Spring 2015

Learning compact hashing codes with complex objectives from multiple sources for large scale similarity search

Qifan Wang
Purdue University

Graduate School Form 30
Updated 1/15/2015

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Qifan Wang

Entitled
LEARNING COMPACT HASHING CODES WITH COMPLEX OBJECTIVES FROM MULTIPLE SOURCES FOR LARGE SCALE SIMILARITY SEARCH

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

Luo Si
Chair

Yi Wu

David Gleich

Dongyan Xu

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): Luo Si

Approved by: William J. Gorman 4/14/2015
Head of the Departmental Graduate Program Date

LEARNING COMPACT HASHING CODES WITH COMPLEX OBJECTIVES FROM MULTIPLE SOURCES FOR LARGE SCALE SIMILARITY SEARCH

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Qifan Wang

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2015

Purdue University

West Lafayette, Indiana

ACKNOWLEDGMENTS

My greatest gratitude goes to my advisor, Prof. Luo Si, for guiding me towards one of the major achievements in my life. His enthusiasm in research and his patience in teaching have made me become an independent researcher. I still remember the day when he first introduce the research area of information retrieval to me. He always provides useful guidance and help when I most needed in research. His work ethics, his insights, and his accomplishments have been and will be an inspiration to me for years to come. I am very proud and lucky to have him as my Ph.D. advisor. I truly miss the time working with him.

I would like to show my special thanks to my Ph.D. committee members, Prof. David Gleich, Yi Wu, Jennifer Neville and Dongyan Xu, for their excellent suggestion and advice during my exams and on my dissertation. My appreciation also goes to many other faculty members in the computer science department, for the courses they offered that helped me build a solid foundation. I would like to extend my thanks to Dr. William J. Gorman for his dedication to students and to the department, and Sandra Freeman who helps me a lot.

The information retrieval lab is my academic home. I would like to thank all of my lab mates, Dan Zhang, Yi Fang, Suleyman Cetintas, Dzung Hong, Ahmet Bugdayci, Bin Shen, Mariheida Cordova, Lei Cen, Lingyun Ruan, Praveen Kumar, Zhiwei Zhang, Ning Zhang, Chuanfei Ouyang, Tao Wu, Kexin Pei and Chang Li, for their valuable collaborations and assistance on my research, as well as for the great friendship. My life at Purdue would have been much less fun without my friends. I have always enjoyed the great time with Mu Wang, Liang Li, Wei Feng, Yingkai Hu, Zhui Deng, Lian Duan, Wenchang Zhou, Chao Wu, Xiahong Lin, Jingjing Mao, Yangyang Hou, Jiahong Zhu, Yuelin Wang, Di Jin, Dicong Chen, for all the joy they have brought to me in these years.

Finally, I would like to thank my family, especially my fiancee Shumiao, for their love, encouragement, understanding, and continuous support during my Ph.D. studies.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

ABSTRACT

Wang, Qifan Ph.D., Purdue University, May 2015. Learning Compact Hashing Codes with Complex Objectives from Multiple Sources for Large Scale Similarity Search . Major Professor: Luo Si.

Similarity search is a key problem in many real world applications including image and text retrieval, content reuse detection and collaborative filtering. The purpose of similarity search is to identify similar data examples given a query example. Due to the explosive growth of the Internet, a huge amount of data such as texts, images and videos has been generated, which indicates that efficient large scale similarity search becomes more important.

Hashing methods have become popular for large scale similarity search due to their computational and memory efficiency. These hashing methods design compact binary codes to represent data examples so that similar examples are mapped into similar codes. This dissertation addresses five major problems for utilizing supervised information from multiple sources in hashing with respect to different objectives. Firstly, we address the problem of incorporating semantic tags by modeling the latent correlations between tags and data examples. More precisely, the hashing codes are learned in a unified semi-supervised framework by simultaneously preserving the similarities between data examples and ensuring the tag consistency via a latent factor model. Secondly, we solve the missing data problem by latent subspace learning from multiple sources. The hashing codes are learned by enforcing the data consistency among different sources. Thirdly, we address the problem of hashing on structured data by graph learning. A weighted graph is constructed based on the structured knowledge from the data. The hashing codes are then learned by preserving the graph similarities. Fourthly, we address the problem of learning high ranking quality hashing codes by utilizing the relevance judgments from users. The hashing code/function is learned via optimizing a commonly used non-smooth non-convex ranking measure, NDCG. Finally, we deal with the problem of insufficient supervision by active learning. We propose to actively select the most informative data examples and tags in a joint manner based on the selection criteria that both the data examples and tags should be most uncertain and dissimilar with each other.

Extensive experiments on several large scale datasets demonstrate the superior performance of the proposed approaches over several state-of-the-art hashing methods from different perspectives.

1 INTRODUCTION

Similarity search identifies similar data examples given a query example, which has many different applications in various research areas including information retrieval, machine learning, data mining and computer vision. Due to the explosive growth of the Internet, a huge amount of data such as texts, images and videos has been generated, which indicates that efficient large scale similarity search becomes more important. When there is only a low-dimensional feature space, similarity search can be carried out by some space partitioning index structures, such as TF-IDF methods [1, 2], KD-tree, or data partitioning index structures, like R-tree [3]. Several types of structures and operations of inverted indexing are also proposed [4–6] for traditional ad-hoc text search with relatively short user queries. However, traditional similarity search may fail to work efficiently within a high-dimensional vector space [7], which is often the case for many real world information retrieval applications. Therefore, it is important to design effective and efficient methods for similarity search with large scale data. Two major challenges have to be addressed for using similarity search in large scale datasets such as storing the data efficiently and retrieving the large scale data in an effective and efficient manner.

Traditional similarity search methods are difficult to be directly used for large scale datasets since the computational cost of similarity calculation using the original data features (i.e. often in high dimensional space) is impractical for large scale applications. Recently, hashing [8–24] has become a popular approach in large scale problems such as similar document detection [25–27], content-based image retrieval [28–31] and collaborative filtering [32, 33], etc. Hashing methods design compact binary codes to represent data examples so that similar data examples are mapped into similar codes. In the retrieving process, these hashing methods first transform query examples into the corresponding hashing codes and then similarity search can be simply conducted by calculating the Hamming distances between the query code and the codes in the database, and selecting data examples within small Hamming distances. Therefore, hashing method addresses the two major challenges of large scale similarity search in the following ways: (1) The encoded data is highly compressed within a low-dimensional binary space, and thus can often be dealt with in main memory and stored efficiently; (2) The retrieval process is very efficient, since the distance between two codes is simply the number of bits that they differ, which can be computed using bit XOR operations.

Recent hashing methods have shown that the code performance could be boosted by incorporating supervised information from multiple sources into hashing codes learning, such as semantic tags/labels [34–37], structure data [38] and relevance values/judgments [31]. These supervised multi-source information provides useful knowledge and guidance for achieving more effective hashing codes. Although existing hashing methods generate promising results in large scale similarity search, the supervised knowledge is not fully exploited in previous methods. Most of the existing hashing methods only utilize a small portion of the knowledge extracted from different sources such as pairwise similarity and listwise ranking information, which might not be accurate or reliable. There are several main problems for leveraging the supervised knowledge from multiple sources into learning effective hashing codes for different objectives: 1. How to leverage these supervised knowledge from multiple sources to achieve more effective hashing codes; 2. How to handle the missing data issue from multiple sources; 3. How to incorporate the structure information into hashing code learning; 4. How to utilize multiple sources to learn high quality code for ranking-oriented measure; 5. How to obtain the most informative knowledge in an active manner with low human labeling cost for generating supervised information, when the supervision is insufficient. In this dissertation, we discuss these problems in the following sections respectively.

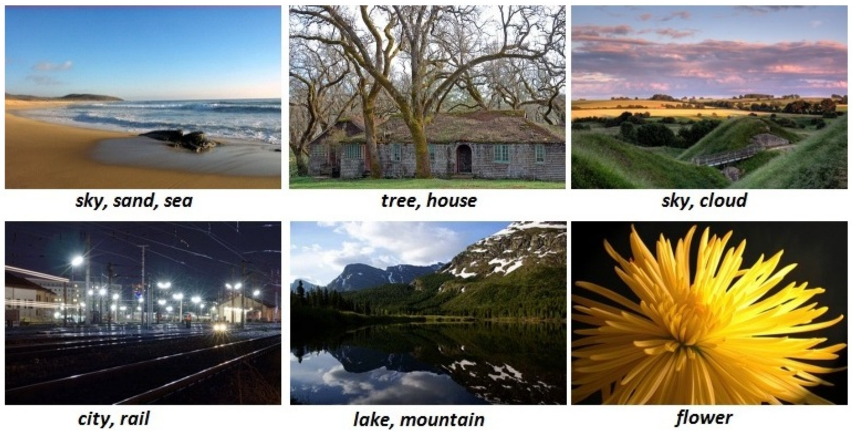


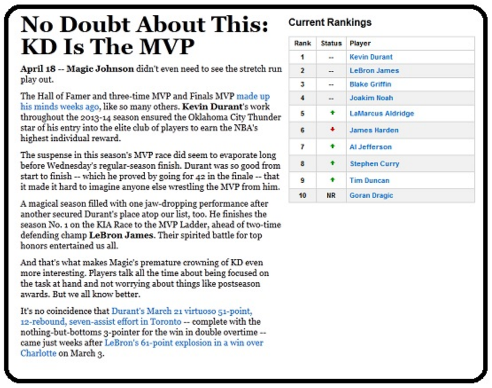
Figure 1.1. Several image examples with tags from *MIRFLICKR*-1*M* dataset.

1.1 Hashing Methods for Large Scale Similarity Search

1.1.1 Learning to Hash with Semantic Tags

Tags or labels have been popularly utilized in many applications with image and text data for better managing, organizing and searching for useful information. For example, Flickr has more than 2 billion images with millions of newly uploaded photos per day and YouTube contains hundreds of millions of videos [23]. These data examples are usually associated with multiple tags assigned by users. Figure 1.1 and 1.2 show some examples of images and webpages associated with multiple tags. This tag information source provides useful supervised knowledge for users to better categorize or search desired data. Therefore, it is an important and practical research problem to design efficient and effective hashing methods that can incorporate these supervised information for large scale similarity search.

Several supervised/semi-supervised hashing methods have been proposed to utilize the tag information into their hashing function learning, such as semi-supervised hashing (SSH) [28, 29], kernel supervised hashing (KSH) [30], Canonical Correlation Analysis with Iterative Quantization (CCA-ITQ) [35, 36], etc. For example, in SSH



No Doubt About This: KD Is The MVP

April 18 -- **Magic Johnson** didn't even need to see the stretch run play out.

The Hall of Famer and three-time MVP and Finals MVP made up his minds weeks ago, like so many others. **Kevin Durant**'s work throughout the 2013-14 season ensured the Oklahoma City Thunder star of his entry into the elite club of players to earn the NBA's highest individual reward.

The suspense in this season's MVP race did seem to evaporate long before Wednesday's regular-season finish. Durant was so good from start to finish -- which he proved by going for 42 in the finale -- that it made it hard to imagine anyone else wrestling the MVP from him.

A magical season filled with one jaw-dropping performance after another secured Durant's place atop our list, too. He finishes the season No. 1 on the KIA Race to the MVP Ladder, ahead of two-time defending champ **LeBron James**. Their spirited battle for top honors entertained us all.

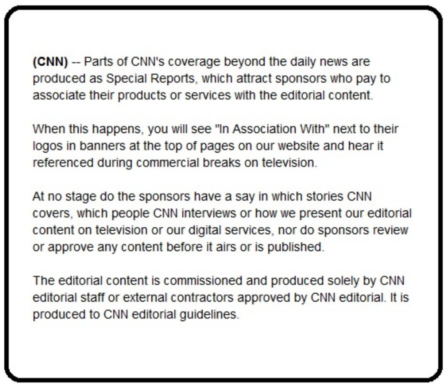
And that's what makes Magic's premature crowning of KD even more interesting. Players talk all the time about being focused on the task at hand and not worrying about things like postseason awards. But we all know better.

It's no coincidence that Durant's March 21 virtuoso 51-point, 12-rebound, seven-assist effort in Toronto -- complete with the nothing-but-bottoms 3-pointer for the win in double overtime -- came just weeks after LeBron's 61-point explosion in a win over Charlotte on March 3.

Current Rankings

Rank	Status	Player
1	--	Kevin Durant
2	--	LeBron James
3	--	Blake Griffin
4	--	Joakim Noah
5	↑	LaMarcus Aldridge
6	↓	James Harden
7	↑	Al Jefferson
8	↑	Stephen Curry
9	↑	Tim Duncan
10	NR	Goran Dragic

NBA, sports, basketball



(CNN) -- Parts of CNN's coverage beyond the daily news are produced as Special Reports, which attract sponsors who pay to associate their products or services with the editorial content.

When this happens, you will see "In Association With" next to their logos in banners at the top of pages on our website and hear it referenced during commercial breaks on television.

At no stage do the sponsors have a say in which stories CNN covers, which people CNN interviews or how we present our editorial content on television or our digital services, nor do sponsors review or approve any content before it airs or is published.

The editorial content is commissioned and produced solely by CNN editorial staff or external contractors approved by CNN editorial. It is produced to CNN editorial guidelines.

economy, commercial

Figure 1.2. Several document examples with tags.

and KSH, pairwise constrains between data examples besides their original features are imposed for learning more effective hashing function. More precisely, these pairwise similarity constraints are also called Must-Link and Cannot-Link, which are generated from tags. A Must-Link is created when two data examples share a common tag and a Cannot-Link is created when two examples share no tag. Their basic motivation is that the hashing codes of data example pairs with Must-Link should be as close as possible, while the hashing codes of example pairs with Cannot-Link should be as different as possible. For the CCA-ITQ method, it treats the data features and tags as two different views. The hashing function is then learned by extracting a common space from these two views.

One major assumption of most existing supervised hashing methods is that the tags associated with data examples are complete and clean. But in many applications, tags tend to be incomplete and noisy. Moreover, tags may have different representations for a similar semantic meaning (e.g.,'car' versus 'automobile'). In this situation, the pairwise constraints extracted from the semantic tags only represent a small portion of tag information rather than the complete supervised knowledge and thus are not reliable. Similarly, CCA-ITQ may generate low quality codes when only incomplete tags is available. We need to design a scheme that could fully leverage the supervision knowledge in hashing function learning while at the same time preserves the data similarity.

1.1.2 Learning to Hash on Partial Multi-Modal Data

In many applications, data examples are usually represented by multiple modalities captured from different sources. For example, in web page search, the web page content and its linkage information can be regarded as two modalities. In web image retrieval, the image visual feature, text description and textual tags can be viewed as multiple modalities. Recently, several multi-modal hashing methods (also known as multi-view or cross-view hashing) have been proposed to handle multi-modal data. Roughly speaking, these multi-modal hashing approaches can be divided into two categories: modality-specific hashing methods and modality-integrated ones. The modality-specific methods learn independent hashing codes for each modality of an example, and then concatenate multiple modality-specific binary codes into the final hashing codes [39–42], whereas the modality-integrated ones directly learn unified hashing codes for data examples [10, 43–45].

Although existing multi-modal hashing methods generate promising results in dealing with multi-modal data, most of them assume that all data examples have full information in all modalities, or there exists at least one modality which contains all the examples. However, in real world tasks, it is often the case that every modality suffers from some missing information, which results in many partial examples. Consider again the aforementioned two examples, for web page search, many web pages may not contain any linkage information. For web image retrieval, not all images are associated with tags or text descriptions. Moreover, the image itself may be inaccessible due to deletion or invalid url. Therefore, it is a practical and important research problem to design effective hashing methods for partial multi-modal data.

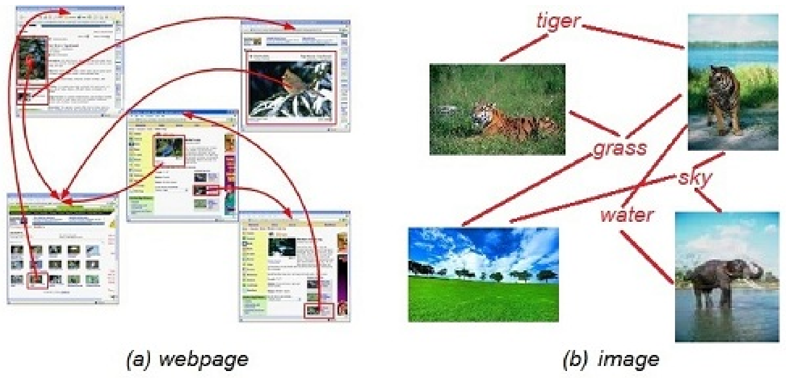


Figure 1.3. Data examples with structure information. (a) Webpages link to each other. (b) Images share semantic labels.

1.1.3 Learning to Hash on Structured Data

Hashing methods generate promising results by successfully addressing the storage and search efficiency challenges. However, most existing hashing methods assume that data examples are independently and identically distributed. But in many applications, the dependencies between data examples naturally exist and if incorporated in models, they can potentially improve the hashing code performance significantly.

For example, many webpages have hyperlinks pointing to other related webpages (see Fig.1.3(a)). The contents of these linked webpages are usually relevant, which present similar topics. The hyperlinks among webpages provide important structure knowledge. Another example is that similar images often share semantic labels (see Fig.1.3(b)). The more labels two images have in common, the more similar the images are. The shared semantic labels among images offer valuable information in binary codes learning. These structure information have been utilized in clustering [46] and classification [38] problems, and proven to be helpful knowledge. Therefore, it is important to design hashing method that preserve the structure information among data examples in the learned Hamming space.

1.1.4 Learning Ranking Preserving Hashing Codes

In many information retrieval applications, such as similarity search, learning to rank, recommendation, etc., it is more realistic and desirable that the most relevant examples to a query can be presented in front of less relevant ones. In other words, users prefer the retrieval results with better ranking performance. Although existing hashing methods have achieved promising results, very limited work explores the search/ranking accuracy, which is important for evaluating the quality of hashing codes in real world applications. Consider the following scenario: given a query example x_q and three relevant/similar data examples x_1, x_2, x_3 but with different relevance values as $r_1 > r_2 > r_3$ to the query. Most existing hashing methods only model the relevance of a data example to a query in a binary way, i.e., each example is either relevant to the query or irrelevant. These methods treat x_1, x_2 and x_3 as relevant examples to x_q with no difference. But in practice it will be more desirable if x_1 could be presented before x_2 and x_3 since it is more relevant to x_q than the other two. Some ranking based hashing methods [31, 47–49] have been recently proposed to improve the hashing performance by modeling the ranking order with respect to relevance values. However, these methods do not fully preserve the specific relevance values in learning hashing function, while the relevance values are important in evaluating the search accuracy. In other words, they do not differentiate the situations where $(r_1, r_2, r_3) = (3, 2, 1)$ and $(10, 2, 1)$ due to their identical ranking orders, i.e., $r_1 > r_2 > r_3$. But ideally, the Hamming distance between the learned hashing codes of x_1 and x_q should be smaller in the later situation than in the former one since the relevance value of x_1 to the query example is much larger in the later situation (10 versus 3). Therefore, it is important to design effective hashing method to incorporate relevance value/judgement information from users in learning more effective hashing codes that could achieve high ranking performance.

1.1.5 Active Learning with Insufficient Supervision

One of the basic assumptions in most existing supervised hashing methods is that the labeled data are provided beforehand. These hashing methods are regarded as passive methods. But in many real world applications, such supervised information may not be sufficient or available and it is often expensive to acquire for a large dataset. Therefore, it is important to design effective methods to actively identify only a small set of the most informative data examples for users to label. The only prior work we found using active learning in hashing is [50], which directly chooses the most uncertain data examples based on the hashing function. A batch mode algorithm is also proposed in this work to speed up their active selection.

The labeling cost not only depends on the number of data examples that are selected but also depends on the total number of tags that the users label to the selected data examples. In many large scale applications, there are often hundreds or thousands of tags for users to label. Moreover, similar tags usually carry similar semantic meanings. For instance, 'car' and 'automobile' have similar meanings and choosing both of them may not gain substantial new information over just selecting one. However, the method in [50] only considers identifying the most informative data examples and tries to label all possible tags to these selected examples, which requires a great amount of labeling efforts for those datasets associated with a huge number of tags. Therefore, it is important to design effective method that jointly selects the most informative data examples and tags such that the hashing function can be learned efficiently with only a small number of labeled data, which can greatly reduces the labeling cost.

1.2 Main Contributions

Hashing methods generate promising results in large scale similarity search. As can be seen from the above discussion, the problem of leveraging supervised information from multiple sources has not been fully explored. The major contribution of this dissertation is that we propose a unified framework to incorporate different types of supervised knowledge from multi-sources into learning effective hashing codes. We summarize our contributions in more details as follows.

- As shown in Section 1.1.1, semantic tags or labels are usually associated with data examples and have been popularly utilized in many applications. This tag information source provides useful supervised knowledge for users to better categorize or search desired data. In this dissertation, a research problem, Learning to Hash with Semantic Tags, is proposed to incorporate the semantic tags into hashing codes learning. To solve this problem, a novel semi-supervised tag hashing (SSTH) approach is proposed to fully exploit tag information by modeling the semantic correlation between tags and hashing bits. The hashing function is learned in a unified learning framework by simultaneously ensuring the tag consistency and preserving the similarities between data examples. An iterative coordinate descent algorithm is designed as the optimization procedure. We also improve the effectiveness of hashing function through orthogonal transformation by minimizing the quantization error. Furthermore, we extend this framework by preserving the topic level similarity between data examples to obtain more effective codes when original feature distances do not reflect the similarity between data examples.

- In many applications, data examples are usually represented by multiple modalities captured from different sources. However, in real world tasks, it is often the case that every modality suffers from some missing information, which results in many partial examples. For example, not all images are associated with tags or text descriptions. Moreover, the image itself may be inaccessible due to deletion or invalid url. In this dissertation, we propose a novel Partial Multi-Modal Hashing (PM^{2}H) approach to deal with such partial data. More specifically, a unified learning framework is developed to learn the binary codes, which simultaneously ensures the data consistency among different modalities via latent subspace learning, and preserves data similarity within the same modality through graph Laplacian. A block gradient descent algorithm is applied as the optimization procedure.

- As discussed in Section 1.1.3, in many applications, the dependencies between data examples naturally exist and if incorporated in models, they can potentially improve the hashing code performance significantly. In this dissertation, a novel approach of learning to Hash on Structured Data (HSD) is proposed, which incorporates the structure information associated with data. The hashing function is learned in a unified learning framework by simultaneously ensuring the structural consistency and preserving the similarities between data examples. In particular, the objective function of the proposed HSD approach is composed of two parts: (1) Structure consistency term, which ensures the hashing codes to be consistent with the structure information. (2) Similarity preservation term, which aims at preserving the similarity between data examples in the learned hashing codes.

- In many learning to rank and recommendation systems, it is more realistic and desirable that the most relevant examples to a query can be presented in front of less relevant ones. In other words, users prefer the retrieval results with better ranking performance. This dissertation proposes a novel Ranking Preserving Hashing (RPH) approach that directly optimizes the popular ranking accuracy measure, Normalized Discounted Cumulative Gain (NDCG), to learn effective ranking preserving hashing codes that not only preserves the ranking order but also models the relevance values of data examples to the queries in the training data. The main difficulty in direct optimization of NDCG is that it depends on the rankings of data examples rather than their hashing codes, which forms a non-convex non-smooth objective. We then address this challenge by optimizing the expectation of NDCG measure calculated based on a linear hashing function to convert the problem into a smooth and differentiable optimization problem. A gradient descent method is applied to solve this relaxed problem.

- When the supervised information is insufficient or even not available, it is important to design effective methods to actively identify only a small set of the most informative data examples for users to label. In this dissertation, we proposes a novel active hashing approach to actively select the most informative data examples and tags in a joint manner for hashing function learning. We first identify a set of informative data examples and tags for users to label based on the selection criteria that both the data examples and tags should be most uncertain and dissimilar with each other. Then this labeled information is combined with the unlabeled data to generate an effective hashing function. An iterative procedure is proposed for learning the optimal hashing function and selecting the most informative data examples and tags.

The rest of the dissertation is organized as follows: Chapter 2 proposes a novel hashing approach to leverage the semantic tag knowledge by modeling semantic correlations between hashing codes and tags. Chapter 3 discusses the solution of dealing with partial multi-modal data. A novel hashing on structured data approach is given in Chapter 4 to incorporate the structure knowledge among data examples. Chapter 5 presents a hashing method to learn ranking based hashing codes with relevance supervision via optimizing the NDCG measure. Chapter 6 designs an active hashing method with joint data example and tag selection to handle the insufficient supervision problem. Finally, Chapter 7 gives the conclusions and future work.

1.3 Origins of the Material

The material in this dissertation is based on a number of papers listed below, some of which have already been published, while others are currently in submission.

1. **Qifan Wang**, Luo Si and Bin Shen. Learning to Hash on Partial Multi-Modal Data. In submission, 2015.

2. **Qifan Wang**, Zhiwei Zhang and Luo Si. Ranking Preserving Hashing for Large Scale Similarity Search. In submission, 2015.

3. Dennis Strelow, **Qifan Wang**, Anders Eriksson and Luo Si. General, Nested, and Constrained Wiberg Minimization. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2015. In submission.

4. **Qifan Wang**, Luo Si and Bin Shen. Learning to Hash on Structured Data. The Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI), 2015.

5. Bin Shen, Baodi Liu, **Qifan Wang**, Yi Fang and Jan Allebach. SP-SVM: Large Margin Classifier for Data on Multiple Manifolds. The Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI), 2015.

6. Xiaojun Quan, **Qifan Wang**, Ying Zhang, Luo Si and Wenyin Liu. Latent Discriminative Models for Social Emotion Detection with Emotion Dependency. ACM Transactions on Information Systems (TOIS), 2015.

7. Zhongshu Gu, Kexin Pei, **Qifan Wang**, Luo Si, Xiangyu Zhang and Dongyan Xu. LEAPS: Detecting Camouflaged Attacks with Statistical Learning Guided by Program Analysis. The 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2015.

8. Bin Shen, **Qifan Wang** and Jan Allebach. Piecewise Linear Dimension Reduction for Nonnegative Data. Imaging and Multimedia Analytics in a Web and Mobile World SPIE Electronic Imaging, 2015.

9. **Qifan Wang**, Luo Si, Zhiwei Zhang and Ning Zhang. Active Hashing with Joint Data Example and Tag Selection. The 37th Annual ACM SIGIR Conference (SIGIR), 2014.

10. Zhiwei Zhang, **Qifan Wang**, Lingyun Ruan and Luo Si. Preference Preserving Hashing for Efficient Recommendation. The 37th Annual ACM SIGIR Conference (SIGIR), 2014.

11. Ning Zhang, Ying Zhang, Luo Si, Yanshan Lu, **Qifan Wang** and Xiaojie Yuan. Cross-Domain and Cross-Category Emotion Tagging for Comments of Online News. The 37th Annual ACM SIGIR Conference (SIGIR), 2014.

12. **Qifan Wang**, Lingyun Ruan and Luo Si. Adaptive Knowledge Transfer for Multiple Instance Learning in Image Classification. The Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI), 2014.

13. **Qifan Wang**, Luo Si and Dan Zhang. Learning to Hash with Partial Tags: Exploring Correlation Between Tags and Hashing Bits for Large Scale Image Retrieval. The 13th European Conference on Computer Vision (ECCV), 2014.

14. **Qifan Wang**, Bin Shen, Shumiao Wang, Liang Li and Luo Si. Binary Codes Embedding for Fast Image Tagging with Incomplete Labels. The 13th European Conference on Computer Vision (ECCV), 2014.

15. **Qifan Wang**, Bin Shen, Zhiwei Zhang and Luo Si. Sparse Semantic Hashing for Efficient Large Scale Similarity Search. The 23rd ACM International Conference on Information and Knowledge Management (CIKM), 2014.

16. Bin Shen, Baodi Liu, **Qifan Wang** and Rongrong Ji. Robust Nonnegative Matrix Factorization via L1 Norm Regularization. The 21st IEEE International Conference on Image Processing (ICIP), 2014.

17. Bin Shen, Baodi Liu and **Qifan Wang**. Elastic Net Regularized Dictionary Learning for Image Classification. Multimedia Tools and Applications, 2014.

18. **Qifan Wang**, Dan Zhang and Luo Si. Semantic Hashing using Tags and Topic Modeling. The 36th Annual ACM SIGIR Conference (SIGIR), 2013.

19. **Qifan Wang**, Lingyun Ruan, Zhiwei Zhang and Luo Si. Learning Compact Hashing Codes for Efficient Tag Completion and Prediction. The 22nd ACM International Conference on Information and Knowledge Management (CIKM), 2013.

20. **Qifan Wang**, Dan Zhang and Luo Si. Weighted Hashing for Fast Large Scale Similarity Search. The 22nd ACM International Conference on Information and Knowledge Management (CIKM), 2013.

21. **Qifan Wang**, Luo Si and Dan Zhang. A Discriminative Data-dependent Mixture-Model Approach for Multiple Instance Learning in Image Classification. The 12th European Conference on Computer Vision (ECCV), 2012.

22. Dzung Hong, **Qifan Wang**, Dan Zhang and Luo Si. Query Expansion and Message-passing Algorithms for TREC Microblog Track Embeddings. Text Retrieval Conference (TREC), 2011.

23. **Qifan Wang** and Luo Si. A Robust One-Class Bayesian Approach for Masquerade Detection. Conference on Computer and Communications Security: AI and Security Workshop (AISec), 2011.

2 LEARNING TO HASH WITH SEMANTIC TAGS

2.1 Motivation

Most of existing supervised and semi-supervised hashing methods assume the tags are complete and clean. But in many real world applications such as web image retrieval, tags tend to be noisy and are usually partially assigned to images by users. Moreover, semantic similar tags may have different representations (e.g.,'car' versus 'automobile'). Therefore, the assumption made by the existing hashing methods may limit the performance of learned hashing codes. In this dissertation, we propose a novel semi-supervised tag hashing approach to fully exploit tag information in learning effective hashing function by modeling the semantic correlation between tags and hashing bits. The hashing function is learned in a unified learning framework by simultaneously ensuring the tag consistency and preserving the similarities between data examples. An iterative coordinate descent algorithm is designed as the optimization procedure. We also improve the effectiveness of hashing function through orthogonal transformation by minimizing the quantization error. Furthermore, we extend this framework by preserving the topic level similarity between data examples to obtain more effective codes when original feature distances do not reflect the similarity between data examples.

2.2 Background and Related Work

2.2.1 Introduction

Due to the explosive growth of the Internet, a huge amount of data such as texts, images and videos has been generated, which indicates that efficient similarity search becomes more important. Traditional similarity search methods are difficult to be directly used for large scale datasets since the computational cost of similarity calculation using the original data features (i.e. often in high dimensional space) is impractical for large scale applications. Recently, hashing has become a popular approach in large scale problems, which designs compact binary codes to represent data examples so that similar examples are mapped into similar codes. In the retrieving process, these hashing methods first transform query examples into the corresponding hashing codes and then similarity search can be simply conducted by calculating the Hamming distances between the query code and the codes in the database, and selecting data examples within small Hamming distances.

Recently, hashing methods have shown that the code performance could be boosted by leveraging supervised information into hashing function learning, i.e., semantic tags/labels [34–37]. Although existing supervised hashing methods generate promising results in large scale similarity search, tag information is not fully exploited in previous methods, especially when tags are incomplete and noisy. Most of the existing hashing methods only utilize a small portion of the knowledge contained in tags such as pairwise similarity and listwise ranking information, which might not be accurate or reliable under the situation where only partial tags are available. There are three main challenges to incorporate tag information into hashing function learning: (1) we have no knowledge about how tags are related to the hashing bits; (2) we need to deal with noisy and incomplete tags when only partial tags are available; (3) we need to deal with the ambiguity of semantically similar tags.

This dissertation proposes a novel semi-supervised tag hashing (SSTH) approach to fully exploit tag information in learning effective hashing function by modeling the semantic correlation between tags and hashing bits. The hashing function is learned in a unified framework by simultaneously ensuring the tag consistency and preserving the similarities between data examples. In particular, the objective function of the proposed SSTH approach is composed of two components. (1) Tag consistency term (supervised), which ensures the hashing codes to be consistent with the observed tags. The key observation is that: the more common tags two data examples share, the more similar their hashing codes should be. This observation is then transformed into a latent factor model. (2) Similarity preservation term (unsupervised), which aims at preserving the original feature similarity between data examples in the learned hashing codes. This term is important especially when the data examples are associated with very few tags. An iterative algorithm is then derived based on the relaxed objective function using a coordinate descent optimization procedure. Moreover, we prove the orthogonal invariant property of the optimal relaxed solution and learn an orthogonal matrix by minimizing the quantization error to further improve the code effectiveness. We also extend this framework by preserving the topic level similarity between data examples to obtain more effective codes when original feature distances do not reflect the similarity between data examples. Extensive experiments on several large scale datasets demonstrate the superior performance of the proposed approach over several state-of-the-art hashing methods.

2.2.2 Related Work

Efficiency is a crucial issue for large scale information retrieval applications with a huge amount of data examples. When there is only a low-dimensional feature space, similarity search can be carried out by some space partitioning index structures, such as TF-IDF methods [1, 2], KD-tree, or data partitioning index structures, like R-tree [3]. Several types of structures and operations of inverted indexing are also proposed [4–6] for traditional ad-hoc text search with relatively short user queries. However, traditional similarity search may fail to work efficiently within a high-dimensional vector space [7], which is often the case for many real world information retrieval applications.

Hashing method [14, 48, 49, 51–63] is proposed to address the similarity search problem within a high-dimensional feature space. In particular, hashing methods try to represent each data example by using a small fixed number of binary bits so that the queries can be answered in a short time [64]. The hashing based fast similarity search can be viewed as a strategy to transform data examples from a high-dimensional space into a low-dimensional binary space, and at the same time preserve the semantic similarity between data examples as much as possible. Hashing methods generate binary codes for efficient search, which is different from traditional dimensionality reduction methods such as Principal Component Analysis (PCA) and Latent Semantic Indexing (LSI) [65, 66].

Locality-Sensitive Hashing (LSH) [15, 67] is one of the most popularly used hashing methods. It simply utilizes random linear projections to map data examples from a high-dimensional Euclidean space to a low-dimensional one. It has already been shown that the Hamming distance between different data examples will asymptotically approach their Euclidean distance in the original feature space with the increase of the hashing bits. LSH has been extended to Kernelized Locality-Sensitive Hashing (KLSH) [21, 23] by exploiting kernel similarity for better retrieval efficacy. Recently, the work in [68] further extends the KLSH to the scheme of Boosting Multi-Kernel Locality-Sensitive Hashing (BMKLSH) that improves the retrieval performance of KLSH by making use of multiple kernels.

Several machine learning approaches have been proposed to solve the hashing problem. For example, the PCA Hashing [17] method projects each example to the top principal components of the training set, and then binarizes the coefficients by setting a bit to 1 when its value is larger than the median value seen for the training set, and -1 otherwise. The work in [8] uses stacked Restricted Boltzman Machine (RBM) [69, 70] to generate compact binary hashing codes, which can be viewed as binarized LSI. Recently, Spectral Hashing (SH) [19] is proposed to learn compact binary codes that preserve the similarity between data examples by forcing the balanced and uncorrelated constraints into the learned codes, which be viewed as an extension of spectral clustering [71]. A graph-based hashing method has been proposed in work [12] to automatically discover the neighborhood structure inherent in the data to learn appropriate compact codes. A Self-taught Hashing method [22] combines an unsupervised step with a supervised step to learn more accurate hashing codes. More recently, the work [10] proposes a Composite Hashing with Multiple Information Sources (CHMIS) method to integrate information from different sources. In another recent work [56], an isotropic hashing (IsoHash) method is proposed to learn projection functions of individual hashing codes with equal variances. A bit selection method [72] has been proposed to select the most informative hashing bits from a pool of candidate bits generated from different hashing methods. A Topology Preserving Hashing (TPH) [73] method is proposed to preserve the neighborhood rankings of data points in Hamming space. Most recently, several supervised/semi-supervised hashing methods have been proposed. For example, a Canonical Correlation Analysis with Iterative Quantization (CCA-ITQ) method has been proposed in [35, 36] which treats the data features and tags as two different views. The hashing function is then learned by extracting a common space from these two views. The semi-supervised hashing (SSH) method in [28, 29] utilizes pairwise knowledge between data examples besides their original features for learning more effective hashing function. A kernelized supervised hashing (KSH) framework proposed in [30] imposes the pairwise relationship between data examples to obtain good hashing codes. Complementary Hashing (CH) [9] uses pairwise information to learn multiple complementary hash tables in a boosting manner. A ranking-based supervised hashing (RSH) [31] method is proposed to leverage the listwise ranking information to improve the search accuracy.

In the following sections, we mainly discuss several state-of-the art hashing methods including three unsupervised methods, i.e., Locality-Sensitive Hashing (LSH) [15], Spectral Hashing (SH) [19] and Self-taught Hashing (STH) [22], and three supervised/semi-supervised hashing methods, i.e., Semi-Supervised Hashing (SSH) [28,29], Kernelized Supervised Hashing (KSH) [30] and Canonical Correlation Analysis with Iterative Quantization (CCA-ITQ) [35,36].

Locality-Sensitive Hashing (LSH)

Locality-Sensitive Hashing [15] is one of the most popularly used hashing methods. It simply utilizes k random linear projections to map data examples from a high-dimensional Euclidean space to a low-dimensional binary space as follows:

$$y_p = sgn(w_p^T x + b_p) \tag{2.1}$$

where y_p is the p-th bit for data example x and w_p is the linear projection which is randomly samples from Gaussian distribution. b_p is the bias. We also illustrate LSH in figure 2.1 and 2.2. Although LSH is a data-independent hashing method, it has already been shown that the Hamming distance between different data examples will asymptotically approach their Euclidean distance in the original feature space with the increase of the hashing bits. LSH method has been extended to Kernelized Locality-Sensitive Hashing (KLSH) [21, 23] by exploiting kernel similarity for better retrieval efficacy. Recently, the work in [68] further extends the KLSH to the scheme

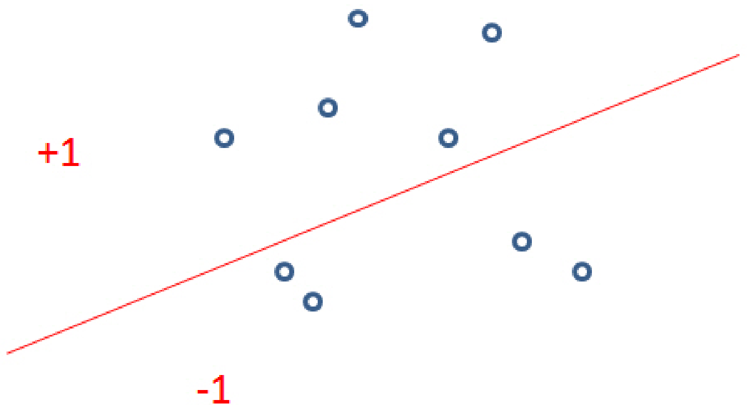


Figure 2.1. LSH example: one bit.

