

國立臺灣大學電機資訊學院資訊網路與多媒體研究所



博士論文

Graduate Institute of Networking and Multimedia
College of Electrical Engineering and Computer Science

National Taiwan University

Doctoral Dissertation

以複合式模型學習法探究多社群網路媒體之使用者資訊

Multi-Modal Learning over User-Contributed Content from

Cross-Domain Social Media

李文瑜

Wen-Yu Lee

指導教授：徐宏民 博士

Advisor: Winston H. Hsu, Ph.D.

中華民國 105 年 4 月

April 2016



國立臺灣大學博士學位論文
口試委員會審定書

以複合式模型學習法探究多社群網路媒體之使用者資訊
Multi-Model Learning over User-Contributed Content from
Cross-Domain Social Media

本論文係李文瑜君（學號 D99944007）在國立臺灣大學資訊網路與多媒體研究所完成之博士學位論文，於民國一百零五年四月十五日承下列考試委員審查通過及口試及格，特此證明

口試委員：



（簽名）

陳祝嵩
（指導教授）

陳信辛

陳文進

葉梅珍

余斛定

所長：

逢愛君



Acknowledgements



First, I would like to express my sincere gratitude to my advisor, Professor Winston H. Hsu, whose expertise, guidance, and patience, raised considerably to my graduate experience. Without his precious support and guidance, it would not be possible to conduct this research.

Besides my advisor, I greatly appreciate the rest of my dissertation committee, Professor Chu-Song Chen, Professor Hsin-Hsi Chen, Professor Wen-Chin Chen, Professor Mei-Chen Yeh, and Professor Neng-Hao Yu, for their invaluable comments and suggestions.

My sincere appreciation also goes to Professor Kiyoharu Aizawa, who provided me an opportunity to join his team during a summer research program, and Professor Shin'ichi Satoh, who provided me an opportunity to join his team during an international internship program. I thank the Interchange Association of Japan and the National Institute of Informatics of Japan, for the aforementioned two programs they organized, respectively.

I also acknowledge all members of the Communications and Multimedia Laboratory in National Taiwan University for many useful discussions, comments, friendship, and constant encouragement. In particular, I would like to express my appreciation to Yin-Hsi Kuo for her great help on publications.

Finally, my deepest appreciation is dedicated to my parents. Without their constant support and endless love, the completion of this dissertation would not have been possible.

Wen-Yu Lee

National Taiwan University

April 2016



以複合式模型學習法探究多社群網路媒體之 使用者資訊



學生：李文瑜 指導教授：徐宏民

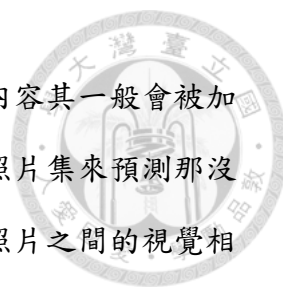
國立臺灣大學資訊網路與多媒體研究所

摘要

近年來，社群網路的蓬勃發展逐漸地改變了人們的生活，越來越多人習慣在社群網站上分享生活的點滴；據統計，平均每一天有數百萬的多媒體資料，例如照片，被上傳至社群網站，而這樣龐大數目的多媒體資料裡極可能蘊藏著豐富的資訊；在這篇論文裡，我們期望能從大量的多媒體資料裡萃取出有用的資訊，並藉此解決人們日常生活裡常見的一些問題。

從一張照片出發，我們期望建立一套資訊系統，可以找出這張照片是在哪裡拍攝的，其他人可能會對這照片給予怎樣的文字標籤來描述這拍攝的內容，又這張照片的拍攝地點附近是否有熱門的活動或事件；這樣的資訊，在日常生活裡可以有許多應用，例如在旅遊的時候，我們常會想知道自己在哪裡、眼前的事物是什麼，又或者在這附近平常會有什麼樣的活動或事件；這時只要隨手地對眼前事物拍一張照，再將這照片送入此資訊系統，系統就可以快速地提供我們這些資訊。

過去找出照片拍攝地點的常見作法是利用照片的視覺特徵與地理標籤，透過與現有的資料比對，以推測出照片最可能的拍攝地點；但當照片是在室內被拍攝或是拍攝地點有多個建築物時，可能會降低這類方法的準確度；為此，此論文進一步地加入了打卡資料來輔助定位，同時提出一個有效的影像重新分群技術，來



提升推測的照片拍攝地點的準確度。而對於找出照片就拍攝的內容其一般會被加註什麼樣的文字標籤，常見的技巧是利用已有文字標籤的現有照片集來預測那沒有標籤的照片其可能的文字標籤；一種有效作法是根據照片與照片之間的視覺相似度來建構圖，這圖是以照片作為節點，相似度作為連接節點與節點之間的邊的權重，之後考慮到邊的權重，將標籤自節點傳遞分享至沒有標籤的照片的節點，這種作法即使是當沒有標籤的照片數目大於有標籤的照片數目仍經常有效；而考慮到過去的方法大多是一次只傳遞單一個標籤，對於多個標籤的情況下則需要進行多次，並且現有照片的數目一直以來持續地在增長，方法的效率逐漸成為一個重要的議題；有鑑於此，此論文提出了一個基於分散式運算原理的多標籤傳遞法，來有效率地同時傳遞多個標籤。對於找出熱門活動，過往的方法多專注於觀察單一的多媒體資料集的資料變化，例如字詞的談論頻率變化，異常的變化則表示可能有活動或事件發生；而此論文考慮到不同資料集其性質都有所不同，彼此之間可能可以互補不足的地方，來提出了一個有效的兩階段式架構，能有效地結合資料流類型的資料集及打卡類型的資料集，以找出熱門的活動或事件的發生時間、發生地點及其內容資訊；由於已能找出照片可能的拍攝地點，則可以透過比較可能的拍攝地點和熱門活動或事件的發生地點來找出照片拍攝地點附近的熱門活動或事件。

