sources to alleviate the rating sparsity. This thesis studies extensively the challenges on extracting contexts from reviews, and how to utilize such extracted contexts for personalizing recommendations that would satisfy both accuracy and sparsity. In addition to contexts and reviews, this thesis also studies on how to improve the prediction accuracy with multi-criteria ratings. These studies result in three main proposed methods as follows.

First, the context-aware region embedding (CARE), a novel unsupervised method for defining and extracting contexts from reviews, is proposed. CARE deals with the problems of obtaining and identifying relevant contexts in a standard context-aware recommendations by applying region embedding techniques to extract and represent contexts from reviews. Such relevant contexts are represented as text regions that influence the distributions of ratings. The experiments demonstrated that CARE has a flexibility to extract contexts from review data in any recommendation domains. Moreover, the extracted contexts effectively captured the polarity of reviews' ratings, which can be useful for the rating prediction task.

Next, the attentional interaction model for context-aware region embedding (CARE-AI), an extended model of CARE for rating prediction is proposed. CARE-AI aims to overcome the challenges of learning user and item representations from reviews of the recent review-based recommendations. This model introduces the interaction and attention modules for constructing the user and item representations from the extracted contexts, which are derived from CARE. Such representations are generated specifically for each particular review, based on the different level of relevance among contexts in that review. Extensive experiments show that, not only achieving better prediction accuracy compared to the state-of-the-art review-based recommendations, CARE-AI was also more robust in generating recommendations in the rating sparsity situations.

Finally, a novel method for multi-criteria rating conversion is proposed. This method aims to overcome the limitation of applying a standard rating conversion techniques to multi-criteria recommendations, which might cause a loss in implicit relation among multi-criteria ratings. The proposed method applies the principle component analysis to extract the multi-criteria rating patterns from users, which are then used to convert all multi-criteria ratings simultaneously to maintain their implicit relation. The experiments demonstrated that the proposed method outperforms both single and multi-criteria rating conversion techniques with higher accuracy and prediction coverage.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Motivation

Recommender system (RS) was invented to provide a personalized recommendation on the suitable items that would be interested by an individual user. A standard RS approach such as collaborative filtering utilizes past interactions from users (e.g. their numeric ratings on items) to build a predictive model for future recommendations on unseen items. However, considering only rating data might not be sufficient to overcome two common challenges that need to be addressed in order to produce effective recommendations: how to improve the prediction accuracy, and how to alleviate the rating sparsity. First, the success of the recommendation engines are often measured by their prediction accuracy. The highly accurate recommendations would increase user satisfactions on the systems, and consequently increases their revenues. On the other hand, the rating sparsity occurs when most users only rate few portion of all available items. Having insufficient amount of ratings, the recommendation engines would not be able to learn high-quality preference patterns from users, and therefore, result in low recommendation accuracy.

In order to improve recommendation accuracy, a context-aware recommendation approach has been introduced. In recommender systems, the contextual information or simply "context" is defined as any information that influences the users' decisions when they are choosing items. For example, most users tend to travel to the beaches in summer, whereas the ski-resorts are more preferred in winter. Incorporating contexts could help suggesting more relevant items to users, which can be a crucial factor for producing more accurate recommendations than the standard rating-only recommendation methods. However, there are two main concerns that could have significant impact on the performances of context-aware recommendations. First, obtaining contexts is not a trivial task. Many works collected them by first predefining a list of contexts, and then ask users to select from those values as their contexts at the time they consumed items. Predefining optimal values of contexts, however, could potentially be very expensive since their values are varied across different recommendation domains. After obtaining contexts, the next challenge is to identify which contexts are relevant to a specific recommendation task. Incorporating too many contexts not only degrades the quality of recommendations, but also increases dimension of data and thus triggering a sparsity problem.

On the other hand, a review-based recommendation approach has been proposed with the main goal to alleviate rating sparsity problem occurred in standard RS by utilizing user-generated reviews. In many systems, users have options to write text reviews on products or services they have purchased, in addition to the ratings. In reviews, users can provide comments explaining reasons behind their decisions on items, which offer more meaningful and useful information than numeric ratings. Many review-based recommendation methods take this opportunity to effectively extract the user personal preferences and item unique features from reviews in the form of numeric user and item representation vectors. Recent works in review-based recommendations proved that, learning such representations from reviews and use them for rating prediction, is more robust to rating sparsity than learning them from ratings alone [19, 47, 64, 86, 103].

Furthermore, in reviews users can express opinions describing their experiences, situations, and and satisfaction on their selected items, which can be a valuable source of contexts [21]. Extracting contexts from reviews could be the key to overcome the challenge of obtaining contexts, and provides recommendations with high accuracy as well as robustness to rating sparsity. Moreover, since contexts are information that affects users' decisions on items, they might be useful in constructing high-quality user and item representations for review-based recommendations.

Finally, in addition to a contextual information and user-generated reviews, this thesis also studies another type of data, the multi-criteria ratings. Unlike a standard RS approach that considers only single overall ratings from users, the multi-criteria recommendation approach lets users express their preferences toward items in multiple aspects, such as service or cleanliness of hotels, with multi-criteria ratings. Effectively utilizing such multi-criteria ratings could help analyzing user preferences and item features in more details, and yielding even more accurate recommendations.

1.2 Problem Definition

This thesis concerns three main problems: how to extract contexts from reviews, how to construct user and item representations for rating prediction, and how to effectively utilize multi-criteria ratings for making recommendations.

1.2.1 Extracting contexts from reviews

Recently, many works in context-aware recommendations tried to extract contexts from user reviews based on supervised and unsupervised approach [13, 20, 38, 58, 75, 101]. The supervised approach [20, 38, 58] extracts contexts as words that corresponds to the predefined values, which makes them encounter the same challenge of predefining optimal values for each context. On the other hand, the unsupervised approach [13, 75, 101] automatically extract contexts from reviews without having to predefine them, which make this approach widely applicable for various recommendation domains. However, unlike the traditional context-aware recommendation techniques that utilize predefined type of contexts, most context extraction methods lack the process of identifying the relevant contexts for each specific domains. Furthermore, contexts extracted from most methods have been restricted in a single word format, e.g. “family” or “breakfast”. In fact, the actual meaning of contexts might be involved with more than one word, such as “family trip” or “continental breakfast.”

1.2.2 Constructing user and item representations

Recent works in review-based recommendations employed deep learning techniques to learn representations of users and items from reviews, and use them for rating prediction [19, 47, 86, 103]. These methods, however, share two similar principles that could potentially be their limitations. First, they take into account every word in a review as an input to learn the user and item representations. In fact, some words might not be related to user preferences or item features, and therefore should not be taken into account when modeling their representations. Moreover, the user and item representations are constructed in a static manner by aggregating all of their corresponding past reviews. To predict a rating for a particular review, it is more important to concentrate and leverage more relevant information embedded in that review for modeling a user preferences and item features, with respect to their current situations. By effectively extracting contexts from reviews, they might be helpful in constructing more meaningful representations that help overcoming these two limitations of review-based recommendations.

1.2.3 Utilizing multi-criteria ratings

Most multi-criteria recommendation techniques are derived from the collaborative filtering-based recommendation approach, which utilize the ratings from neighbors for making rating predictions. However, one concern is that there might be a bias in a pattern of providing rating from each individual user. For example, one user might always rate items with high ratings, whereas the other user might always rate item with low ratings. Exploiting ratings from the neighbors without concerning such biases might deteriorate the prediction accuracy. By taking into account the user preference biases, the rating conversion techniques have been introduced. However, those techniques only applicable to a single criterion recommendation approach. For multi-criteria recommendation approach, converting each criterion rating independently might result in loss of implicit relation among criteria ratings.

1.3 Objective

According to the problem definition, this thesis is proposed under the following three main objectives.

- **Unsupervised context extraction from reviews**: contexts should be extracted from reviews based in unsupervised way, which allows the model to avoid predefining optimal contextual values—making it applicable on any kind of review dataset. In addition, a model should include the process of identifying relevant contexts, to make sure that only those affect the users' ratings on items will be used to create a predictive model. Finally, an extracted context should not be restricted in a single word format. In many occasion, users might express their contexts involving more than one word, extracting them appropriately could make a process of identifying relevant contexts more effective.

- **Dynamic user and item representations from contexts**: to make an effective rating prediction, a user and item representations should be generated with respect to these following properties. First, they should be constructed focusing on words that are related to contexts, which are the ones that affects users' rating patterns on items. Moreover, the user and item representations should be dynamically constructed for each particular review, to effectively capture the contexts embedded in that review. Finally, to make more personalized rating prediction, it is important to model the relevance of each context to each individual user preferences and each specific item features, which consequently influences the rating.

- **Multi-criteria rating conversion**: to exploit the multi-criteria ratings from the neighbors for rating prediction, such ratings should be firstly converted in an active user preference scale, to prevent the biases in rating patterns. More importantly, every criterion rating should be converted simultaneously, to maintain the implicit relation among the other criteria ratings.

1.4 Contribution

This thesis is composed of three main works for achieving each previously mentioned objective.

1.4.1 Unsupervised Context Extraction via Region Embedding for Context Aware Recommendation

First, a novel unsupervised method for defining and extracting contexts from reviews, namely the context-aware region embedding (CARE), is proposed [91]. Unlike any previous work, a relevant context in this work is defined and extracted not only in the single word format, but also includes its adjacent or nonadjacent words that occupy in the same text region and having significant influences on the distributions of ratings. A region embedding technique [77] is derived to emphasize the words in a small text region to be considered as a context, and represent it by a region embedding to be used for a rating prediction. By not having to predefine contexts, CARE has a flexibility to extract relevant contexts from any specific recommendation domain.

The extensive experiments show that CARE is able to extract the set of unique contexts from any specific recommendation domain. Moreover, the extracted contexts effectively explain the distribution of ratings in reviews, which is useful for modeling the polarity of the reviews' ratings.

1.4.2 Context-Aware User and Item Representations Based on Unsupervised Context Extraction from Reviews

This work is extended from CARE by utilizing the contexts extracted from the extraction method for personalizing the rating prediction [92]. Specifically, an efficient method for constructing a user and item representations based on the extracted contexts, namely the attentional interaction model for context-aware region embedding (CARE-AI), is proposed. This model introduces the interaction and attention modules, which help constructing a user and item representations based on different levels of relevance among the extracted contexts with an individual user preferences and item features. Unlike most deep learning-based methods that learn one static user or item representation for all reviews, CARE-AI dynamically generates a unique user and item representations for each particular review, which are more proper for capturing the specific contextual information embeded in that review.

The experiments demonstrate that CARE-AI performs better than state-of-the-art rating prediction methods including review-based and context-aware recommendation techniques. In addition, the effectiveness of utilizing the proposed interaction and attention modules, the impact of the model parameters, the impact of review quality, as well as the performance on the rating sparsity situations, are analyzed in detail.

1.4.3 Multi-Criteria Rating Conversion Without Relation Loss For Recommender Systems

Finally, a novel method for simultaneously converting multi-criteria ratings between users is proposed [90]. The multi-criteria ratings are first normalized by variances of users and principle component analysis is applied to extract user preference patterns. Such patterns are then used for multi-criteria rating conversion, which preserves the implicit relation among criteria ratings.

The experiment results show that proposed multi-criteria rating conversion technique outperforms both current single and multi-criteria rating conversion techniques with higher accuracy on TripAdvisor and Yahoo datasets respectively, while maintaining considerably high level of prediction coverages

1.5 Thesis Organization

The remaining chapters of this thesis are organized as follows. Chapter 2 introduces the background of recommender systems, including review-based and context-aware, and multi-criteria recommendations. Chapter 3 presents the related techniques to this work, including the review-based representation techniques, the context extraction techniques, and the rating conversion techniques. Chapter 4 explains the first work, the Unsupervised Context Extraction via Region Embedding for Context Aware Recommendation. Chapter 5 then presents the extension, the Context-Aware User and Item Representations Based on Unsupervised Context Extraction from Reviews. Chapter 6 presents the Multi-Criteria Rating Conversion without Relation Loss For Recommender Systems. Chapter 7 finally summarizes the thesis.

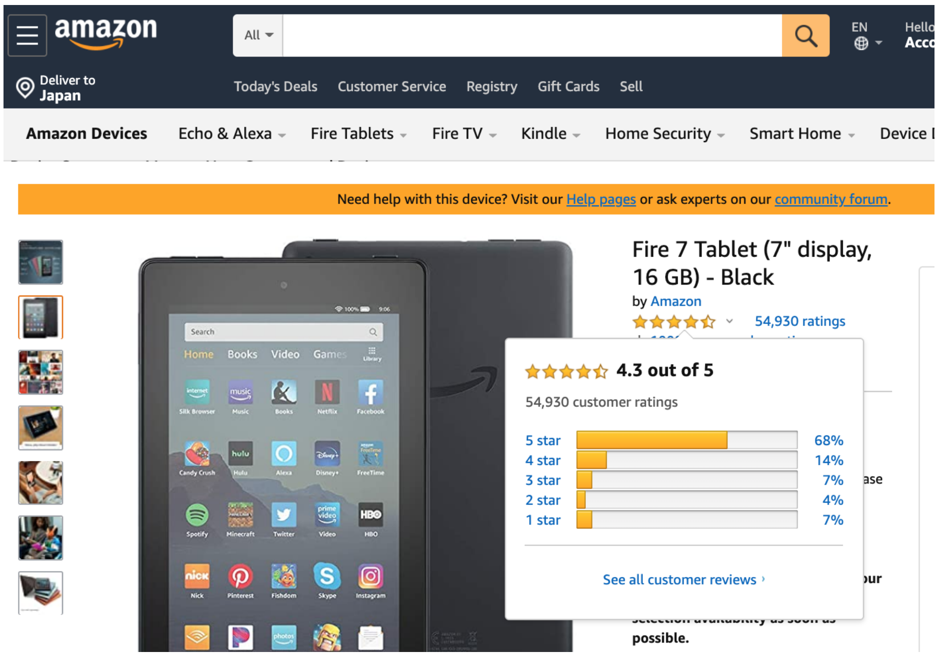
Chapter 2

Background

2.1 Recommender Systems

The rapid growth of an Internet of Things (IoT) in past decades leads us to an era where online information is easily accessible anywhere and anytime. Many online retailers and service providers take this opportunity to provide a variety of products and services to their customers via online websites and mobile applications. The e-commerce consumers, on the other hand, are often overwhelmed by the unlimited amount of products and services available for them to make a selection. Offering too many choices, however, could consequently lead to an information overload problem since most of them might not match with a unique interest of each individual consumer. Motivated by this cause, a recommender system (RS) was invented with one main objective—to provide a personalized recommendation on the suitable items that would be interested by an individual user.

Most e-commerce systems rely on users' past interactions with the systems as an input for producing a personalized recommendation. Such interactions can come in various formats, such as views, clicks, likes, shares, but the most common form is the numeric ratings. Figure 2.1, for example, shows the user rating system from Amazon.com, where users can provide ratings to their purchased products with a score in the range of 1 to 5. These ratings indicating the user preference levels toward those items—higher the score, the higher their satisfactions toward the items. Note that the range of rating scores can be varied on the different systems. For example, Agoda.com and IMDb use 10-scale rating system, where users can provide a rating from 1 to 10. These ratings are utilized as the main input for the recommendation algorithms for making the future rating prediction of unseen items for the individual users.



Need help with this device? Visit our Help pages or ask experts on our community forum.

Fire 7 Tablet (7" display, 16 GB) - Black

by Amazon

54,930 ratings

4.3 out of 5

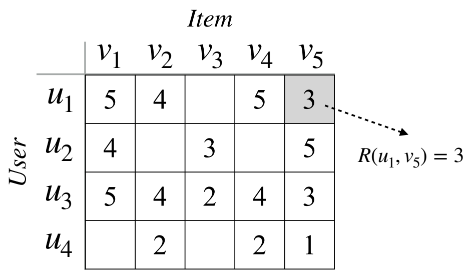
54,930 customer ratings

5 star	68%
4 star	14%
3 star	7%
2 star	4%
1 star	7%

See all customer reviews ›

possible.

Figure 2.1: A user rating system from Amazon.com



$$\begin{array}{c|ccccc} & v_1 & v_2 & v_3 & v_4 & v_5 \\ \hline u_1 & 5 & 4 & & 5 & \boxed{3} \\ u_2 & 4 & & 3 & & 5 \\ u_3 & 5 & 4 & 2 & 4 & 3 \\ u_4 & & 2 & & 2 & 1 \end{array}$$

Item
User
$R(u_1, v_5) = 3$

Figure 2.2: A user-item rating matrix.

2.1.1 Recommendation Strategies

After the ratings are collected, they can be represented by a user-item rating matrix, as shown by the example on Figure 2.2. Each row and each column in this matrix respectively represent each user and each item in the system, whereas each entry of the matrix represents the corresponding rating of a user given to an item. In Figure 2.2, for example, the user u_1 assigned a rating score "5" to the item i_1. The main task of a standard recommender system is to predict the ratings of items that do not yet rated by the individual users, i.e. those denoted by empty entries in Figure 2.2.

Formally, given $User$ a set of users, $Item$ a set of items, and $Rating$ a set of possible rating values (e.g. 1 to 5), an objective function R of a standard recommender system can be defined by:

$$R : User \times Item \to Rating. \tag{2.1.1}$$

After the ratings are computed, the items retrieving the highest prediction scores are selected as the ones to be recommended to an individual user, implying they are highly relevant to his/her personal preferences.

Many recommendation methods were proposed under a collaborative filtering-based (CF-based) approach. The main idea of CF-based approach is to utilize the preferences from the other users in the system for making a recommendation. The CF-based approach can be categorized further into two common sub-branches: the memory-based (or heuristic) and model-based approaches [6].

The memory-based CF approach relies on the ratings from the users who shared similar interests for making a rating prediction. From Figure 2.2, for example, the user u_3 is very similar to user u_1 since they provided similar ratings to similar set of items. Therefore, the ratings from user u_3 on items that do not yet rated by u_1 (e.g. v_3) can be helpful for predicting the rating for u_1. In this case, u_3 can be considered as a *neighbor* of u_1. The simplest method in a memory-based CF approach is the k-nearest neighbors (k-NN) technique [6], which makes a prediction by incorporating the ratings from the top k most similar neighbors to the target user. This method first computes the similarity between every pair of users by using a similarity metrics such as Pearson correlation coefficient, as expressed by

$$sim(u_a,u_b) = \frac{\sum_{v_j \in I(u_a,u_b)} (r_{u_a,v_j} - \bar{r}_{u_a})(r_{u_b,v_j} - \bar{r}_{u_b})}{\sqrt{\sum_{v_j \in I(u_a,u_b)} (r_{u_a,v_j} - \bar{r}_{u_a})^2} \sqrt{\sum_{v_j \in I(u_a,u_b)} (r_{u_b,v_j} - \bar{r}_{u_b})^2}}, \tag{2.1.2}$$

where $sim(u_a,u_b)$ denotes the similarity between users u_a and u_b, $I(u_a,u_b)$ is the set of items rated by both u_a and u_b, and $\bar{r}_{u_a}$ is the average rating of u_a. The rating of the user u_a on an unseen item v_k can then be estimated by

$$\hat{r}_{u_a,v_k} = \frac{\sum_{u_b \in Neighbors(u_a)} (r_{u_b,v_k} - \bar{r}_{u_b}) \times sim(u_a,u_b)}{\sum_{u_b \in Neighbors(u_a)} |sim(u_a,u_b)|}, \tag{2.1.3}$$

where $Neighbors(u_a)$ denotes the set of k most similar neighbors of user u_a. To produce an accurate prediction, the memory-based approach requires a significant amount of mutually-rated-items by users, in order to identify the high-quality neighbors.

On the other hand, the model-based approach exploits the rating data to build a predictive model, and uses it for future rating prediction. Many model-based CF techniques have been proposed, among them the latent factor model [50] is the most popular one due to its effectiveness in delivering high accurate prediction. The standard method in latent factor model is the matrix factorization (MF) technique [50] that models a rating as an interaction between the latent features of user and item. Specifically, each user u_i and each item v_j are respectively associated with user and item latent-feature vectors $\mathbf{x}_{u_i}$ and $\mathbf{x}_{v_j} \in \mathbb{R}^k$, where k is the number of latent dimensions. The rating of user u_i for item v_j is then estimated by

$$\hat{r}_{i,j} = \mathbf{x}_{u_i}^T \mathbf{x}_{v_j}. \tag{2.1.4}$$

The parameters $\mathbf{x}_{u_i}$ and $\mathbf{x}_{v_j}$ are learned and optimized by minimizing the regularized square error through a loss function defined by

$$L = \sum_{(i,j)\in O} (r_{i,j} - \hat{r}_{i,j})^2 + \lambda(\|\mathbf{x}_{u_i}\|^2 + \|\mathbf{x}_{v_j}\|^2), \tag{2.1.5}$$

where O denotes the set of observed user–item rating pairs, $r_{i,j}$ is the observed rating score of user u_i toward item v_j, and λ is a constant controlling the regularization rate. (The regularization term is added to avoid overfitting the observed rating data.)

The parameters $\mathbf{x}_{u_i}$ and $\mathbf{x}_{v_j}$ can be considered as the representations of user u_i and item v_j, respectively, in the latent space. Because they are learned and optimized from observed ratings, their representation quality depends on the quantity of available historical ratings. For most rating datasets, however, users typically rate only a few items among all the available items in the system, which leads to a rating sparsity problem. Such a problem would directly affect the quality of the user and item representations.

2.1.2 Challenges in Recommender Systems

In order to provide the effective recommendations, there are two common challenges that RS methods aim to achieve: to improve the recommendation accuracy, and to deal with the data sparsity issue.

Improving Accuracy

Accuracy is the most discussed property in the RS literature. The basic assumption of the RS indicates that the systems which provide more accurate prediction would be more preferable by users [35, 36]. Therefore, the vast majority of RS methods measure the effectiveness of their recommendation algorithms based on the prediction accuracy. An improvement in the recommendation algorithm can have a value of million dollars, and can be the factor that determines the success or failure of a business [9]. For example, Netflix offered a million dollar to whomever improve their prediction accuracy in terms of Root Mean Square Error (RMSE) by 10% [50].

The standard RS strategies such as CF-based approach, which utilize only the rating data for making a recommendation, often fails to deliver highly accurate prediction. The accuracy of a memory-based CF approach relies heavily on the quality of the neighbors who have rated the target items. The process of computing user similarity becomes non-trivial on a large-scale system with increasing numbers of users and items, and when the user-item rating matrix is sparse. Several efforts tried to enhance the computation of similarity [40, 60]. For example, [40] introduced the weighted Pearson correlation coefficient by modeling the confidence level among the neighbors. [60] proposed a novel similarity measure by considering the proportion of common ratings between two users.

By building a predictive model for future rating prediction, the model-based CF approach, in contrast, is more scalable than the memory-based CF approach. However, the prediction made by the model-based CF approach might not be as accurate as the memory-based CF approach when the user-item rating matrix is dense (when there is sufficient amount of ratings to identify high-quality neighbors). Therefore, several model-based CF methods have attempted to invent the models with more accurate prediction. For example, many variants of MF have been proposed to improve the prediction accuracy upon the baselined MF [50, 57, 81]. The most common variant of baselined MF that has been derived in many studies is a biased MF [50]. This method considered the biases in preferences from users and items to predict the ratings, as expressed by

$$\hat{r}_{i,j} = \mathbf{x}_{u_i}^T \mathbf{x}_{v_j} + b_{u_i} + b_{v_j} + \mu, \tag{2.1.6}$$

where b_{u_i}, b_{v_j}, and $\mu \in \mathbb{R}$ respectively denote the bias for user u_i, the bias for item v_j, and the global bias. These biases are added to take into account the deviations in ratings that might cause by some users or items. For example, some users always give high rating scores to any item, whereas some popular items tend to receive high rating scores from any user. Recently, *Item*

User	v_1	v_2	v_3	v_4	v_5	v_6	v_7
u_1		4					3
u_2			3				
u_3	5			4			
u_4		2				1	
u_5	4				5		
u_6						3	4

Figure 2.3: A sparse user-item rating matrix.

a neural network model has been integrated into a latent factor model to further enhance the prediction accuracy [39].

In fact, there are many additional sources of information that can be exploited to improve the prediction accuracy upon the standard RS. For example, the content-based recommendation approach [62] recommends items with similar characteristics (or features) to the ones the users liked in the past. On the other hand, the context-aware recommender systems [5] improves the prediction accuracy by incorporating the contextual information such as time, weather or season, which have significant influences on user preferences toward items. Finally, the multi-criteria recommendation approach [3, 4] takes into account the ratings in many aspects of an items (such as rating scores for cleanliness, service or location in hotel booking websites) to improve the accuracy of the recommendation. This thesis focuses mainly on utilizing the contextual information and multi-criteria ratings, which will be explained in more details in Chapter 2.2 and 2.4.1, respectively.

Dealing with Sparsity

In many large-scale recommender systems, there are tremendous amount of users and items. However, in most of them, many users only provide ratings to a few number of items, compared to all available items, and many items might be rated only few times by the users. Representing this kind of data with the user-item rating matrix would consequently result in a sparse matrix where most of its entries are missing, as shown by the example on Figure 2.3. This leads to one of the major challenges in RS, called *data sparsity* problem, that directly impacts the predictive performances of many recommendation algorithms.

Since a CF-based approach relies on historical ratings for future rating prediction, it is often suffered from rating sparsity problem, where there are insufficient amount of ratings for producing an effective recommendation. For example, the memory-based CF approach relies on the overlapped ratings on the mutually rated items between two users to compute their similarity. With the sparse user-item rating matrix, such overlapped ratings might be unavailable—making the computation of user similarity unreliable [22, 34]. This leads to the ineffective process of identifying the high-quality neighbors, and consequently degrade the prediction accuracy.

Similary, the predictive performance of the model-based CF approach is also affected by the sparsity of ratings. By building the predictive models on the sparse rating data, those models might be trained to overfit with the only available user-item pairs. In spite of providing high accuracy on the training (already seen) data, the overfitted model tends to provide poor accuracy on the new (or unseen) data.

Many works have attempted to alleviate the sparsity problem [33, 42, 82]. For example, [33, 42] combined content-based recommendation approach with the CF-based approach by considering similarity of the item characteristics. Moreover, some works applied the dimensionality reduction techniques such as the singular value decomposition (SVD) [48], the principle component analysis (PCA) [31], latent semantic analysis (LSA) [23], or the probabilistic latent semantic analysis (pLSA) [41] to overcome the rating sparisty. For example, [32] used a predicted ratings of SVD to fill in missing values in the user-item rating matrix, and then applied the traditional item-based CF to make a recommendation.

In the recent years, the user-generated reviews have been recognized by many studies [19, 21, 47, 64, 86, 101, 103] as the additional source for alleviating the sparsity problem. Therefore, in this work, we utilize the review data as the main source to overcome the sparsity problem.

2.2 Context-Aware Recommender Systems

2.2.1 What is Context?

The generic dictionary definition of context is "conditions or circumstances which affect something" [100]. However, this definition is too broad to be easily interpretable by researchers and developers into the specific real-world applications. This is because the concept of context has been widely studied across multiple research domains, such as computer science, cognitive science, philosophy or psychology [7], each of which tends to associate with a unique definition that is more specific than the previously mentioned generic definition from dictionary. One study [14] tried to examine and compare over 150 definitions of context from various disciplines, which led to a conclusion that it is not trivial to find such a unifying definition of context. This work, specifically, will be focused on the definition of context from the research area of human-computer interaction (HCI), which is more relevant to a recommender systems than the other areas.

According to the survey on context-aware computing applications conducted by [74], the definition of context that is widely accepted by many researchers in the field of HCI comes from Abowd et.al [1]. Abowd et.al [1] studied and tried to refine various definitions of contexts from the previous researches in HCI. They first categorized those definitions into three main types: by example, by synonym, and by aspect.

- By example: the definition of context is defined by its example. For example, Schilit and Theimer [84] defined context as a location, identities of nearby people and objects, and changes to those object. Ryan et al. [80] defined it as the user's location, environment, identity, and time. Finally, Dey [25] defined context as the user's emotional state, focus of attention, location and orientation, date and time, objects, and people in the user's environment. These definitions, however, are too specific, making them difficult to determine whether an unlisted information is context or not.
- By synonym: context is defined by its synonym. For example, Franklin and Flaschbart [30] defined it as the situation of the user. Brown [16] defined context as the elements of the user's environment that the user's computer knows about. Ward et al. [99] defined context as the state of the application's surroundings. These definitions, however, are too general, making them difficult to apply in practice.
- By aspect: this approach uses the aspects of context to determine whether the information is context or not. For example, Schilit et al. [83] defined three important aspects of context as: where you are, who you are with, and what resources are nearby. According to these three aspects, context is then defined by the computing environment (e.g. available devices), user environment (e.g. location or nearby people), and physical environment (e.g. lightning or noise level). Similary, Dey et al. [26] defined context as the user's physical, social, emotional or informational state. These definitions are also too specific. It is ineffective to identify which aspects of context are important to all situations, since such aspects will change from situation to situation.

Abowd et.al [1] stated that some of these definitions are too general, whereas some are too specific, which make them difficult to apply to any context-aware application. In order to make it easier to enumerate context for any given application scenario, Abowd et.al [1] provided the refinement from these three categories with their own definition of context as:

> "*Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is consider relevant*

to the interaction between a user and an application, including the user and applications themselves."

This definition makes it easier to identify context from any HCI application in general. If a piece of information can be used to characterize the situation of a participant in an interaction, then that information is context. Determining the context of use can allow the application to modify its current behaviour to better interact with the user.

The next question is, what is context in recommender system? From the definition of context defined by Abowd et.al [1], such "entity" in recommender systems would include a user and an item, and "interaction" between those user and item can be expressed in various forms such as clicks, likes, watches, or ratings, which indicates the user's decision on specific items. In other words, context in recommender systems, can be defined as "*any information that affects or influences the users' decisions when they are choosing items.*" [7]. For example, the season (e.g. summer or winter) could be considered as context since it strongly influences the users' decisions on which type of clothes they would buy.

Context in recommender systems can be classified further into representational and interactional views [28]. In *representational* view, context is defined with a predefined set of observable attributes, and its structure does not change significantly over time. For example, the context "daytype" might be defined with the static set of values {weekday, weekend}. On the other hand, the interactional view assumes that the user behavior could be induced by contexts, but the contexts themselves do not necessarily observable. This type of contexts might be unstructured and hidden in various sources of data such as mobile sensors [94], social media new feeds [59], or user-generated reviews [21, 63, 67].

Context could be a crucial factor for provide more relevant items to users, and leading to more accurate recommendations. There exists one approach in recommender system that dedicates to a study of utilizing context for improving the effectiveness of recommendations, namely *context-aware recommender systems*.

2.2.2 Improving Accuracy with Contexts

From past decades, contexts have been widely utilized as an additional source of information to improve the prediction accuracy upon the standard RS, which considers only the rating data. This is due to the fact that contexts can be very useful for the systems to narrow down the recommendation lists, and offer more relevant items that are suitable for each individual user in a specific contextual situation. For example, knowing that a user want to travel in summer, the system would suggest the hotels around beach areas, rather than the mountainside ski resorts that are less relevant to this user's contextual situation. Incorporating

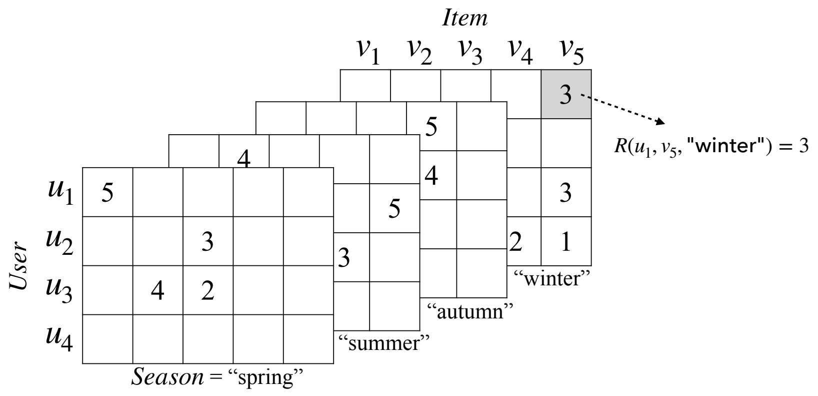


Figure 2.4: A user-item rating tensor for context *Season*.

contexts have been shown to improve the accuracy of the recommendations over the standard CF-based recommendations that do not consider contexts [5, 7, 11, 88].