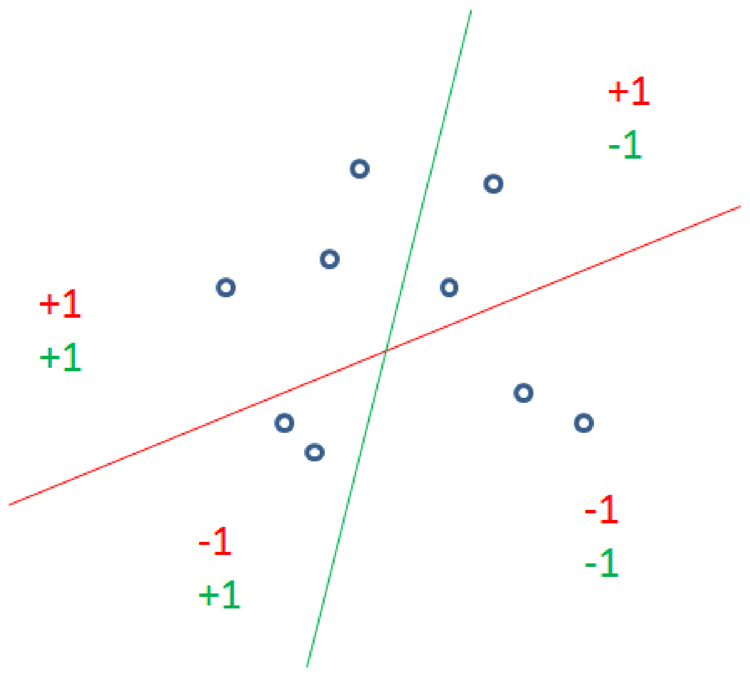


Figure 2.2. LSH example: two bits.

of Boosting Multi-Kernel Locality-Sensitive Hashing (BMKLSH) that improves the retrieval performance of KLSH by making use of multiple kernels.

Spectral Hashing (SH)

Spectral Hashing (SH) [19] method tries to seek hashing codes that satisfy (1) easily computed for a novel input query (2) requires a small number of bits to code the full dataset and (3) maps similar data examples to similar binary codes. For a code to be efficient, it also require that each bit has a 50% chance of being $+1$ or -1, and that different bits are independent of each other. Among all codes that have this property, it seek the ones where the average Hamming distance between similar examples is minimal.

Let y_i be the hashing code with k bits for data example x_i. $\boldsymbol{S}_{i,j}$ be the affinity matrix. Since it is assuming the inputs are embedded in R^d so that Euclidean distance correlates with similarity, it use $\boldsymbol{S}_{i,j} = exp(-\|x_i - x_j\|^2/\sigma^2)$, where σ is the scaling parameter that defines the distance in R^d. Using this notation, the average Hamming distance between similar neighbors can be written as $\sum_{i,j} \boldsymbol{S}_{i,j}\|y_i - y_j\|^2$. Then the optimal hashing codes can be obtain by solving the following problem:

$$\begin{aligned} \min_{y} \quad & \sum_{i,j} \boldsymbol{S}_{i,j}\|y_i - y_j\|^2 \\ s.t. \quad & y_i \in \{-1, +1\}^k \\ & \sum_i y_i = 0 \\ & \frac{1}{n}\sum_i y_i y_i^T = \boldsymbol{I} \end{aligned} \tag{2.2}$$

where the constraint $\sum_i y_i = 0$ is the bit balance constraint which requires each bit to appear 50% of the time, and the constraint $\frac{1}{n}\sum_i y_i y_i^T = \boldsymbol{I}$ requires the bits to be uncorrelated or orthogonal. The objective function incurs a heavy penalty if two similar data examples are mapped far away, which preserves the similarity between data examples. Obtaining exact solution of the above problem turns out to be NP-hard by the following theorem.

Theorem 2.2.1 *For k=1, solving problem 2.2 is equivalent to balanced graph partitioning and is NP hard.*

Proof Consider an undirected graph whose vertices are the data points and where the weight between data i and j is given by $\boldsymbol{S}_{i,j}$. Consider a code with a single bit. The bit partitions the graph into two equal parts (A, B), vertices where the bit is on and vertices where the bit is off. For a single bit, $\sum_{i,j} \boldsymbol{S}_{i,j}\|y_i - y_j\|^2$ is simply the weight of the edges cut by the partition: $cut(A, B) = \sum_{i \in A, j \in B} \boldsymbol{S}_{i,j}$. Thus problem 2.2 is equivalent to minimizing cut(A, B) with the requirement that $\|A\| = \|B\|$ which is known to be NP hard [74]. ■

The problem in Eqn.2.2 is then approximated using spectral relaxation. By introducing a $n \times k$ matrix $\boldsymbol{Y}$ whose *i-th* row is y_i^T and a diagonal $n \times n$ matrix $\boldsymbol{D}_{i,i} = \sum_j \boldsymbol{S}_{i,j}$ and relaxing the binary and bit balance constraint, the problem can be rewritten as:

$$\begin{aligned} \min_{y} \quad & tr(\boldsymbol{Y}^T(\boldsymbol{D}-\boldsymbol{S})\boldsymbol{Y}) \\ s.t. \quad & \boldsymbol{Y}^T\boldsymbol{Y} = \boldsymbol{I} \end{aligned} \tag{2.3}$$

Then the problem becomes an easy problem whose solutions are simply the k eigenvectors of $\boldsymbol{D}-\boldsymbol{S}$ with minimal eigenvalue (after excluding the trivial eigenvector 1 which has eigenvalue 0).

Self-Taught Hashing (STH)

Self Taught Hashing (STH) [22] usually generally provides more effective hashing solutions than LSH and SH. STH combines an unsupervised learning step with a supervised learning step to learn hashing codes.

In the unsupervised learning step, STH constructs a similarity graph using a fix number of nearest neighbors for the given dataset, and then embeds all the data examples into a k dimensional space through spectral analysis similar to SH, and finally uses simple thresholding to obtain the binary hashing code for each data example. This step is exactly the same as in Eqn.2.3 of SH method.

In the supervised learning step, a set of k SVM classifiers are trained based on existing documents and their binary hashing codes learned from the previous step. Then, the k classifiers can be used to generate the hashing codes for the query documents as a classification problem. STH does not assume that data are uniformly distributed in a hyper-rectangle as requested by SH, which is often too restrictive for real world applications. STH often generates more effective hashing codes than SH.

Semi-Supervised Hashing (SSH)

The work in [28, 29] proposes a Semi-Supervised Hashing (SSH) approach for incorporating the pairwise relationships between data examples into learning hashing function. More precisely, these pairwise similarity constraints are also called Must-Link and Cannot-Link, which could be partially generated from tags. For example, a Must-Link is created when two data examples share a common tag, i.e., $(x_i, x_j) \in M$, and a Cannot-Link is created when two examples share no tag , i.e., $(x_i, x_j) \in C$. Their basic motivation is that the hashing codes of data example pairs with Must-Link should be as close as possible, while the hashing codes of example pairs with Cannot-Link should be as different as possible. This motivation is then incorporated into the objective function for learning the hashing codes.

In SSH method, it first define a pairwise constraint matrix $\boldsymbol{S}$ incorporating the pairwise link information as:

$$\boldsymbol{S}_{ij} = \begin{cases} 1, \; if \; (x_i, x_j) \in M \\ -1, \; if \; (x_i, x_j) \in C \\ 0, \qquad otherwise \end{cases} \tag{2.4}$$

Then the supervised part of the objective function can be represented as:

$$J(\boldsymbol{W}) = tr\left(sgn(\boldsymbol{W}^T\boldsymbol{X})\boldsymbol{S}sgn(\boldsymbol{W}^T\boldsymbol{X})^T\right) \tag{2.5}$$

The unsupervised part of the objective is constructed based on the maximum entropy principle that a binary bit that gives balanced partitioning of $\boldsymbol{X}$ provides maximum information. Then it shows that maximum entropy partitioning is equivalent to maximizing the variance of a bit and thus the unsupervised term is defined as:

$$R(\boldsymbol{W}) = \sum_k var[sgn(w_k^T x)] \tag{2.6}$$

Maximizing the above function with respect to $\boldsymbol{W}$ is still hard due to its non-differentiability. To overcome this problem, it shows that the maximum variance of a hash function is lower-bounded by the scaled variance of the projected data.

Theorem 2.2.2 *The maximum variance of a hash function is lower-bounded by the scaled variance of the projected data, i.e., $max\ var[sgn(w_k^T x)] \geq \alpha\ var[w_k^T x]$*

Proof Suppose $\|x_i\|^2 \leq \beta$. Since we assume $\|w_k\|^2 = 1$, from Cauchy-Schwarz inequality,
$\|w_k^T x\|^2 \leq \|w_k\|^2 \|x\|^2 \leq \beta = \beta \|sgn(w_k^T x)\|^2$
$\Rightarrow E[\|sgn(w_k^T x)\|^2] \geq \frac{1}{\beta} E[\|w_k^T x\|^2]$
$\Rightarrow max\ var[sgn(w_k^T x)] \geq \alpha\ var[w_k^T x]$
since the data is zero centered, i.e., $E[w_k^T x] = 0$, and for maximum bit variance $E[sgn(w_k^x)] = 0$. ■

Given the above theorem, it uses the lower bound on the maximum variance of a hash function as a regularizer, which is easy to optimize:

$$R(\boldsymbol{W}) = \frac{1}{\beta} \sum_k E[\|w_k^T x\|^2] = \frac{1}{n\beta} tr\left(\boldsymbol{W}^T \boldsymbol{X} \boldsymbol{X}^T \boldsymbol{W}\right) \tag{2.7}$$

Combining equations 2.5 and 2.7 and relaxing the sign function, the overall semi-supervised objective function is given as:

$$\begin{aligned} J(\boldsymbol{W}) &= tr\left(\boldsymbol{W}^T \boldsymbol{X} \boldsymbol{S} \boldsymbol{X}^T \boldsymbol{W}\right) + \gamma tr\left(\boldsymbol{W}^T \boldsymbol{X} \boldsymbol{X}^T \boldsymbol{W}\right) \\ &= tr\left(\boldsymbol{W}^T (\boldsymbol{X} \boldsymbol{S} \boldsymbol{X}^T + \boldsymbol{X} \boldsymbol{X}^T) \boldsymbol{W}\right) = tr\left(\boldsymbol{W}^T \boldsymbol{M} \boldsymbol{W}\right) \end{aligned} \tag{2.8}$$

The learning of optimal projections $\boldsymbol{W}$ becomes a typical eigen-problem, which can be easily solved by doing an eigenvalue decomposition on matrix $\boldsymbol{M}$. Mathematically, it is very similar to finding maximum variance direction using PCA except that the original covariance matrix gets 'adjusted' by another matrix arising from the labeled data. A sequential projection method is also porposed to solve the resulting optimization problem.

The SSH has shown promising results for improving hashing effectiveness by leveraging the pairwise information, but there are several limitations for SSH. Firstly, the SSH method only utilizes the pairwise similarity constraints as the summary of tag information, which is suboptimal with respect to the complete information in the tags.

Secondly, the pairwise link information may not be accurately generated when tags are missing, incomplete or mismatched, which is often the case for many real world applications. Furthermore, SSH also directly works in the original keyword feature space for modeling content similarity of documents. These problems may potentially limit the performance of the hashing methods based on pairwise constraints.

Kernelized Supervised Hashing (KSH)

Kernelized Supervised Hashing (KSH) [30] method employs kernel trick similar to that in Kernelized Locality-Sensitive Hashing (KLSH) [75] algorithm. The hashing function is defined as follows:

$$y = sgn(\sum_{j=1}^{m} \kappa(x_{(j)}, x)a_j - b) \tag{2.9}$$

where $x_{(1)}, \ldots, x_{(m)}$ are m samples uniformly selected at random from X, $a_j \in R$ is the coefficient, and $b \in R$ is the bias. Note that m is fixed to a constant much smaller than the data set size n in order to maintain fast hashing. Based on the balancing criterion, the bias b is set to be the median as $b = \sum_{i=1}^{n} \sum_{j=1}^{m} \kappa(x_{(j)}, x_i)a_j/n$. Then the hashing function can be rewritten as:

$$y = sgn(a^T \hat{\kappa}(x)) \tag{2.10}$$

where $a = [a_1, \ldots, a_m]^T$. The objective function of KSH is given as:

$$\min_{A} \ \|\frac{1}{k} sgn(\boldsymbol{KA}) sgn(\boldsymbol{KA})^T - \boldsymbol{S}\|_F^2 \tag{2.11}$$

Here $\boldsymbol{S}$ is the pairwise constraint matrix defined in Eqn.2.4. A greedy optimization method is utilized to solve the above problem. In KSH method, it also proposes to use a sigmoid smoothing function to approximate the $sgn(x)$.

The time complexities for training KSH are both bounded by $O(nmk + l^2mk + m^2lk + m^3k)$ which scales linearly with n given $n > l > m$.

Canonical Correlation Analysis with Iterative Quantization (CCA-ITQ)

The Canonical Correlation Analysis with Iterative Quantization (CCA-ITQ) method [35, 36] consists of two main components. The first part is the Canonical Correlation Analysis which treats the tags and features of data examples as two views and extracts a common latent space from these two views. Denote the tag matrix as $\boldsymbol{T} \in \{0,1\}^{n \times l}$. The goal of CCA is to find projection directions w_k and u_k for feature and tag vectors to maximize the correlation between the projected data $\boldsymbol{X}w_k$ and $\boldsymbol{T}u_k$ as:

$$\begin{gathered} \max_{w,u} \; w_k^T \boldsymbol{X}^T \boldsymbol{T} u_k \\ s.t. \; w_k^T \boldsymbol{X}^T \boldsymbol{X} w_k = 1, \;\; u_k^T \boldsymbol{T}^T \boldsymbol{T} u_k = 1. \end{gathered} \tag{2.12}$$

Maximizing the above objective function involves solving the following generalized eigenvalue problem to get w_k:

$$\boldsymbol{X}^T \boldsymbol{T} (\boldsymbol{T}^T \boldsymbol{T} + \rho I)^{-1} \boldsymbol{T}^T \boldsymbol{X} w_k = \lambda_k^2 (\boldsymbol{X}^T \boldsymbol{X} + \rho I) w_k \tag{2.13}$$

in which ρ is a small regularization constant used to prevent a trivial solution. The leading generalized eigenvectors of the above equation then give us a sequence of orthogonal w_k directions that span the solution space, just as for PCA. Note that once we have w_k, we can also solve for the corresponding u_k, but in our case, we only care about the projection directions in the data space, since we assume that tag information will be unavailable at test time.

The second component is iterative quantization. Assume we obtain the projection direction from the first part of CCA-ITQ denoted as $\boldsymbol{W}$. Then the relaxed solution before binarization is $\boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}$. By the observation that if $\boldsymbol{W}$ is the optimal solution of Eqn.2.12, then so is $\hat{\boldsymbol{W}} = \boldsymbol{W}\boldsymbol{R}$ for any orthogonal $k \times k$ matrix $\boldsymbol{R}$. Therefore, we are free to orthogonally transform the projected data $\boldsymbol{V}$ in such a way as to minimize the quantization loss:

$$Q(\boldsymbol{Y}, \boldsymbol{R}) = \|\boldsymbol{Y} - \boldsymbol{V}\boldsymbol{R}\|_F^2 \tag{2.14}$$

Intuitively, we seek binary codes that are close to some orthogonal transformation of the relaxed solution. The orthogonal transformation not only preserves the optimality of the relaxed solution but also provides us more flexibility to achieve better hashing codes with low quantization error. Then the optimal orthogonal matrix $\boldsymbol{R}$ and binarization codes $\boldsymbol{Y}$ can be obtained by iteratively optimizing the above equation.

2.3 Algorithm

2.3.1 Problem Definition and Notation

Assume there are total n training examples. Let us denote their features as: $\boldsymbol{X} = \{x_1, x_2, \ldots, x_n\} \in R^{d \times n}$, where d is the dimensionality of the feature. Denote the observed/partial tags as: $\boldsymbol{T} = \{t_1, t_2, \ldots, t_l\} \in \{0,1\}^{n \times l}$, where l is the total number of possible tags for each data example. A label $\boldsymbol{T}_{ij} = 1$ means the *i-th* data example is associated with the *j-th* tag, while a label 0 means a missing tag or the tag is not associated with that example. The goal is to obtain a linear hashing function $f : R^d \rightarrow \{-1,1\}^b$, which maps data examples $\boldsymbol{X}$ to their binary hashing codes $\boldsymbol{Y} = \{y_1, y_2, \ldots, y_n\} \in \{-1,1\}^{b \times n}$ (b is the binary code length). The linear hashing function is defined as:

$$y_i = f(x_i) = sgn(\boldsymbol{W}^T x_i) \tag{2.15}$$

where $\boldsymbol{W} \in R^{d \times b}$ is the coefficient matrix representing the hashing function and sgn is the sign function. $y_i \in \{-1,1\}^b$ is the binary hashing code[1] of x_i. Without loss of generality, we assume the data are zero-centered, thus there is no bias in Eqn.2.15.

The objective function of Semi-Supervised Tag Hashing (SSTH) is composed of two components: (1) Tag consistency term, the supervised part which ensures that the hashing codes are consistent with the observed tags. (2) Similarity preservation term, the unsupervised part which aims at preserving the data similarity in the learned hashing codes. In the rest of this section, we will present the formulation of these two

[1]We generate hashing bits as $\{-1,1\}$, which can be simply converted to $\{0,1\}$ valued hashing codes.

components respectively. Then in the next section, we will describe the optimization algorithm together with a scheme that can further improve the quality of the hashing function by minimizing the quantization error.

2.3.2 Tag Consistency

In many real-world applications, data examples are often associated with various tags. These tag information provides useful supervised knowledge in learning effective hashing function. Therefore, it is necessary to design a scheme for leveraging tag information. There are three main challenges to incorporate tags. (1) We have no knowledge about how tags are related to the hashing bits. Therefore, we need to explore the correlation between them in order to bridge tags with hashing codes. (2) Tags tend to be noisy and missing, and we need to deal with the situation of incomplete tags. (3) We need to deal with the ambiguity of semantically similar tags (e.g., ‘human’ versus ‘people’, ‘car’ versus ‘automobile’).

In this dissertation, we propose to model the consistency between observed tags and hashing codes via matrix factorization using the latent factor model [76, 77]. Semantically similar tags are represented by different tags (e.g., ‘human’ and ‘people’ are two distinct tags) in our model and we will discuss how this issue can be addressed later. In the latent factor model, a set of latent variables c_j for each tag t_j is first introduced to model the correlation between tags and hashing bits, where $j \in \{1, 2, \dots, l\}$ and c_j is a $b \times 1$ vector indicating the correlation between the j-th tag and the b hashing bits. Then a tag consistency component can be naturally formulated as:

$$\sum_{i=1}^{n} \sum_{j=1}^{l} \|\boldsymbol{T}_{ij} - y_i^T c_j\|^2 + \alpha \sum_{j=1}^{l} \|c_j\|^2 \tag{2.16}$$

here $\boldsymbol{T}_{ij}$ is the label of j-th tag on the i-th data example. Intuitively, $y_i^T c_j$ can be essentially viewed as a weighted sum that indicates how the j-th tag is related to the i-th data example, and this weighted sum should be consistent with the observed label $\boldsymbol{T}_{ij}$ as much as possible. $\sum_{j=1}^{l} \|c_j\|^2$ is a regularizer to avoid overfitting and α is the trade-off parameter. In this way, the latent correlation between tags and hashing bits can be learned by ensuring this consistency term.

The ambiguity issue for semantically similar tags is addressed by the latent factor model since these tags are often associated with common data examples, and thus the learned corresponding latent variables will be similar by ensuring the tag consistency term. This can also be explained by the formulation above, which ensures the consistency between tag t and $\boldsymbol{Y}c$ (i.e., $t \approx \boldsymbol{Y}c$). Therefore, if two tags t_i and t_j are associated with similar set of data examples (indicating these two tags are semantically similar), their corresponding c_i and c_j will be close as well. In the extreme case, if two tags appear in exactly the same set of examples, their latent variables will be identical.

An importance matrix $\boldsymbol{I} \in R^{n \times l}$ is introduced to deal with the missing tag problem. As mentioned above, $\boldsymbol{T}_{ij} = 0$ can be interpreted into two ways: *j-th* tag on the *i-th* data example is either missing or not related. Therefore, we set $\boldsymbol{I}_{ij} = u$ with a higher value when $\boldsymbol{T}_{ij} = 1$ than $\boldsymbol{I}_{ij} = v$ when $\boldsymbol{T}_{ij} = 0$, where u and v are parameters satisfying $u > v > 0$[2]. Then the whole tag consistency term becomes:

$$\sum_{i=1}^{n} \sum_{j=1}^{l} \boldsymbol{I}_{ij} \| \boldsymbol{T}_{ij} - y_i^T c_j \|^2 + \alpha \sum_{j=1}^{l} \| c_j \|^2 \tag{2.17}$$

By substituting Eqn.2.15, the above equation can be rewritten as a compact matrix form:

$$\| \boldsymbol{I}^{\frac{1}{2}} \cdot (\boldsymbol{T} - sgn(\boldsymbol{X}^T \boldsymbol{W}) \boldsymbol{C}) \|_F^2 + \alpha \| \boldsymbol{C} \|_F^2 \tag{2.18}$$

where $\boldsymbol{I}^{\frac{1}{2}}$ is the element-wise square root matrix of $\boldsymbol{I}$, and $\cdot$ is the element-wise matrix multiplication. $\| \|_F$ is the matrix $Frobenius$ norm and $\boldsymbol{C}$ is a $b \times l$ correlation matrix bridging the hashing codes with tags. By minimizing this term, the consistency between tags and the learned hashing codes is ensured.

[2]In our experiments, we set the importance parameters u=1 and v=0.01.

2.3.3 Similarity Preservation

One of the key problems in hashing algorithms is similarity preserving, which indicates that similar data examples should be mapped to similar hashing codes within a short Hamming distance. Preserving the original data similarity is important in learning effective hashing function especially when training tags are limited or even not available. The Hamming distance between two binary codes y_i and y_j can be calculated as $\frac{1}{4}\|y_i - y_j\|^2$. To measure the similarity between data examples represented by the binary hashing codes, one natural way is to minimize the weighted average Hamming distance as follows:

$$\sum_{i,j} \boldsymbol{S}_{ij} \|y_i - y_j\|^2 \tag{2.19}$$

Here, $\boldsymbol{S}$ is the similarity matrix which is calculated based on the data features. To meet the similarity preservation criterion, we seek to minimize this quantity, because it incurs a heavy penalty if two similar examples are mapped far away.

There are many different ways of defining the similarity matrix $\boldsymbol{S}$. In SH [19], the authors used the global similarity structure of all data pairs, while in [22], the local similarity structure, i.e., k-nearest-neighborhood, is used. In this dissertation, we use the local similarity, due to its nice property in many machine learning applications. In particular, the corresponding weights are computed by Gaussian function as follows:

$$\boldsymbol{S}_{ij} = \begin{cases} e^{-\frac{\|x_i - x_j\|^2}{\sigma_{ij}^2}}, & if\ x_i \in N_k(x_j)\ \ or\ \ x_j \in N_k(x_i) \\ 0, & otherwise \end{cases} \tag{2.20}$$

The variance σ_{ij} is determined automatically by local scaling [71], and $N_k(x)$ represents the set of k-nearest-neighbors of the data example x.

By introducing a diagonal $n \times n$ matrix $\boldsymbol{D}$, whose entries are given by $\boldsymbol{D}_{ii} = \sum_{j=1}^{n} \boldsymbol{S}_{ij}$. Eqn.2.19 can be rewritten as:

$$\begin{aligned} tr\left(\boldsymbol{Y}(\boldsymbol{D} - \boldsymbol{S})\boldsymbol{Y}^T\right) &= tr\left(\boldsymbol{Y}\boldsymbol{L}\boldsymbol{Y}^T\right) \\ &= tr\left(sgn(\boldsymbol{W}^T\boldsymbol{X})\boldsymbol{L}sgn(\boldsymbol{X}^T\boldsymbol{W})\right) \end{aligned} \tag{2.21}$$

where $\boldsymbol{L}$ is called graph *Laplacian* [19,78] and $tr()$ is the matrix trace function. The similarity preservation term plays an important role in hashing function learning especially when the supervised information is limit due to noisy and incomplete tags. By minimizing this term, the similarity between different data examples can be preserved in the learned hashing codes.

2.3.4 Overall Objective and Optimization

The entire objective function consists of two components: the tag consistency term in Eqn.2.18 and the data similarity preservation term given in Eqn.2.21 as follows:

$$\begin{aligned}\min_{\boldsymbol{W},\boldsymbol{C}} \quad & \|\boldsymbol{I}^{\frac{1}{2}} \cdot (\boldsymbol{T} - sgn(\boldsymbol{X}^T\boldsymbol{W})\boldsymbol{C})\|_F^2 + \alpha\|\boldsymbol{C}\|_F^2 \\ & +\gamma\ tr\left(sgn(\boldsymbol{W}^T\boldsymbol{X})\boldsymbol{L}sgn(\boldsymbol{X}^T\boldsymbol{W})\right) \\ s.t. \quad & \boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{I_b}\end{aligned} \tag{2.22}$$

where α and γ are trade-off parameters to balance the weights among the terms. The hard orthogonality constraints enforce the hashing bits to be uncorrelated with each other and therefore the learned hashing codes can hold least redundant information.

Relaxation

Directly minimizing the objective function in Eqn.2.22 is intractable since it is a constrained integer programming, which is proven to be NP-hard to solve. Therefore, we first convert the hard constraints into a soft penalty term by adding a regularizer to the objective and use the signed magnitude instead of the sign function as suggested in [29,30]. Then the relaxed objective function becomes:

$$\begin{aligned}\min_{\tilde{\boldsymbol{W}},\boldsymbol{C}} \quad & \|\boldsymbol{I}^{\frac{1}{2}} \cdot (\boldsymbol{T} - \boldsymbol{X}^T\tilde{\boldsymbol{W}}\boldsymbol{C})\|_F^2 + \alpha\|\boldsymbol{C}\|_F^2 \\ & +\gamma\ tr\left(\tilde{\boldsymbol{W}}^T\tilde{\boldsymbol{L}}\tilde{\boldsymbol{W}}\right) + \beta\|\tilde{\boldsymbol{W}}^T\tilde{\boldsymbol{W}} - \boldsymbol{I_b}\|_F^2\end{aligned} \tag{2.23}$$

where $\tilde{\boldsymbol{L}} \equiv \boldsymbol{X}\boldsymbol{L}\boldsymbol{X}^T$ and can be pre-computed. However, even after the relaxation, the objective function is still difficult to optimize since $\tilde{\boldsymbol{W}}$ and $\boldsymbol{C}$ are coupled together and it is non-convex with respect to $\tilde{\boldsymbol{W}}$ and $\boldsymbol{C}$ jointly. We propose to split the optimization problem into two simpler sub-problems. The idea is that given $\tilde{\boldsymbol{W}}$, $\boldsymbol{C}$ has a closed form solution with respect to $\tilde{\boldsymbol{W}}$ (see details in $SP2$ below). Thus we split the relaxed objective with respect to $\tilde{\boldsymbol{W}}$ and $\boldsymbol{C}$ and solve the two sub-problems iteratively using coordinate descent method. The two sub-problems are given as:

$$\begin{aligned} SP1 : \min_{\tilde{\boldsymbol{W}}} \quad & \|\boldsymbol{I}^{\frac{1}{2}} \cdot (\boldsymbol{T} - \boldsymbol{X}^T\tilde{\boldsymbol{W}}\boldsymbol{C})\|_F^2 + \gamma\ tr\left(\tilde{\boldsymbol{W}}^T\tilde{\boldsymbol{L}}\tilde{\boldsymbol{W}}\right) \\ & +\beta\|\tilde{\boldsymbol{W}}^T\tilde{\boldsymbol{W}} - \boldsymbol{I_b}\|_F^2 \end{aligned} \tag{2.24}$$

$$SP2 : \min_{\boldsymbol{C}} \quad \|\boldsymbol{I}^{\frac{1}{2}} \cdot (\boldsymbol{T} - \boldsymbol{X}^T\tilde{\boldsymbol{W}}\boldsymbol{C})\|_F^2 + \alpha\|\boldsymbol{C}\|_F^2 \tag{2.25}$$

$SP1$ is still non-convex, but it is smooth and differentiable which enables gradient descent methods for efficient optimization. The gradient of $SP1$ is calculated as follows:

$$\begin{aligned} \frac{\partial SP1}{\partial \tilde{\boldsymbol{W}}} &= 2\boldsymbol{X}(\boldsymbol{I} \cdot (\boldsymbol{X}^T\tilde{\boldsymbol{W}}\boldsymbol{C} - \boldsymbol{T}))\boldsymbol{C}^T + 2\gamma\tilde{\boldsymbol{L}}\tilde{\boldsymbol{W}} \\ & +4\beta\tilde{\boldsymbol{W}}(\tilde{\boldsymbol{W}}^T\tilde{\boldsymbol{W}} - \boldsymbol{I_b}) \end{aligned} \tag{2.26}$$

With this obtained gradient, L-BFGS quasi-Newton method [79] is applied to solve $SP1$.

By taking the derivative of $SP2$ w.r.t. $\boldsymbol{C}$ and setting it to $\mathbf{0}$, we can obtain the closed form solution of $SP2$ below:

$$\begin{aligned} \frac{\partial SP2}{\partial \boldsymbol{C}} &= 2\tilde{\boldsymbol{W}}^T\boldsymbol{X}(\boldsymbol{I} \cdot (\boldsymbol{X}^T\tilde{\boldsymbol{W}}\boldsymbol{C} - \boldsymbol{T})) + 2\alpha\boldsymbol{C} = \mathbf{0} \\ \Rightarrow c_j &= (\tilde{\boldsymbol{W}}^T\boldsymbol{X}\boldsymbol{I}_j\boldsymbol{X}^T\tilde{\boldsymbol{W}} + \alpha\boldsymbol{I_b})^{-1}\tilde{\boldsymbol{W}}^T\boldsymbol{X}\boldsymbol{I}_j\boldsymbol{T}_j \end{aligned} \tag{2.27}$$

where $\boldsymbol{I}_j$ is a $n \times n$ diagonal matrix with $\boldsymbol{I}_{ij}, i = 1, 2, \ldots, n$ as its diagonal elements and $\boldsymbol{T}_j = (\boldsymbol{T}_{ij}), i = 1, 2, \ldots, n$ is a $n \times 1$ label vector of *j-th* tag.

We alternate the process of updating $\tilde{\boldsymbol{W}}$ and $\boldsymbol{C}$ for several iterations to find a locally optimal solution. In practice, we have found that a reasonable small number of iterations (i.e., 30 in our experiments) can achieve good performance.

Orthogonal Transformation

After obtaining the optimal hashing function $\tilde{\boldsymbol{W}}$ for the relaxation, the hashing codes $\boldsymbol{Y}$ can be generated using Eqn.2.15. It is obvious that the quantization error can be measured as $\|\boldsymbol{Y} - \tilde{\boldsymbol{W}}^{\boldsymbol{T}}\boldsymbol{X}\|_F^2$. Inspired by [36], we propose to further improve the hashing function by minimizing this quantization error using an orthogonal transformation. We first prove the following orthogonal invariant theorem.

Theorem 2.3.1 *Assume $\boldsymbol{Q}$ is a $k \times k$ orthogonal matrix, i.e., $\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I_b}$. If $\tilde{\boldsymbol{W}}$ and $\boldsymbol{C}$ are an optimal solution to the relaxed problem in Eqn.2.23, then $\tilde{\boldsymbol{W}}\boldsymbol{Q}$ and $\boldsymbol{Q}^T\boldsymbol{C}$ are also an optimal solution.*

Proof By substituting $\tilde{\boldsymbol{W}}\boldsymbol{Q}$ and $\boldsymbol{Q}^T\boldsymbol{C}$ into Eqn.2.23, we have:
$\|\boldsymbol{I}^{\frac{1}{2}} \cdot (\boldsymbol{T} - \boldsymbol{X}^T\tilde{\boldsymbol{W}}\boldsymbol{Q}\boldsymbol{Q}^T\boldsymbol{C})\|_F^2 = \|\boldsymbol{I}^{\frac{1}{2}} \cdot (\boldsymbol{T} - \boldsymbol{X}^T\tilde{\boldsymbol{W}}\boldsymbol{C})\|_F^2$,
$tr\left((\tilde{\boldsymbol{W}}\boldsymbol{Q})^T\tilde{\boldsymbol{L}}\tilde{\boldsymbol{W}}\boldsymbol{Q}\right) = tr\left(\boldsymbol{Q}^T\tilde{\boldsymbol{W}}^T\tilde{\boldsymbol{L}}\tilde{\boldsymbol{W}}\boldsymbol{Q}\right) = tr\left(\tilde{\boldsymbol{W}}^T\tilde{\boldsymbol{L}}\tilde{\boldsymbol{W}}\right)$, $\|\boldsymbol{Q}^T\boldsymbol{C}\|_F^2 = \|\boldsymbol{C}\|_F^2$
and $\|(\tilde{\boldsymbol{W}}\boldsymbol{Q})^T\tilde{\boldsymbol{W}}\boldsymbol{Q} - \boldsymbol{I_b}\|_F^2 = \|\boldsymbol{Q}^T(\tilde{\boldsymbol{W}}^T\tilde{\boldsymbol{W}} - \boldsymbol{I_b})\boldsymbol{Q}\|_F^2 = \|\tilde{\boldsymbol{W}}^T\tilde{\boldsymbol{W}} - \boldsymbol{I_b}\|_F^2$.
Thus, the value of the objective function in Eqn.2.23 does not change by the orthogonal transformation. ■

According to the above theorem, we propose to find a better hashing function $\boldsymbol{W} = \tilde{\boldsymbol{W}}\boldsymbol{Q}$ by minimizing the quantization error between the binary hashing codes and the orthogonal transformation of the relaxed solution as follows:

$$\begin{aligned} &\min_{\boldsymbol{Y},\boldsymbol{Q}} \|\boldsymbol{Y} - (\tilde{\boldsymbol{W}}\boldsymbol{Q})^T\boldsymbol{X}\|_F^2 \\ s.t. \quad &\boldsymbol{Y} \in \{-1,1\}^{k\times n}, \quad \boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I_b} \end{aligned} \tag{2.28}$$

Intuitively, we seek binary codes that are close to some orthogonal transformation of the relaxed solution. The orthogonal transformation not only preserves the optimality of the relaxed solution but also provides us more flexibility to achieve better hashing codes with low quantization error. The idea of orthogonal transformation is also utilized in ITQ [36]. However, ITQ method is not designed for incorporating partial tag information into learning effective hashing function and it does not preserve the local similarities among data examples. The above optimization problem can be solved by minimizing Eqn.2.28 with respect to $\boldsymbol{Y}$ and $\boldsymbol{Q}$ alternatively as follows:

Fix Q **and update** Y. The closed form solution can be expressed as:

$$\boldsymbol{Y} = sgn\left((\tilde{\boldsymbol{W}}\boldsymbol{Q})^T\boldsymbol{X}\right) = sgn(\boldsymbol{W}^T\boldsymbol{X}) \tag{2.29}$$

which is identical with our linear hashing function in Eqn.2.15.

Fix Y **and update** Q. The objective function becomes:

$$\min_{\boldsymbol{Q}^T\boldsymbol{Q}=\boldsymbol{I_b}} \|\boldsymbol{Y} - \boldsymbol{Q}^T\tilde{\boldsymbol{W}}^T\boldsymbol{X}\|_F^2 \tag{2.30}$$

In this case, the objective function is essentially the classic Orthogonal Procrustes problem [80], which can be solved efficiently by singular value decomposition using the following theorem (we refer to [80] for the detailed proof).

Theorem 2.3.2 *Let* $\boldsymbol{S}\boldsymbol{\Lambda}\boldsymbol{V}^T$ *be the singular value decomposition of* $\boldsymbol{Y}\boldsymbol{X}^T\tilde{\boldsymbol{W}}$. *Then* $\boldsymbol{Q} = \boldsymbol{V}\boldsymbol{S}^T$ *minimizes the objective function in Eqn.2.30.*

We then perform the above two steps alternatively to obtain the optimal hashing codes and the orthogonal transform matrix. In our experiments, we find that the algorithm usually converges in about 40∼60 iterations. The full learning algorithm is described in Table 2.1.

2.3.5 Discussion

This section discusses the connections between the proposed SSTH approach with several previous hashing methods. It also provides some analysis on the time complexity of the optimization algorithm. The unsupervised hashing methods Spectral Hashing (SH) [19] and Self-Taught Hashing (STH) [22] can be viewed as a reformulation of the unsupervised part of SSTH. In other words, both these methods only consider preserving the original similarity between data examples but not leveraging the supervised information contained in tags while our SSTH approach also ensures the tag consistency. For some supervised hashing methods such as

Table 2.1.
Semi-Supervised Tag Hashing (SSTH)

```
Input: Data examples $\boldsymbol{X}$, Observed tags $\boldsymbol{T}$ and trade-off parameters
Output: Hashing function $\boldsymbol{W}$, Hashing codes $\boldsymbol{Y}$ and Correlation $\boldsymbol{C}$
Initialize $\boldsymbol{C} = \mathbf{0}$ and $\boldsymbol{Q} = \boldsymbol{I_b}$, Calculate $\tilde{\boldsymbol{L}}$.
Repeat
  Optimize $SP1$ using Eqn.2.26 and update $\tilde{\boldsymbol{W}}$
  Optimize $SP2$ using Eqn.2.27 and update $\boldsymbol{C}$
Until the solution converges
Repeat
  Update $\boldsymbol{Y}$ using Eqn.2.29
  Update $\boldsymbol{Q} = \boldsymbol{V}\boldsymbol{S}^T$ according to Theorem 2.
Until the solution converges
```

semi-supervised hashing (SSH) [29], supervised hashing with kernels (KSH) [30] and ranking-based supervised hashing (RSH) [31], they only consider pairwise or listwise supervised information for learning hashing codes. The pairwise or listwise constraints are very coarse representation of tag information, which are also subject to noise within tags. On the other side, the proposed SSTH approach fully incorporates tag knowledge in a more desired manner by directly exploring the semantic correlation between tags and hashing bits.

For the training complexity, the optimization algorithm of SSTH consists of two main loops. In the first loop, we iteratively solve $SP1$ and $SP2$ to obtain the optimal solution, where the time complexities for solving $SP1$ and $SP2$ are bounded by $O(nlb+nbd+nb^2)$ and $O(nb^2+nbl)$ respectively. The second loop iteratively optimizes the binary hashing codes and the orthogonal transformation matrix, where the time complexities for updating $\boldsymbol{Y}$ and $\boldsymbol{Q}$ are bounded by $O(nb^2 + nbd + b^3)$. Moreover, both two loops take less than 60 iterations to converge as mentioned before. Thus, the total time complexity of the learning algorithm is bounded by $O(nlb+nbd+nb^2+b^3)$, which scales linearly with n given $n \gg l > d > b$. For each query, the hashing time is constant $O(db)$. In our implementation, the stop criteria for $SP1$ is that the difference between two consecutive objectives is less than $\epsilon=10^{-5}$. And we also set the maximum number of iterations for both loops to 100.