此論文所提的方法都已透過實驗分析來顯示這些方法的有效性。最後，此論文提出一些未來可能的研究方向。

關鍵字：事件挖掘、影像標註、影像地點辨識、複合式模型學習、社群網路媒體、使用者資訊

MULTI-MODAL LEARNING OVER USER-CONTRIBUTED CONTENT FROM CROSS-DOMAIN SOCIAL MEDIA



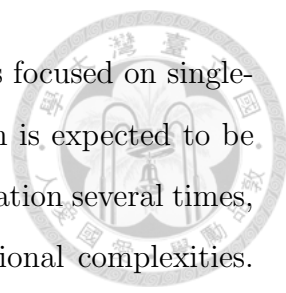
Student: Wen-Yu Lee Advisor: Dr. Winston H. Hsu

Graduate Institute of Networking and Multimedia
National Taiwan University

Abstract

Social media have changed the world and our lives. Every day, millions of media data are uploaded to social-sharing websites. The goal of the research is to discover and summarize large amounts of media data from the emerging social media into information of interests. Our basic idea is to perform multi-modal learning for given data, leveraging user-contributed data from cross-domain social media. Specifically, given a photo, we intend to discover geographical information, people's description or comments, and events of interest, closely related to the photo. These information then can be used for various purposes, such as being a real-time guide for the tourists to improve the quality of tourism. As a result, this dissertation studies modern challenges of image location identification, image annotation, and event discovery, followed by presenting promising ways to conquer the challenges.

For image location identification, most previous works directly integrated visual features and geo-tags of the given photos. The performance of the existing approaches, however, could be limited if the given photos were taken indoors, and/or their image contents contain a number of buildings in a close proximity. As a solution, this dissertation unifies visual features, geo-tags, and check-in data, and further presents an image cluster refinement approach, for image location identification. For image annotation, label propagation is widely used to annotate photos



based on similarity graphs of photos, where most previous works focused on single-label propagation. Although performing multi-label propagation is expected to be more efficient for annotation than performing single-label propagation several times, performing multi-label propagation may increase the computational complexities. Further, sizes of image datasets continue to increase and thus increase the problem complexity. As a solution, this dissertation presents a scalable multi-label propagation leveraging the power of distributed computing. For event discovery, most previous works investigated a specific media stream. Potentially, mining multiple media streams is capable of achieving better performance than mining a media stream alone, but could be more challenging. As a solution, this dissertation presents a two-stage framework that combines a flow-based media dataset and check-in-based media dataset for events-of-interest discovery.

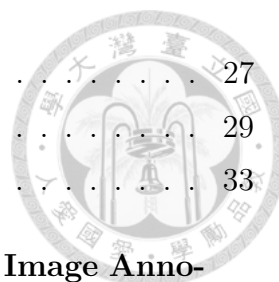
Experimental results on real media datasets show the effectiveness of all of the proposed approaches. Finally, this dissertation provides some possible directions for future studies.

Keywords: Event Discovery, Image Annotation, Image Location Identification, Multi-Modal Learning, Social Media, User-Contributed Content

Table of Contents



Acknowledgements	v
Abstract (Chinese)	vii
Abstract	ix
List of Tables	xv
List of Figures	xvii
Chapter 1. Introduction	1
1.1 Modern Challenges	3
1.1.1 Image Location Identification	3
1.1.2 Image Annotation	4
1.1.3 Event Discovery	5
1.2 Overview of the Dissertation	6
1.2.1 Geo-Social Media Mining for Image Location Identification	6
1.2.2 Graph-based Semi-Supervised Learning for Image Annotation	7
1.2.3 Cross-Domain Media Learning for Event Discovery	7
1.3 Organization of the Dissertation	8
Chapter 2. Geo-Social Media Mining for Image Location Identification	9
2.1 Introduction	9
2.2 City-View Image Location Identification	12
2.2.1 Data Collection	13
2.2.2 Location Identification	17
2.3 Macro-Cluster Regrouping	18
2.3.1 A Naive Approach	19
2.3.2 Regrouping by Proximity	20
2.4 Sparse Coding for Macro Clusters	23
2.4.1 Formulation	23
2.4.2 Dictionary Selection within a Macro Cluster	25



2.5	Experiments	27
2.5.1	Results with Manual-Annotated Ground Truth	29
2.5.2	Results with Automatic-Annotated Ground Truth	33