In order to utilize contexts for making a recommendation, a system needs to incorporate the contextual information, in addition to the user and item information. The basic objective function R of a context-aware recommendation can be defined by:

$$R : User \times Item \times Context_1 \times \ldots \times Context_N \rightarrow Rating, \tag{2.2.1}$$

where $Context_n$ is a contextual variable, such as *Season* or *Weather*. By introducing a context as an additional variable, a multi-dimensional array or a tensor is required to represent the ratings under different contextual values, instead of a user-item rating matrix. Figure 2.4 for example, shows a three-dimensional user-item rating tensor for representing the ratings collected under the different values of a context *Season*.

2.2.3 Incorporating Contexts for Making Recommendations

With the multi-dimensional ratings presented in Figure 2.4, the standard recommendation techniques such as the CF-based approach which are designed for two-dimensional user-item rating matrix, cannot be applied directly. Therefore, several context-aware recommendation methods were proposed especially for dealing with contextual rating data [5, 7, 11, 88]. Adomavicius et al. [5] categorized the context-aware recommendation techniques into three main categories, based on the stages of the recommendation process in which the contextual information is applied, as follows:

- Contextual pre-filtering [5, 12]: only the ratings collected under the corresponding contexts are incorporated into the recommendation process. For example, if a user want to find a restaurant for having a dinner, only the ratings provided under the contextual value “dinner” are selected as an input for the standard recommendation techniques (such as CF-based approach).

- Contextual post-filtering [71]: the system first applies a standard recommendation technique to produce the recommendation list for each user. The items that are irrelevant to the user’s contexts are then filtered out from the list. For example, if a system detects that the user does not watch horror movie with his family, all horror movies are then remove from the recommendation list.

- Contextual modeling [11, 37, 46, 88, 89]: contexts are directly utilized in the process of learning the predictive model.

Campos et al. [17] conducted the experiments comparing the predictive performances among these three categories. The strength of both contextual pre-filtering and post-filtering is apparently their ability to exploit any standard recommendation techniques. On the other hand, the contextual modeling requires a specific technique designed for dealing with contexts. Its performance, therefore, depends on the effectiveness of the predictive model.

The earliest method among context-aware recommendation techniques is a context-aware collaborative filtering (CACF) [5], which is categorized as contextual prefiltering approach. The predictive performance of CACF is depended directly on the predefined values of contexts, which are to be used for selecting the ratings. Defining too specific values might result in too small number of the corresponding ratings, whereas defining too general value might incorporate the irrelevant ratings, and consequently degrading the prediction accuracy. Moreover, since this method used memory-based CF approach for making rating prediction, it also encountered the data sparsity problem.

One of the well-known context-aware recommendation techniques is a context-aware matrix factorization (CAMF) [11], which provided more accurate prediction and more robust to the rating sparsity than CACF. This method derived the matrix factorization technique [50, 49] to incorporate contextual variables as the bias, as expressed by:

$$\hat{r}_{i,j} = \mathbf{x}_{u_i}^T \mathbf{x}_{v_j} + \mu_{u_i} + b_{u_i} + \sum_{k=1}^{K} b_{j,c_k}, \tag{2.2.2}$$

where μ_{u_i} is an average rating of user u_i, and b_{v_j,c_k} is a bias of item v_j under contextual value c_k. By including the contextual biases into rating prediction, CAMF was able to achieve higher accuracy than the standard MF, and performed better on the sparse data than CACF.

Table 2.1: Examples of predefined contextual variables and their values.

Contextual variables	values
Daytype	Weekday, Weekend, Holidays
Time of day	morning, afternoon, evening, night
Weather	sunny, cloudy, rainy, stormy, snowy
Season	spring, summer, autumn, winter
Companion	alone, partner, family, friends
Location	home, public, office, school

However, it considered only the influences of contexts on the items, which is contradicted to the fact the contexts can have influences on both user preferences and item features.

2.2.4 Challenges for Contextual Recommendations

There are two common challenges that need to be addressed when incorporating contexts into the recommendations: how to obtain them, and how to identify which of them are relevant to the recommendation task.

Obtaining Contexts

Although context can improve recommendations, obtaining it is not trivial. In traditional recommendation schemes, users review items they have previously chosen and assign rating scores to indicate their preference levels for those items. Context is, however, rarely provided.

As previously mentioned in Section 2.2.1, Dourish [28] classified contexts into representational and interactional approaches. Most of the context-aware recommendation techniques [5, 11, 12, 46, 71, 88, 89] adopt the representational approach, whereby contexts are defined by predefined sets of contextual variables and their corresponding static values, such as those presented in Table 2.1. In order to obtain this kind of contexts, early context-aware recommendation techniques explicitly asked users to supply them by selecting from the predefined values their contexts at the time the items were being consumed. In order to deliver the satisfying prediction accuracy, such contexts need to be predefined with optimal values, which is not a trivial task since their values depends on the specific recommendation domains. For example, a context *Location* in movie recommendation domain might consist of the values {home, cinema}, whereas its values for travel recommendation might be {mountain, beach}. Moreover, most users have no intention of providing such contexts this way, which makes the representational approach less useful in real-world scenarios where.

The interactional approach, on the other hand, assumes that user behaviors are influenced by context, but the contexts themselves are not necessarily observable. In this approach, the unsupervised techniques are required to extract and represent contexts from additional sources of data [37, 59, 94]. In one such example, [37] applied topic modeling to represent context as sequences of latent topics that capture changes in users' interests. This interactional approach has the advantage of not having to predefine context values, which enables the discovery of hidden or unobserved contexts, and is therefore applicable to a wider range of recommendation domains than the representational approach. In recent years, one of the most popular source of context has been user-generated reviews [21, 63, 67].

Identifying Relevant Contexts

After obtaining the context, it is then important to separate the relevant context from the irrelevant context. For instance, a context *Season* should be more proper for hotel recommendation (e.g. hotels in beach areas for summer), whereas *Time of the day* is more suitable for restaurant recommendation (e.g. breakfast restaurants in the morning). Incorporating too many contexts not only degrades the quality of recommendations, but also increases dimension of data and thus triggering a sparsity problem [5].

The simplest but most expensive way to identify the relevant contexts is with the help of domain experts, who manually select or conduct a user survey to decide which contexts are relevant and which ones are not [10]. In addition to such manual methods, several works define a relevant context as one that has a significant influence on the distribution of ratings [7, 51, 61, 70], which can be identified automatically. Figure 2.5, for example, visualizes the rating distributions for two contextual variables, *Companion* and *Season*. Each cell contains the frequency of each contextual value for each rating value. For instance, users who watch movies alone provided rating "1" 17 times. The distributions of ratings can be visualized by the cell's grayscale shading, which represents the densities of context frequencies from highest (black) to the lowest (white). Note that each contextual value for *Companion* influences the distributions of ratings differently, whereas there is no significant difference in the distributions for *Season*. We can therefore hypothesize that *Companion* is a relevant context and *Season* is an irrelevant context for this data. In practice, most context-aware methods [61, 70] identify the relevance of contexts by applying the statistical testing such as the paired *t* test or Pearson's chi-squared test. However, these approaches is only applicable to a representational approach of contexts (when using predefined types of context such as those presented in Table 2.1). Moreover, most methods identify relevant context based on its influence on ratings of an entire dataset. In reality, the relevance of an

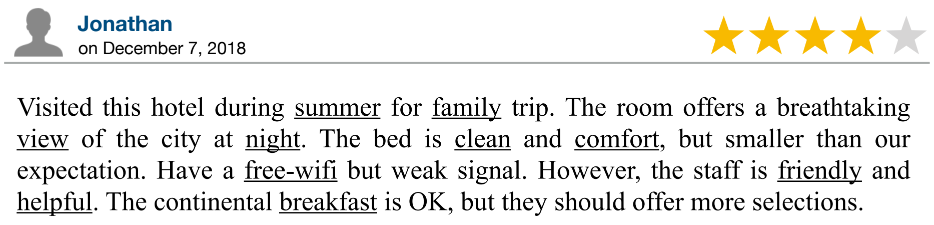
	value	rating '1'	rating '2'	rating '3'	rating '4'	rating '5'
Companion	alone	17	254	535	221	96
	partner	9	21	201	407	550
	family	511	334	186	21	13
	friends	34	301	341	413	200

(a)

	value	rating '1'	rating '2'	rating '3'	rating '4'	rating '5'
Season	spring	128	254	280	300	311
	summer	151	237	302	351	304
	autumn	177	200	321	328	344
	winter	145	274	294	345	338

(b)

Figure 2.5: Examples of context-rating co-occurrences for contexts (a) *Companion* and (b) *Season*


Jonathan
on December 7, 2018
★★★★☆

Visited this hotel during summer for family trip. The room offers a breathtaking view of the city at night. The bed is clean and comfort, but smaller than our expectation. Have a free-wifi but weak signal. However, the staff is friendly and helpful. The continental breakfast is OK, but they should offer more selections.

Figure 2.6: Example of a user-generated review and rating

element of context might well depend on individual user preferences and the specific target item features, which would, therefore, influence their ratings.

2.3 Review-Based Recommender Systems

2.3.1 Alleviating Rating Sparsity with Reviews

In many e-commerce websites, users can write textual reviews on the products they have purchased, in addition to the rating data. In reviews, users can provide comments explaining reasons behind their ratings on items, which offer richer and more meaningful information than numeric ratings. Such meaningful contents in reviews have been recognized as a valuable source of information to alleviate the rating sparsity occurred in a standard CF-based approach [21]. For example, consider the review in Fig. 2.6. Even if this user only provided one review, some very useful information can still be mined from that review, such as having traveled in summer with family members, liking city views at night, and being concerned about the cleanliness of the room. If such useful information is extracted effectively, they might be helpful in modeling the user preferences or item features in more efficient way than utilizing only ratings.

By leveraging a review text *Review* into rating prediction, the objective function *R* of a review-based recommendation can be defined by:

$$R : User \times Item \times Review \rightarrow Rating. \tag{2.3.1}$$

2.3.2 Leveraging Reviews for Making Recommendations

Since a review is provided in an unstructured textual form, it cannot be easily interpreted by the system. Therefore, a method for extracting and representing information from reviews is required for making a recommendation. For example, [29] built a user and item profiles based on the frequency of words extracted from reviews by applying a term frequency–inverse document frequency (TF-IDF) technique. The recommendation is then made based on the items with similar profiles to the user. Moreover, Poirier et at. [76] inferred the rating to create the user-item rating matrix to be used for CF-based approach. They represented a review with a word frequency vector, and combined it with a rating to train the Naive Bayes classifier, which is then used to infer ratings for the new reviews. Furthermore, McAuley and Leskovec [64] proposed a Hidden-Factors as Topic (HFT) model, which combined the latent topics learned from reviews with the latent factor model learned from ratings. The author represented a review with the set of topic distributions retrieved from latent dirichlet allocation (LDA) [66], which is linked with the item latent factors. Specifically, a distribution $\theta_{v_j,k}$ of topic k of item v_j is defined by the following transformation:

$$\theta_{v_j,k} = \frac{\exp(\kappa \mathbf{x}_{v_j,k})}{\sum_{k'} \exp(\kappa \mathbf{x}_{v_j,k'})}, \tag{2.3.2}$$

where $\mathbf{x}_{v_j,k}$ denotes a latent feature k of the representation of item v_j, and κ is a parameter to control the peakiness of the transformation. With Eq. 2.3.2, the latent topics can be incorporated into the process of learning the latent factor model.

In recent years, many deep-learning techniques have been adopted to model user and item representations from reviews due to their superior predictive performances [19, 47, 86, 103]. For example, DeepCoNN [103] applied a convolutional neural network (CNN) [102] to learn such representations, which were then used to predict ratings based on the latent factor model.

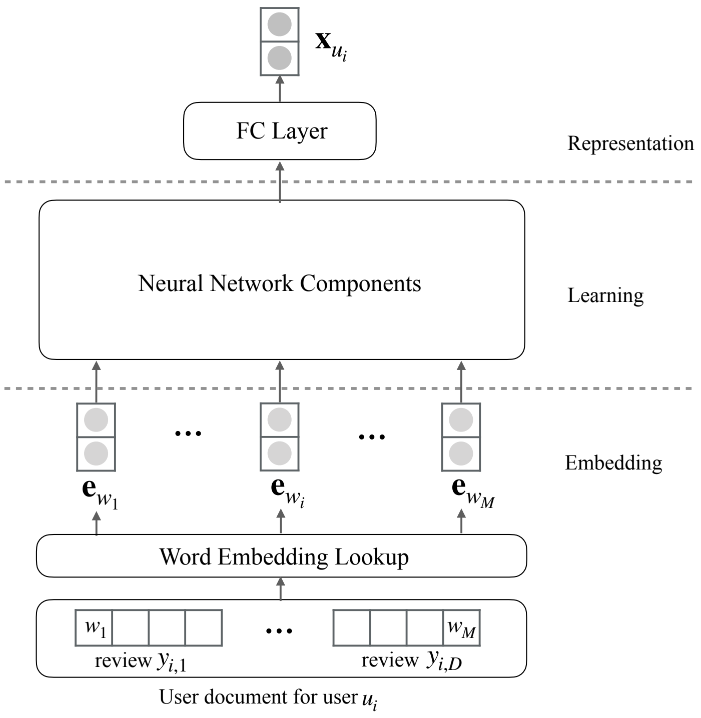


Figure 2.7: A neural network framework for constructing user representations from reviews

In an extension to the CNN, NARRE [19] applied an attention mechanism [96] to construct representations by considering different contributions from reviews based on their usefulness. Despite variations, these techniques share a common network framework for constructing representations, which is shown schematically in Fig. 2.7. To learn a representation for user u_i, this technique first creates a user document for u_i by concatenating all the user's previous reviews. Each of the M words in u_i's document is then looked up and mapped with its word embedding, which can be initialized randomly or by utilizing a pretrained word embedding such as Word2Vec [65], GloVe [73], or BERT [24]. These word embeddings are then fed into the neural network components to learn $\mathbf{x}_{u_i}$ as a representation of u_i. Note that an item representation can be constructed in the same way as a user representation. The output of such a framework is a static representation for every user and item in the training data.

2.3.3 Challenges for Review-Based Recommendations

Although the deep learning based models for review-based recommendations utilize different types of networks to learn the user and item representations, they share two similar principles that could limit their potential: the way they utilized the relevant words, and the way they incorporated the relevant reviews to learn such representations.

Identifying Relevant Words

First, most of the deep-learning based methods consider every word in a review as an input when learning user and item representations. Given that some words are not relevant either to user preferences or item features, such words should not be given any weight when modeling their representations. For example, in hotel recommendations, words such as “clean” or “breakfast” are more relevant to a user preferences toward a hotel features than words such as “he” or “run”. If only the words relevant to a specific recommendation domain are identified and utilized, the user and item representations could be constructed in a more efficient and meaningful way.

Utilizing a Particular Review’s Content

Moreover, the user and item representations are constructed in a static manner by aggregating their relevant previous reviews. This means that each user or item has one fixed representation per review. However, to predict a rating for a particular review, with the aim of modeling a user preferences and an item features for application to the user’s current situation, I believe that it is more important to concentrate and leverage the more relevant information embedded in that review. For example, the review in Fig. 2.6 mentions that the room offers breathtaking views of a city at night. To generate user and item representations for predicting a rating for this review, it would be beneficial to know how much the user prefers, and how much the hotel’s rooms are well known for, its city views at night. That is, my assumption is that the user and item representations should be dynamically constructed for each particular review, to capture the interactions between user preferences (or item features) and the relevant information in that review.

2.3.4 Extracting Contexts from User Reviews

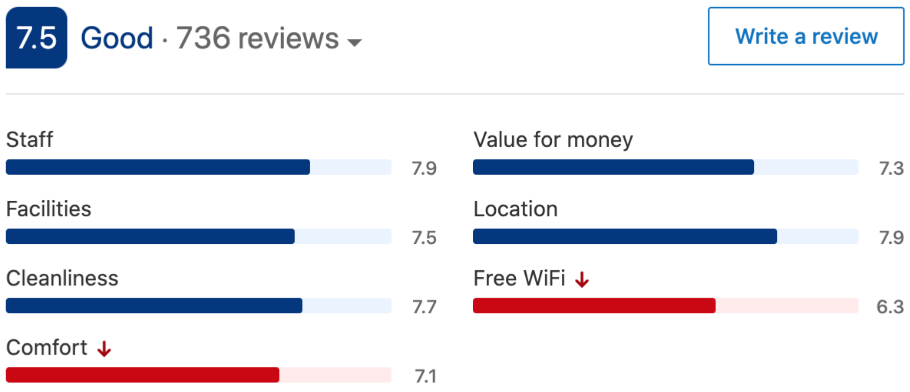
When writing reviews, users can express opinions describing their experiences and their satisfaction with items, which can be a valuable source of contexts [21]. As shown in Fig. 2.6, for example, underlined words such as “summer”, “family”, or “night” can be considered as contexts embedded in a review. Successfully identifying and utilizing contexts from reviews could be the key to satisfying both recommendation accuracy and alleviating rating sparsity in recommender systems.

However, unlike the context-aware recommendation methods that relied on predefined list of contexts [5, 7, 11, 88], the contexts embedded in reviews need to be recognized first before they can further be used for making recommedations. There are two main approaches to extracting contexts from reviews, namely supervised and unsupervised approaches. A supervised approach extracts contexts based on a predefined list of contextual variables and their corresponding values [2, 20, 38, 58, 54, 55]. Using the predefined contexts in Table 2.1, words such as “summer”, “family”, or “night” could be extracted as contexts from the review in Fig. 2.6. However, non-predefined words in Fig. 2.6 such as “clean”, “free-wifi”, or “breakfast”, which could potentially be considered as contexts, are overlooked. For a supervised approach to be robust, therefore, it will require the contextual variables and their corresponding values to be predefined optimally for each specific recommendation domain.

In contrast, an unsupervised approach aims to infer contexts from reviews without having to predefine them [13, 75, 101]. Some of these approaches [13, 75] classify reviews into context-rich and context-free reviews, based on features of each review such as the number of words, verbs, and verbs in the past tense. The contexts are then extracted as those words or topics that occur more often in the context-rich reviews. These two methods, however, require manual annotation of the review data (as context or noncontext) as part of the training process. Recently, CARL [101] has applied CNN and word-level attention to semantically infer contexts from reviews. Its user and item representations are constructed by modeling the attention weight of each word as its influence in each context on a user–item pair. This method was, however, presented using the framework shown in Fig. 2.7, which means that it suffers from the limitations of utilizing irrelevant words and constructing only static representations.

In addition, most context extraction methods [2, 13, 20, 38, 54, 55, 58] define and extract a context in the form of a single word such as those shown in Table 2.1. However, when users write reviews, they have flexibility in how their contexts are presented, including using phrases in addition to single words. For example, some contexts from the review in Fig. 2.6 might be best extracted as “family trip”, “night city view”, or “friendly staff”, which are more meaningful than just “family”, “night”, or “friendly.” I believe that other words that often accompany (or are present in the same text region as) context words might help in capturing the appropriate meaning of contexts, and should therefore also be extracted to represent contexts accurately.

I strongly believe that effectively extracting and utilizing contexts in reviews could help overcome the challenges of obtaining and identifying relevant contexts in context-aware methods, in addition to the limitations of modeling user and item representations from reviews via deep learning techniques.


7.5 Good · 736 reviews

Write a review

Staff 7.9

Facilities 7.5

Cleanliness 7.7

Comfort 7.1

Value for money 7.3

Location 7.9

Free WiFi 6.3

Figure 2.8: Example of multi-criteria ratings from Booking.com

2.4 Multi-Criteria Recommender Systems and Rating Conversion

2.4.1 Improving Accuracy with Multi-Criteria Ratings

In most of recommendation framework, a user provides a single rating to an item, indicating his overall preference toward that item. Such framework can be called single criterion (SC) recommendation. The SC recommendation defines a global objective function that represents the relationship among a user, an item and an overall rating score, as in Eq. 2.1.1. However, taking into account only the overall ratings to justify the user preferences on items might lead to a limitation of the recommendations. This is because in some systems, the users can express their preferences in multiple aspects of items in addition to an overall rating. Figure 2.8, for example, shows a rating system from Booking.com, where users can provide ratings to the criteria of the hotel, such as staff, facilities, cleanliness or value. This kind of system is referred to as multi-criteria (MC) recommendation [4].

The MC system grants a better opportunity to analyze the user preferences in more detail, and leads to more personalized and effective recommendations [4]. For example, as shown in Table 2.2, the SC system that considers only the overall rating will claim that user u_1 and u_2 are more similar than u_1 and u_3. However, by exploring the multi-criteria ratings that have been given under the criteria *Price*, *Location*, *Cleanliness*, and *Service*, the MC system proves that u_1 and u_3 are actually more similar. This means that letting the users express their preferences in multiple aspects can contribute a better understanding of their individual characteristics. Utilizing MC ratings can produce a better *TopN* precision results up to 3.8% when comparing to SC recommendations [3].

Table 2.2: Example of the overall and multi-criteria ratings on hotel rating data.

User	*Hotel*	Multi-criteria ratings				
		Overall	*Price*	*Location*	*Cleanliness*	*Service*
u_1	v_1	10	8	10	9	7
u_1	v_2	8	10	8	6	9
u_1	v_3	9	7	9	8	9
u_2	v_1	10	9	7	7	10
u_2	v_2	8	7	10	9	10
u_3	v_1	7	8	9	10	7
u_3	v_3	7	7	10	8	9

By considering the multi-criteria ratings $Rating_1, \ldots, Rating_K$ from K criteria in addition to the overall rating $Rating_0$, the objective function R for the MC recommendation can be defined by:

$$R : User \times Item \rightarrow Rating_0 \times Rating_1 \times \ldots \times Rating_K. \tag{2.4.1}$$

From Eq. 2.4.1, the prediction of MC recommendation can be made in three ways: predict each criterion rating individually, use multi-criteria ratings to predict the overall rating, or predict both multi-criteria and the overall rating.

2.4.2 Similarity Aggregation

Most of MC recommendation techniques are derived from the CF-based recommendation approach, which can then be categorized as the memory-based and model-based approaches.

The traditional multi-criteria recommendation is based on the memory-based CF approach [3], which relied on the computation of user similarity. In their work, the set of similar users are determined based on a multi-criteria user profile. In order to measure the user similarities, they extended the single criterion similarity methods for the multi-criteria scheme by applying the aggregation techniques.

The similarity aggregation methods firstly exploit the traditional similarity metrics to measure the similarities among users on each criterion independently [3, 72]. After the similarities on all criteria are calculated, they are aggregated into a single similarity through one of the following techniques. First, by averaging the similarities of all criterion:

$$sim(u_a,u_b) = \frac{1}{K}\sum_{k=1}^{K} sim_k(u_a,u_b), \tag{2.4.2}$$

where K is the number of the criteria. For another technique, a set of criteria weights are incorporated to corresponding criterion similarity for performing weighted average, as expressed by:

$$sim(u_a,u_b) = \sum_{k=1}^{K} w_k sim_k(u_a,u_b); \sum_{k=1}^{K} w_k = 1. \tag{2.4.3}$$

After the aggregated similarity is computed, the prediction of the overall rating can be done by any traditional CF-based approaches. Examples of the other MC recommendation techniques are the following. Schmitt et al. applied the multi-attribute utility theory (MAUT) [85] to the case study of car recommender system, while Le Roux et al. [79] constructed the course recommender system based on multi-criteria decision making. Also, Tangphoklang et al. [93] proposed adaptive user-variant weight to express preference of each user on each criterion.

2.4.3 Rating Conversion

In order to provide an effective rating prediction, the memory-based CF approach rely crucially on the ratings from neighbors. However, exploiting those ratings to make a prediction for the other users directly might lead to a problem. This is because the habits or patterns on giving ratings among users vary due to their personal biases. For example, on the rating range of 1 to 10, User u_1 might give rating score from 2 to 5 indicating 'dislike' to 'like', while User u_2, instead, gives the rating from 5 to 8 with the same intention. This means that 'like' for user u_1 equals to 'dislike' to user u_2. Therefore, using ratings from neighbors to predict the rating for an active user directly may not be practical.

In order to deal with the user personal biases in the ratings, many rating conversion techniques have been introduced in single criterion domain [8, 18, 43, 44, 56]. The main idea is to convert the ratings from the neighbors into the same scale as the active user, before utilizing them for a rating prediction. The most simplest approach that can be applied for converting the ratings is a normalization.

The normalization approach converts the user ratings into a specific range. Such range is usually between 0 and 1 where everyone's 'most like' and 'most dislike' will be mapped to score '1' and '0', respectively. Many normalization methods are proposed based on different assumptions, such as linear normalization, the Gaussian normalization, and the decoupling normalization.

Linear Normalization

This method maps ratings based on the maximum and minimum of personal user ratings. By using the linear function, the normalized rating value $r_{u_a}^{new}$ for the user u_a's specific rating is computed as:

$$r_{u_a}^{new} = \frac{r_{u_a}^{old} - r_{u_a,min} + 1}{r_{u_a,max} - r_{u_a,min} + 1}, \tag{2.4.4}$$

where $r_{u_a}^{old}$ is an original rating of u_a, $r_{u_a,max}$ and $r_{u_a,min}$ denote the maximum and minimum ratings user u_a has rated, respectively. This normalization method maps ratings based only on maximum and minimum of the personal user ratings.

Gaussian Normalization

This method considers two factors that affect the variance of ratings among users with similar interests [43]. The first factor is a difference of a rating from the average ratings. This factor relates to the fact that some users are more tolerant and tend to give higher ratings than others. Another factor is the difference of users rating scales. This comes from the fact that some users tend to assign items to a narrow range of ratings, whereas other users tend to assign items to a wide range. Combining these two factors, the ratings of each user are subtracted with his average and divided by the variance of his ratings, as expressed by:

$$r_{u_a}^{new} = \frac{r_{u_a}^{old} - \bar{r}_{u_a}}{\sigma_{u_a}}, \tag{2.4.5}$$

where $\bar{r}_{u_a}$ and σ_{u_a} are an average and a standard deviation of user ratings, respectively.

Decoupling Normalization

This method converts a user rating on item into a probability for that item to be favored by the user [44]. When the rating r_{u_a} is going to be normalized, the probability is determined based on two factors. First, a ratio between two numbers: the number of items which was rated no more than value r_{u_a} by the user u_a and the number of all items that the user u_a has rated. The high ratio means the rating r_{u_a} are likely to be favored by the user. The second factor is a ratio between the other two numbers: the number of items which was rated value r_{u_a} by the user u_a and the double number of all items that the user has rated. The low ratio means the rating r_{u_a} are likely to be favored by the user. Based on these two factors, a special formula; called halfway accumulative distribution was proposed as:

$$r^{new}_{u_a} = \frac{|\{v_j \in I_{u_a} | r_{a,j} \leq r^{old}_{u_a}\}|}{I_{u_a}} - \frac{|\{v_j \in I_{u_a} | r_{a,j} = r^{old}_{u_a}\}|}{2|I_{u_a}|}, \tag{2.4.6}$$

where I_{u_a} denotes the set of items to which user u_a has rated.

Although the normalization techniques are able to convert a user's ratings into the same range, the conversions are based only on the rating data of the only one user. This might lead to an inaccurate recommendation if there are two active users whose rating patterns are different but having the same neighbors. If the normalized ratings are used for the recommendations to these two active users, the results will be the same. For example, an active user u_a usually rates '0.4' (normalized ratings) while another active user u_b usually rates '0.7'. If these two users share the same neighbor u_c whose rated the target item with '0.8', they will receive the same predicted ratings of '0.8'. Although '0.8' seems like no effect on user u_b, it seems to be high value of rating for user u_a since his usual rating is '0.4'. Thus, the better solution is to find the relationship between each pair of user ratings: original user and target user, in order to convert neighbor ratings to individual active user ratings. The examples of such conversion techniques include linear mapping [8], Lathia's rating conversion [56] and Warat's rating conversion [18], which are explained further in Chapter 3.3.

Furthermore, the rating conversion techniques have been proposed only in the SC domain. Such SC rating conversion techniques can be applied to MC ratings by converting ratings from each criterion independently. However, this could cause a scalability problem and consume a lot of resources. Moreover, usually there are implicit relation among the criteria ratings when user makes decision to select an item. For example, a user may choose a room that have high score on both service and location, while ignore its price. If each criterion rating is converted independently, such implicit relation could be lost.

Chapter 3

Related Work

3.1 Review-Based Recommendation Techniques

3.1.1 DeepCoNN

The deep cooperative neural networks (DeepCoNN) [103] is a first method that introduced two neural networks for jointly constructing a user and item representations from reviews. The user network learns the representation for each user by exploiting the reviews written by that user, whereas the item network learns the representation for each item from the reviews written for that item. The learned representations are then used to predict the corresponding ratings in a layer on top of these two networks.

The architecture of DeepCoNN is presented in Figure 3.1. This model consists of two parallel neural networks, one for constructing the representation of the user u_i, and the other for constructing the representation of the item v_j. The user network first takes all reviews written by the u_i as inputs, and concatenates them to create the user document of u_i. Every word in such document is then mapped with the corresponding word embeddings, in which the authors initialized with pre-trained embeddings by Word2Vec [65]. After that, the word embeddings are fed into convolutional neural network (CNN), and are passed through the fully-connected (FM) layer to construct the representation $\mathbf{x}_{u_i}$ for user u_i. Note that the explanation for the item modeling is omitted here since a process of learning the representation $\mathbf{x}_{v_j}$ of item v_j on item network is very similar of the user presentation.

After both $\mathbf{x_{u_i}}$ and $\mathbf{x_{v_j}}$ are constructed, they are fed into the prediction layer. In prediction layer, $\mathbf{x}_{u_i}$ and $\mathbf{x}_{v_j}$ are first concatenated into single vector $\hat{\mathbf{z}} = (\mathbf{x}_{u_i}, \mathbf{x}_{y_j})$, which is used as the input for a factorization machine (FM) [78] for estimating the rating. The cost function of DeepCoNN is computed by:

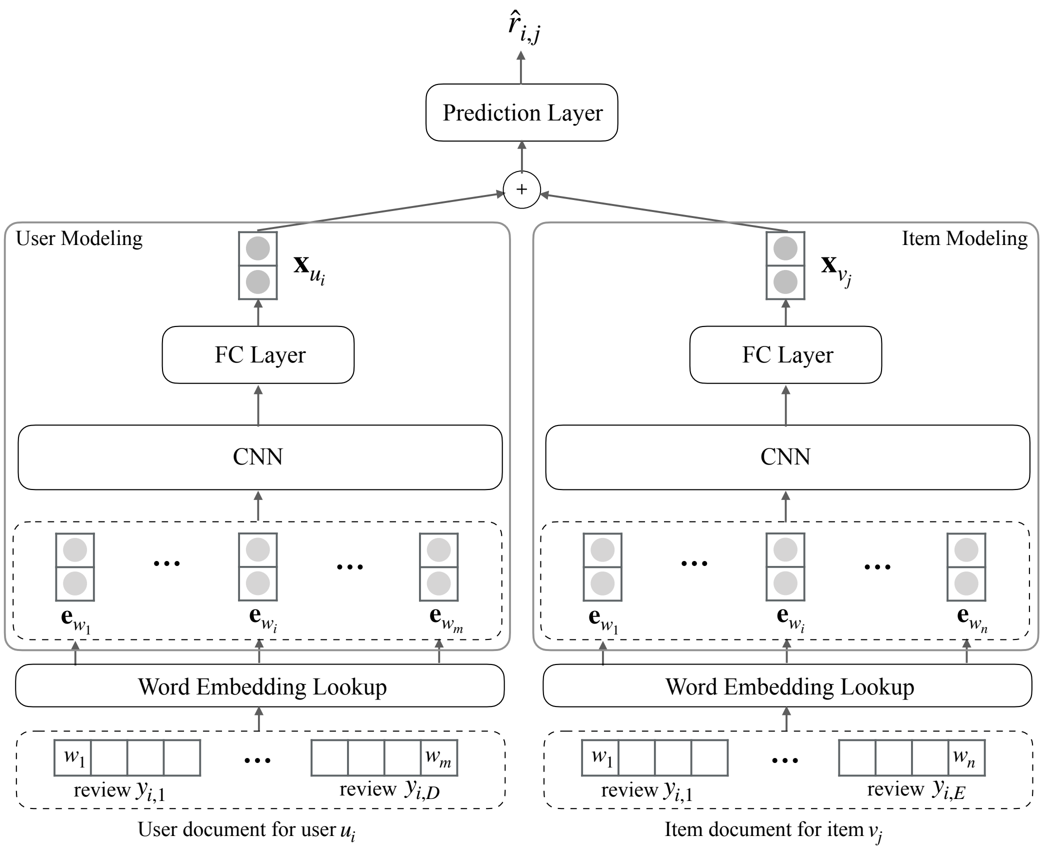


Figure 3.1: The model architecture of DeepCoNN.

$$J = \hat{w}_0 + \sum_{i=1}^{|\hat{z}|} \hat{w}_i \hat{z}_i + \sum_{i=1}^{|\hat{z}|} \sum_{j=i+1}^{|\hat{z}|} \langle \hat{v}_i, \hat{v}_j \rangle \hat{z}_i \hat{z}_j, \tag{3.1.1}$$

where $\hat{w}_0$ is the global bias, $\hat{w}_i$ models the strength of the i_{th} variable in $\hat{z}$ and $\langle \hat{v}_i, \hat{v}_j \rangle = \sum_{f=1}^{|\hat{z}|} \hat{v}_{i,f} \hat{v}_{j,f}$ is for modeling the second order interactions.