2.3.6 Extension using Topic Modeling

In the application of document similarity search, document similarity in the original keyword feature space is used as guidance for generating hashing codes, which may not fully reflect the semantic relationship. For example, two documents in the same topic may have low document content similarity in keyword space due to the vocabulary gap, although their semantic similarity can be high. Based on this observation, features from topic modeling are used to measure the semantic similarity between documents instead of features from the original keyword space. Topic modeling algorithms (e.g., [81, 82]) are used to discover a set of 'topics' from a large collection of documents and provide an interpretable low-dimensional representation of the documents associated with the topics. Topic modeling has been widely used in many information retrieval applications such as document clustering and classification. Here we exploit the Latent Dirichlet Allocation (LDA) [83] approach of topic modeling to extract k latent topics from the document corpus. Each document x_j corresponds to a distribution θ_j over the topics where two semantically similar documents have similar topic distributions. In this way, document semantic similarity is preserved in the extracted topic distributions $\boldsymbol{\theta}$. Since we require the hashing codes to reflect the topic distributions, a document similarity preservation component can be naturally defined as follows instead of equation 2.21:

$$\sum_{j=1}^{n} ||y_j - \theta_j||^2 = ||\boldsymbol{Y} - \boldsymbol{\theta}||^2 \tag{2.31}$$

Then we can combine the above similarity preservation term with the tag consistency term and using a similar iterative optimization method to obtain the optimal hashing codes. We call this extended method Semantic Hashing using Tags and Topic Modeling (SHTTM).

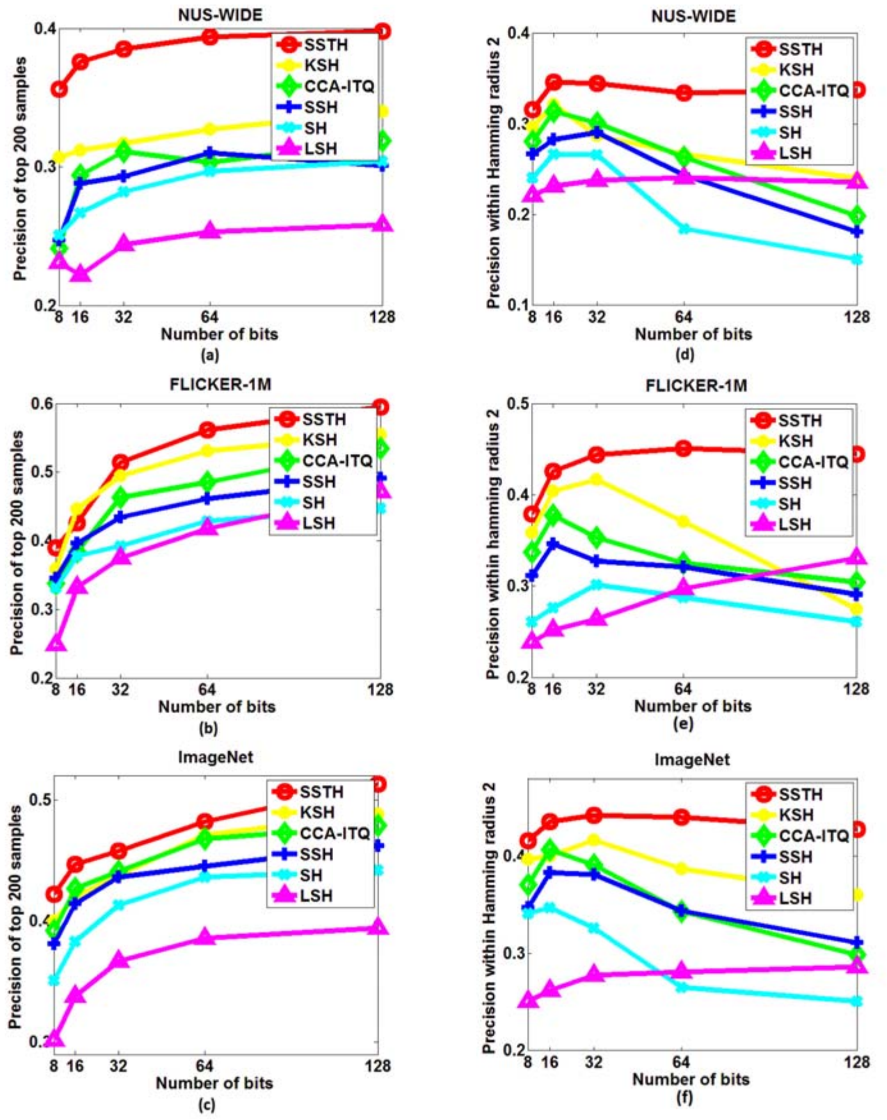


Figure 2.3. Precision results of SSTH on three image datasets. (a)-(c): Precision of the top 200 returned examples using Hamming Ranking. (d)-(f): Precision within Hamming radius 2 using Hash Lookup.

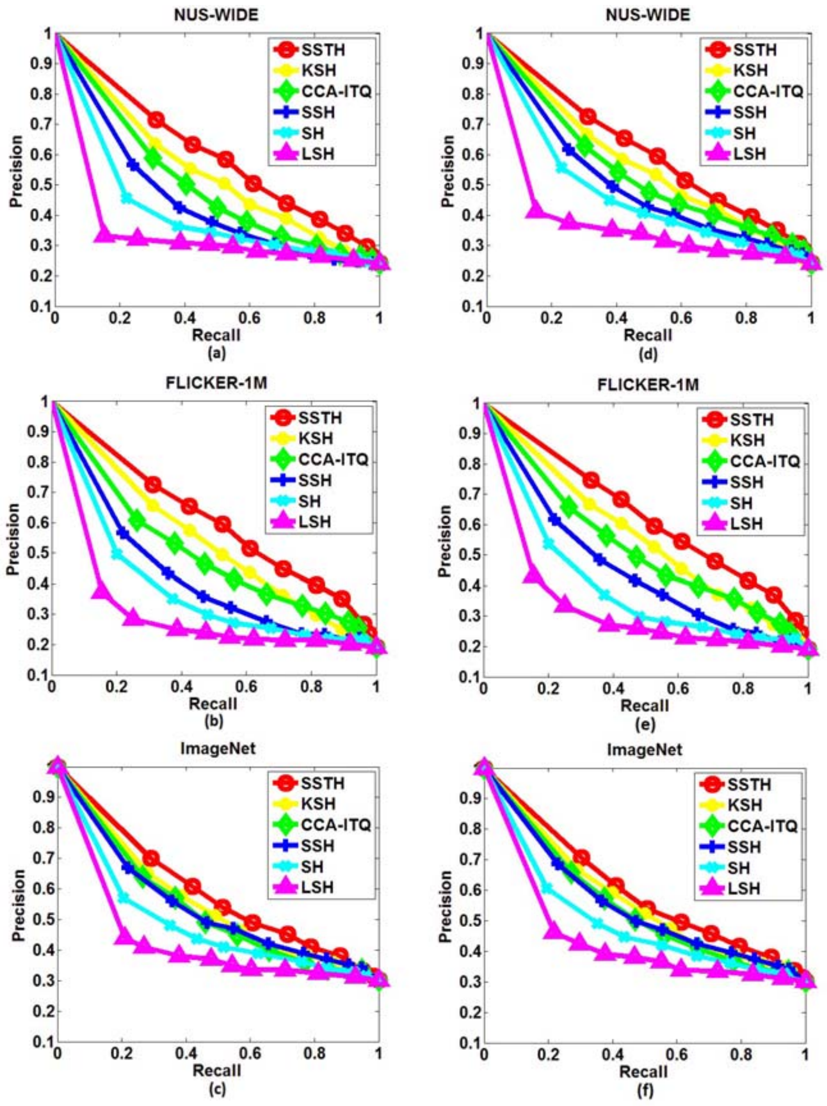


Figure 2.4. Precision-Recall behavior of SSTH on three image datasets. (a)-(c): Precision-Recall curve with 16 hashing bits. (d)-(f): Precision-Recall curve with 32 hashing bits.

2.4 Experiments

2.4.1 Datasets and Setup

We evaluate the SSTH method for large scale image retrieval on three image benchmarks: *NUS-WIDE*[3], *Flickr-1M*[4] and *ImageNet*[5]. *NUS-WIDE* [84] is created

[3]http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm

by NUS lab for evaluating image annotation and retrieval techniques. It contains $270k$ images associated with $5k$ unique tags. 500-dimensional visual features are extracted using a bag-of-word model with local SIFT descriptor [85]. We randomly partition this dataset into two parts, $1k$ for testing and around $269k$ for training. $Flickr$-$1M$ [86] is collected from Flicker images for image retrieval tasks. This benchmark contains 1 million image examples associated with more than $7k$ unique tags. 512-dimensional GIST descriptors [87] are extracted from these images and are used as image features for hashing function learning. We randomly choose $990k$ image examples as the training set and $10k$ for testing. $ImageNet$ [88] contains $1.2m$ images collected from flickr and other search engines. It is hand labeled with the presence or absence of 1000 object categories. 500-dimensional bag-of-word SIFT features are also used. We randomly select $10k$ images as test queries and the rest are used for training.

We also evaluate the extended SHTTM method using topic modeling for large scale text retrieval on four text collections: $ReutersV1$ (Reuters-Volume I) contains over 800,000 manually categorized newswire stories [89]. There are in total 126 tags associated with this dataset. A subset of 365001 documents of ReutersV1 is used in our experiment. 328501 documents are randomly selected as the training data, while the remaining 36500 documents are used as testing queries. 106 tags are selected for training and 20 for testing. $Reuters$ (Reuters21578)[6] is a collection of documents that appeared on Reuters newswire in 1987. It contains 21578 documents, and 135 tags/categories. In our experiments, documents corresponding to the top 57 categories are kept[7], with approximately 10376 documents. 9339 documents are randomly chosen as the training set, while 1037 for testing. 47 tags are utilized in training and 10 left for testing. $20Newsgroups$[8] corpus is collected and originally used for document categorization in [90]. We use the popular '18828' version which

[4]http://press.liacs.nl/mirflickr/

[5]http://image-net.org/challenges/LSVRC/2010/index

[6]http://daviddlewis.com/resources/textcollections/reuters2 1578/.

[7]we removed the tags which only have a limited number of examples.

[8]http://people.csail.mit.edu/jrennie/20Newsgroups/

contains 18828 documents. The data is organized into 20 different newsgroups, each corresponding to a different topic. Since some of the newsgroups are very closely related to each other (*e.g.* comp.sys.ibm.pc.hardware / comp.sys.mac.hardware), while others are highly unrelated (*e.g.* misc.forsale / soc.religion.christian). Therefore, we partition these documents according to subject matter into 6 categories, which are denoted as 6 different tags. 16946 documents are randomly chosen for training and the rest 1882 documents are used for testing. 3 tags are utilized in training process and 3 for testing. *WebKB*[9] consists of 6883 webpages, collected from four universities, and is divided into 7 categories/tags. 90% documents (6195) are randomly selected as training data, while the remaining (688) documents are used for testing. 4 tags are used in training while 3 for testing. In all text datasets, term frequency (i.e. tf) features are used as content features and also used for learning topic distributions. Note that tags in each dataset are divided into two groups, one set is used only in training and the other is treated as ground truth only for testing.

We implement our algorithm using Matlab on a PC with Intel Duo Core i5-2400 CPU 3.1GHz and 16GB RAM. The parameters α, β and γ are tuned by 5-fold cross validation on the training set and we will discuss how these parameters affect the results later. The number of nearest neighbors k is fixed to be 7 when constructing the graph *Laplacian* (we found that setting k to a larger number gives similar performance). Note that the graph *Laplacian* is a very sparse matrix consist of $n \times k$ non-zero elements and is only used to compute $\tilde{\boldsymbol{L}}$, which can be calculated efficiently by sparse matrix multiplication.

The proposed SSTH and SHTTM approach is compared with several different hashing algorithms, including four unsupervised methods Spectral Hashing (SH) [19], Latent Semantic Hashing (LSH) [15], PCA Hashing (PCAH) [17] and Self Taught Hashing (STH:2010), and three supervised methods Canonical Correlation Analysis with Iterative Quantization (CCA-ITQ) [35,36], Semi-Supervised Hashing (SSH) [29]

[9]CMU world wide knowledge base (WebKB) project. Available at http://www.cs.cmu.edu/WebKB/.

and Kernel Supervised Hashing (KSH) [30] as they are the state-of-the-art supervised hashing methods which achieve good performance in incorporating tag knowledge into hashing function learning. For LSH, we randomly select projections from a Gaussian distribution with zero-mean and identity covariance to construct the hash tables. For SSH and KSH, we sample $2k$ (which is used in their work) random points from the training set to construct the pairwise constraint matrix. Gaussian RBF kernel is used in KSH.

Table 2.2.
Precision of the top 200 returned examples using Hamming Ranking on *NUS-WIDE* with different hashing bits.

	NUS-WIDE			
code length / method	16	32	64	128
SSTH	**0.379**	**0.384**	**0.393**	**0.398**
SSTH^0	0.356	0.366	0.384	0.392
KSH [30]	0.312	0.316	0.327	0.341
CCA-ITQ [35,36]	0.306	0.308	0.315	0.322
SSH [29]	0.289	0.295	0.311	0.298
SH [19]	0.264	0.282	0.297	0.304
LSH [15]	0.226	0.247	0.258	0.261

2.4.2 Evaluation Method

To conduct fair evaluation, we follow two criteria which are commonly used in the literature [29,31,35,36]: Hamming Ranking and Hash Lookup. Hamming Ranking ranks all the points in the database according to their Hamming distance from the query and the top k points are returned as the desired neighbors. Hash Lookup

Table 2.3.
Precision of the top 200 returned examples using Hamming Ranking on *Flickr*-1*M* with different hashing bits.

	Flickr-1*M*			
code length / method	16	32	64	128
SSTH	0.432	**0.518**	**0.563**	**0.597**
SSTH^0	0.428	0.505	0.549	0.580
KSH [30]	**0.448**	0.494	0.531	0.556
CCA-ITQ [35,36]	0.386	0.464	0.491	0.539
SSH [29]	0.396	0.439	0.452	0.486
SH [19]	0.377	0.392	0.428	0.447
LSH [15]	0.337	0.371	0.423	0.464

returns all the points within a small Hamming radius r of the query. The search results are evaluated based on whether the retrieved examples and the query image share the same semantic tags in the testing. For Hamming Ranking based evaluation, we calculate the precision at top k which is the percentage of true neighbors among the top k returned examples, where we set k to be 200 in the experiments. We also compute the precision-recall value which is a widely used metric in information retrieval applications. A hamming radius of $r = 2$ is used to retrieve the neighbors in the case of Hash Lookup. The precision of the returned examples falling within Hamming radius 2 is reported. Note that if a query image returns no points inside Hamming ball with radius 2, it is treated as zero precision.

Table 2.4.
Precision of the top 200 returned examples using Hamming Ranking on *ImageNet* with different hashing bits.

	ImageNet			
method \ code length	16	32	64	128
SSTH	**0.447**	**0.458**	**0.482**	**0.513**
SSTH^0	0.438	0.446	0.477	0.501
KSH [30]	0.420	0.438	0.471	0.488
CCA-ITQ [35, 36]	0.427	0.441	0.468	0.479
SSH [29]	0.414	0.436	0.445	0.462
SH [19]	0.373	0.381	0.416	0.442
LSH [15]	0.338	0.367	0.386	0.394

Table 2.5.
Precision of the top 200 returned examples under different training tag ratios on *NUS-WIDE* with 32 hashing bits.

	NUS-WIDE			
method \ tag ratio	0.2	0.4	0.6	0.8
SSTH	**0.337**	**0.341**	**0.354**	**0.369**
SSTH^0	0.328	0.332	0.347	0.351
KSH [30]	0.288	0.296	0.301	0.308
CCA-ITQ [35, 36]	0.287	0.290	0.305	0.330
SSH [29]	0.283	0.285	0.291	0.297

2.4.3 Results and Discussion

We conduct several sets of experiments to evaluate the proposed approach from different perspectives.

Evaluation on different number of hashing bits

In the first set of experiments, we report the precisions for the top 200 retrieved images and the precisions for retrieved images within Hamming ball with radius 2 by varying the number of hashing bits in the range of $\{8, 16, 32, 64, 128\}$ in Fig.2.3 and Table 2.2, 2.3 and 2.4. We also evaluate our method without orthogonal transformation (by setting $\boldsymbol{Q} = \boldsymbol{I_b}$) and call this SSTH^0 in the tables. The precision-recall curves with 16 and 32 hashing bits on all datasets are reported in Fig.2.4.

Table 2.6.
Precision of the top 200 returned examples under different training tag ratios on *Flickr*-1*M* with 32 hashing bits.

	Flickr-1*M*			
tag ratio / method	0.2	0.4	0.6	0.8
SSTH	**0.453**	**0.461**	**0.476**	0.480
SSTH^0	0.443	0.449	0.464	0.476
KSH [30]	0.422	0.448	0.459	**0.481**
CCA-ITQ [35, 36]	0.410	0.427	0.445	0.467
SSH [29]	0.398	0.416	0.422	0.435

From these comparison results, we can see that SSTH provides the best results among all six hashing methods in most cases. LSH does not perform well on all datasets since LSH method is data-independent, which may generate inefficient codes compared to those data-depend methods. The unsupervised SH method only tries to preserve image similarity in learned hashing codes, but does not utilize the supervised information contained in tags. SSH and KSH achieves better performance than SH and LSH due to the modeling of pairwise information. However, as pointed out in section 2, the coarse pairwise constraints generated from tags do not fully represent tag knowledge. The supervised method CCA-ITQ have similar performance to KSH since it also incorporates tags into learning better data representations. But in CCA-ITQ, it treats tags as another independent source where it may not be even reliable as tags can be incomplete, noisy and partially available. Moreover, the visual similarity is not well preserved in its hashing function learning. On the other hand, our SSTH not only exploits tag information via modeling the correlation between tags and hashing bits, but also preserves image similarity at the same time in the learned hashing function, which enables SSTH to generate higher quality hashing codes than the other supervised hashing methods. In Fig.2.3(d)-(f), we observe the precision of Hash Lookup for most of the compared methods decreases significantly with the increasing number of hashing bits. The reason is that the Hamming space becomes increasingly sparse with longer hashing bits and very few data points fall within the Hamming ball with radius 2, which makes many queries have 0 precision results. However, the precision of SSTH is still consistently higher than the other methods for Hash Lookup.

Table 2.7.
Precision of the top 200 returned examples under different training tag ratios on *ImageNet* with 32 hashing bits.

	ImageNet			
tag ratio / method	0.2	0.4	0.6	0.8
SSTH	**0.409**	**0.421**	**0.429**	**0.437**
SSTH^0	0.402	0.415	0.423	0.429
KSH [30]	0.391	0.408	0.415	0.421
CCA-ITQ [35, 36]	0.383	0.402	0.411	0.428
SSH [29]	0.387	0.403	0.412	0.424

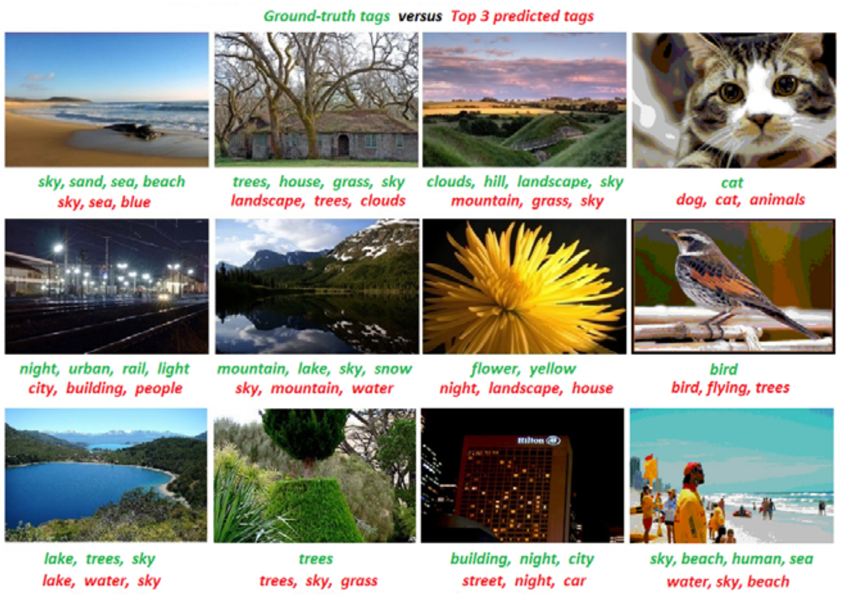


Figure 2.5. Tag prediction results on *Flickr*-1*M*.

Evaluation on different training tag ratio

In the second set of experiments, we evaluate the effectiveness of the proposed SSTH if only partial tags are available. We progressively increase the number of training tags by varying the training tag ratio from {0.2, 0.4, 0.6, 0.8}[10] and compare our SSTH with the other supervised hashing methods[11], CCA-ITQ, SSH and KSH on all datasets by fixing the hashing bits to 32. The precision results of top 200 retrieved images are reported in Table 2.5, 2.6 and 2.7. It can be seen from the results that our SSTH gives the best performance among all supervised hashing methods in most cases. We also observe that the precision result of compared supervised hashing methods drop faster than SSTH when the number of training tags decreases. Our hypothesis is that when training tags are very sparse and incomplete, the pairwise constraints generated by SSH and KSH from these partial tags are not accurate

[10]Tags are randomly sampled from the training data based on the ratio.
[11]SH and LSH do not utilize tags and thus are not necessary to be compared here.

and reliable, resulting in less effective hashing codes. Similarly, the common space learned from partial tags and visual features by CCA-ITQ is not very meaningful due to the lack of training tags. On the other side, the unsupervised part of SSTH plays an important role in similarity preservation, which preserves the neighbor structure in the learned hashing codes even with very few tags. We also observe that the comparison results of SSTH and SSTH^0 in all tables demonstrate that the orthogonal transformation can further improve the effectiveness of the hashing function, which is consistent with our expectation.

Table 2.8.
Training and testing time (in second) on three datasets with 32 hashing bits.

	NUS-WIDE		*Flickr*-1*M*		*ImageNet*	
time / method	training	testing	training	testing	training	testing
SSTH	83.57	0.4x10^{-4}	219.03	0.6x10^{-4}	284.25	0.4x10^{-4}
KSH [30]	248.85	2.4x10^{-4}	592.16	2.5x10^{-4}	686.82	2.4x10^{-4}
CCA-ITQ [35,36]	46.13	0.5x10^{-4}	135.37	0.5x10^{-4}	168.54	0.5x10^{-4}
SSH [29]	23.56	0.4x10^{-4}	40.83	0.5x10^{-4}	58.39	0.4x10^{-4}
SH [19]	51.63	3.6x10^{-4}	173.68	4.1x10^{-4}	224.07	3.7x10^{-4}
LSH [15]	3.75	0.4x10^{-4}	3.84	0.4x10^{-4}	3.76	0.4x10^{-4}

Qualitative results on tag prediction

The third set of experiments demonstrate how the learned correlations, $\boldsymbol{C}$, can bridge tags and hashing codes[12]. We conduct the experiments on *Flickr*-1*M* to predict tags for query images based on their hashing codes. In particular, we first

[12]In this set of experiments, we choose the most popular 50 tags in the training process and remove all images that are not associated with any tags.

generate hashing code for each query image by $y_q = \boldsymbol{W}^T x_q$, and predict its tag vector using $t_q = \boldsymbol{C}^T y_q$. Then we select the top 3 tags with largest values in tag vector t_q as the predicted tags for the query image. The comparison results of the top 3 predicted tags with ground truth tags on several images are shown in Fig.2.5. From this figure we can see that our SSTH can generate reasonable accurate tags for query images. The reason is that our method not only incorporates tags in learning effective hashing function, but also extracts the correlation between tags and hashing bits. Therefore, the tag information is fully explored in our SSTH. Note that previous supervised (pairwise or listwise) hashing methods can not directly generate tags for unseen images[13].

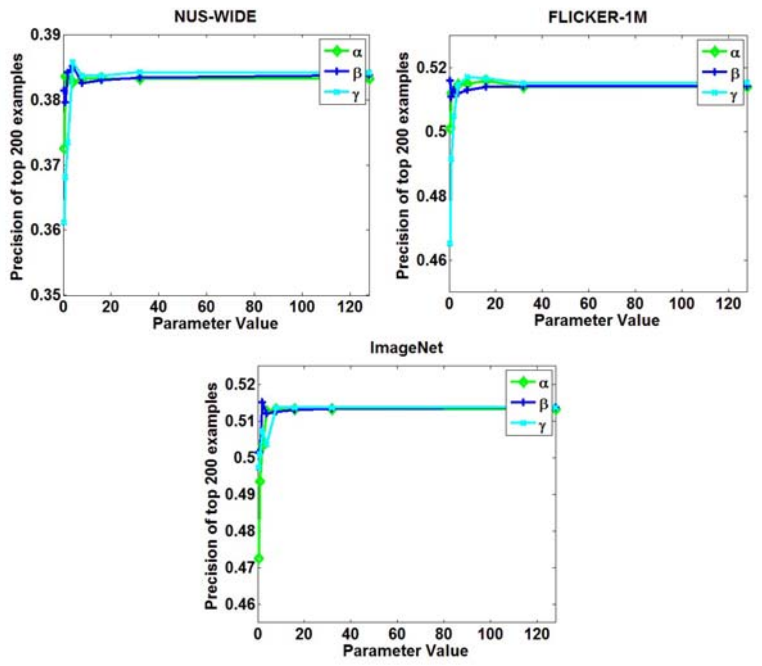


Figure 2.6. Parameter sensitivity results of precision of the top 200 retrieved examples with 32 hashing bits.

[13]Although in [35] tags can also be predicted by employing additional decoding methods, it is more complicated and time consuming for learning another set of decoding parameters, which is also pointed out by the authors.

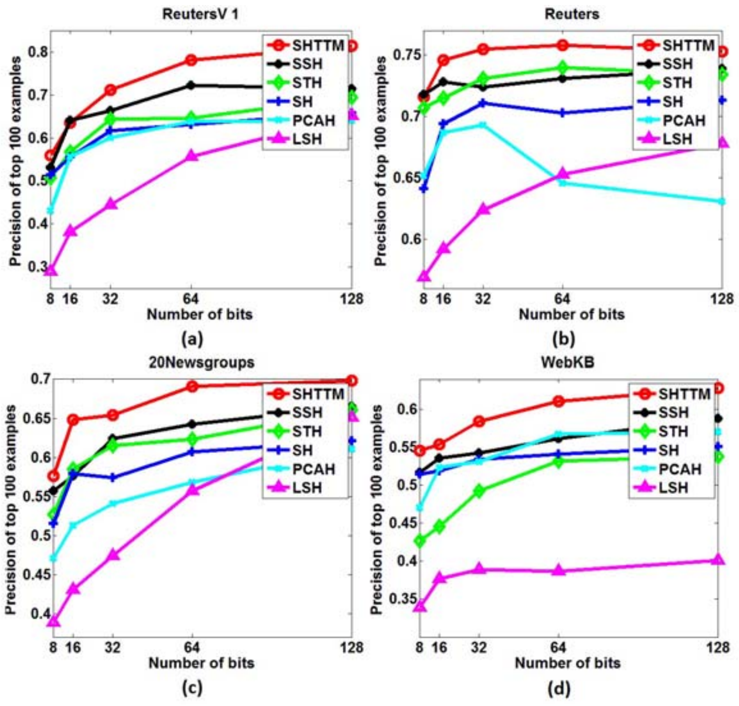


Figure 2.7. Precision of the top 100 retrieved examples on four text datasets.

Training and testing cost

In the fourth set of experiments, the training time for learning hashing function and testing time for encoding each query image on all datasets (with 32 bits) are reported in Table 2.8. Note that we do not include the cross-validation time and any pre-calculation cost in all methods for fair comparison. We can see from this table that the training cost of SSTH is around several hundred seconds, which is comparable with most of the other hashing methods and it is not slow in practice considering the complexity of training. In contrast to the offline training, the online code generation time is more critical for real-world search applications. The test time for SSTH is sufficiently fast especially when compared to the nonlinear hashing method SH and kernel hashing method KSH. The reason is that it only needs linear projection and binarization to generate the hashing codes for queries. Moreover, we also conduct another experiment by fixing the performance of different methods to figure out how many bits and how much time does each method cost. We found that our method utilizes much less time and bits than other methods to achieve a certain performance. For example, to achieve 0.55 precision under Hamming Ranking on $Flickr$-$1M$. Our method uses about 150 seconds and 60 bits, while CCA-ITQ takes about 374 seconds with 270 bits and SSH costs around 420 seconds with 500 bits.

Parameter sensitivity

The fifth set of experiments study the performance of SSTH with respect to the parameters α, β and γ. To show the robustness of the proposed method, we conduct parameter sensitivity experiments on all datasets. In each experiment, we tune only

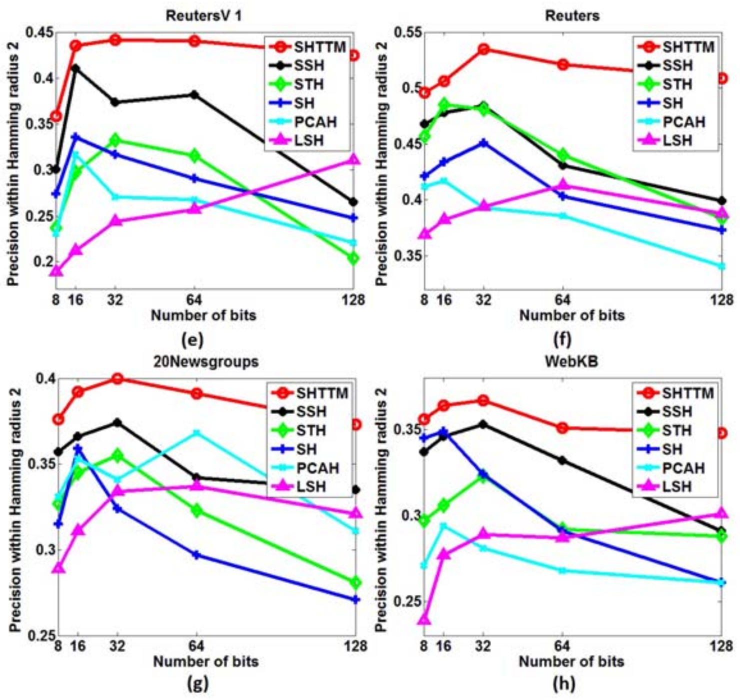


Figure 2.8. Precision of the retrieved examples within Hamming radius 2 on four text datasets.

one parameter from the grid {0.5, 1, 2, 4, 8, 32, 128}, while fixing the other two to the optimal values obtained from the first set of experiments.

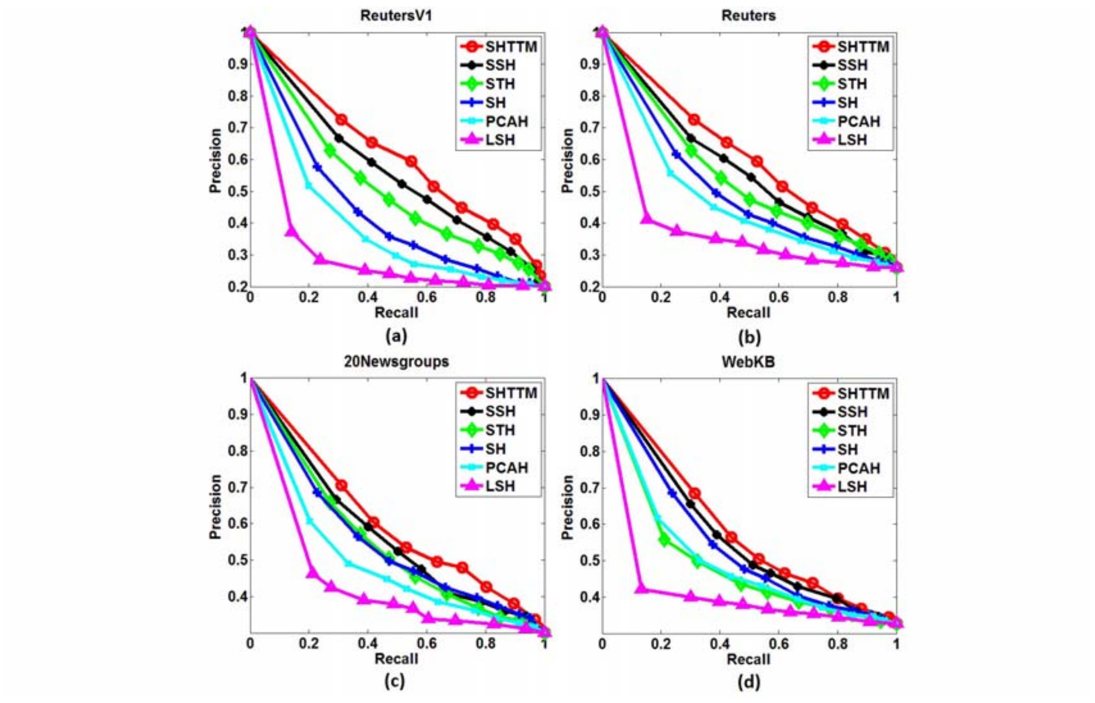


Figure 2.9. Results of Precision-Recall curve with 32 hashing bits on four text datasets.

We report the results of top 200 returned examples with 32 hashing bits in Fig.2.6. It is clear from these experimental results that the performance of SSTH is relatively stable with respect to α, β and γ in a wide range of values. The results also prove that using soft penalty with an appropriate weight parameter is better than enforcing the hard orthogonality constraint (corresponds to infinite β).

Evaluation of Topic Modeling

In the sixth set of experiments, we evaluation the performance of SHTTM method on four text datasets. The precision for the top 100 retrieved documents with different numbers of hashing bits is reported in Fig.2.7. The precisions for the retrieved documents within hamming radius 2 are shown in Fig.2.8. We also report the precision-recall curves of different methods with 32 hashing bits on different datasets in Fig.2.9. From these comparison results, it can be seen that SHTTM gives the overall best performance among all six hashing methods on all four datasets.

In Fig.2.8, the precision of most compared methods decreases when the number of hashing bits increases from 16 to 128 This is because when using longer hashing bits, the Hamming space becomes increasingly sparse and very few data points fall within the Hamming ball of radius 2, resulting in even queries with precision 0. Similar behavior is also observed in [30] and [29]. In this situation, the precision results of top 100 documents from Fig.2.7 provide better performance measurement, while the precision results of SHTTM are still consistently better than other methods. For methods SH and STH, although these methods try to preserve the similarity between documents in their learned hashing codes, they do not utilize the supervised information contained in tags. Moreover, the similarity matrices in both methods are computed from the original keyword feature space, which may not fully reflect the semantic similarity between documents that goes beyond keyword matching. Therefore, the SHTTM method substantially outperforms these two methods by leveraging tag information and topic modeling.

3 LEARNING TO HASH ON PARTIAL MULTI-MODAL DATA

3.1 Motivation

In many applications, data examples are usually represented by multiple modalities captured from different sources. Although existing multi-modal hashing methods generate promising results in dealing with multi-modal data, most of them assume that all data examples have full information in all modalities, or there exists at least one modality which contains all the examples. However, in real world tasks, it is often the case that every modality suffers from some missing information, which results in many partial examples. Therefore, it is a practical and important research problem to design effective hashing methods for partial multi-modal data.

3.2 Background and Related Work

3.2.1 Introduction

In various real world tasks, data examples usually have multiple modalities extracted from different sources. For example, in web page search, the web page content and its linkage information can be regarded as two modalities. In web image retrieval, the image visual feature, text description and textual tags can be viewed as multiple modalities. Recently, several multi-modal hashing methods (also known as multi-view or cross-view hashing) have been proposed to handle multi-modal data. Roughly speaking, these multi-modal hashing approaches can be divided into two categories: modality-specific hashing methods and modality-integrated ones. The modality-specific methods learn independent hashing codes for each modality of an example, and then concatenate multiple modality-specific binary codes into the final hashing codes [39–42], whereas the modality-integrated ones directly learn unified hashing codes for data examples [10, 43–45].

Although existing multi-modal hashing methods generate promising results in dealing with multi-modal data, they assume that all data examples have full information in all modalities, or there exists at least one modality which contains all the examples. However, in real world tasks, it is often the case that every modality suffers from some missing information, which results in many partial examples. Consider again the aforementioned two examples, for web page search, many web pages may not contain any linkage information. For web image retrieval, not all images are associated with tags or text descriptions. Moreover, the image itself may be inaccessible due to deletion or invalid url. Therefore, it is a practical and important research problem to design effective hashing methods for partial multi-modal data.

In order to apply existing multi-modal hashing methods to partial data, we can either remove the data examples that suffer from missing information, or preprocess the partial examples by first filling in the missing data. The first strategy is clearly not suitable since the purpose is to map all examples to their corresponding binary codes, whereas our experiments show that the second strategy does not achieve good performance either. In this dissertation, we propose a novel Partial Multi-Modal Hashing (PM^{2}H) approach to deal with such partial data. More specifically, a unified learning framework is developed to learn the binary codes, which simultaneously ensures the data consistency among different modalities via latent subspace learning, and preserves data similarity within the same modality through graph Laplacian. A coordinate descent algorithm is applied as the optimization procedure. We then further reduce the quantization error via orthogonal rotation based on the orthogonal invariant property of our formulation. Experiments on two datasets demonstrate the advantages of the proposed approach over several state-of-the-art multi-modal hashing methods.

3.2.2 Related Work

Hashing methods [13,15,24,48,91,92] are proposed to generate reasonably accurate search results in a fast process with compact binary vector representation. These methods transform the original features into a low dimensional binary space, while at the same time preserve the similarity between data examples as much as possible. In this section, we mainly review the previous hashing methods on multiple modality data.

Multi-modal hashing methods have recently been developed to extend the single-modal methods to multi-modal scenarios. The key problem of hashing code learning for multi-modal is to deal with multiple modalities sampled from different probability distributions. Existing multi-modality hashing methods can be divided into two categories: modality-specific and modality-integrated methods. The modality-specific methods [39–42, 93] learn independent hashing codes for each modality of data examples, and then merge multiple binary codes from different modalities into the final hashing codes. A cross-modality similarity search hashing (CMSSH) method [39] is proposed to embed data from different feature space into a common metric space. The hashing codes are learned through eigen-decomposition with AdaBoost framework. In work [40], a cross-view hashing method is designed based on spectral hashing, which generates the hashing codes by minimizing the distance of hashing codes for the similar data and maximizing the distance for the dissimilar data. Co-Regularized Hashing [42] method intends to project data from multiple sources, and at the same time, preserve the inter-modality similarity effectively.