Chapter 3. Graph-based Semi-Supervised Learning for Image Annotation 35

3.1	Introduction	35
3.2	Learning System Overview	37
3.3	Methodologies	39
3.3.1	Multi-Layer Learning Structure	39
3.3.2	Multi-Layer Multi-label Propagation Algorithm	41
3.3.3	Convergence Criterion of Label Propagation	43
3.3.4	Tag Refinement	45
3.4	Experiments	49
3.4.1	Experimental Setup	49
3.4.2	Evaluation Criteria	50
3.4.3	Multi-Layer Multi-Label Propagation	51
3.4.4	Tag Refinement	51
3.4.5	Convergence Criterion - Shortest Path	52
3.4.6	Sensitivity Test	53

Chapter 4. Cross-Domain Media Learning for Event Discovery 55

4.1	Introduction	55
4.2	Problem Formulation	58
4.3	Proposed Approach	60
4.3.1	Flow Normalization	62
4.3.2	Buzz-Score Calculation	63
4.3.3	Variability-Score Calculation	64
4.3.4	Spanning-Graph Construction	66
4.3.5	Weight Determination	69
4.3.6	Flow Propagation	70
4.4	Experiments	72
4.4.1	Experimental Setup	73
4.4.2	Cross-Dataset Analysis	73
4.4.3	Evaluation of the Proposed Approach	75

Chapter 5. Conclusion and Future Work

5.1 Conclusion 79

5.2 Future Work 80

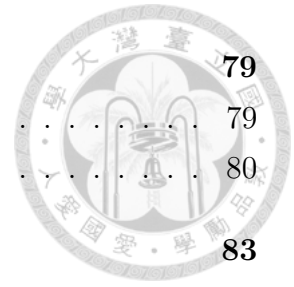
Bibliography

79

79

80

83

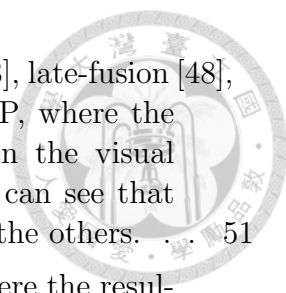






List of Tables

1.1	Pairwise comparison of percentages of overlapped attractions, <i>i.e.</i> , similarity, over Twitter, Instagram, TripAdvisor, and subway traffics, in our preliminary study [27]. As can be seen, their popular places are different to some extent. It is desirable to unify cross-domain media for high diversity.	6
2.1	Comparison between visual-based, location-based, visual and location-based, and our approach, on the Foursquare-based image dataset of San Francisco, and on the Foursquare and Flickr-based image dataset of San Francisco. Note that “Ours” combines visual features, GPS locations, and check-in data.	30
2.2	Comparison between visual-based, location-based, visual and location-based, and our approach, on the Foursquare-based image dataset of Manhattan, and on the Foursquare and Flickr-based image dataset of Manhattan. Note that “Ours” combines visual features, GPS locations, and check-in data.	30
2.3	Comparison between our overall approach and its two variants for the effects of using our regrouping approach on search performance. The two variants include our approach without macro-cluster regrouping and our approach with simplified regrouping (<i>i.e.</i> , removing ambiguous images in macro clusters). Note that “Ours” uses macro-cluster regrouping.	32
2.4	Comparison between our overall approach and its variant for the effect of using the proposed dictionary selection approach (see Section 2.4.2) or not. “VC” denotes the vertex-cover-based approach, which is the proposed dictionary selection approach. Note that “Ours” uses the dictionary selection approach.	32
2.5	Comparison between visual-based, location-based, visual and location-based, and our approach. Note that here the same dataset as that for Tables 2.3 and 2.4 was used, but this experiment was based on automatic annotation (the prior experiments were based on manual-annotated ground truth). We did not show “Ours”, but “Ours w/o regrouping” because automatic annotation assumes that Foursquare provides the golden image clustering results, while the motivation of regrouping is that the clustering results might not be the best at all aspects.	34



3.1 Compare the visual graph, textual graph, early-fusion [48], late-fusion [48], and multi-layer (ours) -based learning methods in MAP, where the percentages (%'s) are the improvement ratios between the visual graph only and the multi-layer learning methods. We can see that the multi-label method can achieve better results than the others. 51

3.2 Compare the use of tag refinement and not in MAP, where the resultant top-1, 2, 3, 4, 5, and 10 labels are considered. 52

3.3 Compare the shortest path-based approach and an ad-hoc approach in MAP. Based on the results, we can see that the shortest path-based results are comparable to those by the ad-hoc approach. 53

3.4 Consider the setting of α (*cf.* Eq. (3.1)), *i.e.*, the inferring parameter in the graph-based multi-layer multi-label propagation in MAP. 53

3.5 Consider the setting of β (*cf.* Eq. (3.2)), *i.e.*, the fusion parameter in the graph-based multi-layer multi-label propagation in MAP. 54

3.6 Consider the setting of γ (*cf.* Eq. (3.4)), *i.e.*, the balance control parameter in tag refinement in MAP. 54

4.1 Comparison of a flow dataset of taxis and an Instagram's check-in dataset for the City of Sapporo. 56

4.2 Illustration of a ranked list of events, where "Cmnt" is an abbreviation of "Comment." Two items (*i.e.*, rows) are listed, where each item is associated to an event. Each item gives the date, the location, the relevant terms, and users' comments which the relevant terms are derived from, for the event. 60

