The Pennsylvania State University
The Graduate School

MODELING AND INFLUENCING ONLINE USER BEHAVIOR WITH MACHINE LEARNING

A Dissertation in
Information Sciences and Technology
by
Jiasheng Zhang

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2020

The dissertation of Jiasheng Zhang was reviewed and approved by the following:

Abstract

The Internet reshapes the way people connect to the world. The online platforms such as social media and e-commerce websites collect a large number of various users' behavior data (e.g., retweets in Twitter or orders in Amazon). Such data provides an unprecedented opportunity to apply machine learning methods to modeling user behavior and further influencing it to improve the way people connect to the world. In this thesis, several specific real-world scenarios related to modeling and influencing user behavior are studied using various machine learning methods. On the other hand, challenges and observations from those scenarios inspire improvement on general machine learning methods.

First, a challenge from representation of user behavior is studied. When emotional reactions of users toward posts in social media (i.e., Facebook) are represented as a ranking over a given set of emoticons, the task to predict the users' reactions given post content can be formulated as a label ranking problem. On the other hand, the imbalance property in emotional reaction data requires robustness in both label ranking performance measure and algorithms. Second, what influences user behavior is studied. More specifically, the task is to find the influence of news channels on their readers' reactions besides the content they post. It can be formulated as a multi-task learning problem. On the other hand, an observation, that the influence from channels can be different for different news content, inspires a novel multi-task learning architecture. Third, how to influence user behavior is studied. More specifically, an industry problem, recommending search story to improve search experience of users, is solved. In order to address the cross-channel challenge, a reinforcement learning framework is designed. Finally, this work is closed by a discussion on a future direction, that what users can do to combat the influence of the systems powered by machine learning methods. Recommendation problem is used as a playground to illustrate its necessity.

Table of Contents

List of Figures

List of Tables

Acknowledgments

This work was supported in part by NSF CNS-1422215, NSF IUSE1525601, NSF CNS-1742702 and Samsung GRO 2015 awards.Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsors.

Chapter 1 | Introduction

The Internet reshapes the way people connect to the world. The world includes other people, i.e., users of social media platforms (e.g., Facebook, Twitter), and also items, i.e., goods in e-commerce platforms (e.g., Amazon, JD.com). Those platforms easily collect a large number of user's behaviors, such as retweeting a post she finds interested in Twitter, or clicking an item recommended in JD.com. That amount of data provides an unprecedented opportunity to apply machine learning methods to modeling user behavior and further influencing it to improve the way people connect to the world.

1.1 Background and Scope

In this section, some related areas are introduced as background and compared with this proposal to clarify the scope here.

1.1.1 Sentiment Analysis

The user behavior in social media platforms often involves users' interaction with, or reactions to user generated contents, i.e., posts. The analysis of the relationship between users' reactions and the content they read is related to but not exactly the same with sentiment analysis.

There are a significant amount of works in analyzing human emotion underlying certain content. The term sentiment analysis, usually refers to mining emotions of text content, but here it can be generalized to any content forms. The major difference with this thesis is that sentiment analysis tries to recognize emotions

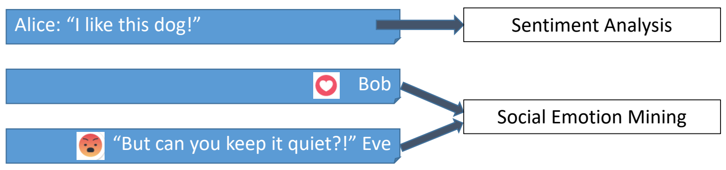


Figure 1.1: Examples of sentiment analysis and Social Emotion Mining

conveyed by authors via some form of content; while here, I try to analyze the emotions of readers triggered by some form of content. An illustrating example can be found in Fig. 1.1

This difference makes most conventional sentiment analysis works not able to be applied to our problem directly. A key difficulty can be shown by the above example, where the quoted sentence of the first example is clearly emotional, while that of the second is more neutral. This can be summarized as the difference between author’s and readers’ emotion.

Instead of digging deeper in the textual content itself as in sentiment analysis, this work turns to the representation of users’ reactions and other meta data that may influence users’ reactions besides content.

1.1.2 Recommendation

When it comes to user behavior as their interactions with items in e-commerce platforms, one of the most well-known problem dealt with machine learning is recommendation. The goal is to help each user find her desired items based on her previous behavior, when the number of items is so large that prohibits exhausted search of any user. When it is formulated as a machine learning problem, it is a problem that given part of users behavior, a method is required to predict the remaining part.

The recommendation problem, because of its wide applications in real world, is taken as a playground to test our proposals in this work. First, in order to observe how we can influence users’ behavior, we use an industrial problem as recommending search stories to influence users’ search behavior in an e-commerce platform. Second, the interplay between the recommender system and users provides a good playground to show what users can do to combat the control of machines (i.e., recommendation system).

In all the adoption cases of recommendation problem in this work, only the original goal but not the machine learning formulation of recommendation is used.

1.2 Subproblems in Different Aspects

In this work, I study several sub-problems as different aspects of modeling and influencing user behavior with machine learning

1.2.1 How to represent user behavior?

User behavior can be represented in different forms in different scenarios, such as retweeting a post in twitter and click of items on e-commerce platforms. The representation should be both enough to convey users' reactions, but also easy for other users to understand. Some representations bring extra challenges to machine learning methods.

In this work, we study the problem of finding the relationship between online content and emotional reactions it triggered in the readers in Facebook, which is called SEM (Social Emotion Mining). For example, what is the emotional reactions of all readers of the news example shown in Sec. 1.1.1. Here in this scenario, the user behavior is represented as the ranking of six emoticons given number of votes on each by readers. The rationale behind this representation can be seen from a comparison with two alternatives. On one hand, if the representation is simply the number of votes of each emoticons, though the users' reactions are fully conveyed, the extra information makes other users more difficult to obtain the main idea of users' reactions quickly. On the other hand, the representation as the most voted emoticon is not sufficient to show users' emotional reactions. Because it is not the focus of this work to analyze or design the best representation, the fact that such ranking representation is deployed in a popular social media platform, i.e., Facebook, is enough to study it.

When users' emotional reactions are represented as rankings of emoticons, the problem can be straightforwardly formulated as a label ranking problem in machine learning. However, the nature of emotional reactions bring an extra challenge, imbalance, to solving the problem as label ranking. In this work, we propose formal definition of imbalance in label ranking context, a performance measure that is robust in imbalanced data and a robust label ranking model [1].

1.2.2 What influences user behavior?

User behavior can be influenced by many factors. Some of them are obvious, i.e., the influence of post content on reader's emotional reactions, while other implicit factors are also worth studying.

In this work, I aim at the problem of finding the influence of news channels on their readers' reactions besides the content they post. For instance, emotional reactions to news of similar if not the exact same content can be various given different news channels. The cause can be two-fold. On one hand, the readers toward different news channels can be different, which naturally results in different reactions because of different stance; on the other hand, the cumulative tastes of news channels enforce prior impression in their readers. Here influence of news channels is used to warp the two factors. The problem is formulated as a multi-task learning problem, where each news channel is considered a task. The influence of news channels can be represented by the task difference. The answer format it will provide is that given some content as input, what will be the reactions of readers as output, given the channel as a task.

An observation in the news channel influence evokes a new challenge or opportunity in multi-task learning problem, i.e., the influence from channels can be different for different news content. For instance, readers' reactions can be highly consistent across different news channels for news on the topics of natural events and festivals; however, different standing points of different channels often result in exactly opposite readers' reactions for the news on the topics of sports and political news. Inspired by this observation, I propose a new multi-task learning method in this work. It obtains better performance by noticing the task relationship difference in different groups of data.

1.2.3 How to influence user behavior?

The objective of user behavior modeling for business platforms (e.g., e-commerce platforms) is to influence user behavior and create more profit. However, conventional machine learning (i.e., supervised and unsupervised learning) is only finding relationship among data. For example in recommendation problem, the machine learning solution only predicts the relationship among items (i.e., items a user will click if she has clicked certain items), but not tries to influence user behavior on purpose. Notice that the recommendation system does influence user behavior, as can be seen later, but not on the purpose of the machine learning solution.

In this work, I take a reinforcement learning framework to influence users' searching experience by recommending search stories to them, which is called personalized search story recommendation. This application scenario illustrates the difficulty of influencing user behavior. The actions taken by the system is recommending search stories, while the goal is to influence users' searching experience. It is different from conventional recommendation problem where the actions taken by the system is recommending items, and the goal is to recommend the “correct" items (i.e., which the user will click). This cross-channel difficulty can be addressed by reinforcement learning.

1.2.4 What users can do?

The Sec. 1.2.3 describe that machine learning methods can be applied to influence user behavior on purpose. However, user behavior is more often shaped by systems powered by machine learning methods unconsciously. In this case, are users capable to combat such influence?

In the end of this work, I propose an intelligent user model to test whether user can obtain better recommendation results from a recommendation system by reacting intelligently. The recommendation systems based on machine learning methods are known to provide sub-optimal results for users, though it shows great success in applications. As people become more and more relying on recommendation system, such sub-optimal results can be severe. Instead of designing more advanced recommendation methods, I propose to consider whether users themselves are under-estimated, which is the user model assumed by current works.

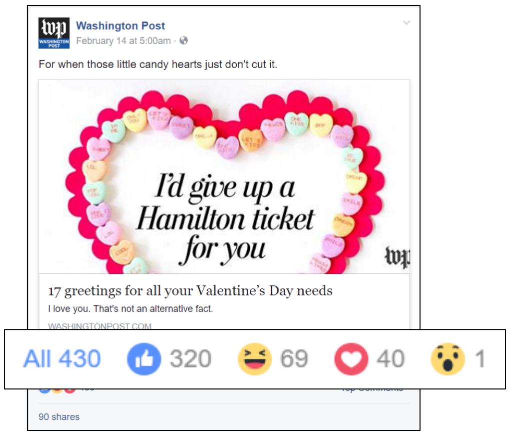
Chapter 2 | Representation of User Behavior

In this chapter, a imbalanced label ranking study is conducted to illustrate the challenge and its solution of representation of user behavior, i.e., social emotion mining.

2.1 Introduction

It has become increasingly important for businesses to better understand their users and leverage the learned knowledge to their advantage. One popular method for such a goal, so-called *social emotion mining*, is to mine users' digital footprints to unearth users' "emotions" toward particular products or services on social platforms. Users' latent emotions can be indirectly peeked via various channels–e.g., low star rating given to an Amazon review, angry comments left to a YouTube video, upvote to a news story in Reddit, or re-twitting a friend's post. In particular, we note one recently-introduced function to social platforms where users may select one emoticon, out of many choices, to more precisely express their emotions. Facebook introduced this function in 2016, while Chinese news portal, Sina, supports a similar function. Two Facebook posts are shown in Fig. 2.1 as examples. Then, a natural question is whether one can predict the emotions expressed as emoticons in such a setting.

Problem 2.1 (Social Emotion Mining (SEM))**.** *Model and Understand the correlation between online post content and cumulative emotional reactions of users.*


Washington Post
February 14 at 5:00am ·

For when those little candy hearts just don't cut it.

17 greetings for all your Valentine's Day needs

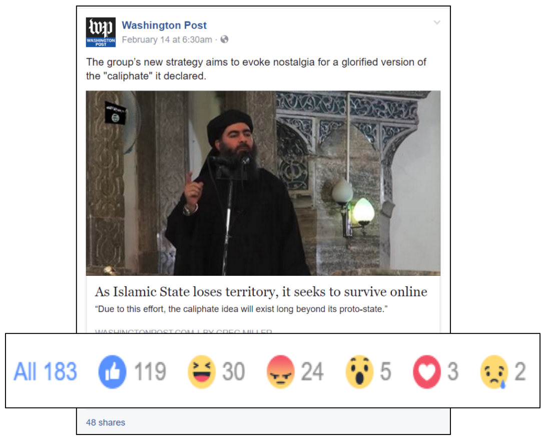
I love you. That's not an alternative fact.

WASHINGTONPOST.COM

All 430 👍 320 😆 69 ❤️ 40 😮 1

90 shares

(a)


Washington Post
February 14 at 6:30am ·

The group's new strategy aims to evoke nostalgia for a glorified version of the "caliphate" it declared.

As Islamic State loses territory, it seeks to survive online

"Due to this effort, the caliphate idea will exist long beyond its proto-state."

All 183 👍 119 😆 30 😡 24 😮 5 ❤️ 3 😢 2

48 shares

(b)

Figure 2.1: Two Washington Post Facebook posts with different emoticon reactions @ www.facebook.com/washingtonpost/

Most existing research on social emotion mining focuses on extracting informative features to infer emotions from data [2–9]. On the one hand, as taken by most present works, predicting one dominant emoticon as a *classification* problem may fail to catch the nuance of human emotions. For example, two posts in Fig. 2.1 share the same top-2 dominating emoticons, *like* and *haha*, rendering such a classification approach be less useful. On the other hand, the subjectivity makes predicting the exact composition of emotions as a *regression* problem to be less useful too. For instance, in Fig. 2.1(a), reporting the emotion of users as 69/430 *haha*, 40/430 *love* and 1/430 *wow* conveys little extra information than simply saying that users feel more *haha* than *love* and few *wow*. Therefore, to reflect the nuance and subjectivity of human emotions, we propose to formulate social emotion mining as a **label ranking** problem, where the emotions of users toward a given post are represented by a ranking among a set of emotion labels. In this way, for nuance, the number of all possible emotions is $d!$ compared with d in a classification framework, where d is the size of emotional label set; for subjectivity, only relative rather than absolute strength of different emotional labels is mined.

The label ranking problem asks *if one can learn a model to annotate an instance with a ranking over a finite set of predefined labels.* Label ranking can be seen as a specific type of the preference learning problem [10] in AI. However, in the case of social emotion mining, some labels may be preferred, causing a skewed distribution of chosen labels. For example, ordinary Facebook users (i.e., posters) tend to post more happy stories and their friends (i.e., readers) are more willing to give positive feedback such as *like* or *haha*. Therefore, the ranking distribution is highly biased toward those rankings with positive labels ranked higher than negative ones. However, posts with dominating negative labels are usually more informative. None of existing label ranking methods has considered this “imbalance" issue.

Although there have been methods to address the imbalance issue in classification, as will be illustrated in next section, imbalance in label ranking is still anything but trivial due to its large and nontrivial target space (i.e., $d!$ correlated possible rankings). To the best of our knowledge, we are the first to point out and give a formal definition of imbalance in label ranking and the first to formulate social emotion mining as a “robust" label ranking problem. Toward this challenge, we make two contributions: (1) we first show the inadequacy of popular performance measures in label ranking to handle the imbalanced data, propose a novel robust performance measure, named as G-mean-rank (GMR), and experimentally demonstrate the superiority of GMR over existing measures; and (2) we propose a novel robust label ranking model, ROAR, for imbalanced data without any re-sampling or costs as hyper-parameters, and show that ROAR outperforms 6 competing models, in real-life Facebook emoticon prediction task and achieves competitive performance in semi-synthetic benchmark label ranking data sets.

2.2 Related Works

There are three classes of label ranking methods. First, label-wise methods [11–13] treat label ranking as the regression problem for the relevant score of each label or position of ranking. Second, pair-wise methods [10, 14–17] decompose label ranking problem to binary classification problem for each pair of labels and then aggregating pairwise results into rankings. Third, list-wise methods employ different ranking distance measures to directly predict rankings without decomposing, such as Mallows model [18] based methods [19, 20], Plackett-Luce model based method [21] and weighted distance model [22] based methods [23, 24]. Our proposed solution, ROAR, belongs to the third class.

Previous label ranking works [10–15, 19, 25] typically evaluate preformance using ranking distance measure such as *Kendall tau correlation* [26] or Spearman’s rank correlation [27]. On the other hand, social emotion mining works [3,4,7,9] typically measure performance using metrics from information retrieval community, such as $ACC@k$ and $nDCG@k$ [28], emphasizing the intuition that higher ranked positions are more informative. Similarly, some rank modeling works in statistics [22–24] weight the distance between ranks to model such bias. Note that the bias there is rewarding heterogeneity of different ranking positions, rather than bias in ranking distribution considered in this work. Hence, in imbalanced data, those performance measures are not good enough.

Imbalanced data problem has been previously investigated under the classification framework [29]. Popular methods include random sampling [30,31] and cost-sensitive methods [32–34]. Both methods try to first obtain balanced data from original imbalanced data so that the problem is reduced to the balanced classification. However, these methods involve tricky hyper-parameter tuning, especially in multi-class classification [35], which will become even more severe in label ranking framework. Besides, there is nontrivial correlation among rankings rather than independent labels in classification. Hence it is hard to determine a sampling parameter or a cost for each ranking. In contrast, the robust label ranking method proposed in this work is free of hyper-parameters related to data imbalance.

2.3 Preliminaries

2.3.1 Social Emotion Mining

Here we formulate social emotion mining as the label ranking problem. Given a post x in social media, with $x \in \mathcal{X}$ as feature vector, and a set of emotional labels $\mathcal{Y} = \{y_1, y_2, ..., y_d\}$, called emoticons, the goal is to associate the post with an aggregated emotion of crowd $\phi(\mathbf{x})$ it triggers, represented by the emoticons. As argued, we choose $\phi(\mathbf{x})$ to be a ranking over the emoticon set, $\phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), ..., \phi_d(\mathbf{x}))$, where $\phi_i(\mathbf{x}) \in \mathcal{Y}$ and $\phi_i(\mathbf{x}) \neq \phi_j(\mathbf{x})$, $\forall i \neq j$. $\phi_i = y_l$ indicates that label y_l ranks on position i. For consistent annotation, a ranking position vector is defined as $\pi(\mathbf{x}) = (\pi_{y_1}(\mathbf{x}), \pi_{y_2}(\mathbf{x}), ..., \pi_{y_d}(\mathbf{x}))$, where $y_i \in \mathcal{Y}$ and $\pi_{y_i} \in \{1, 2, ..., d\}$, which means that label y_i ranks on position $\pi_{y_i}(\mathbf{x})$. With a ranking, the represented emotion consists of more of emoticons ranking higher and less of those lower. Therefore, the social emotion mining is formulated as a label ranking problem.

Problem 2.2 (Label Ranking)**.** *Find a mapping $f\colon \mathcal{X} \to \Omega_d$, where Ω_d is the set of all possible rankings over a label set of size d, such that given an instance with feature vector $\mathbf{x}$, predict ranking $\hat{\phi}(\mathbf{x}) = f(\mathbf{x})$.*

2.3.2 Imbalance in Label Ranking

In social emotion mining context, imbalance in data refers to the characteristics of data where documents with some emotional reactions are rarer than those with others. In the context of label ranking, it means that instances with some rankings are rarer than those with others. As for a formal definition of this intuition, a naive choice is treating different rankings as different classes and the problem reduces to a classification problem. However, classification framework ignores the fact that different rankings are not independent or equal-interval with each other. Instead, therefore, here imbalance is defined based on pairwise comparisons.

Given any pair of labels $\{y_i, y_j\}$, $y_i, y_j \in \mathcal{Y}$, and an instance ν, a pairwise comparison function is defined as:

$$I_\nu(y_i, y_j) = \begin{cases} 1, & \text{if } \pi_{y_i} < \pi_{y_j} \text{ for instance } \nu \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

Then, for each label pair y_i, y_j, imbalance in data distribution $\mathcal{D} = \{(x, \phi(x))\} \subset \mathcal{X} \times \Omega_d$ can be seen as the difference between $\sum_{\nu \in \mathcal{D}} I_\nu(y_i, y_j)$ and $\sum_{\nu \in \mathcal{D}} I_\nu(y_j, y_i)$. Since a ranking consists of pairwise comparisons of all pairs, a single-value imbalance measure for label ranking, *IMBA-rank* (or *IMBA* without ambiguity), of $\mathcal{D}$ is defined as:

$$IMBA(\mathcal{D}) = \frac{1}{2} \sum_{i,j=1, i \neq j}^{d} \left| log(\frac{\sum_{\nu \in \mathcal{D}} I_\nu(y_i, y_j) + 1}{\sum_{\nu \in \mathcal{D}} I_\nu(y_j, y_i) + 1}) \right| . \tag{2.2}$$

When data is perfectly balanced, *IMBA-rank* should be 0. The more imbalanced the data is, the larger *IMBA-rank* gets.

2.4 Robust Performance Measure

We first show that commonly used performance measures in label ranking are no longer adequate in imbalance case, and then introduce a robust one.

2.4.1 Previous Measures for Label Ranking

One of the most popular performance measures in label ranking community is *Kendall's tau correlation* [26]. The correlation tau for two rankings $\{\pi, \hat{\pi}\}$ is formally defined as:

$$tau = \frac{C(\pi, \hat{\pi}) - D(\pi, \hat{\pi})}{C(\pi, \hat{\pi}) + D(\pi, \hat{\pi})}, \tag{2.3}$$

where $D(\pi, \hat{\pi}) = |\{(i,j)|i < j, \pi_{y_i} > \pi_{y_j} \wedge \hat{\pi}_{y_i} < \hat{\pi}_{y_j}\}|$ and $C(\pi, \hat{\pi}) = |\{(i,j)|i < j, \pi_{y_i} > \pi_{y_j} \wedge \hat{\pi}_{y_i} > \hat{\pi}_{y_j}\}|$ denote the number of discordant and consistent-ordered pairs of labels between two rankings, respectively. To emphasize the importance of higher positions in ranking, previous works on social emotion mining usually use $ACC@k$ as performance measure. The $ACC@k$ of an instance is defined as:

$$ACC@k(\phi, \hat{\phi}) = I(\phi_i = \hat{\phi}_i | \forall i \in \{1, 2, ..., k\}), \tag{2.4}$$

where $I()$ is the indicator function.