The modality-integrated hashing methods [10, 35, 36, 43, 44] directly learn unified hashing codes for each data example. In the work of [10], a Composite Hashing with Multiple Information Sources (CHMIS) method is proposed to incorporate information from multiple sources into final integrated hashing codes by linearly combining the hashing codes from different modalities. Multi-View Spectral Hashing (MVSH) [44] integrates multi-view information into binary codes, and uses product of codewords to avoid undesirable embedding. More recently, A Canonical Correlation Analysis with Iterative Quantization (CCA-ITQ) method has been proposed in [35,36] which treats the image features and tags as two different modalities. The hashing function is then learned by extracting a common space from these two modalities. The work in [43] introduces collective matrix factorization into multi-modal hashing (CMFH), which learns unified hashing codes by collective matrix factorization with latent factor model from different modalities. However, existing multi-modal hashing methods fail to handle the situation where only partial examples are available in different modalities.

3.3 Algorithm

3.3.1 Problem Setting and Overview

We introduce some notations in our problem of PM^{2}H. For the convenience of discussion, assume that we are handling two-modality data, i.e., given a data set of N data examples $\boldsymbol{X}=\{(x_i^1, x_i^2), i = 1, \ldots, N\}$, where $x_i^1 \in \mathbb{R}^{d_1}$ is the instance of the *i-th* example in the first modality and $x_i^2 \in \mathbb{R}^{d_2}$ is the *i-th* example in the second modality (usually $d_1 \neq d_2$). In the partial modality setting, a partial data set $\hat{\boldsymbol{X}}=\{\hat{\boldsymbol{X}}^{(1,2)}, \hat{\boldsymbol{X}}^{(1)}, \hat{\boldsymbol{X}}^{(2)}\}$ instead of $\boldsymbol{X}$ is given, where $\hat{\boldsymbol{X}}^{(1,2)}=\{(x_1^1, x_1^2), \ldots, (x_c^1, x_c^2)\} \in \mathbb{R}^{c\times(d_1+d_2)}$ denotes the common examples present in both modalities, $\hat{\boldsymbol{X}}^{(1)} = \{x_{c+1}^1, \ldots, x_{c+m}^1\} \in \mathbb{R}^{m\times d_1}$ denotes the examples only present in the first modality and $\hat{\boldsymbol{X}}^{(2)}=\{x_{c+m+1}^2, \ldots, x_{c+m+n}^2\} \in \mathbb{R}^{n\times d_2}$ denotes the examples only present in the second modality. Note that the number of examples present and only present in both modalities, the first modality, and the second modality are c, m and n, $N = c + m + n$. The purpose of PM^{2}H is to learn unified hashing codes $\boldsymbol{Y} = \{y_1, y_2, \ldots, y_N\} \in \{-1, 1\}^{N\times k}$ together with the modality-specific hashing functions $\boldsymbol{H}^1$ and $\boldsymbol{H}^2$ to map each data example x_i to the corresponding hashing codes y_i:

$$y_i = sgn(v_i) = sgn(\boldsymbol{H}^t x_i^t) \quad t = 1, 2 \tag{3.1}$$

where $\boldsymbol{H}^t \in \mathbb{R}^{k \times d_t}$ is the coefficient matrix representing the hashing function for the *t-th* modality and *sgn* is the sign function. k is the length of the code. v_i is the signed magnitude relaxation of binary code y_i, which is widely adopted in previous hashing approaches [30, 31, 94].

The objective function of PM^{2}H is composed of two components: (1) Data consistency between modalities, latent subspace learning is utilized to ensure that the hashing codes generated from different modalities are consistent. (2) Similarity preservation within modality, graph Laplacian is applied to enforce that similar data examples within each modality are mapped into similar codes. A coordinate descent method is utilized for solving the optimization problem to achieve the relaxed solution v_i. Then the binary hashing codes y_i will be obtained from v_i with orthogonal rotation by minimizing the quantization error.

3.3.2 Data Consistency between Modalities

In the partial modality setting, $\hat{\boldsymbol{X}}^{(1,2)}, \hat{\boldsymbol{X}}^{(1)}, \hat{\boldsymbol{X}}^{(2)}$ are represented by heterogeneous features of dimensions $(d_1 + d_2)$, d_1, d_2, which makes it hard for their hashing codes learning. But investigating the problem from modality perspective, in each individual modality, the data instances are sharing the same feature space. The two different modalities are coupled/bridged by the shared common examples. If we can learn a common latent subspace for the two modalities, where instances belonging to the same example between different modalities are consistent, while at the same time for each modality, the representations for similar instances are close in the latent subspace. Then the hashing codes can be directly learned from this subspace, and we do not need to fill in or complete the partial modality examples. Let $\hat{\boldsymbol{X}}^{(1,2)} = [\hat{\boldsymbol{X}}_c^{(1)}, \hat{\boldsymbol{X}}_c^{(2)}]$, where $\hat{\boldsymbol{X}}_c^{(1)} \in \mathbb{R}^{c \times d_1}$, $\hat{\boldsymbol{X}}_c^{(2)} \in \mathbb{R}^{c \times d_2}$ are the instances of the common examples coming from the two modalities. We denote the instances of each modality as: $\bar{\boldsymbol{X}}^{(1)} = [\hat{\boldsymbol{X}}_c^{(1)}, \hat{\boldsymbol{X}}^{(1)}] \in \mathbb{R}^{(c+m)\times d_1}$, $\bar{\boldsymbol{X}}^{(2)} = [\hat{\boldsymbol{X}}_c^{(2)}, \hat{\boldsymbol{X}}^{(2)}] \in \mathbb{R}^{(c+n)\times d_2}$. Following the above idea, the latent subspace learning can be formulated as:

$$\min_{\bar{\boldsymbol{V}}^{(1)},\boldsymbol{B}^{(1)}} \|\bar{\boldsymbol{X}}^{(1)} - \bar{\boldsymbol{V}}^{(1)}\boldsymbol{B}^{(1)}\|_F^2 + \lambda\ R(\bar{\boldsymbol{V}}^{(1)}, \boldsymbol{B}^{(1)}) \tag{3.2}$$

$$\min_{\bar{\boldsymbol{V}}^{(2)},\boldsymbol{B}^{(2)}} \|\bar{\boldsymbol{X}}^{(2)} - \bar{\boldsymbol{V}}^{(2)}\boldsymbol{B}^{(2)}\|_F^2 + \lambda\ R(\bar{\boldsymbol{V}}^{(2)}, \boldsymbol{B}^{(2)}) \tag{3.3}$$

where $\boldsymbol{B}^{(1)} \in \mathbb{R}^{k\times d_1}$ and $\boldsymbol{B}^{(2)} \in \mathbb{R}^{k\times d_2}$ are the basis matrix for each modality's latent space. $\bar{\boldsymbol{V}}^{(1)} = [\hat{\boldsymbol{V}}_c^{(1)}, \hat{\boldsymbol{V}}^{(1)}] \in \mathbb{R}^{(c+m)\times k}$ and $\bar{\boldsymbol{V}}^{(2)} = [\hat{\boldsymbol{V}}_c^{(2)}, \hat{\boldsymbol{V}}^{(2)}] \in \mathbb{R}^{(c+n)\times k}$ are the latent representation of instances in the latent space, which can also be viewed as the relaxed representation of binary codes $\boldsymbol{Y}$. The same latent space dimension k is shared between the two modalities. $R(\cdot) = \|\cdot\|_F^2$ is the regularization term to avoid overfitting and λ is the tradeoff parameter. By Eqn.3.2 and Eqn.3.3, the latent space basis $\boldsymbol{B}$ and corresponding instance latent representation $\boldsymbol{V}$ are simultaneously learned to minimize the reconstruction error from each individual modality.

In the above equations, the latent space are learned independently for each modality. But in the partial modality setting, for examples present in both modalities $\hat{\boldsymbol{X}}_c^{(1)}$, $\hat{\boldsymbol{X}}_c^{(2)}$, their latent representation $\hat{\boldsymbol{V}}_c^{(1)}$, $\hat{\boldsymbol{V}}_c^{(2)}$ should also be consistent. Incorporating the above formulations by ensuring $\hat{\boldsymbol{V}}_c^{(1)} = \hat{\boldsymbol{V}}_c^{(2)} = \hat{\boldsymbol{V}}_c$, we seek to minimize the following problem:

$$\begin{aligned} \min_{\boldsymbol{V},\boldsymbol{B}} & \left\| \begin{bmatrix} \hat{\boldsymbol{X}}_c^{(1)} \\ \hat{\boldsymbol{X}}^{(1)} \end{bmatrix} - \begin{bmatrix} \hat{\boldsymbol{V}}_c \\ \hat{\boldsymbol{V}}^{(1)} \end{bmatrix} \boldsymbol{B}^{(1)} \right\|_F^2 \\ + & \left\| \begin{bmatrix} \hat{\boldsymbol{X}}_c^{(2)} \\ \hat{\boldsymbol{X}}^{(2)} \end{bmatrix} - \begin{bmatrix} \hat{\boldsymbol{V}}_c \\ \hat{\boldsymbol{V}}^{(2)} \end{bmatrix} \boldsymbol{B}^{(2)} \right\|_F^2 + \lambda\ R(\boldsymbol{V}, \boldsymbol{B}) \end{aligned} \tag{3.4}$$

By solving the above problem, we can obtain the homogeneous feature (relaxed hashing) representation for all examples as $\boldsymbol{V} = [\hat{\boldsymbol{V}}_c, \hat{\boldsymbol{V}}^{(1)}, \hat{\boldsymbol{V}}^{(2)}] \in \mathbb{R}^{(c+m+n)\times k}$, whether they are originally partial or not. Then the hashing codes $\boldsymbol{Y}$ can be directly achieved via binarization from this relaxed latent representation. Note that Eqn.3.4 is different from previous subspace based multi-modal hashing approaches, which either requires $\bar{\boldsymbol{V}}^{(1)}$ and $\bar{\boldsymbol{V}}^{(2)}$ to be the same or do not require $\bar{\boldsymbol{V}}^{(1)}$ and $\bar{\boldsymbol{V}}^{(2)}$ to share any common part.

In the above formulation, $\bar{\boldsymbol{V}}^{(1)}$ and $\bar{\boldsymbol{V}}^{(2)}$ share one common representation $\hat{\boldsymbol{V}}_c$, while at the same time have their own individual components. Moreover, the individual basis matrix $\boldsymbol{B}^{(1)}$ and $\boldsymbol{B}^{(2)}$, which are learned by using all available instances from both modalities, are connected by the common $\hat{\boldsymbol{V}}_c$.

3.3.3 Similarity Preservation within Modality

One of the key problems in hashing algorithms is similarity preserving, which indicates that similar data examples should be mapped to similar hashing codes within a short Hamming distance. Therefore, besides the data consistency between different modalities, we also preserve the data similarity within each individual modality. In other words, we want the learned relaxed representation $\boldsymbol{V}$ to preserve the similarity structure in each modality. In this dissertation, we use the L_2 distance to measure the similarity between v_i and v_j as $\|v_i - v_j\|^2$, which is consistent with the Hamming distance between the binary codes y_i and y_j ($\frac{1}{4}\|y_i - y_j\|^2$). Then one natural way to preserve the similarity in each modality is to minimize the weighted average distance as follows:

$$\sum_{i,j} \boldsymbol{S}_{ij}^{(t)} \|v_i - v_j\|^2 \quad t = 1, 2 \tag{3.5}$$

Here, $\boldsymbol{S}^{(t)}$ is the similarity matrix in *t-th* modality, which can be calculated from the instances $\bar{\boldsymbol{X}}^{(t)}$. In this dissertation, we adopt the local similarity [10,94], due to its nice property in many machine learning applications. To meet the similarity preservation criterion, we seek to minimize this quantity in each modality since it incurs a heavy penalty if two similar examples have very different latent representations.

By introducing a diagonal $n \times n$ matrix $\boldsymbol{D}^{(t)}$, whose entries are given by $\boldsymbol{D}_{ii}^{(t)} = \sum_{j=1}^{n} \boldsymbol{S}_{ij}^{(t)}$. Eqn.3.5 can be rewritten as:

$$tr\left(\bar{\boldsymbol{V}}^{(t)^T}(\boldsymbol{D}^{(t)} - \boldsymbol{S}^{(t)})\bar{\boldsymbol{V}}^{(t)}\right) = tr\left(\bar{\boldsymbol{V}}^{(t)^T}\boldsymbol{L}^{(t)}\bar{\boldsymbol{V}}^{(t)}\right) \quad t = 1, 2 \tag{3.6}$$

where $\boldsymbol{L}$ is called graph *Laplacian* [19] and $tr(\cdot)$ is the matrix trace function. By minimizing the above objective in all modalities, the similarity between different examples can be preserved in the latent representation.

3.3.4 Overall Objective and Optimization

The entire objective function consists of two components: the data consistency between modalities in Eqn.3.4 and similarity preservation within modality given in Eqn.3.6 as follows:

$$\begin{aligned} \min_{\boldsymbol{V},\boldsymbol{B}} O \; &= \|\bar{\boldsymbol{X}}^{(1)} - \bar{\boldsymbol{V}}^{(1)}\boldsymbol{B}^{(1)}\|_F^2 + \|\bar{\boldsymbol{X}}^{(2)} - \bar{\boldsymbol{V}}^{(2)}\boldsymbol{B}^{(2)}\|_F^2 \\ &+\alpha \left(tr\left(\bar{\boldsymbol{V}}^{(1)^T}\boldsymbol{L}^{(1)}\bar{\boldsymbol{V}}^{(1)}\right) + \; tr\left(\bar{\boldsymbol{V}}^{(2)^T}\boldsymbol{L}^{(2)}\bar{\boldsymbol{V}}^{(2)}\right)\right) \\ &+\lambda \; R(\boldsymbol{V},\boldsymbol{B}) \end{aligned} \tag{3.7}$$

where α and λ are trade-off parameters to balance the weights among the terms. Note that $\bar{\boldsymbol{V}}^{(1)}$ and $\bar{\boldsymbol{V}}^{(2)}$ share an identical part $\hat{\boldsymbol{V}}_c$ corresponding to the common examples present in both modalities.

Directly minimizing the objective function in Eqn.3.7 is intractable since it is a non-convex optimization problem with $\boldsymbol{V}$ and $\boldsymbol{B}$ coupled together. We propose to use coordinate descent scheme by iteratively solving the optimization problem with respect to $\boldsymbol{V}$ and $\boldsymbol{B}$ as follows:

(1) Optimizing O with respect to $\hat{\boldsymbol{V}}_c$, $\hat{\boldsymbol{V}}^{(1)}$ and $\hat{\boldsymbol{V}}^{(2)}$ by fixing $\boldsymbol{B}$. Given the basis matrix $\boldsymbol{B}^{(t)}$ for both modalities, we can decompose the objective since $\hat{\boldsymbol{V}}_c$ and $\hat{\boldsymbol{V}}^{(t)}$ will not depend on each other.

$$\begin{aligned} \min_{\hat{\boldsymbol{V}}^{(t)}} O(\hat{\boldsymbol{V}}^{(t)}) \; &= \|\hat{\boldsymbol{X}}^{(t)} - \hat{\boldsymbol{V}}^{(t)}\boldsymbol{B}^{(t)}\|_F^2 \\ +\alpha \; tr\left(\hat{\boldsymbol{V}}^{(t)^T}\hat{\boldsymbol{L}}^{(t)}\hat{\boldsymbol{V}}^{(t)}\right) &+ \lambda \; R(\hat{\boldsymbol{V}}^{(t)}) + const \quad t = 1,2 \end{aligned} \tag{3.8}$$

$$\begin{aligned} \min_{\hat{\boldsymbol{V}}_c} O(\hat{\boldsymbol{V}}_c) \; &= \|\hat{\boldsymbol{X}}_c^{(1)} - \hat{\boldsymbol{V}}_c\boldsymbol{B}^{(1)}\|_F^2 + \|\hat{\boldsymbol{X}}_c^{(2)} - \hat{\boldsymbol{V}}_c\boldsymbol{B}^{(2)}\|_F^2 \\ +\alpha \; tr\left(\hat{\boldsymbol{V}}_c{}^T(\hat{\boldsymbol{L}}_c^{(1)} + \hat{\boldsymbol{L}}_c^{(2)})\hat{\boldsymbol{V}}_c\right) &+ \lambda \; R(\hat{\boldsymbol{V}}_c) + const \end{aligned} \tag{3.9}$$

where $\hat{\boldsymbol{L}}^{(t)}$ and $\hat{\boldsymbol{L}}_c^{(t)}$ can be simply derived from $\boldsymbol{L}^{(1)}$ with some addition mathematical operation. *const* is the constant value independent with the parameter that to be optimized with. Although Eqn.3.8 and Eqn.3.9 are still non-convex, but they are smooth and differentiable which enables gradient descent methods for efficient optimization. The gradients of Eqn.3.8 and Eqn.3.9 are calculated as follows:

$$\begin{aligned} \partial\frac{O(\hat{\boldsymbol{V}}^{(t)})}{\hat{\boldsymbol{V}}^{(t)}} &= -2\hat{\boldsymbol{X}}^{(t)}\boldsymbol{B}^{(t)^T} + 2\hat{\boldsymbol{V}}^{(t)}\boldsymbol{B}^{(t)}\boldsymbol{B}^{(t)^T} \\ &+2\alpha\hat{\boldsymbol{L}}^{(t)}\hat{\boldsymbol{V}}^{(t)} + 2\lambda\hat{\boldsymbol{V}}^{(t)} \end{aligned} \tag{3.10}$$

$$\begin{aligned} \partial\frac{O(\hat{\boldsymbol{V}}_c)}{\hat{\boldsymbol{V}}_c} &= -2\hat{\boldsymbol{X}}_c^{(1)}\boldsymbol{B}^{(1)^T} - 2\hat{\boldsymbol{X}}_c^{(2)}\boldsymbol{B}^{(2)^T} + 2\lambda\hat{\boldsymbol{V}}_c \\ +2\hat{\boldsymbol{V}}_c(\boldsymbol{B}^{(1)}\boldsymbol{B}^{(1)^T} &+ \boldsymbol{B}^{(2)}\boldsymbol{B}^{(2)^T}) + 2\alpha(\hat{\boldsymbol{L}}_c^{(1)} + \hat{\boldsymbol{L}}_c^{(2)})\hat{\boldsymbol{V}}_c \end{aligned} \tag{3.11}$$

With these obtained gradients, L-BFGS quasi-Newton method [79] is applied to solve Eqn.3.8 and Eqn.3.9.

(2) Optimizing O with respect to $\boldsymbol{B}^{(t)}$ by fixing $\boldsymbol{V}$. It is equivalent to solve the following least square problems:

$$\min_{\boldsymbol{B}^{(t)}} O(\boldsymbol{B}^{(t)}) = \|\bar{\boldsymbol{X}}^{(t)} - \bar{\boldsymbol{V}}^{(t)}\boldsymbol{B}^{(t)}\|_F^2 + \lambda\|\boldsymbol{B}^{(t)})\|_F^2 \quad t = 1, 2 \tag{3.12}$$

By taking the derivative of Eqn.3.12 w.r.t. $\boldsymbol{B}^{(t)}$ and setting it to $\boldsymbol{0}$, a closed form solution can be simply obtained. We then alternate the process of updating $\boldsymbol{V}$ and $\boldsymbol{B}$ for several iterations to find a locally optimal solution.

3.3.5 Orthogonal Rotation

After obtaining the optimal latent representation $\boldsymbol{V}$, the hashing codes $\boldsymbol{Y}$ and modality-specific hashing functions $\boldsymbol{H}^t$ can be generated using Eqn.3.1. It is obvious that the quantization error can be measured as $\|\boldsymbol{Y} - \boldsymbol{V}\|_F^2$. Inspired by [36], we propose to further improve the hashing codes by minimizing this quantization error using an orthogonal rotation. We first prove the following orthogonal invariant theorem.

Theorem 3.3.1 *Assume $\boldsymbol{Q}$ is a $k \times k$ orthogonal matrix, i.e., $\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}$. If $\boldsymbol{V}$ and $\boldsymbol{B}$ are an optimal solution to the problem in Eqn.3.7, then $\boldsymbol{VQ}$ and $\boldsymbol{Q}^T\boldsymbol{B}$ are also an optimal solution.*

Proof By substituting $\boldsymbol{VQ}$ and $\boldsymbol{Q}^T\boldsymbol{B}$ into Eqn.3.7, it is obvious that: $\|\bar{\boldsymbol{X}}^{(t)} - \bar{\boldsymbol{V}}^{(t)}\boldsymbol{Q}\boldsymbol{Q}^T\boldsymbol{B}^{(t)}\|_F^2 = \|\bar{\boldsymbol{X}}^{(t)} - \bar{\boldsymbol{V}}^{(t)}\boldsymbol{B}^{(t)}\|_F^2$, $tr\left((\bar{\boldsymbol{V}}^{(t)}\boldsymbol{Q})^T\boldsymbol{L}^{(t)}\bar{\boldsymbol{V}}^{(t)}\boldsymbol{Q}\right) = tr\left(\boldsymbol{Q}^T\bar{\boldsymbol{V}}^{(t)T}\boldsymbol{L}^{(t)}\bar{\boldsymbol{V}}^{(t)}\boldsymbol{Q}\right)$ $= tr\left(\bar{\boldsymbol{V}}^{(t)T}\boldsymbol{L}^{(t)}\bar{\boldsymbol{V}}^{(t)}\right)$, and $\|\boldsymbol{VQ}\|_F^2 = \|\boldsymbol{V}\|_F^2$, $\|\boldsymbol{Q}^T\boldsymbol{B}\|_F^2 = \|\boldsymbol{B}\|_F^2$. Thus, the value of the objective function in Eqn.3.7 does not change by the orthogonal rotation. ■

According to the above theorem, we propose to seek for better hashing codes by minimizing the quantization error between the binary hashing codes $\boldsymbol{Y}$ and the orthogonal rotation of the latent representation $\boldsymbol{VQ}$ as follows:

$$\begin{aligned} &\min_{\boldsymbol{Y},\boldsymbol{Q}} \|\boldsymbol{Y} - \boldsymbol{VQ}\|_F^2 \\ s.t. \quad &\boldsymbol{Y} \in \{-1, 1\}^{N\times k}, \quad \boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I} \end{aligned} \tag{3.13}$$

Intuitively, we seek binary codes that are close to some orthogonal transformation of the latent representation. The orthogonal rotation not only preserves the optimality of the solution but also provides us more flexibility to achieve better hashing codes with low quantization error. The idea of orthogonal rotation is also utilized in ITQ [36]. However, ITQ method is not designed for handling partial multi-modal data and it does not preserve the local similarities among data examples. The above optimization problem can be solved by minimizing Eqn.3.13 with respect to $\boldsymbol{Y}$ and $\boldsymbol{Q}$ alternatively.

Fix Q and update Y. The closed form solution can be expressed as:

$$\boldsymbol{Y} = sgn\left(\boldsymbol{VQ}\right) \tag{3.14}$$

which is identical with Eqn.3.1 except the rotation.

Fix Y and update Q. The objective function becomes:

$$\min_{\boldsymbol{Q}^T\boldsymbol{Q}=\boldsymbol{I}} \|\boldsymbol{Y} - \boldsymbol{VQ}\|_F^2 \tag{3.15}$$

In this case, the objective function is essentially the classic Orthogonal Procrustes problem [80], which can be solved efficiently by singular value decomposition using the following theorem.

Theorem 3.3.2 *Let $\boldsymbol{S\Lambda U}^T$ be the singular value decomposition of $\boldsymbol{Y}^T\boldsymbol{V}$. Then $\boldsymbol{Q} = \boldsymbol{US}^T$ minimizes the objective function in Eqn.3.15.*

We perform the above two steps alternatively to obtain the optimal hashing codes and the orthogonal rotation matrix.[1] The modality-specific hashing functions can be then derived by minimizing the projection error as:

$$\min_{\boldsymbol{H}^t} \|\boldsymbol{H}^t\bar{\boldsymbol{X}}^{(t)} - \bar{\boldsymbol{V}}^{(t)}\boldsymbol{Q}\|_F^2 + \gamma\|\boldsymbol{H}^t)\|_F^2 \quad t = 1,2 \tag{3.16}$$

where γ is the tradeoff parameter of the regularization term. The full learning algorithm is described in Table 3.1.

Table 3.1.
Partial Multi-Modal Hashing (PM^{2}H)

```
Input: Partial data $\{\hat{\boldsymbol{X}}^{(1,2)}, \hat{\boldsymbol{X}}^{(1)}, \hat{\boldsymbol{X}}^{(2)}\}$, trade-off parameters $\alpha$, $\lambda$ and $\gamma$
Output: Unified hashing codes $\boldsymbol{Y}$ and hashing functions $\boldsymbol{H}^1$, $\boldsymbol{H}^2$
Initialize $\boldsymbol{B}$ using Eqn.3.17, Calculate $\boldsymbol{L}$.
Repeat
  Optimize Eqns.3.8 and 3.9 and update $\hat{\boldsymbol{V}}_c$, $\hat{\boldsymbol{V}}^{(1)}$ and $\hat{\boldsymbol{V}}^{(2)}$.
  Optimize Eqn.3.12 and update $\boldsymbol{B}^{(1)}$ and $\boldsymbol{B}^{(2)}$.
Until the solution converges
Repeat
  Update $\boldsymbol{Y}$ using Eqn.3.14.
  Update $\boldsymbol{Q} = \boldsymbol{US}^T$ according to Theorem 2.
Until the solution converges
Obtain the hashing functions $\boldsymbol{H}^1$ and $\boldsymbol{H}^2$ from Eqn.3.16.
```

[1] In our experiments, we find that the algorithm usually converges in about 30∼60 iterations.

3.3.6 Analysis

This section provides some complexity analysis on the training cost of the learning algorithm. The optimization algorithm of PM^{2}H consists of two main loops. In the first loop, we iteratively solve $\boldsymbol{V}$ and $\boldsymbol{B}$ to obtain the optimal solution, where the time complexities for solving $\boldsymbol{V}$ and $\boldsymbol{B}$ are bounded by $O(Nkd_1 + Nkd_2 + Nk^2 + N^2k)$ and $O(Nk^2 + Nkd_1 + Nkd_2)$ respectively. The second loop iteratively optimizes the binary hashing codes and the orthogonal rotation matrix, where the time complexities for updating $\boldsymbol{Y}$ and $\boldsymbol{Q}$ are bounded by $O(Nk^2 + k^3)$. Thus, the total time complexity of the learning algorithm is bounded by $O(Nkd_1 + Nkd_2 + N^2k + Nk^2 + k^3)$. For each query, the hashing time is constant $O(d_1k)$ and $O(d_2k)$.

We also want to point out that the efficiency of the iterative coordinate descent optimization is greatly affected by the initialization step. Therefore in this dissertation, we learn the initial value of $\boldsymbol{B}$ rather than random assignment as follows:

$$\begin{aligned}\min_{\boldsymbol{B}^{(1)},\boldsymbol{B}^{(2)},\hat{\boldsymbol{V}}_c} \; & \|\hat{\boldsymbol{X}}_c^{(1)} - \hat{\boldsymbol{V}}_c\boldsymbol{B}^{(1)}\|_F^2 + \|\hat{\boldsymbol{X}}_c^{(2)} - \hat{\boldsymbol{V}}_c\boldsymbol{B}^{(2)}\|_F^2 \\ & +\lambda \; R(\boldsymbol{B}^{(1)}, \boldsymbol{B}^{(2)}, \hat{\boldsymbol{V}}_c)\end{aligned} \tag{3.17}$$

It can be seen that $\boldsymbol{B}^{(1)}$ and $\boldsymbol{B}^{(2)}$ are essentially initialized by applying standard multi-modal subspace learning methods on examples without partial modalities. The above optimization can also be solved iteratively. But the resulting subproblems are all convex with closed form solutions, which can be efficiently obtained. In our experiments, we find out that the proposed method converges much faster using this initialization than random assignment.

3.4 Experiments

3.4.1 Datasets

We evaluate our method on two image datasets: NUS-WIDE and MIRFLICKR-25k. NUS-WIDE[2] contains 270k images associated with more than 5k unique tags. 81

[2]http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm

Table 3.2.
Precision of top 100 retrieved examples with PDR=0.4.

modality 1	NUS-WIDE					MIRFLICKR-25k				
# of bits	8	16	32	64	128	8	16	32	64	128
PM^{2}H	**0.45**	**0.47**	**0.51**	**0.52**	**0.53**	**0.54**	**0.56**	**0.58**	**0.60**	**0.61**
CMFH	0.43	0.44	0.46	0.47	0.48	0.52	0.53	0.54	0.56	0.57
CCA-ITQ	0.39	0.41	0.43	0.44	0.44	0.45	0.47	0.48	0.49	0.51
CMSSH	0.36	0.38	0.40	0.41	0.42	0.40	0.41	0.42	0.42	0.43
CVH	0.28	0.30	0.32	0.33	0.33	0.43	0.45	0.47	0.47	0.47
modality 2	NUS-WIDE					MIRFLICKR-25k				
# of bits	8	16	32	64	128	8	16	32	64	128
PM^{2}H	**0.42**	**0.44**	**0.46**	**0.47**	**0.48**	**0.55**	**0.57**	**0.59**	**0.61**	**0.62**
CMFH	0.38	0.40	0.41	0.43	0.43	0.50	0.52	0.53	0.54	0.55
CCA-ITQ	0.34	0.36	0.37	0.38	0.39	0.50	0.51	0.52	0.53	0.54
CMSSH	0.35	0.37	0.39	0.38	0.38	0.47	0.49	0.50	0.50	0.50
CVH	0.31	0.33	0.34	0.35	0.35	0.45	0.46	0.48	0.47	0.47

ground-truth concepts are annotated on these images. We filter out those images with less than 10 tags, resulting in a subset of 110k image examples. Visual features are represented by 500-dimension SIFT [85] histograms, and text features are represented by index vectors of the most common 2k tags. We use 90% of the data as the training set and the rest 10% as the query set. MIRFLICKR-25k[3] is collected from Flicker images for image retrieval tasks. This dataset contains 25k image examples associated with 38 unique labels. 100-dimensional SIFT descriptors and 512-dimensional GIST descriptors [87] are extracted from these images as the two modalities. We randomly choose 23k image examples as the training set and 2k for testing. Two image examples are considered to be similar if they share at least one ground-truth concept/label. In

[3]http://press.liacs.nl/mirflickr/

our experiments, SIFT feature is viewed as modality 1, while text and GIST features are viewed as modality 2.

To simulate the partial modality setting, we randomly select a fraction of training examples to be partial examples, i.e., they are represented by either of the modality but not both, and the remaining ones appear in both modalities. We refer the fraction number of partial examples as Partial Data Ratio (PDR), i.e., $\frac{m+n}{N}$.

3.4.2 Evaluation Method

Hamming Ranking ranks all the examples in the database according to their Hamming distance from the query and the top k examples are returned as the desired neighbors. Hash Lookup returns all the examples within a small Hamming radius r of the query. For Hamming Ranking based evaluation, we calculate the precision at top k which is the percentage of true neighbors among the top k returned examples, where we set k to be 100 in the experiments. For Hash Lookup, the precision of the returned examples falling within Hamming radius $r = 2$ is recorded. Note that the hashing code for a query can be generated from either modality (using $\boldsymbol{H}^1$ or $\boldsymbol{H}^2$), therefore we report the precision results on both modalities.

3.4.3 Baselines and Setting

The proposed PM^{2}H approach is compared with four different multi-modal hashing methods, i.e., CVH [40], CMSSH [39], CCA-ITQ [35, 36] and CMFH [43].[4] We implement our algorithm using Matlab on a PC with Intel Duo Core i5-2400 CPU 3.1GHz and 8GB RAM. The parameters α, λ and γ are tuned by 5-fold cross validation on the training set. To remove any randomness caused by random selection of training set, all of the results are averaged over 10 runs.

[4]We implement CVH and obtain the codes of CMSSH and CMFH from the authors. The code of CCA-ITQ is public available.

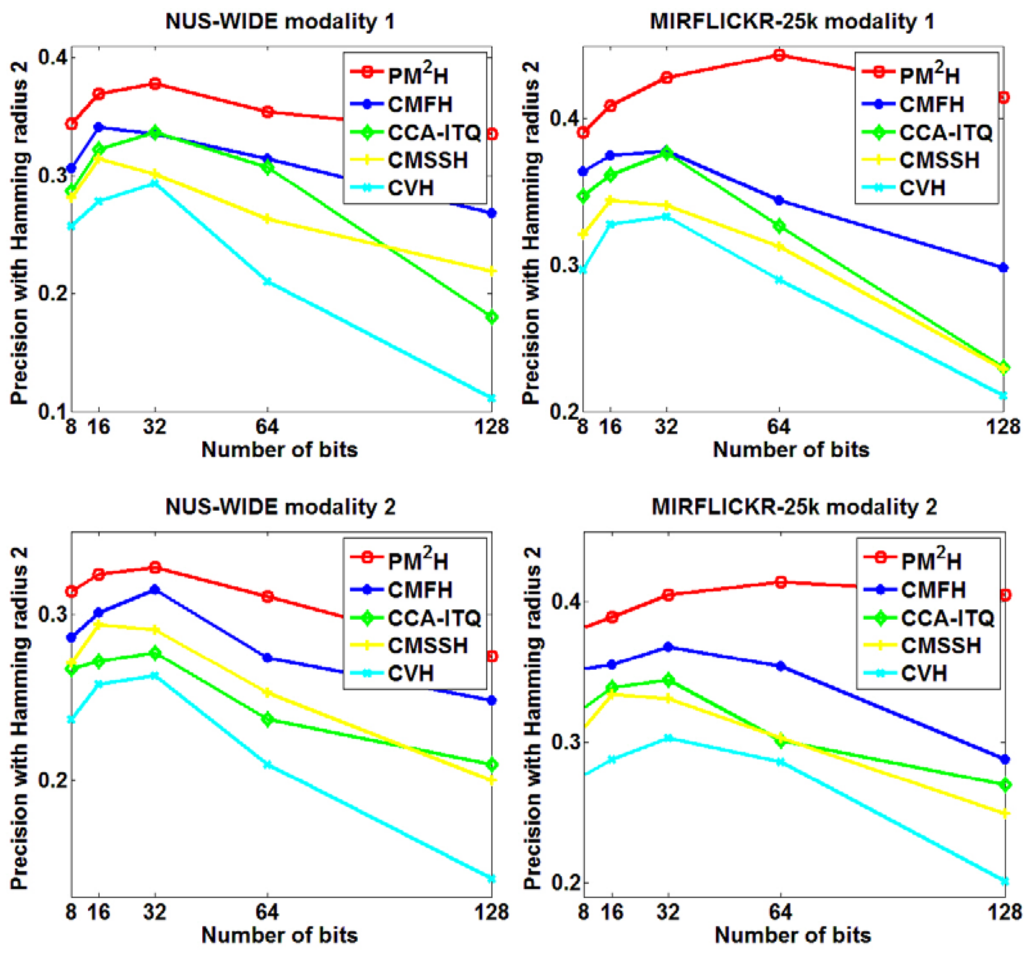


Figure 3.1. Precision within Hamming radius 2 using different hashing bits with PDR=0.4.

3.4.4 Results and Discussion

We first evaluate the performance of different methods by varying the number of hashing bits in the range of $\{8, 16, 32, 64, 128\}$, with fixed PDR 0.4. To apply the compared multi-modal hashing methods to the partial data, a simple way is to fill in the missing data with 0. However, this may result in large fitting errors between two modalities for the multi-modal methods, since the hashing code for the missing instance will be 0. Therefore, to achieve stronger baseline results, we replace the missing instance using the linear combination of its 5[5] nearest neighbor examples (weighed by their similarities) which appear in both modalities. Then the baseline multi-modal hashing methods can be directly applied on these extended data.

The precisions for the top 100 retrieved examples are reported in Table 3.2. We also show the precision results for retrieved examples within Hamming radius 2 in Fig.3.1. From these comparison results, we can see that PM^{2}H provides the best results among all five hashing methods on both datasets. For example, the precision of PM^{2}H increases over 8% and 15% on average compared with CMFH and CCA-ITQ on NUS-WIDE under modality 1. The reason is that PM^{2}H can effectively handle the partial data by common subspace learning between modalities and similarity preservation within modality, while the compared methods fail to accurately extract a common space from the partial examples. It can be seen from Table 3.2 that CMSSH and CVH do not perform well especially with 64 or 128 bits. This phenomenon has also been observed in [12, 43]. Actually, in CMSSH and CVH methods, the hashing codes are learned by eigenvalue decomposition under the hard bit orthogonality constraint, which makes the first few projection directions very discriminative with high variance. However, the hashing codes will be dominated by bits with very low variance when the code length increases, resulting in many meaningless and ambiguous bits. We also observe from Fig.3.1 that the precision of Hash Lookup for most of the compared methods decreases significantly with the increasing number of hashing bits. The reason is that the Hamming space becomes increasingly sparse with longer hashing bits and very few data points fall within the Hamming ball with radius 2, which makes many queries have 0 precision results. However, the precision of PM^{2}H is still consistently higher than the other methods. Another interesting observation is that the retrieval result from modality 1 is better than that from modality 2 on NUS-WIDE. This coincides with our expectation that the image modality is more informative than the tag modality since tags are usually noisy and incomplete.

[5]We empirically choose 5 in our experiments. But other numbers can also be applied.

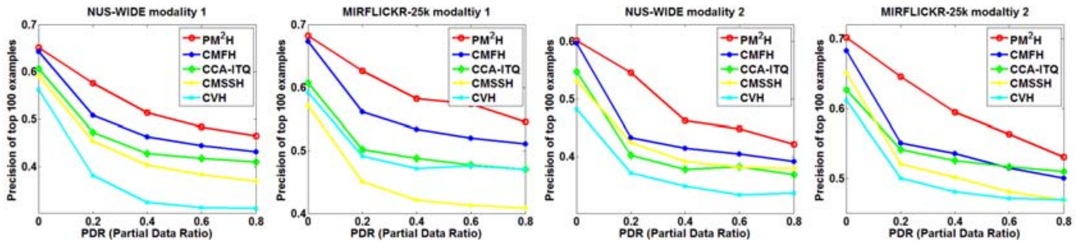


Figure 3.2. Precision of top 100 retrieved examples under different PDRs with 32 bits.

To evaluate the effectiveness of the proposed PM^{2}H under different partial data ratios, we progressively increase the PDR from {0, 0.2, 0.4, 0.6, 0.8} and compare our method with the other baselines by fixing the hashing bits to 32. The precision results of top 100 retrieved examples are shown in Fig.3.2. It can be seen from the figure that when the partial data ratio PDR is 0, the data actually becomes the traditional multi-modal setting with each example appears in both modalities. In this case, PM^{2}H is also able to perform better than most baselines and is comparable with CMFH. As the PDR increases from 0 to 0.8, our PM^{2}H approach always achieves the best performance among all compared methods. Although the missing instances are recovered from the common examples in both modalities, the baseline methods seem less effective in the modality missing case. Our hypothesis is that the missing data may not be accurately recovered when the data are missing blockwise for the partial data setting. In other words, the missing examples can be dissimilar to all the examples appear in both modalities.