3.1.2 NARRE

The neural attentional regression model with review-level explanations (NARRE) [19] is an improved version of DeepCoNN that incorporated with attention mechanism. This method learned different contribution of reviews to construct the user and item representations based on the usefulness of reviews. The idea is that the highly-useful reviews are providing review-level explanations to help users make better and faster decisions, and should have more contributions for construct such representations.

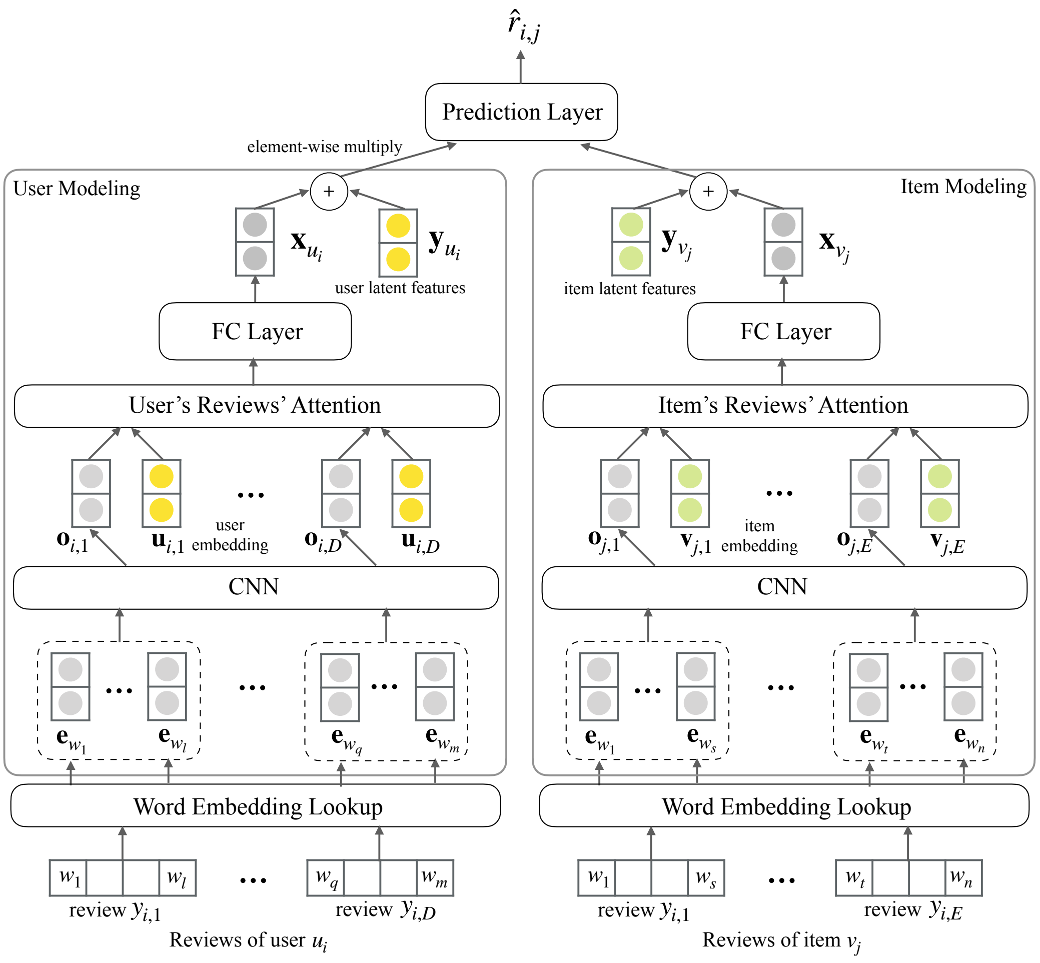


Figure 3.2: The model architecture of NARRE.

Specifically, the architecture of NARRE is presented in Figure 3.2. Similar to DeepCoNN, NARRE comprises of two parallel neural networks, one for modeling the user, and one for modeling the item. However, rather than concatenating all reviews into a single user or item document, each review is transformed into a matrix of word vectors, and is fed into CNN to learn its representation, separately from other reviews. Let $\mathbf{o}_{i,d}$ denotes the representation of review $y_{i,d}$ written by user u_i. NARRE first computes the attention score of $y_{i,d}$ by:

$$a^*_{i,d} = \mathbf{h}^T ReLu(\mathbf{W}_O \mathbf{o}_{i,d} + \mathbf{W}_u \mathbf{u}_{i,d} + \mathbf{b}_1) + b_2 \tag{3.1.2}$$

where $\mathbf{W}_O \in \mathbb{R}^{t \times k_1}$, $\mathbf{W}_u \in \mathbb{R}^{t \times k_2}$, $\mathbf{b}_1 \in \mathbb{R}^t$, $\mathbf{h} \in \mathbb{R}^t$, $b_2 \in \mathbb{R}$ are model parameters, t denotes the hidden layer size of the attention network, *ReLu* [68] is a nonlinear activation function, and $\mathbf{u}_{i,d} \in \mathbb{R}^{k_2}$ is an user embedding of user u_i. The attention score is then normalized with softmax function as expressed by:

$$a_{i,d} = \frac{\exp(a^*_{i,d})}{\sum_{d=1}^{D} \exp(a^*_{i,d})} \tag{3.1.3}$$

The representation of $\mathbf{o}_i$ is then computed by a weighted sum of the attention scores of all D reviews written by u_i as:

$$\mathbf{o}_i = \sum_{d=1}^{D} a_{i,d}\mathbf{o}_{i,d} \tag{3.1.4}$$

The final representation of u_i is created by passing $\mathbf{o}_i$ to the fully-connected layer as:

$$\mathbf{x}_{u_i} = \mathbf{W}_0\mathbf{o}_i + \mathbf{b}_0 \tag{3.1.5}$$

where $\mathbf{W}_0 \in \mathbb{R}^{p \times k_1}$ and $\mathbf{b}_0 \in \mathbb{R}^p$ are model paramters.

After the representation from reviews of user u_i, $\mathbf{x}_{u_i}$ and item v_j, $\mathbf{x}_{v_j}$ are constructed, they are combined with the user and item latent feature vectors $\mathbf{y}_{u_i}$ and $\mathbf{y}_{v_j}$ from a latent factor model, as expressed by:

$$h_0 = (\mathbf{y}_{u_i} + \mathbf{x}_{u_i}) \odot (\mathbf{y}_{v_j} + \mathbf{x}_{v_j}) \tag{3.1.6}$$

Finally, the rating of user u_i on item v_j is estimated by:

$$\hat{r}_{i,j} = \mathbf{W}_1^T h_0 + b_i + b_j + \mu \tag{3.1.7}$$

where $\mathbf{W}_1 \in \mathbb{R}^p$ denotes the weight matrix in prediction layer, b_i, b_j and μ denote the user bias, item bias and the global bias, respectively. The training was done using Adam [27] as the optimizer.

3.2 Context Extraction Techniques from User Reviews

3.2.1 Rich-Context

The rich-context [75] represented each review with the distribution of the contextual topics defined by Bauman and Tuzhilin [13], which is used as an input for factorization machine to predict the rating.

In Bauman and Tuzhilin [13], the author proposed an unsupervised method for extracting contexts from reviews. First, the author separated set of reviews Y into two groups: the *context-rich* reviews Y^r, which are rich in contextual information, and *context-free* reviews Y^f, which are less (or absent) of contextual information. The separation is done utilizing the classical K-means clustering methods based on the following characteristics of a review:

- *LogSentences*: logarithm of the number of sentences in the review plus one (to avoid empty review).
- *LogWords*: logarithm of the number of words in the review plus one.
- *VBDsum*: logarithm of the number of verbs in past tense in the review plus one.
- *Vsum*: logarithm of the number of verbs in the review plus one.
- *VRatio*: the ratio of *VBDsum* and *Vsum* ($\frac{VBDsum}{Vsum}$)

After all reviews are separated, the contextual information can be extracted with two approaches: the word-based and the topic-based approaches.

Word-Based Context Extraction

The word-based approach extracts contexts as those words that occur more in the context-rich reviews than in context-free reviews, The identification process is explained by the following steps:

1. For each review y_i, identify the set of nouns N_i in that review.
2. For each noun n_k, compute its weight frequencies $w^r(n_k)$ in the context-rich reviews Y^r, and $w^f(n_k)$ in the context-free reviews Y^f, as follows:

$$w^r(n_k) = \frac{|y_i : y_i \in Y^r \text{ and } n_k \in N_i|}{|y_i : y_i \in Y^r|}, \tag{3.2.1}$$

and,

$$w^f(n_k) = \frac{|y_i : y_i \in Y^f \text{ and } n_k \in N_i|}{|y_i : y_i \in Y^f|}. \tag{3.2.2}$$

The weight frequency $w^r(n_k)$ is computed as the ratio of the number of all context-rich reviews that contain n_k, as compared to all context-rich reviews. Similarly, $w^f(n_k)$ is computed as the ratio of those context-free reviews that contain n_k, as compared to all context-free reviews.

3. Remove all nouns that have lower overall frequency than α (which is usually set as 0.005).

4. For each noun n_k, compute the ratio of its weighted frequencies on context-rich and context-less reviews: $ratio(n_k) = \frac{w^r(n_k)}{w^f(n_k)}$.

5. Filter out all nouns n_k with $ratio(n_k) < 1$.

The remaining nouns after the identification process are then considered as context words.

Topic-Based Context Extraction

The topic-based approach extracts contexts as the topics that have higher probabilities in the context-rich reviews than in context-free reviews. This methods applies the latent dirichlet allocation (LDA) [15] to identify such contextual topics by the following steps:

1. Build the LDA model on the set of context-rich reviews Y^r.

2. Apply the learned LDA model on all reviews to obtain the topic distribution T_i for each review y_i.

3. Similar to the word-based context extraction approach, for each topic t_k, compute its weight frequencies $w^r(t_k)$ in the context-rich reviews Y^r, and $w^f(t_k)$ in the context-free reviews Y^f, as follows:

$$w^r(t_k) = \frac{|y_i : y_i \in Y^r \text{ and } t_k \in T_i|}{|y_i : y_i \in Y^r|}, \tag{3.2.3}$$

and,

$$w^f(t_k) = \frac{|y_i : y_i \in Y^f \text{ and } t_k \in T_i|}{|y_i : y_i \in Y^f|}. \tag{3.2.4}$$

4. Remove all topics that have lower overall frequency than α (which is usually set as 0.005).
5. For each topic t_k, compute the ratio of its weighted frequencies on context-rich and context-less reviews: $ratio(t_k) = \frac{w^r(t_k)}{w^f(t_k)}$.
6. Filter out all topics t_k with $ratio(t_k) < 1$.

After the process of topic identification, the remaining topics are considered as the contextual topics to represent contexts extracted from reviews.

3.2.2 CARL

The context-aware user-item representation learning model for rating prediction (CARL) [101] models the user and item representations from reviews with word-level attention, which can be considered as the implicit influences of contexts to the rating. Note that the full model of CARL combined review-based feature learning with interaction-based feature learning (which was derived from latent factor model) for predicting ratings. This section, however, only focuses on the modeling the contribution of words, as they represent the influences of contexts.

The model architecture of review-based feature learning of CARL is presented in Figure 3.3. Similar to DeepCoNN and NARRE, this model comprises of two parallel neural networks for constructing the user and item presentations from reviews. Following DeepCoNN, CARL concatenates all reviews written by user u_i to create a user document, and maps their words to the corresponding word embeddings. This method then applies the convolution operation on every M word embeddings of the user document to create the contextual feature vectors $\mathbf{c}_{w_1} \dots \mathbf{c}_{w_M}$. Next, the contextual feature vectors of both user document and item documents are fed into the shared attention layer. Let $\mathbf{c}_m^{u_i}$ and $\mathbf{c}_n^{v_j}$ denote the contextual feature vectors of words at position m and n of user u_i's document and item v_j's document, respectively. The attention score of each contextual feature vector is computed based on the pair-wise relatedness between $\mathbf{c}_m^{u_i}$ and $\mathbf{c}_n^{v_j}$ as:

$$\mathbf{R}_{j,k} = \tanh(\mathbf{c}_m^{u_i} \mathbf{T} \mathbf{c}_n^{v_j}), \tag{3.2.5}$$

where $\mathbf{R}_{j,k}$ is the relatedness between $\mathbf{c}_m^{u_i}$ and $\mathbf{c}_n^{v_j}$, tanh is the hypobolic tangent function, and $\mathbf{T} \in \mathbb{R}^{j \times j}$ is an attentive matrix, where j is the number of convolutional filter.

Such relatedness is used to compute attention score for each contextual feature vectors, and the attention score is used as a weight to modify each contextual feature vector. Finally, the

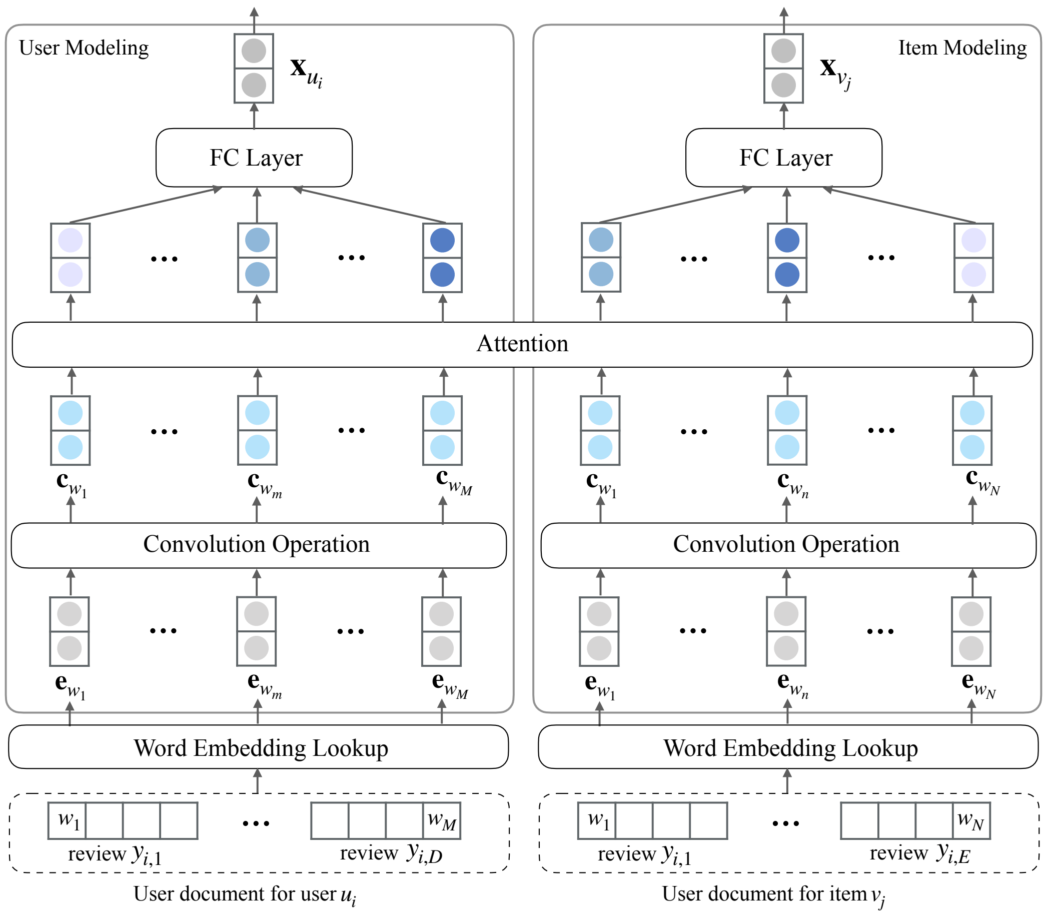


Figure 3.3: The architecture of review-based feature learning of CARL.

weighted contextual feature vectors are sent to the fully-connected layer to create the user and item representations. The different contributions of the weighted contextual feature vectors can be considered as the different influences of contexts to the user individual preferences and item specific features, which consequently affect the corresponding rating of each review.

3.3 Rating Conversion Techniques

3.3.1 Linear Mapping

Similar to the linear normalization, the linear mapping technique [8] also uses the linear function to map the ratings. The difference is that it maps original user rating's range to the target user rating's range instead of the range [0, 1]. For example, user u_a has rated (3, 3, 2, 4, 4) and user u_b has rated (5, 4, 6, 8). It maps user u_a's ratings, range [2, 4], into range [4, 8] of user u_b. Let $r_{u_b,max}$ and $r_{u_b,min}$ denote the maximum and minimum rating that the target user u_b has rated, respectively. The linear mapping function is defined as the following equation:

$$Linear_{u_a \to u_b}(r) = (r_{u_a} - r_{u_a,min}) \cdot \frac{r_{u_b,max} - r_{u_b,min}}{r_{u_a,max} - r_{u_a,min}} + r_{u_b,min}. \tag{3.3.1}$$

Using Eq. (3.3.1), the rating 2, 3 and 4 of user u_a is linearly mapped to rating 4, 6 and 8 of user u_b, respectively.

Although the linear mapping approach is able to solve the some problems of normalization, some are still occurred. One is that if there are users who have different rating patterns but having the same maximum and minimum rating, the prediction results from the same neighbors are still the same as occurred in the normalizations. Another problem is that if the target user has rated items by only one rating value, all ratings from all neighbors will be mapped to that value. This makes an invalid prediction since all items are received the same predicted rating.

3.3.2 Lathia's Rating Conversion function

The Lathia's rating conversion technique [56] converts the ratings based on the ratings of the two users on *co-rated items*. The co-rated items is the set of items which have been mutually rated by both users. Consider a set I_{u_a,u_b} of co-rated items rated by both user u_b and u_a. If u_b has rated for almost all items in I_{u_a,u_b} greater than ratings given by u_a, Lathia's function assumes that the u_a's original rating will be converted to greater value in u_b's aspect. The converted rating is computed by a weighted mean between quantity of each group of ratings in I_{u_a,u_b}. Let $lower_r$, $same_r$ and $higher_r$ denote the groups of items that a target user u_b has rated less, equal and more than original rating r of user u_a, respectively. The $lower_r$ group decreases the rating from r to $(r-1)$, $same_r$ unchanges r and $higher_r$ group increase the rating to $r+1$. This is described by the following equation:

$$Lathia_{u_a \to u_b}(r) = \frac{(r-1)|lower_r| + r|same_r| + (r+1)|higher_r|}{|lower_r| + |same_r| + |higher_r|}. \tag{3.3.2}$$

For example, suppose user u_a gives rating '4' to fourteen co-rated items, and user u_b rates less than '4' on five of them, equal to '4' on three of them, and greater than '4' on the remaining six of them. With Eq. (3.3.2), Lathia's function converts rating '4' of user u_a to '4.07' of user u_b. As for the rating that has no record, the transposed rating remains the same as the original one.

3.3.3 Warat's Rating Conversion Function

Since most systems suffer from the sparsity problem, applying Lathia's function on very limited ratings on co-rated items would be quite challenging. Moreover, the converted ratings by Lathia's function are only converted to the range [-1, 1] of the original ratings. Even user u_a has rated only '3' and user u_b has rated on '5' on all items, the rating '3' of u_a will be converted to '4' of user u_b (not '5'). This problem is referred to as rating *shifting* problem.

Warat et. al [18] proposed a conversion method named Warat's transpose function to overcome the problems of insufficient co-rated items and the rating shifting. It was designed for single rating conversion which converts the original rating to be an average rating of the target user. The Warat's function consists of two components: the original value and the adjusting term. Original value is the rating to be converted and the adjusting term is an average of difference between the original rating and the related target ratings (ratings of target user on items corresponding to the original rating). Due to sparsity of data, there is a chance that there is no actual related target ratings. In that case the pseudo ratings derived from matrix factorization [50] are used instead.

The Warat's function controls the adjusting term with the reducing term which are confident of generated pseudo ratings and distribution of ratings as shown by Eq. (3.3.3). The less error and less deviation of the considered rating lead to more suitable converted ratings.

$$W_{u_a \to u_b}(r) = r + \frac{\sum_{v_j \in \beta_{u_a,r}} (r_{b,j} - r)}{|\beta_{u_a,r}|} \cdot Dist_{u_b,u_a} \cdot Conf_{u_b}. \tag{3.3.3}$$

We show how the Warat's method work in three cases. For the first case, if user u_a and u_c shared the same set of items with ratings (2, 2, 2) and (4, 4, 4), respectively. Since their ratings contain no distribution, if the pseudo ratings from u_b have no error, '2' of u_a will be converted exactly to '4' of u_b (i.e. the adjusting term is '+2'). On second case, there exist another user u_d who rated the same set of items with distributed ratings (3, 4, 5), no error. Although the average rating is '4', the converted rating from u_a is only '3.1' since the distribution term reduced the adjusting term from '+2' to '+1.1'. The final case, if there is an error on u_d's derived pseudo ratings with confident of 0.8, the adjusting term is reduced to '0.6' and the converted rating from u_a is '2.6'.

Chapter 4

Unsupervised Context Extraction via Region Embeddings for Context-Aware Recommendations

4.1 Introduction

Recommender systems (RS) were devised to provide personalized recommendations about specific items to individual users. The most common approach in RS is the collaborative filtering-based (CF-based) approach, which exploits the user past preferences about items, such as their ratings, to create a predictive model for their future rating of unseen items. In addition to using rating data, context-aware recommenders offer more effective recommendations by taking into account contextual information (or simply "context"). Context such as location, time or weather can have a major influence on users' decisions when they are choosing items. For example, if a user is seeking a hotel for a summer vacation, the recommendation engine should suggest hotels in a beach area, rather than in mountainside ski resorts.

By incorporating contexts, many context-aware methods have been able to achieve improved prediction accuracy, when compared with standard CF-based approache [5, 7, 11, 88]. However, as previously mentioned in Chapter 2.2.4, two significant challenges remain for context-aware methods. First, obtaining contexts is not a trivial task because they are rarely provided directly by users. Many context-aware datasets collect contextual information by predefining a list of contextual variables and possible corresponding values, and asking users to select appropriate values for the contextual variables at the time they rate the items. To deliver the best possible predictive performance, this approach requires the expensive process of carefully predefining optimal values for the contextual variables, each of which tends to be associated with relatively few recommendation domains. Moreover, incorporating too many contexts tends to increase the dimensionality of the data, thereby triggering a sparsity issue. After obtaining the contexts, the second challenge is to identify and utilize only the contexts that are relevant to a specific recommendation task. Several methods define a relevant context as one that has a significant influence on the distribution of ratings [7, 61, 70]. In practice, most context-aware methods [61, 70] identify the relevance of contexts by applying statistical tests such as the paired t test to each contextual variable. However, this approach is only applicable when using predefined types of context that have static values and are of fixed size.

In recent years, many works tried to incorporate contexts from the other sources of data, and one of the most popular sources is the user-generated reviews. In reviews, users can express opinions about their experience and level of satisfaction with the items concerned, which can, therefore, be a rich source of context data [21]. However, the context in reviews has to be recognized as such before it can be used. Two common approaches to this task are supervised and unsupervised approaches. The supervised approach [20, 38, 58] utilizes techniques such as text mining to identify and extract words in a review that match a predefined list of context values. Determining the optimal values of contexts for a specific domain then becomes the main challenge in this approach because it can significantly affect the quality of the recommendations. To address this issue, some approaches have applied unsupervised techniques to extract context from reviews [13, 75, 101]. Moreover, contexts extracted from most such methods have been restricted to a single word format, e.g., “family” or “breakfast”. In fact, the precise meaning of a context might require more than one word for its expression, such as “family trip” or “continental breakfast.”

In this chapter, the *context-aware region embedding* (CARE), a novel unsupervised method for defining, extracting and representing context from review data is proposed [91]. A context is defined as any word in a region of text that has an influence on the distribution of ratings. Such words can be in unigram, n-gram or even skip-gram format (nonadjacent words), provided that they reside within the same region of text. By applying region embedding with the local context unit proposed by [77], the positions of context words in a text region can be emphasized as those that contribute the highest variance in ratings. As a result, contexts can be represented by the region embeddings that capture their influence on the rating distributions of a review data.

The experiments and extensive discussion regarding the extracted contexts were conducted. These include the analysis of the contexts extracted from various recommendation domains, the influence of the extracted contexts to the rating distributions, the investigation on quality

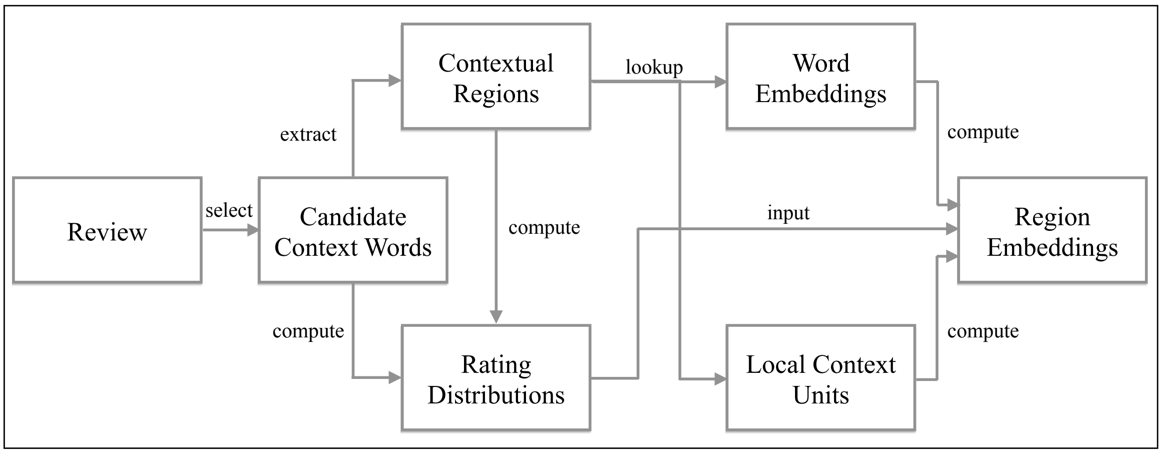


Figure 4.1: A workflow architecture of CARE.

of the region embeddings, the analysis on the merits of defining and extracting contexts as contextual regions with a review sentiment classification, and finally, the discussion about the applications suitable for CARE.

The main contributions of this CARE can be summarized as follow:

- CARE is capable of automatically extracting relevant contexts from review datasets in any recommendation domain, providing that review texts and ratings are available.
- A relevant context extracted by CARE is defined not only by a single word format, but also by including its adjacent and/or nonadjacent words in the region of text that influence the distributions of ratings.
- The extracted contexts effectively capture the polarities of reviews, which is helpful for explaining the reviews' ratings.

4.2 Model Overview

In this section, an overview of the proposed context extraction method, i.e. CARE, is presented. The workflow of CARE is illustrated in Figure 4.1. First, the candidates for context words are identified. After that, all text regions containing those candidate context words are extracted, and their associated rating distributions are generated. Those text regions and their rating distributions are then fed into a neural network to learn the word embeddings and local context units, which are then used to compute the region embeddings.

word	rating '1'	rating '2'	rating '3'	rating '4'	rating '5'	variance
"clean"	17	89	251	432	789	95584.8
"good"	54	185	325	724	551	73539.7
"dirty"	716	334	186	21	13	84209.5
"not"	542	734	485	128	78	79098.8
"service"	377	501	558	482	325	9058.3

Figure 4.2: Example of word-rating co-occurrences and their corresponding variances.

bigram	rating '1'	rating '2'	rating '3'	rating '4'	rating '5'	variance
"good service"	7	15	122	288	311	21103.3
"worst service"	351	291	78	8	1	26935.7

Figure 4.3: Example of rating co-occurrences with two bigrams containing the word "service".

4.3 Identifying Candidate Context Words

CARE adopt the definition used in Odić et al. [70], which defines relevant contexts as those that contribute to explaining the variance in ratings. By applying this definition to review data, a context can be considered as any word in the reviews that influences the distribution of ratings. For example, Fig. 4.2 presents a word-rating co-occurrence matrix, which gives the word frequency for each rating value. From this figure, words such as "clean" or "good" have frequent mentions in reviews with more positive rating scores such as "4" and "5", whereas "dirty" or "not" were mentioned more frequently for more negative scores such as "1" or "2". The implication is that these words influence the distribution of ratings, and could therefore be considered as "candidates" for contexts. To measure the significance of the influence of a word on the distribution of ratings, CARE first compute its variance on such a distribution, as also shown by the example in Fig. 4.2. After computing variances for every word in the review corpus, only those words having variances above the predefined minimum-variance threshold min_{var} are selected as *candidate context words* and stored in the candidate list *Cand*. Note that, in addition to direct context words (e.g., "clean"), *Cand* also includes opinion or sentiment words such as "good" or "not" if they also have significant influence on the distributions of ratings.

4.4 Extracting Contextual Regions

Depending solely on the candidate context words might not be sufficient to cover the variety of influences of contexts on the distributions of review ratings. This is because some words that often accompany candidate context words (neighboring words) might significantly alter the ways they influence the distributions of ratings. For example, as shown in Fig. 4.2, the word “service” has a mixed frequencies of rating scores “2”, “3”, and “4”, indicating a neutral distribution toward middle-rank ratings. However, when considering its co-occurrence with the word “good” (i.e., “good service”), the rating distribution could change from neutral to strongly positive, whereas “worst service” could result in a strongly negative distribution, as shown in Fig. 4.3. This means that neighboring words might be opinion, sentiment, or other words that could change the semantic meaning of a candidate context word, and therefore influence their rating distributions. This emphasizes the importance of considering neighboring words in addition to the candidate context words if the modeling of the influence of contexts is to be effective.

Consider a candidate context word $c_n \in Cand$. The neighboring words of c_n are defined as any word w_t that occupies the same “text region” of c_n. More specifically, considering $w_t \in region(c_n, d)$, where d is the window size for a region of length $2 \times d + 1$. The $region(c_n, d)$ is called a *contextual region* of c_n. Note that w_t can be in any position within $region(c_n, d)$, not necessarily directly adjacent to c_n. This takes account of the different writing styles users may adopt for the same meaning in writing reviews. For example, “view of a city at night” and “night view of a city” are simply alternative expressions of the same context.

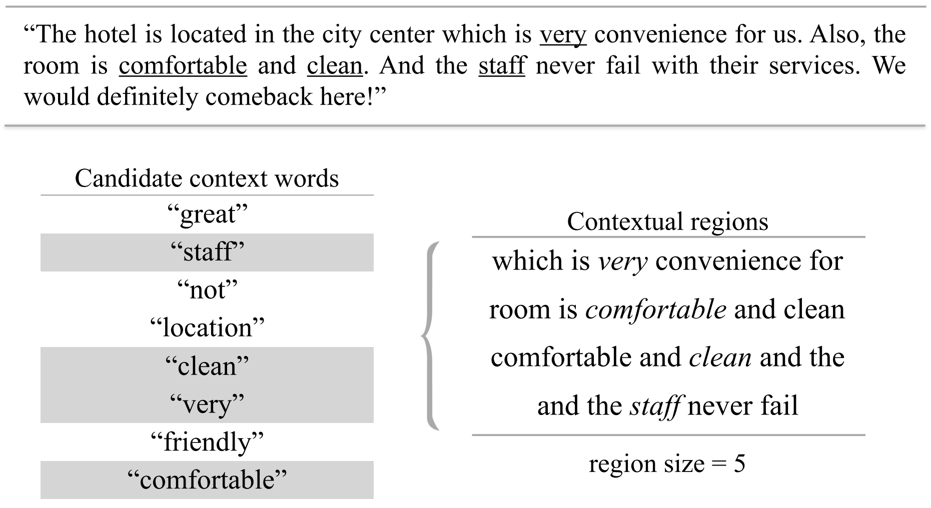
A relevant context in this work is, therefore, formally define as:

> “*any word in a region of text that has a significant influence on the distributions of ratings. Such context can be in a format of a single word, adjacent words, or even nonadjacent words, provided they occupy the same text region.*”

To identify the positions of these words, their associated contextual regions are firstly needed to be extracted. Figure 4.4, for example, shows the contextual regions of size 5 extracted from one review. Given that this review contains four candidate context words, four contextual regions are extracted.