Concerning one pair of labels and two candidate ranking positions, the imbalanced label ranking problem reduces to imbalanced classification problem. Both tau and $ACC@k$ consider only true fractions without distinguishing true positives and true negatives, which has been well known to be inadequate in imbalanced classification [29]. This is similarly true for other label ranking performance measures, such as Spearman's rank correlation and $nDCG@k$.

For a better illustration, consider a toy data set as an example. Here the label set $\mathcal{Y} = \{y_i | i \in \{1, 2, 3, 4\}\}$ with $d = 4$. The dataset contains 100 instances where 90 of them are associated with rank $\phi^{\mathbf{9}} = (y_1, y_2, y_3, y_4)$ while the rest are associated with $\phi^{\mathbf{1}} = (y_1, y_4, y_2, y_3)$. Then, a trivial label ranking model predicts all instances to be with rank $\hat{\phi} = \phi^{\mathbf{9}}$ for this toy data set. Then, the performance of the trivial model is $tau \approx 93\%$ and $ACC@k = 90\%$ if $k = 2$, which is relatively high compared with a perfect model with $tau = 100\%$ and $ACC@k = 100\%$. Hence, both measures help little in recognizing such trivial solution in imbalanced data or giving sufficient attention to minority rankings.

2.4.2 Robust Measure for Label Ranking: GMR

As shown above, it is critical to distinguish negative and positive classes for performance measure in imbalanced classification problem. Similarly in label ranking problem, we first decompose it into pairwise comparison classification problem. For each ordered pair of labels (y_i, y_j), the class of instance ν is defined as Positive if $I_\nu(y_i, y_j) = 1$ or Negative if $I_\nu(y_j, y_i) = 1$. Since the Negative class of ordered pair (y_i, y_j) is the same as Positive class of (y_j, y_i), only positive class for each ordered pair is considered. Hence only recall can be defined. Similar to classification, the recall for ordered pair (y_i, y_j) in data set $\mathcal{D}$ is defined as

$$recall_{\mathcal{D}}(y_i, y_j|f) = \frac{\sum_{\nu \in \mathcal{D}} I_\nu(y_i, y_j) I_{\hat{\nu}}(y_i, y_j) + 1}{\sum_{\nu \in \mathcal{D}} I_\nu(y_i, y_j) + 2}, \tag{2.5}$$

where $I_{\hat{\nu}}(y_i, y_j)$ is the pairwise comparison function for predicted ranking for instance ν, and the extra $+1$ term in numerator and $+2$ term in denominator are smooth terms. To combine recalls of different pairs, inspired by G-Mean [35] for imbalanced multi-class classification, geometric mean is used and *G-mean-rank* (GMR) for data set $\mathcal{D}$ is defined as:

$$GMR_{\mathcal{D}}(f) = \sqrt[P]{\prod_{i \neq j}^{d} recall_{\mathcal{D}}(y_i, y_j|f)}\,, \tag{2.6}$$

where P is the number of ordered pairs involved. For each pair of labels, GMR is the same as *G-mean* for two-class classification. Hence according to the definition of imbalance in label ranking, GMR is insensitive to imbalanced label ranking data.

Using the aforementioned toy dataset again, the performance of the trivial algorithm in terms of GMR is $GMR \approx 53\%$, which is not high compared to 97% for a perfect one. Therefore, GMR rightfully gives sufficient penalty to a trivial solution which is not supposed to perform well, a behavior that we intended.

2.4.3 Is GMR Superior to Previous Measures?

To show the robustness of GMR compared to two popular measures–i.e., *tau* and *ACC@k*, we apply the idea of toy example above to real datasets. We use four

		Datasets			
Measures	Methods	ROU	NYT	WSJ	WaPo
IMBA		3.03	1.94	2.96	2.14
ACC@3	RPC	0.850	0.185	0.191	0.182
	KNN-PL	0.770	0.125	0.169	0.144
	NAIVE	0.837	0.0532	0.171	0.0976
	Gain (%)	-3	191	5	67
tau	RPC	0.933	0.497	0.595	0.541
	KNN-PL	0.929	0.495	0.584	0.544
	NAIVE	0.932	0.399	0.567	0.503
	Gain (%)	-0.1	24	4	8
GMR	RPC	0.241	0.429	0.345	0.358
	KNN-PL	0.408	0.458	0.513	0.461
	NAIVE	0.152	0.0796	0.0881	0.0770
	Gain (%)	**113**	**457**	**387**	**432**

Table 2.1: Comparison of GMR with ACC@3 and tau using real datasets

Facebook post datasets with emoticon set size of $d = 6$, whose detail can be found in Empirical Validation section later.

We extend the idea of the trivial model in the toy example by designing a naive model, denoted as **NAIVE**, that assigns the most common ranking in a training set to all instances in a test set regardless of their feature values. As the datasets we use are rankings converted from number of votes for different emoticons, the most common ranking (i.e., the output) in NAIVE is set as the ranking of emoticons according to the number of accumulated votes in the training set. Therefore, NAIVE is a "dumb" solution and is not supposed to work well.

Next, we choose 2 state-of-the-art models, **RPC** and **KNN-PL**, as examples of good models, whose detail will be explained in Robust Label Ranking Model section. The idea is that a robust performance measure should be able to clearly distinguish good models (e.g., RPC and KNN-PL) from bad ones (e.g., NAIVE) even when a dataset is severely unbalanced.

The result is shown in Table 2.1. For *ACC*@*k*, *k* is set as 3 to mimic the behavior of Facebook, where only top-3 emoticons of posts are shown by default. The row, Gain (%), in Table 2.1 shows the *average* improvement of two good models over NAIVE in terms of three different measures. In all four datasets, the improvement in terms of GMR is always far larger than ACC@3 and tau, which illustrates the robustness of GMR. Table 2.1 also shows the $IMBA$-$rank$ of each dataset as $IMBA$. Note that the improvement in terms of ACC@3 and tau decreases as $IMBA$-$rank$ increases. For the most imbalanced dataset, ROU, the improvement in terms of both ACC@3 and tau is even negative, which indicates that GMR is capable of capturing the fact that two state-of-the-art models far outperform a naive poorly-designed model.

Now we are ready to formally define robust label ranking problem.

Problem 2.3 (Robust Label Ranking)**.** *Find a mapping $f\colon \mathcal{X} \to \Omega_d$, for data distribution $\mathcal{D}$, with large $IMBA$-$rank(\mathcal{D})$, such that $GMR_{\mathcal{D}}(f) \geq GMR_{\mathcal{D}}(f')$, $\forall f' : \mathcal{X} \to \Omega_d$.*

2.5 Robust Label Ranking Model

2.5.1 Competing models

In this work, to our best knowledge, we consider all existing state-of-the-art label ranking models as follows.

- Ranking by Pairwise Comparison (RPC) [10]: It predicts pairwise order for each pair of labels using logistic regression and then combines them into ranking output with Borda count [36].
- Label-Wise Decomposition (LWD) [13]: It predicts position probability distribution of each label and then combines them to minimize expected Spearman's footrule [37].
- Soft Multi-Prototype (SMP) [17]: It fits label ranking data with multiple prototypes both in feature and ranking space, and combines prototypes into ranking prediction given feature values.
- K-Nearest-Neighbor with Plackett-Luce model (KNN-PL) [21]: It predicts ranking by aggregating rankings of instances whose feature values are nearest to given feature value. The aggregation is based on Plackett-Luce model.
- K-Nearest-Neighbor with Mallows model (KNN-M) [19]: It is the same with KNN-PL except the aggregation is based on Mallows model [18].

- Log-Linear model (LogLinear) [12]: It learns utility functions for each label via pairwise comparison and sorts labels by utility function values into ranking. Here the utility function is adopted from [10], in which case LogLinear is equivalent to the Constraint Classification algorithm [11].
- Label Ranking Tree (LRT) [19]: It is a decision tree method whose induction is based on Mallows model [18].

2.5.2 Robust Label Ranking Model: ROAR

Now, we propose a robust label ranking model, named as **ROAR** (RObust lAbel Ranking), which is a simple, efficient, and effective tree based model. The performance measure GMR is difficult to be directly optimized, as it is not an average over some performance measure for each instance. Hence an alternative learning objective function, an induction criterion in decision tree, is proposed. This supports the model searching for finest structure in feature and target space without overfitting, which makes it robust against imbalanced data.

Learning. To learn a decision tree, a general algorithm begins with all instances in the root node. Then, it partitions the training data recursively, by one-dimension splits according to the comparison between thresholds and a feature value. The decision tree in this work is a binary tree.

The threshold and the feature for each split are selected by exhaustive search so that the sizes of the neighborhoods in the target space, estimated by training data in the resultant child nodes, become the smallest. The size of a neighborhood is estimated by the impurity of the set of rankings in a node. One intuition about the impurity of a set of rankings is the impurity of labels on each ranking position. Because labels are independent of each other, for a given position, we choose the popular Gini index, and the Gini index of a tree node T for position i is defined as:

$$Gini_i(T) = \sum_{y \in \mathcal{Y}} \frac{(n_i(T) - n_{iy}(T))n_{iy}(T)}{n_i(T)^2}\,, \tag{2.7}$$

where $n_{iy}(T) = \sum_{\nu \in T} I(\phi_i(\mathbf{x}_\nu) = y)$ is the number of instances with label y ranking on position i and $n_i(T) = \sum_{y \in \mathcal{Y}} n_{iy}(T)$ denotes the number of instances with any label ranking on position i. Then the impurity for rankings of the node T can be measured by weighted sum of Gini index for each position, that is,

$$Gini(T) = \sum_{i=1}^{d} \frac{n_i(T)Gini_i(T)}{|T|}, \tag{2.8}$$

which is called point-wise Gini index. For parent node T and potential child nodes T^- and T^+, the split criterion is defined as

$$criterion = |T|^{-1}(|T^+|Gini(T^+) + |T^-|Gini(T^-)). \tag{2.9}$$

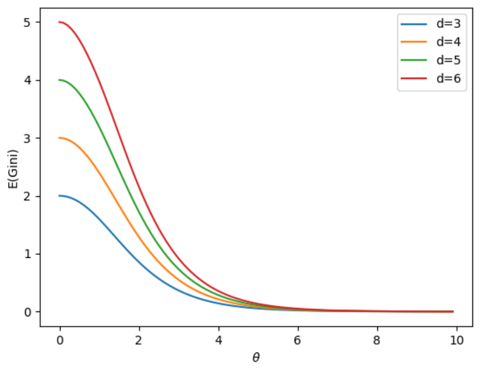
The stopping criterion is straightforward. The partitioning stops when no further partitioning is possible, that is, when there is no partitioning whose *criterion* is smaller than $Gini(T)$ for current node T.

Prediction. Here for ROAR, we use a position-wise ranking aggregation method. From highest to lowest ranking position, given a position, it assigns the label that has not been assigned and appears most frequently at that position, to each position. When there is no such label for a position, it resorts to label distributions of other positions, from highest to lowest and does the same.

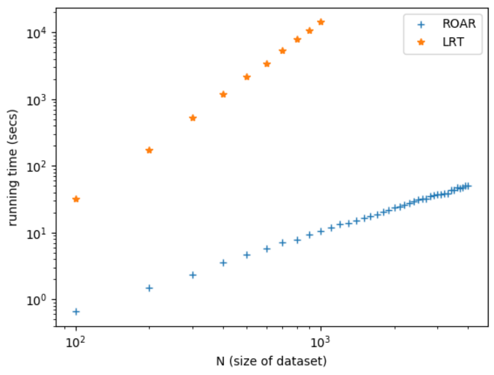
Consistency with Ranking Theory. This point-wise Gini index is consistent with our intuition about the purity of a set of rankings. To show that, we have to assume a measurement of the size of a neighborhood $\Omega(T)$ around a point in Ω_d, noting that the center ranking π_0 is unknown. Mallows model [18] is a popular assumption of probability model of rankings, using the annotation from [19], defined as

$$P(\pi|\theta, \pi^0) = \frac{exp(-\theta D(\pi, \pi^0))}{\psi(\theta)}, \tag{2.10}$$

where $\psi(\theta) = \sum_{\pi \in \Omega_d} exp(-\theta D(\pi, \pi^0))$ is a normalization constant, θ the spread parameter, π^0 the center ranking and $D(\cdot, \cdot)$ the distance between two rankings, which is the number of discordant pairs between two rankings. Assuming that the rankings in T are independently generated according to Mallows model, the spreading parameter θ measures the size of $\Omega(T)$. Under the independence assumption,



(a) E(Gini) versus θ



(b) Running time

Figure 2.2

the expectation of point-wise Gini index for node T is

$$\begin{aligned} E(Gini(T)) &= E(\sum_i^d \frac{n_i(T)Gini_i(T)}{|T|}) \\ &= \sum_i^d E(Gini_i(T)), \end{aligned} \tag{2.11}$$

$$E(Gini_i) = \frac{(n_i - 1)}{n_i}(1 - \sum_{y \in \mathcal{Y}} P(\phi_i = y)^2). \tag{2.12}$$

According to Mallows model, without loss of generality, we assume the center ranking $\phi_i^0 = y_i$, $\forall i \in \{1, 2, ..., d\}$. Then ranking ϕ with $\phi_i = y_j$, is with probability $P(\phi) = \frac{\mathcal{O}(exp(-|i-j|\theta))}{\psi(\theta)}$. Therefore, for large enough θ, $P(\phi_i = y_j) = \mathcal{O}(exp(-|i-j|\theta))$. Hence, when θ is larger, which is when the spread of Mallows model is smaller, then the probabilities $P(\phi_i = y_j)$ over $y_j \in \mathcal{Y}$ are more skewed toward smaller $|i - j|$. Therefore according to eq. 2.12, $E(Gini_i)$ is smaller, so is $E(Gini(T))$, as illustrated in Fig. 2.2a with different sizes of label set in the limitation of $n_i \to \infty$. Therefore, *Gini* is a good estimator of the impurity of rankings in a node.

Time Complexity. In ROAR, the amortized running time for each potential partition is $\theta(d^2)$, constant in terms of $|T|$. Hence the running time for each induction of a tree node is $\Theta(m|T|(log|T| + d^2))$. In contrast, LRT takes $\Omega(|T|)$ steps for each potential partition, so that the running time for each induction of a tree node is $\Omega(m|T|^2d^2)$. The running time of two methods applied to data of different sizes are shown in Fig. 2.2b. As LRT becomes prohibitively slow as data gets large, it is not considered in following empirical validation.

2.6 Empirical Validation

We attempt to validate if: (1) our proposed G-mean-rank is superior to two popular label ranking measures in imbalance datasets, which has been done in Robust Performance Measure section; and (2) our proposed ROAR outperforms 6 competing label ranking models. The data sets and codes will be publicly available.

2.6.1 Datasets and Set-Up

In this work, we use emoticon clicks data of Facebook posts. For each post, there are six emoticon labels, {like, love, haha, wow, sad, angry}. Each user (i.e., reader) can select one of the six labels for each post. For evaluating NAIVE in Robust Performance Measure section, we use the number of votes for labels per post as the input. To obtain ranking, for each post, the labels are sorted according to their number of votes. If the number is zero for some labels, they are considered ranked at an extra tail position attached to the normal ranking without preference to each other and the ranking positions without labels are treated as missing. There are four data sets: (1) public posts from random ordinary users, denoted as ROU (Random Ordinary Users); (2) New York Times (NYT)[1] posts; (3) the Wall Street Journal (WSJ)[2] posts; and (4) the Washington Post (WaPo)[3] posts. We have crawled all four sets of posts in 2016 after Facebook introduced six emoticons.

As our focus is on the evaluation of our two proposals for the robust label ranking problem (instead of finding effective features), we avoid sophisticated features (e.g., user related or network structure based), and instead use fundamental textual features, extracted via AlchemyLanguage API (by IBM Watson Lab). For posts in ROU, only posts with text are included, and the document emotion of the text given by AlchemyLanguage is used as features. For posts in other three sets, if there is a link to external original full news, the document emotion of the full news is used as feature, and otherwise, only the text in posts is used. The returned document

[1]www.facebook.com/nytimes/
[2]www.facebook.com/wsj/
[3]www.facebook.com/washingtonpost/

	ROU	NYT	WSJ	WaPo
# posts	$17,394$	$4,684$	$7,464$	$6,117$
IMBA	3.03	1.94	2.96	2.14
# *like*	$834K$	$7.99M$	$2.44M$	$3.81M$
# *love*	$14K$	$578K$	$105K$	$222K$
# *haha*	$3,281$	$434K$	$130K$	$248K$
# *wow*	$2,610$	$328K$	$84K$	$179K$
# *sad*	$2,430$	$786K$	$70K$	$332K$
# *angry*	678	$1,07M$	$93K$	$549K$

Table 2.2: Summary of four datasets

emotion from AlchemyLanguage consists of $[0,1]$ scores, for five emotion dimensions, “anger", “joy", “fear", “sadness" and “disgust". The scores measure the amplitude of each emotion conveyed by the text. Then the four data sets are with the same feature and target format, that is, $\mathcal{Y} = \{like, love, haha, wow, sad, angry\}$ with $d = 6$ and $m = 5$ dimensional feature space.

The details of four data sets are shown in Table 2.2. Comparing ROU and the other three sets, the *IMBA-rank* of ROU is much higher. This is due to the fact that readers of the posts from ordinary users are usually their friends, who tend to give positive feedback, $\{like, love, haha\}$ rather than negative one, $\{sad, angry\}$ All our datasets are significantly imbalanced in that *like* or other positive labels are more frequent. This is partially due to the interface limitation such that users have to hover their mouse over the *Like* button to be able to select other emoticons. To illustrate imbalance in label ranking more clearly, we show the pairwise comparison matrix of WaPo, as Table 2.3, where the number in each entry (y_i, y_j) counts the number of posts (support) with y_i being higher ranked than y_j. For instance, there are only 209 posts where *angry* is ranked higher than *like* compared with $5,906$ in contrast.

We also use 16 semi-synthetic data sets obtained by converting benchmark multi-class classification using Naive Bayes and regression data using feature-to-label technique from the UCI and Statlog repositories into label ranking [19]. These data sets are widely used as benchmark in label ranking works.

	like	*love*	*haha*	*wow*	*sad*	*angry*
like	–	6,117	6,093	6,115	5,982	5,906
love	0	–	2,994	2,654	2,978	3,116
haha	23	2,003	–	1,872	2,415	2,295
wow	2	2,623	3,036	–	3,093	2,968
sad	130	2,203	2,104	1,717	–	1,880
angry	209	2,014	1,979	1,821	2,040	–

Table 2.3: Pairwise comparison matrix of WaPo

		Datasets			
	Methods	ROU	NYT	WSJ	WaPo
tau	RPC	0.933	0.497	0.595	0.541
	LWR	0.937	0.499	0.603	**0.562**
	SMP	0.933	0.495	0.601	0.547
	KNN-PL	0.929	0.495	0.584	0.544
	KNN-M	0.928	0.504	0.595	0.550
	LogLinear	0.935	0.488	0.593	0.537
	ROAR	**0.954****	**0.634****	**0.612***	0.554
GMR	RPC	0.241	0.429	0.345	0.358
	LWR	0.295	0.433	0.289	0.390
	SMP	0.246	0.351	0.257	0.247
	KNN-PL	**0.408***	0.458	0.513	0.461
	KNN-M	0.387	0.455	0.468	0.435
	LogLinear	0.209	0.287	0.253	0.203
	ROAR	0.343	**0.680****	**0.534***	**0.478***

Table 2.4: Summary of results on Facebook posts datasets (* means significance level of 0.1, and ** 0.01)

2.6.2 Results

2.6.3 Competing Models

First we test models on Facebook posts data sets, which are imbalanced. All results are obtained with 5-fold cross validation. We compare ROAR with 6 existing

	tau							**GMR**						
Data	RPC	LWR	SMP	KNN-PL	KNN-M	LogLinear	**ROAR**	RPC	LWR	SMP	KNN-PL	KNN-M	LogLinear	**ROAR**
glass	0.877	**0.884**	0.476	0.809	0.807	0.812	0.851	0.785	**0.795**	0.471	0.582	0.625	0.625	0.772
authorship	0.919	0.912	0.636	**0.931**	0.930	0.497	0.873	**0.912**	0.910	0.571	0.903	0.903	0.808	0.870
pendigits	0.928	0.930	0.488	0.934	0.933	0.539	**0.936**	0.945	0.948	0.523	0.949	0.951	0.828	**0.954**
elevators	0.725	**0.789****	0.753	0.724	0.726	0.556	0.715	0.732	0.769	0.618	0.751	0.739	**0.793****	0.768
segment	0.927	0.950	0.123	0.940	0.940	0.733	**0.956***	0.948	0.963	0.220	0.955	0.958	0.885	**0.969****
wine	0.914	0.910	0.944	0.936	**0.948**	**0.948**	0.892	0.901	0.898	0.908	0.909	0.917	**0.917**	0.887
vowel	0.623	0.750	0.503	0.746	0.749	0.558	**0.796****	0.768	0.844	0.517	0.812	0.834	0.766	**0.870****
cpu	0.445	0.462	-0.010	0.496	**0.497**	0.358	0.370	0.534	0.672	0.035	0.672	**0.693****	0.675	0.656
vehicle	0.844	**0.860**	0.817	0.843	0.835	0.756	0.833	0.906	**0.908**	0.870	0.895	0.896	0.839	0.894
housing	0.667	0.685	0.469	0.647	0.661	0.606	**0.775****	0.806	0.817	0.632	0.798	0.806	0.784	**0.866****
iris	0.884	**0.982***	0.760	0.956	0.960	0.804	0.929	0.890	**0.927****	0.823	0.918	0.919	0.847	0.903
stock	0.750	0.850	0.697	**0.900**	0.899	0.643	0.888	0.859	0.910	0.820	**0.936**	0.936	0.806	0.932
calhousing	0.243	0.243	0.256	0.325	0.334	0.190	**0.338**	0.576	0.583	0.558	0.626	0.640	0.589	**0.663****
wisconsin	**0.626****	0.510	0.056	0.477	0.486	0.563	0.311	**0.777****	0.721	0.289	0.697	0.702	0.747	0.628
bodyfat	**0.292**	0.282	0.139	0.227	0.231	0.272	0.074	**0.623**	0.618	0.509	0.593	0.591	0.617	0.525
fried	**1.000**	0.990	0.470	0.902	0.905	0.994	0.879	**1.000**	0.995	0.697	0.951	0.952	0.997	0.939
# of common wins in both measures in ascending order: SMP=0, KNN-PL=1, KNN-M=1, LogLinear=1, RPC=3, LWR=3, and **ROAR=5**														

Table 2.5: Summary of results on semi-synthetic data sets

state-of-the-art label ranking models, RPC[4], LWD, SMP[5], KNN-PL[6], KNN-M and LogLinear.