We also evaluate the code effectiveness with and without orthogonal rotation. The comparison results (before and after rotation) in Table 3.3 demonstrate that the orthogonal rotation can further improve the effectiveness of the codes, which is consistent with our expectation since the quantization error is minimized through the rotation. Similar results on modality 2 are observed. But due to space limit, they are not presented here.

Table 3.3.
Precision of top 100 examples before and after orthogonal rotation with PDR=0.4 on modality 1.

mod 1	NUS-WIDE			MIRFLICKR-25k		
bits	16	32	64	16	32	64
After	0.476	0.514	0.522	0.567	0.582	0.601
Before	0.463	0.504	0.515	0.552	0.566	0.587

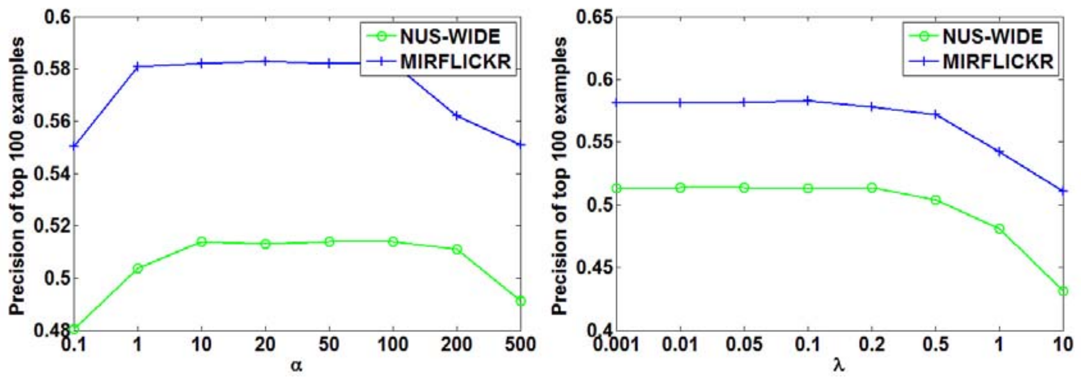


Figure 3.3. Parameter sensitivity results with 32 bits under PDR=0.4.

To prove the robustness of the proposed method, we conduct parameter sensitivity experiments on both datasets for α and λ. In each experiment, we tune only one parameter while fixing the other one to the optimal values obtained from the previous experiments. We report the results in Fig.3.3 with 32 bits and PDR to be 0.4. It is clear from these experimental results that the performance of PM^{2}H is relatively stable with respect to $\alpha \in (2, 100)$ and $\lambda \in (0.001, 0.1)$.

4 LEARNING TO HASH ON STRUCTURED DATA

4.1 Motivation

Hashing techniques have been widely applied for large scale similarity search problems due to the computational and memory efficiency. However, most existing hashing methods assume data examples are independently and identically distributed. But there often exists various additional dependency/structure information between data examples in many real world applications. This structure information have been utilized in clustering [46] and classification [38] problems, and proven to be helpful knowledge. Ignoring this structure information may limit the performance of existing hashing algorithms. Therefore, it is important to design hashing method that preserves the structure information among data examples in the learned Hamming space.

4.2 Background and Related Work

4.2.1 Introduction

With the explosive growth of the Internet, a huge amount of data has been generated, which indicates that efficient similarity search becomes more important. Traditional similarity search methods are difficult to be directly used for large scale applications since linear scan between query example and all candidates in the database is impractical. Moreover, the similarity between data examples is usually conducted in high dimensional space. Recently, hashing methods [8, 12, 48, 53, 95–97] are proposed to address the similarity search problem within large scale data. These hashing methods design compact binary code in a low-dimensional space for each data example so that similar examples are mapped to similar binary codes. In the retrieval process, these hashing methods first transform each query example into its corresponding binary code. Then similarity search can be simply conducted by calculating the Hamming distances between the codes of available data examples and the query and selecting data examples within small Hamming distances, which can be calculated using efficient bitwise operator XOR.

Hashing methods generate promising results by successfully addressing the storage and search efficiency challenges. However, most existing hashing methods assume that data examples are independently and identically distributed. But in many applications, the dependencies between data examples naturally exist and if incorporated in models, they can potentially improve the hashing code performance significantly. For example, many webpages have hyperlinks pointing to other related webpages (see Fig.1.3(a)). The contents of these linked webpages are usually relevant, which present similar topics. The hyperlinks among webpages provide important structure knowledge. Another example is that similar images often share semantic labels (see Fig.1.3(b)). The more labels two images have in common, the more similar the images are. The shared semantic labels among images offer valuable information in binary codes learning. These structure information have been utilized in clustering [46] and classification [38] problems, and proven to be helpful knowledge. Therefore, it is important to design hashing method that preserve the structure information among data examples in the learned Hamming space.

This dissertation proposes a novel approach of learning to Hash on Structured Data (HSD) that incorporates the structure information associated with data. The hashing function is learned in a unified learning framework by simultaneously ensuring the structural consistency and preserving the similarities between data examples. In particular, the objective function of the proposed HSD approach is composed of two parts: (1) Structure consistency term, which ensures the hashing codes to be consistent with the structure information. (2) Similarity preservation term, which aims at preserving the similarity between data examples in the learned hashing codes. An iterative gradient descent algorithm is designed as the optimization procedure. We further improve the quality of hashing function by minimizing the quantization error. Experimental results on two datasets demonstrate the superior performance of the proposed method over several state-of-the-art hashing methods.

4.2.2 Related Work

Locality-Sensitive Hashing (LSH) [15] is one of the most commonly used data-independent hashing methods. It utilizes random linear projections, which are independent of training data, to map data points from a high-dimensional feature space to a low-dimensional binary space. This method has been extended to Kernelized Locality-Sensitive Hashing [21] by exploiting kernel similarity for better retrieval efficacy. Another class of hashing methods are called data-dependent methods, whose projection functions are learned from training data. These data-dependent methods include spectral hashing (SH) [19], principal component analysis based hashing (PCAH) [17], self-taught hashing (STH) [22] and iterative quantization (ITQ) [36]. SH learns the hashing codes based on spectral graph partitioning and forcing the balanced and uncorrelated constraints into the learned codes. PCAH utilizes principal component analysis (PCA) to learn the projection functions. STH combines an unsupervised learning step with a supervised learning step to learn effective hashing codes. ITQ learns an orthogonal rotation matrix to refine the initial projection matrix learned by PCA so that the quantization error of mapping the data to binary codes is minimized. Compared with the data-independent methods, these data-dependent methods generally provide more effective hashing codes.

Recently, supervised hashing methods [20, 30, 31, 94, 98] have incorporated labeled data/information, e.g. semantic tags, for learning more effective hashing function. For example, in semi-supervised hashing [29] method, pairwise similarity constraints are imposed in the learning framework. In work [94], tags are incorporated to obtain more effective hashing codes via a matrix factorization formulation. More recently, some multi-view hashing methods [10, 35, 43] have been proposed to deal with multi-modal data for cross-view similarity search. These multi-view methods can be applied by treating the structure information as second view. However, structure information is usually very sparse, e.g., each webpage may contain very few hyperlinks. Directly using it as a second information source can lead to unreliable results. More discussion will be provided later in the experiments.

4.3 Algorithm

4.3.1 Problem Setting

Before presenting the details, we first introduce some notations. Assume there are total n training examples. Let us denote their features as: $\boldsymbol{X} = \{x_1, x_2, \ldots, x_n\} \in R^{d \times n}$, where d is the dimensionality of the feature. A directed or undirected graph $G = (V, E)$ is used to depict the structure between data examples. Each node $v \in V$ corresponds to a data example, and an edge $e = (i, j) \in E$ with a weight w_{ij} represents a link/connection between nodes i and j. The larger the weight w_{ij} is, the more relevant x_i and x_j should be. We will discuss how to assign w later. The goal is to obtain a linear hashing function $f : \mathbb{R}^d \to \{-1, 1\}^k$, which maps data examples $\boldsymbol{X}$ to their binary hashing codes $\boldsymbol{Y} = \{y_1, y_2, \ldots, y_n\} \in \{-1, 1\}^{k \times n}$ (k is the length of hashing code). The linear hashing function is defined as:

$$y_i = f(x_i) = sgn(\boldsymbol{H}^T x_i) \tag{4.1}$$

where $\boldsymbol{H} \in \mathbb{R}^{d \times k}$ is the coefficient matrix representing the hashing function and sgn is the sign function. $y_i \in \{-1, 1\}^k$ is the binary hashing code[1] of x_i.

The objective function of HSD is composed of two components: (1) Structure consistency, which ensures that the hashing codes are consistent with the structure information. (2) Similarity preservation, which aims at preserving the data similarity in the learned hashing codes. In the rest of this section, we will present the formulation of these two components respectively. Then we will describe the optimization

[1]We generate hashing bits as $\{-1, 1\}$, which can be simply converted to $\{0, 1\}$ valued hashing codes.

algorithm together with a scheme that can further improve the quality of the hashing function by minimizing the quantization error.

4.3.2 Structure Consistency

Our motivation is that the similarity between the learned hashing codes should agree or be consistent with the structure information defined on the graph G. Specifically, a pair of nodes linked by an edge tend to have similar hashing codes. The larger the weight between nodes i and j, the smaller the Hamming distance between their codes should be. For webpages, we define the weight w_{ij} associated with edge (i, j) to be the number of hyperlinks between the two webpages. Similarly for images, we assign weight w_{ij} using the number of common labels shared by image x_i and x_j.

Then a structure consistency component can be directly formulated as:

$$\sum_{(i,j)\in E} w_{ij} d_H(y_i, y_j) = \frac{1}{4} \sum_{(i,j)\in E} w_{ij} \|y_i - y_j\|^2 \tag{4.2}$$

where $d_H(y_i, y_j)=\frac{1}{4}\|y_i - y_j\|^2$ is the Hamming distance between binary codes y_i and y_j. This definition says that for each linked node pair (i, j), the Hamming distance between the corresponding hashing codes y_i and y_j should be consistent with the edge weight w_{ij}. In other words, we assign a heavy penalty if two strongly connected data examples are mapped far away.

By substituting Eqn.4.1 with some additional mathematical operations, the above equation can be rewritten as a compact matrix form as:

$$tr\left(\boldsymbol{Y}\bar{\boldsymbol{W}}\boldsymbol{Y}^T\right) = tr\left(sgn(\boldsymbol{H}^T\boldsymbol{X})\bar{\boldsymbol{W}}sgn(\boldsymbol{X}^T\boldsymbol{H})\right) \tag{4.3}$$

where $tr()$ is the matrix trace function. $\bar{\boldsymbol{W}} = \boldsymbol{D} - \boldsymbol{W}$ is called graph *Laplacian* [19] and $\boldsymbol{D}$ is a diagonal $n \times n$ matrix whose entries are given by $\boldsymbol{D}_{ii} = \sum_{j=1}^{n} w_{ij}$. By minimizing the structure consistency term, structure information is well preserved in Hamming space by hashing function $\boldsymbol{H}$.

4.3.3 Similarity Preservation

One of the key problems in hashing algorithms is similarity preserving, which indicates that similar data examples should be mapped to similar hashing codes within a short Hamming distance. To measure the similarity between data examples represented by the binary hashing codes, one natural way is to minimize the weighted average Hamming distance as follows:

$$\sum_{i,j} \boldsymbol{S}_{ij} \|y_i - y_j\|^2 \tag{4.4}$$

Here, $\boldsymbol{S_{ij}}$ is the pairwise similarity between data example x_i and x_j. To meet the similarity preservation criterion, we seek to minimize this quantity, because it incurs a heavy penalty if two similar examples have very different hashing codes.

There are many different ways of defining the similarity matrix $\boldsymbol{S}$. In SH [19], the authors used the global similarity structure of all data pairs, while in [10], the local similarity structure, i.e., k-nearest-neighborhood, is used. In this dissertation, we use the local similarity, due to its nice property in many machine learning applications. In particular, the corresponding weights are computed by Gaussian functions:

$$\boldsymbol{S}_{ij} = \begin{cases} e^{-\frac{\|x_i - x_j\|^2}{\sigma_{ij}^2}}, & if\ x_i \in N_k(x_j)\ \ or\ \ x_j \in N_k(x_i) \\ 0, & otherwise \end{cases} \tag{4.5}$$

The variance σ_{ij} is determined automatically by local scaling [71], and $N_k(x)$ represents the set of k-nearest-neighbors of data example x. Similarly, Eqn.4.4 can be rewritten as a compact form:

$$tr\left(\boldsymbol{Y}\bar{\boldsymbol{S}}\boldsymbol{Y}^T\right) = tr\left(sgn(\boldsymbol{H}^T\boldsymbol{X})\bar{\boldsymbol{S}}sgn(\boldsymbol{X}^T\boldsymbol{H})\right) \tag{4.6}$$

By minimizing this term, the similarity between different data examples can be preserved in the learned hashing codes.

4.3.4 Overall Objective

The entire objective function consists of three components: the structure consistency term in Eqn.4.3, the similarity preservation term given in Eqn.4.6 and an orthogonal constraint term as follows:

$$\begin{aligned}\min_{\boldsymbol{H}} \quad & tr\left(sgn(\boldsymbol{H}^T\boldsymbol{X})\bar{\boldsymbol{W}}sgn(\boldsymbol{X}^T\boldsymbol{H})\right) \\ +\alpha\ tr & \left(sgn(\boldsymbol{H}^T\boldsymbol{X})\bar{\boldsymbol{S}}sgn(\boldsymbol{X}^T\boldsymbol{H})\right) + \beta\ \|\boldsymbol{H}^T\boldsymbol{H} - \boldsymbol{I}\|_F^2\end{aligned} \tag{4.7}$$

where α and β are trade-off parameters to balance the weights among the terms. The orthogonality constraint enforce the hashing bits to be uncorrelated with each other and therefore the learned hashing codes can hold least redundant information.

4.3.5 Relaxation

Directly minimizing the objective function in Eqn.4.7 is intractable since it is an integer programming problem, which is proven to be NP-hard to solve. Therefore, we use the signed magnitude instead of the sign function as suggested in [31, 94]. Then the relaxed objective function becomes:

$$\min_{\tilde{\boldsymbol{H}}} \quad tr\left(\tilde{\boldsymbol{H}}^T\boldsymbol{L}\tilde{\boldsymbol{H}}\right) + \beta\ \|\tilde{\boldsymbol{H}}^T\tilde{\boldsymbol{H}} - \boldsymbol{I}\|_F^2 \tag{4.8}$$

where $\boldsymbol{L} \equiv \boldsymbol{X}(\bar{\boldsymbol{W}} + \alpha\bar{\boldsymbol{S}})\boldsymbol{X}^T$ and can be pre-computed. $\tilde{\boldsymbol{H}}$ represents the relaxed solution. Although the relaxed objective in Eqn.4.8 is still non-convex, it is smooth and differentiable which enables gradient descent methods to be applied for efficient optimization. The gradients of the two terms with respect to $\tilde{\boldsymbol{H}}$ are given below:

$$\frac{d\ Eqn.4.8}{d\ \tilde{\boldsymbol{H}}} = 2\boldsymbol{L}\tilde{\boldsymbol{H}} + 4\beta\tilde{\boldsymbol{H}}(\tilde{\boldsymbol{H}}^T\tilde{\boldsymbol{H}} - \boldsymbol{I}) \tag{4.9}$$

With this obtained gradient, L-BFGS quasi-Newton method [79] is applied to solve the optimization problem.

4.3.6 Orthogonal Transformation

After obtaining the optimal hashing function $\tilde{\boldsymbol{H}}$ from the relaxation, the hashing codes $\boldsymbol{Y}$ can be generated using Eqn.4.1. It is obvious that the quantization error can be measured as $\|\boldsymbol{Y} - \tilde{\boldsymbol{H}}^{\boldsymbol{T}}\boldsymbol{X}\|_F^2$. Inspired by [36], we propose to further improve the hashing function by minimizing this quantization error using an orthogonal transformation. We first prove the following orthogonal invariant theorem.

Theorem 4.3.1 *Assume $\boldsymbol{Q}$ is a $k \times k$ orthogonal matrix, i.e., $\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}$. If $\tilde{\boldsymbol{H}}$ is an optimal solution to the relaxed problem in Eqn.4.8, then $\tilde{\boldsymbol{H}}\boldsymbol{Q}$ is also an optimal solution.*

Proof By substituting $\tilde{\boldsymbol{H}}\boldsymbol{Q}$ into Eqn.4.8, we have:
$tr\left((\tilde{\boldsymbol{H}}\boldsymbol{Q})^T\boldsymbol{L}\tilde{\boldsymbol{H}}\boldsymbol{Q}\right) = tr\left(\boldsymbol{Q}^T\tilde{\boldsymbol{H}}^T\boldsymbol{L}\tilde{\boldsymbol{H}}\boldsymbol{Q}\right) = tr\left(\tilde{\boldsymbol{H}}^T\boldsymbol{L}\tilde{\boldsymbol{H}}\right)$, and $\|(\tilde{\boldsymbol{H}}\boldsymbol{Q})^T\tilde{\boldsymbol{H}}\boldsymbol{Q} - \boldsymbol{I}\|_F^2 = \|\boldsymbol{Q}^T(\tilde{\boldsymbol{H}}^T\tilde{\boldsymbol{H}} - \boldsymbol{I})\boldsymbol{Q}\|_F^2 = \|\tilde{\boldsymbol{H}}^T\tilde{\boldsymbol{H}} - \boldsymbol{I}\|_F^2$.
Thus, the value of the objective function in Eqn.4.8 does not change by the orthogonal transformation. ■

According to the above theorem, we propose to find a better hashing function $\boldsymbol{H} = \tilde{\boldsymbol{H}}\boldsymbol{Q}$ by minimizing the quantization error between the binary hashing codes and the orthogonal transformation of the relaxed solution as follows:

$$\begin{aligned} &\min_{\boldsymbol{Y},\boldsymbol{Q}} \|\boldsymbol{Y} - (\tilde{\boldsymbol{H}}\boldsymbol{Q})^T\boldsymbol{X}\|_F^2 \\ s.t. \quad &\boldsymbol{Y} \in \{-1,1\}^{k\times n}, \quad \boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I} \end{aligned} \tag{4.10}$$

Intuitively, we seek binary codes that are close to some orthogonal transformation of the relaxed solution. The orthogonal transformation not only preserves the optimality of the relaxed solution but also provides us more flexibility to achieve better hashing codes with low quantization error. The idea of orthogonal transformation is also utilized in ITQ [36]. However, ITQ method is not designed for incorporating structure information into learning effective hashing function and it does not preserve the local similarities among data examples. The above optimization problem can be solved by minimizing Eqn.4.10 with respect to $\boldsymbol{Y}$ and $\boldsymbol{Q}$ alternatively as follows:

Fix Q **and update** Y. The closed form solution can be expressed as:

$$\boldsymbol{Y} = sgn\left((\tilde{\boldsymbol{H}}\boldsymbol{Q})^T\boldsymbol{X}\right) = sgn(\boldsymbol{H}^T\boldsymbol{X}) \tag{4.11}$$

which is identical with our linear hashing function in Eqn.4.1.

Fix Y **and update** Q. The objective function becomes:

$$\min_{\boldsymbol{Q}^T\boldsymbol{Q}=\boldsymbol{I}} \|\boldsymbol{Y} - \boldsymbol{Q}^T\tilde{\boldsymbol{H}}^T\boldsymbol{X}\|_F^2 \tag{4.12}$$

In this case, the objective function is essentially the classic Orthogonal Procrustes problem [80], which can be solved efficiently by singular value decomposition using the following theorem (we refer to [80] for the detailed proof).

Theorem 4.3.2 *Let* $\boldsymbol{S\Lambda V}^T$ *be the singular value decomposition of* $\boldsymbol{YX}^T\tilde{\boldsymbol{H}}$. *Then* $\boldsymbol{Q} = \boldsymbol{VS}^T$ *minimizes the objective function in Eqn.4.12.*

We then perform the above two steps alternatively to obtain the optimal hashing codes and the orthogonal transform matrix. In our experiments, we find that the algorithm usually converges in about 40∼60 iterations. The full learning algorithm is described in Table 4.1.

4.3.7 Complexity Analysis

This section provides some analysis on the training cost of the optimization algorithm. The optimization algorithm of HSD consists of two main loops. In the first loop, we iteratively solve for $\tilde{\boldsymbol{H}}$ to obtain the relaxed solution, where the time complexities for computing the gradient in Eqn.4.9 are bounded by $O(nkd + nk^2)$. The second loop iteratively optimizes the binary hashing codes and the orthogonal transformation matrix, where the time complexities for updating $\boldsymbol{Y}$ and $\boldsymbol{Q}$ are bounded by $O(nk^2 + nkd + k^3)$. Moreover, both two loops take less than 60 iterations to converge in our experiments. Thus, the total time complexity of the learning algorithm is bounded by $O(nkd + nk^2 + k^3)$, which scales linearly with n given $n \gg d > k$. For each query, the hashing time is constant $O(dk)$.

Table 4.1.
Hashing on Structured Data (HSD)

```
Input: Data examples $\boldsymbol{X}$, Structure graph $G$ and trade-off parameters.
Output: Hashing function $\boldsymbol{H}$ and Hashing codes $\boldsymbol{Y}$.
Initialize $\boldsymbol{H}$ and $\boldsymbol{Q} = \boldsymbol{I}$, Calculate $\boldsymbol{L}$.
Repeat
    Compute the gradient in Eqn.4.9 and update $\tilde{\boldsymbol{H}}$
Until the solution converges
Repeat
    Update $\boldsymbol{Y}$ using Eqn.4.11.
    Update $\boldsymbol{Q} = \boldsymbol{V}\boldsymbol{S}^T$ according to Theorem 2.
Until the solution converges
Compute hashing function $\boldsymbol{H} = \tilde{\boldsymbol{H}}\boldsymbol{Q}$.
```

4.4 Experiments

4.4.1 Datasets and Setting

We evaluate our method on two datasets: $WebKB$ and *NUS-WIDE*. $WebKB$[2] contains 8280 webpages in total collected from four universities. The webpages without any incoming and outgoing links are deleted, resulting in a subset of 6883 webpages. The tf-idf (normalized term frequency and log inverse document frequency) [99] features are extracted for each webpage. 90% documents (6195) are randomly selected as training data, while the remaining (688) documents are used for testing. *NUS-WIDE*[3] is created by NUS lab for evaluating image annotation and retrieval techniques. It contains $270k$ images associated with 81 ground truth labels. A subset of $21k$ images associated with these semantic labels are used in our experiments. 500-dimensional visual features are extracted using a bag-of-visual-word model with

[2]http://www.cs.cmu.edu/∼WebKB
[3]http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm

local SIFT descriptor [85]. We randomly partition this dataset into two parts, $1k$ for testing and $20k$ for training.

We implement our algorithm using Matlab on a PC with Intel Duo Core i5-2400 CPU 3.1GHz and 8GB RAM. The parameter α and β are tuned by 5-fold cross validation through the grid $\{0.01, 0.1, 1, 10, 100\}$ on the training set and we will discuss more details on how it affects the performance of our approach later. We repeat each experiment 10 times and report the result based on the average over these runs. Each run adopts a random split of the dataset.

4.4.2 Comparison Methods

The proposed Hashing on Structure Data (HSD) approach is compared with five different hashing methods, including Locality Sensitivity Hashing (LSH) [15], Spectral Hashing (SH) [19], ITerative Quantization (ITQ) [36], Composite Hashing from Multiple Information Sources (CHMIS) [10] and Collective Matrix Factorization Hashing (CMFH) [43]. LSH, SH and ITQ methods do not use any structure knowledge for learning hashing codes. We use the standard settings in their papers for our experiments. For the multi-view hashing methods CHMIS and CMFH, the structure information is treated as the second view. Specifically, the link information from webpages and the binary labels on images are used as the additional view in these methods.

4.4.3 Evaluation Metrics

To conduct fair evaluation, we follow two criteria which are commonly used in the literature [29, 36]: Hamming Ranking and Hash Lookup. Hamming Ranking ranks all the points in the database according to their Hamming distance from the query and the top k points are returned as the desired neighbors. Hash Lookup returns all the points within a small Hamming radius r of the query. We use several metrics to measure the performance of different methods. For Hamming Ranking

Table 4.2.
Precision of the top 100 retrieved examples using Hamming Ranking on both datasets with different hashing bits.

	WebKB					*NUS-WIDE*				
bits	8	16	32	64	128	8	16	32	64	128
HSD	**0.606**	**0.669**	**0.732**	**0.763**	**0.786**	**0.406**	0.409	**0.445**	**0.478**	**0.493**
CMFH	0.571	0.635	0.650	0.704	0.722	0.371	**0.411**	0.427	0.436	0.468
CHMIS	0.511	0.558	0.613	0.646	0.674	0.334	0.367	0.369	0.373	0.386
ITQ	0.523	0.548	0.604	0.637	0.652	0.253	0.306	0.308	0.315	0.322
SH	0.504	0.513	0.536	0.541	0.547	0.251	0.264	0.282	0.297	0.304
LSH	0.339	0.377	0.389	0.387	0.401	0.234	0.226	0.247	0.258	0.261

based evaluation, we calculate the precision at top K which is the percentage of true neighbors among the top K returned examples, where we set K to be 100 in the experiments. A hamming radius of $R = 2$ is used to retrieve the neighbors for Hash Lookup. The precision of the returned examples falling within Hamming radius 2 is reported. Note that if a query returns no points inside Hamming ball with radius 2, it is treated as zero precision.

4.4.4 Results and Discussion

We first report precisions for the top 100 retrieved examples and the precisions for retrieved examples within Hamming ball with radius 2 by varying the number of hashing bits in the range of $\{8, 16, 32, 64, 128\}$ in Table 4.2 and Fig.4.1. From these comparison results, we can see that HSD provides the best results among all compared methods in most cases. LSH does not perform well since LSH method is data-independent, which may generate inefficient codes compared to those data-depend methods. SH and ITQ methods learn the hashing codes from data and

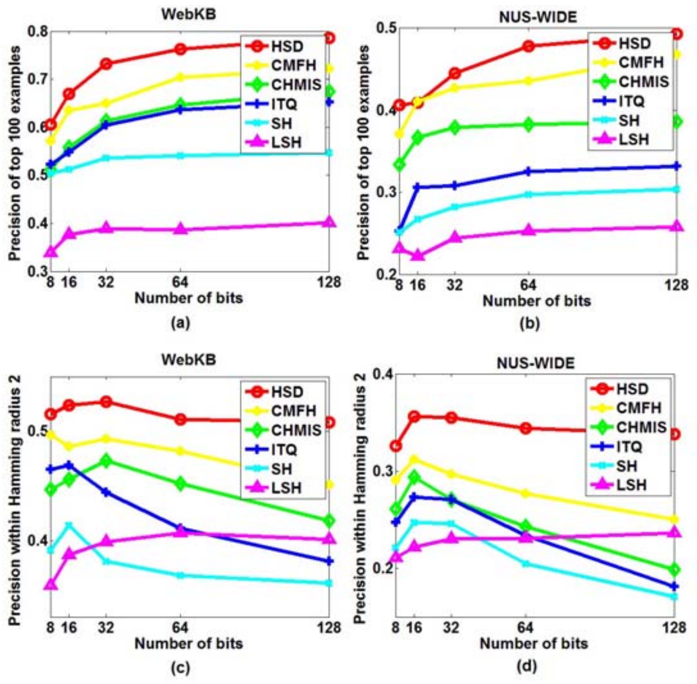


Figure 4.1. Precision on both datasets with different bits. (a)-(b): Precision of the top 100 returned examples using Hamming Ranking. (c)-(d): Precision within Hamming radius 2 using Hash Lookup.

try to preserve similarity between data examples. Thus they usually obtain higher precision results than LSH method. But both SH and ITQ methods do not utilize the structure information contained in data. CHMIS and CMFH methods achieve better performance than SH and ITQ due to incorporating structure information as an additional view into hashing codes learning. However, learning a common space between the two views by treating the structure as a second view may lead to unreliable results especially when structure information is very sparse or incomplete (more discussion will be provided later). Moreover, the data similarity is not well preserved in their hashing function learning. On the other hand, our HSD not only exploits structure information via modeling the structure consistency, but also preserves data similarity at the same time in the learned hashing function, which enables HSD to generate higher quality hashing codes than the hashing methods. In

Fig.4.1(c)-(d), we observe the precision of Hash Lookup for most of the compared methods decreases significantly with the increasing number of hashing bits. The reason is that the Hamming space becomes increasingly sparse with longer hashing bits and very few data points fall within the Hamming ball with radius 2, which makes many queries have 0 precision. However, the precision of HSD is still consistently higher than the other methods for Hash Lookup.

Table 4.3.
Precision of the top 100 examples under different training ratios on both datasets with 32 hashing bits.

	WebKB					*NUS-WIDE*				
ratio	0.2	0.4	0.6	0.8	1.0	0.2	0.4	0.6	0.8	1.0
HSD	**0.657**	**0.688**	**0.702**	**0.715**	**0.732**	**0.363**	**0.391**	**0.416**	**0.430**	**0.445**
CMFH	0.528	0.546	0.584	0.628	0.650	0.304	0.336	0.369	0.402	0.427
CHMIS	0.517	0.549	0.565	0.580	0.613	0.270	0.287	0.314	0.331	0.369

We also evaluate the effectiveness of the proposed HSD when partial structure information is available since the structure knowledge may be very sparse in real world applications. For example, labels associated with image tend to be incomplete and hyperlinks on the webpage are often missing. We progressively increase the number of edges in the structure graph by varying the edge ratio from {0.2, 0.4, 0.6, 0.8, 1} (edges are randomly sampled based on the ratio) and compare HSD with the two multi-view hashing methods[4] using 32 bits. The precision results of top 100 retrieved examples are reported in Table 4.3. It can be seen from the results that our HSD gives the best performance among all methods. We also observe that the precision result of the other two methods drops much faster than HSD when the structure information reduces. Our hypothesis is that when structure graph is very sparse, the common space learned from structure information and data features by the multi-view

[4]LSH, SH and ITQ do not utilize structure information thus are not necessary to be compared here.

hashing methods is not accurate and reliable. Therefore, the hashing codes generated by these methods have lower performance compared to the HSD method, which not only ensures the structure consistency but also preserves the similarity between data examples. However, we would like to point out that if the structure information is inconsistent with the true correlation among data examples, the learned hashing codes could be less effective. To better demonstrate this, we conduct another experiment by constructing an inconsistent graph on $WebKB$. Specifically, we assign large weights on a certain portion of edges that connect uncorrelated nodes (based on the ground-truth labels). Then we apply our approach on this newly constructed graph and obtain the precision result 0.327, which is much lower than the result without using structure knowledge (0.574). Therefore, in practice, it is important to examine the reliability of structure graph in advance.

Table 4.4.
Training and testing time (in second) on both datasets with 32 hashing bits.

	$WebKB$		*NUS-WIDE*	
Methods	training	testing	training	testing
HSD	24.13	0.6x10^{-4}	54.33	0.4x10^{-4}
CMFH	58.82	0.6x10^{-4}	138.64	0.4x10^{-4}
CHMIS	42.59	0.7x10^{-4}	92.16	0.5x10^{-4}
ITQ	10.67	0.6x10^{-4}	20.96	0.4x10^{-4}
SH	13.22	4.2x10^{-4}	27.38	3.3x10^{-4}
LSH	4.75	0.6x10^{-4}	2.62	0.4x10^{-4}

The training cost for learning hashing function and testing cost for encoding each query on both datasets with 32 bits are reported in Table 4.4. We can see from this table that the training cost of HSD is less than a hundred seconds, which is comparable with most of the other hashing methods and it is not slow in practice considering the complexity of training. In contrast to the offline training, the online code generation time is more critical for real-world search applications. The test time for HSD is sufficiently fast especially when compared to the nonlinear hashing method SH. The reason is that it only needs linear projection and binarization to generate the hashing codes for queries.

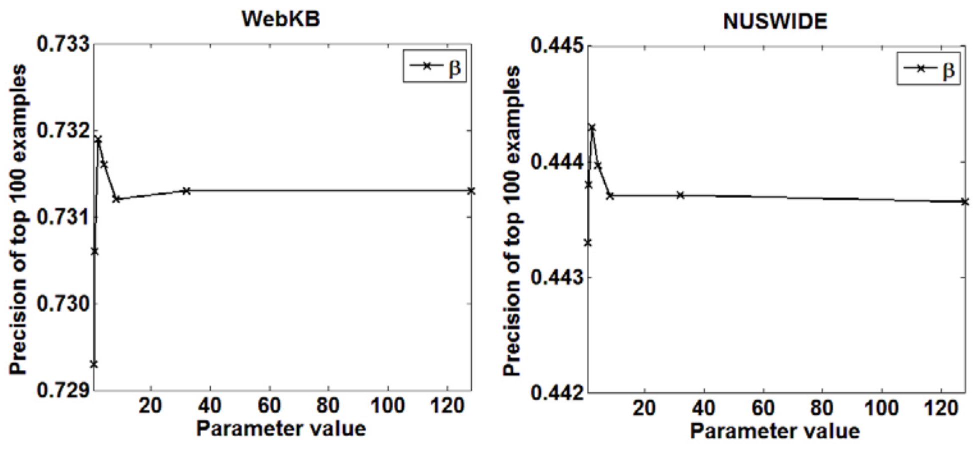


Figure 4.2. Parameter sensitivity results of precision of the top 100 retrieved examples with 32 hashing bits.

To prove the robustness of the proposed method, we conduct parameter sensitivity experiments on both datasets. In each experiment, we tune the trade-off parameter β from the grid {0.5,1,2,4,8,32,128}. We report the precision of top 100 examples with 32 hashing bits in Fig.4.2. It is clear from these experimental results that the performance of HSD is relatively stable with respect to β in a wide range of values. We also observe similar behavior of parameter α. But due to space limit, they are not presented here.

5 RANKING PRESERVING HASHING WITH RELEVANCE JUDGEMENT

5.1 Motivation

Hashing method becomes popular for large scale similarity search due to its storage and computational efficiency. Many machine learning techniques, ranging from unsupervised to supervised, have been proposed to design compact hashing codes. However, most of the existing hashing methods generate binary codes to satisfy pairwise or listwise supervision but do not model the search/ranking accuracy. But in many information retrieval applications, such as similarity search, learning to rank, recommendation, etc., it is more realistic and desirable that the most relevant examples to a query can be presented in front of less relevant ones. In other words, users prefer the retrieval results with better ranking performance. Therefore, it is important to design effective hashing method to incorporate relevance value/judgement information from users in learning more effective hashing codes that could achieve high ranking accuracy.

5.2 Background and Related Work

Traditional similarity search methods are difficult to be used directly on a large dataset since computing the similarity using the original features (usually in high dimensional space) exhaustively between the query example and every candidate example is impractical for large applications. Recently, hashing methods [53,54,56,95,100–103] have been proposed for fast similarity search in many large scale problems including document retrieval [20], object recognition [104], image matching [14], etc. Existing hashing methods can be divided into two groups: unsupervised and semi-supervised/supervised hashing methods.

Unsupervised hashing methods generate hashing codes without the requirement of supervised information (e.g., tags for images or documents). Locality-Sensitive Hashing (LSH) [15] is one of the most popular methods, which simply uses random linear projections to map data examples from a high dimensional Euclidean space to a low-dimensional binary space. The work in [23] extended LSH by exploiting kernel similarity for better retrieval efficacy. The Principle Component Analysis (PCA) Hashing [17] method utilize the coefficients from the top k principal components to represent each example, and the coefficients are further binarized using the median value. A Restricted Boltzman Machine (RBM) [69, 70] is used in [8] to generate compact binary hashing codes. Recently, Spectral Hashing (SH) [19] is proposed to design compact binary codes with balanced and uncorrelated constraints. A graph-based hashing method has been proposed in work [12] to automatically discover the neighborhood structure inherent in the data to learn appropriate compact codes. In work [13], a hyperplane hashing method is proposed for efficient active learning, which can find nearest points to a query hyperplane in sublinear time. Isotropic Hashing (IsoHash) [56] tries to learn an orthogonal matrix to make the data variance as equal as possible along each projection dimension (i.e., hashing bit). Most recently, the work in [48] proposes a weighted Hamming distance ranking algorithm to rank the binary codes by assigning different bit-level weights to different hash bits. A bit selection method [72] is proposed to select the most informative hashing bits from a pool of candidate bits generated from different hashing methods.

Semi-supervised or supervised hashing methods utilize some supervised information such as semantic labels for generating effective hashing codes. Iterative Quantization (ITQ) method has been proposed in [36, 51] that treats the content features and tags as two different views, and the hashing codes are then learned by extracting a common space from these two views. This method has been extended to multi-view hashing [35]. A semi-supervised hashing (SSH) method is proposed in [29] which utilizes pairwise knowledge between data examples besides their content features for learning more effective hashing codes. A kernelized supervised hashing

(KSH) framework proposed in [30] imposes pairwise relationship between data examples to obtain good hashing codes. Complementary Hashing (CH) [9] uses pairwise information to learn multiple complementary hash tables in a boosting manner. More recently, the work in [20] proposes a semantic Hashing method which combines tag information with topic modeling by extracting topics from texts for document retrieval. A ranking-based supervised hashing (RSH) [31] method is proposed to leverage listwise ranking information to preserve the ranking order.

Although existing hashing methods have achieved promising results, very limited work explores the ranking accuracy, which is important for evaluating the quality of hashing codes in real world applications. Consider the following scenario: given a query example x_q and three relevant/similar data examples x_1, x_2, x_3 but with different relevance values as $r_1 > r_2 > r_3$ to the query. Most existing hashing methods only model the relevance of a data example to a query in a binary way, i.e., each example is either relevant to the query or irrelevant. In other words, these methods treat x_1, x_2 and x_3 as relevant examples to x_q with no difference. But in practice it will be more desirable if x_1 could be presented before x_2 and x_3 since it is more relevant to x_q than the other two. Some ranking based hashing methods [31, 47, 48] have been recently proposed to improve the hashing code performance by modeling the ranking order with respect to relevance values. However, these methods do not differentiate the situations where $(r_1, r_2, r_3) = (3, 2, 1)$ and $(r_1, r_2, r_3) = (10, 2, 1)$ due to their identical ranking orders, i.e., $r_1 > r_2 > r_3$. But ideally, the Hamming distance between the learned hashing codes of x_1 and x_q should be smaller in the later situation than in the former one since the relevance value of x_1 to x_q is much larger in the later situation (10 versus 3). Therefore, these methods may fail to preserve the specific relevance values in the learned hashing codes, while the relevance values are important in evaluating the search accuracy.

This dissertation proposes a novel Ranking Preserving Hashing (RPH) approach that directly optimizes the popular ranking accuracy measure, Normalized Discounted Cumulative Gain (NDCG), to learn effective ranking preserving hashing codes that not only preserves the ranking order but also models the relevance values of data examples to the queries in the training data. The main difficulty in direct optimization of NDCG is that it depends on the rankings of data examples rather than their hashing codes, which forms a non-convex non-smooth objective. We then address this challenge by optimizing the expectation of NDCG measure calculated based on a linear hashing function to convert the problem into a smooth and differentiable optimization problem. A gradient descent method is applied to solve this relaxed problem. We conduct an extensive set of experiments on two large scale datasets of both images and texts to demonstrate the superior search accuracy of the proposed approach over several state-of-the-art hashing methods.