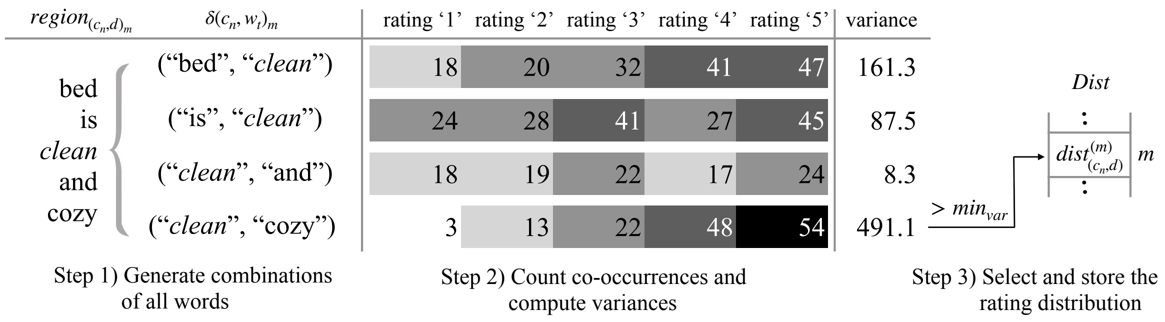
Let *Region* denotes a set of all contextual regions extracted from reviews and let a contextual region at index m in *Region* is denoted by $region(c_n, d)_m$. The positions of words to be constructed as context can then be identified by the following steps.

1. Generate all possible combinations of words $w_t \in region(c_n, d)_m$ of size θ (where $\theta \leq 2 \times d + 1$) that include c_n, denoted by $\delta(c_n, w_t)_m$.



"The hotel is located in the city center which is very convenience for us. Also, the room is comfortable and clean. And the staff never fail with their services. We would definitely comeback here!"

Figure 4.4: Extracting the contextual regions from a review



$region_{(c_n,d)_m}$	$\delta(c_n, w_t)_m$	rating '1'	rating '2'	rating '3'	rating '4'	rating '5'	variance
bed is *clean* and cozy	("bed", "*clean*")	18	20	32	41	47	161.3
	("is", "*clean*")	24	28	41	27	45	87.5
	("*clean*", "and")	18	19	22	17	24	8.3
	("*clean*", "cozy")	3	13	22	48	54	491.1

Dist
$> min_{var}$
$dist^{(m)}_{(c_n,d)}$
m
Step 1) Generate combinations of all words
Step 2) Count co-occurrences and compute variances
Step 3) Select and store the rating distribution

Figure 4.5: Example process for identifying context words in a contextual region.

2. Count the number of times each combination in $\delta(c_n, w_t)_m$ co-occurs in the same region with each rating value on the entire training data and compute the variance from the frequency distribution of ratings.

3. Choose the combination that contributes the highest variance in rating distribution as context for $region(c_n, d)_m$. If no combination has a variance above min_{var}, c_n alone is considered as context for that region. Store the rating distribution of this combination, $dist(c_n, d)_m \in \mathbb{R}^{|Rating|}$ in the list of rating distributions *Dist* for index *m*.

Figure 4.5 illustrates the procedure for identifying the highest contributed variance combination of size $\theta = 2$ for the region "bed is *clean* and cozy.". Since, (*"clean"*, "cozy") yields the highest variance, it is chosen to represent context for this region.

4.5 Learning the Region Embeddings

From the previous step, the contextual regions *Region* and their associated rating distributions *Dist* were successfully extracted from the review data. They are now utilized for training the predictive model so that, given a contextual region $region(c_n, d)_m$ as an input, it predicts the rating distribution $dist(c_n, d)_m$ as an output. To achieve this, a model with an ability to identify those words in $region(c_n, d)_m$ that contribute to $dist(c_n, d)_m$ is required. CARE therefore adopt the model used for region embedding with local context proposed by Qiao et al. [77] as the training model. This technique learns two representations for each word, namely a word embedding of itself and a local context unit, which is a weight matrix for its interaction with its neighboring words. CARE aims to modifiy the local context unit to emphasize the positions of the words that have influence on the rating distributions and can therefore be considered as contexts for each contextual region.

The derived region embedding method for context extraction in CARE is shown in Fig. 4.6. This technique is a simple feedforward neural network model that takes a text region $region(c_n, d)_m$ as an input and produces a rating distribution vector $dist(c_n, d)_m$ as an output. Every word $w_t \in Vocab$ is mapped to its word embedding, whereas only a candidate context word $c_n \in Cand$ is mapped to its local context unit, which is used to produce the projected word embeddings. Finally, a region embedding, which is a representation of a text region, is generated from the projected word embeddings for use in computing a rating distribution.

Formally, every word w_t has an associated word embedding $\mathbf{e}_{w_t}$, which is stored in the column of the embedding matrix $\mathbf{E} \in \mathbb{R}^{h \times |Vocab|}$, where h is the embedding size and *Vocab* is the vocabulary of all words in the training data. In addition to the word embeddings, a candidate context word c_n also has an associated local-context unit matrix $\mathbf{K}_{c_n} \in \mathbb{R}^{h \times (2 \times d+1)}$, which is stored in the tensor $\mathbf{C} \in \mathbb{R}^{h \times (2 \times d+1) \times |Cand|}$.

Given a contextual region $region(c_n, d)_m$ as an input, the projected word embedding $\mathbf{p}_{w_t}$ of word w_t at index position l of $region(c_n, d)_m$ is calculated by

$$\mathbf{p}_{w_t} = \mathbf{K}_{c_n, l} \odot \mathbf{e}_{w_t}. \tag{4.5.1}$$

A word embedding $\mathbf{e}_{w_t}$ of word w_t at position l of $region(c_n, d)_m$ is projected into the region of the candidate context word c_n by element-wise multiplication with the corresponding column l of $\mathbf{K}_{c_n}$. This indicates that $\mathbf{e}_{w_t}$ can alter the semantic meaning of c_n. For example, $w_t =$ “very” in the region of $c_n =$ “clean” yields a positive meaning for the region, whereas $w_t =$ “not” would result in a negative meaning.

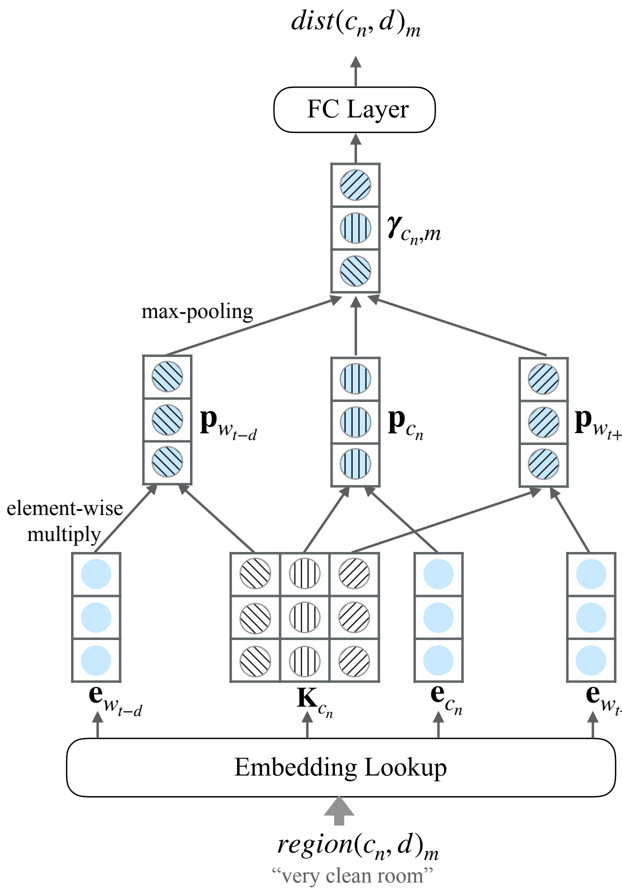


Figure 4.6: Illustration of the proposed context extraction model for a contextual region of size = 3.

After obtaining all projected word embeddings, the region embedding $\boldsymbol{\gamma}_{c_n,m} \in \mathbb{R}^h$ of a contextual region $region(c_n, d)_m$ is computed by

$$\boldsymbol{\gamma}_{c_n,m} = max([\mathbf{p}_{w_{t-d}} \ \dots \ \mathbf{p}_{c_n} \ \dots \ \mathbf{p}_{w_{t+d}}]), \tag{4.5.2}$$

where *max* is a max pooling operation over all projected word embeddings, which is applied to extract the most predictive features in the region [77].

This indicates that the meaning of $region(c_n, d)_m$ is now defined semantically by the meaning of neighboring words w_t with respect to the candidate context word c_n. For example, the two contextual regions "very clean room" and "not clean room" would give totally different region embeddings for the same c_n = "clean."

Finally, $\boldsymbol{\gamma}_{c_n,m}$ is fed into the fully connected layer to calculate the rating distribution $dist(c_n,d)_m$. Its objective is to predict a vector of rating distributions and a multivariate linear-regression model is adopt for the prediction, as expressed by

$$dist(c_n,d)_m \approx \mathbf{W}_f \cdot \boldsymbol{\gamma}_{c_n,m} + \mathbf{b}_f. \tag{4.5.3}$$

Here, $\mathbf{W}_f \in \mathbb{R}^{|Rating| \times h}$ and $\mathbf{b}_f \in \mathbb{R}^{|Rating|}$ are the weight matrix and bias vector in the fully connected layer, respectively, where $|Rating|$ is the size of the categorical rating scores (e.g., $|Rating|$ = 5 for a five-point rating score). The L2 was chosen as the loss function, following Qiao et al. [77], and Adam [27] as the optimizer. No regularization was applied.

After all model parameters are learned, each contextual region $region(c_n,d)_m$ can now be mapped with its region embedding representation $\boldsymbol{\gamma}_{c_n,m}$. Such a region embedding is trained to capture the global influence, i.e., the influence on the rating distribution of the entire review dataset, of its associated contextual region. This means that if two region embeddings are similar in the embedding space, they will be expected to contribute similar rating distributions.

4.6 Experiment Settings

The review datasets from multiple recommendation domains were used in the experiments. The first was from TripAdvisor[1], which contains hotel review data. The second dataset was from Yelp [2], which also contains hotel and restaurant reviews. Finally, six categories of Amazon 5-core datasets[3] dataset [69], including Fashion, Grocery & Food, Software, Toys & Games, Digital Music, and Movies & TV were incorporated.

First, the review text from all dataset were preprocessed by the following steps:

1. Tokenize the review text and convert all words to lower case.
2. Remove all punctuation marks and infrequently used words (i.e., those of appearance frequency below 0.01% in all reviews)
3. Remove all stopwords listed by NLTK[4], except for those indicating sentiment meanings such as "very" or "not."

[1]http://www.cs.cmu.edu/ jiweil/html/hotel-review.html
[2]https://www.yelp.com/dataset
[3]https://nijianmo.github.io/amazon/index.html
[4]https://www.nltk.org/nltk_data/

Table 4.1: Statistics of review datasets from multiple recommendation domains

	Reviews	Vocab. Size	Word/Reviews	Candidates
TripAdvisor	873,214	22,353	82.984	226
Yelp	4,735,962	18,069	62.383	199
A. Fashion	2,917	1,438	15.404	187
A. Grocery & Food	1,065,735	8,351	22.766	84
A. Software	12,405	35,103	104.193	257
A. Toys & Games	1,702,027	9,247	24.451	110
A. Digital Music	144,969	10,442	20.307	134
A. Movies & TV	3,166,002	20,949	47.472	129

After the preprocessing, reviews of less than two words were marked as uninformative and were therefore discarded. The statistics of the preprocessed datasets are presented in Table 4.1.

To extract a list of candidate context words, a word-rating co-occurrence matrix is firstly created for each dataset. Here, the main problem is that many datasets contain biases in the proportion of ratings provided by users. For example, in the TripAdvisor dataset, more than 80% of all reviews were rated as “4” or “5”, meaning that most users preferred to provide high rating scores to most hotels. This causes almost every word in the corpus to be distributed toward high rating scores, as shown by the example in Fig. 4.7 (a). This is in contrast to the fact that reviews containing words such as “rude” should be rated with a low rating score, rather than a high rating score. To properly analyze the actual influence of a word on the rating distribution, a data standardization technique is therefore applied, as expressed by

$$x_{t,r}^{new} = \frac{x_{t,r} - \mu_r}{\sigma_r}, \tag{4.6.1}$$

where $x_{t,r}$ is the original frequency of word w_t given for rating r, μ_r is the average of the frequencies of all words given for rating r, and σ_r is the standard deviation of the frequencies of all words given for rating r. The rating distribution after applying this standardization is shown in Fig. 4.7 (b). The frequencies of ratings with the word “rude” are now distributed toward low rating scores, which is appropriate to its negative meaning.

After standardization, the variance of the rating distribution of each word was computed, and the words with variances exceeding $min_{var} = 1$ were selected as a set of candidate context words for each dataset. The numbers of such words extracted from each dataset and the

word	rating ‘1’	rating ‘2’	rating ‘3’	rating ‘4’	rating ‘5’
“rude”	32373	35551	60391	103517	98043

(a)

word	rating ‘1’	rating ‘2’	rating ‘3’	rating ‘4’	rating ‘5’
“rude”	35.47	34.41	31.09	23.78	19.54

(b)

Figure 4.7: Rating distributions for example words: (a) before standardization, (b) after standardization.

average number for each review are given in Table .... To extract the contextual regions for each candidate context word, the region size was set to five and a padding of length d = 2 was applied to the head and tail of every review (a candidate context word might be the first or last word in a review). Because some candidate context words might be associated with millions of contextual regions, using all of them for training could cause scalability problems. In fact, this is unnecessary because using only a sampled portion can cover all unique patterns in the rating distributions. A criterion for sampling a subset of the contextual regions for a candidate context word c_n was therefore required. Specifically, let $Region_{c_n}$ denote the set of all contextual regions of c_n. If $|Region_{c_n}| > 100$k, only a 10% subset was used for training. If $10\text{k} \leq |Region_{c_n}| \leq 100\text{k}$, 10k were used. If $|Region_{c_n}| < 10$k, all were used. To assign a rating distribution to each contextual region the size of word combination was set to $\theta = 2$, and those with variances exceeding $min_{var} = 1$ were selected.

In the training process, the word embeddings $\mathbf{E}$ and local context units $\mathbf{K}$ were initialized randomly in terms of a uniform distribution with values between -1 and $+1$. The embedding size h for all datasets was set to 300. The learning rate was optimized from {0.0001, 0.001}, and the batch size was selected from {128, 256, 512, 1024, 2048}, using a validation set.

4.7 Results and Discussion

In this section, the list of contexts extracted from various recommendation domains is first analyzed. Next are the visualization and discussion about the influences of the extracted contexts on the rating distributions. Moreover, an analysis of the quality of the region embeddings learned by CARE is conducted to investigate its suitability for the rating prediction task. Furthermore, the merits of defining and extracting contexts as contextual regions are analyzed in terms of an illustrative example and a sentiment classification task. Finally, a discussion on the applications suitable for CARE is conducted in detail.

Table 4.2: Examples of candidate context words extracted from the TripAdvisor and Yelp datasets.

TripAdvisor		Yelp	
area	friendly	area	helpful
bathroom	helpful	atmosphere	manager
breakfast	manager	business	professional
central	night	clean	reasonable
clean	noisy	delicious	restaurant
comfortable	restaurant	dinner	rude
convenient	rude	family	service
dirty	service	friendly	staff
downtown	staff	happy	tasty

Table 4.3: Examples of candidate context words extracted from six categories within the Amazon Product dataset.

Fashion	Grocery & Food	Software	Toys & Games	Digital Music	Movies & TV
comfy	awful	buggy	birthday	album	acting
dancing	best	computer	broken	beautiful	action
fit	coffee	crashes	christmas	catchy	boring
lightweight	delicious	digital	cute	classic	cast
looking	disappointed	features	daughter	collection	character
training	flavor	installation	fun	download	dvd
tightly	fresh	interface	game	instrument	enjoy

great	good	very	nice	love	not	but	however	even	all

4.7.1 Context Analysis

This subsection aims to show that by defining contexts in reviews as words that influence the distribution of ratings, CARE has the flexibility to extract contexts from review data across a variety of recommendation domains. To achieve this, the list of candidate context words extracted from multiple review datasets across different domains are analyzed, including TripAdvisor and Yelp (hotel and restuarant), and Amazon Product (Fashion, Grocery & Food, Software, Toys & Games, Digital Music, and Movies & TV). Examples of candidate context words extracted for each dataset are given in Tables 4.2 and 4.3.

First, the list of candidate context words extracted from TripAdvisor and Yelp, which involve similar recommendation domains are analyzed. As highlighted in Table 4.2, CARE was able to discover mutual words across these two datasets, such as “area”, “clean” and “friendly”, which indicate the user preferences toward the hotel features. This demonstrates that CARE has a generalized ability to extract a similar set of contexts from different datasets in similar domains. Table 4.3 gives a list of candidate context words extracted across six categories in the Amazon Product dataset. These results indicate that CARE is capable of extracting exclusive words that are strongly related to each domain. Examples include “fit” for fashion, “delicious” for food, “catchy” for music, and “acting” for movies. This demonstrated that CARE would have the flexibility to extract contexts from many kinds of review-rating datasets, independent of the dataset’s domain. Moreover, in addition to these exclusive words from particular domains, CARE was able to extract sentiment and polarity words such as “great”, or “not” from across all the domains. This supports the assumption that these words also influence the distribution of ratings and should therefore accompany the context words used in making rating predictions.

4.7.2 Influences of Candidate Context Words and Their Neighboring Words

As discussed in Section 4.4, their neighboring words might alter the influence of candidate context words on their rating distributions. To analyze such influences, I follow [77] by applying the L2-norm to each column of the local context unit. This enables the influence levels of the candidate context and their neighboring words to be emphasized, as shown in Figure 4.8. For example, words that follow “staff” and “very” have more influence on rating distributions than the words that come before them. This corresponds to the following words often being “good,” “helpful” or “friendly” for “staff,” and “clean,” “convenient” or “comfortable” for “very.” On the other hand, words such as “breakfast” are less influenced by neighboring words, meaning that the word itself sufficiently describes the rating distributions without any help from neighboring words. Moreover, the local context units can differentiate the influence of positive words such as “good” or “excellent.” Although the rating distributions of “good” are influenced by its neighboring words, the word “excellent” is not. This is because the word “excellent” itself indicates the strongest positive meaning, whereas the semantic meaning of “good” can be altered if it follows words such as “not” or “very.”

For these reasons, we see that the local context units can capture the influences of the candidate context words efficiently, together with their neighboring words, on the rating distributions. This further helps to produce high-quality region embeddings, which are capable of semantically representing the distribution of ratings for the individual contextual regions.

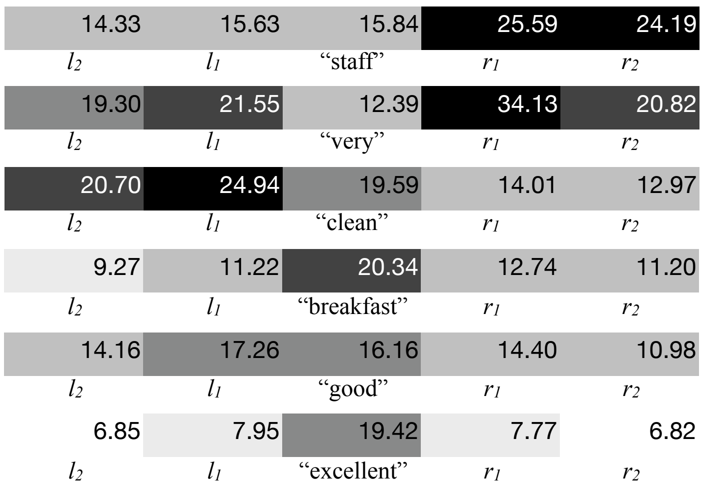


Figure 4.8: Visualization of the local context units for some chosen candidate context words.

4.7.3 Embedding Analysis

The previous subsection showed that CARE is able to extract words representing contexts from multiple recommendation domains. This subsection aims to show that the region embeddings, which are generated from context words and their neighboring words, accurately capture the rating distributions of their corresponding contextual regions and are therefore useful for rating prediction. The assumption is that contextual regions that contribute similar rating distributions should generate region embeddings that are close to each other in the embedding space.

To investigate this assumption, I first define a method for categorizing the distributions of ratings into classes. The idea is to assign a class to each rating distribution based on its direction (positive or negative). For example, the frequencies of ratings in the distribution $dist(c_n,d)_1$ = [8, 25, 34, 56, 95] are positively distributed toward high rating scores, whereas those of $dist(c_n,d)_2$ = [103, 75, 41, 18, 3] are negatively distributed toward low rating scores. The $dist(c_n,d)_1$ would then be categorized as belonging to a positive class, whereas $dist(c_n,d)_2$ should belong to a negative class. To implement this categorization, Pearson correlation coefficient was chosen to compute a correlation score between the rating distribution and an ordinal rating vector, as expressed by

$$\rho_{dist(c_n,d)_m,score_R} = \frac{\text{cov}(dist(c_n,d)_m, score_R)}{\sigma_{dist(c_n,d)_m}\sigma_{score_R}}, \tag{4.7.1}$$

Table 4.4: Criteria for categorizing a rating distribution based on correlation score.

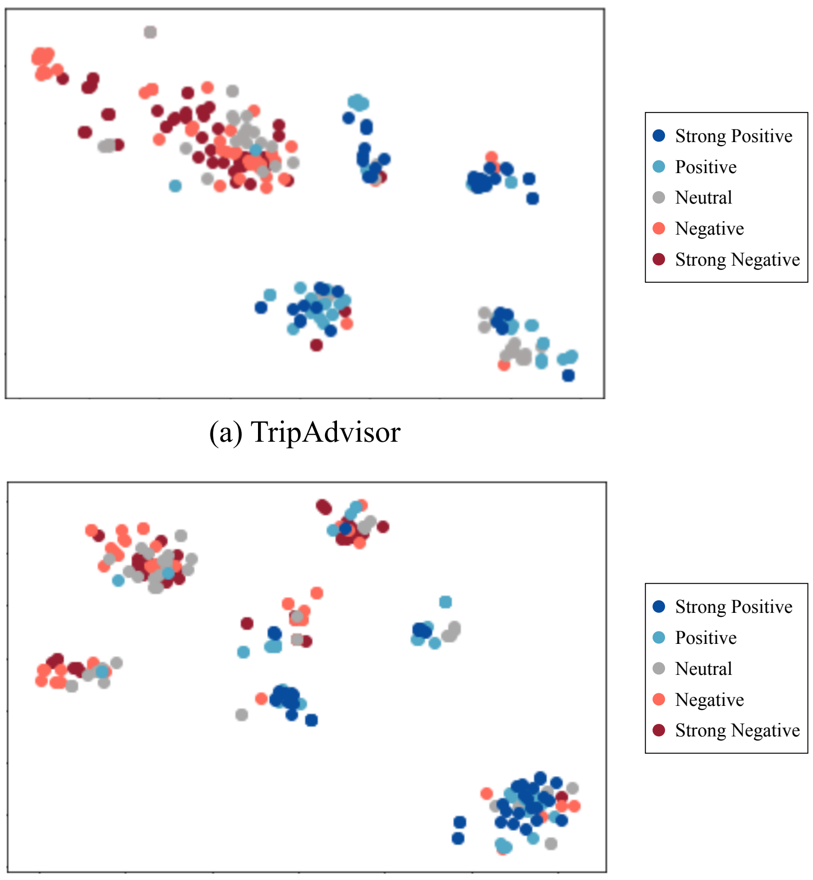
Correlation	Class
$\rho \geq 0.9$	Strong Positive
$0.4 \leq \rho < 0.9$	Positive
$-0.4 < \rho < 0.4$	Neutral
$-0.9 < \rho \leq 0.4$	Negative
$\rho \leq -0.9$	Strong Negative

where cov denotes the covariance function , σ is a standard deviation, and $score_R \in \mathbb{Z}^{|Rating|}$ is an ordinal rating score vector (sorted in ascending order) for which $|score_R| = |dist(c_n, d)_m|$. For example, $score_R = [1,2,3,4,5]$ could be used for rating data via a five-point rating score.

After computing the correlation score for each $dist(c_n, d)_m$ using $score_R$, we can then assign it to a class by using the categorization criteria given in Table 4.4, where the rating distributions were categorized into five classes: Strong Positive, Positive, Neutral, Negative and Strong Negative.

To visualize the subtle differences between the region embeddings, the contextual regions from the TripAdvisor and Amazon Movies & TV datasets, were sampled and categorized based on their corresponding rating distributions and generating their region embeddings. Figures 4.9 and 4.10 were obtained by applying t-distributed stochastic neighbor embedding (t-SNE) [95] to the sampled region embeddings from each dataset, where the color of each point denotes the class of its associated rating distribution. In Figure 4.9, for each dataset, 50 contextual regions from each class (250 in total) were sampled and their corresponding region embeddings were plotted. Note that the group of region embeddings representing positive and negative classes is fairly distinguishable. This supports the assumption that contextual regions with similar rating distributions are mapped close to each other in the embedding space.

The region embeddings can be analyzed in more detail by visualizing those that are associated with the contextual regions of each candidate context word. As shown in Figure 4.10, two candidate context words, “location” from TripAdvisor and “acting” from Amazon Movies & TV were selected; 10 contextual regions that contained them from each class (50 in total) were sampled and their corresponding region embeddings were plotted. Note that words that contribute positive distributions such as “great”, “good”, or “excellent” are grouped close to each other and are visually separated from negatively distributed words such as “not”, “bad”, or “but”. This again supports the assumption that neighboring words in



(a) TripAdvisor

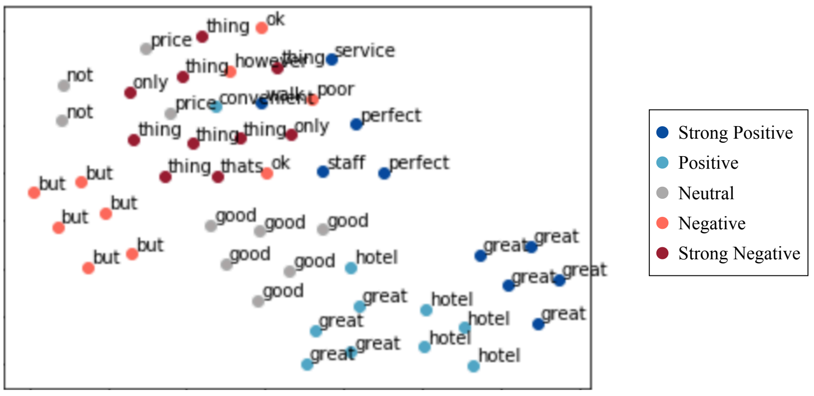
(b) Amazon Movies & TV

Figure 4.9: Projection of sampled region embeddings for the TripAdvisor and Amazon Movies & TV datasets.

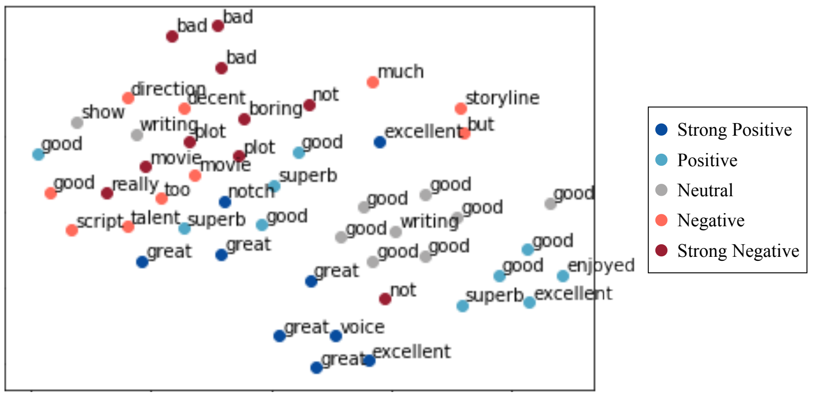
the same text region as a candidate context word influence the distribution of ratings, and should be considered when extracting contextual information from reviews.

4.7.4 Contexts as Regions

The previous subsection demonstrated that the region embedding representations of contextual regions were effective in capturing their associated rating distributions. In this subsection further analyzes the merits of defining and extracting a context as a contextual region. First, we show that the rating distributions captured by contextual regions explain the polarity of a review more effectively than those captured by single words. I investigate this assumption by utilizing the embeddings of single candidate context words and the corresponding contextual regions for review sentiment classification.



(a) Word “location”

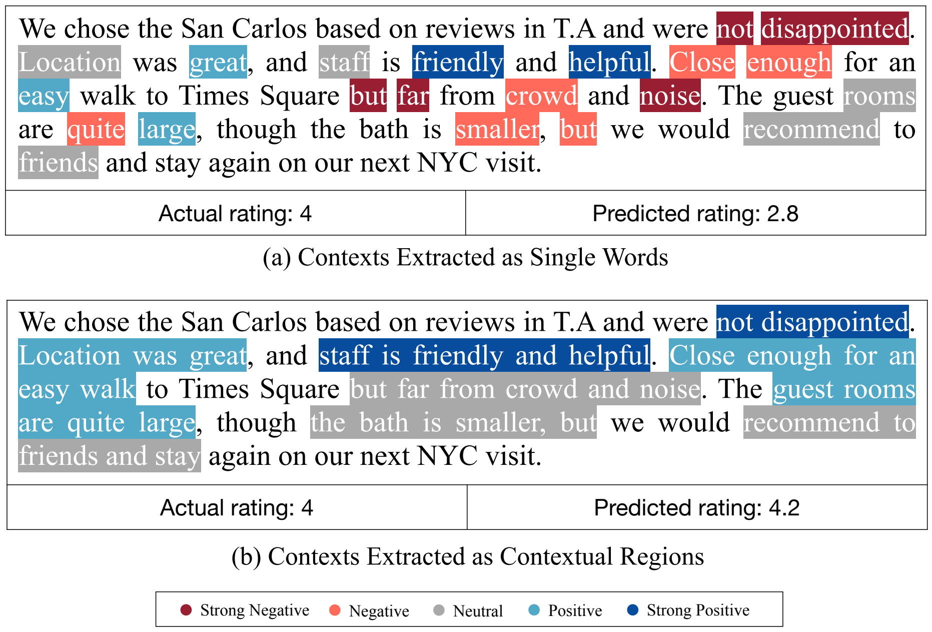


(b) Word “acting”

Figure 4.10: Projection of sampled region embeddings associated with two candidate context words.

Review Polarity

Figure 5.8 (a) in Section 5.7.3 showed that utilizing a contextual region of size = 1 (i.e., considering only candidate context words not influenced by neighboring words) produced the least accurate rating prediction among those for a range of region sizes. This indicated that defining a context as a single word is less effective in modeling a review’s rating than defining it as a text region. To support this finding, the goal here is to visualize the difference in influence on a review’s polarity between considering a context as a single word and considering it as a text region. To demonstrate this, CARE was applied to an example review from the TripAdvisor dataset for the two types of context, as shown in Figure 4.11. Figure 4.11 (a) shows the contexts extracted as single candidate context words, whereas Fig. 4.11 (b) shows them extracted as contextual regions. The highlight colors reflect the class of the associated rating distribution for each context, as defined in Section 4.7.3.



We chose the San Carlos based on reviews in T.A and were not disappointed. Location was great, and staff is friendly and helpful. Close enough for an easy walk to Times Square but far from crowd and noise. The guest rooms are quite large, though the bath is smaller, but we would recommend to friends and stay again on our next NYC visit.