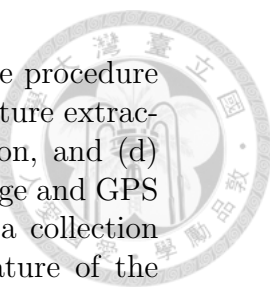
4.3 Illustration of an original ranked list of events, where "Cmnt" is an abbreviation of "Comment." Items of the list will be merged together if they are for the same date and the same location. 72

4.4 Comparison of the results of no data normalization and the results of using data normalization. Note that "Ours w/o Normalization" did not use flow normalization and variability-score calculation, but buzz-score calculation. 78



List of Figures

1.1	Illustration of the desired social-media mining system, including the components of image location identification, image annotation, and event discovery.	2
1.2	Illustration of challenges of city-view image location identification. (a) Photos that cover a small part of a building. (b) Photos that were taken in the morning and in the evening, respectively. (c) Photos that were taken indoors and outdoors, respectively. (d) Photos that show several stores being in a close proximity. These conditions may degrade the performance of existing techniques that are based on visual features or geo-tags of photos.	4
2.1	Distribution of places of interest retrieved from Yahoo! Travel [5] for (a) San Francisco and (b) Manhattan, on the Google Map [3]. Each marker is associated to a place. It is apparent that many places are in a very close proximity. Distinguishing a building in a city from the others may be challenging for traditional techniques.	10
2.2	Overall flow of our data collection procedure. The procedure is mainly composed of five steps, including (a) macro-cluster generation, (b) macro-cluster expansion, (c) macro-cluster regrouping, (d) visual feature extraction, and (e) sparse coding. Given a name list of place of interest, we first generate a set of macro clusters, where each macro cluster consists of a text-tag, a geo-tag, and a set of images. For better diversity and performance, we then add images from Flickr to each of the macro clusters. (This step can easily be extended to consider other media streams for cross-domain media combination.) Afterwards, the set of macro clusters are regrouped for better search performance. We then extract visual features of all images in the macro clusters. Finally, sparse coding is used to create sparse vectors and a dictionary for each macro cluster, based on the visual features of images of the macro cluster. (Note that we do not create sparse vectors for all images at once, but cluster-by-cluster, aiming to find appropriate vectors and dictionaries for the macro clusters of images.)	14



2.3 Overall flow of our location identification procedure. The procedure is mainly composed of four steps, including (a) visual feature extraction, (b) candidate selection, (c) sparse vector generation, and (d) image ranking. We are given a query consisting of an image and GPS location, and the macro clusters obtained from the data collection procedure (Figure 2.2). We first extract the visual feature of the query image, followed by removing the macro clusters that are far from the GPS location. Note that each macro cluster has its own dictionary, and two sparse vectors can make a fair comparison only if they were derived from the same dictionary. We then generate sparse vectors for the query image, one for each of the remaining macro clusters. Finally, we will generate a ranked list for the images in the remaining macro clusters. 16

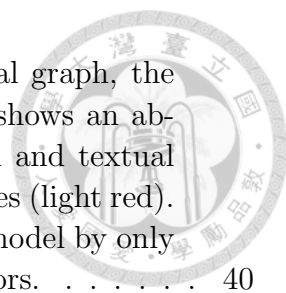
2.4 Illustration of constructing a spanning graph for four macro clusters, C_1, C_2, C_3 , and C_4 , in a plane. (a) The macro clusters are initially disconnected. (b) Assume that C_1 is first visited. The plane is divided into eight regions with respect to C_1 , and C_1 is connected to the macro cluster that is the nearest one of C_1 in each region, *i.e.*, C_2 and C_4 . (c) The resultant spanning graph for C_1, C_2, C_3 , and C_4 , after the four macro clusters have been visited. (Details of constructing a spanning graph can be found in Section 2.3.2.) With the spanning graph, images of a macro cluster will not consider macro clusters which are not adjacent to it for regrouping, because the graph is expected to find critical neighborhoods for these macro clusters. 22

2.5 Algorithm of the proposed graph-based regrouping process. Given macro clusters, the process moves images to their desired macro clusters. 24

2.6 Algorithm of finding desired cluster for the proposed graph-based regrouping process. Given a spanning graph, a macro cluster, and an image, the process finds the desired macro cluster with the associated similarity between the image and a micro cluster of the macro cluster for the image. 25

2.7 Illustration of our graph-based selection approach: (a) consider a macro cluster with images; (b) construct a visual similarity graph based on the visual similarities of images; (c) remove the edges whose weights are too small; (d) remove images while retaining representative images by solving the vertex cover problem. Finally, the images associated to the vertices of the resulting graph are used to create a dictionary. (*cf.* Section 2.4.2.) 28

3.1 Illustration of our graph-based SSL system. Given an image dataset, we first perform feature extraction to obtain both visual and textual features. Then a sparse visual graph and a sparse textual graph are constructed to enable multi-layer label propagation. Based on the two graphs, we then perform multi-layer multi-label propagation. Finally, we apply tag refinement to improve the tag results. 38



3.2 Multi-layer learning structure. The top layer is a visual graph, the bottom layer is a textual graph, and the middle layer shows an abstract fusion layer for the communication of the visual and textual graphs. Note that initial labels are shown in double circles (light red). Graph structure can also benefit from the MapReduce model by only propagating and receiving information from the neighbors. 40