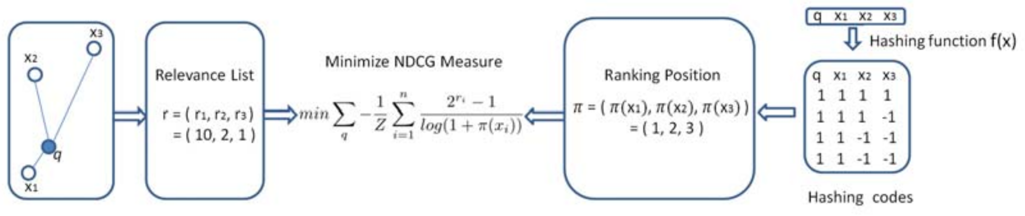


Figure 5.1. An overview of the proposed RPH approach.

5.3 Algorithm

5.3.1 Approach Overview

The proposed Ranking Preserving Hashing (RPH) approach via optimizing NDCG measure mainly contains three ingredients as shown in Figure 5.1: (1) Ground-truth relevance list to a query, which is constructed from the training data (the left part in Fig.5.1). (2) Ranking positions of data examples to a query, which are computed based on the hashing codes (the right part in Fig.5.1). (3) NDCG value, which measures the consistency between the ground-truth relevance list and the calculated ranking positions (the middle part in Fig.5.1). In other words, the more the hashing codes agree with the relevance list, the higher the NDCG value will be. Then the ranking preserving hashing codes are learned by optimizing the NDCG measure on the training data.

5.3.2 Problem Statement

We first introduce the problem of RPH. Assume there are n data examples in the dataset, denoted as: $\boldsymbol{X} = \{x_1, x_2, \ldots, x_n\} \in \mathbb{R}^{d \times n}$, where d is the dimensionality of the features. In addition, there is a query set $\boldsymbol{Q} = \{q_1, q_2, \ldots, q_m\}$ and for each query example q_j, we have a relevance list of n_j data examples from $\boldsymbol{X}$, which can be written as:

$$r(q_j, \boldsymbol{X}) = (r_1^j, r_2^j, \ldots, r_{n_j}^j) \tag{5.1}$$

where each element r_i^j represents the relevance of data example x_i^j to the query q_j. If $r_u^j > r_v^j$, it indicates that data example x_u^j is more relevant or more similar to q_j than x_v^j and x_u^j should rank higher than x_v^j. The goal is to obtain a linear hashing function $f : \mathbb{R}^d \to \{-1, 1\}^B$, which maps each data example x_i to its binary hashing code c_i (B is the number of hashing bits) to maximize the search/ranking accuracy. The linear hashing function is defined as:

$$c_i = f(x_i) = sgn(\boldsymbol{W} x_i) \tag{5.2}$$

where $\boldsymbol{W} \in \mathbb{R}^{B \times d}$ is the coefficient matrix representing the hashing function and sgn is the sign function. $c_i \in \{-1, 1\}^B$ is the binary hashing code of x_i.

Note that the ground-truth relevance list can be easily obtained if a relevance measure between data examples is predefined, e.g., l_2 distance in Euclidean space. On the other hand, if given the semantic label/tag information, it is also fairly straightforward to convert semantic labels to relevance values through counting the number of shared labels between the query and the data example.

5.3.3 Problem Formulation

Hashing methods are popularly used for large scale similarity search. As aforementioned, most of existing hashing methods only focus on retrieving all relevant or similar data examples to a given query without exploring the ranking accuracy. However, in many real world applications, it is desirable and important to present a more relevant example to a query in front of a less relevant one. Different from existing hashing method, in this dissertation, we propose to learn ranking preserving hashing codes that not only retrieve all possible relevant examples but at the same time preserve their rankings based on their relevance values to the query.

Given the binary hashing codes, the ranking positions of data examples to a query q are determined by the Hamming distances between their hashing codes and the query code. Specifically, if a data example is similar or relevant to a query, then their Hamming distance should be small. In other words, the higher the rank of a data example to a query, the smaller the Hamming distance between the hashing codes is. The Hamming distance between two binary hashing codes is given by the number of bits that are different between them and can be calculated as:

$$Ham(c_q, c_i) = \frac{1}{4}\|c_q - c_i\|^2 = \frac{1}{2}(B - c_q^T c_i) \tag{5.3}$$

Then the ranking position $\pi(x_i)$ can be calculated as:

$$\begin{aligned}\pi(x_i) &= 1 + \sum_{k=1}^{n} I\left(Ham(c_q, c_i) > Ham(c_q, c_k)\right) \\ &= 1 + \sum_{k=1}^{n} I\left(c_q^T(c_k - c_i) > 0\right)\end{aligned} \tag{5.4}$$

where $I(s)$ is the indicator function that outputs 1 when statement s is true and 0 otherwise. Intuitively, the ranking position of a data example to a query is equivalent to 1 plus the number of data examples whose hashing codes are closer to the query code.

In order to achieve high ranking quality hashing codes, we want the ranking positions calculated in the Hamming space in Eqn.5.4 to be consistent with the ground-truth relevance list in Eqn.5.1. Then a natural question to ask is how to measure the ranking consistency? In this dissertation, we use a well-known measure, Normalized Discounted Cumulative Gain (NDCG) [105, 106] which is widely applied in many information retrieval and machine learning applications, to evaluate the ranking consistency as:

$$NDCG = \frac{1}{Z}\sum_{i=1}^{n}\frac{2^{r_{\pi^{-1}(i)}}-1}{log(1+i)} = \frac{1}{Z}\sum_{i=1}^{n}\frac{2^{r_i}-1}{log(1+\pi(x_i))} \tag{5.5}$$

where Z is the normalization factor so that the maximum value of NDCG is 1, which can be calculated by ranking the examples based on their relevance to the query. $\pi(x_i)$ is the ranking position of x_i to the query based on the Hamming distance of their hashing codes and $\pi^{-1}(i)$ denotes the data example at i-th ranking position. r_i is the corresponding relevance value. $\frac{1}{log(1+i)}$ can be viewed as the weight of the i-th rank data example, which indicates that NDCG emphasizes the importance of the higher ranked data examples than those examples with lower ranks. Therefore, NDCG is usually truncated at a particular rank level (e.g., top K retrieved examples) instead of all n examples. From the above definition of NDCG, it can be seen that the larger the NDCG value is, the more the hashing codes agree with the relevance list, and the maximal NDCG value is obtained when the ranking positions of data examples are completely consistent with their relevance values to the query. By optimizing the NDCG measure, the learned hashing function not only preserves the ranking order of the data examples but also ensures that the hashing codes are consistent with the relevance values in the training data. Then the entire objective is to minimize the negative summation of NDCG values on all training queries:

$$J(\boldsymbol{W}) = -\sum_{j=1}^{m}\frac{1}{Z_j}\sum_{i=1}^{n_j}\frac{2^{r_i^j}-1}{log(1+\pi_j(x_i^j))} \tag{5.6}$$

Directly minimizing the objective function in Eqn.5.6 is intractable since it depends on the ranking positions of data examples (Eqn.5.4), resulting in a non-convex non-smooth optimization problem. We then address this challenge by using the expectation of ranking position $\hat{\pi}_j(x_i^j)$ instead of $\pi_j(x_i^j)$ as:

$$\begin{aligned}\hat{\pi}_j(x_i^j) &= 1 + E\left[\sum_{k=1}^{n} I(c_{q_j}^T(c_k - c_i) > 0)\right] \\ &= 1 + \sum_{k=1}^{n} Pr\left(c_{q_j}^T(c_k - c_i) > 0\right)\end{aligned} \tag{5.7}$$

where $Pr(c_{q_j}^T(c_k - c_i) > 0)$ means the probability that the ranking position of data example x_k is higher than the position of x_i to query q_j and we use a logistic function to model this probability as:

$$\begin{aligned}Pr\left(c_{q_j}^T(c_k - c_i) > 0\right) &= \frac{1}{1 + exp(-c_{q_j}^T(c_k - c_i))} \\ = \frac{1}{1 + exp(-sgn(\boldsymbol{W}q_j)^T(sgn(\boldsymbol{W}x_k) - sgn(\boldsymbol{W}x_i)))}\end{aligned} \tag{5.8}$$

The motivation of the derivation in Eqn.5.7 and Eqn.5.8 is that we approximate the intractable optimization for NDCG with a tractable probabilistic framework. Firstly, the ranking position of each data example can be calculated exactly based on Eqn.5.4. However, due to the intractability, we model the problem in a probabilistic framework by computing the expectation of the ranking position. The using of expectation to represent the true ranking position is widely adopted in learning to rank approaches due to its good probability approximation and computational tractability. Secondly, the using of logistic function in Eqn.5.8 to model the probability is based on the intuition that a data example should be ranked higher if its hashing code is closer to the query. There are also other alternatives to model the probability. Due to the popularity of logistic function used in learning to rank, we adopt it in our formulation.

The above probability function is still non-differentiable with respect to $\boldsymbol{W}$ due to the embedded sign function. Therefore, as suggested in [31, 94], we drop off the sign function and use the signed magnitude in the probability function as:

$$Pr\left(c_{q_j}^T(c_k - c_i) > 0\right) = \frac{1}{1 + exp(-q_j^T\boldsymbol{W}^T\boldsymbol{W}(x_k - x_i))} \tag{5.9}$$

By substituting the expected ranking position into the NDCG measure, the final objective in Eqn.5.6 can be rewritten as:

$$\begin{aligned} \min \;\; J(\boldsymbol{W}) &= -\sum_{j=1}^{m} \frac{1}{Z_j} \sum_{i=1}^{n_j} \frac{2^{r_i^j}-1}{log(1+\hat{\pi}_j(x_i^j))} \\ s.t. \qquad & \boldsymbol{W}\boldsymbol{W}^T = \boldsymbol{I} \end{aligned} \tag{5.10}$$

where $\boldsymbol{W}\boldsymbol{W}^T = \boldsymbol{I}$ is the orthogonality constraint which ensures the learned hashing codes to be uncorrelated with each other and hold least redundant information.

5.3.4 Optimization

We first convert the hard constraint into a soft penalty term by adding a regularizer to the objective. The reason is that most of the variance is contained in a few top projections for many real world datasets. The orthogonality constraint forces hashing methods to choose those directions with very low variance progressively, which may substantially reduce the quality of hashing codes. This issue is also pointed out in [29, 30]. Therefore, instead of adding hard orthogonality constraint, we impose a soft orthogonality/penalty term as:

$$J(\boldsymbol{W}) = -\sum_{j=1}^{m} \frac{1}{Z_j} \sum_{i=1}^{n_j} \frac{2^{r_i^j}-1}{log(1+\hat{\pi}_j(x_i^j))} + \alpha \|\boldsymbol{W}\boldsymbol{W}^T - \boldsymbol{I}\|_F^2 \tag{5.11}$$

where α is a trade-off parameter to balance the weights between the two terms. Although the objective in Eqn.5.11 is still non-convex, it is smooth and differentiable which enables gradient descent methods to be applied for efficient optimization. The gradients of the two terms with respect to $\boldsymbol{W}$ are given below:

$$\begin{aligned} \frac{d\hat{\pi}_j(x_i^j)}{d\boldsymbol{W}} = &\sum_{k=1}^{n_j} exp(-q_j^T \boldsymbol{W}^T \boldsymbol{W}(x_k - x_i)) \\ &\frac{\boldsymbol{W}\left((x_k - x_i)q_j^T + q_j(x_k - x_i)^T\right)}{(1 + exp(-q_j^T \boldsymbol{W}^T \boldsymbol{W}(x_k - x_i)))^2} \end{aligned} \tag{5.12}$$

$$\frac{d\|\boldsymbol{W}\boldsymbol{W}^T - \boldsymbol{I}\|_F^2}{d\boldsymbol{W}} = 4\boldsymbol{W}^T(\boldsymbol{W}\boldsymbol{W}^T - \boldsymbol{I}) \tag{5.13}$$

Then the gradient of $\frac{dJ(\boldsymbol{W})}{d\boldsymbol{W}}$ can be computed by combining the above two gradients with some additional mathematical calculation. With this obtained gradient, L-BFGS quasi-Newton method [79] is applied to solve the optimization problem. Note that if the number of training queries is large, a stochastic optimization method [107] can be applied to Eqn.5.11 by only updating the solution with respect to a randomly selected subset of queries during each iteration as follows: More precisely, during each iteration, we randomly pick a set of queries, S, of small size from the m queries and a random set of examples M in the relevance list. And we calculate the gradients on these selected queries and examples as:

$$\frac{dJ}{d\boldsymbol{W}}_{|\{S,M\}} = \sum_{j \in S} \frac{1}{Z_j} \sum_{i \in M} \frac{2^{r_i^j} - 1}{(log(1 + \hat{\pi}_j(x_i^j)))^2 (1 + \hat{\pi}_j(x_i^j))} \frac{d\hat{\pi}_j(x_i^j)}{d\boldsymbol{W}} \tag{5.14}$$

and we use this gradient in each iteration for updating the new solution. The full RPH approach is summarized in Table 5.1.

Table 5.1.
Ranking Preserving Hashing (RPH)

```
Input: Training examples $\boldsymbol{X}$, query examples $\boldsymbol{Q}$ and parameters $\alpha$.
Output: Hashing function $\boldsymbol{W}$ and hashing codes $\boldsymbol{C}$.
---
Compute the relevance vector $r_i^j$ in Eqn.5.1.
Initialize $\boldsymbol{W}$.
Repeat  Gradient Descent
    Compute the gradient in Eqn.5.12.
    Compute the gradient in Eqn.5.13.
    Update $\boldsymbol{W}$ by optimizing the objective function.
Until the solution converges
Compute the hashing codes $\boldsymbol{C}$ using Eqn.5.2.
```

5.3.5 Discussion

The idea of modeling the NDCG measure to maximize the search/ranking accuracy is also utilized in learning to rank [108, 109]. However, these learning to rank methods are not based on binary hashing codes, but on learning effective document permutation. Unlike in our formulation, the NDCG measure modeled in learning to rank methods does not involve linear-projection based hashing function, on which the ranking position is determined. Moreover, we need to find the expected ranking position of each data example according to the Hamming distance between the hashing codes, which is very different to learning to rank methods.

The learning algorithm of RPH for deriving the optimal hashing function is fairly fast. During each iteration of the gradient descent method, we need to compute the gradients in Eqns.5.12 and 5.13, which involves some matrix multiplications. The complexity for calculating the gradient in Eqn.5.12 is bounded by $\mathrm{O}(mn_j dB)$ since both $\boldsymbol{W}(x_k - x_i)$ and $\boldsymbol{W}(x_k - x_i)q_j^T$ requires $\mathrm{O}(dB)$. The complexity for calculating the gradient in Eqn.5.13 is simply $\mathrm{O}(d^2B)$ which only involves $\boldsymbol{W}\boldsymbol{W}^T$. Therefore, the total complexity of each iteration of the gradient descent method is $\mathrm{O}(m\hat{n}dB + d^2B)$ and the learning algorithm is fairly scalable since its time complexity is linear in the number of training queries m and the average number of data examples $\hat{n}$ associated with each query. Note that $\hat{n}$ is much smaller than the total number of data examples. We will provide more details in the experiments.

5.4 Experiments

5.4.1 Datasets and Implementation

We evaluate proposed research on two image benchmarks: *NUSWIDE* and *Flickr*1*m*, which have been widely used in the evaluation of hashing methods [31, 36, 94]. *NUSWIDE*[1] is created by NUS lab for evaluating image retrieval

[1]http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm

techniques. It contains $270k$ images associated with about $5k$ different tags. We use a subset of $110k$ image examples with the most common $1k$ tags in our experiment. $Flickr1m$[2] is collected from Flicker images for image annotation and retrieval tasks. This dataset contains 1 million image examples associated with more than $7k$ unique semantic tags. A subset of $250k$ image examples with the most common $1k$ tags is used in our experiment. 512-dimensional GIST descriptors [87] are used as image features. Since both datasets are associated with multiple semantic labels/tags, the ground-truth relevance values can be naturally derived based on the number of shared semantic labels between data examples.

We implement our algorithm using Matlab on a PC with Intel Duo Core i5-2400 CPU 3.1GHz and 8GB RAM. The parameter α is tuned by cross validation through the grid $\{0.01, 0.1, 1, 10, 100\}$ and we will discuss more details on how it affects the performance of our approach later. For each experiment, we randomly choose $1k$ examples as testing queries. Within the remaining data examples, we randomly sample 500 training queries and for each query, we randomly sample 1000 data examples to construct the ground-truth relevance list. We will discuss the performance with different number of training queries later in our experiments. Finally, we repeat each experiment 10 times and report the result based on the average over the 10 runs.

5.4.2 Comparison Methods

The proposed RPH approach is compared with four different hashing methods, including Spectral Hashing (SH) [19], Semi-Supervised Hashing (SSH) [29], Kernel Supervised Hashing (KSH) [30] and Ranking-based Supervised Hashing (RSH) [31]. SH is an unsupervised method and does not use any label information. We use the standard settings in [19] in our experiments. For SSH and KSH, we randomly sample $2k$ data examples and use their ground-truth labels to generate pairwise similarity

[2]http://press.liacs.nl/mirflickr/

matrix as part of the training data. Gaussian RBF kernel is used in KSH. To get a fair comparison, for RSH, we randomly sample 500 query examples and 1000 data examples to compute the ground-truth ranking lists.

Table 5.2.
Results of NDCG@K using *Hamming Ranking* on both datasets, with 64 hashing bits.

	NUSWIDE			*Flickr1m*		
NDCG@K	5	10	20	5	10	20
RPH	**0.257**	**0.249**	**0.234**	**0.313**	**0.298**	**0.283**
RSH	0.242	0.238	0.226	0.288	0.271	0.259
KSH	0.223	0.217	0.198	0.265	0.252	0.237
SSH	0.216	0.209	0.195	0.251	0.242	0.230
SH	0.193	0.185	0.172	0.250	0.234	0.221

5.4.3 Evaluation Metrics

To conduct fair evaluation, we follow two criteria which are commonly used in the literature: *Hamming Ranking* and *Hash Lookup*. *Hamming Ranking* ranks all the examples in the dataset according to their Hamming distance from the query and the top K examples are returned as the desired neighbors. We use NDCG@K to evaluate the ranking quality of the top K retrieved examples for each individual query and calculate the average NDCG values over all test queries. For those data examples falling in the same Hamming distance to query examples, the expectation of NDCG is computed.

Hash Lookup returns all the examples within a certain Hamming radius r of the query. Since hash lookup does not provide ranking for returned points with equal

Hamming distance to the queries, we use average cumulative gain (ACG) to measure the quality of these returned examples, which is calculated as:

$$ACG_r = \frac{1}{|N_r|} \sum_{x_i \in N_r} r_i \tag{5.15}$$

where N_r is the set of the retrieved data examples within a Hamming radius r and r_i is the relevance value of a retrieved data example x_i. Here the scale value of $|N_r|$ is the total number of retrieved data examples. Hence, ACG essentially measures the average precision weighted by the relevance of each retrieved example. In summary, both metrics emphasize the quality of ranking, which is important in practical. A hamming radius of $r = 2$ is used to retrieve the neighbors in the experiments.

5.4.4 Results and Discussion

Table 5.3.
Precision of the top 100 retrieved data examples on both datasets with different hashing bits.

	NUSWIDE			*Flickr1m*		
Methods	16 bits	64 bits	256 bits	16 bits	64 bits	256 bits
RPH	0.298	**0.327**	**0.346**	**0.370**	**0.499**	**0.561**
RSH	0.281	0.314	0.343	0.340	0.474	0.537
KSH	**0.306**	0.316	0.341	0.359	0.494	0.556
SSH	0.246	0.295	0.298	0.345	0.439	0.486
SH	0.251	0.282	0.304	0.331	0.392	0.447

We first report the results of NDCG@5, NDCG@10 and NDCG@20 of different hashing methods using *Hamming Ranking* on two datasets with 64 hashing bits in Table 5.2. From these comparison results, it can be seen that RPH gives the overall best performance among all five hashing methods on both datasets. For example, the performance of our method boosts about 4.6% on *NUSWIDE* dataset, with 9.9%

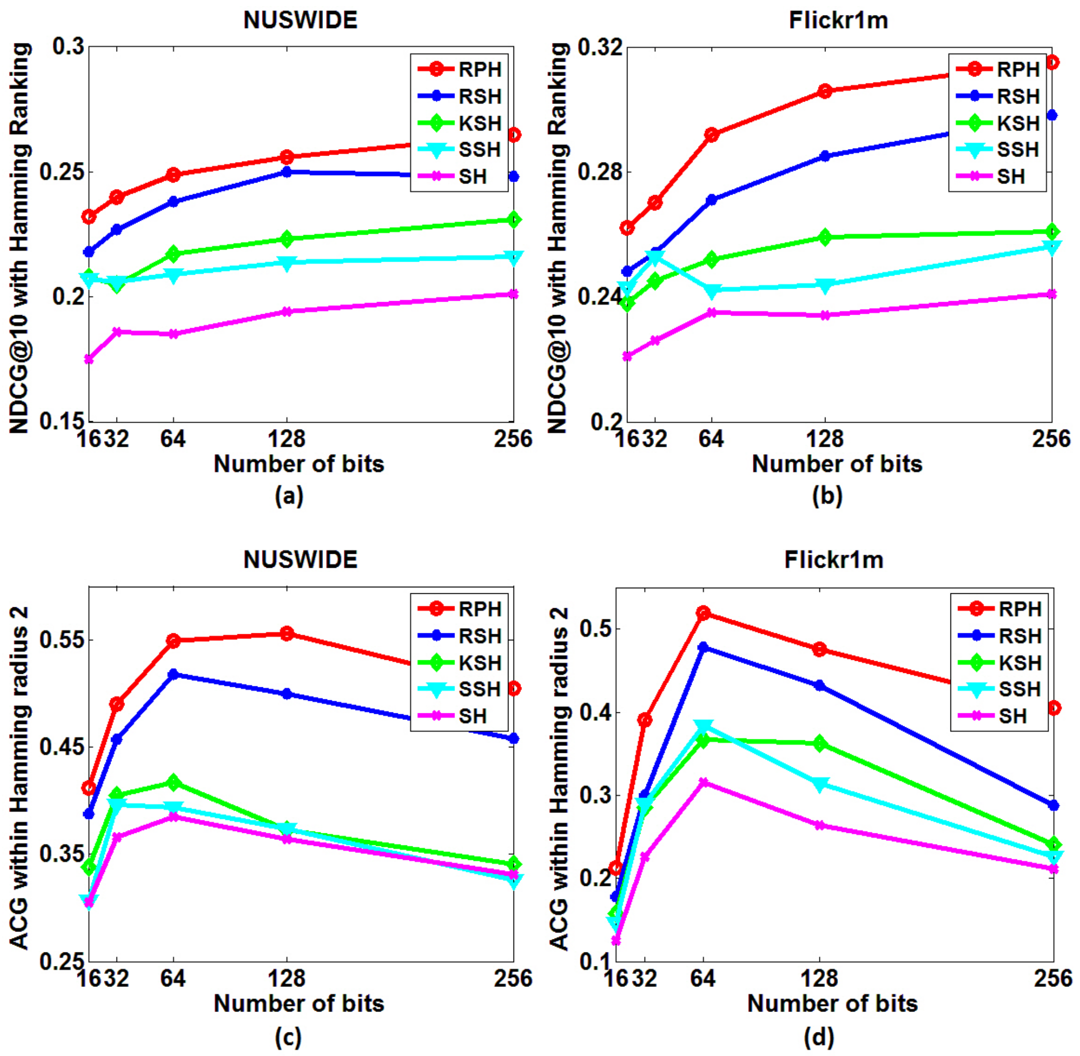


Figure 5.2. Performance evaluation on both datasets with different number of hashing bits. (a)-(b): NDCG@10 using *Hamming Ranking*. (c)-(d): ACG with Hamming radius 2 using *Hash Lookup*.

improvement on *Flickr1m* dataset compared to RSH under NDCG@10 measure. We can see from Table 5.2 that SH does not perform well in all cases. This is because SH is an unsupervised hashing method which does not utilize any supervised information into learning hashing codes. For methods SSH and KSH, they both achieve better results than SH since these methods incorporate some pairwise knowledge between data examples in addition to the content features for learning effective hashing codes. KSH obtains slightly larger NDCG values than SSH due to the exploitation of kernel similarity. However, the ranking order is not preserved in the learned hashing codes of these two methods and thus, the ranking-based supervised hashing method RSH which models the listwise ranking information can generate more accurate hashing codes with larger NDCG values than SSH and KSH. On the other hand, our RPH method substantially outperforms RSH since it directly optimizes the NDCG measure to learn high quality hashing codes that not only preserve the ranking order but also preserve the relevance values of data examples to the query in the training data. Therefore, the search/ranking accuracy can be maximized which is coincides with our expectation.

The second set of experiments evaluate the performance of different hashing methods by varying the number of hashing bits in the range of $\{16, 32, 64, 128, 256\}$. The results of NDCG@10 using *Hamming Ranking* on both datasets are reported in Fig.5.2(a)-(b), with the ACG results of Hamming radius 2 using *Hash Lookup* shown in Fig.5.2(c)-(d). Not surprisingly, from Fig.5.2(a)-(b) we can see that the performance of different methods improves when the number of hashing bits increases from 16 to 256 and our RPH method outperforms the other compared hashing methods which is consistent with the results in Table 5.2. However, we can also observe from Fig.5.2(c)-(d) that the ACG result of most compared methods decreases when the number of hashing bits increases after 64. The reason is that when using longer hashing bits, the Hamming space becomes increasingly sparse and very few data examples fall within the Hamming ball of radius 2, resulting in many queries with empty returns (we count the ACG as zero in this case). Similar behavior is also observed in [31] and [94]. In this situation, the NDCG results from Fig.5.2(a)-(b) provide better performance measurement, while the ACG results of RPH are still consistently better than other baselines.

In the third set of experiments, we examine the performance of RPH under non-ranking measure, i.e., precision of top 100 retrieved data examples using *Hamming*

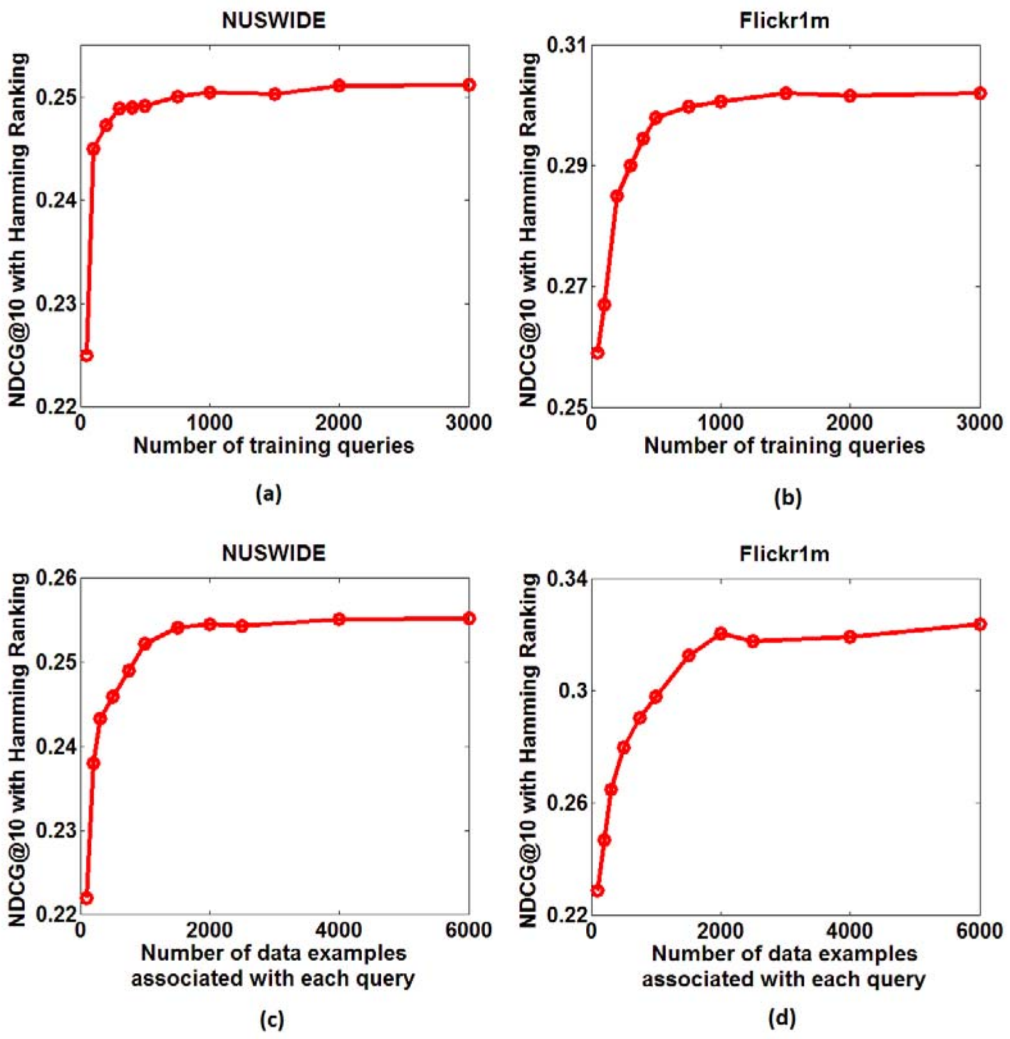


Figure 5.3. (a)-(b): NDCG@10 with different number of training queries using 64 hashing bits on both datasets. (c)-(d): NDCG@10 with different number of data examples associated with each query using 64 hashing bits on both datasets.

Ranking. This measure is widely used in previous hashing methods, which do not emphasize on the ranking accuracy, to evaluate how the hashing codes perform in finding similar data examples. The precision results of all compared methods with different hashing bits are reported in Table 5.3. It can be seen from Table 5.3 that the RPH method achieves best results among all compared methods in most cases. The reason is that although RPH does not directly preserve the pairwise similarity between data examples as in KSH, SSH and SH, the similarities among data examples are implicitly preserved in the modeling of relevance and ranking consistency. In other words, the strategy of ensuring ranking order consistency enables a more similar example to the query to rank higher than a less similar example and to be found in top retrieval results. We also observe that the precisions of all hashing methods increase when the number of hashing bits increases, which is consistent with the results in Fig.5.2(a)-(b).

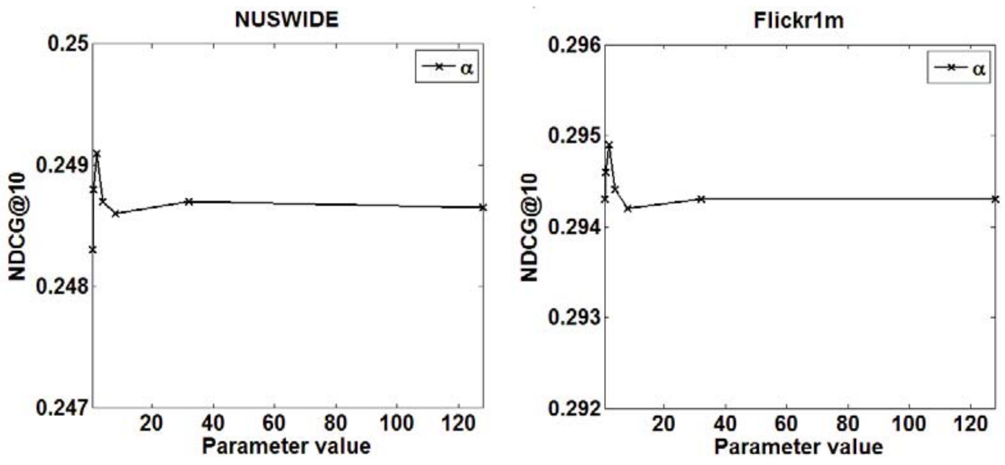


Figure 5.4. Parameter sensitivity results of NDCG@10 on both datasets with 64 hashing bits.

We also evaluate the ranking performance of RPH by varying the number of training queries m and the number of data examples associated with each query respectively. We report the NDCG@10 results by fixing the number of data examples associated with each training query to 1000, and vary the number of training queries from 50 to 3000 in Fig.5.3(a)-(b). Similarly, the NDCG@10 results of fixing the number of training queries to 500, and varying the number of data examples associated with each training query from 100 to 6000 are shown in Fig.5.3(c)-(d).

Not surprisingly, we can observe that the NDCG value increases with the increasing number of training queries and data examples. We also found that, take $Flickr1m$ for example, the performance of RPH does not increase much after around 700 training queries (Fig.5.3(b)) and 1800 data examples (Fig.5.3(d)). Our hypothesis is that we have gained sufficient ranking information to learn a good hashing function using these many training queries and data examples. However, the training cost increases almost linearly with the size of training data. Therefore, we choose 500 queries and 1000 data examples associated with each query consistently in our method to form the training data, which can obtain good performance with reasonable training cost.

To prove the robustness of the proposed method, we conduct parameter sensitivity experiments on both datasets. In each experiment, we tune the trade-off parameter α from the grid $\{0.5, 1, 2, 4, 8, 32, 128\}$. We report the results of NDCG@10 with 64 hashing bits in Fig.5.4. It is clear from these experimental results that the performance of RPH is relatively stable with respect to α in a wide range of values. The results also prove that using soft penalty with an appropriate weight parameter is better than enforcing the hard orthogonality constraint (corresponds to infinite α).

6 ACTIVE LEARNING VIA JOINT DATA EXAMPLE AND LABEL SELECTION

6.1 Motivation

Hashing methods have been widely used for large scale similarity search. Recent research has shown that hashing quality can be dramatically improved by incorporating supervised information, e.g. semantic tags/labels, into hashing function learning. However, most existing supervised hashing methods can be regarded as passive methods, which assume that the labeled data are provided in advance. But in many real world applications, such supervised information may not be sufficient or available and it is often expensive to acquire for a large dataset. Therefore, it is important to design effective methods to actively identify only a small set of the most informative data examples for users to label. On the other side, the labeling cost also depends on the total number of tags that the users label to the selected data examples. In many large scale applications, there are often hundreds or thousands of tags for users to label. Moreover, similar tags usually carry similar semantic meanings. For instance, 'car' and 'automobile' have similar meanings and choosing both of them may not gain substantial new information over just selecting one. Therefore, it is important to design effective method that jointly selects the most informative data examples and tags such that the hashing function can be learned efficiently with only a small number of labeled data, which can greatly reduces the labeling cost.

6.2 Background and Related Work

6.2.1 Introduction

Similarity search is a key problem in many information retrieval applications including image and text retrieval, content reuse detection and collaborative filtering. The purpose of similarity search is to identify similar data examples given a query example. With the explosive growth of the internet, a huge amount of data such as texts, images and video clips have been generated, which indicates that efficient similarity search with large scale data becomes more important. Traditional similarity search methods are difficult to be used directly for large scale data since computing the similarity using the original features (i.e., often in high dimensional space) exhaustively between the query example and every candidate example is impractical for large applications. There are two major challenges for using similarity search in large scale data: storing the large data and retrieving desired data efficiently.

Hashing methods [8, 10, 19, 22] have been proposed for addressing these two challenges and have achieved promising results. These hashing methods design compact binary code in a low-dimensional space for each data example so that similar data examples are mapped to similar binary codes. In the retrieval process, these hashing methods first transform each query example into its corresponding binary code. Then similarity search can be simply conducted by calculating the Hamming distances between the codes of available data examples and the query and selecting data examples within small Hamming distances. In this way, the two major challenges for large scale similarity search can be addressed as: data examples are encoded and highly compressed within a low-dimensional binary space, which can usually be loaded in main memory and stored efficiently. The retrieval process is also very efficient, since the Hamming distance between two codes is simply the number of bits that differ, which can be calculated using bitwise XOR.

Recently, some supervised hashing methods [20, 29, 30] have incorporated labeled data/information, e.g. semantic tags, for learning more effective hashing function than unsupervised hashing methods. It has been shown that hashing quality could be dramatically improved by leveraging supervised information. For example, in text retrieval applications, semantic tags (e.g. documents labeled with the same tag) reflect the semantic relationship between documents and thus can be very important and helpful for learning hashing function. However, most existing supervised hashing methods can be regarded as passive methods, which assume that the labeled data are provided beforehand. But in many real world applications, such supervised information may not be available and it is often expensive to acquire for a large dataset. Therefore, it is important to design effective methods to actively identify only the most informative data examples for users to label. On the other hand, the labeling cost will also depend on the total number of tags that the users label to the selected data examples. In many large scale applications, there are often hundreds or thousands of tags for users to label. Moreover, similar tags usually carry similar semantic meanings. For instance, 'car' and 'automobile' have similar meanings and choosing both of them may not gain substantial new information over just selecting one. Therefore, selecting a small set of most informative tags is also important to lower the efforts of user labeling.

This dissertation proposes a novel active hashing approach, Active Hashing with Joint Data Example and Tag Selection (AH-JDETS), to achieve the goal of learning accurate hashing function with a limited amount of labeling efforts. AH-JDETS actively selects the most informative data examples and tags in a joint manner for hashing function learning. Specifically, it first identifies a set of informative data examples and tags for users to label based on the selection criteria that both the data examples and tags should be most uncertain and dissimilar with each other. Then the supervised information is combined with the unlabeled data to generate an effective hashing function. An iterative procedure is proposed for learning the optimal hashing function and selecting the most informative data examples and tags. Extensive experiments on four different datasets have been conducted to demonstrate that AH-JDETS can achieve good performance with much less labeling cost when compared to state-of-the-art supervised hashing methods, which overcomes the limitations of passive hashing methods. Moreover, the experiments have clearly shown the advantages of the proposed AH-JDETS approach against several other selection methods for obtaining training data.

6.2.2 Related Work

This section reviews the related work in two research areas: hashing function learning and active learning.

Hashing Function Learning

Hashing methods [27, 41, 110–113] are proposed to address the similarity search problem within large scale data. These hashing methods try to encode each data example by using a small fixed number of binary bits while at the same time preserve the similarity between data examples as much as possible. In this way, data examples are transformed from a high-dimensional space into a low-dimensional binary space and therefore, the similarity search can be done very fast by only computing the Hamming distance between binary codes. Existing hashing methods can be divided into two groups: unsupervised and supervised hashing methods.

Among the unsupervised hashing methods, Locality-Sensitive Hashing (LSH) [15] is one of the most popular methods, which simply uses random linear projections to map data examples from a high dimensional Euclidean space to a low-dimensional binary space. This method has been extended to Kernelized and Multi-Kernel Locality-Sensitive Hashing [23] by exploiting kernel similarity for better retrieval efficacy. The Principle Component Analysis (PCA) Hashing [17] method represents each example by coefficients from the top k principal components of the training set, and the coefficients are further binarized using the median value. A Restricted Boltzman Machine (RBM) [69, 70] is used in [8] to generate compact binary hashing codes. Recently, Spectral Hashing (SH) [19] is proposed to design compact binary codes with balanced and uncorrelated constraints. Self-Taught Hashing (STH) [22] combines an unsupervised learning step with a supervised learning step to learn hashing codes. More recently, the work in [10] proposes a Composite Hashing with Multiple Information Sources (CHMIS) method to integrate information from different sources. In work [13], a hyperplane hashing method is proposed for efficient active learning, which can find nearest points to a query hyperplane in sublinear time.