Actual rating: 4	Predicted rating: 2.8

(a) Contexts Extracted as Single Words

We chose the San Carlos based on reviews in T.A and were not disappointed. Location was great, and staff is friendly and helpful. Close enough for an easy walk to Times Square but far from crowd and noise. The guest rooms are quite large, though the bath is smaller, but we would recommend to friends and stay again on our next NYC visit.

Actual rating: 4	Predicted rating: 4.2

(b) Contexts Extracted as Contextual Regions

● Strong Negative ● Negative ● Neutral ● Positive ● Strong Positive

Figure 4.11: Example of a review with extracted contexts highlighted to show their rating distribution classes.

Consider first the contexts extracted as single words in Fig. 4.11 (a). Note that many extracted words in this review such as “not”, “disappointed’, “crowd”, or “noise” were classified as contexts with negative rating distributions. Utilizing those words individually could negatively affect the polarity of the review, and consequently result in a low rating prediction score, which does not accord with the actual rating score. This implies that extracting contexts as single words might not adequately explain the polarity of a review. This can happen because some single words fail to capture the actual rating distribution patterns of contexts. In fact, their combination with some neighboring words could radically alter their rating distribution pattern and lead to a totally different interpretation for a rating prediction score. For example, “not disappointed” is classified as a context with a positive rating distribution even though both “not” and “disappointed” are associated with negative rating distributions.

By extracting contexts as contextual regions, a more appropriate rating distribution class to each context is assigned. As illustrated in Figure 4.11 (b), some contextual regions such as “not disappointed”, “close enough for an easy walk”, and “but far from crowd and noise” are no longer assigned with negative rating distributions. Many positive rating distributions could positively affect the polarity of this review and produce a high rating score that is closer to the actual rating score. From this example, we can hypothesize that extracting contexts as contextual regions is more effective in explaining the polarity of a review than extracting

Table 4.5: Classification results for the single word-context and the contextual-region models.

Dataset	Context	Classification	
		Accuracy	F1 Score
Amazon Software	Single	0.71861	0.82371
	Region	**0.74361**	**0.83584**
TripAdvisor	Single	0.79399	0.85615
	Region	**0.83858**	**0.89189**
Amazon Movies & TV	Single	0.81664	0.88300
	Region	**0.84097**	**0.89842**

them as single words. We can investigate this hypothesis by evaluating the comparative performance using single context words against that using contextual regions for the review sentiment classification task.

Review Sentiment Classification

Here, the goal is to demonstrate that, in addition to their use in making rating predictions, the use of contextual regions is more effective in modeling the polarity of a review than the use of single context words. This demonstration involves the sentiment classification of reviews.

For comparison, I set up two models for review classification, namely a "single" and a "region" model. For the single model, I utilized the word embeddings of all candidate context words learned by CARE for region size = 1. The word embeddings of all candidate context words in each review was then averaged to create an embedding representation for that review. For the region model, the region embeddings for all corresponding contextual regions were computed, using word embeddings and local context units learned by CARE for region size = 5. A representation of each review was then created by averaging all region embeddings of all contextual regions in that review.

To evaluate the classification accuracy of these two models, a logistic regression was chosen as a binary classifier of the review representations for both the single and region models. All reviews from all three datasets (Amazon Software, TripAdvisor, and Amazon Movies & TV) having rating scores of more than 3 were labelled as "positive" reviews, and those with scores less than or equal to 3 were labelle as "negative" reviews. The classification results for the single and region models on all datasets are given in Table 4.5. These results show that the region model achieves a higher classification accuracy and F1 score for all three datasets compared with the single model. This supports the hypothesis that considering contexts as contextual regions is helpful in explaining the polarity of a review.

4.7.5 Applications for CARE

In this chapter, it has been shown that the main goal of CARE is to extract contextual information from reviews. However, the information extracted by CARE is not necessary limited only to contexts. By inspecting the list of extracted words, CARE is also able to extract the following valuable review elements:

- Feature opinions: the words relating to the features of an item, which are unique to each specific review domain. For example, “bathroom” or “wifi” are extracted from hotel reviews, “lightweight” or “training” from fashion item reviews, “fresh” or “flavor” from food reviews, and “action” or “horror” from movie reviews.

- Aspect opinions: some of the feature opinions extracted by CARE can be grouped further into the certain aspects or criteria of an item. For example, the words “central”, or “downtown” are related to the location aspect of a hotel, whereas “friendly” or “rude” are related to the service aspect. These aspects help in emphasizing the highlighted characteristics of each item, as compared with the other items in the same system.

- Sentiment words: by extracting words having significant influences on the rating distributions, a portion of them might capture a user sentiment orientation (i.e. positive, negative, or neutral) toward items. The example of positive sentiment words include “good”, “great”, “nice” or “love”, whereas negative sentiment words include “not”, “never”, “but”, or “however”, and the neutral sentiment words might include “ok” or “quite”.

- Frequent and useful words: with a merit of utilizing rating distributions as training labels, the extracted words by CARE are not only frequent, but also useful. First, in order to identity the words having significant influences on the rating distributions, such words need to be written in reviews with significant number of times. Therefore, applying CARE also indirectly remove the infrequent words from reviews. However, not all frequent words are always useful for rating prediction task. For example, in hotel recommendation, a word like “clean” should have more impact on a review’s rating than a common word like “hotel”. Fortunately, the frequent words extracted by CARE are emphasized on the ones having significant influences on rating distributions. Since the frequencies of “clean” tend to be positively distributed toward high ratings, whereas the frequencies of “hotel” are neutralized on all ratings, only “clean” is then extracted by CARE.

With these review elements being extracted, CARE can also be applied to a certain kind of applications, in addition to a context-aware recommendation. The following are two examples of the suitable applications for CARE.

- Content-based recommendation: the simplest application that can make use of CARE is a content-based recommendation. The main goal of a content-based recommendation is to recommend items having similar features to those previously liked by the user. One challenge of a content-based recommendation is that some items might lack of explicit item features, which leads to a difficulty in making recommendations. With the ability to extract feature opinions from reviews, CARE can easily build a user profile from the extracted relevant item features from his/her previous reviews, and use it for future recommendations. For example, if CARE can extract the word “action” with a significant number of times from the user previous movie reviews, this would mean that this user might have special concern on action movies. The system can then use this as a guideline by focusing on movies related to action genre when making future recommendation for this user.
- Multi-criteria recommendation: another recommender system application that could be benefitted from CARE is a multi-criteria recommendation. In multi-criteria recommender system, users can provide ratings in multiple aspects of an item (such as location or service of a hotel), which are utilized by the system to make more effective recommendation. However, the multi-criteria rating system is not very common since most systems only allow the user to provide a single overall rating to an item. Fortunately, the aspect opinions extracted by CARE might be helpful in representing multi-criteria preferences from users, which can be a replacement of the actual multi-criteria ratings. Recall that CARE can associate each extracted text region with its corresponding rating distribution, the idea is that this rating distribution might be useful for inferring the preference level of the user in each aspect of an item. For example, suppose that the text regions containing “nice location” are associated with a highly positive rating distribution. If a review of one hotel is written with “nice location”, we can then inferred that the user who write this review implicitly assign high rating score on the location aspect to this hotel. By analyzing a set of reviews written by each user, we might be able to model the user preference profile for each aspect of an item. Similarly, the item preference profile for each aspect of each item could be modeled by analyzing a set of reviews provided to that item. The system can then suggest

the items having similar profile in each aspect to the user as a recommendation (e.g. recommending the hotel with nice location to the user who concerns about location aspect).

The utilization of contextual information extracted by CARE for context-aware recommender system application, on the other hand, will be presented in detail in the next chapter.

4.8 Conclusion

In this chapter, a novel unsupervised method for extracting relevant contexts from reviews, namely CARE, is proposed. Unlike any previous context-aware methods, a relevant context of CARE can be automatically extracted not only in single word format but also in combination with those neighboring words from the same text region that influence the distributions of ratings. This makes CARE applicable to a wide variety of recommendation domains and suitable for extracting contexts from sets of reviews that may involve a variety of styles. Experiments showed that the CARE was able to extract the meaningful list of contexts that are relevant to specific recommendation domains. Furthermore, the extracted contexts are represented with high-quality region embeddings, which effectively capture the polarity of reviews, and which can be useful for explaining the reviews' ratings.

Chapter 5

Context-Aware User and Item Representations Based on Unsupervised Context Extraction from Reviews

5.1 Introduction

User-generated reviews can supply valuable information to mitigate the rating sparsity problem that can occur in the standard collaborative filtering-based (CF-based) recommendation approach, which utilizes rating data alone [21, 64, 97]. In particular, recent work has employed deep learning techniques and attention mechanisms to learn representations of users and items from reviews and use them for rating prediction [19, 47, 86, 103].

However, as mentioned in Chapter 2.3.4, constructing a representation using those techniques has two common limitations. First, words that are not related to a user preferences or an item features are (but should not be) used in constructing the representations. Second, to predict a rating for a review $y_{i,j}$, the representations of u_i and v_j are not (but should be) dynamically constructed by emphasizing the relevant information in the review, which explains the reason behind its rating, rather than relying on information from other reviews.

Extracting and exploiting contexts from reviews could be the key to overcoming these two limitations. Context words such as “friend” or “summer” are often related to a user preferences and an item features, and are therefore appropriate for use in constructing representations. Moreover, contexts help characterize the situation within which the rating was being given, which is unique and specific for each review. This means that user and item representations can be constructed dynamically by considering the contexts embedded in each particular review.

In Chapter 4, the CARE, an unsupervised method for defining and extracting contexts from reviews is proposed [91]. A relevant context is defined not by a single word alone but by the word plus those of its neighboring words that influence the distributions of ratings. A region embedding technique [77] is derived to emphasize the words in a small text region for consideration as a context, and represent it by region embedding. By not having to predefine contexts, CARE can be used to extract relevant contexts from reviews in any recommendation domain.

In this chapter, an extension of CARE which utilizes the extracted contexts for rating prediction is proposed [91]. The extended model, namely *attentional interaction model for context-aware region embedding* (CARE-AI), derives region embedding representations for the extracted contexts that are output from CARE. These are then input to the proposed rating prediction procedure, which contains two neural network modules for interaction and attention. The interaction module first models the relevance of each context in a particular review based on its past interaction with an individual user preferences and item features. The attention module then generates the user and item representations based on different relevance levels among contexts in each review. These two modules enable the model to dynamically construct unique user and item representations for each specific review, rather than have one static representation for all reviews, as found in most deep-learning-based methods. Finally, the user and item representations are used to predict ratings by exploiting a latent factor model [50].

The experiments on three well-known review datasets demonstrated that CARE-AI outperforms existing state-of-the-art rating prediction methods that include both review-based and context-aware recommendation techniques [19, 57, 75, 81, 101, 103]. In addition, the performance of CARE in the various aspects, including the effectiveness of interaction and attention modules, the parameter sensitivity, the impact of review quality, and the performances on sparse data, are discussed in detail.

The main contributions of CARE-AI can be summarized as follows.

- The user and item representations of CARE-AI are dynamically constructed for each particular review to effectively capture the contextual information embedded in that review.
- The proposed interaction and attention modules help modeling the different relevance levels among the contexts in a review to the individual user preferences and item features.
- Finally, CARE-AI produced more accurate prediction than the state-of-the-art review-based recommendation techniques on both normal and sparsity situations.

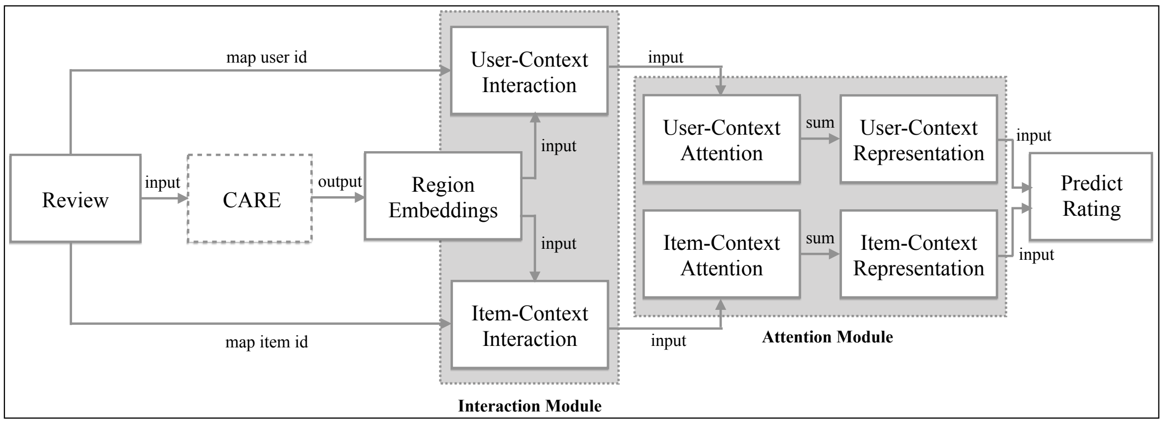


Figure 5.1: An overview of the workflow of CARE-AI.

5.2 Model Overview

In this section, the overview of CARE-AI model, including the workflow and the model architecture, are presented,

5.2.1 Workflow

Figure 5.1 presents an overview of the workflow of CARE-AI. This model first derives the region embeddings, which represent a review's contextual regions extracted by CARE as an input. The user and item representations are then dynamically constructed from the region embeddings via the proposed interaction and attention modules. The interaction module models the relevance of each context to the individual user and item by its past interaction with the user preferences and the item features. The attention module then generates user and item representations based on the different relevance levels among contexts extracted from a particular review. Finally, these representations are used in a latent factor model that predicts the rating.

5.2.2 Model Architecture

To utilize the region embeddings for making personalized rating predictions, it is important to model the influence of each contextual region on the user preferences and the item features, which are used in determining a particular review's rating. To achieve this, two important aspects of each contextual region, namely its relevance to each user preferences or item features and its contribution to a particular review's rating, are needed to be considered.

First, the relevance of each contextual region to a user's personal preferences depends on how it has previously been expressed by that user in previous reviews. For example,

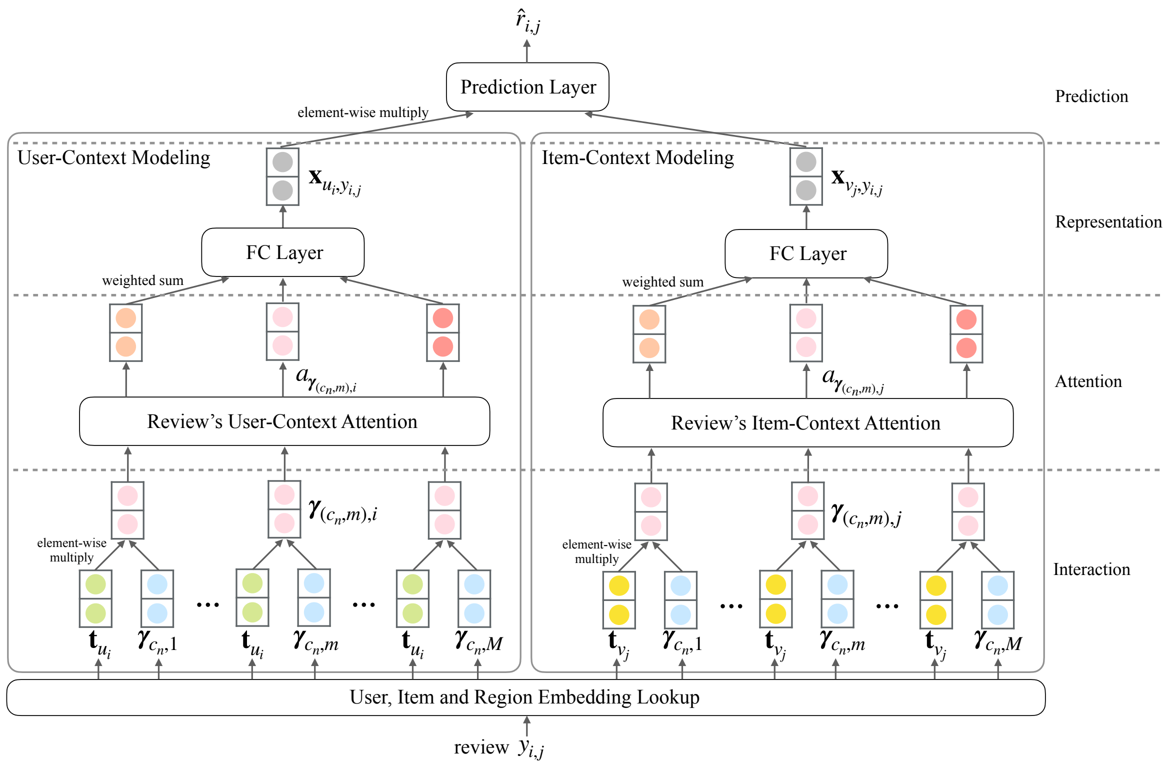


Figure 5.2: Illustration of the CARE-AI model.

many of the user's hotel reviews might have contained words such as "cheap", "expensive", or "worth", whereas "small", "large", or "spacious" might have been used less often. The implication is that this user is highly interested in the price when choosing a hotel, but is less concerned about the size of the room. Therefore, those contextual regions containing words related to the price of a room should be more relevant to this user preferences than those containing words related to the size of a room. The same assumption can also be applied to the relevance of contextual regions to the item's unique features. Depending on what has been frequently described in their reviews, some hotels, for example, might be famous for their service, whereas others are better known for their convenient location.

Moreover, a review usually contains more than one contextual region and different contextual regions might have unequal influences on the user's decision about the item, which would consequently affect the rating. I believe that, the more relevant a contextual region is to each individual user preferences or item features, the more it should contribute to the rating of a particular review compared with the other regions within the same review.

By properly analyzing the relevance and the contribution made by contexts in a particular review, the user and item representations can be dynamically constructed specifically and uniquely for that review. To implement this, the CARE-AI model, whose architecture is shown schematically in Fig. 5.2, is proposed. CARE-AI is composed of two parallel neural networks, one each for user-context and item-context modeling. The model takes a review of user u_i on item v_j, denoted by $y_{i,j}$, that contains M contextual regions, denoted by $Region(y_{i,j})$, as an input. By looking up its corresponding word embeddings and local context unit (learned from the context extraction step), a region embedding for each contextual region is generated. The region embeddings are then fed into the user-context and item-context modeling networks. Each of these networks comprises two modules, namely an interaction module that models the relevance of contextual regions and an attention module that learns the contributions of those regions to a review's rating. Finally, the outputs of the user-representation and item-representation networks are fed into the prediction layer, which generates the final prediction of a review's rating by using a latent factor model.

5.3 Interaction Module

To model the relevance of contextual regions to user preferences and item features, a *user-context interaction* matrix $\mathbf{T}_u \in \mathbb{R}^{|User| \times h}$ and a *item-context interaction* matrix $\mathbf{T}_v \in \mathbb{R}^{|Item| \times h}$ are introduced. Each row $\mathbf{t}_{u_i} \in \mathbf{T}_u$ and $\mathbf{t}_{v_j} \in \mathbf{T}_v$ contains a vector representing the interaction with the contextual regions for user u_i and item v_j, respectively. To fully capture the interactions with the contextual regions, the dimensionalities of $\mathbf{t}_{u_i}$ and $\mathbf{t}_{v_j}$ are set to h, which is the dimensionality of the region embedding. The interaction of contextual region $region(c_n, d)_m \in Region(y_{i,j})$ with user u_i and item v_j are then modeled by using element-wise multiplication between its region embedding $\boldsymbol{\gamma}_{c_n,m}$ and $\mathbf{t}_{u_i}$ or $\mathbf{t}_{v_j}$, respectively, as expressed by

$$\boldsymbol{\gamma}_{(c_n,m),i} = \mathbf{t}_{u_i} \odot \boldsymbol{\gamma}_{c_n,m}, \qquad \boldsymbol{\gamma}_{(c_n,m),j} = \mathbf{t}_{v_j} \odot \boldsymbol{\gamma}_{c_n,m}. \tag{5.3.1}$$

The vectors $\mathbf{t}_{u_i}$ and $\mathbf{t}_{v_j}$ can be considered as projection vectors for converting the region embedding $\boldsymbol{\gamma}_{c_n,m}$ to the *user-relevance* region embedding $\boldsymbol{\gamma}_{(c_n,m),i}$ and *item-relevance* region embedding $\boldsymbol{\gamma}_{(c_n,m),j}$, respectively. They are learned with the main objective of capturing previous interactions of a contextual region with each individual user preferences and a specific item features. If a contextual region $region(c_n, d)_m$ was mentioned a significant number of times in user u_i's reviews, its interaction with $\mathbf{t}_{u_i}$ will result in high values for $\boldsymbol{\gamma}_{(c_n,m),i}$, indicating that it is highly relevant to u_i's preferences.

After all region embeddings $\boldsymbol{\gamma}_{c_n,1},\cdots,\boldsymbol{\gamma}_{c_n,M}$ of $region(c_n,d)_1,\cdots,region(c_n,d)_M \in Region(y_{i,j})$ are converted into user-relevance and item-relevance region embeddings, they are fed into the attention module to compute their contributions to a rating of the review $y_{i,j}$.

5.4 Attention Module

The contribution of a contextual region $region(c_n,d)_m$ to a particular review's rating depends on its degree of relevance to the user preferences and the item features compared with the other regions in that review. Because a user-relevance region embedding $\boldsymbol{\gamma}_{(c_n,m),i}$ and an item-relevance region embedding $\boldsymbol{\gamma}_{(c_n,m),j}$ indicate the relevance of $region(c_n,d)_m$ to u_i's preferences and v_j's features, they can be utilized for modeling their contributions. To achieve this, attention mechanism is adopt since it has been successfully utilized in many deep-learning-based recommendation methods [19, 86]. Specifically, the attention network for modeling the contribution of $region(c_n,d)_m$ in a user-context modeling network is defined by

$$a^*_{\boldsymbol{\gamma}_{(c_n,m),i}} = \mathbf{W}^T_{attn} g(\mathbf{W}_{attn_u}\boldsymbol{\gamma}_{(c_n,m),i} + \mathbf{b}_{attn_u}) + b_{attn}, \tag{5.4.1}$$

where $\mathbf{W}_{attn_u} \in \mathbb{R}^{k_1\times h}$, $\mathbf{W}_{attn} \in \mathbb{R}^{k_1}$, $\mathbf{b}_{attn_u} \in \mathbb{R}^{k_1}$, and $b_{attn} \in \mathbb{R}$ are model parameters. The size of the hidden layer in the attention network is denoted by k_1 and g is a nonlinear activation function. A softmax function is then applied to compute a normalized attention score for a contextual region $region(c_n,d)_m$ with respect to the other M-1 contextual regions in review $y_{i,j}$, $Region(y_{i,j})$ as

$$a_{\boldsymbol{\gamma}_{(c_n,m),i}} = \frac{exp(a^*_{\boldsymbol{\gamma}_{(c_n,m),i}})}{\sum_{m=1}^{M} exp(a^*_{\boldsymbol{\gamma}_{(c_n,m),i}})}. \tag{5.4.2}$$

The attention scores are then used to compute the weighted sum of the user-relevance region embeddings, which are fed into a fully connected layer to create a user representation of user u_i that is specific to review $y_{i,j}$, $\mathbf{x}_{u_i,y_{i,j}}$, as expressed by

$$\mathbf{x}_{u_i,y_{i,j}} = \mathbf{W}_u \sum_{m=1}^{M} a_{\boldsymbol{\gamma}_{(c_n,m),i}} \boldsymbol{\gamma}_{(c_n,m),i} + \mathbf{b}_u, \tag{5.4.3}$$

where $\mathbf{W}_u \in \mathbb{R}^{k_2 \times h}$ and $\mathbf{b}_u \in \mathbb{R}^{k_2}$ are a weight matrix and bias vector, respectively, in a fully connected layer with a hidden layer of size k_2. This user representation is dynamically generated to give different levels of relevance among the contexts in $y_{i,j}$. A user who has rated N reviews will therefore have N review-based representations, rather than the one static representation across all reviews used in most deep-learning-based methods. (The computational details for the item representation $\mathbf{x}_{v_j,y_{i,j}}$ are omitted because they are very similar to those for the user representation.)

5.5 Prediction Layer

The user and item representations are now ready to be used for the final rating prediction task. To predict a rating, the latent factor model [50] which has been shown to be effective in many deep-learning-based prediction approaches [19, 47, 103], is adopt for modeling the interaction between $\mathbf{x}_{u_i,y_{i,j}}$ and $\mathbf{x}_{v_j,y_{i,j}}$. Specifically, the rating of user u_i toward item v_j is estimated by

$$\hat{r}_{i,j} = \mathbf{x}_{u_i,y_{i,j}}^T \mathbf{x}_{v_j,y_{i,j}} + b_{u_i} + b_{v_j} + \mu, \tag{5.5.1}$$

where b_{u_i}, b_{v_j}, and $\mu \in \mathbb{R}$ respectively denote the bias for user u_i, the bias for item v_j, and the global bias. If a contextual region in $y_{i,j}$ is highly relevant to both user u_i's preferences and item v_j's features, it should result in a high rating score. For example, (5.5.1) implies that a user who likes a city view at night will be recommended a hotel famous for its nighttime city view. The training is done using Adam [27] as an optimizer, and, to prevent overfitting, a dropout operation is applied on the hidden layer. The L2 is chosen as a loss function, with regularization expressed as

$$L = \sum_{(i,j)\in O} (r_{i,j} - \hat{r}_{i,j})^2 + \lambda_\Theta ||\Theta||^2, \tag{5.5.2}$$

where O denotes the set of observed user–item rating pairs, $r_{i,j}$ is the observed rating score of user u_i toward item v_j, and Θ denotes the model parameters.

5.6 Experimental Evaluation

5.6.1 Data Preparation

Three publicly available review datasets were used for conducting the experiments. The first was from TripAdvisor[1], which contains hotel review data. The two other datasets, Amazon Software and Amazon Movies & TV were from the Amazon 5-core datasets[2] [69]. All datasets use a five-point rating system (users select an integer from 1 through 5). The same preprocessing procedure on the review text as described in Chapter 4.6 was performed. After the preprocessing, the one-word reviews were discarded as uninformative.

The statistics of these two datasets after preprocessing are summarized in Table 5.1. These datasets differ in some respects. For example, Amazon Software is the smallest and densest, whereas Amazon Movies & TV is the largest and also the sparsest among the three datasets. Most of the reviews in all datasets were rated with very high scores, with Amazon Movies & TV having the highest average score. Despite being the smallest dataset, Amazon Software contains the highest number of unique words (i.e., vocabulary) in its reviews and also has the longest reviews, on average. This contrasts with the Amazon Movies & TV dataset, which is the largest dataset, but which contains the smallest vocabulary and the shortest reviews. After applying the context extraction method by CARE, the number of candidate context words and their frequencies per review were found to correspond with the size of the vocabulary and the average number of words per review for each dataset.

5.6.2 Baselines

To establish baselines for evaluation, the performance of CARE-AI were compared with seven existing state-of-the-art rating prediction models: PMF, NMF, RC-Topic, RC-Word, DeepCoNN, NARRE, and CARL. The comparative characteristics of the baselines and CARE-AI are listed in Table 5.2. The first two methods involve latent factor models that utilize ratings alone to learn user and item representations. The remaining methods incorporate review data to learn such representations, except for RC-Topic and RC-Word, which learn a representation from each review. DeepCoNN and NARRE are review-based methods that do not consider contexts in reviews. RC-Topic, RC-Word, CARL, and CARE-AI consider the influence of contexts in reviews on rating predictions. More details about each method are summarized as follows.

[1]http://www.cs.cmu.edu/ jiweil/html/hotel-review.html
[2]https://nijianmo.github.io/amazon/index.html

Table 5.1: Statistics for the three review datasets.

	Amazon Software	TripAdvisor	Amazon Movies & TV
Reviews	12,405	873,214	3,166,002
Users	1,802	574,450	293,867
Items	801	3,940	60,169
Density	0.008594	0.000386	0.000179
Rates/User	6.884	1.388	10.774
Rates/Item	15.487	221.628	52.618
Average Rating	3.871	3.937	4.202
Vocab. Size	35,103	22,353	20,949
Word/Review	104.193	82.984	47.472
Candidates	257	226	129
Cand./Review	37.369	31.931	12.159

- **PMF** [81]. Probabilistic matrix factorization (PMF) is a standard matrix factorization approach that models user and item latent factors as Gaussian distributions.

- **NMF** [57]. Nonnegative matrix factorization (NMF) is a matrix factorization technique for which each element in the latent factor is nonnegative.

- **RC-Topic** [75]. Rich-context topic (RC-Topic) learns a review representation based on a distribution of contextual topics defined by Bauman and Tuzhilin [13] and uses the factorization machine (FM) [78] for rating prediction.

- **RC-Word**. I modified RC-Topic [75] by identifying a set of context words, following Bauman and Tuzhilin [13], and used them to represent a review with a term frequency–inverse document frequency (TF-IDF) vector.

- **DeepCoNN** [103]. Deep cooperative neural network (DeepCoNN) employs two parallel CNNs to independently construct user and item representations from their reviews, which are then used by FM for rating prediction.

- **NARRE** [19]. Neural attentional regression model with review-level explanation (NARRE) is an extension of DeepCoNN that applies review-level attention to model the contribution of each review to the rating.

Table 5.2: Comparison of the characteristics of all methods.

	Rating	Review	Represent	Attention	Contexts	Interaction
PMF	✓	-	User/Item	-	-	-
NMF	✓	-	User/Item	-	-	-
RC-Topic	✓	✓	Review	-	Topic	-
RC-Word	✓	✓	Review	-	Word	-
DeepCoNN	✓	✓	User/Item	-	-	-
NARRE	✓	✓	User/Item	Review	-	-
CARL	✓	✓	User/Item	Word	Word	-
CARE-AI	✓	✓	User/Item	Region	Region	✓

- **CARL** [101]. Context-aware user-item represention learning model (CARL) applies CNN and word-level attention to represent a context as the influence of each word in reviews on the ratings.

5.6.3 Experimental Settings

For evaluations, 80% of each dataset was randomly selected as the training set, 10% as the validation set, and the remaining 10% as the test set.

Note that all comparative methods require a significant number of ratings per user or per item to learn high-quality representations. This significant number was set as 5 for the Amazon Software and TripAdvisor datasets and 20 for the Amazon Movies & TV dataset. Those reviews that did not meet this significance criterion were therefore eliminated. Furthermore, this experiment assumed that the review texts would be available for both training and testing stages because this is exploited by RC-Topic, RC-Word, and CARE-AI when extracting contexts and making rating predictions.

For PMF and NMF, the number of latent dimensions was set to 15, the learning rate to 0.005, and the regularization parameter to 0.001.

For RC-Topic and RC-Word, the reviews were separated by applying K-means clustering for the set of review features defined by Bauman and Tuzhilin [13], including the number of words, number of verbs, and number of verbs in the past tense. The number of top contextual topics was selected from {20, 30, 50, 100, 150, 200}, and the number of top contextual words was selected from {1000, 2000, 5000, 10000}. The LibFM [78] is used to implement the FM, following Peña [75]. All FM parameters were set to default values.

For DeepCoNN, NARRE, and CARL, the number of convolutional kernels was selected from {50, 100}, the window size for CNN was set to 3, the learning rate was selected from {0.0001, 0.0005, 0.001}, the regularization parameter was selected from {0.001, 0.01, 0.1}, and the dropout rate was optimized between 0.1 and 0.5. The number of latent dimensions was set to 32 and the embedding size was set to 300 for all these models.

For CARE-AI, the learning rate was selected from {0.0001, 0.0005, 0.001, 0.005, 0.01}, the regularization parameter λ_Θ was selected from {0.01, 0.1, 1}, and the dropout rate was set to 0.2. The numbers of latent dimensions k_1 and k_2 were both set to 32. The hyperbolic tangent (tanh) was selected as the activation function for (5.4.1). Because different reviews contain different numbers of contextual regions, the maximum number of contextual regions that would be extracted from each review was set to 128. The batch sizes for both the proposed model and the baseline models were optimized from among {16, 32, 64, 128, 256}.