3.3 The first stage of an inferring process with MapReduce. 44

3.4 The second stage of an inferring process with MapReduce. 45

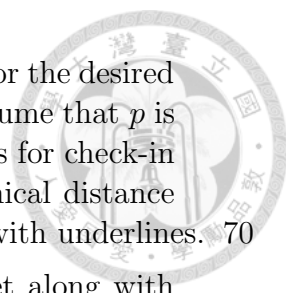
3.5 Illustration of tag refinement. After tag refinement, the tag order of a tag list is changed while the number of tags is the same. A tag in the tag list of an image precedes if it is more relevant to the image than the tag(s) placed behind. 47

4.1 Illustration of two items listed in the original flow dataset of taxis, and two items in the original Instagram’s dataset that are associated to Sapporo Dome. The two datasets can be processed for a flow dataset with a set of flow data, and a check-in dataset with a set of check-in data. 59

4.2 Overall flow of the proposed approach for finding events of interest. The framework of the proposed approach has two stages, including data normalization and graph-based data fusion. Note that before data fusion, data normalization is essential because the given datasets are different in nature. Specifically, the approach uses six techniques, including (a) flow normalization, (b) buzz-score calculation, (c) variability-score calculation, (d) spanning-graph construction, (e) weight determination, and (f) flow propagation. For these techniques, the first three are used for data normalization, and the other three are used for data fusion. Overall, the given datasets will be normalized individually, and then combined together by graph-based techniques. 61

4.3 Illustration of spanning-graph construction. Consider a set of vertices: v_1, v_2, v_3 , and v_4 . (a) Assume that v_1 is first visited. We will divide the plane evenly into eight regions with respect to v_1 . We then add an edge between v_1 and v_2 , and an edge between v_1 and v_3 , because each of v_2 and v_3 is the nearest vertex to v_1 in one of the regions. (b) The spanning graph for the given set of vertices. 67

4.4 Illustration of spanning-graph construction for the two datasets. (a) The resultant graph after the vertices associated to the flow-based dataset have been traversed. (Assume that the vertices associated to the check-in-based dataset have not been traversed.) (b) The resultant spanning graph after all of the vertices have been traversed. . . . 68



4.5 Illustration of assigning values for edges (partial values for the desired weights) that are adjacent to a vertex for flow data. Assume that p is a vertex for flow data, and v_1, v_2 , and v_3 are the vertices for check-in data that are adjacent to p . If $g(,)$ gives the geographical distance for two vertices, the assigned values will be the values with underlines. 70

4.6 Change of coverage ratios from the Instagram’s dataset along with changes of percentages of boarding and alighting locations for the taxi’s dataset, based on the popularity. Most of the popular taxi’s locations are close to some Instagram’s locations, but the ratio decreases as more and more non-popular taxi’s locations are involved. . 74

4.7 Change of coverage ratios from the taxi’s dataset along with changes of percentages of check-in locations for the Instagram’s dataset, based on the popularity. Most of the popular Instagram’s locations are close to some taxi’s locations, but the ratio decreases as more and more non-popular Instagram’s locations are involved. 75

4.8 Comparison of the results of using the Instagram’s dataset alone and the results of using both the taxi’s dataset and the Instagram’s dataset. 76

4.9 Compares the results of using the k -nearest neighbors (k NN) algorithm for graph construction and the results of using spanning graphs (*c.f.*, Section 4.3.4) for our work. 77



Chapter 1

Introduction

Every day, millions of media data are uploaded to various social-sharing websites. For example, on average, more than 300 million images are uploaded to Facebook in a day [16]. It is desirable to discover what the large number of data can be used to make our lives better. In our daily life, “where am I?” and “what is this?” and are two common questions that arise when people go out. Earlier, it could be troublesome or even hard to find the answers, especially when there is a language barrier, *e.g.*, when travelling abroad. Moreover, people are often like to explore more about the places they are visiting. “Are there special events or festivals?” “What happened?” It is desirable to discover events of interest for a specific region.

As a solution, this dissertation attempts to build an effective social-media mining system for these questions, by developing effective techniques of (1) image location identification, (2) image annotation, and (3) event discovery, considering modern challenges. Figure 1.1 illustrates the idea. Ideally, people may upload a photo onto the system. The system then performs social-media information retrieval based on the photo. Specifically, the system is expected to generate geo-tags that indicates where the photo might be taken by image location identification, textual tags that reflect the image content of the photo by image annotation, and an event list that gives events of interest by event discovery. Although image location identification, image annotation, and event discovery, are classical problems, modern challenges have reshaped the problems.