For the supervised hashing methods, a Canonical Correlation Analysis with Iterative Quantization (CCA-ITQ) method has been proposed in [36] which treats the content features and tags as two different views. The hashing codes are then learned by extracting a common space from these two views. Recently, several pairwise hashing methods have been proposed. The semi-supervised hashing (SSH) method in [29] utilizes pairwise knowledge between data examples besides their content features for learning more effective hashing codes. A kernelized supervised hashing (KSH) framework proposed in [30] imposes the pairwise relationship between data examples to obtain good hashing codes. More recently, a ranking-based supervised hashing (RSH) [31] method is proposed to leverage listwise ranking information to improve the search accuracy. Most recently, the work in [20] proposes a Semantic Hashing method which combines Tag information with Topic Modeling (SHTTM) by extracting topics from texts and exploiting the correlation between tags and hashing codes for document retrieval. It has been shown that supervised hashing methods achieve better performance than unsupervised methods. However, as aforementioned, most existing supervised hashing methods can be regarded as passive methods which assume the labeled data are provided beforehand but in practice, such supervised information may not be available or can be expensive to obtain. Therefore, it is important to design effective methods to actively identify only the most informative data for users to label for generating accurate hashing codes with low cost.

Active Learning

The purpose of active learning [114–116] is to select data examples from an unlabeled pool which will be very beneficial in training the model, thereby reducing the labeling cost since noninformative instances are not selected. Many strategies have been proposed to measure the informativeness of unlabeled data examples. Uncertainty sampling [117] selects the data examples whose predicted labels are the most uncertain. A batch mode active learning method is proposed in [115] that applies the Fisher information matrix to select a number of informative examples simultaneously based on the criteria that the selected set of unlabeled examples should result in the largest reduction in the Fisher information. The work in [114] proposes a discriminative batch mode active learning strategy that exploits information from an unlabeled set to learn a good classifier directly, which obtains high likelihood on the labeled training instances and low uncertainty on labels of the unlabeled instances. A comprehensive survey of active learning can be found in [118].

Recently, active learning has been extended to various tasks including image classification [115], learning to rank [119, 120], query selection [121, 122] and collaborative filtering [123,124]. For example, the work in [119] addresses active rank learning based on expected hinge rank loss minimization. Inspired by the expected loss reduction strategy, [120] recently introduces an expected loss optimization framework for ranking, where the selection of query and documents is integrated in a principled manner. The work in [122] generalizes the empirical risk minimization principle to active learning which identifies the most uncertain and representative queries by preserving the source distribution as much as possible. A Bayesian selection approach is proposed in [124] for collaborative filtering, which identifies the most informative items such that the updated user model will be close to the expected user model.

The only work we found using active learning in hashing is [50], which directly chooses the most uncertain data examples based on the hashing function. A batch

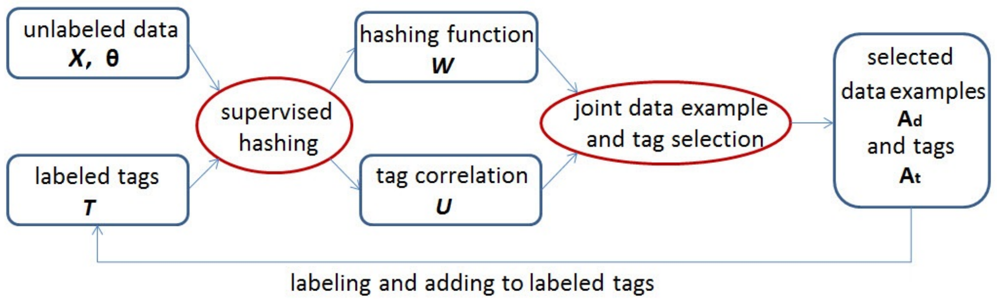


Figure 6.1. An overview of the proposed AH-JDETS approach.

mode algorithm is also proposed in this work to speed up their active selection. However, the method in [50] only considers identifying the most informative data examples and tries to label all possible tags to these selected examples, which requires a great amount of labeling efforts for those datasets associated with a huge number of tags. Therefore, in this dissertation, we develop a novel active hashing approach that jointly selects the most informative data examples and tags such that the hashing function can be learned efficiently with only a small number of labeled data, which greatly reduces the labeling cost.

6.3 Algorithm

6.3.1 Approach Overview

The proposed AH-JDETS approach mainly consists of two components as shown in Figure 6.1: (1) Supervised hashing, which incorporates the labeled information into hashing function learning. In this dissertation, we modify the recent supervised hashing method in [20] to learn effective hashing function and present the details in next section. (2) Joint data example and tag selection, which actively selects a set of most informative data examples and tags for users to label. These newly labeled data are added to existing labeled information for learning a more accurate and effective hashing function.

6.3.2 Supervised Hashing

Our active hashing method is related to a recent supervised hashing method, Sematic Hashing using Tags and Topic Modeling (SHTTM) [20], which utilizes tags and topic modeling together to learn effective hashing function. One main advantage of this method is that it not only learns high quality hashing codes but extracts a set of tag correlation variables, which reflect the correlation between tags and learned hashing codes. Therefore, the relationship among different tags is also represented in the tag correlation variables (more details will be given later).

We first introduce some notation. Assume there are n training examples total, denoted as: $\boldsymbol{X} = \{x_1, x_2, \ldots, x_n\} \in \boldsymbol{R}^{m \times n}$, where m is the dimensionality of the content feature. Denote the labeled tags as: $\boldsymbol{T} \in \{0, 1\}^{l \times n}$, where l is the total number of possible tags associated with each data example. A label 1 in $\boldsymbol{T}$ means an example is associated with a certain tag, while a label 0 means the example is not associated with that tag. The goal is to obtain optimal binary hashing codes $\boldsymbol{Y} = \{y_1, y_2, \ldots, y_n\} \in \{-1, 1\}^{k \times n}$ for the training examples, and a hashing function $f : \boldsymbol{R}^m \to \{-1, 1\}^k$, which maps each example to its hashing code (i.e., $y_i = f(x_i)$). Here k is the code length. A linear hashing function is utilized:

$$y_i = f(x_i) = sgn(\boldsymbol{W} x_i) \tag{6.1}$$

where $\boldsymbol{W}$ is a $k \times m$ parameter matrix representing the hashing function and sgn is the sign function.

There are two key problems that need to be addressed in supervised hashing methods: (1) how to incorporate the labeled tag information into learning effective hashing codes, and (2) how to preserve the similarity between data examples in the learned hashing codes. The SHTTM method solves the first problem by ensuring the learned hashing codes to be consistent with labeled tag information through a tag consistency component. In particular, a latent variable u_j for each tag t_j is first introduced, where u_j is a $k \times 1$ vector indicating the correlation between tag t_j and hashing codes. Then a tag consistency component can be formulated as follows:

$$\begin{aligned} &\sum_{i=1}^{n}\sum_{j=1}^{l} \|\boldsymbol{T}_{ij} - y_i^T u_j\|^2 + \alpha \sum_{j=1}^{l} \|u_j\|^2 \\ &= \|\boldsymbol{T} - \boldsymbol{Y}\boldsymbol{U}\|_F^2 + \alpha\|\boldsymbol{U}\|_F^2 \end{aligned} \tag{6.2}$$

where $\boldsymbol{T}_{ij}$ is the binary label of the *j-th* tag on the *i-th* data example. $y_i^T u_j$ can be viewed as a weighted sum that indicates how the *j-th* tag is related to the *i-th* example, and this weighted sum should be consistent with the label $\boldsymbol{T}_{ij}$ as much as possible. The second regularization term, $\sum_{j=1}^{l} \|u_j\|^2$, is introduced to avoid the overfitting issue [125]. α is trade-off parameter and $\|\|_F$ is the matrix Frobenius norm. By minimizing this component, the consistency between tags and the learned hashing codes can be ensured. Note that semantically similar tags will have similar latent variables, since these tags are often associated with common data examples, and thus the learned corresponding latent variables will be similar by ensuring the tag consistency term. In the extreme case, if two tags are assigned in exactly the same set of examples, their latent variables will be identical.

The second problem in supervised hashing methods is similarity preserving, which indicates that semantically similar examples should be mapped to similar hashing codes within a short Hamming distance. In SHTTM, it points out that the similarity calculated using the original feature vector may not reflect the semantic similarity between data examples. Therefore, it proposes to utilize features extracted from topic modeling [81] to measure the semantic similarity between data examples instead of original features, since topic modeling provides an interpretable low-dimensional representation of the data examples associated with a set of topics. SHTTM exploits the Latent Dirichlet Allocation (LDA) [83] approach of topic modeling to extract k latent topics from the data examples. Each example x_i corresponds to a distribution θ_i over the topics where two semantically similar examples have similar topic distributions. In this way, semantic similarity between data examples is preserved in the extracted topic distributions $\boldsymbol{\theta}$ and the similarity preservation component in SHTTM is defined as follows:

$$\sum_{i=1}^{n} ||y_i - \theta_i||^2 = ||\boldsymbol{Y} - \boldsymbol{\theta}||_F^2 \tag{6.3}$$

By minimizing this component, the similarity between different examples is preserved in the learned hashing codes.

Combining the tag consistency and similarity preservation components from Eqns.6.2 and 6.3, and substituting Eqn.6.1, the overall objective function for learning the hashing function and tag correlation can be formulated as:

$$\min_{\boldsymbol{W},\boldsymbol{U}} \|\boldsymbol{T} - \boldsymbol{W}\boldsymbol{X}\boldsymbol{U}\|_F^2 + \alpha\|\boldsymbol{U}\|_F^2 + \gamma\|\boldsymbol{W}\boldsymbol{X} - \boldsymbol{\theta}\|_F^2 \tag{6.4}$$

where α and γ are trade-off parameters to balance the weight between the components. Note that the sgn operator in Eqn.6.1 is relaxed to make the above optimization problem tractable. Since the objective function in Eqn.6.4 is convex with respect to either one of the two sets of parameters ($\boldsymbol{W}$ and $\boldsymbol{U}$) when the other one is fixed, the above problem can be solved iteratively by coordinate descent optimization with guaranteed convergence similar to that in SHTTM.

6.3.3 Joint Data Example and Tag Selection

The main research problem in our AH-JDETS approach is to actively select a small set of informative data examples and tags for users to label, which can be utilized for learning a better hashing function. The goal of joint data example and tag selection is that we hope to identify a set of L data examples, $\mathcal{A}_d$, together with a set of M tags, $\mathcal{A}_t$, such that the labeling information of the selected data can best boost the performance of supervised hashing. During the labeling process, for a given data example and a tag, users label 1 or 0 to denote whether this specific tag is assigned to this data example or not. The labeling cost will depend on the total number of selected tags that the users need assign to the selected data examples, which can be measured as LM. The reason is that users need to label either 1 or 0 for each data example and tag pair, and there are total LM such pairs. We will provide more details and possibilities of measuring the labeling cost with respect to the sizes of L and M for learning hashing function later in the experiments. As mentioned, for large scale datasets, there might be millions of data examples associated with thousands of different tags, which makes it impractical for labeling all tags to every data example. Therefore, it is important to select only a small set of most informative data examples and tags for users to label to save user labeling efforts.

An important question to ask is how to measure the informativeness of data examples and tags? In this dissertation, we propose to combine two measuring criteria, data uncertainty and data dissimilarity, which are widely used in active learning literature [120–122]. In particular, the selected data (both data examples and tags) are more informative if they are more uncertain. For example, the hashing codes of some potential data examples are not certain/well-learned or the predicted labeling results for some potential tags are not certain. The intuitive idea is that we would gain more knowledge by labeling on uncertain data than on certain ones. On the other hand, the selected data are more informative if they are more dissimilar to each other, since similar data may provide redundant information which is not helpful to the learning process. In the following sections, we first describe the data certainty and similarity modeling based on the selection criteria respectively. Then the final objective together with the optimization algorithm will be elaborated. Finally, we discuss the computational cost of the learning algorithm.

Data Certainty Modeling

Recall that in our supervised hashing model, the binary hashing code y_i, is obtained by thresholding a linear projection of a data example x_i, i.e., $y_i = sgn(\boldsymbol{W}x_i)$, and the linear hashing function $\boldsymbol{W}$ can be viewed as k decision/classification hyperplanes, each of which is used to generate one bit of a code. More precisely, if a data example sits on the positive side of a decision hyperplane, then its corresponding hashing bit is 1, otherwise -1. The data example certainty with respect to coding can be measured by its distance to the hyperplane, which is $|wx|$. Intuitively speaking, the smaller the distance of a data example to a hyperplane, the more uncertain the data example is. Considering an extreme case where a data example lies exactly on a decision hyperplane, then it is highly uncertain to decide whether its corresponding bit should be 1 or -1. Since there are total k decision hyperplanes, we use the l_2 norm[1] to calculate the certainty, c_i^d, of a data example as:

$$c_i^d = \|\boldsymbol{W} x_i\|_2 \tag{6.5}$$

It can be seen that the data example certainty is inversely related to the informativeness, i.e., the smaller the certainty value is, the more informative it is. Then the total certainty of the selected data examples can be written as $\sum_{i \in \mathcal{A}_d} c_i^d$, where $\mathcal{A}_d$ is the active set of data examples.

Besides exploiting data example certainty, we also need to model the tag certainty. In our supervised hashing method based on SHTTM, the correlation between tags and hashing codes are learned in the latent variables $\boldsymbol{U}$. Inspired by the data example selection, we use the magnitude of u_j to represent the tag certainty as:

$$c_j^t = \|u_j\|_2 \tag{6.6}$$

where small value indicates an uncertain tag. $u = 0$ is an extreme which means this tag contributes no information in the hashing function learning. Therefore, we can compute the total certainty of the selected tags as $\sum_{j \in \mathcal{A}_t} c_j^t$. Combining the data example certainty and tag certainty terms, the joint data certainty can be modeled as:

$$\min_{\mathcal{A}_d, \mathcal{A}_t} \sum_{i \in \mathcal{A}_d} c_i^d + \phi \sum_{j \in \mathcal{A}_t} c_j^t \tag{6.7}$$

[1]Other norms such as l_1 norm may also be used.

where ϕ is a trade-off parameter to balance the weight between two data certainty terms. By minimizing the above objective function, we can select a set of data examples and tags that are most uncertain.

Data Similarity Modeling

Similar data may contain similar knowledge, which provides redundant information in the learning process and is not desirable. For instance, the tags of 'car' and 'automobile' have similar meanings and choosing both of them may not gain substantial new information over just selecting one. Therefore, selecting a set of dissimilar data to label is very important for acquiring more information, which can make the learning process more effective. The pairwise similarity $\boldsymbol{S}_{ij}$ between data example x_i and x_j can be pre-calculated as:

$$\boldsymbol{S}_{ij} = e^{-\frac{\|x_i - x_j\|^2}{\sigma^2}} \tag{6.8}$$

where σ^2 is the bandwidth parameter. Note that we use the Gaussian function/kernel to calculate the similarity in this dissertation due to its popularity in many hashing methods [10, 22], but other similarity criteria may also be used, such as cosine similarity or inner product similarity. Then the total sum of similarity values among the selected data examples can be calculated as $\sum_{(i,j)\in\mathcal{A}_d} \boldsymbol{S}_{ij}$.

For the tag similarity between t_i and t_j, we directly calculate it as the inner product of their corresponding tag correlation vectors, $u_i^T u_j$. Then the sum of similarity values between selected tags can be written as $\sum_{(i,j)\in\mathcal{A}_t} u_i^T u_j$. Combining the above two similarity terms, the joint data similarity can be modeled as:

$$\min_{\mathcal{A}_d,\mathcal{A}_t} \sum_{(i,j)\in\mathcal{A}_d} \boldsymbol{S}_{ij} + \lambda \sum_{(i,j)\in\mathcal{A}_t} u_i^T u_j \tag{6.9}$$

where λ is also a trade-off parameter to balance two data similarity terms. By minimizing the above objective function, we can select a set of data examples and tags that are most dissimilar to each other.

6.3.4 Overall Objective and Optimization

The entire objective function of joint data example and tag selection integrates two components such as data certainty component in Eqn.6.7 and data similarity component given in Eqn.6.9 as:

$$\min_{\mathcal{A}_d,\mathcal{A}_t} \sum_{i\in\mathcal{A}_d} c_i^d + \phi \sum_{j\in\mathcal{A}_t} c_j^t + \beta \sum_{(i,j)\in\mathcal{A}_d} \boldsymbol{S}_{ij} + \lambda \sum_{(i,j)\in\mathcal{A}_t} u_i^T u_j \tag{6.10}$$

To formalize the above objective, we introduce an indicating vector $\mu_d \in \{0,1\}^n$ whose entries specify whether or not the corresponding data examples are selected, i.e., $\mu_{d_i} = 1$ when x_i is selected and $\mu_{d_i} = 0$ when x_i is not selected. Similarly, an indicating vector $\mu_t \in \{0,1\}^l$ for tags is also used. Then the above formulation can be rewritten as:

$$\begin{aligned} \min_{\mu_d,\mu_t} \quad & {\mu_d}^T C_d + \phi {\mu_t}^T C_t + \beta \ {\mu_d}^T \boldsymbol{S} \mu_d + \lambda \ {\mu_t}^T \boldsymbol{U}^T \boldsymbol{U} \mu_t \\ s.t. \ \ & \mu_d \in \{0,1\}^n, \ \mu_t \in \{0,1\}^l, \ {\mu_d}^T \mathbf{1} = L, \ {\mu_t}^T \mathbf{1} = M \end{aligned} \tag{6.11}$$

where $C_d = [c_1^d, c_2^d, \ldots, c_n^d]^T$ is the data example certainty vector and $C_t = [c_1^t, c_2^t, \ldots, c_l^t]^T$ is the tag certainty vector. ϕ, β and λ are trade-off parameters. $\mathbf{1}$ is a vector of all ones and the constraints ${\mu_d}^T\mathbf{1} = L$ and ${\mu_t}^T\mathbf{1} = M$ mean we wish to select exact L data examples together with M tags. The first two terms in the objective function are the sum of certainty values of the selected data, and the last two terms are the sum of similarity values between them. By minimizing the above objective function, we can jointly select a set of data examples and tags that are most uncertain and dissimilar with each other.

Directly minimizing the objective function in Eqn.6.11 is intractable due to the integer constraints, which makes the problem NP-hard to solve. Therefore, we propose to relax these constraints to the continuous constraints $\mathbf{0} \le \mu_d \le \mathbf{1}$ and $\mathbf{0} \le \mu_t \le \mathbf{1}$ and further decompose the optimization problem into two sub-problems as:

$$\begin{aligned} \min_{\mu_d} \quad & {\mu_d}^T C_d + \beta \ {\mu_d}^T \boldsymbol{S} \mu_d \\ s.t. \quad & \mathbf{0} \le \mu_d \le \mathbf{1}, \quad {\mu_d}^T \mathbf{1} = L \end{aligned} \tag{6.12}$$

and

$$\begin{aligned} \min_{\mu_t} \quad & {\mu_t}^T C_t + \lambda\ {\mu_t}^T \boldsymbol{U}^T \boldsymbol{U} \mu_t \\ s.t. \quad & \mathbf{0} \leq \mu_t \leq \mathbf{1}, \quad {\mu_t}^T \mathbf{1} = M \end{aligned} \tag{6.13}$$

where $\mathbf{0}$ is a vector of all zeros. These two relaxed sub-problems are standard constrained quadratic programs (QP) which can be solved efficiently using convex optimization methods, such as successive linear programming (SLP) [126] and the bundle method [127]. After obtaining the relaxed solution from Eqn.6.12, we select L data examples with the largest μ_d values to form the active data example set $\mathcal{A}_d$. Similarly, we choose M tags with the largest μ_t values based on the relaxed solution from Eqn.6.13 to form the active tag set $\mathcal{A}_t$. Finally, we will request the users to label all tags from the selected tag set $\mathcal{A}_t$ to every data example in $\mathcal{A}_d$, and update the labeled information $\boldsymbol{T}$ to retrain the supervised hashing model (see Figure 6.1). The alternative process of learning hashing function and actively selecting data can be repeated for several iterations. We will discuss more in the experiments. The full AH-JDETS algorithm including supervised hashing and joint data example and tag selection is summarized in Table 6.1.

6.3.5 Computational Complexity Analysis

This section provides some analysis on the training cost of our AH-JDETS approach. The learning algorithm of AH-JDETS consists of two main parts: supervised hashing and joint selection of data examples and tags. For the supervised hashing method, we iteratively solve the optimization problem in Eqn.6.4 to obtain the hashing function and tag correlation, where the time complexity is bounded by $O(nlk + nk^2)$. The second part involves selecting data examples and tags by solving the two relaxed optimization problem in Eqns.6.12 and 6.13. Since these are standard quadratic program problems, the time complexity for obtaining μ_d and μ_t is bounded by $O(n^2 + l^2)$. Thus, the total time complexity of the learning algorithm is bounded by $O(nlk + nk^2 + n^2 + l^2)$. For large scale dataset, $O(n^2)$ cost for data example

Table 6.1.
Active Hashing with Joint Data Example and Tag Selection

```
Input: Training examples $\boldsymbol{X}$, initial labeled data $\boldsymbol{T}$ and parameters
    $\alpha, \beta, \gamma$ and $\lambda$.
Output: Hashing function $\boldsymbol{W}$, tag correlation $\boldsymbol{U}$ and hashing codes $\boldsymbol{Y}$.
Compute $\boldsymbol{S}$ and $\boldsymbol{\theta}$, initialize $L$ and $M$.
Supervised Hashing
    Solve the optimization problem in Eqn.6.4 to obtain $\boldsymbol{W}$ and $\boldsymbol{U}$.
Joint Data Example and Tag Selection
    Calculate data certainty $C_d$ and $C_t$ using Eqns.6.5 and 6.6 based on
        $\boldsymbol{W}$ and $\boldsymbol{U}$.
    Calculate tag similarity $\boldsymbol{U}^T\boldsymbol{U}$.
    Solve the optimization problem in Eqn.6.12 to select a set of data
        examples $\mathcal{A}_d$.
    Solve the optimization problem in Eqn.6.13 to select a set of tags $\mathcal{A}_t$.
Labeling tags from $\mathcal{A}_t$ to data examples in $\mathcal{A}_d$ and update labeled data $\boldsymbol{T}$.
Repeat for several iterations.
Generate the hashing codes $\boldsymbol{Y}$ using Eqn.6.1.
```

selection might not be feasible. In practice, we reduce the computational cost in the experiments by only considering the top 10% data examples corresponding to the smallest certainty values without much loss in accuracy. However, the training process is always conducted off-line and our focus of efficiency is on the retrieval process. This process of generating hashing code for a query example only involves some dot products and comparisons between two binary vectors, which can be done in $O(mk + k)$ time.

6.4 Experiments

This section presents an extensive set of experiments to demonstrate the advantages of the proposed approach.

6.4.1 Datasets and Implementation

We conduct experiments on four datasets, including two image datasets and two text collections as follows:

1. $Flickr1m$ [86] is collected from Flicker images for image annotation and retrieval tasks. This dataset contains 1 million image examples associated with more than $7k$ unique tags. A subset of $250k$ image examples with the most common $3k$ tags is used in our experiment. We randomly choose $240k$ image examples as a training set and $10k$ for query testing.

2. *NUS-WIDE* [84] is created by NUS lab for evaluating image retrieval techniques. It contains $270k$ images associated with about $5k$ different tags. We use a subset of $110k$ image examples with the most common $3k$ tags in our experiment. We randomly partition this dataset into two parts, $100k$ for training and $10k$ for query testing.

3. $ReutersV1$ (Reuters-Volume I): This dataset contains over $800k$ manually categorized newswire stories [89]. There are in total 126 different tags associated with this dataset. A subset of $130k$ documents of $ReutersV1$ is used in our experiment by discarding those documents with less than 3 labels. $120k$ text documents are randomly selected as the training data, while the remaining $10k$ documents are used as testing queries.

4. $Reuters$ (Reuters21578)[2] is a collection of text documents that appeared on Reuters newswire in 1987. It contains 21578 documents, and 135

[2]http://daviddlewis.com/resources/textcollections/reuters2 1578/.

tags/categories. In our experiments, documents with less than 3 labels are removed. The remaining 13713 documents are randomly partitioned into a training set with 12713 documents and 1000 test queries.

512-dimensional GIST descriptors [87] are used as image features and *tf-idf* features are used to represent the documents.

We implement our method using Matlab on a PC with an Intel Duo Core i5-2400 CPU 3.1GHz and 8GB RAM. The parameters α, β, γ and λ are tuned by 3-fold cross validation on the training set through the grid $\{0.01, 0.1, 1, 10, 100\}$ and we will discuss more details how they affect the performance of our approach later. We randomly select $2k$ labeled data examples in the training set to form the initial tag matrix $\boldsymbol{T}$. We repeat each experiment 10 times and report the result based on the average over the 10 runs. Each run adopts a random split of the dataset.

6.4.2 Evaluation Metrics

To conduct similarity search, each example in the testing set is used as a query example to search for similar examples in the corresponding training set based on the Hamming distance of their hashing codes. We follow two evaluation criteria that are commonly used in the literature [10, 20, 36]: Hamming Ranking and Hash Lookup. Hamming Ranking ranks all the data examples in the training set according to their Hamming distance from the query and the top k examples are returned as the desired neighbors. Hash Lookup returns all the data examples within a small Hamming radius r of the query. The performance is measured with standard metrics of information retrieval: precision as the ratio of the number of retrieved relevant examples to the number of all returned examples and recall as the ratio of the number of retrieved relevant examples to the number of all relevant examples. The performance is averaged over all test queries in the datasets.

Table 6.2.
Precision and recall of the top 200 retrieved examples of different selection methods on four datasets with 32 hashing bits.

	Flickr1m		*NUS-WIDE*		*ReutersV1*		*Reuters*	
Methods	Pr	Re	Pr	Re	Pr	Re	Pr	Re
$\mathcal{A}_d\mathcal{A}_t$	**0.459**	**0.114**	**0.348**	**0.085**	**0.675**	**0.306**	**0.742**	**0.237**
$\mathcal{A}_d\mathcal{R}_t$	0.431	0.101	0.334	0.081	0.663	0.295	0.738	0.236
$\mathcal{R}_d\mathcal{A}_t$	0.437	0.104	0.315	0.077	0.658	0.292	0.727	0.232
$\mathcal{R}_d\mathcal{R}_t$	0.412	0.092	0.287	0.070	0.653	0.286	0.724	0.230

6.4.3 Results and Discussions

We conduct the experiments under five different configurations to evaluate the proposed AH-JDETS approach from different perspectives.

Comparison of different selection methods

In this set of experiments, we compare our joint selection method against three other selection methods: 1. Actively select data examples and randomly select tags ($\mathcal{A}_d\mathcal{R}_t$). 2. Randomly select data examples and actively select tags ($\mathcal{R}_d\mathcal{A}_t$). 3. Randomly select data examples and Randomly select tags ($\mathcal{R}_d\mathcal{R}_t$). Clearly, our method can be regarded as $\mathcal{A}_d\mathcal{A}_t$. The size of data examples, L, is set to be 1000 for all datasets and the size of tags, M, is set to be 30 for the two image datasets and 10 for the other two text datasets. Note that for fair comparison, we adopt a modified version of [50], the $\mathcal{A}_d\mathcal{R}_t$ selection method, which has the same labeling cost as our method by randomly selecting a set of tags instead of all tags.

We report the precision and recall for the top 200 retrieved examples with 32 hashing bits in Table 6.2. The precision and recall for the retrieved examples within Hamming radius 2 are shown in Table 6.3. It can be seen that AH-JDETS gives

Table 6.3.
Precision and recall of the retrieved examples within Hamming radius 2 of different selection methods on four datasets with 32 hashing bits.

	Flickr1m		*NUS-WIDE*		*ReutersV1*		*Reuters*	
Methods	Pr	Re	Pr	Re	Pr	Re	Pr	Re
$\mathcal{A}_d\mathcal{A}_t$	**0.403**	0.186	**0.312**	**0.134**	**0.422**	**0.246**	**0.521**	**0.149**
$\mathcal{A}_d\mathcal{R}_t$	0.385	**0.189**	0.301	0.129	0.405	0.227	0.516	0.147
$\mathcal{R}_d\mathcal{A}_t$	0.367	0.175	0.296	0.123	0.387	0.232	0.504	0.144
$\mathcal{R}_d\mathcal{R}_t$	0.346	0.171	0.278	0.116	0.372	0.215	0.498	0.132

overall the best performance among all four selection methods on all datasets. From these comparison results, we can see that $\mathcal{R}_d\mathcal{R}_t$ does not perform well in terms of both precision and recall. The reason is that the randomly selected data examples and tags may be noninformative. For example, these selected data might carry much redundant information if they are very similar to each other. It is also possible that the selected data examples have high certainties. In other words, the hashing codes of these data examples are already well-learned with high quality, and thus cannot contribute much for learning more effective hashing codes and function. We can also observe from the results that our AH-JDETS method outperforms the two methods $\mathcal{A}_d\mathcal{R}_t$ and $\mathcal{R}_d\mathcal{A}_t$ which either actively select data examples or select tags. This can be attributed to the joint selection strategy in our method, since it not only identifies the most informative data examples but at the same time selects the most informative tags. In this way, the learner could gain most information from labeling these selected data. Moreover, a two-sided paired t-test [128] is used to determine the statistical significance improvements in Tables 6.2 and 6.3. T-test shows that AH-JDETS significantly outperforms $\mathcal{R}_d\mathcal{R}_t$ ($p < 0.01$), $\mathcal{R}_d\mathcal{A}_t$ ($p < 0.03$) and $\mathcal{A}_d\mathcal{R}_t$ ($p < 0.03$) on all datasets. The precision-recall curves of different selection methods with 32 hashing bits on *Flickr1m* and *ReutersV1* are reported in Fig.6.2. It can be seen that among all of these comparison methods, AH-JDETS shows the best performance, which is consistent with the results in Tables 6.2 and 6.3. We have also observed similar results on the other two datasets. But due to the limit of space, they are not presented here.

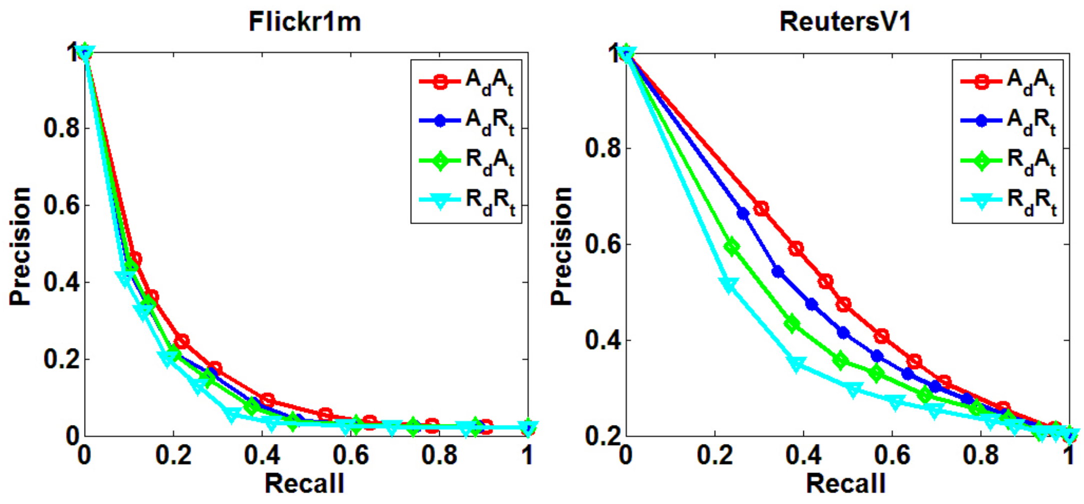


Figure 6.2. Results of Precision-Recall curve with 32 hashing bits on $Flickr1m$ and $ReutersV1$ datasets.

To evaluate the effect of different code lengths, we conduct another experiment by varying the number of hashing bits in the range of $\{8, 16, 32, 64, 128\}$. The precisions for the top 200 retrieved examples with different numbers of hashing bits on four datasets are shown in Fig.6.3. From this figure, we observe that larger code length gives better precision results on all datasets. It also can be seen that our method consistently outperforms the other three methods under different code lengths.

Comparison with SHTTM and STH

We compare our AH-JDETS with two state-of-the-art hashing methods, Semantic Hashing using Tags and Topic Modeling (SHTTM) [20] and Self-Taught Hashing (STH) [22], on all datasets to demonstrate the advantages of our active hashing

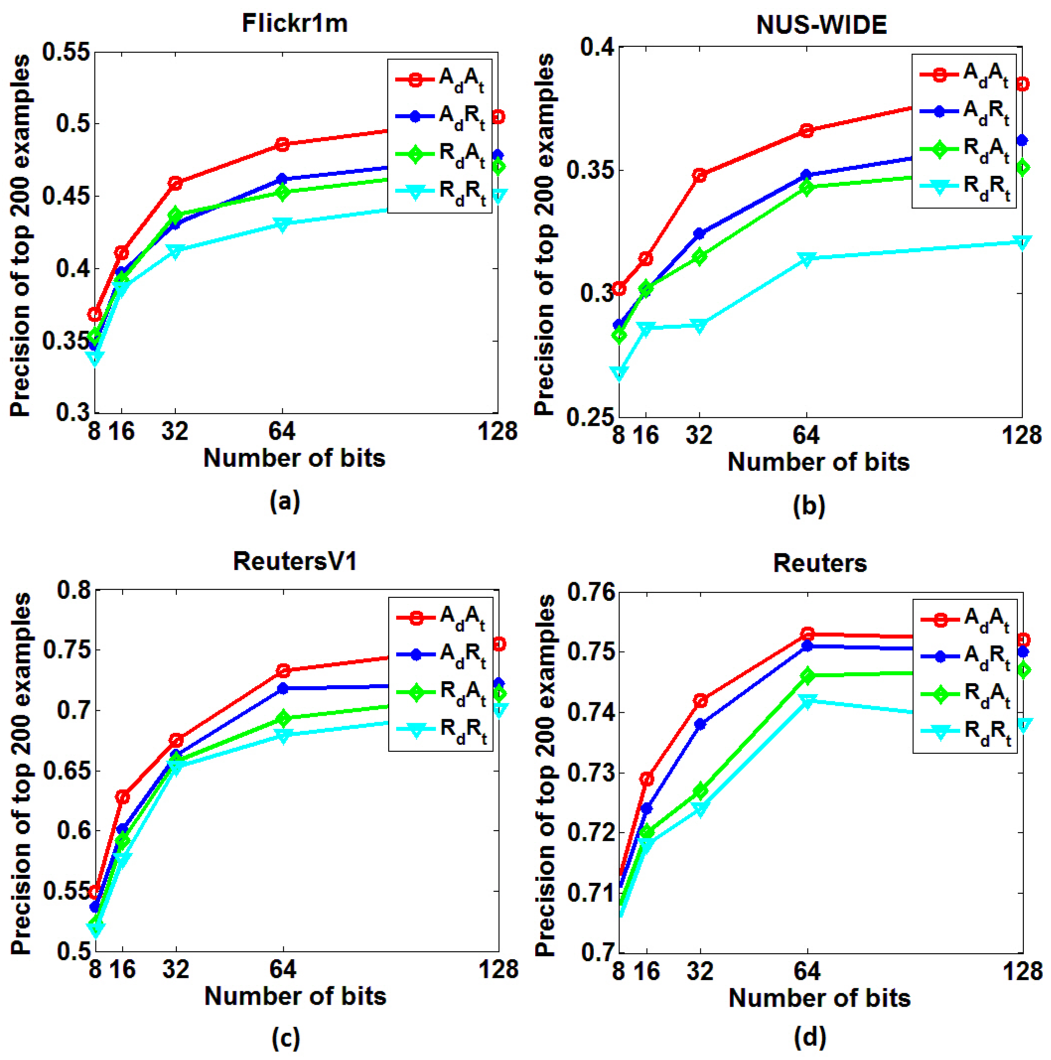


Figure 6.3. Results of precision of top 200 examples on four datasets with different hashing bits.

approach. SHTTM is a passive supervised hashing method which is briefly discussed in section 4. STH is an unsupervised hashing method that does not utilize any of the label information.

Precisions for the top 200 retrieved examples and the labeling costs of all three methods using 32 hashing bits are reported in Table 6.4. Note that there is no labeling cost for STH since it does not require any label knowledge. SHTTM is a

Table 6.4.
Comparison with two state-of-the-art hashing methods. Results of precision and labeling cost of the top 200 retrieved examples with 32 hashing bits.

	$Flickr1m$		*NUS-WIDE*	
Methods	Precision	Cost	Precision	Cost
SHTTM [20]	0.508	$33.5m$	0.363	$16.6m$
AH-JDETS $(5L, 2M)$	0.499	$300k$	0.358	$300k$
AH-JDETS (L, M)	0.459	$30k$	0.348	$30k$
STH [22]	0.393	0	0.285	0
	$ReutersV1$		$Reuters$	
Methods	Precision	Cost	Precision	Cost
SHTTM [20]	0.712	$4.8m$	0.754	$2.64m$
AH-JDETS $(5L, 2M)$	0.704	$100k$	0.751	$100k$
AH-JDETS (L, M)	0.675	$10k$	0.742	$10k$
STH [22]	0.633	0	0.721	0

supervised hashing method that incorporates all labels into hashing function learning. Its labeling cost is simply the number of total labels associated with the training examples. For our AH-JDETS, the labeling cost is LM since we have to label each data example and tag pair and there are total LM such pairs. We evaluate AH-JDETS with two different labeling costs, (L,M) and (5L,2M), where L=1k and M=30 for image datasets and L=1k and M=10 for text datasets. From these comparison results we can see that AH-JDETS achieves much better performance than the unsupervised method STH, while it also obtains comparable results with SHTTM (especially on $(5L, 2M)$ setting) but requiring much less labeling cost. For example, the labeling cost for SHTTM is about 1100 times more than our AH-JDETS with 1k data examples and 30 tags on $Flickr1m$, which is impractical. Therefore, our AH-JDETS approach can be viewed as a good trade-off between unsupervised hashing and passive supervised hashing methods, which achieves good performance but saving much labeling efforts and thus overcomes the limitation of passive hashing methods.

Varying set size L and M

In this set of experiments, we evaluate the performance of AH-JDETS and $\mathcal{A}_d\mathcal{R}_t$ methods by varying the set size L and M. It is obvious that we would gain more information by selecting more data examples and tags simultaneously. An interesting question would be: given the same labeling effort, how should we choose L and M to achieve best performance? Therefore, to answer this question, we fix the labeling cost $LM = 30k$ for image datasets and $LM = 10k$ for text datasets, and vary both L and M in the experiments. The code length is set to be 32 in all experiments.