Evaluation Metrics

The performance of CARE-AI was evaluated against the baseline systems in terms of prediction accuracy using three ranking evaluation metrics. The first evaluation metric was the normalized discounted cumulative gain (NDCG), which evaluated the ranking accuracy of the Top-K recommendation list for each user, as expressed by

$$DCG@K = \sum_{j=1}^{K} \frac{2^{rel_j} - 1}{log_2(j+1)}, \quad NDCG@K = \frac{DCG@K}{IDCG@K}. \tag{5.6.1}$$

Here, $rel_j \in \{1,2,3,4,5\}$ is set as the actual rating scores for an item at rank position j. Because most users in Amazon Software and TripAdvisor had rated items less than seven times, the NDCG@3, NDCG@5, and NDCG@7 were chosen for their evaluation, whereas for Amazon Movies & TV, with more than 10 ratings per user, the NCDG@5, NCDG@10, and NCDG@15 were therefore chosen for that dataset.

In addition to NDCG, the performances were also evaluated by the hit ratio (HR) and mean reciprocal rank (MRR). The reviews in test data that had ratings above "3" were first classified as positive reviews, and the remainder as negative reviews. An HR@K is calculated as the number of positive reviews appearing in the Top-K recommendation list for each user, whereas MRR is computed as the rank of the first positive review in each user recommendation list, as given by (5.6.2) and (5.6.3), respectively.

Table 5.3: The NDCG values for the compared methods on the Amazon Software dataset.

	NDCG@3	NCCG@5	NDCG@7
PMF	0.89160	0.90009	0.87302
NMF	0.88848	0.84849	0.81870
RC-Topic	0.89944	0.95023	0.93930
RC-Word	0.93513	0.93698	0.89889
DeepCoNN	0.92396	0.92357	0.86373
NARRE	**0.93901**	0.93200	0.86880
CARL	0.92811	0.91501	0.91289
CARE-AI	0.92224	**0.98448**	**0.97039**

$$HR@K = \frac{1}{K} \sum_{j=1}^{K} rel_j \tag{5.6.2}$$

$$MRR = \frac{1}{|User|} \sum_{u_i \in User} \frac{1}{rank_{u_i}} \tag{5.6.3}$$

Here, rel_j was set to 1 for a positive review and 0 otherwise. The value for $rank_{u_i}$ is the first rank position of a positive review in u_i's recommendation list. The HR@5 was chosen in evaluating the performance for all datasets.

5.6.4 Experimental Results

The values for NDCG of Amazon Software, TripAdvisor, and Amazon Movies & TV are respectively presented in Table 5.3, 5.4, and 5.5, whereas the HR, and MRR for all baseline systems and CARE-AI with each of the three datasets are presented Table 5.6. From all tables, note that CARE-AI achieves the highest accuracy for almost every MRR and rank of NDCG and HR across all datasets. Next are the deep-learning-based methods that utilize review data (DeepCoNN, NARRE, and CARL). Although CARL, which is a context-aware method, performs quite well on Amazon Software, DeepCoNN and NARRE perform better on TripAdvisor and Amazon Movies & TV datasets. DeepCoNN and NARRE obtain very similar results across all datasets, although DeepCoNN seems to perform slightly better on TripAdvisor and Amazon Movies & TV. Furthermore, the other two context-aware baseline

Table 5.4: The NDCG values for the compared methods on the TripAdvisor dataset.

	@3	@5	@7
PMF	0.87673	0.78935	0.77752
NMF	0.88341	0.82420	0.80706
RC-Topic	0.91597	0.86226	0.84023
RC-Word	0.89744	0.81977	0.80363
DeepCoNN	0.92963	0.88715	0.87026
NARRE	0.92841	0.89495	0.87020
CARL	0.92335	0.86649	0.85077
CARE-AI	**0.93147**	**0.91826**	**0.93168**

Table 5.5: The NDCG values for the compared methods on the Amazon Movies & TV dataset.

	@5	@10	@15
PMF	0.90051	0.87458	0.86025
NMF	0.89939	0.87110	0.85774
RC-Topic	0.91223	0.89489	0.88759
RC-Word	0.90175	0.87899	0.86520
DeepCoNN	**0.94070**	0.92895	0.92294
NARRE	0.94018	0.92757	0.91909
CARL	0.92798	0.91375	0.90308
CARE-AI	0.91707	**0.93527**	**0.92328**

systems (RC-Topic and RC-Word) achieved quite good NDCG values on Amazon Software but were less effective on the other two datasets, as was CARL. Finally, PMF and NMF, which do not consider review information, yielded the lowest accuracies on all evaluation metrics when compared with the other systems.

5.7 Discussion

In this section, a detailed analysis of the performance of the proposed method and the baseline systems in various aspects is presented. This includes, the predictive performances, the effectiveness of the interaction and attention modules, an analysis of model parameters, the performance under the sparsity situation, and finally, the impact of review quality.

Table 5.6: The HR and MRR values for the compared methods on the three review datasets.

	A. Software		TripAdvisor		A. Movies & TV	
	HR@5	MRR	HR@5	MRR	HR@5	MRR
PMF	0.84000	0.85534	0.62667	0.84485	0.77305	0.53863
NMF	0.84000	0.85067	0.62667	0.84438	0.77162	0.53847
RC-Topic	0.88000	0.85555	0.61333	0.84282	0.78697	0.54063
RC-Word	0.84000	0.85802	0.60667	0.84075	0.77688	0.54233
DeepCoNN	0.92000	0.86054	0.63333	0.85249	0.81763	**0.55235**
NARRE	0.92000	0.86204	**0.64000**	0.85212	0.81608	0.55206
CARL	0.92000	0.86067	0.63333	0.85226	0.80390	0.54810
CARE-AI	**0.96000**	**0.86466**	**0.64000**	**0.85456**	**0.82005**	0.54922

5.7.1 Predictive Performance

Here, three main aspects of predictive performances are discussed. First, the effectiveness of leveraging review content in making predictions is analyzed. This follows by an analysis on how identifying and incorporating contexts from reviews affect the prediction accuracy. Finally, the discussion on how the dynamic modeling of user and item representations differs from and improves on the use of the static-representation approach, is conducted.

Utilizing Review Data

First, the merit of utilizing review data for making recommendations is discussed. As shown by every table, all methods that leverage review content to learn user and item representations (DeepCoNN, NARRE, CARL, and CARE-AI) returned better values for NDCG, HR, and MRR than the standard CF-based methods that ignore reviews (PMF and NMF). This demonstrated that the rich and useful information embedded in reviews helps the learning of more appropriate representations, which more accurately capture the personal preferences of a user or the unique features of an item. Utilizing these representations consequently resulted in more accurate rating predictions.

Although leveraging review data, the RC-Topic and RC-Word methods did not always provide better prediction accuracy than the standard CF-based methods that did not consider reviews. For example, they gave lower HR and MRR values on the TripAdvisor dataset than PMF and NMF. This might be because both RC-Topic and RC-Word utilize review content to learn a representation of the review itself and use it directly for rating prediction, rather than learning representations for users and items based on a latent factor model, as do the other methods. I believe that their review representations do not capture the personalized information relating to user preferences or item features, which makes their predictions less accurate than latent factor-based methods.

Incorporating Contextual Information

The effect of incorporating contextual information hidden in reviews on the rating prediction is now analyzed. By comparing the results of the review-based context-aware methods (RC-Topic, RC-Word, CARL, and CARE-AI), only CARE-AI could surpass the accuracy of the review-based methods that did not consider contexts (DeepCoNN and NARRE).

This begins with an analysis of RC-Topic and RC-Word, which are review-based context-aware methods based on topic modeling and TF-IDF representations. Although they mostly performed better than the standard-CF based methods (PMF and NMF), they were the least accurate of the review-based methods. This might be because the performance of their context extraction depends on the quality of the reviews. Based on the definition used in RC-Topic and RC-Word, contexts can be inferred only from reviews of high quality, which often contain a significant number of words. For review data containing many low-quality reviews, the context extraction would be less effective. In contrast, CARE-AI is capable of extracting contexts from any kind of review, provided there is at least one candidate context word embedded in that review. This makes CARE-AI more robust toward review quality than RC-Topic and RC-Word. The impact of review quality on CARE-AI is further analyzed in Section 5.7.4.

Now, the results from CARL, which is a context-aware method based on a deep-learning technique, is analyzed. First, with the advantage of utilizing both deep learning and an attention mechanism, CARL outperformed both RC-Topic and RC-Word on almost every dataset. However, although it outperformed the noncontext-deep-learning techniques (DeepCoNN and NARRE) on Amazon Software, it obtained less accurate results on the other two datasets. According to my assumption, I believe this occurred for two possible reasons. First, CARL considers the contribution of every word in reviews to the rating as the influence of contexts. Some of these words, however, are irrelevant to the user preferences or item features and could degrade the quality of the representations. The second reason is that CARL computes its attention score based on words from all previous reviews, rather than focusing on those contained in a recent review. In context-aware recommendations, contexts are relevant at the time the rating is created and, therefore, only applicable to a particular review and not to others. By considering words from all previous reviews, CARL incorporates irrelevant contexts that are not associated with the current rating situation, thereby constructing less effective representations for users and items. On the other hand, CARE-AI constructs the representations based only on those words (and their neighbors) in a particular review that influence the rating distribution as a context and achieved better results on all datasets. This supports my assumption that considering only the words in a single review that are relevant to the user preferences and item features is better for capturing the contextual information and results in constructing more effective and meaningful representations.

Static and Dynamic Representations

Finally, the effectiveness of constructing user and item representations dynamically rather than relying on a static set of representations, is analyzed.

First, the predictive performances of the methods that learn static user and item representations (DeepCoNN, NARRE, and CARL) are discussed. Although the attention-based baseline systems (NARRE and CARL) outperform DeepCoNN on the Amazon Software dataset, they deliver lower overall NDCG, HR, and MRR values on the other two datasets. This implies that applying an attention mechanism does not always improve the prediction accuracy if it is not properly integrated into the model. One concern about the attention mechanisms of both NARRE and CARL is that their attention scores are computed from the contents of previous user or item reviews, rather than utilizing the content of the target review (the review for which to predict a rating). The representations of NARRE, CARL, and DeepCoNN consider a user's past preferences or item's past features but barely capture more relevant information such as contexts, which can be extracted only from the target review.

In contrast, the dynamic approach for constructing user and item representations achieves a higher overall prediction accuracy than the static-representation approach. This focuses on utilizing the text of the target review as the main source from which to extract the relevant contexts for a user–item pair. CARE-AI first apply interaction module to model the relevance of each extracted context in the review to the individual user preferences and item features. CARE-AI then compute the attention score for each context, based on its relevance level when compared with the other contexts embedded the same review. This helps to construct fine-grained user and item representations that dynamically capture the relevance of contexts in a particular review to the user preferences and item features.

The next section studies further the impact of the attention and the interaction modules in CARE-AI. This follows by the analysis on the performance of CARE-AI for various parameter settings. Moreover, the impact of the review quality on the performance of context-aware methods is discussed. Finally, the robustness of CARE-AI in situations of rating sparsity is addressed. All evaluations in this section were conducted on the TripAdvisor dataset.

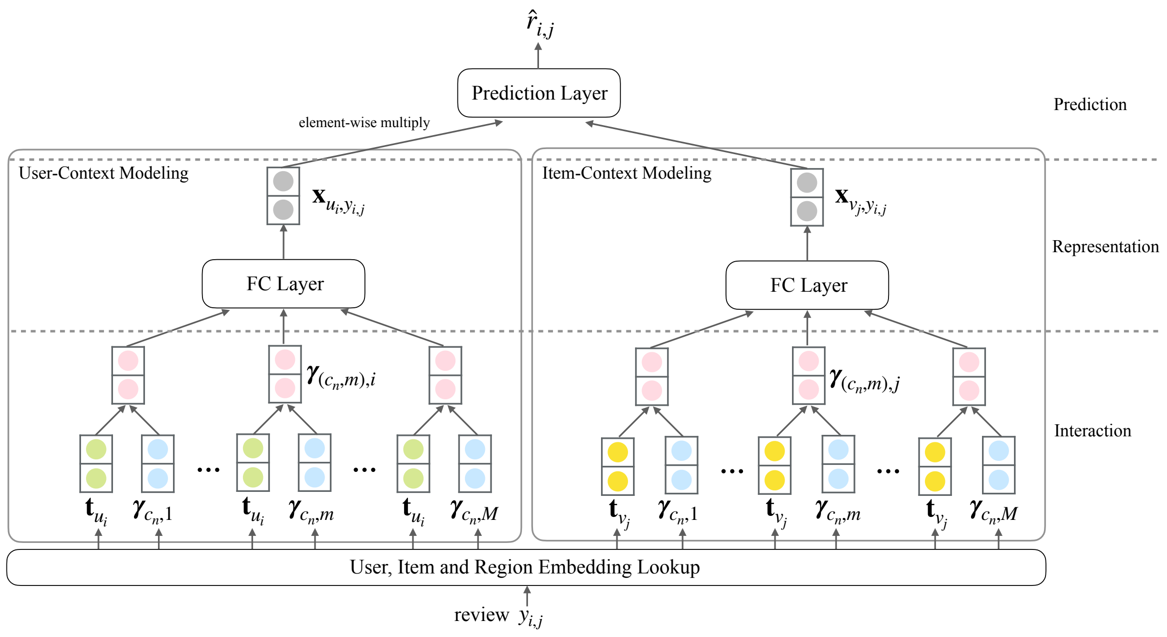

$\hat{r}_{i,j}$
Prediction Layer
element-wise multiply
Prediction
User-Context Modeling
$\mathbf{x}_{u_i,y_{i,j}}$
Item-Context Modeling
$\mathbf{x}_{v_j,y_{i,j}}$
Representation
FC Layer
FC Layer
$\gamma_{(c_n,m),i}$
$\gamma_{(c_n,m),j}$
Interaction
$\mathbf{t}_{u_i}$ $\gamma_{c_n,1}$ … $\mathbf{t}_{u_i}$ $\gamma_{c_n,m}$ … $\mathbf{t}_{u_i}$ $\gamma_{c_n,M}$
$\mathbf{t}_{v_j}$ $\gamma_{c_n,1}$ … $\mathbf{t}_{v_j}$ $\gamma_{c_n,m}$ … $\mathbf{t}_{v_j}$ $\gamma_{c_n,M}$
User, Item and Region Embedding Lookup
review $y_{i,j}$

Figure 5.3: Illustration of CARE-I model.

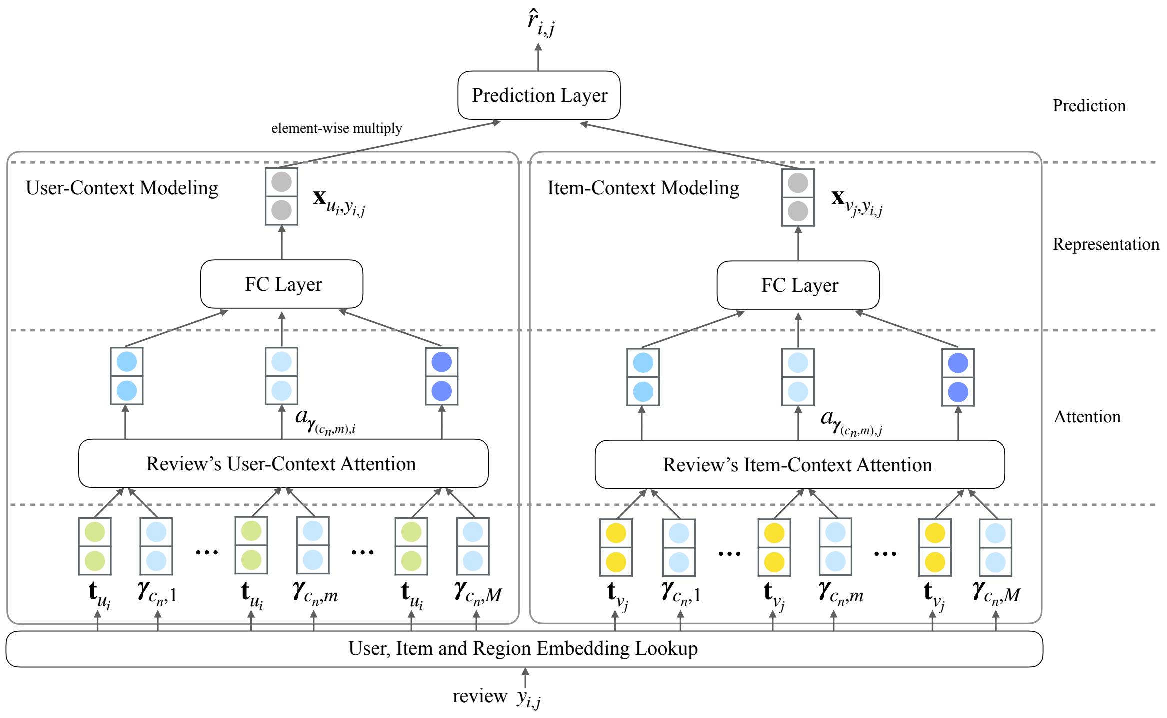

$\hat{r}_{i,j}$
Prediction Layer
element-wise multiply
Prediction
User-Context Modeling
$\mathbf{x}_{u_i,y_{i,j}}$
Item-Context Modeling
$\mathbf{x}_{v_j,y_{i,j}}$
Representation
FC Layer
FC Layer
$a_{\gamma_{(c_n,m),i}}$
$a_{\gamma_{(c_n,m),j}}$
Attention
Review's User-Context Attention
Review's Item-Context Attention
$\mathbf{t}_{u_i}$ $\gamma_{c_n,1}$ … $\mathbf{t}_{u_i}$ $\gamma_{c_n,m}$ … $\mathbf{t}_{u_i}$ $\gamma_{c_n,M}$
$\mathbf{t}_{v_j}$ $\gamma_{c_n,1}$ … $\mathbf{t}_{v_j}$ $\gamma_{c_n,m}$ … $\mathbf{t}_{v_j}$ $\gamma_{c_n,M}$
User, Item and Region Embedding Lookup
review $y_{i,j}$

Figure 5.4: Illustration of CARE-A model.

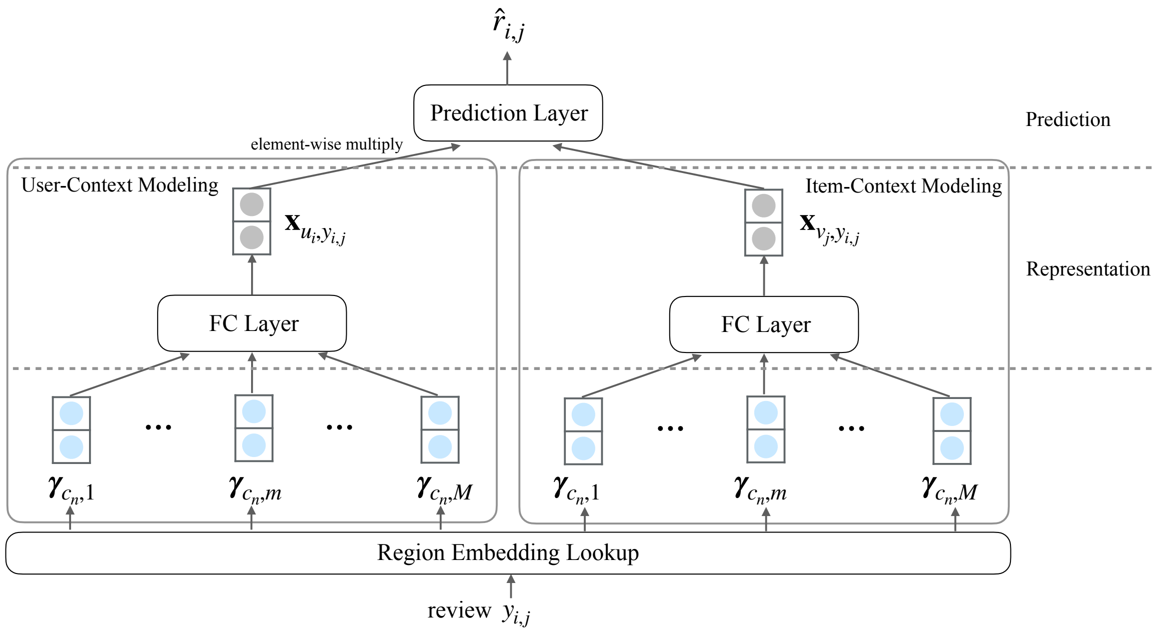


Figure 5.5: Illustration of CARE-0 model.

5.7.2 Attention and Interaction Modules

To evaluate the effect of incorporating attention and interaction modules, four variants of rating prediction for CARE based on the four possible combinations of these two modules, as listed below.

- CARE-AI: a model that incorporates both attention and interaction modules, that is, the main model, as presented previously in Figure 5.2.

- CARE-I: a model that considers only the interaction module, with the attention module being ignored, as presented in Figure 5.3. The user and item representations are constructed only from their previous interactions with contextual regions in a review. Specifically, instead of computing an attention score and using it as a weight for each region embedding in (5.4.3), all projected region embeddings obtained from (5.3.1) were directly averaged to form the user and item representations.

- CARE-A: a model that considers only the attention module, with the interaction module being ignored, as presented in Figure 5.4. This model aims to find the contribution of each contextual region to a review's rating, when compared with the other regions in the same review. Technically, the region embeddings generated from context extraction directly as are used an input for (5.4.1), without applying the projection operation of (5.3.1).

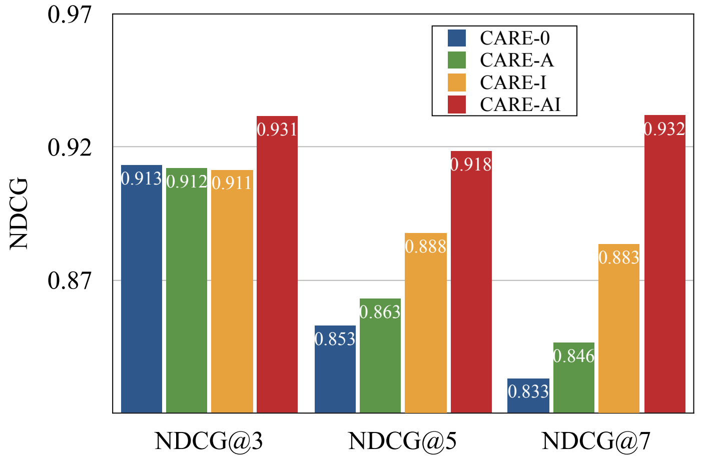


Figure 5.6: The NDCG values for variants of the prediction models of CARE

- CARE-0: a model without attention or interaction modules, as presented in Figure 5.5. The user and item representations are obtained by directly averaging all input region embeddings for a review. In the fully connected layer, one shared weight matrix are used for all users, and another for all items.

The hyperparameters of each variant are tuned with the same settings as the main model, as described in Section 5.6.3. Their predictive performances are presented in Figure 5.6.

From Figure 5.6, the CARE-0 model (no attention or interaction module) gave the lowest overall NDCG values of all the variants. The implication is that applying the attention and/or interaction modules improves the performances of the proposed model.

By modeling the varying influences of contexts in a review to the rating by applying the standard attention mechanism, the CARE-A model made more accurate predictions than the base model. This demonstrates that different contexts can have different impacts on the rating behaviors of all users and items. However, because CARE-A does not consider the relevance of each context to an individual user preferences or item features, two reviews containing the same set of contexts would produce the same user and item representations. That is, the representations generated by this model depend only on the different contributions of contexts embedded in the review, regardless of any personalized interaction with the user or item. This makes CARE-A suitable for a sparse dataset where most users participate only rarely or most items are rated only infrequently. However, if a review contains too few or too many candidate context words, CARE-A might not be able to exploit the information effectively, thereby degrading the recommendation quality. There is more analysis of the impact of the number of candidates in Section 5.7.3.

Although applying an attention mechanism gives improved accuracy compared with the base method, CARE-A still did not match the performance of CARE-I, which models the relevance of contexts to an individual user preferences and item features. The representations for this model are unique for each user or item even for reviews containing the same set of contexts, which is more appropriate when making a personalized recommendation. However, because CARE-I relies on previous interactions with contexts, it requires a significant number of reviews to precisely capture the relevance of each context. Moreover, unlike CARE-A, this model does not consider the varying influence of different contexts on a review's rating, which should be a factor in improving the recommendation quality.

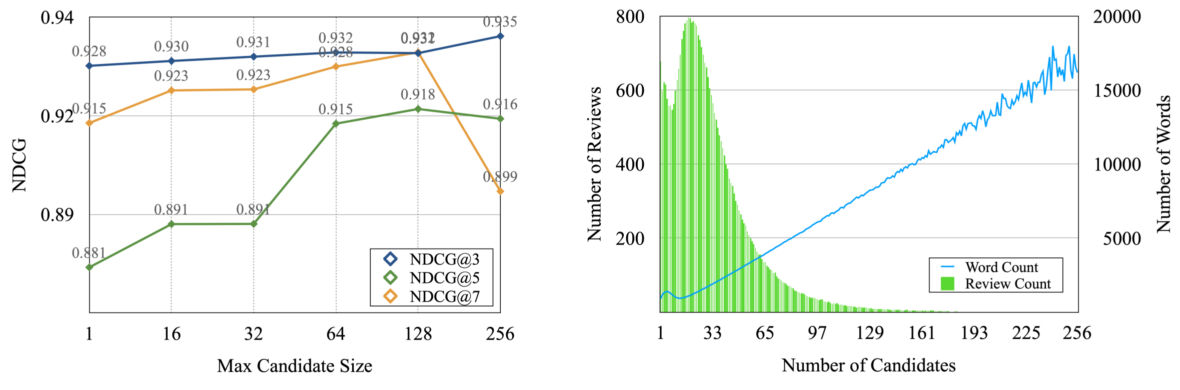
So far, both CARE-A and CARE-I have their own advantages, which contribute to improved predictive performances. By recognizing the trade-off between these two methods, it can be found that a combined model, CARE-AI, achieved the best predictive performance among all the variants. This is convincing evidence that the influence of each context can be adequately modeled based not only on its relevance to the target user preferences and item features but also to its contribution to the rating of a particular review when compared with other contexts in the same review. This model, however, also inherits the characteristics of both CARE-A and CARE-I, in that it requires an appropriate number of candidate context words per review and enough reviews per user and per item to be able to learn high-quality user and item representations.

5.7.3 Parameter Sensitivity

This subsection first studies the impact of model parameters on the predictive performance of CARE-AI. These parameters include the number of candidate context words, the region size, and the embedding size. In addition, an investigation on the computational cost required for these model parameters is conducted.

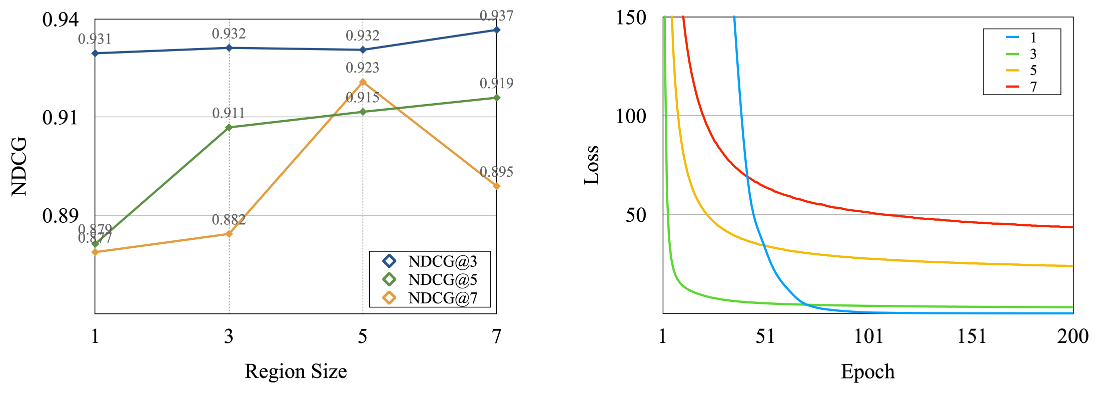
Impact of Candidate Context Words

Figure 5.7 shows the impact of the number of candidate context words per review on the prediction accuracy. Because different reviews contain different numbers of candidate context words, how these numbers affect the performance of the method should be addressed. To investigate this, a fixed maximum number of candidate context words to be extracted from each review was fixed. As shown in Figure 5.7 (a), increasing the number of candidates across {1, 16, 32, 64, 128} yields a higher accuracy. However, for a maximum number of 256, the NDCG@5 and NDCG@7 values start to decrease. The main reason might be that the reviews containing very many candidates are unusually long reviews. As shown in



(a) Predictive Performances (b) Statistics

Figure 5.7: Impact of different numbers of candidate context words.



(a) Predictive Performances (b) Validation Loss

Figure 5.8: Impact of different region sizes.

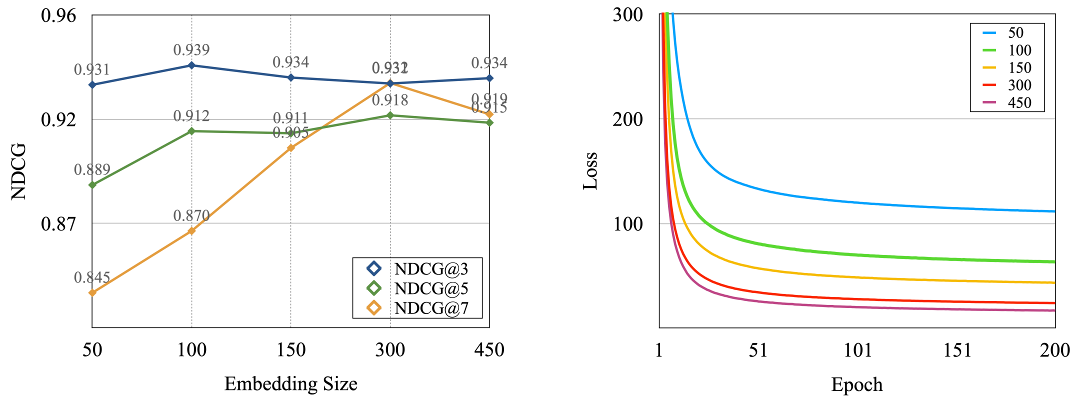
Figure 5.7 (b), most reviews in the TripAdvisor dataset contain only about 30 candidates with less than an average of 2500 words per review. Some of the very long reviews (having more than 100 candidates) could potentially be spam or otherwise less useful reviews, which could degrade the prediction accuracy. In addition, considering too many candidate context words might weaken the effectiveness of the attention module, because uniformly distributed attention scores would become more likely. Because using 128 candidates yields the highest predictive accuracy, this value was used in the training process.

Impact of Region Size

The impact of differently sized text regions are now investigated. Increasing the region size means having more neighboring words to be identified and incorporated as contexts together with the candidate context words. Figure 5.8 shows the predictive performance and the validation loss in context extraction for each region size in $\{1, 3, 5, 7\}$. (Region size = 1 means that a context is constructed only from the candidate context word itself, without any consideration of its neighboring words.) Despite having a very high validation loss at the beginning, using this region size completed its training with the lowest loss when compared with other region sizes. This is because it only learns one word embedding to represent one rating distribution for each candidate, which has minimal complexity. However, its prediction accuracy is also the lowest for the various region sizes. This implies that considering only a single word as context is insufficient to accurately capture the actual influence of relevant contexts on the distributions of ratings. Because increasing the region size increases the possible number of combinations between neighboring words and the candidates to be constructed as contexts, it significantly increases the number of corresponding rating distributions, raises the model complexity, and increases the validation loss. The prediction accuracy, however, improves significantly as the region size increases from 1 to 5. This supports my assumption that incorporating neighboring words benefits the construction of more relevant contexts for capturing rating distributions, resulting in more accurate recommendations. However, an excessively large region size may also degrade the performance, as shown by the NDCG@7 value for region size = 7. This is because the model might mistakenly incorporate words from different phrases or sentences to construct incorrect contexts. For example, suppose that the task is to extract a context from the text region “is really worst services, the only good”. If the region size is set to 3 or 5, the context could be extracted as “worst services”, but if region size is set to 7, the context might be constructed as “services good.” Exploiting such unintended contexts could consequently affect the rating prediction score. The region of size 5 was then selected for the parameter setting because of its optimal performance.

Impact of Embedding Size

The impact of different embedding sizes are analyzed, which involves the dimensions of the word embeddings, the local context units, and the corresponding region embeddings. Figure 5.9 gives the predictive performance and the validation loss of the model when trained with different embedding sizes chosen from $\{50, 100, 150, 300, 450\}$. The loss values in Figure 5.9 (b) show that the larger embedding sizes are more effective for representing the rating distributions, although the improvement is small between 300 and 450. Accordingly, the larger embedding sizes also produce more accurate recommendations. A size of 300 was chosen for the model setting because it gives a near-optimal overall prediction accuracy and requires less computation time than using a size of 450.