For evaluation, as we have shown the superiority of GMR in imbalanced data, here GMR is used. For consistency with previous label ranking works, results in terms of tau are also included in Tabel. 2.4.

Table 2.4 shows that ROAR achieves significantly better performance in all four data sets except ROU in terms of GMR. In ROU dataset, ROAR loses only to two KNN based methods. As pointed out previously, that posts emotion extracted from posts in ROU may not be meaningful enough, hence lack of structural correlation between feature and target favors instance-based learning method such as KNN. Hence the experiment shows that ROAR outperforms other models in real-world imbalanced label ranking data.

Next, ROAR and other label ranking models are applied to benchmark semi-synthetic data sets, evaluated by tau and GMR. As shown in Table. 2.5, ROAR achieves competitive results against other models and wins in the most data sets in terms of both tau and GMR.

2.6.4 Case study

What does it mean that an model performs better in terms of GMR in imbalanced label ranking data sets? Here we use WaPo data set result to answer it. Because imbalance measure of label ranking data $IMBA-rank$ is defined based on imbalance between two orders of each pair of labels (eq. 2.2), here we want to know whether an model can recall those minority orders in imbalanced pairs. In WaPo (Table. 2.3), we choose $(haha, like)$, $(sad, like)$ and $(angry, like)$ three minority pair orders, where (y_i, y_j) means y_i ranks higher than y_j. The number of posts with each of these orders is 23, 130 and 209, respectively, compared with that of those opposite, 6 093, 5 982 and 5 906. The $recall$ of each pair order is shown in Fig. 2.3. It is obvious that ROAR is superior than any other models. Actually any models

[4]There is only one minor modification. In case of missing labels, missing label pairs indicate that they are not preferred to each other, as abstention means. Therefore, missing label pairs are assigned a small weight α, and counted as one preference relation for each order. The weight is picked empirically, $\alpha = 1/64$ used in this work, and results do not appear sensitive to the weight for a range of α values.

[5]There is a hyperparameter k [17], which is set to default value 100 for Facebook data, and slightly smaller than number of all possible rankings for each semi-synthetic data.

[6]Two KNN based model takes default $K = 20$.

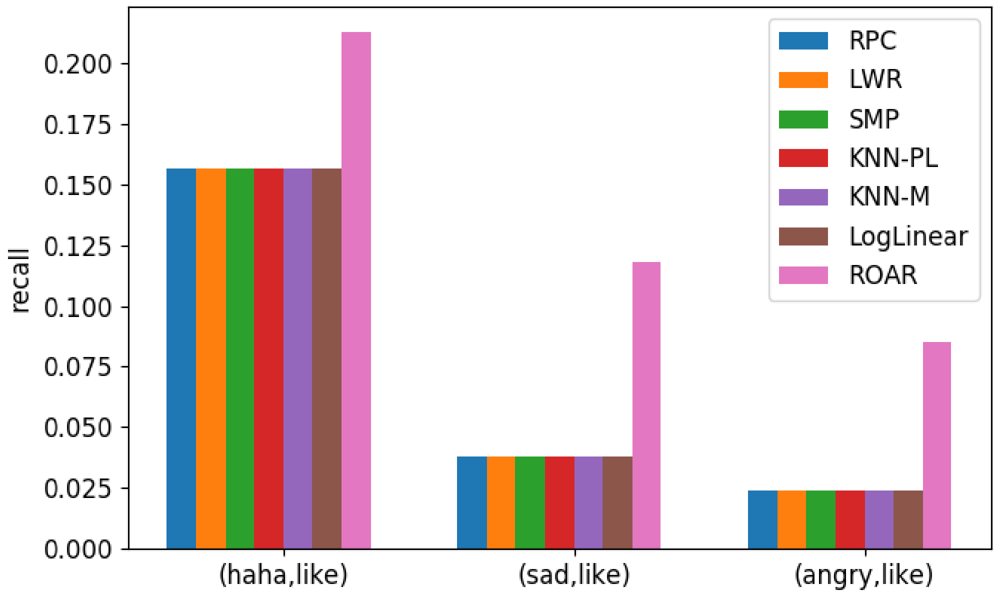


Figure 2.3: *recall* of minority pair orders. Actually any models except ROAR do not recall any posts with those minority pair orders, which is why they get same *recall* (not vanishing due to the smooth term in *recall* definition).

except ROAR do not recall any posts with those minority pair orders, which is why they get same *recall*. Hence ROAR works better in recalling minority pair orders, as it achieves highest GMR in WaPo (Table. 2.4). However, this advantage is not well appreciated by *tau* as shown in Table. 2.4.

2.7 Conclusion

In this work, we formally define robust label ranking problem for social emotion mining. To overcome the challenges, we first propose a robust measure, GMR, as the criterion for the problem. Both synthetic and experimental analysis show the superiority of GMR over popular measures such as *Kendall's tau correlation* and *ACC@k*. Then, we also propose a robust model, ROAR, and empirically validate its superiority over 6 competing label ranking models in Facebook posts data sets and benchmark semi-synthetic data sets.

Chapter 3 | What Influences User Behavior?

In this chapter, a multi-task learning study is conducted aiming at finding the influence of news channels on their readers' reactions besides the content they post. On the other direction, an observation from this specific problem inspires a new framework design of multi-task learning.

3.1 Introduction

The development of advanced machine learning techniques (e.g., deep learning) often requires a large amount of labeled samples to train a good model. However, this requirement is hard to meet for many applications due to the prohibitive cost of data collection and labeling. To mitigate this problem, the *Multi-Task Learning (MTL)* approach takes an advantage of multiple related tasks to facilitate the training of some or all of the tasks that have limited training samples [38]. MTL has been successfully applied to many learning problems in various domains (e.g., computer vision and natural language processing).

The principle of MTL is to leverage the relationship assumptions among tasks through a model design–e.g., commonalities across tasks. Some well-known MTL designs can be categorized as feature selection, task structure, the low rank structure of model parameters of tasks in linear models [39], and parameter sharing and information sharing in neural network models [40]. The validity of the task-relationship assumption in these models is vital to achieve successful learning. However, we observe that, such a task relationship used in previous methods does not always hold. More specifically, a task relationship can often hold only *within topics*–i.e., commonalities across tasks hold only for certain topics (or groups) of data. Consider the following two motivating examples.

Example 3.1 (Predicting User Emotions)**.** *Consider the problem to accurately predict news readers' reactions (e.g., LIKE, ThumbsDown) toward news posts from different news channels (e.g., NYT, Wapo, Fox). To overcome insufficient data per new channel, one models the problem as MTL (i.e., news channels as tasks), assuming that readers' reactions across tasks be similar. However, in practice, such an assumption on the task relationship may not hold. For instance, readers' reactions can be highly consistent across different news channels for news on the topics of natural events and festivals; however, different standing points of different channels often result in exactly opposite readers' reactions for the news on the topics of sports and political news.*

Example 3.2 (Searching Relevant Products)**.** *In e-commerce applications, consider a problem of searching products for different user groups. For instance, both male and female users may have similar taste for products related to food (i.e., topic), but different taste for books or music (i.e., topic). In this case, considering the same task relationship across all products will either miss the similarity (i.e., treating two tasks as independent) or cause negative knowledge transfer (i.e., treating two tasks the same).*

Based on these observations, therefore, we propose a "within-topic" task relationship hypothesis to reveal the data-dependent task relationship. This hypothesis assumes that task relationship may appear different within data if from different topics. The topics are determined by input features of data (clusters of data), different from task groups in within-group clustering design [41]. Compared with the recent works [42,43] on data-dependent task relationship, with the clear notion of topics, the data dependency and task relationship can be "decoupled" here, which enables the application of any existing task relationship designs to reveal within-topic task relationship. In this work, therefore, we propose a topic-wise multi-task architecture using a topic module to distribute data from different topics to different modules, so that different task relationship can be learned. Within each topic, we propose two topic-task-sparsity constraints to enforce a multi-task sparsity structure for task relationship, where only a few tasks are allowed to deviate from a global structure shared by all other tasks. This multi-task sparsity structure is consistent with the aforementioned example, where only a few news channels are different from the others per topic.

Our contributions can be summarized as follows: (1) we propose the within-topic task relationship hypothesis for the MTL problem; (2) we propose a topic-wise multi-task architecture based on the hypothesis; (3) we propose two types of topic-task sparsity constraints, topic-task-element and topic-task-exclusive and the optimization algorithms with proof; and (4) the proposed topic-wise multi-task sparsity model consistently outperforms state-of-the-art MTL models in experiments on both synthetic and real world datasets.

3.2 Related Works

In this section, we review related works on linear MTL models, MTL neural networks and sparsity constraints used in neural networks.

There are a lot of works on linear MTL models. Interested readers are referred to [39] for a comprehensive survey. Those models are designed based on different assumptions of task relationships. More specifically, [44–46] assume different tasks share similar sparse feature selection pattern. [47–52] assume that the weight vectors. With similar spirit of above task structure assumption, [41, 53, 54] directly assume that the weight matrix should be low-rank, which enforce different tasks to share the same low-dimension feature transformation. Though the simplicity of the linear structure provides such flourishing of MTL designs, it is less flexible compared with neural network models.

The neural network MTL models are based on two designs, parameter sharing and information sharing. The most common shared-bottom model is similar to the feature selection design in linear MTL models. Built upon the shared-bottom design, [55–57] propose further constraints on parameter sharing. Unique for neural network MTL models is information sharing [58], where cross-stitch structures are used to enable information flows from one task to another. Though neural network provides more flexibility of model design, as the information sharing, task relationship still relies only on design assumptions but not further information.

There are two recent works [42, 43], whose task-specific gates can be considered as data-dependent task relationship design. The distribution of weights given to different experts by different tasks are determined by the inputs. When such distributions of two tasks given a group of input samples are similar, those two tasks are related and vice versa. However, both data-dependency and task relationship are modeled by the weights of different tasks, which excludes the application of more flexible task relationship designs. Moreover, it can be seen later that MMoE [42] can be seen as a special instantiation of our proposed architecture.

Many task relationships in linear MTL models are achieved by constraints over weight matrix, especially sparsity constraints (e.g., $l_{1,q}$ penalty). Within neural network models, the sparsity constraints are recently applied to model compression [59–62]. For example, [60, 61] use group sparsity ($l_{1,q}$) loss to zero-out the entire neurons to learn a sparse model for both memory and computation efficiency. [62] combines both group sparsity ($l_{2,1}$) and exclusive penalty ($l_{1,2}$). In this work, we adopt group sparsity as topic-task-element penalty ($l_{1,1,2}$) and propose group exclusive penalty as topic-task-exclusive penalty ($l_{2,1,2}$), together with its optimization algorithm.

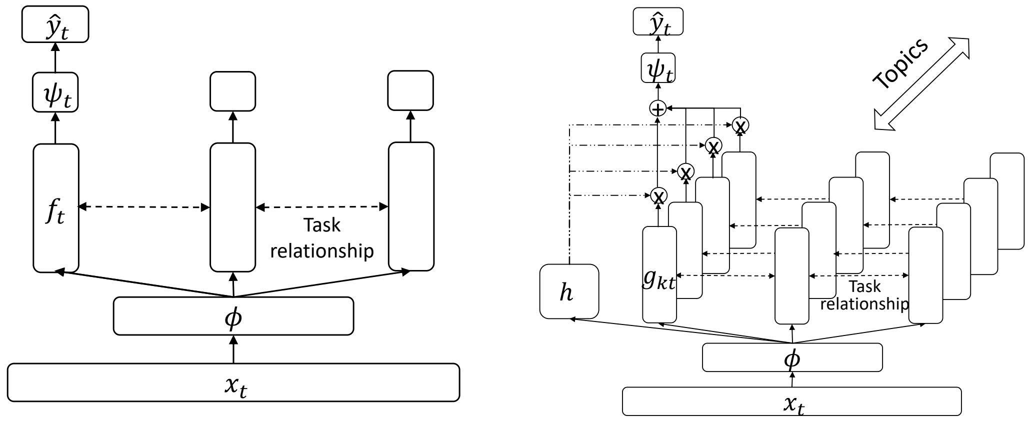
3.3 Problem Definition

We formally define the multi-task learning (MTL) problem.

Definition 3.1 (Multi-Task Learning Problem)**.** *Given T tasks, for each task $t \in [T]$, there are N_t samples (X_t, Y_t), with each $x_t \in \mathbf{R}^{d_t}$ as input feature and $y_t \in \mathbf{R}^{p_t}$ as labels. Here in this work, we take homogeneous MTL setting, where the dimensions and types of the features and labels for different tasks are the same, respectively, that is, for $\forall t \in [T]$, $d_t = d$ and $p_t = p$. Then, the MTL problem is to find a mapping $f : \mathbf{R}^d \times [T] \mapsto \mathbf{R}^p$, such that the overall cost $\mathcal{L} = \sum_{t\in[T]} \frac{1}{N_t} \sum_{n_t\in[N_t]} L(f(x_t, t), y_t)$ is minimized.*

The proposed within-topic task relationship hypothesis can be formally defined as follows:

Definition 3.2 (Within-Topic Task Relationship Hypothesis)**.** *Given each sample input x, there is a topic $h(x)$ given by $h : \mathbf{R}^d \mapsto [K]$, where K is the number of topics. The prediction function $f : \mathbf{R}^d \times [T] \mapsto \mathbf{R}^p$ can be decomposed as $f(x, t) = g(h(x), x, t)$. Within each topic $k \in [K]$, $g(k, ., .)$ shows the task relationship between each $g(k, ., t_1)$ and $g(k, ., t_2)$ with $t_1 \neq t_2$.*



(a) (b)

Figure 3.1: Neural Network MTL architectures: (a) Existing neural network MTL architectures; (b) Topic-wise multi-task architecture

3.4 Topic-Wise Multi-Task Sparsity Model

In this section, we describe the proposed topic-wise multi-task sparsity model. First, the topic-wise multi-task architecture is described as the overview of the model, which can be combined with any existing MTL design as within-topic task relationship. Second, the two sparsity constraints are introduced for within-topic task relationship. Third, the optimization algorithm is described.

3.4.1 Topic-Wise Multi-Task Architecture

The topic-wise multi-task architecture is designed based on the within-topic task relationship hypothesis. Specifically, given input x, it is cast by a set of topic-task-specific functions $\{g(k, x, t) \| k \in [K]\}$ into the topic-task-specific hidden layers, and the task-specific layer afterward is obtained by aggregating topic-task-specific layers over different topics weighted by topic distribution $h(x)$ such that $\sum_k h(x)_k = 1$, which can be formulated as

$$f(x, t) = \sum_k h(x)_k g(k, x, t). \tag{3.1}$$

When a task relationship is enforced in topic-task-specific functions $\{g(k, x, t)\}$ within each topic k, the topic-wise multi-task architecture reveals Definition. 3.2.

Compared with the existing shared-bottom architecture (Fig. 3.1), the topic module $h(x)$ distributes data samples to different within-topic task relationship, rather than all data with the same task relationship. This clearer task relationship within each topic leads to more compact structure of $g(k, x, t)$ (i.e., low-rank structure, parameter sharing), compensating the redundancy by the extra topic dimension and boosts the performance.

We compare the proposed architecture to the recent MMoE work [42], that models data-dependent task relationship. From its viewpoint, our work decouples the data-dependent task relationship into data-dependence ($h(x)$) and within-topic task relationship ($g(k, x, t)$), which enables the application of all existing task relationship designs for the latter. To see this, if we choose the factorization structure (DMTRL) [55] for within-topic task relationship, $g(k, x, t) = \sum_e p(k, t)_e q(x, e)$, Eq. 3.1 becomes $f(x, t) = \sum_k \sum_e h(x)_k p(k, t)_e q(x, e)$. Compared with Eq. 7 in [42], MMoE can be seen as a special instantiation of the proposed architecture by setting $gate(x, t)_e = \sum_k h(x)_k p(k, t)_e$.

3.4.2 Topic-Task Sparsity

In this subsection, we describe a new MTL design, called *topic-task sparsity*, to capture task relationship with the help of the proposed topic-wise multi-task architecture.

We assume that, within each topic, only a few tasks (news channels) may deviate from the majority. For example, within political topic, the readers' reactions to similar posts under extreme conservative or liberal news channels are usually different from those under the majority milder channels. We proposed the topic-task-sparsity design that

$$\theta_{k,t} = \theta^0 + \theta^s_{k,t}, \tag{3.2}$$

where $\theta_{k,t}$ is the vector of the parameters of topic-task-specific function $g(k, x, t) = g(x|\theta_{k,t})$, θ^0 is the global parameters that shared by different topics k and tasks t, and $\theta^s_{k,t}$ is the topic-task-sparse part of the parameters. We note Θ^s as the tensor combining $\theta^s_{k,t}$ for all topics and tasks.

To enforce sparsity structure in Θ^s, we proposed two types of topic-task-sparsity constraints $\Omega(\Theta^s)$. First, an element-wise sparsity structure is assumed for Θ^s, which is enforced by topic-task-element constraint defined as

$$\Omega^{el}(\Theta^s) = \sum_k \sum_t ||\theta^s_{k,t}||_2, \tag{3.3}$$

where $||.||_q$ is the l_q norm. The entire topic-task-element constraint $\Omega^{el}()$ is a $l_{2,1,1}$ norm, which is also known as group sparsity constraints. It is used in [60, 61] to zero out entire neurons for compression. Here, similar property is used to enforce certain topic-task-specific parameters $\theta_{k,t}$ to be the same as the global ones θ^0. The effect of the additional topic dimension in the above topic-task-element constraint lies in its element-wise sparsity. Without topics, it is reduced to task-wise sparsity, often a too-strong assumption for task relationship.

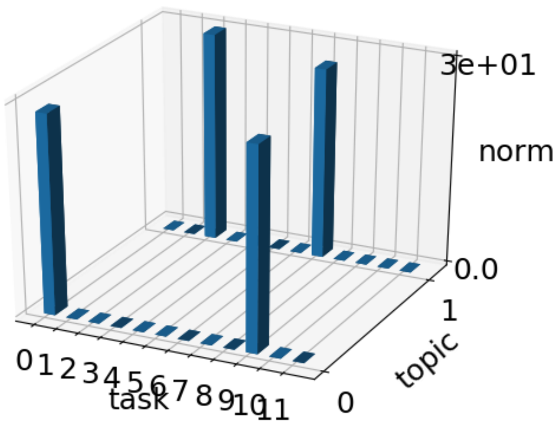
Next, we consider another topic-task-sparsity constraint that more explicitly takes advantage of the topic dimension. It is called topic-task-exclusive constraint, defined as

$$\Omega^{ex}(\Theta^s) = \frac{1}{2} \sum_k (\sum_t ||\theta^s_{k,t}||_2)^2. \tag{3.4}$$

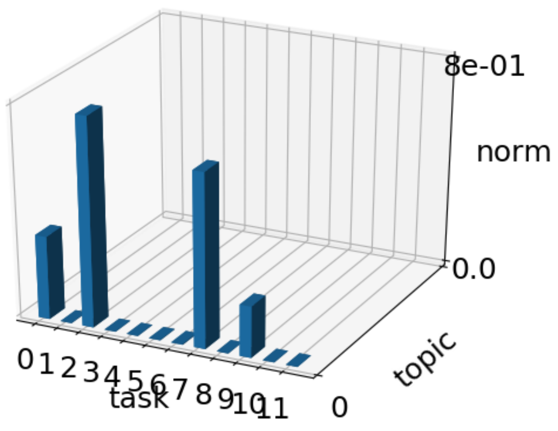
The entire topic-task-exclusive constraint Ω^{ex} is the square of a $l_{2,1,2}$ norm. The l_1 norm for the task dimension still enforces the entire $\theta^s_{k,t}$ parameters to zero for certain topics k and task t. The l_2 norm for the topic dimension however, tends to balance the deviation of topic-task-specific parameters $\theta_{k,t}$ from the global θ^0 to be similar. In other words, the competition is now across tasks within each topic rather than among topic-task pairs under topic-task-element constraint. This norm is first applied to sparsity constraint, to our best knowledge. The usage of similar exclusive sparsity constraint, the square of $l_{1,2}$ norm in [62] shows its effect to find sparse feature selection structure for each neuron. The topic-task-sparsity designs given two proposed constraints are visualized as the norms of the learned topic-task-sparse parameters in Fig. 3.2.