The precision results of top 200 retrieved examples with respect to data example size on all datasets are shown in Fig.6.4. We can observe that the performances of both two methods are not satisfactory when selecting either very few data examples or tags (correspond to two end points in the figure). For example, in $Flickr1m$ dataset, the right most red point corresponds to the combination of selecting 15000 data examples with 2 tags, while the left most red point represents the choice of 10 data examples with all 3000 tags. Although these two configurations have the same labeling cost $30k$, the performance is the worst among all possible combinations. It can be seen that the optimal combination is around 1200 data examples with 25 tags for our method on $Flickr1m$, which is roughly proportional to the total number of data examples and tags. Therefore, it is important to choose a good combination of L and M to achieve the best performance.

Consider a more realistic labeling cost measure scenario, where users are assigning M possible tags to a document or an image. Usually people first read through the content of the document or view the content of an image, which takes a reading cost r (may vary for documents and images), and then assign M labels to it. In this

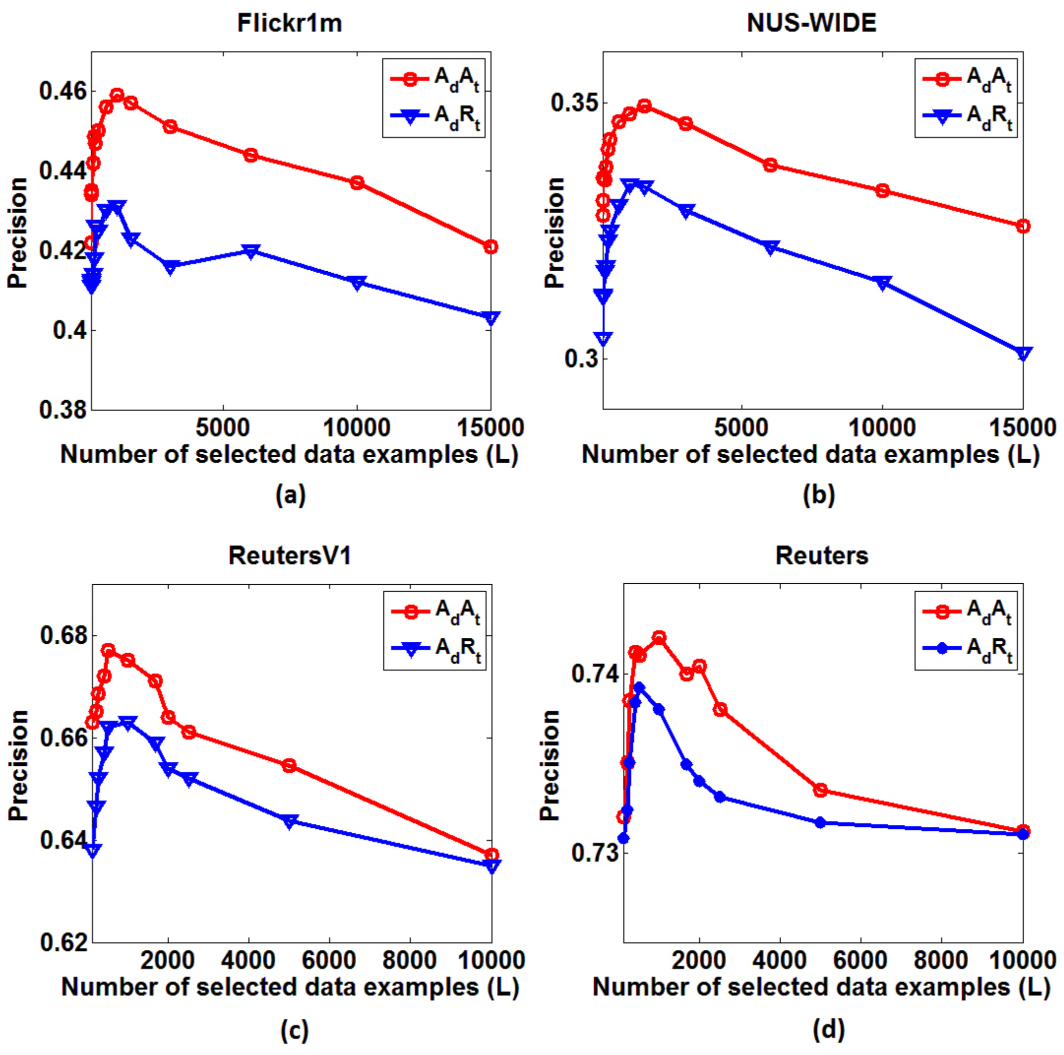


Figure 6.4. Precision results of varying batch sizes while fixing the labeling cost. Hashing bits are set to be 32 for all datasets.

case, the labeling cost for a data example is $r + M$ and the total labeling cost for L examples is $(r + M)L$. Using this cost measure, we conduct another experiment on $Flickr1m$ by fixing r=5 and have found similar pattern as in Fig.6.4(a). However, we observe that the optimal combination drifts toward smaller number of data examples with more tags (around 940 data examples with 27 tags), which is consistent with our expectation due to the reading cost associated with each data example.

Varying number of iterations

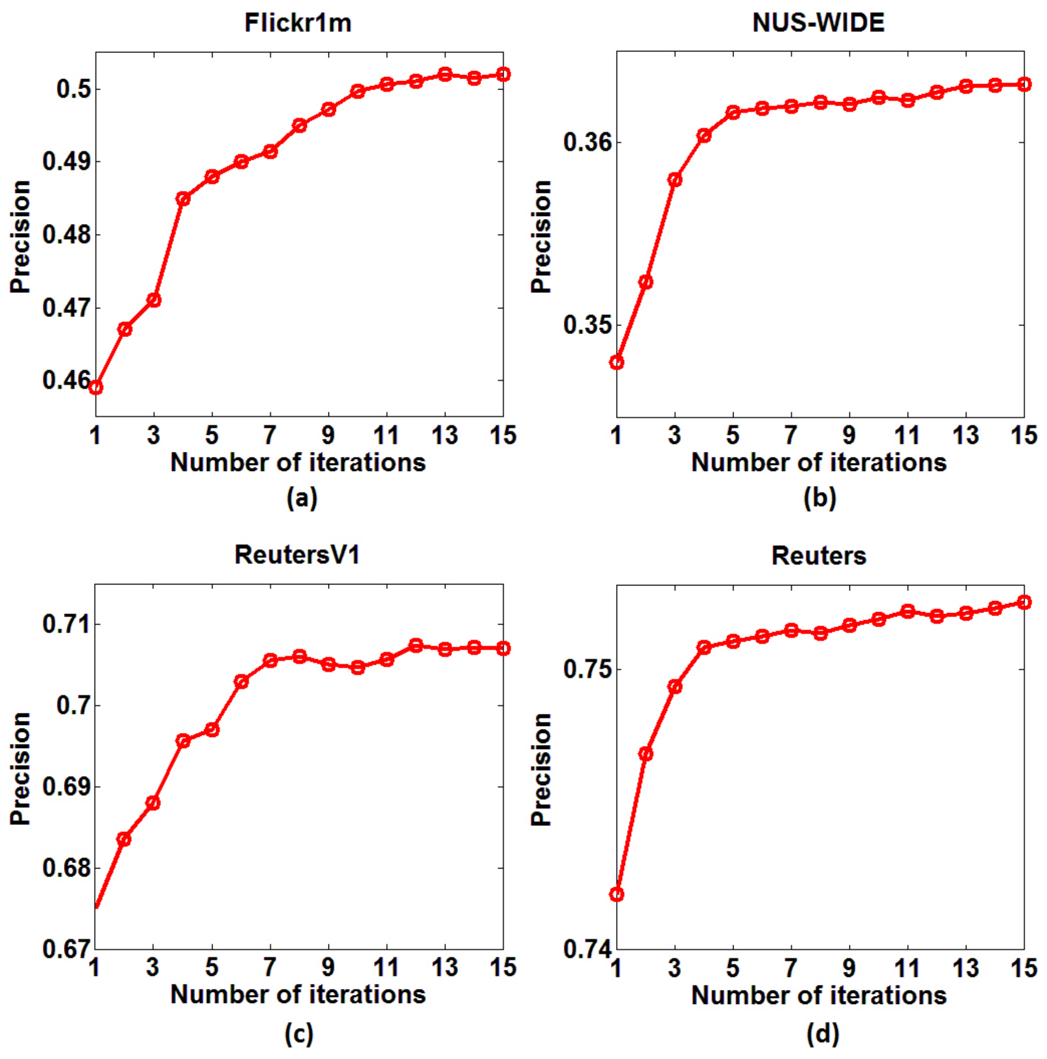


Figure 6.5. Precision results of increasing number of iterations on four datasets with 32 hashing bits.

In our AH-JDETS approach, after obtaining the label information on the active set, we will update the labeled tags $\boldsymbol{T}$ to retrain the supervised hashing model as shown in Figure 6.1. The alternative process of learning hashing function and actively selecting data can be repeated for several iterations. In this set of experiments, we evaluate the performance of AH-JDETS by varying the number of iterations from 1 to 15 on all datasets. Note that the labeling cost grows linearly with the number of iterations since during each iteration the labeling cost is identical, i.e., LM.

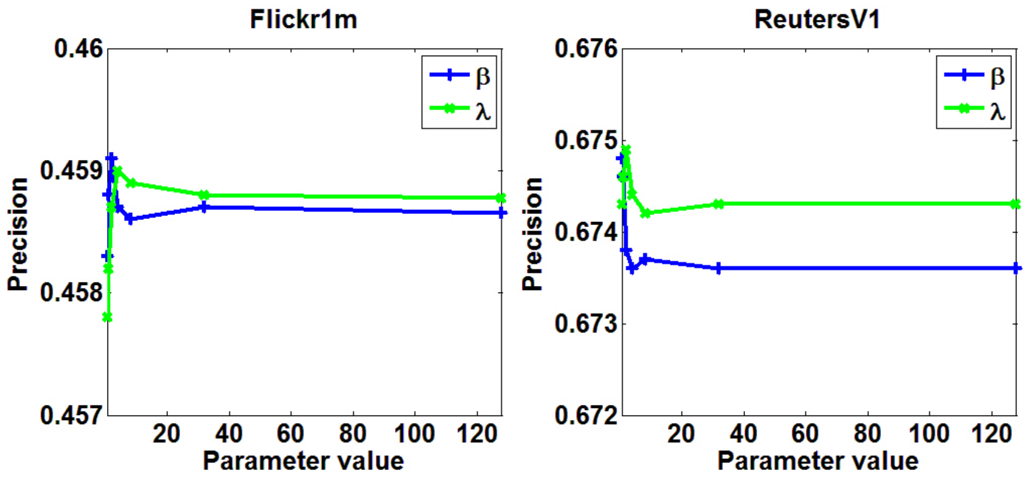


Figure 6.6. Parameter Sensitivity for β and λ. Results of precision of the top 200 examples with 32 hashing bits.

Precisions for the top 200 examples with 32 hashing bits are reported in Fig.6.5. Not surprisingly, we can observe that the precision increases with the increasing number of iterations. However, we have found that the performance of AH-JDETS does not increase much after the first few iterations. Our hypothesis is that we have gained sufficient knowledge to learn a good hashing function within the first few iterations. In other words, the labeled data from the later iterations contains more and more redundant information, which does not contribute much for retraining a better hashing function. Therefore, the proposed AH-JDETS approach can obtain good performance in a few iterations, which also saves the labeling cost.

Parameter Sensitivity

There are four trade-off parameters in AH-JDETS, α and γ in supervised hashing component, and β and λ in the joint data example and tag selection method. To prove the robustness of the proposed joint selection method, we conduct parameter sensitivity experiments of β and λ on all datasets. In each experiment, we tune the parameter from $\{0.5, 1, 2, 4, 8, 16, 32, 128\}$. We report the results on $Flickr1m$ and $ReutersV1$ in Fig.6.6. It is clear from these experimental results that the performance of AH-JDETS is relatively stable with respect to β and λ. We have also observed similar results of the proposed method in the other two datasets. On the other side, it has already been shown in SHTTM that the supervised hashing method is robust with respect to a wide range of α and γ.

7 CONCLUSIONS

The past few years have witnessed tremendous interest in learning compact codes for efficient large scale similarity search. One major challenge in these methods is that it is often difficult to incorporate different types of supervision from multiple sources in training. This dissertation analyzes five problems of learning to hash based on different objectives from multiple information sources: 1. learning from semantic tags; 2. learning with partial multi-modal data; 3. learning from structured data; 4. learning ranking preserving hashing codes; 5. active hashing for insufficient supervision. This chapter concludes the dissertation by summarizing the contributions in Chapters 2, 3, 4, 5 and 6, providing some further discussions based on that, and pointing out future directions.

7.1 Main Contributions

Hashing methods generate promising results in large scale similarity search. However, the problem of leveraging supervised information from multiple sources has not been fully explored. The major contribution of this dissertation is to leverage different types of supervised knowledge from multi-sources into learning effective hashing codes. In particular, we propose a unified framework that learns effective hashing codes by simultaneously ensuring the data consistency to the supervised knowledge (e.g., semantic tags, structure graph, relevance feedback) and preserving the data similarity. In addition, we explicitly dealing with the missing data problem from multiple sources. Furthermore, we also handle the situation where the supervision is insufficient or even not available. The unified learning framework is formulated as follows:

$$\begin{aligned} &\min_{Y}\ C(Y,S) + \alpha\ D(Y,X) + \beta\ R(Y) \\ &\quad s.t. \quad constraints \quad on \quad Y \end{aligned} \tag{7.1}$$

where Y is the hashing codes we are trying to learn. $C()$ is the consistency term which ensures the consistency between hashing codes and the supervised information. $D()$ is the data similarity preservation term and $R()$ is the regularization term. The optimization problem in Eqn.7.1 is then solved by relaxation and efficient coordinated gradient descent method, which scales linearly with the number of data examples. Therefore, our framework is suitable and can be applied to large scale data. We summarize our contributions in details.

Semantic tags or labels are usually associated with data examples and have been popularly utilized in many applications, which provide useful supervised knowledge for users to better categorize or search desired data. This dissertation presents a novel semi-supervised tag hashing approach to incorporate the semantic tags into hashing codes learning. The proposed method fully exploits tag information by modeling the semantic correlation between tags and hashing bits. The hashing function is learned in a unified learning framework by simultaneously ensuring the tag consistency and preserving the similarities between data examples. Moreover, the effectiveness of hashing function is further improved through orthogonal transformation by minimizing the quantization error. Furthermore, we extend this framework by preserving the topic level similarity between data examples to obtain more effective codes when original feature distances do not reflect the similarity between data examples. The experimental results on several image and text datasets demonstrate the advantages of the proposed method.

In many applications, data examples are usually represented by multiple modalities captured from different sources, such as tags, texts, etc. However, in real world tasks, it is often the case that every modality suffers from some missing information, which results in many partial examples. This dissertation describes a novel partial multi-modal hashing approach to deal with such partial data. A unified learning framework is developed to learn the binary codes, which simultaneously ensures the data consistency among different modalities via latent subspace learning, and preserves data similarity within the same modality through graph Laplacian. A block gradient descent algorithm is applied as the optimization procedure. Experiments on two datasets show the superior performance of the proposed approach over several state-of-the-art multi-modal hashing methods.

The dependencies between data examples naturally exist and if incorporated in models, they can potentially improve the hashing code performance significantly. In this dissertation, a novel approach of hashing on structured data (HSD) is proposed, which incorporates the structure information associated with data. The hashing function is learned in via ensuring the structural consistency and preserving the similarities between data examples jointly. We also develop an iterative gradient descent algorithm as the optimization procedure. Experimental results on two datasets demonstrate that structure information is indeed useful in hashing codes learning.

In many real world search systems, it is more realistic and desirable that the most relevant examples to a query can be presented in front of less relevant ones. In other words, users prefer the retrieval results with better ranking performance. This dissertation proposes a ranking preserving hashing model that directly optimizes the popular ranking accuracy measure, Normalized Discounted Cumulative Gain (NDCG), to learn effective hashing codes that not only preserves the ranking order but also models the relevance values of data examples to the queries in the training data. We handle the difficulty of non-convex non-smooth optimization by using the expectation of NDCG measure calculated based on the linear hashing function and then solve the relaxed smooth optimization problem with a gradient descent method. Extensive sets of experiments on two large scale datasets demonstrate the superior search accuracy of the proposed approach over several state-of-the-art hashing methods.

When the supervised information is insufficient or even not available, it is important to design effective methods to actively identify only a small set of the most informative data examples for users to label. In this dissertation, we proposes a novel active hashing approach to actively select the most informative data examples and tags in a joint manner for hashing function learning. The selection principle is to identify data examples and tags that are both uncertain and dissimilar with each other. The labeled information is utilized together with unlabeled data to generate an effective hashing function. Extensive experiments on four different datasets demonstrate the advantage of the proposed approach for learning a more effective hashing function with small labeling costs than the baseline passive supervised learning and some other active learning methods.

7.2 Future Directions

In this section, several possible future topics are discussed to extend the research in this dissertation.

First of all, in this dissertation, we propose several models to incorporate the supervised tag information into the hashing codes learning. In some certain applications, tags may be very sparse or even do not exists in the data, resulting in insufficient supervision for training effective hashing function. In Chapter 6, we trying to solve this issue by actively select a few number of data examples for user to label. However, with a limit amount of resource, still a very few labels could be acquired. On the other side, tags or semantic labels commonly exist in many well known applications/domains. Therefore, it will be interesting if we can apply transfer learning techniques to propagate or transfer the tag/label knowledge from source domains to the target domain, such that the performance of the learned hashing codes on the target domain could be boosted by leverage the supervised information from other source domains. Several very recent work are aiming at learning hashing codes for different domains jointly by extract some common structure from the data in different domains. However, how to transfer knowledge from source domains to target domain still remails an open problem.

Secondly, in Chapter 5, we discuss the learning of ranking preserving hashing codes. In our experiments, we found that to achieve a reasonable performance, it usually requires large number of bits to represent the data examples. The reason is that with very few hashing bits, the number of possible Hamming distance to a query code is small (equal to the number of hashing bits). Therefore, many candidate examples will have equal Hamming distance from the query, which make it difficult to rank these equal-distance examples. One possible solution to this problem is to assign different weights to different hashing bits and then re-rank the data examples based on the weighted Hamming distances. Actually, some recent work have been proposed to learning weighted hashing codes. But there still exist some challenges: 1) how to determine the weights and how to adjust them according to the query? 2) how to retrieve or rank the examples efficiently since computing the weighted Hamming distance will involve real value operations rather than bit operations.

Thirdly, in this dissertation, we mainly focus on learning effective hashing codes given all the training data available. However, in many real world applications, data comes as a stream and it is difficult to have all the training data available. Therefore, it is a very important research problem to design hashing method that can dealing with data examples that are gathered in an online manner. A naive and straightforward way would be learning hashing functions periodically, which means we train the model once after we obtain a certain amount of new data. However, there are several main drawbacks: 1) it will be very computational intensive to re-train the whole model frequently; 2) it is difficult to determine when to re-train the model. Thus, it remains an open research area to develop an online learning method that can efficiently adjust the hashing function based on the newly arriving data without training the whole model again.

Fourthly, traditional hashing methods usually learn the hashing codes based on the global features of the data examples. However, in image and text retrieval tasks, the semantic similarity between two examples is often determined by certain local features. More specifically, the concept of an image may come from a certain area or region of the image instead of the whole image. Similarly, the most important part of a document can only comes from a certain paragraph. In this case, it is possible to explore multiple instance learning techniques to design more effective hashing codes based on local features. In multiple instance learning, each data example is represented by multiple instances. For example, each image can be divided/segmented into several regions (instances). Each documents can be separated into different paragraphs (instances). Therefore, how to combine the multiple instance learning method into learning hashing codes will be an interesting research direction.

Last but not least, as discussed in this dissertation, when dealing with large scale data, the scalability is an very important and realistic problem especially for real world applications. Although we apply stochastic optimization method during our training process to accelerate the convergence speed, there are still a lot of opportunities to improve the learning speed. For example, some sequential learning approaches might be able to make the convergence faster. Moreover, if we can derive a way to learn the hashing bits independently during each iteration, then the whole learning process can be simply paralleled and thus terminated much faster. We will closely look at this direction.

LIST OF REFERENCES

LIST OF REFERENCES

[1] Gerard Salton. Developments in automatic text retrieval. *Science*, 253(5023):974–980, August 1991.

[2] Gerard Salton and Chris Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[3] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*. The MIT Press, 2001.

[4] J. Shane Culpepper and Alistair Moffat. Efficient set intersection for inverted indexing. *ACM Transactions on Information Systems*, 29(1):1, 2010.

[5] Fabrizio Silvestri and Rossano Venturini. Vsencoding: efficient coding and fast decoding of integer lists via dynamic programming. In *CIKM*, pages 1219–1228, 2010.

[6] Justin Zobel and Alistair Moffat. Inverted files for text search engines. *ACM Comput. Surv.*, 38(2), 2006.

[7] Roger Weber, Hans-Jörg Schek, and Stephen Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB*, pages 194–205, 1998.

[8] Ruslan Salakhutdinov and Geoffrey E. Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

[9] Hao Xu, Jingdong Wang, Zhu Li, Gang Zeng, Shipeng Li, and Nenghai Yu. Complementary hashing for approximate nearest neighbor search. In *ICCV*, pages 1631–1638, 2011.

[10] Dan Zhang, Fei Wang, and Luo Si. Composite hashing with multiple information sources. In *SIGIR*, pages 225–234, 2011.

[11] Simon Korman and Shai Avidan. Coherency sensitive hashing. In *ICCV*, pages 1607–1614, 2011.

[12] Wei Liu, Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Hashing with graphs. In *ICML*, pages 1–8, 2011.

[13] Wei Liu, Jun Wang, Yadong Mu, Sanjiv Kumar, and Shih-Fu Chang. Compact hyperplane hashing with bilinear functions. *ICML*, 2012.

[14] Christoph Strecha, Alexander A. Bronstein, Michael M. Bronstein, and Pascal Fua. LDAHash: Improved matching with smaller descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(1):66–78, 2012.

[15] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Symposium on Computational Geometry*, pages 253–262, 2004.

[16] Yair Weiss, Rob Fergus, and Antonio Torralba. Multidimensional spectral hashing. In *ECCV (5)*, pages 340–353, 2012.

[17] Ruei-Sung Lin, David A. Ross, and Jay Yagnik. Spec hashing: Similarity preserving algorithm for entropy-based coding. In *CVPR*, pages 848–854, 2010.

[18] Gregory Shakhnarovich, Paul A. Viola, and Trevor Darrell. Fast pose estimation with parameter-sensitive hashing. In *ICCV*, pages 750–759, 2003.

[19] Yair Weiss, Antonio Torralba, and Robert Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2008.

[20] Qifan Wang, Dan Zhang, and Luo Si. Semantic hashing using tags and topic modeling. In *SIGIR*, pages 213–222, 2013.

[21] Maxim Raginsky and Svetlana Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *NIPS*, pages 1509–1517, 2009.

[22] Dell Zhang, Jun Wang, Deng Cai, and Jinsong Lu. Self-taught hashing for fast similarity search. In *SIGIR*, pages 18–25, 2010.

[23] Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*, pages 2130–2137, 2009.

[24] Alexis Joly and Olivier Buisson. Random maximum margin hashing. In *CVPR*, pages 873–880, 2011.

[25] Jong Wook Kim, K. Selçuk Candan, and Jun'ichi Tatemura. Efficient overlap and content reuse detection in blogs and online news articles. In *WWW*, pages 81–90, 2009.

[26] Qi Zhang, Yue Zhang, Haomin Yu, and Xuanjing Huang. Efficient partial-duplicate detection based on sequence matching. In *SIGIR*, pages 675–682, 2010.

[27] Qi Zhang, Yan Wu, Zhuoye Ding, and Xuanjing Huang. Learning hash codes for efficient content reuse detection. In *SIGIR*, pages 405–414, 2012.

[28] Jun Wang, Ondrej Kumar, and Shih-Fu Chang. Semi-supervised hashing for scalable image retrieval. In *CVPR*, pages 3424–3431, 2010.

[29] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Semi-supervised hashing for large-scale search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(12):2393–2406, 2012.

[30] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. Supervised hashing with kernels. In *CVPR*, pages 2074–2081, 2012.

[31] Jun Wang, Wei Liu, Andy Sun, and Yu-Gang Jiang. Learning hash codes with listwise supervision. In *ICCV*, 2013.

[32] Ke Zhou and Hongyuan Zha. Learning binary codes for collaborative filtering. In *KDD*, pages 498–506, 2012.

[33] Zhiwei Zhang, Qifan Wang, Lingyun Ruan, and Luo Si. Preference preserving hashing for efficient recommendation. In *SIGIR*, 2014.

[34] Zeynep Akata, Florent Perronnin, Zaïd Harchaoui, and Cordelia Schmid. Label-embedding for attribute-based classification. In *CVPR*, pages 819–826, 2013.

[35] Yunchao Gong, Qifa Ke, Michael Isard, and Svetlana Lazebnik. A multi-view embedding space for modeling internet images, tags, and their semantics. *International Journal of Computer Vision*, 106(2):210–233, 2014.

[36] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.

[37] Albert Gordo, José A. Rodríguez-Serrano, Florent Perronnin, and Ernest Valveny. Leveraging category-level labels for instance-level image retrieval. In *CVPR*, pages 3045–3052, 2012.

[38] Dan Zhang, Yan Liu, Luo Si, Jian Zhang, and Richard D. Lawrence. Multiple instance learning on structured data. In *NIPS*, pages 145–153, 2011.

[39] Michael M. Bronstein, Alexander M. Bronstein, Fabrice Michel, and Nikos Paragios. Data fusion through cross-modality metric learning using similarity-sensitive hashing. In *CVPR*, pages 3594–3601, 2010.

[40] Shaishav Kumar and Raghavendra Udupa. Learning hash functions for cross-view similarity search. In *IJCAI*, pages 1360–1365, 2011.

[41] Mingdong Ou, Peng Cui, Fei Wang, Jun Wang, Wenwu Zhu, and Shiqiang Yang. Comparing apples to oranges: a scalable solution with heterogeneous hashing. In *KDD*, pages 230–238, 2013.

[42] Yi Zhen and Dit-Yan Yeung. Co-regularized hashing for multimodal data. In *NIPS*, pages 1385–1393, 2012.

[43] Guiguang Ding, Yuchen Guo, and Jile Zhou. Collective matrix factorization hashing for multimodal data. In *CVPR*, pages 2083–2090, 2014.

[44] Saehoon Kim, Yoonseop Kang, and Seungjin Choi. Sequential spectral learning to hash with multiple representations. In *ECCV*, pages 538–551, 2012.

[45] Xianglong Liu, Junfeng He, Cheng Deng, and Bo Lang. Collaborative hashing. In *CVPR*, pages 2147–2154, 2014.

[46] Bin Shen and Luo Si. Non-negative matrix factorization clustering on multiple manifolds. In *AAAI*, 2010.

[47] Jay Yagnik, Dennis Strelow, David A. Ross, and Ruei-Sung Lin. The power of comparative reasoning. In *ICCV*, pages 2431–2438, 2011.

[48] Lei Zhang, Yongdong Zhang, Jinhui Tang, Ke Lu, and Qi Tian. Binary code ranking with weighted hamming distance. In *CVPR*, pages 1586–1593, 2013.

[49] Jianfeng Wang, Jingdong Wang, Nenghai Yu, and Shipeng Li. Order preserving hashing for approximate nearest neighbor search. In *ACM Multimedia*, pages 133–142, 2013.

[50] Yi Zhen and Dit-Yan Yeung. Active hashing and its application to image and text retrieval. *Data Mining and Knowledge Discovery*, 26(2):255–274, 2013.

[51] Yunchao Gong and Svetlana Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, pages 817–824, 2011.

[52] Yannis S. Avrithis, Giorgos Tolias, and Yannis Kalantidis. Feature map hashing: sub-linear indexing of appearance and global geometry. In *ACM Multimedia*, pages 231–240, 2010.

[53] Alessandro Bergamo, Lorenzo Torresani, and Andrew W. Fitzgibbon. Picodes: Learning a compact code for novel-category recognition. In *NIPS*, pages 2088–2096, 2011.

[54] Yunchao Gong, Sanjiv Kumar, Vishal Verma, and Svetlana Lazebnik. Angular quantization-based binary codes for fast similarity search. In *NIPS*, pages 1205–1213, 2012.

[55] Jianqiu Ji, Jianmin Li, Shuicheng Yan, Bo Zhang, and Qi Tian. Super-bit locality-sensitive hashing. In *NIPS*, pages 108–116, 2012.

[56] Weihao Kong and Wu-Jun Li. Isotropic hashing. In *NIPS*, pages 1655–1663, 2012.

[57] Yin-Hsi Kuo, Kuan-Ting Chen, Chien-Hsing Chiang, and Winston H. Hsu. Query expansion for hash-based image object retrieval. In *ACM Multimedia*, pages 65–74, 2009.

[58] Mohammad Rastegari, Ali Farhadi, and David A. Forsyth. Attribute discovery via predictable discriminative binary codes. In *ECCV (6)*, pages 876–889, 2012.

[59] Jingkuan Song, Yi Yang, Zi Huang, Heng Tao Shen, and Richang Hong. Multiple feature hashing for real-time large scale near-duplicate video retrieval. In *ACM Multimedia*, pages 423–432, 2011.

[60] Liangliang Cao, Zhenguo Li, Yadong Mu, and Shih-Fu Chang. Submodular video hashing: A unified framework towards video pooling and indexing. In *ACM Multimedia*, pages 299–308, 2012.

[61] Guangnan Ye, Dong Liu, Jun Wang, and Shih-Fu Chang. Large scale video hashing via structure learning. In *ICCV*, 2013.

[62] Xiaofeng Zhu, Zi Huang, Heng Tao Shen, and Xin Zhao. Linear cross-modal hashing for efficient multimedia search. In *ACM Multimedia*, pages 143–152, 2013.

[63] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Sequential projection learning for hashing with compact codes. In *ICML*, pages 1127–1134, 2010.

[64] Benno Stein. Principles of hash-based text retrieval. In *SIGIR*, pages 527–534, 2007.

[65] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407, 1990.

[66] Thomas Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, pages 50–57, 1999.

[67] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *FOCS*, pages 459–468, 2006.

[68] Hao Xia, Pengcheng Wu, Steven C. H. Hoi, and Rong Jin. Boosting multi-kernel locality-sensitive hashing for scalable image retrieval. In *SIGIR*, pages 55–64, 2012.

[69] Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[70] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[71] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In *NIPS*, pages 1601–1608, 2004.

[72] Xianglong Liu, Junfeng He, Bo Lang, and Shih-Fu Chang. Hash bit selection: A unified solution for selection problems in hashing. In *CVPR*, pages 1570–1577, 2013.

[73] Lei Zhang, Yongdong Zhang, Jinhui Tang, Xiaoguang Gu, Jintao Li, and Qi Tian. Topology preserving hashing for similarity search. In *ACM Multimedia*, pages 123–132, 2013.

[74] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. In *CVPR*, pages 731–737, 1997.

[75] Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(6):1092–1104, 2012.

[76] Hanhuai Shan and Arindam Banerjee. Generalized probabilistic matrix factorizations for collaborative filtering. In *ICDM*, pages 1025–1030, 2010.

[77] Chong Wang and David M. Blei. Collaborative topic modeling for recommending scientific articles. In *KDD*, pages 448–456, 2011.

[78] Dell Zhang, Jun Wang, Deng Cai, and Jinsong Lu. Laplacian co-hashing of terms and documents. In *ECIR*, pages 577–580, 2010.

[79] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.

[80] Peter Schonemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

[81] David M. Blei and John D. Lafferty. Topic models. *Text Mining: Theory and Applications*, 2009.

[82] Xing Yi and James Allan. Evaluating topic models for information retrieval. In *CIKM*, pages 1431–1432, 2008.

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

[84] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. NUS-WIDE: A real-world web image database from National University of Singapore. In *CIVR*, 2009.

[85] David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[86] Mark J. Huiskes, Bart Thomee, and Michael S. Lew. New trends and ideas in visual concept detection: the mir flickr retrieval evaluation initiative. In *Multimedia Information Retrieval*, pages 527–536, 2010.

[87] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 42(3):145–175, 2001.

[88] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.

[89] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[90] Ken Lang. Newsweeder: Learning to filter netnews. In *ICML*, pages 331–339, 1995.

[91] Yunchao Gong, Sanjiv Kumar, Henry A. Rowley, and Svetlana Lazebnik. Learning binary codes for high-dimensional data using bilinear projections. In *CVPR*, pages 484–491, 2013.

[92] Mohammad Norouzi, Ali Punjani, and David J. Fleet. Fast search in hamming space with multi-index hashing. In *CVPR*, pages 3108–3115, 2012.

[93] Novi Quadrianto and Christoph H. Lampert. Learning multi-view neighborhood preserving projections. In *ICML*, pages 425–432, 2011.

[94] Qifan Wang, Luo Si, and Dan Zhang. Learning to hash with partial tags: Exploring correlation between tags and hashing bits for large scale image retrieval. In *ECCV*, pages 378–392, 2014.

[95] Mohammad Rastegari, Jonghyun Choi, Shobeir Fakhraei, Daume Hal, and Larry S. Davis. Predictable dual-view hashing. In *ICML (3)*, pages 1328–1336, 2013.

[96] Qifan Wang, Bin Shen, Zhiwei Zhang, and Luo Si. Sparse semantic hashing for efficient large scale similarity search. In *CIKM*, pages 1899–1902, 2014.

[97] Qifan Wang, Bin Shen, Shumiao Wang, Liang Li, and Luo Si. Binary codes embedding for fast image tagging with incomplete labels. In *ECCV*, pages 425–439, 2014.

[98] Qifan Wang, Luo Si, Zhiwei Zhang, and Ning Zhang. Active hashing with joint data example and tag selection. In *SIGIR*, 2014.

[99] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.

[100] Weihao Kong and Wu-Jun Li. Double-bit quantization for hashing. In *AAAI*, 2012.

[101] Xianglong Liu, Junfeng He, and Bo Lang. Reciprocal hash tables for nearest neighbor search. In *AAAI*, 2013.

[102] Cornelia Caragea, Adrian Silvescu, and Prasenjit Mitra. Combining hashing and abstraction in sparse high dimensional feature spaces. In *AAAI*, 2012.

[103] Yadong Mu and Shuicheng Yan. Non-metric locality-sensitive hashing. In *AAAI*, 2010.

[104] Antonio Torralba, Robert Fergus, and William T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970, 2008.

[105] Tao Qin, Tie-Yan Liu, and Hang Li. A general approximation framework for direct optimization of information retrieval measures. *Information Retrieval*, 13(4):375–397, 2010.

[106] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, and Tie-Yan Liu. A theoretical analysis of NDCG type ranking measures. In *COLT*, pages 25–54, 2013.

[107] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for SVM. *Mathematical Programming*, 127(1):3–30, 2011.

[108] Hamed Valizadegan, Rong Jin, Ruofei Zhang, and Jianchang Mao. Learning to rank by optimizing NDCG measure. In *NIPS*, pages 1883–1891, 2009.

[109] Markus Weimer, Alexandros Karatzoglou, Quoc V. Le, and Alex J. Smola. COFI RANK-Maximum margin matrix factorization for collaborative ranking. In *NIPS*, pages 1593–1600, 2007.

[110] Nilesh N. Dalvi, Vibhor Rastogi, Anirban Dasgupta, Anish Das Sarma, and Tamás Sarlós. Optimal hashing schemes for entity matching. In *WWW*, pages 295–306, 2013.

[111] Weihao Kong, Wu-Jun Li, and Minyi Guo. Manhattan hashing for large-scale image retrieval. In *SIGIR*, pages 45–54, 2012.

[112] Qifan Wang, Lingyun Ruan, Zhiwei Zhang, and Luo Si. Learning compact hashing codes for efficient tag completion and prediction. In *CIKM*, pages 1789–1794, 2013.

[113] Qifan Wang, Dan Zhang, and Luo Si. Weighted hashing for fast large scale similarity search. In *CIKM*, pages 1185–1188, 2013.

[114] Yuhong Guo and Dale Schuurmans. Discriminative batch mode active learning. In *NIPS*, 2007.

[115] Steven C. H. Hoi, Rong Jin, Jianke Zhu, and Michael R. Lyu. Batch mode active learning and its application to medical image classification. In *ICML*, pages 417–424, 2006.

[116] Aibo Tian and Matthew Lease. Active learning to maximize accuracy vs. effort in interactive information retrieval. In *SIGIR*, pages 145–154, 2011.

[117] David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *SIGIR*, pages 3–12, 1994.

[118] Burr Settles. Active learning literature survey. Technical report, University of Wisconsin–Madison, 2010.

[119] Pinar Donmez and Jaime G. Carbonell. Active sampling for rank learning via optimizing the area under the roc curve. In *ECIR*, pages 78–89, 2009.

[120] Bo Long, Olivier Chapelle, Ya Zhang, Yi Chang, Zhaohui Zheng, and Belle L. Tseng. Active learning for ranking through expected loss optimization. In *SIGIR*, pages 267–274, 2010.

[121] Peng Cai, Wei Gao, Aoying Zhou, and Kam-Fai Wong. Relevant knowledge helps in choosing right teacher: active query selection for ranking adaptation. In *SIGIR*, pages 115–124, 2011.

[122] Zheng Wang and Jieping Ye. Querying discriminative and representative samples for batch mode active learning. In *KDD*, pages 158–166, 2013.

[123] Abhay Harpale and Yiming Yang. Personalized active learning for collaborative filtering. In *SIGIR*, pages 91–98, 2008.

[124] Rong Jin and Luo Si. A bayesian approach toward active learning for collaborative filtering. In *UAI*, pages 278–285, 2004.

[125] Vladimir Vapnik. *The nature of statistical learning theory.* Springer Verlag, 2000.

[126] Mokhtar S. Bazaraa, Hanif D. Sherali, and C. M. Shetty. *Nonlinear programming: theory and algorithms (3. ed.).* Wiley, 2006.

[127] Choon Hui Teo, S. V. N. Vishwanathan, Alex J. Smola, and Quoc V. Le. Bundle methods for regularized risk minimization. *Journal of Machine Learning Research*, 11:311–365, 2010.

[128] Mark D. Smucker, James Allan, and Ben Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *CIKM*, pages 623–632, 2007.

VITA

VITA

Qifan Wang received his PhD degree from the Department of Computer Science, Purdue University under the supervision of Prof. Luo Si. Before this, he got his MS degree from the Computer Science Department, Tsinghua University, advised by Prof. Linmi Tao. During the summers of 2008 and 2009, he worked as a research intern at Intel Labs, Beijing. He also interned at LinkedIn and Google Research during the summers of 2012 and 2013. His research interests include mainly machine learning, information retrieval, data mining and computer vision. He has published papers in several top conferences, such as AAAI, SIGIR, IJCAI, ECCV, CIKM, DSN, etc, and journals, such as TPAMI, TOIS etc.