This chapter is organized as follows. Section 1.1 introduces modern challenges of image location identification, image annotation, and event discovery. Section 1.2 gives an overview of the dissertation. Finally, Section 1.3 details the organization

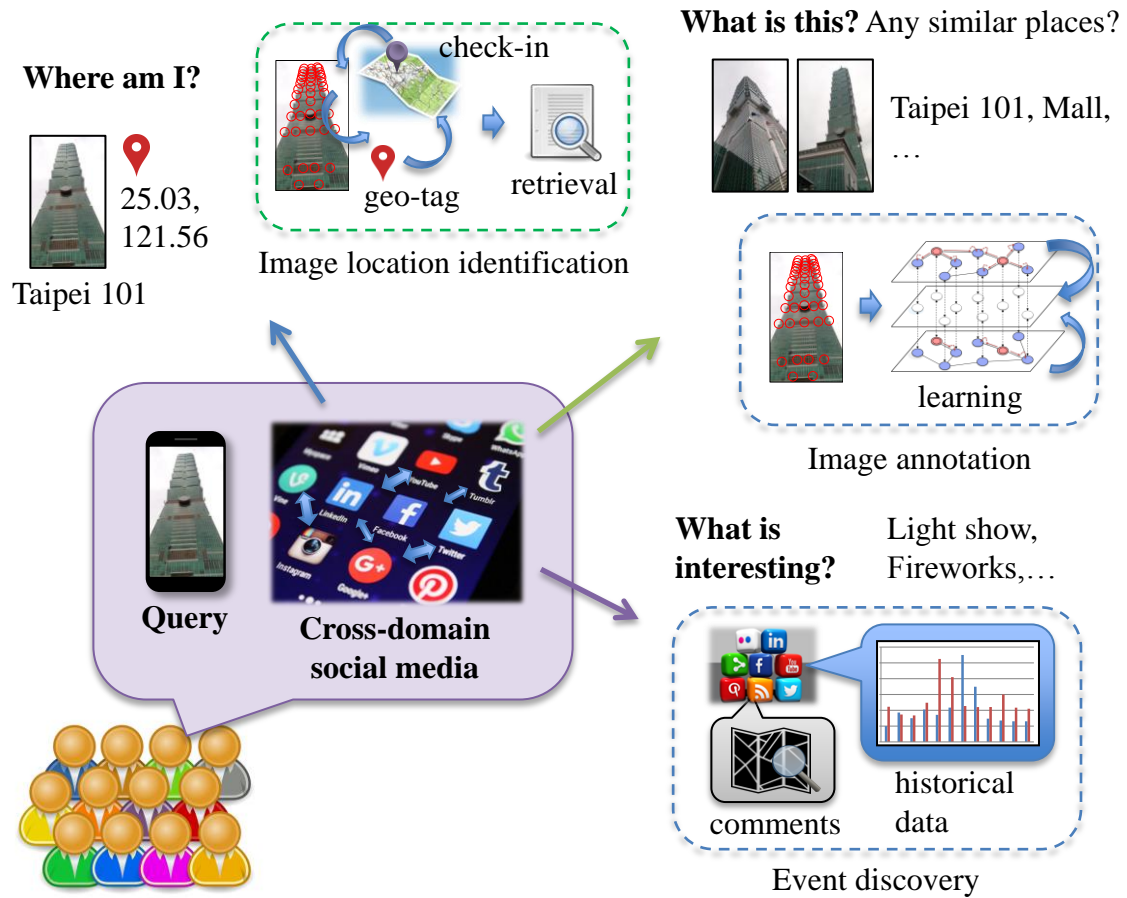


Figure 1.1: Illustration of the desired social-media mining system, including the components of image location identification, image annotation, and event discovery.

of the dissertation.



1.1 Modern Challenges

This section presents challenges for (1) image location identification, (2) image annotation, and (3) event discovery.

1.1.1 Image Location Identification

Image location identification aims to find where a given photo was taken. Many studies for image location identification focus on landmark identification. Given a photo, previous works for landmark identification typically matches the photo with photos of landmarks using features and/or tags of these photos. Afterwards, the text- and geo-tags of the photo of landmark that is the most similar to the given photo will indicate what landmark the photo was taken for and where the landmark is located at (thus, where the photo might be taken), respectively. Consider the phenomenon that people may take a photo of anything anywhere at any moment. In contrast to landmark identification, this dissertation focuses on city-view image location identification.

City-view image location identification is challenging mainly because of four conditions: (1) a photo may cover only a small part of the target object; (2) a photo may be taken under different operating conditions, such as weather conditions and shot sizes; (3) photos for a building may be taken indoors or outdoors; (4) there could be a number of buildings in a very close proximity. See Figure 1.2 as an illustration¹. Among the above four, conditions (1)–(3) and (3)–(4) may degrade the performance of existing techniques based on visual features and geo-tags of photos, respectively. In particular, conditions (3)–(4) are more critical to city-view image location identification than traditional landmark identification. According to a study of the positional accuracy from mobile phones, locations provided by a

¹All of the photos shown in this figure were obtained from Foursquare [2].



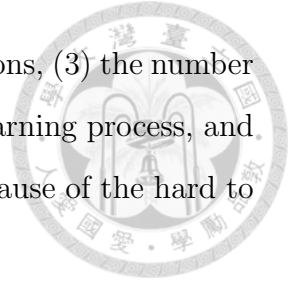
Figure 1.2: Illustration of challenges of city-view image location identification. (a) Photos that cover a small part of a building. (b) Photos that were taken in the morning and in the evening, respectively. (c) Photos that were taken indoors and outdoors, respectively. (d) Photos that show several stores being in a close proximity. These conditions may degrade the performance of existing techniques that are based on visual features or geo-tags of photos.

mobile phone may have a root mean square error (RMSE) of 12.5 or 21.6 meters depending on that the phone is used outdoors or indoors [55]. Sometimes, an error of 47.9 meters might occur when a phone is used indoors [55]. Thus, previous works such as [28] that directly integrated visual features and geo-tags of photos may have a limitation on distinguishing a building within a city from the others, if the given photos were taken indoors and/or these buildings are in a close proximity.