3.4.3 Topic-Wise Multi-Task Sparsity Model

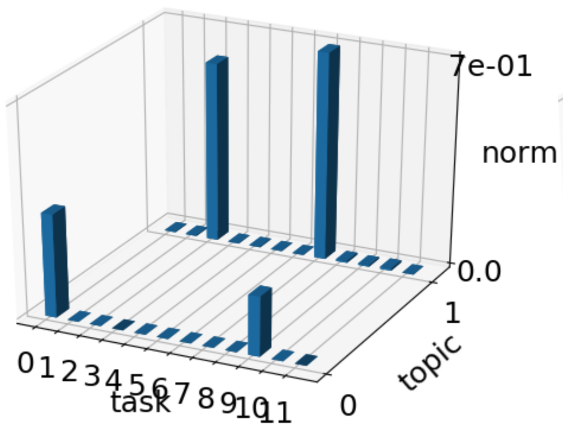
The topic-wise multi-task sparsity model (TMTS) is the combination of the topic-wise multi-task architecture and either of the topic-task-element or topic-task-exclusive constraints. As shown in Fig. 3.1, we use a common shared-bottom module $\phi(x)$ to extract features and task-specific modules $\hat{y} = \psi(f(x,t),t)$ before final output. The topic module $h(x)$ can be normalized by a softmax function.



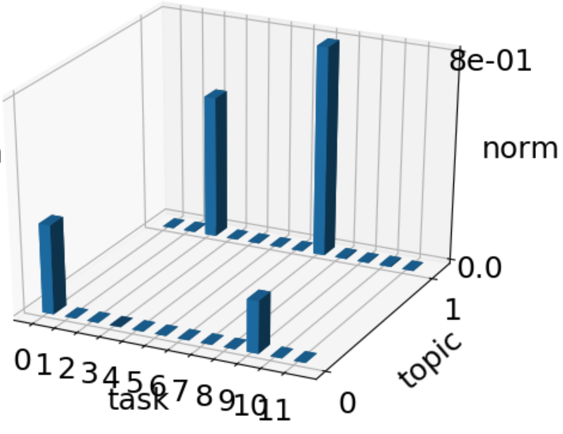
(a)



(b)



(c)



(d)

Figure 3.2: The topic-wise multi-task structures as the l_2 norm of topic-task-sparsity parameters from the synthetic data: (a) ground truth as the weight $w^s_{k,t}$ used to generate the data; (b) without topic as learned by TMTS-el with $K = 1$; (c) TMTS-el, as learned by TMTS-el with $K = 2$; (d) TMTS-ex, as learned by TMTS-ex with $K = 2$.

The overall loss function $\mathcal{L}$ is defined as

$$\mathcal{L} = \sum_{t \in [T]} \frac{1}{N_t} \sum_{n_t \in [N_t]} L(\hat{y}, y_t) + \lambda \Omega(\Theta^s), \tag{3.5}$$

where Ω can either be Ω^{el} or Ω^{ex}, and λ controls the strength of the sparsity penalty. When $\lambda \rightarrow +\infty$, the topic-task-specific functions $g(x, \theta_{k,t})$ reduce to a global function $g(x, \theta^0)$, leading to the closest task relationship, and vice versa.

3.4.4 Optimization

Both topic-task-element and topic-task-exclusive constraints are non-smooth functions. Therefore we use the stochastic proximal gradient method to minimize Eq. 3.5. At each iteration j, it calculates an intermediate parameters $\bar{\Theta}^s$ using the conventional SGD step and optimizes the solution or proximal operator as

$$\Theta^{s,j+1} = \arg\min_{\Theta^s} \frac{1}{2\lambda r} ||\Theta^s - \bar{\Theta}^s||_2^2 + \Omega(\Theta^s), \tag{3.6}$$

where r is the learning rate of the current iteration. As for the topic-task-element constraint $\Omega^{el}(\Theta^s)$ in Eq. 3.3, the proximal operator, the proximal operator from Eq. 3.6 for each topic k and each task t can be calculated independently. Therefore, the proximal operator can be easily derived as

$$prox_{el}(\theta^s_{k,t}) = (1 - \frac{\lambda r}{||\bar{\theta}^s_{k,t}||_2})_+ \bar{\theta}^s_{k,t}, \tag{3.7}$$

where $()_+$ is the clip function $max(,0)$. The proximal operator for the topic-task-exclusive constraint $\Omega^{ex}(\Theta^s)$ is more complicated because the parameters for different tasks are coupled by the l_2 norm at the topic-dimension.

Lemma 3.1. *The solution for Eq. 3.6 with $\Omega = \Omega^{ex}$, is*

$$prox_{ex}(\theta^s_{k,t}) = (1 - \frac{A_k}{||\bar{\theta}^s_{k,t}||_2})_+ \bar{\theta}^s_{k,t}, \tag{3.8}$$

where A_k is maximum of the "diluted" average $A_{\mathcal{T}'}$ of $\{||\theta^s_{k,t}||_2 \,|t \in \mathcal{T}'\}$, $\forall \mathcal{T}' \subset [T]$, $A_k = \max_{\mathcal{T}' \subset [T]} A_{k,\mathcal{T}'}$, s.t., $A_{k,\mathcal{T}'} = \frac{1}{\frac{1}{\lambda r} + |\mathcal{T}'|} \sum_{t' \in \mathcal{T}'} ||\bar{\theta}^s_{k,t'}||_2$.

The key of its proof is to notice that the maximum "diluted" average A_k is a threshold that divide the tasks into two sets, $\mathcal{T}_k$ and $[T] \backslash \mathcal{T}_k$, where $\mathcal{T}_k = \arg\max_{\mathcal{T}'} A_{k,\mathcal{T}'}$.

Lemma 3.2. *If $t \in \mathcal{T}_k$, then $||\bar{\theta}^s_{k,t}||_2 \geq A_k$ and if $t \in [T] \backslash \mathcal{T}_k$, $||\bar{\theta}^s_{k,t}||_2 \leq A_k$.*

Proof. First, assume otherwise $||\bar{\theta}^s_{k,t}||_2 < A_k$ for some $t \in \mathcal{T}_k$. It can be seen that

$$||\bar{\theta}^s_{k,t}||_2 < A_k \Leftrightarrow ||\bar{\theta}^s_{k,t}||_2 < A_{k,\mathcal{T}_k \backslash \{t\}}. \tag{3.9}$$

Using this, we obtain

$$A_k - A_{k,\mathcal{T}_k\backslash\{t\}} = \delta\,[\,||\bar{\theta}^s_{k,t}||_2 - A_{k,\mathcal{T}_k\backslash\{t\}}] < 0 \tag{3.10}$$

where $\delta = \frac{(\frac{1}{\lambda r}+|\mathcal{T}_k\backslash\{t\}|)}{(\frac{1}{\lambda r}+|\mathcal{T}_k|)(\frac{1}{\lambda r}+|\mathcal{T}_k\backslash\{t\}|)} > 0$. It contradicts with the condition that A is the maximum "diluted" average. Second, for the second statement, it is straightforward to prove with $\mathcal{T}_k\backslash\{t\}$ replaced by $\mathcal{T}_k \bigcup\{t\}$. □

Algorithm 3.1 Greedy Calculation of A_k

```
Input: $||\bar{\theta}^s_{k,t}||_2$ for $t \in [T]$, $\lambda, r$
output: $A_k$
1:    Sort $||\bar{\theta}^s_{k,t}||_2$, and denote $a_i = ||\bar{\theta}^s_{k,t_i}||_2$ for $i \in [T]$
       s.t., $a_1 \geq a_2 \geq ... \geq a_T \geq 0$;
2:    $S_0 \leftarrow 0$, $S_i \leftarrow S_{i-1} + a_i$ for $i = 1, 2, ..., T$;
3:    $S_i \leftarrow \frac{1}{\frac{1}{\lambda r}+i} S_i$ for $i \in [T]$;
4:    Return $A_k = \max_i S_i$.
```

With Lemma. 3.2, Lemma. 3.1 can be proved by sub-differential calculus. The maximum "diluted" average A_k in Eq. 3.8 can be calculated by a simple greedy algorithm (Algorithm. 3.1), with time complexity $O(Tlog(T))$. The proof of correctness is straightforward with Lemma. 3.2.

Proof. Using Lemma. 3.2, we can prove that $\mathcal{T}_k$ must be the subset of the largest $|\mathcal{T}_k|$ norms. Formally, $\forall t \in \mathcal{T}_k$, $t' \in [T]\backslash\mathcal{T}_k$, we have $||\bar{\theta}^s_{k,t}||_2 \geq A$, and $||\bar{\theta}^s_{k,t'}||_2 \leq A$. Hence $||\bar{\theta}^s_{k,t}||_2 \geq ||\bar{\theta}^s_{k,t'}||_2$. □

3.5 Experiment

In this section, we apply the proposed models against one synthetic and four real-world datasets to validate the proposal. All codes and datasets will be published. [1]

[1]Link will be added here after paper gets published.

3.5.1 Datasets

3.5.1.1 Synthetic Dataset

We generate a synthetic MTL dataset following the within-topic task relationship hypothesis.

1. Input feature: K topic cores $e_k \in \mathbf{R}^d$ are sampled from normal distributions $\mathcal{N}(0, \sigma_e^2)$ for each $k \in [K]$. Input features are generated as $x_t \in \mathbf{R}^d$ from normal distributions $\mathcal{N}(0, \sigma_x^2)$. The topic distribution $h(x)$ of an input x is determined as $h(x)_k \propto \exp(||x - e_k||_2^2/\sigma_{topic}^2)$.

2. Parameters: a global linear weight $w_0 \in \mathbf{R}^{d \times p}$ is element-wise sampled from $\mathcal{N}(0, \sigma_{w_0}^2)$. Topic-task-sparsity weight $w_{k,t}^s$ are generated in the way that within each topic, $z \ll T$ tasks $\mathcal{T}_k \subset [T]$ are randomly sampled and their topic-task-sparsity weight are assigned random values from $\mathcal{N}(0, \sigma_{w^s}^2)$, while the rest are assigned 0. The topic-task-specific weight is $w_{k,t} = w_0 + w_{k,t}^s$.

3. Label: The linear output $\bar{y}_t$ for input x_t is generated by $\bar{y}_t = softmax(\sum_{i=1}^d x_{t,i} \sum_{k=1}^K h(x_t)_k w_{k,t,i})$. We add non-linearity to the final label $y_t = \beta(\bar{y}_t)$, with β the non-linear function used in [42].

3.5.1.2 MNIST-MTL Dataset

We use the multi-task version of the MNIST data (MNIST-MTL) [63]. Each task is a binary classification problem that distinguish one digit from the others. For each of the $T = 10$ tasks, we sample 900 positive samples and 900 negative samples with 100 samples for each of the other digits. We adopt the feature extraction method used in linear methods [44] to get input of dimension $d = 64$.

3.5.1.3 AwA2 Dataset

AwA2 is a benchmark dataset containing 37,322 images of 50 animals [64]. Each task is a binary classification problem similar to MNIST-MTL data. For each of the $T = 50$ tasks, we sample 50 positive samples and 1 negative samples for each of the other animals. We use the pre-trained features [64] and reduce the dimension to 500 with PCA.

data	Synthetic	MNIST-MTL	AwA2	School	MCSEM
input dim d	64	64	500	28	756
output dim p	5	2	2	3	5
# tasks T	12	10	50	139	12
# samples/task N_t	2,000	1,800	100	111	3,523

Table 3.1: Data statistics summary

3.5.1.4 School Dataset

School data is a benchmark dataset containing performance of 15362 students from 139 schools [65]. The score performance is partitioned to 3 segments, $[0, 10)$, $[10, 20)$ and $[20, 71)$. Each task is to classify the performance of students from a school.

3.5.1.5 MCSEM Dataset

The multi-channel social emotion mining (MCSEM) data is crawled from public posts from 12 news channels on Facebook, together with their public users' emotional reactions (i.e., clicks on the emoticon buttons, love, angry, wow, happy, and sad). We used the pre-trained BERT model [66] to obtain the document embeddings as the input with $d = 756$. The emotional reactions for each post are normalized to label distributions over the five emoticon labels. Each task is to predict the label distribution given the posts of each channel.

3.5.1.6 Statistics of Datasets

The data statistics is summarized in Table. 3.1. We use sample weighting to ensure that the sums of all sample weights for different tasks are the same, following $\mathcal{L}$ in Definition 3.1. The task-wise data imbalance problem is beyond the scope of this work. For each dataset, 20% samples are used for testing the remaining 80% as training. The results reported are averages from 10 iterations of random splits. The split uses stage-wise sampling with tasks as stages to avoid random imbalance across tasks.

3.5.2 Competing Models

We compare the proposed models with a list of state-of-the-art MTL neural network models.

First, we use two baselines to validate the usage of MTL framework: **Separate**, learns each task with separate neural network modules that do not correlate; on the other hand, **Single**, learns all tasks with a single neural network module.

Besides them, a comprehensive list of state-of-the-art models are included. **Shared-bottom**: it is a broadly used MTL model where all tasks share bottom feature extraction module and own their own top modules. **Inter-task-l_2** [56]: the l_2 penalty is applied to constrain task-specific module parameters difference. **DMTRL** [55]: the tensor consisting of task-specific parameters of all tasks are assumed of a low-rank structure, modeled by tensor factorization. **MRN** [57]: the tensor consisting of task-specific parameters of all tasks are assumed with a fully-decomposed tensor normal distribution, whose parameters are jointly learnt with model parameters. **Cross-stitch** [58]: task-specific modules are assumed to be able to communicate with each other by stitches connection between each pair of them. **MMoE** [42]: consists of multiple expert modules and task-specific expert distribution modules to combine the output of experts for each task.

We experiments the two variants of the proposed model TMTS, **TMTS-el**, TMTS with topic-task-element constraint Ω^{el} from Eq. 3.3 and **TMTS-ex**, with topic-task-exclusive constraint Ω^{ex} from Eq. 3.4.

Here in this work, we only focus on their capability to capture task relationship in MTL problems. Therefore, we implement a unified architecture for all models. They share the same shared-bottom module structure as MLP (multi-layer perceptron) of one layer except **Separate** and other modules of different models are MLP. All models are trained using stochastic gradient descent (SGD) with learning rate at iteration i, $r_i = r_0\gamma^{i/\eta}$, where r_0 is the initial learning rate, γ is the decay rate and η is the decay steps. Random dropout for certain layers and l_2 regularization are used to avoid overfitting.

3.5.3 Results

3.5.3.1 Performance metric

As the evaluation metric, we use the *miss-classification rate* for the experiments on Synthetic, MNIST-MTL, AwA2 and School datasets, and the *cross-entropy* for the experiments on MCSEM dataset where label distribution is given.

Q1: Are the proposed models able to capture the within-topic task

Dataset	synthetic	MNIST-MTL	AwA2	School	MCSEM
Separate	16.55	2.59	7.49	50.95	1.371
Shared-bottom	14.51	2.68	4.92	50.01	1.326
Single	14.24	49.98	16.75	51.80	1.342
Inter-task-l_2	14.21	2.49	4.92	50.41	1.322
DMTRL	15.08	2.60	4.75	49.34	1.333
MRN	14.47	2.68	9.81	51.06	1.329
Cross-stitch	14.69	2.68	4.63	50.22	1.327
MMoE	14.40	2.59	13.60	48.18	1.337
TMTS-el	**14.09**	2.44	**4.16**	**45.00**	1.322
TMTS-ex	14.10	**2.32**	5.50	46.72	**1.321**

Table 3.2: Overall performance

relationship?

We show the l_2 norm of topic-task-sparsity parameters learned from the synthetic data in Fig. 3.2. Compared to Fig. 3.2.(a), the ground truth parameters, both the TMTS-el (c) and TMTS-ex (d) models can exactly capture the sparsity structure. We also test the case without topic by TMTS-el (Similar results can be obtained by TMTS-ex) (b), which cannot find the similarity between the majority tasks and task 2 and 7 (task 0 and 9) in data from topic 0 (1), but only treat all of them different from other tasks. This shows the effect of the topic-wise multi-task architecture.

Q2: How do the proposed models perform?

The overall performance results are presented in Table. 3.2. First, the proposed models **TMTS-el** and **TMTS-ex** consistently outperforms all the competing models. This validates the superiority of the proposed topic-wise multi-task architecture and also the proposed two topic-task-sparsity MTL designs. Second, the task relationship varies a lot across different datasets. On the one hand, comparing **Single** and **Separate**, which are the two extreme cases in MTL, their performance difference in different datasets varies. Therefore, some of the datasets (e.g., MNIST-MTL) have task relationship that is hard to catch, while others (e.g., synthetic) make it more beneficial to risk negative transfer for more data. On the other hand, the performance of different models, which are different assumptions of task-relationship, vary across different datasets. For example, **Cross-stitch** performs

Dataset	synthetic	MNIST-MTL	AwA2	School	MCSEM
Separate	0.00	0.00	0.00	0.00	0.00
Shared-bottom	2.44	-0.09	3.05	3.18	3.08
Single	2.75	-48.66	-9.70	-2.04	2.20
Inter-task-l_2	2.79	0.10	3.05	1.96	3.39
DMTRL	1.75	-0.01	3.22	5.56	2.68
MRN	2.49	-0.08	-2.34	-0.05	3.00
Cross-stitch	2.23	-0.08	3.35	2.76	2.98
MMoE	2.57	0.00	-6.29	8.96	2.39
TMTS-el	**2.94**	0.16	**3.88**	**16.74**	3.38
TMTS-ex	2.92	**0.28**	2.41	12.93	**3.62**

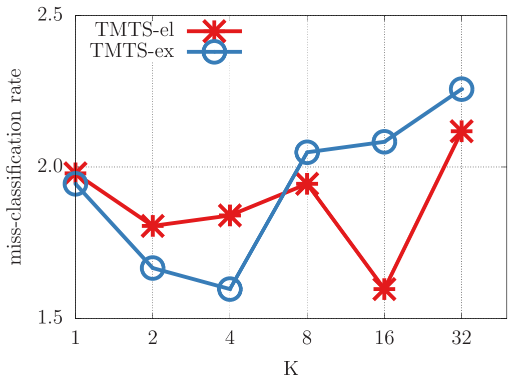
Table 3.3: Average task-wise improvement percentage

good on MCSEM data, but even worse than **Separate** baseline on synthetic data. This shows that the topic-wise multi-task architecture is more flexible in leveraging different task relationship.

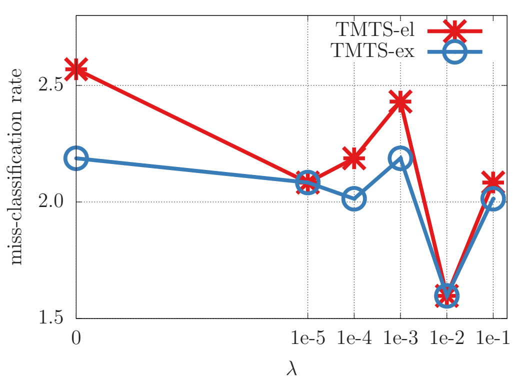
Q3: How is the trade-off between positive and negative transfer?

The topic-wise multi-task architecture is proposed to capture more subtle task relationship so that achieve better trade-off between positive and negative transfer. Table. 3.2 gives an overall view of the answer, that the proposed architecture does perform better. Further more, here we take a detail view. In Table. 3.3, the average task-wise improvement is presented. For each model on each dataset, we calculate its performance on each task t as $perf_{model,t}$ of the dataset. After that, for each task, we calculate the relative improvement over the **Separate** model $100 * (perf_{\mathbf{Separate},t} - perf_{model,t})/perf_{\mathbf{Separate},t}$. This task-wise improvement provides the judgement of positive or negative transfer. When this improvement is negative, for the specific task, there is no benefit to take into account other tasks, which is negative transfer, and vice versa. For each entry in Table. 3.3, we report the average task-wise improvement percentage over all tasks of a datasets from a model. We observe that the proposed models give consistent and better improvement over all datasets. Therefore, it shows that the proposed methods do achieve better trade-off between positive and negative transfer.