(a) Predictive Performances (b) Validation Loss

Figure 5.9: Impact of different embedding sizes.

Computational Cost

The proposed method was implemented with TensorFlow on Intel Xeon X5680 Processor and NVIDIA Tesla P100 GPUs. The most time-consuming part was the context extraction by CARE, which took roughly about 5-10 hours for one training, depending on the model parameters. Two parameters from CARE that had significant impact on the training time were the region size and embedding size, as shown in the Figure 5.10 (a) and (b), respectively. Notice that the region size of 1, i.e. considering only candidate context words as contexts, consumed significantly less amount of time for training, compared to the other region sizes. This corresponds to the fact that the number of candidate context words is significantly smaller than the number of combinations of words in the other region sizes. The larger the regions, the more combination of words are needed to be considered for modeling the rating distribution of the contextual regions, and thus significantly increases the computational time. Similarly, increasing the embedding sizes consequently increased in time for generating region embeddings for the contextual regions.

The learning process of a predictive model by CARE-AI, on the other hand, took less amount of training time, as compared to the context extraction by CARE. Figure 5.10 (c) and (d) respectively show the training times for CARE-AI with different numbers of candidate context words per review, and different sizes of latent dimension. As it can be seen from both figures, increasing the candidate size and the latent size linearly increased the training time. As compared to the region size and embedding size from CARE, the candidate size and latent size seem to have less impact on the computational time. By combining two offline training steps (context extraction and rating prediction models), the proposed method took about 10-15 hours for learning and optimizing the model parameters.

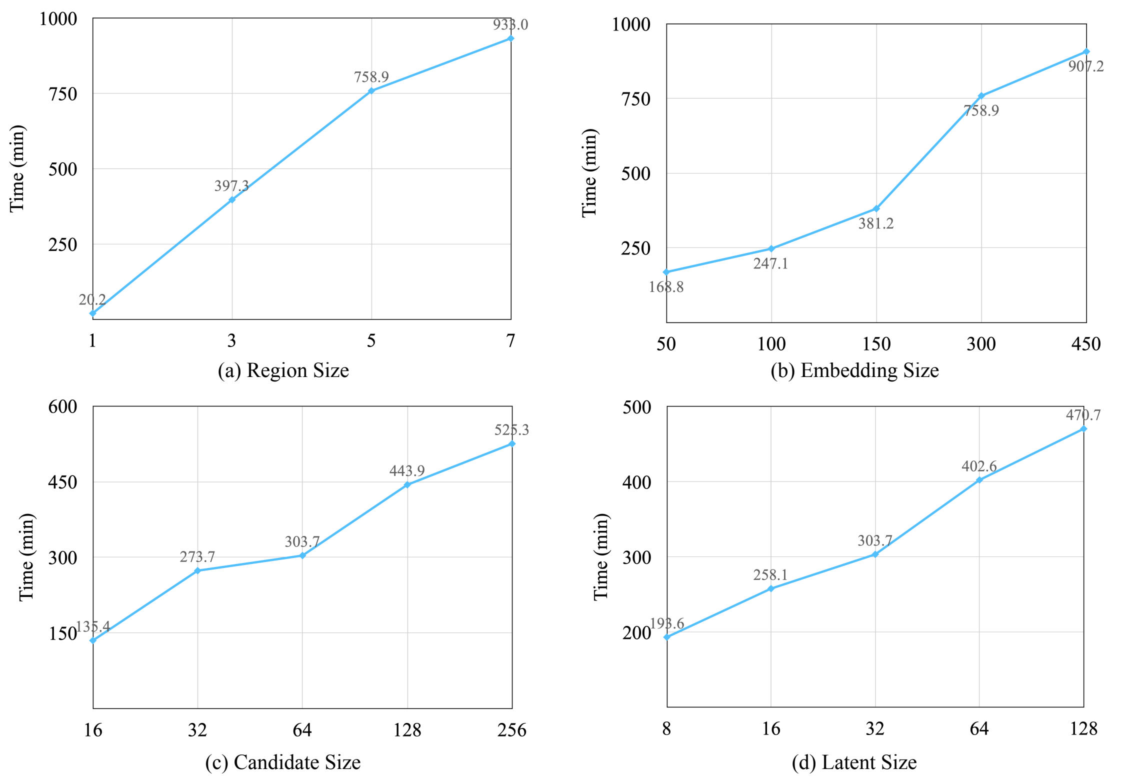


Figure 5.10: Training times with respect to four model parameters.

After the training is done, the online rating prediction for a given query can now be made. Given a review text of a user-item pair, CARE firsts extract all contextual regions from that review, based on the list of candidate context words learned from the offline training. These contextual regions are then mapped with the learned word embeddings and local context units to compute their associated region embeddings. Such region embeddings are then passed through CARE-AI, which looks up for the learned user-context and item-context interaction vectors from the user and item IDs. All components are then fed into the interaction, attention, and prediction modules to compute the rating score. The whole process of the online rating prediction for a given user-item pair took only about a few seconds, which is feasible for making recommendations in real-time.

One main goal for the future work is to optimize the cost of training to make the model more efficient, which might be achieved by combining the context extraction and rating prediction into a single model with less complexity, and having a smaller number of model parameters.

Table 5.7: Comparison of context-rich and context-free reviews.

Dataset	Statistic	Review Type	
		Context-Rich	Context-Free
Amazon Software	Word/Review	173.371	21.814
	Cand./Review	60.780	9.538
TripAdvisor	Word/Review	148.439	51.642
	Cand./Review	57.948	21.643
Amazon Movies & TV	Word/Review	150.165	12.676
	Cand./Review	33.211	4.926

5.7.4 Impact of Review Quality

This subsection analyzes how the quality of reviews affects context extraction and rating prediction. First, a method for dividing reviews with respect to their quality is required. I follow Bauman and Tuzhilin [13] in classifying reviews into context-rich and context-free reviews, based on the richness of contexts. The criteria used for classification includes the review features such a number of words, number of verbs and number of verbs in past tense. After reviews are classified, CARE is applied to extract the candidate context words and their associated contextual regions from each type of reviews. Table 5.7 shows a comparison of the statistics for each type of review on all three datasets. These statistics include the average number of words and the average number of extracted contexts (candidate context words) per review. As shown in Table 5.7, the number of words in context-rich reviews is significantly higher than in context-free reviews. Applying CARE consequently resulted in more candidate context words being extracted from each context-rich review for all datasets. This indicates that the quality of reviews significantly impacts the amount of contexts being extracted by CARE.

In addition, I further analyze how this difference in number of contexts would affect the prediction capability of CARE-AI. To do so, the predictive performance of CARE-AI were further evaluated on each type of reviews. The reviews in test data of all three datasets were divided into context-rich and context-free reviews, and evaluated the performance for each set of reviews separately.

Figure 5.11 shows the comparison of NDCG@5 of CARE-AI on each type of reviews from three datasets. From the figure, the prediction accuracies of CARE-AI on context-rich reviews were significantly higher than those in context-free reviews. The differences in accuracies were even more significant on Amazon Software dataset, in which the number of

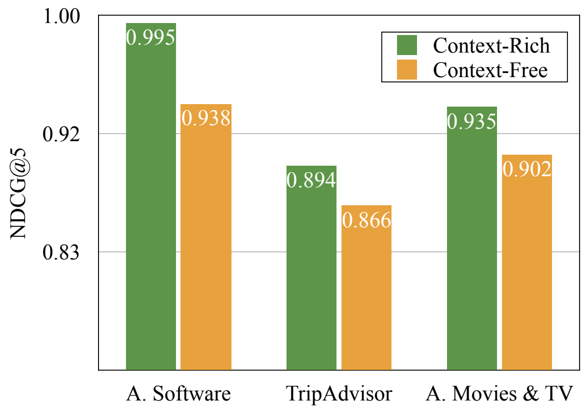


Figure 5.11: Predictive performances of CARE-AI on three datasets based on review quality.

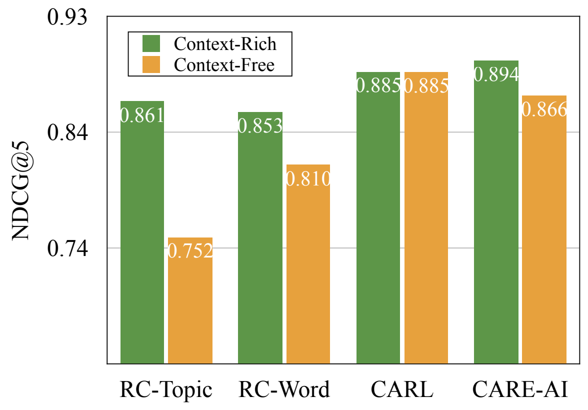


Figure 5.12: Predictive performances for context-aware methods based on review quality.

extracted contexts from two types of reviews were significantly different. From the result, it can be inferred that the review quality also have significant impact on the predictive performance. With high quality reviews, more number of contexts can be extracted, and have more information to model the rating—resulted in more accurate prediction.

Finally, the predictive performance of CARE-AI for each type of review is compared with the other context-aware baseline systems. Figure 5.12 shows the NDCG@5 value for CARE-AI and the other context-aware methods (RC-Topic, RC-Word, and CARL) for context-rich and context-free reviews. First, the RC-Topic and RC-Word methods gave the lowest prediction accuracies for both types of review. Furthermore, their accuracies for context-free reviews were significantly lower than those for context-rich reviews. This results from these two methods treating contexts as topics or words that occur more frequently in context-rich reviews. They are therefore unable to extract much contextual information from the context-free reviews, resulting in lower prediction accuracies.

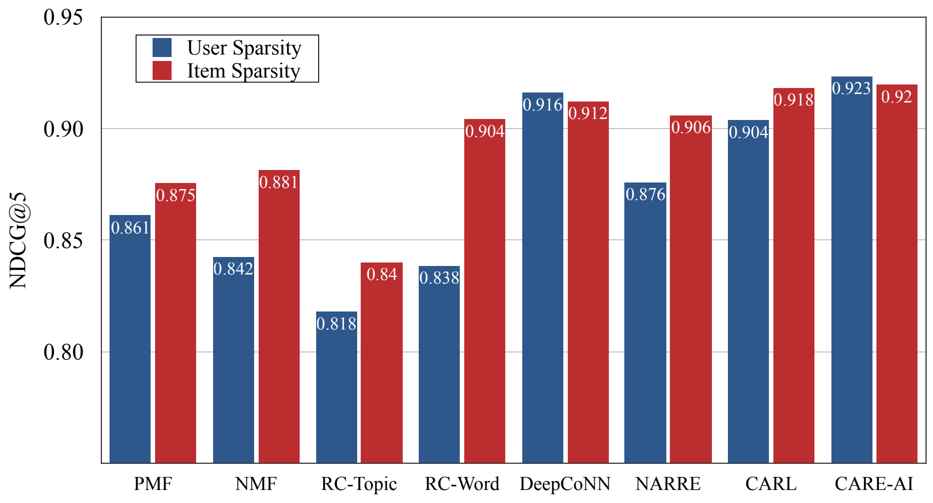


Figure 5.13: Predictive performances with sparse data.

In contrast, for both CARL and CARE-AI, the prediction accuracy for context-rich reviews differs little from that for context-free reviews. That is, both CARL and CARE-AI are robust with respect to the quality of reviews. Note that CARE-AI is slightly less accurate for context-free reviews than the CARL method. This might be because the average number of contexts per review in context-free reviews is smaller than the number of words. Because CARL considers the influence of every word in the review as context, it has more information from which to learn user and item representations. However, if there are sufficient contexts per review, which is more likely for context-rich reviews, CARE-AI is able to achieve a higher accuracy than CARL.

5.7.5 Performance on Sparse Data

Finally, the performance of CARE-AI under conditions of rating sparsity is analyzed. The training data was modified from the TripAdvisor dataset for two types of rating sparsity, namely user-rating sparsity (each user provided only one rating) and item-rating sparsity (each item is rated only once). The sparsity ratios for the user-rating sparsity data and the item-rating sparsity data were 0.99937 and 0.99974, respectively. The NDCG@5 values achieved by CARE-AI and all the baseline systems when using the sparsity-modified data are given in Figure 5.13.

From Figure 5.13, all methods that utilize review data for constructing user and item representations (DeepCoNN, NARRE, CARL, and CARE-AI) produced more accurate predictions than those constructing such representations using only rating data (PMF and NMF). This implies that, in rating-sparsity situations, review content can be used to construct more effective user and item representations than utilizing only the ratings, leading to more accurate rating prediction.

However, utilizing review data does not always solve the sparsity problem. Figure 5.13 shows that both RC-Topic (in particular) and RC-Word produced significantly lower prediction accuracies than the other review-based methods and even lower prediction accuracies than the rating-based methods (PMF and NMF). This may be because both RC-Topic and RC-Word depend heavily on word frequency when building their review representations. Although the review data might be sparse, its vocabulary size could still be very large and have very low word frequencies. These two methods, therefore, do not have sufficient data to effectively model their topic or word distributions from which to build high-quality review representations. Utilizing these poor representations will then result in low-accuracy rating predictions.

Finally, the discussion on the predictive performance of CARE-AI, compared with the other deep-learning-based representations (DeepCoNN, NARRE and CARL) is conducted. Figure 5.13 shows that CARE-AI is able to achieve the highest NDCG@5 values for both the user-rating and item-rating sparsity datasets. Note that in DeepCoNN, NARRE, and CARL, these representations are generated statically from corresponding previous reviews. Such representations, however, tend to overfit with sparse data, for which each user or item provides only one or very few reviews. Utilizing these representations to predict ratings for reviews of unseen items will then be less effective. In contrast, CARE-AI utilizes previous reviews only to model the interactions of users and items with contexts, whereas the representations themselves are generated dynamically based on any context being extracted from each particular review. This makes CARE-AI's representations less affected by the sparsity in previous reviews, thereby achieving more accurate predictions.

5.8 Conclusion

In this chapter, the CARE-AI, an extension model utilizing the extracted contexts from CARE for rating prediction is proposed. In making rating predictions, the user and item representations are generated dynamically for each specific review through the proposed interaction and attention modules, based on the relevance of the contexts extracted from that review to a user preferences and an item features. Utilizing such representations for making rating predictions is more accurate than using state-of-the-art deep-learning-based representation techniques that do not properly consider the relevance of a context. Furthermore, as strongly supported by the experiment results, CARE-AI is also more robust for making recommendations in the situations of rating sparsity.

Chapter 6

Multi-Criteria Rating Conversion without Relation Loss for Recommender Systems

6.1 Introduction

The main mechanism of the collaborative filtering-based (CF-based) approach is based on the usage of opinions from users to form the high quality set of neighbors, which consequently affects the quality of recommendations. In multi-criteria (MC) recommendation approach, users are able to specify their preferences on each item in multiple aspect rather than only one single rating [6]. For example, a user might rate a hotel based on its price, location, cleanliness, or service. Such multi-criteria scheme can improve the process of analyzing the rating data in deeper aspect—forming a better set of neighbors.

However, using ratings from neighbors to make a prediction for an active user directly may result in failure of recommendation. This is due to two following problems. First, as mentioned in Chapter 2.4.3 the habits or patterns on giving ratings among users vary due to their personal biases. Therefore, using ratings from neighbors to predict the active user rating directly may not be practical. Another problem is that, most of RS systems contain the enormous amount of items, but the neighbors do not always provide their ratings toward the item targeted by the active user—leading to invalid prediction.

One possible solution to solve these two problems is by the following steps. First, find *all* users (not only the neighbors) who have rated the target item as the *raters*. Then, convert the ratings of the raters into the ratings in the aspect of the active user by considering and analyzing their rating patterns. If the input ratings are properly analyzed and represented according to the true preference of a user, the recommendation output will be more accurate and more preferable. Some methods can be applied to solve this problem, including linear mapping [8], Lathia's transposed function [56] and Warat's transposed function [18]. However, these methods are designed for converting only a single criterion rating. When it comes to multi-criteria framework, they have to convert each criterion rating separately—causing a scalability problem and consuming a lot of resources. Also, converting each criterion rating independently might cause a loss in implicit relation among the criteria ratings.

In this chapter, a novel method for simultaneously converting the multi-criteria ratings from one user to another user's aspect is proposed. This method can convert the multi-criteria ratings simultaneously into the same space in order to maintain the implicit relationship among the criteria ratings. By applying the principle component analysis (PCA) [87] on high dimensional dataset, the patterns containing the most crucial characteristics of the data are extracted. However, different users have different patterns on different planes with different variances. This means that their planes are not equal and their patterns cannot be converted to each other directly. To solve this problem, the variance normalization is applied on both plane to make a pair of different planes to be viewed as the same plane, before extracting the user preference pattern using PCA. The experiments demonstrated that the proposed method outperformed both single and multi-criteria rating conversion techniques in terms of prediction accuracy and prediction coverage.

The main contributions of this work can be summarized as follows.

- The proposed method maintains the implicit relation among the multi-criteria ratings by simultaneously converting all criteria ratings from one user to another user's aspect.
- The experiment showed that the proposed method yielded better on both accuracy and prediction coverage, compared to well-known rating conversion techniques.

6.2 Model Overview

In this section, an overview of the proposed multi-criteria rating conversion technique is presented. Figure 6.1 illustrates the four main steps of the proposed method. First, the ratings of two users are normalized into the same plane of variance by using the variance normalization technique. Then, PCA is applied to extract the multi-criteria rating patterns from the users. These patterns are then utilized in the multi-criteria rating conversion process. Finally, the converted multi-criteria ratings are used in the rating prediction for making the recommendation.

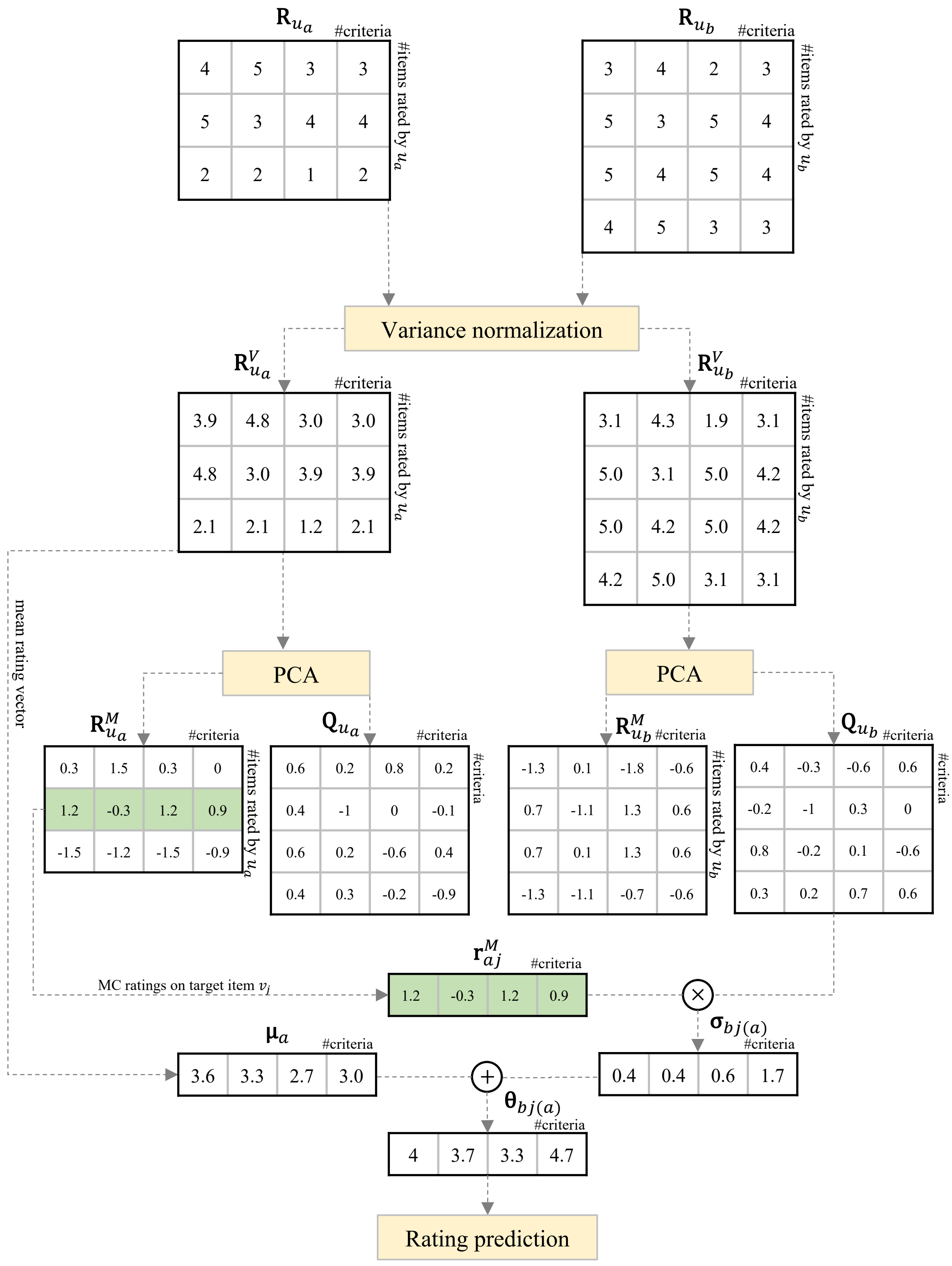


Figure 6.1: An overview of the mechanism of the proposed method.

From now, a term *rater* will be used to denote a user whose ratings will be converted and a term *active user* denotes a user who retrieve the converted ratings from the raters.

6.3 Variance Normalization

Before a pattern from the rater can be used to convert his multi-criteria ratings into an active user's aspect, they must be normalized into the same plane as the active user. This is because different users have preference patterns on different planes with different variances. Since their planes are not equal, the patterns between users cannot be converted to each other directly. To solve this problem, the variance normalization [45] is applied on both planes, to make a pair of different planes to be viewed as the same plane.

Suppose the task is to normalize $\mathbf{R}_{u_a}$ and $\mathbf{R}_{u_b}$, the multi-criteria ratings of user u_a and user u_b, respectively, to make them lie on the same plane. Let $r_{ajk} \in \mathbf{R}_{u_a}$ denotes the rating of u_a on item $v_j \in I_{u_a}$ under criterion $k \in K$. First, the mean rating value for u_a is calculated as:

$$\mu_{u_a} = \frac{1}{|I_{u_a}||K|} \sum_{i \in I_{u_a}, k \in K} r_{ajk} \tag{6.3.1}$$

The variance of ratings of user u_a is then calculated by:

$$var_{u_a} = \frac{1}{|I_{u_a}||K|} \sum_{i \in I_{u_a}, k \in K} (r_{ajk} - \mu_{u_a})^2 \tag{6.3.2}$$

Beside μ_{u_a} and var_{u_a}, μ_{u_b} and var_{u_b} for user u_b have to be calculated by the same way. Finally, the original rating r_{ajk} of u_a is normalized by the variance and lies on the combined plane of both user u_a and user u_b by:

$$r^V_{ajk(b)} = \frac{(r_{ajk} - \mu_u + \mu_{u,v})\sqrt{var_{u_a,u_b}}}{\sqrt{var_{u_a}}} \tag{6.3.3}$$

where μ_{u_a,u_b} is average of μ_{u_a} and μ_{u_b}, var_{u_a,u_b} is an average of var_{u_a} and var_{u_b}, and $r^V_{ajk(b)}$ is the normalized rating, called $variance normalized rating$. Let $\mathbf{R}^V_{u_a}$ and $\mathbf{R}^V_{u_b}$ respectively denote the variance normalized rating matrices for user u_a and u_b. Now the variance is the same for multi-criteria ratings of user u_a and user u_b.

6.4 Finding Multi-Criteria Rating Patterns

After the pair of planes of two users are in the same scale of variance, the principle component analysis or PCA is applied to extract each user multi-criteria rating pattern.

The principle component analysis (PCA) is usually serves as two main purposes in RS [87]. One is the dimensionality reduction, which has been used by many researchers for solving the sparsity problem [31, 52, 53, 98]. The other is to find patterns in high dimensional dataset in order to extract the most crucial characteristics from the data. This paper applies the latter one to find rating patterns of users on multi-criteria rating data. These patterns are then used to convert the user multi-criteria ratings into another user's aspect by projecting the original rating data onto the principle eigenvector (i.e. the eigenvector that has the highest eigenvalue).

By applying the PCA, the user multi-criteria rating patterns can be extracted from his variance normalized rating data from the previous step. In order to find the multi-criteria rating pattern from user u_a, the system firstly needs to create the normalized rating matrix $\mathbf{R}^M_{u_a}$ for user u_a. Each element of $\mathbf{R}^M_{u_a}$ (denoted by r^M_{ajk}) contains the subtraction between user u_a variance normalized rating given on item v_j under criterion k (denoted by r^V_{ajk}) normalized value is defined for corresponding user, and the mean value $\mu_{u_a k}$ of the variance normalized ratings of u_a in the same criteria k.

$$r^M_{ajk} = r^V_{ajk} - \mu_{u_a k} \tag{6.4.1}$$

Along with $\mathbf{R}^M_{u_a}$, the system needs to create the feature matrix $\mathbf{Q}_{u_a}$ of user u_a which represents the his multi-criteria rating patterns of the rated items, which can be done by PCA. First, the system finds the eigenvector $\mathbf{u}_{ak}$ of u_a's covariance rating matrix (covariance matrix of $\mathbf{R}^V_{u_a}$). After computing all eigenvectors, put the one that has highest eigenvalue to become the first row of $\mathbf{Q}_{u_a}$, the second highest to the second row, and so on.

6.5 Multi-Criteria Rating Conversion

After the multi-criteria ratings patterns of all users are extracted, they can be used to convert the rater multi-criteria ratings on each item into the active user's aspect.

Suppose the task is to convert rater u_a multi-criteria ratings on target item i into an active user u_b's aspect. First, the rating vector $\mathbf{r}_{aj}$ of item v_j is selected from $R^V_{u_a}$. This vector is then subtracted by its mean value to create the *modified rating vector* $\mathbf{r}^M_{aj}$. To convert to the

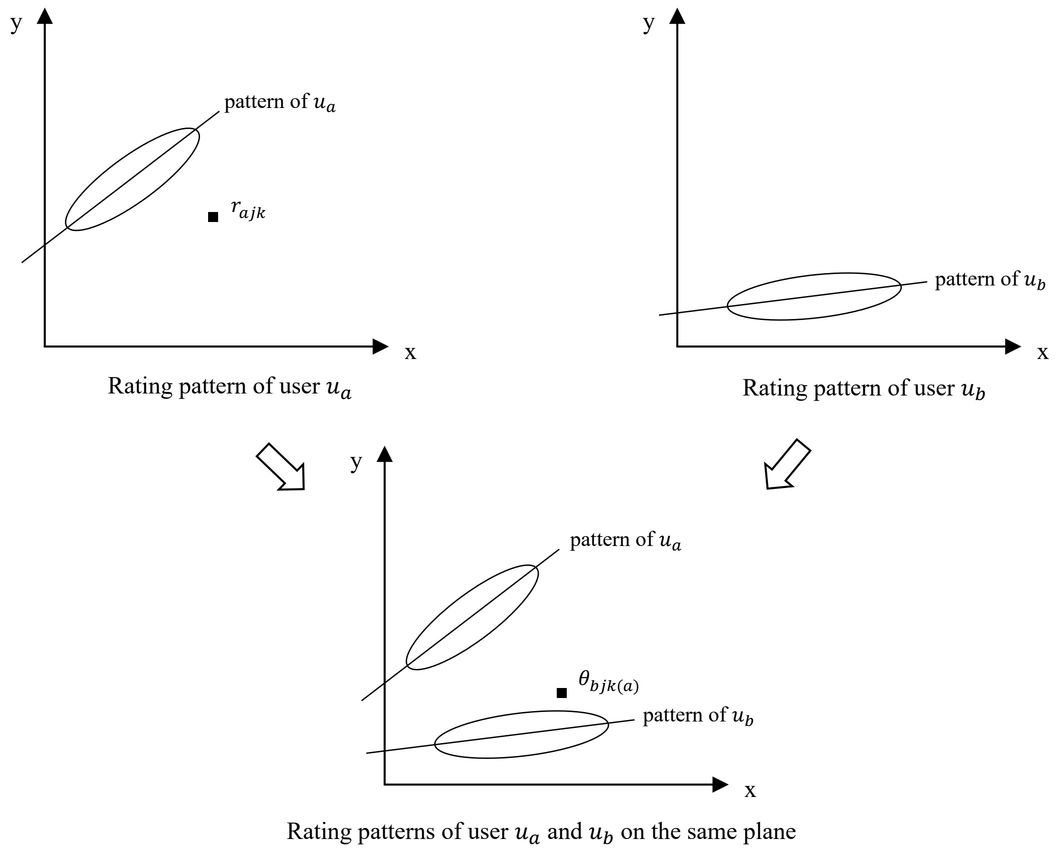

y
pattern of u_a
r_{ajk}
x
Rating pattern of user u_a
y
pattern of u_b
x
Rating pattern of user u_b
y
pattern of u_a
$\theta_{bjk(a)}$
pattern of u_b
x
Rating patterns of user u_a and u_b on the same plane

Figure 6.2: The user pattern transformation.

aspect of u_b, $\mathbf{r}_{aj}^{M}$ is projected into the same plane of u_b by multiplying with the feature matrix $\mathbf{Q}_{u_b}$ of u_b by the following equation.

$$\sigma_{bj(a)} = \mathbf{Q}_{u_b}(\mathbf{r}_{aj}^{M})^T \tag{6.5.1}$$

Before it can be used further, $\sigma_{bj(a)}$ needs to be turned back into original rating scale by the following:

$$\theta_{bj(a)} = \sigma_{bj(a)} + \mu_{u_a} \tag{6.5.2}$$

where $\mu_{u_a} = [\mu_{u_a,1} \cdots \mu_{u_a,K}]$ is a mean rating vector which each element contains the mean value of ratings in each criterion of user u_a. The example of rating conversion process is presented by Figure 6.2, which transforms of the rating of u_a into pattern of u_b on the same plane.

6.6 Rating Prediction

After the multi-criteria ratings from the raters are converted into the active user u_b's aspect. An aggregation method is needed to aggregate each rater converted multi-criteria ratings on the target item v_j, ($\theta_{bj(a)}$) into single overall criterion rating. For the proposed method, the multiple linear regression is selected. First, the system calculates the criteria weight vector ω_{u_a} of the rater u_a by applying the multiple linear regression on the overall variance normalized rating vector $\mathbf{r}_{u_a}^{V_{Overall}}$ (as dependent variable) and multi-criteria variance normalized rating vector $\mathbf{r}_{u}^{V_{Criteria}}$ (as independent variable):

$$\omega_{u_a} = MLR(\mathbf{r}_{u_a}^{V_{Overall}}, \mathbf{r}_{u_a}^{V_{Criteria}}) \tag{6.6.1}$$

where $MLR(.,.)$ is the function of multiple linear regression.

After the criteria weight vector is calculated, the new overall rating for user u_a on item v_j under the active user u_b's aspect can be computed as follow.

$$\hat{r}_{aj}^{Overall} = \omega_{u_a}^{T} \cdot \theta_{bj(a)} \tag{6.6.2}$$

Finally, the rating of the active user u_b to rate the target item v_j is estimated by the average of the overall ratings from all raters $N(v_j)$ who have rated item v_j, as shown by the following.

$$\hat{r}_{bj} = \frac{\sum_{u_a \in N(v_j)} \hat{r}_{aj}^{Overall}}{|N(v_j)|} \tag{6.6.3}$$

6.7 Experimental Evaluation

6.7.1 Dataset

Two multi-criteria rating datasets, Yahoo Movie and TripAdvisor Hotel, are used for the experiment and evaluation of the models. The Yahoo Movie dataset[1] contains 2,550 ratings provided by 200 users over 1,345 movies. Users are able to give the ratings from F to A+ to each movie, which are converted to numerical range of 1 to 13 for the evaluation

[1]https://www.yahoo.com/entertainment/movies/

Table 6.1: Statistical data of Yahoo and TripAdvisor datasets.

	Yahoo Movie	TripAdvisor Hotel
Number of users	200	281
Number of items	1345	2173
Number of ratings	2550	8324
Rating scale	1-13	1-5
Number of criteria	5	7
Max rating count per user	18	93
Min rating count per user	7	16
Average rating count per user	12.75	29.62
Standard Deviation	1.2268	11.315

Table 6.2: Experimental results on Yahoo Movie Dataset.