1.1.2 Image Annotation

Image annotation aims to add proper tags for the images of a given image dataset. Image annotation is challenging mainly because of four reasons: (1) there are relatively few images having tags in a general image dataset, *i.e.*, the resource is strictly limited, (2) an image dataset may have two or more similar images of the

same landscape but with different tags from different interpretations, (3) the number of images continues to grow rapidly, and thus complicates the learning process, and (4) traditional supervised approaches might not be practical because of the hard to form a model for every tag.



As a solution, semi-supervised learning (SSL) is widely-used to realize image annotation. SSL is a learning technique to explore lots of untagged images in the presence of a small amount of tagged images. Among the SSL approaches, graph-based approaches are quite popular due to their higher efficiencies in contrast to other approaches [60]. A typical graph-based approach models both tagged and untagged images as vertices followed by adding a weighted edge between each pair of vertices, where the weight of an edge is the similarity between its two terminal vertices (*i.e.*, images). With the graph, a process called label propagation is activated to add tags to images (*i.e.*, to label images) accordingly.

Traditional label propagation implies *single-label* propagation, where each image considers only a single tag to reduce the complexity on label propagation. Recently, the study of annotation of multiple tags arises [51]. Moreover, sizes of image datasets continue to increase. It is desirable to develop a scalable approach for multi-label propagation.

1.1.3 Event Discovery

Event discovery aims to discover events of interest. Generally, event discovery can be achieved by observing the changes of data numbers from a given media stream along a period of time. Intuitively, every media stream has unique information and its own features. As a case study, an experiment was conducted in our preliminary work [27] that compared the similarity of top-100 popular places over different media streams, including Twitter, Instagram, TripAdvisor, and a NYC open data for subway traffics. The result can be found in Table 1.1. As can be seen, the popular places of the media streams are diverse. Unifying different media streams is capable of achieving better diversity and even performance than using

Table 1.1: Pairwise comparison of percentages of overlapped attractions, *i.e.*, similarity, over Twitter, Instagram, TripAdvisor, and subway traffics, in our preliminary study [27]. As can be seen, their popular places are different to some extent. It is desirable to unify cross-domain media for high diversity.

	Twitter	Instagram	TripAdvisor	Subway
Twitter	1.00	0.35	0.08	0.22
Instagram	0.35	1.00	0.18	0.43
Tripadvisor	0.08	0.18	1.00	0.32
Subway	0.22	0.43	0.32	1.00

one media stream alone. However, combining different media streams is also what makes event discovery on cross-domain media challenging, because the meanings of their data, even only for numerical data, may be quite different.

1.2 Overview of the Dissertation

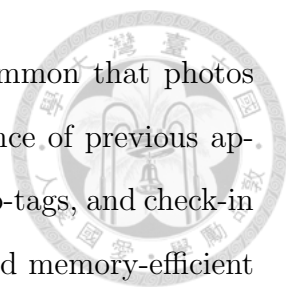
To confront these challenges, we developed three approaches, including (1) a geo-social media-based approach with graph-based image cluster refinement for image location identification, (2) a graph-based multi-label propagation approach with distributed computing for image annotation, and (3) a cross-domain media learning approach with data normalization and data fusion for event discovery. We will detail the three approaches in the following three chapters.

- Chapter 2: Geo-Social Media Mining for Image Location Identification
- Chapter 3: Graph-based Semi-Supervised Learning for Image Annotation
- Chapter 4: Cross-Domain Media Learning for Event Discovery

Below gives the abstracts of the three chapters.

1.2.1 Geo-Social Media Mining for Image Location Identification

Recently, landmark identification has shown great promise for image location identification, where most previous approaches are either visual-based or location-based. Regarding city-view image location identification, however, there could be



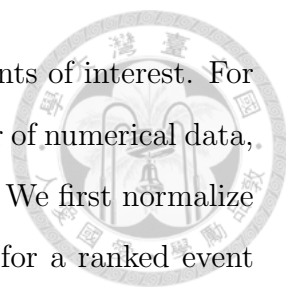
a number of buildings in a close proximity. Moreover, it is common that photos were taken indoors. The conditions may degrade the performance of previous approaches. To remedy the deficiencies, we unify visual features, geo-tags, and check-in data, based on geo-social media. Besides, we use an effective and memory-efficient implementation by sparse coding, where we propose a new dictionary selection approach. Further, we propose a location-aware graph-based regrouping approach, leveraging spanning graph construction, on clusters of photos to refine clustering results. Experimental results show the improvement over the baselines (location-based, visual-based, etc.)

1.2.2 Graph-based Semi-Supervised Learning for Image Annotation

Over the decade, graph-based SSL becomes popular in automatic image annotation due to its power of learning globally based on local similarity. However, recent studies have shown that the emergence of large-scale datasets challenges the traditional approaches. On the other hand, most previous works have concentrated on single-label annotation, which may not describe image contents well. To remedy the deficiencies, we propose a new graph-based SSL technique with multi-label propagation, leveraging the distributed computing power of the MapReduce programming model. For high learning performance, we further use both a multi-layer learning structure and a tag refinement approach, where the former unifies both visual and textual information of image data during learning, while the latter simultaneously suppresses noisy tags and emphasizes the other tags after learning. Experimental results based on a medium-scale and a large-scale image datasets show the effectiveness of the proposed approach.