Q4. Ablation study: Are the proposed architecture and MTL design nontrivial in real world data?



(a)



(b)

Figure 3.3: Ablation study: (a) Performance over different number of topics K; (b) performance over different sparsity constraint strength λ

We notice that there are two hyperparameters that distinguish the proposed models from the existing ones. First when the number of topics $K = 1$, the proposed topic-wise multi-task architecture reduces to conventional MTL architecture. From the ablation study with different K values, Fig. 3.3(a), the performance of TMTS-el (TMTS-ex) with $K = 16$ ($K = 4$) is better than that of trivial model with $K = 1$. This validates the topic-wise multi-task architecture. Second when the sparsity penalty strength, $\lambda = 0$, the sparsity constraints are disabled. From the ablation study with different λ values, Fig. 3.3(b), the best performance is achieved with $\lambda = 1e - 2$. It validates the topic-task-sparsity MTL design.

3.6 Conclusion

In this work, from a closer look into the validity of task relationship, we propose a within-topic task relationship hypothesis and develop a topic-wise multi-task architecture, which is general enough to be combined with any existing MTL design. Further, we propose the topic-task-sparsity MTL design, specially designed for the topic-wise multi-task architecture, along with two types of sparsity constraints. The experiments on both synthetic and real-world datasets show that the proposed models outperform existing state-of-the-art models consistently, which supports the validity of the within-topic-task relationship hypothesis.

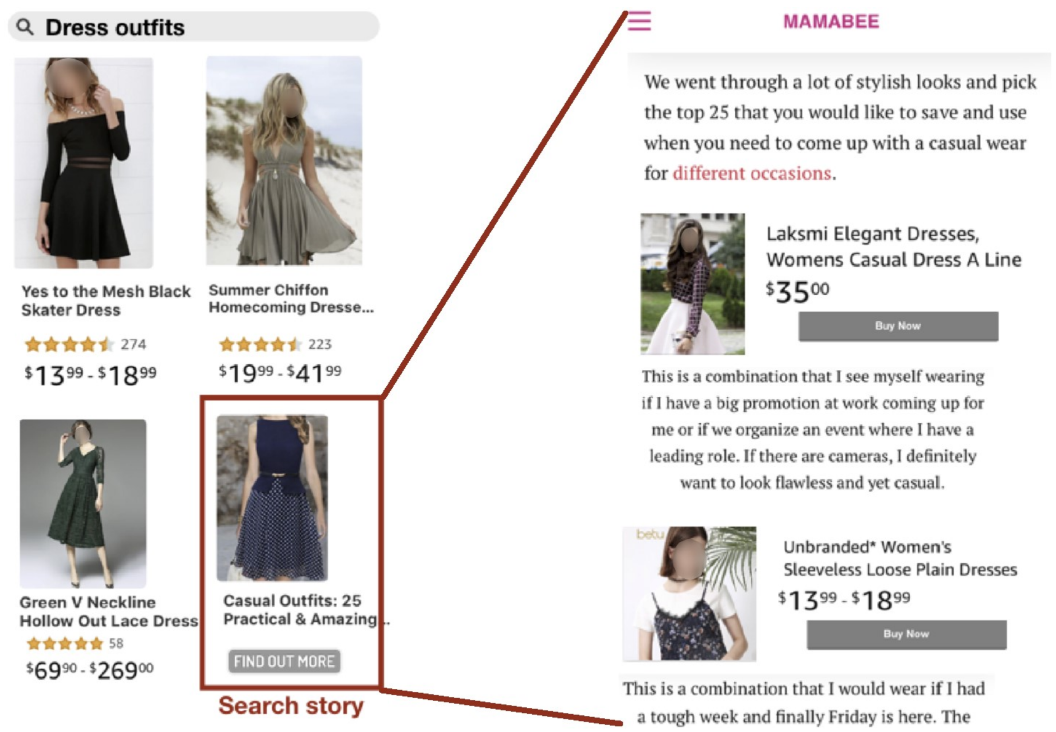
Chapter 4 | How to Influence User Behavior?

In this chapter, a reinforcement learning framework is applied to influence user behavior on purpose. More specifically, it is applied to the personal search story problem in industry.

4.1 Introduction

Imagine that a customer visits a retail shop to purchase a dress that she likes. As the customer walks in, a business assistant is present to assist the customer by answering questions on fashion trend or suggesting related dresses. In online e-commerce applications, more business units are adding a component that plays a similar role as the business assistant in a shop. In this paper, we are interested in a particular component, commonly referred to as **search story**, that has become popular among e-commerce search engines on many online platforms. For instance, in news feed platforms and web and image search platforms, each search story is a display of recommended high-quality content which is relevant to a user's personal interests. In e-commerce search platforms, a search story is instead a display of sponsored article which gives an overview and comparison of several product items. Figure 4.1 illustrates an example of a search story in a real e-commerce search engine, which is embedded within the organic search results. In this example, the search story itself displays, when clicked, a landing page of a short survey that summarizes and compares a list of selected product items and related styles.

The search story recommendation can be naturally formulated as a conventional


Dress outfits
Yes to the Mesh Black Skater Dress
274
$13.99 - $18.99
Summer Chiffon Homecoming Dresse...
223
$19.99 - $41.99
Green V Neckline Hollow Out Lace Dress
58
$69.90 - $269.00
Casual Outfits: 25 Practical & Amazing...
FIND OUT MORE
Search story
MAMABEE
Laksmi Elegant Dresses, Womens Casual Dress A Line
$35.00
Buy Now
Unbranded* Women's Sleeveless Loose Plain Dresses
$13.99 - $18.99
Buy Now

We went through a lot of stylish looks and pick the top 25 that you would like to save and use when you need to come up with a casual wear for different occasions.

This is a combination that I see myself wearing if I have a big promotion at work coming up for me or if we organize an event where I have a leading role. If there are cameras, I definitely want to look flawless and yet casual.

This is a combination that I would wear if I had a tough week and finally Friday is here. The

Figure 4.1: An illustrative (not a screenshot) example of search story recommendation. The left part shows a search result page that displays a search story within organic product item search results; the right part is the landing page after clicking the search story, which contains both shopping guides and shopping product items.

recommendation or ranking problem that aims to suggest relevant items to users based on search keywords. For instance, one may model the problem as a click-through prediction task and recommend the search story with the highest predicted click-through rate. However, compared with conventional recommendation systems or search engines, recommendation of search stories focuses more on guiding users to figure out their own preferences and personal intents. Consider the following concrete example that illustrates a multitude of objectives of a search story recommender.

Example 4.1. *As shown in Figure 4.1, suppose a customer wants to purchase a "dress outfit" for a party, but she does not know what exact style she is looking for (e.g., "sleepless loose plain dress"). The purpose of a search story recommender is to assist and guide the customer within each search session, as if it plays the role of an assistant in a shop. On the one hand, a user's search session history can be leveraged to learn the user's intent and subsequently to build a better recommendation model for future search stories. On the other hand, the recommended search story guides* *the user to figure out her preferences and personal intents, which affects not only her immediate behavior (e.g., clicking or ordering product items from the current page of the search story in Figure 4.1(b)), but also her long-term behavior (e.g., clicking or ordering product items in future search session in Figure 4.1(a)).*

As this example illustrates, the ultimate goal of a search story recommendation in e-commerce search applications is to recommend the best search story that maximizes both *short-term* reward (e.g., purchasing a product shown in the landing page of a search story) and *long-term* reward (e.g., returning back to start another search session in a week). Compared with organic search results, search stories risk disrupting users' current search to achieve better long-term benefit in their following search. Therefore, search story recommendation requires a solution to consider both immediate and future benefits. Although we consider direct feedbacks (i.e., users' clicking or ordering product items in the landing page of search stories), indirect feedbacks (i.e., users' clicking or ordering product items in the search page) is more important. Such a cross-channel effect [67] is difficult to model using the conventional supervised learning framework. This motivates us to propose a novel reinforcement learning framework for personalized search story recommendation.

In this work, we focus on the following characteristics of the personalized search story recommendation problem: the return cannot be directly optimized by supervised learning (i.e., long-term or cross-channel); only heuristic policy has been applied as the existing policy (i.e., recommending the search story with the highest predicted click-through rate); and it is costly to explore the environment. The first factor motivates the use of reinforcement learning as discussed above, and the other two factors bring two additional challenges.

- **Offline data**: Offline data should be effectively used. Because it is too expensive to explore the environment of the RL problem, which is real-world users [68] in our context, only offline data can be used for learning and evaluation.

- **Stability**: The learned policy should not be far from the existing one, which is less mentioned in applications on recommendation. Due to the difference of the optimization goals between supervised learning used by existing systems and RL methods, the latter may provide results (i.e., recommendations) clearly different from existing one, which is highly risky for an industry platform. In personalized search story recommendation, the existing recommender is trained with click-through rate of stories in a supervised manner, while the RL method is to optimize users' long term preference on organic search results. On the other hand, the difficulty of offline evaluation of RL methods enlarge the risk when the learned policy is far from the existing one. In personalized search story recommendation, offline evaluation requires an estimation of user behavior, building the relationship between search story and organic results. When such an estimation is done by a user model, the difference between learned policy and the existing one will cause selection bias [69]; when the estimation is done by importance sampling, large difference leads to large variance [70].

The two challenges intervene each other. Effective use of offline data suggests larger policy improvement given limited data. However, larger policy improvement implies larger difference against the logging policy, which means less stability. Therefore, a trade-off has to be considered between the two challenges.

The characteristics of the personalized search story recommendation problem should also be considered in other real-world application scenarios. For example, in dynamic treatment recommendation [71], where treatments are recommended to patients dynamically given their status, the return, the mortality rate, cannot be directly optimized by supervised learning; existing dynamic treatment is given by doctors based on their knowledge and experience; and it is costly to explore the treatment effect on real patients. For another example, in intelligent traffic signal control [72], the return, the traffic efficiency, cannot be directly optimized by supervised learning; existing traffic lights control is by heuristic rules designed by domain experts [73]; and it is risky to explore real traffic. Therefore, our work can be adapted to those applications or any others with the three characteristics.

In order to address the two challenges, we propose a deep reinforcement learning framework, named as DRESS (Deep REinforcement learning for Search Story recommendation) with (i) a combination of both imitation learning and reinforcement learning, as well as (ii) a combination of both model-based and model-free reinforcement learning. The imitation learning procedure fits a policy to offline data, so that on one hand the stochastic logging policy of offline data is estimated. This imitated policy works as several roles of the architecture to keep the learned policy close to the logging one, as required by the **stability** property. A dynamic model is trained by offline data to infer user behavior pattern (i.e., the environment) and is applied as the virtual environment for further controller learning besides the direct use of offline training data. The dynamic model makes the method more data effective by extrapolating offline data. With the help of imitated policy and dynamic model, we propose a RL learning strategy as safe learning with offline data, augmented by online learning with a dynamic model to achieve both effective data usage and stability.

The main contribution of this work can be summarized as follows:

Novel Problem. We study an emerging search story recommendation problem and develop a solution based on deep reinforcement learning framework, addressing the challenges that originate from its cross-channel and long-term property.

Sound Methodology. We propose a framework combining model-free and model-based reinforcement learning, as well as imitation learning and reinforcement learning to achieve both effective usage of offline data and stability.

Practical Solution. Experiments on real-life data sets from *JD.com* have empirically demonstrated the effectiveness of our proposed solution.

4.2 Related Works

In this session, we briefly review two topics that are relevant to our work, namely reinforcement learning and recommendation/ranking.

4.2.1 Reinforcement Learning

In the general reinforcement learning framework, an agent sequentially interacts with the environment and learns to achieve the best return, which is in the form of accumulated immediate rewards. In the partially observable Markov decision process (`POMDP`) model, at each time step t, when the agent has the observation of the environment o_t, an action a_t is taken to obtain a reward r_t from the environment. As the environment is partially observable, the state s_t of the environment at time t can only be inferred from the whole history up to time t, which can be denoted as $s_t = \delta(o_1, a_1, r_1, ..., o_{t-1}, a_{t-1}, r_{t-1}, o_t)$. The goal of the reinforcement learning problem is to learn an optimal policy, a mapping from state s to action a, to maximize the expected accumulated long term reward.

Deep reinforcement learning has achieved remarkable success in various tasks including but not limited to game playing [74, 75], search and recommendation [76, 77], robotics and autonomous vehicles [78, 79], online advertising [80, 81], several NLP tasks [82, 83], and database management systems [84–86]. We refer readers to surveys [87, 88] for more details.

4.2.2 Reinforcement Learning in Recommendation and Ranking

Conventional works on recommendation focus on one round static optimization of the recommendation model. To better incorporate real-time user's feedback, several contextual bandit based ranking/recommendation approaches [89–91] were proposed to update the selection strategy based on user-click feedback to maximize total user clicks.

However, a major assumption of bandit approaches is the ineffectiveness of action (i.e., choice of arms in bandit) on the environment state transitions, which fails in personalized search story recommendation scenario, where the environment state or users' preference and intent here will be affected by the recommended search story. Hence we turn to reinforcement learning (RL) framework which can take into account the long-term effect of current actions.

There are some pioneering works applying RL to different tasks in recommendation and ranking, such as cross-channel recommendation [67], personalized news recommendation [77], impression allocation of advertisements [92], and learn-to-rank for search sessions [93]. Their motivations to use RL are all based on the long-term effect of current actions in the corresponding problems. For example, in personalized news recommendation, the current recommended piece may shape users' interests so that it can affect later recommendation results [77]. Further works consider variants of the settings of the recommendation problem, such as pairwise [94] and page-wise [95], which are not applicable to our problem.

There are also recent works exploring learning from offline data. The off-policy correction [68] explores the combination of imitation learning and reinforcement learning for recommender system. However, the imitation is only used in estimating the stochastic logging policy, and no model-based learning is used to improve data effectiveness. The user engagement recommender system [69] is learnt by a model-based method. However the environment model only considers reward-related feedback, such as users' clicks or orders of items [93], while the environment model in our problem includes state transition and other feedback predictions due to its cross-nature property. No existing works in RL in recommendation and ranking considers stability and its trade-off with data effectiveness.

4.3 Problem Definition

4.3.1 Preliminary

For ease of presentation, we first introduce the list of notations and basic concepts used through the entire work. The notations are summarized in Table. 4.1. Specifically, we use lower case symbols u, q, d, p to represent a single user, query, story item, and an item from another channel (e.g., the product item), respectively. Upper case symbols U, Q, D, P are used to represent a set of users, queries, story items, and product items respectively. Let f denote the search story recommendation function that maps a context c to a selected story $d \in D$. The context c can be a specific query q for general search or a specific user u for recommendation or a single user u plus a single query q for personalized search. With the above notations, we define the concept of *search session* and *personalized search episode* as follows.

Definition 4.1 (**Search Session**). *A search session is a series of feedback $\mathcal{I}$ (e.g., click, order, page view) by the user u at time t towards the returned page with a search story d addressing a given query q. Formally, we can use a tuple $e =< t, u, q, d, \mathcal{I} >$ to denote a search session.*

Definition 4.2 (**Search Episode**). *A search episode E is a temporal sequence of search sessions by the user u, which is denoted as $E = (e_1, \cdots, e_t, \cdots, e_T)$. We add a subscript to E (e.g., E_u) to denote a search episode of a specific user u.*

4.3.2 Problem Formulation

As introduced earlier, in this work, we focus on reinforcement learning for personalized search story recommendation. Specifically, we aim to find a strategy that updates the search story item recommendation function of a search engine along search episodes to achieve the best reward for each user.

$u(U)$	user (set)	$\mathcal{D}_{Log}$	logging data
$q(Q)$	query (set)	$\mathcal{D}_{RL}$	imagination generated data
$d(D)$	story (set)	M^T	transition model
$p(P)$	product item (set)	M^R	reward model
$t(T)$	time step (episode length)	h	user's hidden feature
y_d	story click	M^S	state module
y_p	product click	$\mathbb{V}_\vartheta$	value function (critic network)
y_{rp}	feature of product clicked	π_Θ	policy (actor network)
$\mathcal{I}$	user feedback (y_d, y_p, y_{rp})	π_Θ^0	imitated policy
e	search session	b	logging policy of $\mathcal{D}_{Log}$
E	search episode	γ	discount factor
o_t	observation	w	weights of loss terms
a_t	action	ϵ	clipping factor
s_t	state	H	horizon of TWIS
r_t	reward or reward function		
R^T	cumulative reward		

Table 4.1: Notations summary

When putting the personalized search story recommendation into the general reinforcement learning framework, the corresponding observation o_t, the action a_t, the state s_t, the transition $\mathbb{T}$, the reward r_t are defined as:

Observation o_t is the user-dependent and query-dependent feature x represented as $x(u, q, t)$ or in short x_t.

Action a_t is the selection of the search story $d \in D$.

State s_t is the combination of users search episode up to time t, i.e., the history $E^{1:t-1} = (e_1, ..., e_\tau, ..., e_{t-1})$, and the observation o_t.

Transition $\mathbb{T}$ is the state transition function dependent on a_t, $s_{t+1} = \mathbb{T}(s_t, a_t)$.

Reward $r_t(s_t, a_t)$ can be quantified as the number of clicks, the number of orders, or gross merchandise volume received from users when users are in state s_t and search story recommender performs action a_t. In this work, we set reward as the binary indicator y_p of whether a user clicks any products in the search session e_t.

Therefore, in this work, we aim to solve the following problem:

Problem 4.1. *Given the entire search episode of a user E_u, we aim to sequentially refine the action towards each search session e_u based on observed feature space X and a policy $\pi(a|s)$ as a distribution of actions conditioned on states. Specifically, the objective is to find the best policy to maximize the estimated cumulative rewards. That is:*

$$\begin{aligned} \arg\max_{\pi} \quad & \mathbb{E}[R^T \mid s_1, \pi] \\ \textit{subject to} \quad & R^T = \sum_{\tau=1}^{T} \gamma^{\tau} r_{\tau}(s_{\tau}, a_{\tau}), \end{aligned} \tag{4.1}$$

where R^T is the discounted cumulative rewards, $\gamma \in [0, 1]$ is the discount factor, and $\mathbb{E}[X]$ denotes the expectation of X.

4.4 Deep Reinforcement Learning for Search Story Recommendation

In this section, we give an overview of our deep reinforcement learning framework for personalized search story recommendation, referred to as DRESS. Given only offline training data, we propose to combine both model-based augmentation and imitation learning with the conventional reinforcement learning. Model-based reinforcement learning requires much less training data compared with model-free reinforcement learning. On the other hand, imitation learning estimates the logging policy (that leads to the offline data) from the offline data, which is both the initialization of the actor network and a critical component in safe policy learning algorithm, which ensures stability of the learned new policy.

The approach is outlined in Algorithm 4.1. Randomly sampled logging search session data are collected and added to dataset $\mathcal{D}_{Log}$, which is first used to train the dynamic model M_θ to fit the environment (user behaviors), as proposed in Section 4.5 (Line 1). Search story recommendation controller is built upon the Actor-Critic framework [83], which is parametrized as π_Θ and $\mathbb{V}_\vartheta$ (Line 2). Next, instead of directly performing reinforcement learning with environment, an initial policy was learned from log data $\mathcal{D}_{Log}$ with the controller imitation learning (Line 3). We thus further improve the initial policy with a standard proximal policy gradient approach [96] from the logging data $\mathcal{D}_{Log}$ (Line 4), where the imitated policy is used as the logging policy since it is an estimate of the logging policy. The proximal policy gradient approach and the imitated initial policy, both enabled by the imitation learning, together ensure that the policy learned is not too far from the logging policy that is currently used in *JD.com*.

However, the policy learned only directly from the offline data $\mathcal{D}_{Log}$ using proximal policy gradient may not be enough for performance improvement in terms of the comparison against the baseline (logging) policy on cumulative rewards, because it is restricted to be close to the estimated logging policy. Therefore it is ideal to apply the learned policy to the environment, obtain on-policy logging data, and iteratively improve the policy. In our framework, our "on-policy" data are generated by the interaction between the dynamic model M_θ and our search story recommendation controller π_Θ. The process is called imagination. The dynamic model serves as a virtual environment that interacts with our search story recommendation controller to alleviate the challenge of using only offline data. We thus repeatedly perform the following procedure to learn a better policy: 1) perform imagination to gather new session data and add them to a separate dataset $\mathcal{D}_{RL}$; 2) perform controller reinforcement learning to improve the recommendation policy from $\mathcal{D}_{RL}$ (Lines 5 – 8).