Methods	Type	RMSE	%Coverage
Proposed Method	MC	3.5574	58.04
Warat's	SC	3.6678	60.74
MC-CF	MC	3.6924	22.98
MC-CF2	MC	3.6624	22.98
MC-Lathia's	MC	6.2121	12.63
MC-Warat's	MC	3.6230	58.94

purpose. Besides the overall rating, users also provide ratings on four criteria, which are acting, story, direction and visual criteria. On the other hand, 8,324 ratings were extracted from TripAdvisor 5-scale multi-criteria rating dataset[2] provided by 281 users over 2,173 items. Beside the overall rating, this dataset contains ratings on six criteria: cleanliness, location, rooms, service, sleep quality, and value. Table 6.1 summarizes the statistics of the two datasets. The five-fold cross validation was conducted to evaluate the performances of the models on both datasets.

Table 6.3: Experimental results on TripAdvisor Hotel Dataset.

Methods	Type	RMSE	%Coverage
Proposed Method	MC	1.1082	87.89
Warat’s	SC	1.1272	88.49
MC-CF	MC	1.2109	63.08
MC-CF2	MC	1.1712	63.08
MC-Lathia’s	MC	1.8633	64.04
MC-Warat’s	MC	1.1038	88.35

6.7.2 Evaluation Results

In order to evaluate the performance, the proposed method is compared with the current rating conversion methods on both single criteria and multi-criteria techniques, along with the well-known recommendation techniques. The evaluation results on Yahoo Movie and TripAdvisor datasets are presented in Tables 6.2 and 6.3, respectively. Note that in column ‘Type’, SC means single criterion and MC means multi-criteria recommendations.

In Table 6.2 and 6.3, MC represents the traditional multi-criteria collaborative filtering method [3] mentioned in Chapter 2.4.2. This method uses multi-criteria ratings only in neighbors selection process, but not in prediction. MC-CF2 is a variant of MC-CF that exploits multi-criteria ratings on both neighbor selection and prediction. Both MC-CF and MC-CF2 do not apply the rating conversion technique. Since both Warat’s and Lathia’s rating conversion methods are designed for single criteria rating conversion, their multi-criteria extensions: MC-Lathia and MC-Warat need to convert each criterion rating independently. Like the proposed method, the MLR is also applied as aggregation method for the prediction of MC-CF2, MC-Lathia and MC-Warat.

As presented in Tables 6.2 and 6.3, it can be summarized that the proposed method outperforms the others in the aspect of prediction accuracy, while producing almost the same level of prediction coverage as Warat’s conversion technique (both on single criterion and multi-criteria ratings). In contrast, MC-Lathia results in the worst performance on both accuracy and prediction coverage from all methods.

[2]http://www.cs.cmu.edu/ jiweil/html/hotel-review.html

6.8 Discussion

The performances of the evaluation methods are now discussed in detail. For a better explanation, the discussion is separated into four points. First, the performance of the traditional multi-criteria collaborative filtering (MC-CF) is discussed. This is followed by the comparison between two current rating conversion techniques: Lathia's and Warat's methods. Next, the proposed method is compared with the multi-criteria Warat's (MC-Warat) rating conversion technique. Finally, a discussion on how to apply the proposed method to the real-world applications is provided.

6.8.1 Multi-Criteria Collaborative Filtering

The only part of traditional multi-criteria collaborative filtering (MC-CF) that is relevant to multi-criteria principle is the process of finding user similarities. MC-CF computes the similarities among users on each criterion independently, and then aggregates such similarities from all criteria into single similarity by Eq. 2.4.2 as mentioned in Chapter 2.4.2. The aggregated similarities along with the overall ratings are then exploited in the neighborhood-based prediction, while other criteria ratings are being ignored. In order to prove that exploiting the multi-criteria ratings on the prediction improves the performance, The prediction of the method is slightly adjusted. In MC-CF2, instead of using the actual overall ratings, they are calculated by using the weighted average on the multi-criteria ratings with criteria weight ω_{u_a} used in Eq. (6.6.1) as the weights. The results from Tables 6.2 and 6.3 show that MC-CF2 is more accurate than MC-CF, which is derived by using the multi-criteria ratings on the prediction.

Since multi-criteria collaborative filtering technique relies on the ratings on co-rated items from neighbors to make the prediction, it has very low prediction coverage (e.g. only 23% on Yahoo dataset). In contrast, the proposed method which uses all ratings from the raters who rated target items, is resulted in better coverage (almost three times higher than MC-CF and MC-CF2 on Yahoo dataset). In the aspect of prediction accuracy, the proposed method also outperforms both MC-CF and MC-CF2. This is due to the fact that the multi-criteria ratings are converted from the raters into the active user's aspect before making a prediction, while both MC-CF and MC-CF2 use original multi-criteria ratings from the raters directly. Since, ratings from the neighbors or raters are used to calculate the predicted rating for the active user, RS can provide better accuracy when the problem about the different rating patterns among users is addressed.

6.8.2 Warat's and Lathia's Rating Conversion

The Warat's single criterion rating conversion has shown that their conversion technique is better than Lathia's method [56] in the single criterion framework. This experimental result also show that on multi-criteria scheme, MC-Warat method is also better than MC-Lathia method in both accuracy and prediction coverage. A very low prediction coverage of Lathia's method (e.g. only 12% on Yahoo dataset) is because the rating conversion of Lathia's method relies on the ratings on co-rated items for prediction. Due to the sparsity of the rating data, such co-rated items is limited—resulting in low coverage. In contrast, Warat's method exploited the pseudo ratings from matrix factorization to fulfill the sparse user rating matrix.

The poor accuracy of Lathia's method is occurred because it can only convert a rating to the range [-1, 1] of the original rating. This is not practically in the real world where a person rating's perspective can be significantly different from one another. In contrast, Warat's method concerned about distribution of active user ratings, which is able to convert rater rating into real scale of active user rating.

6.8.3 Multi-Criteria Warat's Rating Conversion

As shown in Tables 6.2 and 6.3, MC-Warat method achieves more accurate prediction than the single criteria Warat's method. This provides even more proof from section 6.8.1 that exploiting the multi-criteria ratings improves the predictive performance of the method. In the aspect of prediction coverage, MC-Warat method is slightly lower than single criteria Warat's method. This is because if there is at least one criterion rating that is inconvertible, it will result in the invalid prediction of the overall rating of that record.

Now the Warat's method is compared with the proposed method. Since MC-Warat method is better than single-criteria Warat's method, the propose method is compard only with the MC approach. Both of the proposed method and MC-Warat method have almost the same level of the prediction coverage because of the same reason explained earlier. Both proposed and MC-warat methods do not rely on the ratings on co-rated items since MC-Warat can use pseudo ratings while the proposed method apply PCA to extract rating patterns. This means both methods are able to exploit all ratings from the raters for the rating conversion. For the accuracy, the proposed method is better than MC-Warat method on Yahoo Movie dataset. This is because MC-Warat method converts ratings on each criterion independently, causing the loss of relation among the criteria ratings. In contrast, the proposed method converts ratings from all criteria simultaneously, which help maintaining the implicit relationships among the criteria ratings—resulting in a better prediction. The proposed method provides almost similar level of accuracy as MC-Warat on TripAdvisor dataset. This is because the rating range of TripAdvisor is [1, 5] which is narrower than Yahoo Movie dataset which is [1, 13].

6.8.4 Applications of Multi-Criteria Recommendations

Sections 6.8.1 and 6.8.3, demonstrated that the proposed method outperforms the others in the predictive performances. In this section the merits of applying the method to the real-world scenarios is discussed. Many web-based applications let a user rates on multiple aspects of an item. For example, on Agoda[3] and Booking.com[4], users can rate a hotel room under criteria such as location, cleanliness, services and value. A rating under each criterion, however, might has dependency with those from other criteria. For example, John might have high concern on location and service when judging the value of the room, meaning that when he rates high scores on location and service, he tends to rate high score on value as well. In contrast, when he rates low score on those two criteria, he also rates lower score on value. If a single criterion rating conversion technique is applied to convert each criterion rating separately, those relation across criteria ratings might be lost. For example, there is Alex who has a habit on giving very low ratings on location and service to any room. Suppose the task is to convert the 10-scale ratings from John, who rates ‘9’ on location and ‘10’ on service which consequently rates ‘9’ on value, to the aspect of Alex. By applying single criteria rating conversion technique, ‘10’ and ‘9’ from location and service from John might be converted to ‘2’ or ‘1’ for Alex since he always rates very low scores on those two criteria, while ‘9’ on value might be converted to a much higher value (such as ‘9’ or ‘10’) because there is no bias on value criterion from Alex. As a result, the ratings (‘9’, ‘10’, ‘9’) on location, service and value from John could be converted to, for example, (‘1’, ‘2’, ‘9’) for Alex, meaning that the previous relation among those criteria from John has been forever lost. To address this problem, the propose method, on the other hand, simultaneously convert all multi-criteria ratings to maintain the implicit relation among those criteria. Again, with the previous example, the proposed method would convert the ratings (‘9’, ‘10’, ‘9’) from John to (‘1’, ‘2’, ‘1’) for Alex, which means the dependency among those ratings from John remains untouched. I believe that this makes the proposed method more suitable for real-world situations, where recommendation engines should take into account the users’ unique patterns in providing multiple aspects of their preferences toward items, for personalizing the recommendations.

[3]https://www.agoda.com
[4]https://www.booking.com

6.9 Conclusion

In this work, a novel method for simultaneously converting the multi-criteria ratings from one user to another user's aspect is proposed. This method converts the multi-criteria ratings simultaneously into the same space in order to maintain the implicit relation among the criteria ratings. Not only producing the high level prediction coverage, the proposed method also outperforms current rating conversion techniques on both single and multi-criteria ratings with RMSEs of 1.1082 and 3.5574 on TripAdvisor hotel and Yahoo movie datasets respectively. These experiment results prove the following assumptions on multi-criteria recommendations and the ratings conversion. First, exploiting the multi-criteria rating data helps improving the performance of the predictive model, compare to the single rating approaches. Moreover, the ratings from the raters (all users who have rated the target item, not only the neighbors) should be converted into the same aspect as the active user before they can be used further in prediction step. Finally, the multi-criteria ratings should be simultaneously converted together to maintain the implicit relationship among the criteria ratings.

The proposed method can be utilized in a wide range of applications that concern various aspects of users' opinions for making recommendations. For example, in some hotel booking websites users can rate hotel rooms separately by its location, cleanliness, service or value. A rating of each criterion, however, might be depended with those from other criteria. For example, some users might rate the value of a hotel room based on its location and services. With the proposed method, when the multi-criteria ratings from one user is converted into another user's aspect, those unique patterns of the dependencies among the multi-criteria ratings will be maintained. I believe that this makes the proposed model more suitable for a real-world situation where a rating pattern of an individual user is unique and different from others.

Chapter 7

Summary

This thesis presents a novel method for automatically extracting contexts from reviews, as well as an effective method for utilizing the extracted contexts for rating prediction. In addition, this thesis also introduces a novel multi-criteria rating conversion technique. These methods were proposed to produce effective recommendations with respect to prediction accuracy and rating sparsity.

First, a novel unsupervised method for defining and extracting contexts from reviews, namely CARE, is proposed [91]. Unlike any previous work, a relevant context in this work is defined and extracted not only in the single word format, but also includes its neighboring words that have influences on distributions of ratings on a review data. A region embedding technique is derived to emphasize the words in a small text region to be considered as a context, and represent it by a region embedding to be used for a rating prediction. The extensive experiments showed that CARE is able to extract the set of unique contexts from any specific recommendation domain. Moreover, the extracted contexts effectively explain the distribution of ratings in reviews, which is useful for modeling the polarity of the reviews' ratings.

Moreover, an extended model of CARE for rating prediction, namely CARE-AI, is proposed [92]. This model introduces the interaction and attention modules, which help constructing a user and item representations based on different levels of relevance among contexts extracted from CARE with an individual user preferences and item features. Unlike most deep learning-based methods that learn one static user or item representation for all reviews, CARE-AI dynamically generates a unique user and item representations for each particular review, which are more proper for capturing the specific contextual information embeded in that review. The experiments demonstrated that CARE-AI produced more accurate prediction accuracy than the state-of-the-art rating prediction methods including review-based and context-aware recommendation techniques on both normal and sparsity situations.

Finally, a novel rating conversion method for multi-criteria ratings is proposed [90]. This method applies the principle component analysis extract multi-criteria preference patterns from users. Such patterns are then used for simultaneously converting the multi-criteria rating, which preserves the implicit relation among criteria ratings. The experiment results show that proposed multi-criteria rating conversion technique outperforms both current single and multi-criteria rating conversion techniques with higher accuracy on two multi-criteria rating datasets, while maintaining considerably high level of prediction coverages.

For future work, there are three further challenges that I aim to achieve.

- First, the current rating prediction of CARE-AI assumes that the review data of a target item is available at prediction phrase, which is exploited as the main source to extract contexts. In fact, since the target item is not yet being rated, its review, therefore, is not available. The model would be more useful in real-word if it is capable of providing recommendations for any item without requiring their reviews from target users.

- Second, currently the proposed context extraction (CARE) and rating prediction (CARE-AI) are two independent steps. I believe that a model with an ability to identify contexts and predict rating in a single step would produce more efficient recommendations.

- Finally, in Chapter 6, it has been shown that my proposed model for multi-criteria recommendation could produce satisfying rating prediction results. However, one main concern for the multi-criteria recommendation approach is that, it requires explicit ratings in multiple aspects of items from the users. Similar to the predefined types of contexts, such aspects of items are predefined by the system, which are very limited to a specific recommendation domain. In fact, in most recommendation domains, users provide only the overall ratings as well as their reviews, whereas the multi-criteria ratings are not available. The existing multi-criteria recommendation techniques, therefore, are not applicable for this kind of situation. Fortunately, as presented by the example of suitable applications for CARE in Chapter 4.7.5, a portion of extracted words by CARE, such as “clean”, “location”, or “service”, could be considered as the aspects of items. I strongly believe that it is be possible to infer the multi-criteria user preferences or item features from reviews, and utilizes them to produce a novel multi-criteria recommendation technique that requires no explicit multi-criteria ratings on predefined aspects of items.

Publication list

Journals

- Padipat Sitkrongwong, Atsuhiro Takasu, and Saranya Maneeroj, “Context-Aware User and Item Representations Based on Unsupervised Context Extraction from Reviews”, IEEE Access, vol. 8, pp. 87094–87114, 2020.
- Padipat Sitkrongwong, Saranya Maneeroj, and Atsuhiro Takasu, “Multi-Criteria Rating Conversion Without Relation Loss For Recommender Systems”, International Journal of Computers and Applications, December 2019.

International conference

- Padipat Sitkrongwong and Atsuhiro Takasu, “Unsupervised Context Extraction via Region Embedding for Context-Aware Recommendations”, In 23rd International Database Engineering & Applications Symposium (IDEAS’19), pp. 16:1–16:10, June 2019. (Full Paper)

Domestic conference

- Padipat Sitkrongwong, Saranya Maneeroj and Atsuhiro Takasu, "User Bias Manipulation for Multi-Criteria Recommendation", The 12th International Workshop on Information Search, Integration, and Personalization (ISIP2018), Fukuoka, Japan, May 2018. (Oral)

Bibliography

[1] Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., and Steggles, P. (1999). Towards a better understanding of context and context-awareness. In *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing*, HUC '99, pages 304–307, Berlin, Heidelberg. Springer-Verlag.

[2] Aciar, S. (2010). Mining context information from consumers reviews. In *Proceedings of Workshop on Context-Aware Recommender System*, volume 201. ACM.

[3] Adomavicius, G. and Kwon, Y. (2007). New recommendation techniques for multicriteria rating systems. *IEEE Intelligent Systems*, 22(3):48–55.

[4] Adomavicius, G., Manouselis, N., and Kwon, Y. (2011). *Multi-Criteria Recommender Systems*, pages 769–803. in Recommender systems handbook, Springer.

[5] Adomavicius, G., Sankaranarayanan, R., Sen, S., and Tuzhilin, A. (2005). Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems*, 23(1):103–145.

[6] Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *in IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749.

[7] Adomavicius, G. and Tuzhilin, A. (2015). Context-aware recommender systems. In *Recommender Systems Handbook, US*, pages 191–226. Springer USA.

[8] Aggarwal, C. C., Wolf, J. L., Wu, K.-L., and Yu, P. S. (1999). Horting hatches an egg: A new graph-theoretic approach to collaborative filtering. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, pages 201–212, New York, NY, USA. Association for Computing Machinery.

[9] Amatriain, X. and Basilico, J. (2015). *Recommender Systems in Industry: A Netflix Case Study*. Recommender Systems Handbook. Springer, Boston, MA.

[10] Baltrunas, L., Ludwig, B., Peer, S., and Ricci, F. (2012). Context relevance assessment and exploitation in mobile recommender systems. *Personal and Ubiquitous Computing - PUC*, 16:1–20.

[11] Baltrunas, L., Ludwig, B., and Ricci, F. (2011). Matrix factorization techniques for context aware recommendation. In *Proceedings of the fifth ACM conference on Recommender systems - RecSys '11*, pages 301–304.

[12] Baltrunas, L. and Ricci, F. (2009). Context-based splitting of item ratings in collaborative filtering. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 245–248, New York, NY, USA. Association for Computing Machinery.

[13] Bauman, K. and Tuzhilin, A. (2014). Discovering contextual information from user reviews for recommendation purposes. In *CEUR Workshop Proceedings*, volume 1245, pages 2–8.

[14] Bazire, M. and Brezillon, P. (2005). Understanding context before using it. In *Proceedings of the 5th International Conference on Modeling and Using Context*, CONTEXT'05, pages 29–40, Berlin, Heidelberg. Springer-Verlag.

[15] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

[16] Brown, P. J. (1996). The stick-e document: a framework for creating context-aware applications. In *Electronic Publishing '96*, pages 259–272.

[17] Campos, P., Fernández-Tobías, I., Cantador, I., and Rubio, F. (2013). Context-aware movie recommendations: An empirical comparison of pre-filtering, post-filtering and contextual modeling approaches. In *E-Commerce and Web Technologies. EC-Web 2013. Lecture Notes in Business Information Processing*, volume 152, pages 137–149.

[18] Chalermpornpong, W., Maneeroj, S., and Atsuhiro, T. (2013). Rating pattern formation for better recommendation. In *2013 24th International Workshop on Database and Expert Systems Applications*, pages 146–151.

[19] Chen, C., Zhang, M., Liu, Y., and Ma, S. (2018). Neural attentional rating regression with review-level explanations. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW'18*, pages 1583–1592, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

[20] Chen, G. and Chen, L. (2014). Recommendation based on contextual opinions. In *User Modeling, Adaptation and Personalization*, pages 61–73. Springer International Publishing.

[21] Chen, L., Chen, G., and Wang, F. (2015). Recommender systems based on user reviews: the state of the art. *User Modeling and User-Adapted Interaction*, 25(2):99–154.

[22] Chen, Y., Wu, C., Xie, M., and Guo, X. (2011). Solving the sparsity problem in recommender systems using association retrieval. *Journal of Computers*, 6:1896–1902.

[23] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

[24] Devlin, J., Chang, M., Lee, K., and Kristina, T. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT 2019*, pages 4171–4186.

[25] Dey, A. (1998). Context-aware computing: The cyberdesk project. In *AAAI 1998 Spring Symposium on Intelligent Environments*, pages 51–54.

[26] Dey, A. K., Abowd, G. D., and Wood, A. (1998). Cyberdesk: A framework for providing self-integrating context-aware services. *Knowledge-Based Systems*, 11(1):3–13.

[27] Diederik, K. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of 3rd International Conference on Learning Representations - ICLR'15*, San Diego, CA, USA.

[28] Dourish, P. (2004). What we talk about when we talk about context. *Personal and Ubiquitous Computing*, 8(1):19–30.

[29] Esparza, S., O'Mahony, M., and Smyth, B. (2010). Effective product recommendation using the real-time web. In *Proceedings of the 15th ACM international conference on Information and knowledge management (CIKM '06)*, pages 5–18.

[30] Franklin, D. and Flachsbart, J. (2001). All gadget and no representation makes jack a dull environment. In *AAAI 1998 Spring Symposium on Intelligent Environments*, pages 155–160.

[31] Goldberg, K., Roeder, T., Gupta, D., and Perkins, C. (2001). Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151.

[32] Gong, S., Ye, H., and Dai, Y. (2009). Combining singular value decomposition and item-based recommender in collaborative filtering. In *2009 Second International Workshop on Knowledge Discovery and Data Mining*, pages 769–772.

[33] Good, N., Schafer, J. B., Konstan, J. A., Borchers, A., Sarwar, B., Herlocker, J., and Riedl, J. (1999). Combining collaborative filtering with personal agents for better recommendations. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*, AAAI '99/IAAI '99, pages 439–446, USA. American Association for Artificial Intelligence.

[34] Grčar, M., Mladenić, D., Fortuna, B., and Grobelnik, M. (2006). Data sparsity issues in the collaborative filtering framework. In *Advances in Web Mining and Web Usage Analysis: 7th International Workshop on Knowledge Discovery on the Web, WebKDD 2005*, pages 58–76.

[35] Gunawardana, A. and Shani, G. (2009). A survey of accuracy evaluation metrics of recommendation tasks. *in Journal of Machine Learning Research*, 10:2935–2962.

[36] Gunawardana, A. and Shani, G. (2015). *Evaluating Recommender Systems in Recommender Systems Handbook.* Springer, Boston, MA.

[37] Hariri, N., Mobasher, B., and Burke, R. (2012). Context-aware music recommendation based on latenttopic sequential patterns. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 131–138, New York, NY, USA. Association for Computing Machinery.

[38] Hariri, N., Mobasher, B., Burke, R., and Zheng, Y. (2011). Context-aware recommendation based on review mining. In *Proceedings of the 9th International Workshop on Intelligent Techniques for Web Personalization and Recommender Systems - ITWP 2011*, pages 30–36.

[39] He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. (2017). Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 173–182, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

[40] Herlocker, J. L., Konstan, J. A., Borchers, A., and Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 230–237, New York, NY, USA. Association for Computing Machinery.

[41] Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 50–57, New York, NY, USA. Association for Computing Machinery.

[42] Huang, Z., Chung, W., Ong, T.-H., and Chen, H. (2002). A graph-based recommender system for digital library. In *Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '02, pages 65–73, New York, NY, USA. Association for Computing Machinery.

[43] Jin, R. and Si, L. (2004). A study of methods for normalizing user ratings in collaborative filtering. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 568–569.

[44] Jin, R., Si, L., Zhai, C., and Callan, J. (2003). Collaborative filtering with decoupled models for preferences and ratings. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, pages 309–316.

[45] Joshi, V., Prasad, N., and Umesh, S. (2015). Modified mean and variance normalization: Transforming to utterance-specific estimates. *Circuits, Systems, and Signal Processing*, 35.

[46] Karatzoglou, A., Amatriain, X., Baltrunas, L., and Oliver, N. (2010). Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 79–86, New York, NY, USA. Association for Computing Machinery.

[47] Kim, D., Park, C., Oh, J., Lee, S., and Yu, H. (2016). Convolutional matrix factorization for document context-aware recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems - RecSys '16*, pages 233–240.

[48] Klema, V. and Laub, A. (1980). The singular value decomposition: Its computation and some applications. *IEEE Transactions on Automatic Control*, 25(2):164–176.

[49] Koren, Y. (2009). Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD09)*, pages 447–456, New York, NY, USA. Association for Computing Machinery.

[50] Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

[51] Kosir, A., Tasic, J., Odic, A., and Tkalcic, M. (2012). Relevant context in a movie recommender system: Users' opinion vs. statistical detection. *CEUR Workshop Proceedings*, 889.

[52] Kozma, L., Ilin, A., and Raiko, T. (2009). Binary principal component analysis in the netflix collaborative filtering task. In *2009 IEEE International Workshop on Machine Learning for Signal Processing*, pages 1 – 6.

[53] Kumaravel, A. and Dutta, P. (2014). Application of pca for context selection for collaborative filtering. *Middle - East Journal of Scientific Research*, 20:88–93.

[54] Lahlou, F. Z., Benbrahim, H., Mountassir, A., and Kassou, I. (2013a). Inferring context from users' reviews for context aware recommendation. In *Research and Development in Intelligent Systems XXX, SGAI 2013*, pages 227–239. Springer, Cham.

[55] Lahlou, F. Z., Mountassir, A., Benbrahim, H., and Kassou, I. (2013b). A text classification based method for context extraction from online reviews. In *8th International Conference on Intelligent Systems: Theories and Applications (SITA)*, pages 1–5. Rabat.

[56] Lathia, N., Hailes, S., and Capra, L. (2008). Trust-based collaborative filtering. In *Trust Management II*, pages 119–134, Boston, MA. Springer US.

[57] Lee, D. D. and Seung, H. S. (2000). Algorithms for non-negative matrix factorization. In *Proceedings of the 13th International Conference on Neural Information Processing Systems - NIPS '00*, pages 535–541.

[58] Li, Y., Nie, J., Zhang, Y., Wang, B., Yan, B., and Weng, F. (2010). Contextual recommendation based on text mining. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters - COLING '10*, pages 692–700.

[59] Li, Y., Zhang, D., Lan, Z., and Tan, K.-L. (2016). Context-aware advertisement recommendation for high-speed social news feeding. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 505–516.

[60] Liu, H., Hu, Z., Mian, A., Tian, H., and Zhu, X. (2014). A new user similarity model to improve the accuracy of collaborative filtering. *Knowledge-Based Systems*, 56(C):156–166.

[61] Liu, L., Lecue, F., Mehandjiev, N., and Xu, L. (2010). Using context similarity for service recommendation. In *IEEE Internet Computing - INTERNET*, pages 277 – 284.

[62] Lops, P., de Gemmis, M., and Semeraro, G. (2011). *Content-based Recommender Systems: State of the Art and Trends*. in Recommender Systems Handbook. Springer, Boston, MA.

[63] Manotumruksa, J., Macdonald, C., and Ounis, I. (2017). Matrix factorisation with word embeddings for rating prediction on location-based social networks. In *Advances in Information Retrieval. ECIR 2017. Lecture Notes in Computer Science, vol 10193*, pages 647–654.

[64] McAuley, J. and Leskovec, J. (2013). Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems - RecSys '13*, pages 165–172.

[65] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - NIPS*, pages 3111–3119.

[66] Moshfeghi, Y., Piwowarski, B., and Jose, J. M. (2011). Handling data sparsity in collaborative filtering using emotion and semantic based features. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 625–634, New York, NY, USA. Association for Computing Machinery.

[67] Musto, C., Semeraro, G., Degemmis, M., and Lops, P. (2015). Word embedding techniques for content-based recommender systems: An empirical evaluation. In *RecSys Posters*.

[68] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, Madison, WI, USA. Omnipress.

[69] Ni, J., Li, J., and McAuley, J. (2019). Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *2019 Conference on Empirical Methods in Natural Language Processing - EMNLP'19*, pages 188–197, Hong Kong, China.

[70] Odić, A., Tkalčič, M., Tasič, F. J., and Košir, A. (2013). Predicting and detecting the relevant contextual information in a movie-recommender system. *Interacting with Computers*, 25:74–90.

[71] Panniello, U., Tuzhilin, A., Gorgoglione, M., Palmisano, C., and Pedone, A. (2009). Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 265–268, New York, NY, USA. Association for Computing Machinery.

[72] Pearson, K. (1920). Notes on the history of correlation. *Biometrika*, 13(1):25–45.

[73] Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP*, pages 1532–1543.

[74] Perera, C., Zaslavsky, A., Christen, P., and Georgakopoulos, D. (2014). Context aware computing for the internet of things: A survey. *IEEE Communications Surveys & Tutorials*, 16(1):414–454.

[75] Peña, F. J. (2017). Unsupervised context-driven recommendations based on user reviews. In *Proceedings of the Eleventh ACM Conference on Recommender Systems - RecSys '17*, pages 426–430.

[76] Poirier, D., Fessant, F., and Tellier, I. (2010). Reducing the cold-start problem in content recommendation through opinion classification. In *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, volume 1, pages 204–207.

[77] Qiao, C., Huang, B., Niu, G., Li, D., Dong, D., He, W., Yu, D., and Wu, H. (2018). A new method of region embedding for text classification. In *6th International Conference on Learning Representations - ICLR '18, Poster, Vancouver, BC, Canada.*

[78] Rendle, S. (2012). Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology*, 3(3):1–22.

[79] Roux, F., Ranjeet, E., Ghai, V., Gao, Y., and Lu, J. (2007). A course recommender system using multiple criteria decision making method. *International Journal of Computational Intelligence Systems and Knowledge Engineering 2007*, pages 1407–1411.

[80] Ryan, N., Pascoe, J., and Morse, D. (1997). Enhanced reality fieldwork: the context-aware archaeological assistant. In *Computer Applications and Quantitative Methods in Archaeology (CAA97), Oxford.*

[81] Salakhutdinov, R. and Mnih, A. (2007). Probabilistic matrix factorization. In *Proceedings of the 20th International Conference on Neural Information Processing Systems - NIPS '07*, pages 1257–1264.

[82] Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA. Association for Computing Machinery.

[83] Schilit, B., Adams, N., and Want, R. (1994). Context-aware computing applications. In *1994 First Workshop on Mobile Computing Systems and Applications*, pages 85–90.

[84] Schilit, B. N. and Theimer, M. M. (1994). Disseminating active map information to mobile hosts. *IEEE Network*, 8(5):22–32.

[85] Schmitt, C., Dengler, D., and Bauer, M. (2002). The maut-machine: An adaptive recommender system. In *Proceedings of the ABIS Workshop, Hannover, Germany*, Hasselt, Belgium.

[86] Seo, S., Huang, J., Yang, H., and Liu, Y. (2017). Interpretable convolutional neural networks with dual local and global attention for review rating prediction. In *Proceedings of the Eleventh ACM Conference on Recommender Systems - RecSys '17*, pages 297–305.

[87] Shlens, J. (2014). A tutorial on principal component analysis.

[88] Sitkrongwong, P., Maneeroj, S., Samatthiyadikun, P., and Takasu, A. (2015). Bayesian probabilistic model for context-aware recommendations. In *Proceedings of the 17th International Conference on Information Integration and Web-based Applications & Services - iiWAS '15*, volume 22, pages 1–10.

[89] Sitkrongwong, P., Maneeroj, S., and Takasu, A. (2013). Latent probabilistic model for context-aware recommendations. In *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volume 1, pages 95–100.

[90] Sitkrongwong, P., Maneeroj, S., and Takasu, A. (2019). Multi-criteria rating conversion without relation loss for recommender systems. *International Journal of Computers and Applications*.

[91] Sitkrongwong, P. and Takasu, A. (2019). Unsupervised context extraction via region embedding for context-aware recommendations. In *Proceedings of the 23rd International Database Applications & Engineering Symposium - IDEAS '19, Article 16,*, pages 1–10.

[92] Sitkrongwong, P., Takasu, A., and Maneeroj, S. (2020). Context-aware user and item representations based on unsupervised context extraction from reviews. *IEEE Access*, 8:87094–87114.

[93] Tangphoklang, P., Maneeroj, S., and Takasu, A. (2011). A novel weighting scheme for a multi-criteria rating recommender system. In *IADIS International Conference Information Systems 2011*, pages 21–29.

[94] Unger, M. (2015). Latent context-aware recommender systems. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, pages 383–386, New York, NY, USA. Association for Computing Machinery.

[95] van der Maaten, L. and Hinton, G. (2008). Viualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605.

[96] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems - NIPS '17*, pages 6000–6010.

[97] Wang, C. and Blei, D. M. (2011). Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '11*, pages 448–456.

[98] Wang, Z., Yu, X., Feng, N., and Wang, Z. (2014). An improved collaborative movie recommendation system using computational intelligence. *Journal of Visual Languages & Computing*, 25:667–675.

[99] Ward, A., Jones, A., and Hopper, A. (1997). A new location technique for the active office. *IEEE Personal Communications*, 4(5):42–47.

[100] Webster, N. (1980). *Webster's new twentieth century dictionary of the English language*. Merriam-Webster, Inc, Springfield, MA.

[101] Wu, L., Quan, C., Li, C., Wang, Q., Zheng, B., and Luo, X. (2019). A context-aware user-item representation learning for item recommendation. *ACM Transactions on Information Systems*, 37(2):1–29.

[102] Yoon, K. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751.

[103] Zheng, L., Noroozi, V., and Yu, P. S. (2017). Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining - WSDM '17*, pages 425–434.