Algorithm 4.1 DRL for Search Story Recommendation

```
Input: Logging Data $\mathcal{D}_{Log}$
Output: The search story recommender
1: $M_\theta$ = Dynamic_Model_Training($\mathcal{D}_{Log}$) (Section 4.5)
2: Initialize the critic network $\mathbb{V}_\vartheta$, actor network $\pi_\Theta$
  // imitation learning
3: $\pi_\Theta^0$ = Controller_Imitation($\mathcal{D}_{Log}$, $M_\theta^T$) (Section 4.7.1)
  // one step reinforcement learning on $\mathcal{D}_{Log}$
4: $\pi_\Theta$, $\mathbb{V}_\vartheta$ = Controller_Learning($\mathcal{D}_{Log}$, $\mathbb{V}_\vartheta$, $\pi_\Theta^0$)
                                        (Algorithm 4.2)
  // reinforcement learning on $\mathcal{D}_{RL}$ (Section 4.6)
5: repeat
6:     $\mathcal{D}_{RL}$ = Imagine($M_\theta$, $\pi_\Theta$) (Section 4.7.3)
7:     $\pi_\Theta$, $\mathbb{V}_\vartheta$ = Controller_Learning($\mathcal{D}_{RL}$, $\mathbb{V}_\vartheta$, $\pi_\Theta$)
                                        (Algorithm 4.2)
8: return $M_\theta$, $\mathbb{V}_\vartheta$, $\pi_\Theta$
```

4.5 Dynamic Model

4.5.1 Illustrative Overview

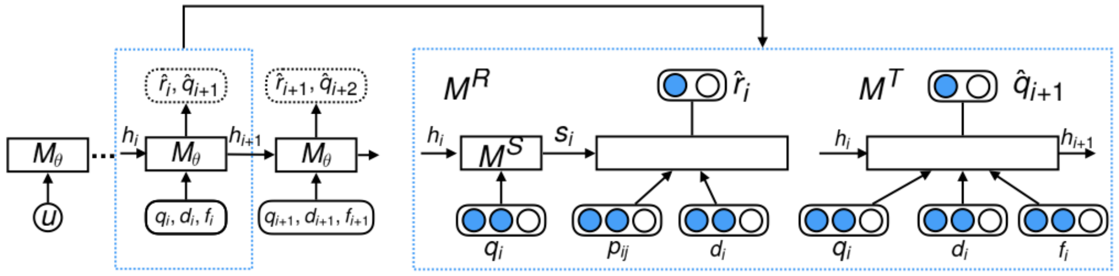


Figure 4.2: The illustrative view of neural network dynamic function.

In order to obtain effective use of offline data, we use a dynamic model to mimic the environment, i.e., the user behavior. In search story recommendation problem, user behaviors include various feedbacks, such as click on search story, click on products, and which products to click and next queries given user history. Compared with conventional environment model for recommender system [69], here only click on products is directly related to the reward function. The other two feedbacks are modeled in order to help the reward function learn the correlation between search story and products, as required by the cross-channel property of the problem. For example, without the knowledge of whether a user clicks a search story, we cannot accurately predict its influence on her future click on products. Moreover, unlike common recommender systems, here user behaviors also include user query as a part of the observation at each time step. Hence the dynamic model should predict next observation in addition to various feedbacks.

We parameterize the dynamic model M_θ as a neural network function and thus θ represents the weights of neural networks. As illustrated in Figure 4.2, our dynamic model consists of two units: a reward model M^R and a transition model M^T. The transition model M^T updates the user's hidden feature h_i to h_{i+1} and predicts the next query q_{i+1}, based on the query q_i, the search story d_i, the feedback f_i, and user's hidden feature h_i as inputs. The user's hidden feature h_i is the hidden state of recurrently applying M^T to the user search sessions until timestamp i and the initial user's hidden feature h_0 is determined by the user profile u.

The reward model can be intuitively interpreted as a click-through prediction model (`CTR` model). The inputs are the user's hidden feature h_i, the query q_i, the product item p_{ij}, and the search story d_i, whereas the output is the reward $\hat{r}_i$. The user state s_i is calculated by the state module M^S as the concatenation of h_i and q_i, representing the user intent. Finally, s_i is combined with d_i as inputs to the core submodule to predict the reward $\hat{r}_i$.

4.5.2 Transition Model

We outline the detailed architecture of transition model on the left side of Figure 4.3.

4.5.2.1 Featurization

The hidden feature h_0 is represented as a user vector constructed from both user's long-term profile and real-time profile. Regarding the user search session e_i, as defined in Definition 4.1, the user search session e_i consists of the query q_i, the story d, and the feedback $\mathcal{I}$. For each query, as shown in Figure 4.3, we represent it as an aggregated vector of its token embeddings (yellow boxes). For each story, we first represent it as raw tokens plus dense human crafted features. The raw tokens were obtained from both the title/description of story itself as well as those of product items within the story d. The raw tokens were fed into the embedding layer (shared with query embedding) and transformed into an aggregated vector of token embeddings (red boxes). The aggregated embedding vector (red box) was concatenated with dense vectors (pink box) as the final representation of the story d.

The feedback $\mathcal{I}$ is represented as the concatenation of two one-hot encoding session-level search story/product item engagement binary indicator vectors (green boxes) and the aggregated vector of token embeddings from user engaged product items (light blue boxes).

4.5.2.2 Layers of Model

The transition model is empowered with a traditional encoding-decoding architecture using the gated recurrent unit (GRU). The inputs are the concatenation of feature vectors of story, query and feedbacks as well as the hidden state h_i. The output is the feature representation of predicted next query q_{i+1}.

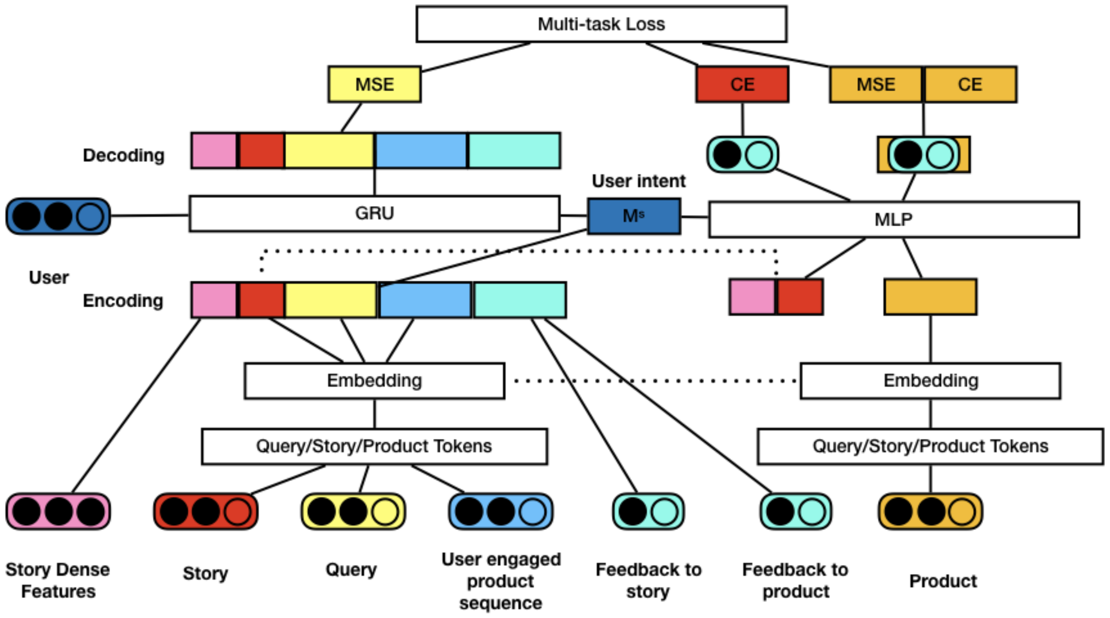


Figure 4.3: The architecture of implemented RNN dynamic model. Colors are used to distinguish different types of objects. Components which are connected by dotted line denote shared module across transition model and reward model (best viewed in color).

4.5.2.3 Loss Functions

We simply use the mean square error `MSE` between the predicted feature vector of query and the ground truth feature vector of query as the loss function for the transition model.

$$\mathcal{L}_T = \text{MSE}(\hat{q}_{i+1}, q_{i+1}), \tag{4.2}$$

where $\text{MSE}(\hat{y}, y) = ||\hat{y} - y||_2^2$.

4.5.3 Reward Model

The architecture of reward model is outlined on the right side of Figure 4.3.

4.5.3.1 Featurization

The featurization of search story d is the same as that in the transition model. For the product item p_{ij}, similar to story, we represent it as an aggregated vector of token embeddings (orange box). The user intent s_i (the dark blue box), was featurized as a hidden representation, which is learned by the state submodule M_s. M_s takes the input of hidden history h_i (shared with the transition model) and observed query q_i (same featurization as the transition model, yellow box) and outputs the user state s_i.

4.5.3.2 Layers of Model

We use a multilayer perceptron (MLP) network, which takes the input of user, search story, and product items, and predicts the feedback for search story and product items. The output layer is formulated as a classification layer for search story feedback prediction and a combination of classification and regression layer for product item feedback prediction.

4.5.3.3 Loss function

We use the cross entropy loss (CE) for the classification layer and use the conditional square error (CSE) for the regression layer. Specifically, assume that the ground truth feedback label for search story and product item, and the ground truth product representation, are $y_d/y_p/y_{rp}$, respectively, and the predicted feedback label for search story and product item and the product representation is $\hat{y}_d/\hat{y}_p/\hat{y}_{rp}$, respectively, then, the loss function is defined as:

$$\begin{aligned}\mathcal{L}_D &= \mathrm{CE}(\hat{y}_d, y_d),\\ \mathcal{L}_P &= \mathrm{CE}(\hat{y}_p, y_p),\\ \mathcal{L}_{P_l} &= \mathrm{CSE}(\hat{y}_{rp}, y_{rp}|y_p),\end{aligned} \tag{4.3}$$

where the cross entropy loss CE is defined as: $\mathrm{CE}(\hat{y}, y) = -y\log\hat{y} - (1-y)\log(1-\hat{y})$ and the conditional square error CSE is defined as: $\mathrm{CSE}(\hat{y}_{rp}, y_{rp}|y_p) = y_p||\hat{y}_{rp} - y_{rp}||_2^2$.

4.5.4 Dynamic Model Training

Given the logging data $\mathcal{D}_{log}$, we thus train the dynamic model by optimizing the following loss function:

$$\mathcal{L}_M = w_T\mathcal{L}_T + w_D\mathcal{L}_D + w_P\mathcal{L}_P + w_{P_l}\mathcal{L}_{P_l} \tag{4.4}$$

where w is the coefficient that is proportional to the contribution of each loss function. For ease of presentation, we use $(M_\theta^R, M_\theta^T) = \text{Dynamic\_Model\_Training}(\mathcal{D}_{Log})$ to denote the procedure of training the dynamic model with the architecture shown in Figure 4.3.

4.6 Controller Architecture

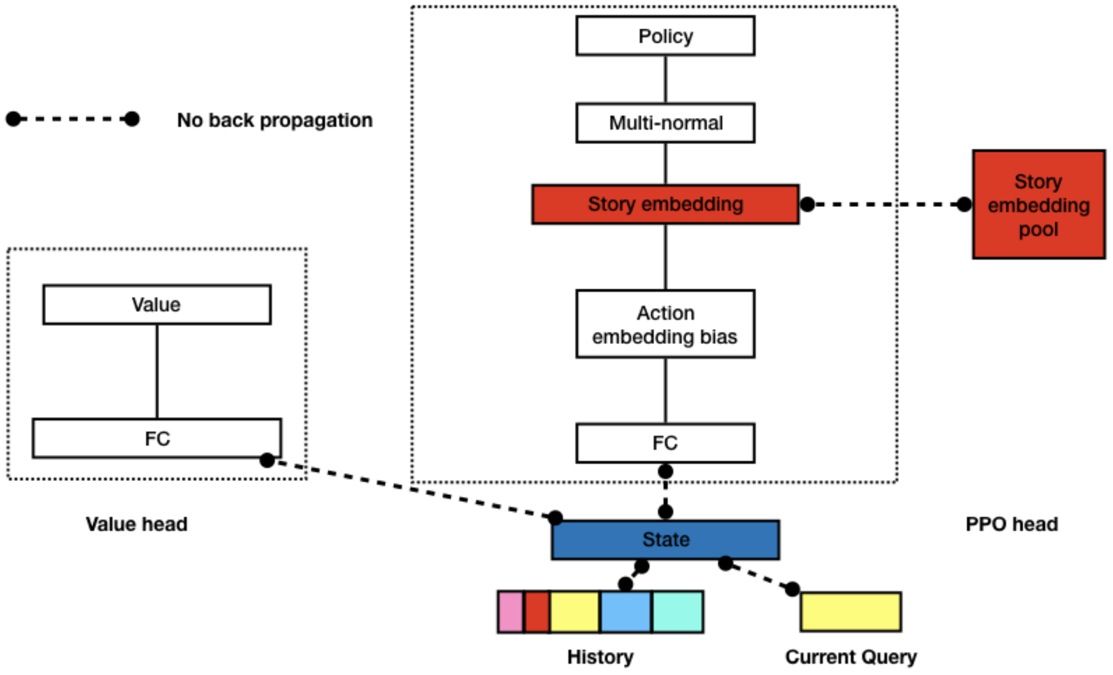


Figure 4.4: Network structure of reinforcement learning controller (best viewed in color).

Our reinforcement learning controller is designed based on the actor-critic architecture [83]. Specifically, the controller is a multi-head neural network, which is used as the function approximator for choosing the best story from a story embedding pool. Figure 4.4 illustrates our network structure of reinforcement learning controller, which consists of the state-value head (i.e., critic network) and policy head (i.e., actor network) with the shared input of state representation, the user hidden feature h_i from the transition model M^T. The details are presented as follows.

4.6.1 Critic Network

As shown in Figure 4.4, the value network is jointly learned with the policy network, where the input is the user hidden feature h_t from the transition model M^T, representing the state s_t, and the output is the $\mathbb{V}_\pi$ value $\mathbb{V}_\pi(s_t)$ of state s_t under policy π. Without ambiguity, we use $\mathbb{V}$, omitting the policy subscription. Our value network uses a neural network to learn the value function $\mathbb{V}$ with parameter ϑ. Specifically, the ϑ is updated by the gradient descent optimizer with the following loss function:

$$\begin{aligned} \mathcal{L}_{\vartheta}^{\mathbb{V}}(\pi_\Theta) &= \text{MSE}(\mathbb{V}_\vartheta(s_t), \mathbb{V}^{\texttt{target}}(s_t)) \\ \mathbb{V}^{\texttt{target}}(s_t) &= r(s_t, a_t) + \gamma \mathbb{V}(s_{t+1}) \end{aligned} \tag{4.5}$$

The updated formula of the parameter ϑ with regard to Equation 4.5 is the stochastic version of the Bellman equation.

4.6.2 Actor Network

The actor network takes the state representation vector as input. Through one fully-connected (FC) layer, it is mapped to a selection vector of the dimension of story features. The stories (actions), are scored as the inner product of the selection vector and the story features. The action is selected by the probability as the scores normalized by softmax function.

Our policy optimization is designed based on the state-of-the-art Proximal Policy Optimization (PPO) controller [96]. It is a safe policy iteration method, whose policy is guaranteed to be improved for each learning step as long as policy update is small enough. This combined with the imitation learning (Section 4.7) improve **stability**. Our policy π is again parametrized as a neural network function with parameter Θ (in order to distinguish with the dynamic model parameter θ). The architecture of our policy neural network, is shown on the right side of Figure 4.4.

In the controller reinforcement learning procedure, it learns the policy π by maximizing the accumulated state value of a policy averaging over the state distribution of a search session history:

$$\begin{aligned} \pi = \quad & \arg\max_{\Theta} \mathcal{L}_R(\pi_\Theta) \\ \text{subject to} \quad & \mathcal{L}_R(\pi_\Theta) = \sum_u \sum_{e \in E_u} [\mathcal{L}_e^{\texttt{clip}}(\pi_\Theta) + w_{\mathbb{H}_l} \mathbb{H}(\pi_\Theta(\cdot|s))], \end{aligned} \tag{4.6}$$

where $\mathcal{L}_e^{\texttt{clip}} = \min\{\frac{\pi_\Theta(\cdot|s)}{\pi_{\texttt{old}}(\cdot|s)}\widehat{A}, \texttt{clip}(\frac{\pi_\Theta(\cdot|s)}{\pi_{\texttt{old}}(\cdot|s)}, 1-\epsilon, 1+\epsilon)\widehat{A}\}$ and $\widehat{A}$ is the estimated advantage function defined as $\widehat{A}_t = r_t + \gamma\mathbb{V}(s_{t+1}) - \mathbb{V}(s_t)$. The advantage function estimator here is equivalent with setting $\lambda = 0$ in the GAE estimate for advantage used in the original PPO paper [96] as the experiment suggests no better performance with a non-zero λ value. $\mathbb{H}$ is the entropy of the policy π_Θ given state s, and $w_{\mathbb{H}_l}$ is the weight.

4.7 Policy Learning

With the design of both dynamic model and controller network, we introduce imitation learning, safe learning on offline data, and imagination learning to learn a policy (i.e., tune the parameters of the actor network π_Θ) that is effective in using offline data and stable at the same time.

4.7.1 Imitation Learning

In our search recommendation task, and most other real-world decision-making problems (e.g., finance and health-care), we have access to the logging data of the

Algorithm 4.2 `Controller_Learning`($\mathcal{D}$, $\mathbb{V}_\vartheta$, π_Θ)

```
Input: Data $\mathcal{D}$, current actor network $\pi_\Theta$
       and the critic network $\mathbb{V}_\vartheta$
Output: The updated actor network $\pi_\Theta$ and
       critic network $\mathbb{V}_\vartheta$
1: Repeat sampling a mini-batch $bs$ of search sessions
       from $\mathcal{D}$
2:         update the critic network $\mathbb{V}_\vartheta$ minimizing eq. 4.5
3:         update the actor network $\pi_\Theta$ minimizing eq. 4.6
4: return $\mathbb{V}_\vartheta$, $\pi_\Theta$
```

system being operated by its previous controller, but we do not have access to an accurate simulator of the system. The goal of the imitation learning is thus to learn to imitate the previous controller with a fixed policy π_0. Specifically, we learn the policy π_0, parameterized by the actor network π_Θ from $\mathcal{D}_{Log}$ by optimizing the likelihood of the actions chosen. Formally, imitation learning can be formulated as the following optimization task:

$$\pi_0 = \arg\min_{\Theta} \mathcal{L}_I(\pi_\Theta), \tag{4.7}$$

where $\mathcal{L}_I(\pi_\Theta)$ is the likelihood function of observing actions in $\mathcal{D}_{Log}$ given the policy π_Θ, together with an entropy penalty,

$$\mathcal{L}_I = -\sum_{u} \sum_{e \in E_u} \log(\pi_\Theta(a|s)) - w_{\mathbb{H}_I} \mathbb{H}(\pi_\Theta(\cdot|s)), \tag{4.8}$$

where $w_{\mathbb{H}_I}$ is the weight for the entropy regularizer $\mathbb{H}$.

The imitated policy π_Θ^0 works as several roles. First, we use it as the initialization of the actor network for further policy learning, which keeps the initial policy close to the logging polity. Second, it is used as the logging policy, π_{old} in Equation 4.6, in the subsequent safe learning on offline data (Section 4.7.2). Finally, importance sampling offline evaluation uses the imitated policy as an estimation of the logging policy.

4.7.2 Safe Policy Learning on Offline data

We propose safe policy learning on offline data as the first step to improve the policy. It follows Algorithm 4.2, with data as offline data $\mathcal{D} = \mathcal{D}_{Log}$ and the current actor network as the imitated policy $\pi_\Theta = \pi_\Theta^0$ in `Controller_learning`. Though it looks similar to normal actor-critic learning, the use of imitated policy makes it safe and unbiased on offline data.

First, the imitated policy π_0 is used as π_{old} in Equation 4.6. As the imitated policy is an estimation of the logging policy of $\mathcal{D}_{Log}$, we obtain an unbiased estimation of the evaluation of the updated policy π_Θ, $\frac{\pi_\Theta(\cdot|s)}{\pi_{\texttt{old}}(\cdot|s)}\widehat{A}$ in Equation 4.6, from offline data [68]. Second, with the current policy input $\pi_\Theta = \pi_0$ in `Controller_learning`, we initialize the actor network by the imitated policy π_Θ^0. Combined with the above choice of π_{old}, such PPO learning (Equation 4.6) ensures improvement with limited policy difference [96]. This distinguishes our method as safe policy learning from off-policy correction [68].

4.7.3 Imagination Learning

It is not effective in using offline data that only model-free reinforcement learning method is applied on the data, especially the previous safe policy learning (Section 4.7.2) is only one iteration of the PPO algorithm [96]. The goal of controller imagination is thus to use the trained dynamic model to further improve the actor network.

Specifically, we use randomly selected sessions in $\mathcal{D}_{Log}$ as starting sessions, from each of which, the dynamic model (M_θ^R, M_θ^T) and the current actor network are applied to rollout T_{img} fictional search sessions, stored in $\mathcal{D}_{RL}$. The imagined data $\mathcal{D}_{RL}$ is then used in the controller reinforcement learning (Section 4.6) to further tune the actor network. Generally, it is similar to the original PPO controller learning [96], except that the real environment is replaced by the dynamic model here.

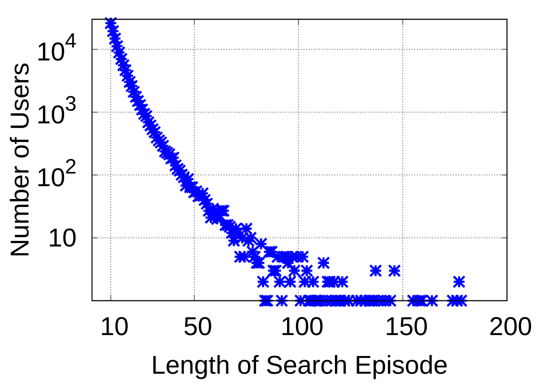
The imagination learning uses the learned dynamic model to extrapolate offline data to updated policy. Hence it uses offline data more effectively. However, with the selection bias caused by the difference between the updated policy and the logging policy, in addition to the learning error of the dynamic model, we should limit imagination learning considering the trade-off between data effectiveness and stability.

4.8 Experimental Validation

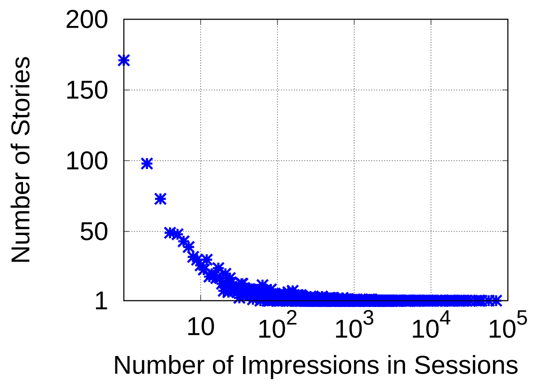
4.8.1 Experimental Setup

4.8.1.1 Dataset

We evaluate our methods on a dataset collected between Apr 2018 and Jul 2018 from *JD.com* [69]. We sampled all search sessions that are related to a category “women dress" and filtered out search episodes with only a few sessions or a huge number of sessions. Our dataset is carefully pre-processed and anonymized. The distributions of the episode length and the number of search sessions in which each



(a)



(b)

Figure 4.5: Histograms of (a) episode length and (b) story impression frequency. Both follow a power-law distribution.

# users	# stories	# products	# sessions
122,886	2,185	304,780	1,842,879

Table 4.2: Statistics of dataset.

search story appears are shown in Fig. 4.5a and 4.5b. Other statistics of our dataset is summarized in Table 4.2. We randomly divide the dataset into 5 folds by users. Hence each fold of the dataset contains equal number of complete search episodes. We conducted 5-fold cross-validation experiments with one random fold as testing data for each experiment.

The processed feature dimensions are summarized as follows. Each query is represented as an aggregation of 200 dimensional word embedding vectors of segmented query words. Each product is represented as an aggregation of 200 dimensional word embedding vectors of words from product titles. For each story, it is featured as a concatenation of 200 dimensional word embedding vectors of words from story titles, 200 dimensional word embedding vectors of title words from the products embedded within a story, and 13 human crafted features of a story.

4.8.1.2 Baseline Methods

We compare the proposed method DRESS as described in Algorithm 4.1 with the following baseline methods:

1. ORIGIN: This is the state-of-the-art implementation of a search story recommendation, that results in the offline data, currently being used by *JD.com*. It is parameterized as the imitated policy π_0.

2. DNNC (Deep Neural Network Classifier): Without considering the cross-channel effect, this method is trained to recommend a search story that is likely to be clicked, given the story feedback data. To be a fair comparison, DNNC uses the architecture with the actor network and is initialized with the imitation policy π^0_Θ, same as DRESS.

3. DNNC-p: Deep Neural Network Classifier for Product click. It is a supervised learning method similar to DNNC baseline, but with target replaced with product click, which considers the cross-channel effect; it is similar to DRESS-m in terms of myopic prediction, but different in optimization.

4. DQN [74]: Deep Q network is shown to be successful in many different applications [77, 97, 98]. We implemented the double q-learning version [99]. As implemented in FeedRec [69], it learns by interacting the dynamic model same as DRESS, because DQN is an online learning method.

5. SRL [71]: Supervised reinforcement learning combines imitation learning and actor-critic policy learning. It is a state-of-the-art method in dynamic treatment recommendation, which emphasizes stability due to the high risk of the problem. We use stochastic policy gradient instead of deterministic policy gradient in accordance with the discrete action space here.

6. OPC [68]: Off-policy correction method uses imitated policy to reweight the sampling from offline data. However, the actor network is not initialized by the imitated policy and there is not model-based imagination learning.

7. DPG-FBE [93]: DPG-FBE is a state-of-the-art method that does imagination learning with an environment model to learn from offline data. We use stochastic policy gradient instead of deterministic policy gradient in accordance with the discrete action space here.

8. DRESS-m: This is the myopic version of DRESS that only considers immediate short-term reward, which is implemented by setting $\gamma = 0$.

9. DRESS-s: This is the simplified version of DRESS with the controller imagination module (Section.4.7.3) removed.

10. DRESS: The full version of the method proposed in this work.

4.8.1.3 Simulation Study

It is risky to evaluate the learned policy on a real-life system. Following recent works [68, 93], we use an environment simulator to evaluate different methods. The environment simulator is the same with the dynamic model in architecture but is trained by datasets different from method learning. We use $CTR_{simulate}$, the click through rate in simulation experiment as evaluation metric.

4.8.1.4 Evaluation Metric

The goal of a search story recommendation is to facilitate users during the search of products. Therefore, we use search session based user feedback on products as the main performance measure. In particular, we use the percentage of search sessions in which users have clicked a product, CTR (Click Through Rate):

$$\text{CTR} = \frac{\sum_{e \in E} clk_e}{|E|} \tag{4.9}$$

where clk_e is a binary indicator whether a user clicked a product in a search session e, which is the same as the reward r_e in the RL framework. Similarly, we also use CVR (Conversion Rate):

$$\text{CVR} = \frac{\sum_{e \in E} ord_e}{|E|} \tag{4.10}$$

where ord_e is a binary indicator whether a user ordered a product in a search session e.

It is risky to evaluate the learned policy on a real-life system. Therefore, we use a statistical estimate method, Truncated Weighted Importance Sampling (TWIS), similar to NCIS [100] used in [69], to estimate the performance from the offline test data as follows:

$$\widehat{R(\pi)} = \frac{\sum_e \sum_{t=T_e-H}^{T_e} r_t \prod_{i=T_e-H}^{T_e} \frac{\pi_i}{b_i}}{\sum_e \sum_{t=T_e-H}^{T_e} \prod_{i=T_e-H}^{T_e} \frac{\pi_i}{b_i}} \tag{4.11}$$

where e is an episode, H is the horizon for the latest sessions to use per episode,

r_t can be clk_t (ord_t) so that $\widehat{R(\pi)}$ is the estimate of CTR (CVR, respectively), $\pi_i = \pi(a_i|s_i)$ is the probability of the observed action given by the evaluated policy π and $b_i = b(a_i|s_i)$ by the logging policy. With only offline data, the imitated policy π_Θ^0 is used as the logging policy. This evaluation metric is invariant toward the arbitrary constant scale of b_i and of π_i. The truncated setting encourages the equal importance of users with the episodes of different lengths. We use horizon $H = 15$ in this work.

The use of the imitated policy here can be justified from two aspects. On one hand, the imitated policy π_0 is an estimation of the logging policy of the offline data. Therefore TWIS provides an unbiased estimation of the performance of the learned policy π. On the other hand, TWIS is a valuable performance measure even if the imitated policy π_0 is not a perfect estimation of the logging policy. Following the idea of importance sampling, $\widehat{R(\pi)}$ is the estimate of CTR (CVR) of the weighted policy $\pi^w = b^* \cdot \frac{\pi}{b}$, where b^* is the true logging policy, which is valid when π and b are close enough. When the true logging policy b^* is different from b, TWIS is an evaluation of the policy factor $\frac{\pi}{b}$ to be applied to the existing policy b^*. In an application, the policy ratio $\frac{\pi}{b}$ can play the role as reinforcement learning complements b^* and validly evaluated by $\widehat{R(\pi)}$. In this sense, the imitated policy π_0 maximizes the denominator b so that minimizes the variance caused by the policy factor $\frac{\pi}{b}$.

Both interpretations require small difference between the learned policy π and the imitated policy π_0 to bound the variance of TWIS [70]. Hence we only use TWIS to evaluate methods with policy close to the imitated policy.

4.8.1.5 Hyperparameters

Most hyperparameters are tuned using the validation set for each experiment. For reproducibility of our experimental results, values of some hyperparameters are summarized in Table. 4.3. More implementation details can be found in Appendix. 4.10

4.8.2 Empirical Results

In this section, we conduct different groups of experiments to empirically validate the proposed approaches. Specifically, we aim to answer the following questions:

hyperparameters	setting
discount factor γ (Eq.4.1)	0.7
transition loss weight w_T (Eq.4.4)	1.0
story loss weight w_D (Eq.4.4)	1.0
product CE loss weight w_P (Eq.4.4)	1.0
product CSE loss weight w_{P_l} (Eq.4.4)	1.0
Entropy weight for controller learning $w_{\mathbb{H}_l}$ (Eq.4.6)	0.01
Entropy weight for controller imitation $w_{\mathbb{H}_I}$ (Eq.4.8)	0.0001
clipping factor ϵ (Eq.4.6)	0.2

Table 4.3: Hyperparameters.

method	$CTR_{simulate}$
DNNC	0.579
DNNC-p	0.585
DQN	0.570
SRL	0.590
OPC	0.560
DPG-FBE	0.586
DRESS-m	0.565
DRESS-s	0.607
DRESS	**0.614**

Table 4.4: Performance comparison in simulation study

(1) Compared with competing methods, how does the proposed DRESS effectively use offline data evaluated by the simulator study? (2) How does the proposed DRESS ensure stability compared with other RL methods? (3) Among methods whose policy is close to the imitated policy, how does DRESS perform in terms of the unbiased estimation TWIS?

Question 4.1. ***Effectiveness****: Compared with competing methods, how does the proposed* DRESS *effectively use offline data evaluated by the simulator study?*

We report the performance of different methods in simulation study in Table 4.4. All results are the average of around 700,000 rollout episodes of length 50 starting from real search sessions. We observe that the proposed DRESS performs best compared with competing methods and its variants. More specifically, DNNC performs worse than DRESS, showing the advantage of considering the cross-channel effect. On the other hand, the results that DRESS-m is worse than DRESS showing the importance of the long-term effect. These results justifies the proposal of an RL framework for the problem.

However, RL methods are not always better. The performance of DQN and OPC is even worse than DNNC. DNNC uses imitated policy as initialization for the actor network and its learning objective, story clicks, is similar to that of the logging policy. Hence it is close to the logging policy. On the other hand, DQN learns by interacting with the dynamic model, and OPC is not initialized with the imitated policy. Their learned policies are far from the logging policy. Without stability, the performance of learning from offline data cannot be guaranteed. The comparison between OPC and DRESS-s supports the superiority of the proposed safe policy learning on offline data.

Comparing DRESS and its variant DRESS-s or SRL, we observe the helpfulness in data effectiveness from the imagination learning with the dynamic model. The reason why SRL performs worse than DRESS-s will be provided after we show the stability results.

Question 4.2. ***Stability****: How does the proposed* DRESS *ensure stability compared with other RL methods?*

policy	ratio	D_{TV}	D_{KL}
DNNC	0.038	0.018	0.002
DNNC-p	0.025	0.012	0.0008
unif	6.044	0.967	5.959
DQN	6.024	0.967	5.947
SRL	**0.008**	**0.004**	**8e-5**
OPC	7.885	0.979	7.886
DPG-FBE	6.144	0.969	6.057
DRESS-m	0.058	0.028	0.003
DRESS-s	0.049	0.024	0.002
DRESS	0.083	0.041	0.007

Table 4.5: Policy difference compared with the imitation policy.

We check the **stability** of the proposed approach and other RL methods by the difference toward the imitated policy. The imitation policy is an estimate of the logging policy currently in production that generates offline data. We expect the learned policy to be close to the logging policy to ensure the stability of an online system. We use three measures of policy distribution difference. For session i,

1. absolute log probability ratio:

$$ratio_i = \left| \log \left(\frac{\pi(a_i|s_i)}{b(a_i|s_i)} \right) \right|;$$

2. total variation divergence [101]:

$$D_{\mathrm{TV}}(b||\pi)_i = \frac{1}{2} \sum_{a'} |\pi(a'|s_i) - b(a'|s_i)|$$

3. KL-divergence:

$$D_{\mathrm{KL}}(b||\pi)_i = \sum_{a'} b(a'|s_i) \log \left(\frac{b(a'|s_i)}{\pi(a'|s_i)} \right).$$

We calculate the averages of each difference measure over sessions in test data.

Results are shown in Table 4.5. First, we observe a clear gap between two groups of methods: large policy difference, *unif*, DQN, OPC and DPG-FBE, and small policy difference, SRL, DRESS-m, DRESS-s and DRESS. We find that the policy difference of most state-of-the-art RL methods is comparable with the uniform *unif* policy. The reason is that those methods do not consider stability, can result in learned policy very far from the logging policy of the offline data. On the other hand, SRL shows the smallest policy difference in all measures. This is as expected because SRL further includes imitation loss into the actor-critic loss function comparing to DRESS where the actor network is only initialized by the imitation learning. However, stability is not the sole objective of the solution. The strong stability penalty in SRL may limit its effectiveness in policy improvement. In order to better show the trade-off between data effectiveness and stability, we use TWIS to get unbiased evaluation of the performance of methods in the small policy difference group.

Question 4.3. ***TWIS evaluation****: Among methods whose policy is close to the imitated policy, how does* DRESS *perform in terms of the unbiased estimation TWIS?*

method	improvement $CTR\%$	improvement $CVR\%$
DNNC	1.380	-0.437
DNNC-p	3.295	0.909
SRL	0.354	0.671
DRESS-m	3.344*	15.463
DRESS-s	3.194*	3.507
DRESS	**3.952****	**19.775**

Table 4.6: CTR and CVR: shown as the improvement percentage over ORIGIN. $*$ indicates statistical significance (p-value < 0.05)

We show TWIS evaluation of methods with small policy difference to the imitated policy in Table 4.6 as improvement percentage from the logging policy, ORIGIN[1]. We see that only DRESS and its two variants obtain significant improvement over the logging policy in CTR. Both supervised learning methods DNNC and DNNC-p cannot obtain significant improvement, which shows the necessity of using reinforcement learning framework. Further, DNNC-p shows better performance than DNNC, which showcases the importance of the cross-channel effect. Large variance is observed in CVR estimation due to the small number of orders compared with clicks. Comparing SRL and DRESS, taking into account the policy difference in Table 4.5, we see that the emphasis on stability of SRL prevents it from obtaining significant policy improvement. On the other hand, comparing DRESS-s and DRESS in both Table 4.5 and Table 4.6, we see DRESS-s is more stable but achieve less policy improvement, while DRESS trades some stability for more performance gain by adding imagination learning to extrapolate the offline data. The stability and data effectiveness trade-off can be tuned by controlling the extent of imagination learning.

4.9 Conclusion

Deep reinforcement learning has been successfully used as a powerful method to capture a wide variety of non-trivial user behavior on online platforms (e.g., news feed recommendation, e-commerce search). In this work, following these successes, we applied the reinforcement learning framework to the challenging problem of cross-channel search story recommendation. Facing the offline data and

[1]We cannot provide the absolute values due to the information protection rule of JD.com

stability challenges, we proposed a unified deep learning architecture employing both imitation learning and reinforcement learning, model-based and model-free. Comprehensive empirical validation indicates that our proposed framework DRESS is effective in improving a conversion rate on real-world data sets from *JD.com*.

4.10 Implementation Details

4.10.1 Data Processing

The input features for both the dynamic model and the controller involve user histories $[(q_i, d_i, f_i) : \forall i \in [t - L_h, t - 1]]$ and the input of the target search sessions. The maximal history length is set to $L_h = 20$, around the middle number of the lengths of search episodes (Figure. 4.5a), which means that we exclude the influence of a search story from a search session more than L_h sessions earlier on users' current behavior.

The target search sessions are a single session for `Dynamic_ Model_Training` and `Controller_Imitation`, but two subsequent sessions for `Controller_Learning`. In order to use the data more effectively and reduce the influence of users with short episodes, we sample target search sessions uniformly in terms of all search sessions rather than one from each user.

4.10.2 Model Implementation

We randomly divide the dataset into 5 folds by users. Within each 4 folds of training data, half-fold is used as validation. The hyperparameters are tuned separately by learning stages.

The dynamic model is tuned by the four losses in Equation. 4.4. The size of the GRU hidden layers (h_i in Figure. 4.2) is 128. The sizes of the decoding hidden layers (Figure. 4.3) are 64 for both next query prediction $\hat{q}_{i+1}$ (Equation. 4.2) and the feedbacks for product items, $\hat{y}_p$ and $\hat{y}_{rp}$ (Equation. 4.3), and 128 for the feedback for stories $\hat{y}_d$ (Equation. 4.3). The learning rate and batch size are $1e - 4$ and 512, respectively.

The `Controller_Imitation` is tuned by the log likelihood loss in the imitation loss (Equation. 4.8). The size of the FC layer of the actor network (Figure. 4.4) is 128. The learning rate and batch size are $1e-4$ and 512, respectively. Note that the same actor network will be used in `Controller_Learning`. We use `Controller_Imitation` to select the hyperparameter because of the higher stability of supervised learning.

The `Controller_Learning` in safe policy learning on offline data and imagination learning is tuned by the TWIS estimated CTR (Equation. 4.11). The size of the FC layer of the critic network (Figure. 4.4) is 64. We use a cloned critic network [74] to improve the stability of the Bellman equation (Equation. 4.5). The update delay is 5 mini-batches. The batch size is set to 64.

In safe policy learning on offline data, the learning rates for the critic network and the actor network are both $1e-6$ in safe policy learning on offline data. The critic network is fit first for 2 epochs, because the logging policy for the training data does not change in this stage. After that the actor network trained for 1 epoch.

In imagination learning, the learning rate is $1e-7$ for the critic network and $1e-5$ for the actor network, respectively. In each iteration of imagination learning (Step 6 and 7 in Algorithm. 4.1), queries of 0.005 portion of search sessions in $\mathcal{D}_{Log}$, together with their history sessions, are sampled as starting points of the imagination. In order to control the bias accumulation, the imagination is limited to 2 sessions since the starting sessions. The critic network and actor network are updated for 1 epoch on the $\mathcal{D}_{RL}$ from the imagination. The imagination learning is repeated for 10 iterations.

The optimization is done with the Adam optimizer [102]. The tanh function is used for non-linear activation.

Chapter 5 | Conclusions and Future Work: What Users Can Do?

This chapter closes this work by asking, what users can do to combat the influence of systems powered by machine learning methods unconsciously. More specifically, the interplay between recommender system and users is used as the playground.

5.1 Conclusions and Limitations

In previous chapters, different perspectives about user behavior is studied. As an example of the representation of user behavior, the emotional reactions of users towards online content are studied within the label ranking framework, in Chap. 2. In the meanwhile, the imbalance challenge found in real world data calls for new designs for label ranking methods. As an instance of the factors that influence user behavior, the effect of news channels on emotional reactions of users towards online content is studied within the multitask learning framework, in Chap. 3. The observation in the real world problem inspires new multitask learning architecture. Finally, the recommendation of search story to influence users' search behavior on e-commerce websites is studied as an example to show how to influence user behavior, in Chap. 4. A reinforcement learning framework is built in accordance with the two challenges, offline data and stability.

However, all the above studies assume user models that do not change when they interact with online systems (e.g., recommender system). For example, in Chap. 2, the users' emotional reaction patterns given the content they read are fixed; in Chap. 3, the effect of news channels on users' emotional reactions is fixed;

and in Chap. 4, though it is claimed to influence user behavior, it is still assumed that the influence of the search stories seen by users on their search behavior is not changed. Such assumption is on one hand unrealistic in the long term when users are influenced by some online systems, and on the other hand underestimating the power of users in deciding their behaviors.

5.2 Where is the Influence

> "Assuming we have a recommender system that constantly provides us recommendations, the question arises whether that resembles a 'Big Brother' who guides us. ... Will the result be a problem for society if we train ourselves to constantly listen to recommendations?" – [103]

Many works of recommendation in machine learning will describe the role of a recommendation system as helping users find their desired items based on their previous behavior given a large item pool. I call this role the first principle of recommendation.

However it is said, most works of recommendation in machine learning do not respect this principle in their design. In conventional works, including recommendation with only rating matrix, content-based recommendation and sequential recommendation, the task is to predict future behavior of users. For example, in sequential recommendation, a sequence of items a users clicked as input, it is to predict future items the user will click. If users will fully follow their desires to move, such prediction does follow the first principle. However users behavior is shaped by the recommendation systems already because users can only access items recommended by the system rather than the large item pool. It is shown to result in sub-optimal user behavior [104]. Advanced recommendation works using multi-armed bandits to correct the selection bias. However, they are only applicable to cold-start problem.

5.3 Users' Perspective

Almost all the works in recommendation take the view of recommendation system, i.e., how to design a better recommendation system, while few take users' perspective, i.e., what users can do to get better recommendation results. It is due to the subjectivity of users, which makes such works often limited in user study [105, 106].

It is worth noticing that users are modeled either implicitly or explicitly in recommendation works. The most popular, if not the only one, is the user model where each user has an interest vector, and the interests of items to the user are determined by the distance between the item embedding vector and the interest vector. For example, with matrix factorization method [107], the input rating matrix is factorized into a user matrix and an item matrix; with GRU4REC, a sequential recommendation model, the user interest vector is the aggregation of users' previous interacted items by a RNN structure. More explicitly, methods requiring online interactions with users, such as multi-armed bandit [108] and reinforcement learning [68, 90] often involves a synthetic user model where user interest vectors and item embedding vectors are randomly generated.

The strategy that users follow their interests honestly when interacting with a recommendation system works well if the recommendation system is perfect that can infer users' true interests from the behavior data. However, it is hard in practice, if not impossible, due to the selection bias above and also inference error that is inevitable. Hence given an imperfect recommendation system, should users still play the honest strategy described above? This defines the question "what users can do".

5.4 Discussion

All studies in this work except this chapter are taken from the perspective of systems that users interact with. For examples, how can a system better predict users' emotional reactions when they are represented as ranking of labels; how a machine learning method can find from data the influence of news channels on users emotional reactions besides post content using a multi-task learning framework; how can a recommendation system influence users' searching behavior by recommending the best search stories? All the above assumed that the users' behavior is fixed. On the other hand, using recommendation problem as an example, the assumptions of users can be over-simplified. Therefore, before applying more complicated and larger models to systems, we may need a better understanding of users whose benefits are the ultimate goal of using those systems.

Bibliography

[1] Zhang, J. J. and D. Lee (2018) "ROAR: Robust Label Ranking for Social Emotion Mining," in *AAAI*.

[2] Jia, Y., Z. Chen, and S. Yu (2009) "Reader emotion classification of news headlines," in *NLP-KE 2009.*, IEEE, pp. 1–6.

[3] Lin, K. H.-Y. and H.-H. Chen (2008) "Ranking reader emotions using pairwise loss minimization and emotional distribution regression," in *EMNLP*, Association for Computational Linguistics, pp. 136–144.

[4] Lin, K. H.-Y., C. Yang, and H.-H. Chen (2008) "Emotion classification of online news articles from the reader's perspective," in *WI-IAT'08.*, vol. 1, IEEE, pp. 220–226.

[5] Tang, Y.-j. and H.-H. Chen (2011) "Emotion modeling from writer/reader perspectives using a microblog dataset," in *Proceedings of IJCNLP Workshop on sentiment analysis where AI meets psychology*, pp. 11–19.

[6] Bai, S., Y. Ning, S. Yuan, and T. Zhu (2012) "Predicting reader's emotion on Chinese web news articles," in *ICPCA-SWS*, Springer, pp. 16–27.

[7] Lei, J., Y. Rao, Q. Li, X. Quan, and L. Wenyin (2014) "Towards building a social emotion detection system for online news," *Future Generation Computer Systems*, **37**, pp. 438–448.

[8] Zhu, C., H. Zhu, Y. Ge, E. Chen, and Q. Liu (2014) "Tracking the evolution of social emotions: A time-aware topic modeling perspective," in *ICDM*, IEEE, pp. 697–706.

[9] Zhang, Y., N. Zhang, L. Si, Y. Lu, Q. Wang, and X. Yuan (2014) "Cross-domain and cross-category emotion tagging for comments of online news," in *SIGIR*, ACM, pp. 627–636.

[10] Hüllermeier, E., J. Fürnkranz, W. Cheng, and K. Brinker (2008) "Label ranking by learning pairwise preferences," *Artificial Intelligence*, **172**(16), pp. 1897–1916.

[11] Har-Peled, S., D. Roth, and D. Zimak (2002) "Constraint classification for multiclass classification and ranking," in *NIPS*.

[12] Dekel, O., C. D. Manning, and Y. Singer (2003) "Log-Linear Models for Label Ranking," in *NIPS*.

[13] Cheng, W., S. Henzgen, and E. Hüllermeier (2013) "Labelwise versus Pairwise Decomposition in Label Ranking." in *LWA*, pp. 129–136.

[14] Cheng, W., M. Rademaker, B. De Baets, and E. Hüllermeier (2010) "Predicting partial orders: ranking with abstention," in *ECML-PKDD*, Springer, pp. 215–230.

[15] Cheng, W., E. Hüllermeier, W. Waegeman, and V. Welker (2012) "Label ranking with partial abstention based on thresholded probabilistic models," in *NIPS*, pp. 2501–2509.

[16] Destercke, S. (2013) "A pairwise label ranking method with imprecise scores and partial predictions," in *ECML-PKDD*, Springer, pp. 112–127.

[17] Grbovic, M., N. Djuric, and S. Vucetic (2013) "Multi-Prototype Label Ranking with Novel Pairwise-to-Total-Rank Aggregation." in *IJCAI*.

[18] Mallows, C. L. (1957) "Non-null ranking models. I," *Biometrika*, **44**(1/2), pp. 114–130.

[19] Cheng, W., J. Hühn, and E. Hüllermeier (2009) "Decision tree and instance-based learning for label ranking," in *ICML*, ACM, pp. 161–168.

[20] Zhou, Y., Y. Liu, X.-Z. Gao, and G. Qiu (2014) "A label ranking method based on gaussian mixture model," *Knowledge-Based Systems*, **72**, pp. 108–113.

[21] Cheng, W., E. Hüllermeier, and K. J. Dembczynski (2010) "Label ranking methods based on the Plackett-Luce model," in *ICML*, pp. 215–222.

[22] Shieh, G. S. (1998) "A weighted Kendall's tau statistic," *Statistics & probability letters*, **39**(1), pp. 17–24.

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

[24] ——— (2010) "Distance-based tree models for ranking data," *Computational Statistics & Data Analysis*, **54**(6), pp. 1672–1682.

[25] Busa-Fekete, R., E. Hüllermeier, and B. Szörényi (2014) "Preference-based rank elicitation using statistical models: The case of Mallows," in *ICML*, pp. 1071–1079.

[26] Kendall, M. G. (1948) *Rank correlation methods.*, Charles Griffin & Co. Ltd., London.

[27] Spearman, C. (1904) "The proof and measurement of association between two things," *The American journal of psychology*, **15**(1), pp. 72–101.

[28] Järvelin, K. and J. Kekäläinen (2002) "Cumulated gain-based evaluation of IR techniques," *ACM Transactions on Information Systems (TOIS)*, **20**(4), pp. 422–446.

[29] He, H. and E. A. Garcia (2009) "Learning from imbalanced data," *TKDE*, **21**(9), pp. 1263–1284.

[30] Batista, G. E., R. C. Prati, and M. C. Monard (2004) "A study of the behavior of several methods for balancing machine learning training data," *ACM Sigkdd Explorations Newsletter*, **6**(1), pp. 20–29.

[31] Japkowicz, N. and S. Stephen (2002) "The class imbalance problem: A systematic study," *Intelligent data analysis*, **6**(5), pp. 429–449.

[32] Chawla, N. V., N. Japkowicz, and A. Kotcz (2004) "Editorial: special issue on learning from imbalanced data sets," *ACM Sigkdd Explorations Newsletter*, **6**(1), pp. 1–6.

[33] Weiss, G. M. (2004) "Mining with rarity: a unifying framework," *ACM Sigkdd Explorations Newsletter*, **6**(1), pp. 7–19.

[34] Maloof, M. A. (2003) "Learning when data sets are imbalanced and when costs are unequal and unknown," in *ICML-2003 workshop on learning from imbalanced data sets II*, vol. 2, pp. 2–1.

[35] Sun, Y., M. S. Kamel, and Y. Wang (2006) "Boosting for learning multiple classes with imbalanced class distribution," in *ICDM*, IEEE, pp. 592–602.

[36] de Borda, J. C. (1781) "Memoire sur les Élections au Scrutin." *Histoire de l'Academie Royale des Sciences.*

[37] Diaconis, P. and R. L. Graham (1977) "Spearman's footrule as a measure of disarray," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 262–268.

[38] Maurer, A., M. Pontil, and B. Romera-Paredes (2016) "The benefit of multitask representation learning," *JMLR*, **17**(1), pp. 2853–2884.

[39] Zhang, Y. and Q. Yang (2017) "A survey on multi-task learning," *arXiv preprint arXiv:1707.08114*.

[40] Ruder, S. (2017) "An overview of multi-task learning in deep neural networks," *arXiv preprint arXiv:1706.05098*.

[41] Kang, Z., K. Grauman, and F. Sha (2011) "Learning with Whom to Share in Multi-task Feature Learning." in *ICML*, vol. 2, p. 4.

[42] Ma, J., Z. Zhao, X. Yi, J. Chen, L. Hong, and E. H. Chi (2018) "Modeling task relationships in multi-task learning with multi-gate mixture-of-experts," in *SIGKDD*, ACM, pp. 1930–1939.

[43] Rosenbaum, C., T. Klinger, and M. Riemer (2018) "Routing networks: Adaptive selection of non-linear functions for multi-task learning," *ICLR*.

[44] Jalali, A., S. Sanghavi, C. Ruan, and P. K. Ravikumar (2010) "A Dirty Model for Multi-task Learning," in *NeurIPS* (J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, eds.), pp. 964–972.

[45] Hernández-Lobato, D., J. M. Hernández-Lobato, and Z. Ghahramani (2015) "A probabilistic model for dirty multi-task feature selection," in *ICML*, pp. 1073–1082.

[46] Wang, X., J. Bi, S. Yu, and J. Sun (2014) "On multiplicative multitask feature learning," in *NeurIPS*, pp. 2411–2419.

[47] Evgeniou, T. and M. Pontil (2004) "Regularized multi–task learning," in *SIGKDD*, ACM, pp. 109–117.

[48] Jacob, L., J.-p. Vert, and F. R. Bach (2009) "Clustered multi-task learning: A convex formulation," in *NeurIPS*, pp. 745–752.

[49] Zhou, J., J. Chen, and J. Ye (2011) "Clustered multi-task learning via alternating structure optimization," in *NeurIPS*, pp. 702–710.

[50] Zhou, Q. and Q. Zhao (2015) "Flexible clustered multi-task learning by learning representative tasks," *IEEE transactions on pattern analysis and machine intelligence*, **38**(2), pp. 266–278.

[51] Liu, S. and S. J. Pan (2017) "Adaptive Group Sparse Multi-task Learning via Trace Lasso." in *IJCAI*, pp. 2358–2364.

[52] Lee, G., E. Yang, and S. Hwang (2016) "Asymmetric multi-task learning based on task relatedness and loss," in *ICML*, pp. 230–238.

[53] Argyriou, A., T. Evgeniou, and M. Pontil (2008) "Convex multi-task feature learning," *Machine Learning*, **73**(3), pp. 243–272.

[54] Kumar, A. and H. Daume III (2012) "Learning task grouping and overlap in multi-task learning," *arXiv preprint arXiv:1206.6417*.

[55] Yang, Y. and T. Hospedales (2016) "Deep multi-task representation learning: A tensor factorisation approach," *arXiv preprint arXiv:1605.06391*.

[56] Duong, L., T. Cohn, S. Bird, and P. Cook (2015) "Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser," in *ACL*, vol. 2, pp. 845–850.

[57] Long, M., Z. Cao, J. Wang, and S. Y. Philip (2017) "Learning multiple tasks with multilinear relationship networks," in *NeurIPS*, pp. 1594–1603.

[58] Misra, I., A. Shrivastava, A. Gupta, and M. Hebert (2016) "Cross-stitch networks for multi-task learning," in *CVPR*, pp. 3994–4003.

[59] Collins, M. D. and P. Kohli (2014) "Memory bounded deep convolutional networks," *arXiv preprint arXiv:1412.1442*.

[60] Alvarez, J. M. and M. Salzmann (2016) "Learning the number of neurons in deep networks," in *NeurIPS*, pp. 2270–2278.

[61] Wen, W., C. Wu, Y. Wang, Y. Chen, and H. Li (2016) "Learning structured sparsity in deep neural networks," in *NeurIPS*, pp. 2074–2082.

[62] Yoon, J. and S. J. Hwang (2017) "Combined group and exclusive sparsity for deep neural networks," in *ICML*, JMLR. org, pp. 3958–3966.

[63] LeCun, Y., L. Bottou, Y. Bengio, P. Haffner, et al. (1998) "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, **86**(11), pp. 2278–2324.

[64] Xian, Y., C. H. Lampert, B. Schiele, and Z. Akata (2018) "Zero-shot learning-a comprehensive evaluation of the good, the bad and the ugly," *TPAMI*.

[65] Argyriou, A., T. Evgeniou, and M. Pontil (2007) "Multi-task feature learning," in *NeurIPS*, pp. 41–48.

[66] Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova (2018) "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv preprint arXiv:1810.04805*.

[67] Abe, N., N. Verma, C. Apte, and R. Schroko (2004) "Cross channel optimized marketing by reinforcement learning," in *SIGKDD*, ACM, pp. 767–772.

[68] Chen, M., A. Beutel, P. Covington, S. Jain, F. Belletti, and E. H. Chi (2019) "Top-k off-policy correction for a REINFORCE recommender system," in *WSDM*, ACM, pp. 456–464.

[69] Zou, L., L. Xia, Z. Ding, J. Song, W. Liu, and D. Yin (2019), "Reinforcement learning to optimize long-term user engagement in recommender systems," `1902.05570`.

[70] Mandel, T., Y.-E. Liu, S. Levine, E. Brunskill, and Z. Popovic (2014) "Offline policy evaluation across representations with applications to educational games," in *AAMAS*, pp. 1077–1084.

[71] Wang, L., W. Zhang, X. He, and H. Zha (2018) "Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation," in *SIGKDD*, ACM, pp. 2447–2456.

[72] Wei, H., G. Zheng, H. Yao, and Z. Li (2018) "Intellilight: A reinforcement learning approach for intelligent traffic light control," in *SIGKDD*, pp. 2496–2505.

[73] Xiong, Y., G. Zheng, K. Xu, and Z. Li (2019) "Learning Traffic Signal Control from Demonstrations," in *CIKM*, pp. 2289–2292.

[74] Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. (2015) "Human-level control through deep reinforcement learning," *Nature*, **518**(7540), p. 529.

[75] Silver, D., A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. (2016) "Mastering the game of Go with deep neural networks and tree search," *Nature*, **529**(7587), p. 484.

[76] Theocharous, G., P. S. Thomas, and M. Ghavamzadeh (2015) "Personalized Ad recommendation systems for life-time value optimization with guarantees." in *IJCAI*, pp. 1806–1812.

[77] Zheng, G., F. Zhang, Z. Zheng, Y. Xiang, N. J. Yuan, X. Xie, and Z. Li (2018) "DRN: a deep reinforcement learning framework for news recommendation," in *WWW*, pp. 167–176.

[78] Levine, S., C. Finn, T. Darrell, and P. Abbeel (2016) “End-to-end training of deep visuomotor policies,” *JMLR*, **17**(1), pp. 1334–1373.

[79] Michels, J., A. Saxena, and A. Y. Ng (2005) “High speed obstacle avoidance using monocular vision and reinforcement learning,” in *ICML*, ACM, pp. 593–600.

[80] Bottou, L., J. Peters, J. Quiñonero-Candela, D. X. Charles, D. M. Chickering, E. Portugaly, D. Ray, P. Simard, and E. Snelson (2013) “Counterfactual reasoning and learning systems: The example of computational advertising,” *JMLR*, **14**(1), pp. 3207–3260.

[81] Cai, H., K. Ren, W. Zhang, K. Malialis, J. Wang, Y. Yu, and D. Guo (2017) “Real-time bidding by reinforcement learning in display advertising,” in *WSDM*, ACM, pp. 661–670.

[82] Li, J., W. Monroe, A. Ritter, M. Galley, J. Gao, and D. Jurafsky (2016) “Deep reinforcement learning for dialogue generation,” *arXiv preprint arXiv:1606.01541*.

[83] Bahdanau, D., P. Brakel, K. Xu, A. Goyal, R. Lowe, J. Pineau, A. Courville, and Y. Bengio (2016) “An actor-critic algorithm for sequence prediction,” *arXiv preprint arXiv:1607.07086*.

[84] Li, T., Z. Xu, J. Tang, and Y. Wang (2018) “Model-free control for distributed stream data processing using deep reinforcement learning,” *VLDB*, **11**(6), pp. 705–718.

[85] Trummer, I., S. Moseley, D. Maram, S. Jo, and J. Antonakakis (2018) “SkinnerDB: regret-bounded query evaluation via reinforcement learning,” *VLDB*, **11**(12), pp. 2074–2077.

[86] Zhang, J., Y. Liu, K. Zhou, G. Li, Z. Xiao, B. Cheng, J. Xing, Y. Wang, T. Cheng, L. Liu, M. Ran, and Z. Li (2019) “An end-to-end automatic cloud database tuning system using deep reinforcement learning,” *SIDMOD*.

[87] Li, Y. (2017) “Deep reinforcement learning: An overview,” *arXiv preprint arXiv:1701.07274*.

[88] Arulkumaran, K., M. P. Deisenroth, M. Brundage, and A. A. Bharath (2017) “A brief survey of deep reinforcement learning,” *arXiv preprint arXiv:1708.05866*.

[89] Radlinski, F., R. Kleinberg, and T. Joachims (2008) “Learning diverse rankings with multi-armed bandits,” in *ICML*, ACM, pp. 784–791.

[90] Li, L., W. Chu, J. Langford, and R. E. Schapire (2010) "A contextual-bandit approach to personalized news article recommendation," in *WWW*, ACM, pp. 661–670.

[91] Zhao, X., W. Zhang, and J. Wang (2013) "Interactive collaborative filtering," in *CIKM*, ACM, pp. 1411–1420.

[92] Cai, Q., A. Filos-Ratsikas, P. Tang, and Y. Zhang (2018) "Reinforcement mechanism design for e-commerce," in *WWW*, pp. 1339–1348.

[93] Hu, Y., Q. Da, A. Zeng, Y. Yu, and Y. Xu (2018) "Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application," in *SIGKDD*, ACM, pp. 368–377.

[94] Zhao, X., L. Zhang, Z. Ding, L. Xia, J. Tang, and D. Yin (2018) "Recommendations with negative feedback via pairwise deep reinforcement learning," in *SIGKDD*, pp. 1040–1048.

[95] Zhao, X., L. Xia, L. Zhang, Z. Ding, D. Yin, and J. Tang (2018) "Deep reinforcement learning for page-wise recommendations," in *RecSys*, pp. 95–103.

[96] Schulman, J., F. Wolski, P. Dhariwal, A. Radford, and O. Klimov (2017) "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*.

[97] Ie, E., V. Jain, J. Wang, S. Narvekar, R. Agarwal, R. Wu, H.-T. Cheng, T. Chandra, and C. Boutilier (2019) "SlateQ: A Tractable Decomposition for Reinforcement Learning with Recommendation Sets," in *IJCAI-19*, International Joint Conferences on Artificial Intelligence Organization, pp. 2592–2599.
URL `https://doi.org/10.24963/ijcai.2019/360`

[98] Li, Y., Y. Zheng, and Q. Yang (2018) "Dynamic bike reposition: A spatio-temporal reinforcement learning approach," in *SIGKDD*, ACM, pp. 1724–1733.

[99] Van Hasselt, H., A. Guez, and D. Silver (2016) "Deep reinforcement learning with double Q-learning." in *AAAI*, vol. 2, Phoenix, AZ, p. 5.

[100] Swaminathan, A. and T. Joachims (2015) "The self-normalized estimator for counterfactual learning," in *NIPS*, pp. 3231–3239.

[101] Schulman, J., S. Levine, P. Abbeel, M. Jordan, and P. Moritz (2015) "Trust region policy optimization," in *ICML*, pp. 1889–1897.

[102] Kingma, D. P. and J. Ba (2014) “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980.*

[103] Motajcsek, T., J.-Y. Le Moine, M. Larson, D. Kohlsdorf, A. Lommatzsch, D. Tikk, O. Alonso, P. Cremonesi, A. Demetriou, K. Dobrajs, et al. (2016) “Algorithms aside: Recommendation as the lens of life,” in *Proceedings of the 10th ACM Conference on Recommender Systems*, pp. 215–219.

[104] Chaney, A. J., B. M. Stewart, and B. E. Engelhardt (2018) “How algorithmic confounding in recommendation systems increases homogeneity and decreases utility,” in *Proceedings of the 12th ACM Conference on Recommender Systems*, pp. 224–232.

[105] Jin, Y., N. Tintarev, and K. Verbert (2018) “Effects of personal characteristics on music recommender systems with different levels of controllability,” in *Proceedings of the 12th ACM Conference on Recommender Systems*, pp. 13–21.

[106] Harambam, J., D. Bountouridis, M. Makhortykh, and J. Van Hoboken (2019) “Designing for the better by taking users into account: a qualitative evaluation of user control mechanisms in (news) recommender systems,” in *Proceedings of the 13th ACM Conference on Recommender Systems*, pp. 69–77.

[107] Mnih, A. and R. R. Salakhutdinov (2008) “Probabilistic matrix factorization,” in *Advances in neural information processing systems*, pp. 1257–1264.

[108] Zeng, C., Q. Wang, S. Mokhtari, and T. Li (2016) “Online context-aware recommendation with time varying multi-armed bandit,” in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 2025–2034.