1.2.3 Cross-Domain Media Learning for Event Discovery

Different media streams have their own features and advantages. Combining different datasets is capable of achieving better performance than using any of them alone. To the best of our knowledge, we present the first work that unifies a flow-



based media dataset and a check-in-based media dataset for events of interest. For the two datasets, the former offers real-time and sufficient number of numerical data, while the latter offers textual data and accurate event locations. We first normalize the two domains of media datasets, followed by combing them for a ranked event list leveraging graph-based algorithms. Experimental results show the effectiveness of our work, based on a flow dataset of taxis and the Instagram's check-in dataset for the City of Sapporo.

1.3 Organization of the Dissertation

The rest of the dissertation is organized as follows. Chapter 2 presents the image location identification approach. Chapter 3 presents the image annotation approach. Chapter 4 presents the event discovery approach. Finally, Chapter 5 concludes the dissertation and presents future research directions.



Chapter 2

Geo-Social Media Mining for Image Location Identification

2.1 Introduction

Image location identification aims to find where a given photo was taken. This chapter studies the problem of city-view image location identification. In contrast to traditional landmark identification, city-view image location identification is challenging because of two conditions: (1) the given photos may be taken indoors and outdoors, and (2) image contents of the given photos may contain a number of building in a close proximity. Consider Condition (1). According to a study of mobile phones, locations from a mobile phone may have a root mean square error (RMSE) of 12.5 or 21.6 meters depending on that the phone is used outdoors or indoors [55]. Sometimes, an error of 47.9 meters might occur when a phone is used indoors [55]. Consider Condition (2). Figure 2.1 sketches the distribution of places of interest retrieved from Yahoo! Travel [5] for San Francisco and that for Manhattan on the Google Map [3]. As can be seen, many of the places are in a close proximity. Thus, integrating visual features and geo-tags of photos directly may have a limitation for city-view image location identification.

Previously, for landmark search, Kennedy and Naaman extracted representative images for landmarks by visual clustering and effective ranking approaches [25]. Subsequently, Avrithis *et al.* attempted to retrieve any kind of images, including landmarks and non-landmarks, followed by showing a two-layer clustering scheme based on visual and geo-clustering [7]. Considering city-scale landmarks, Chen *et al.* identified buildings from street-view images [11]. Later, Kuo *et al.* extended [11] and integrated visual features and geo-tags of images to develop a real-time image identification system [28]. Liu *et al.* combined techniques of large-scale image

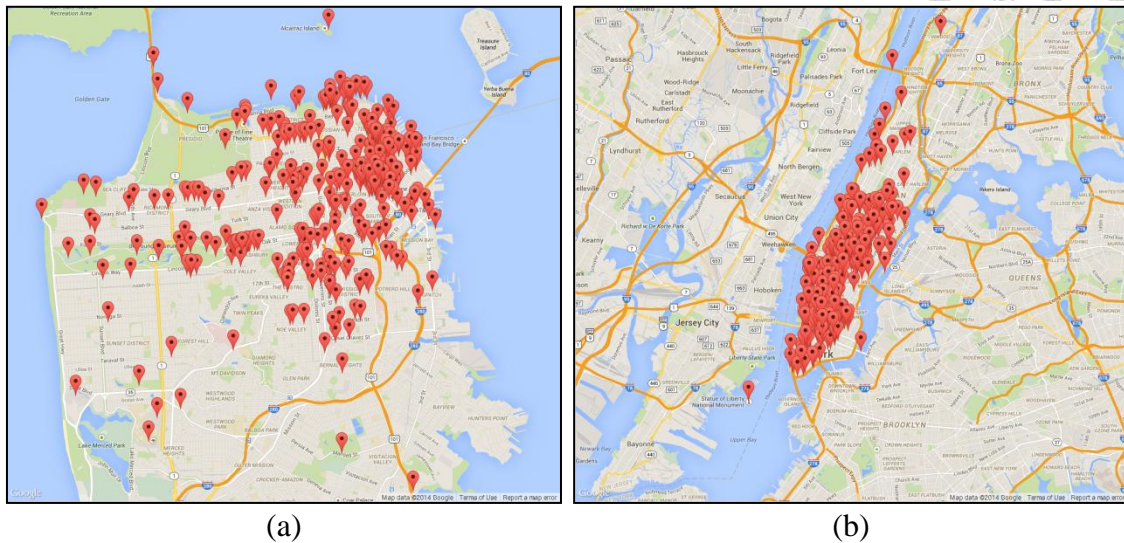
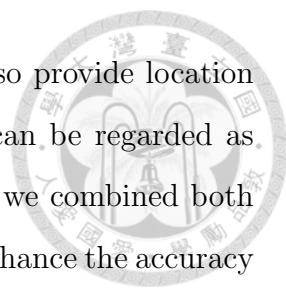


Figure 2.1: Distribution of places of interest retrieved from Yahoo! Travel [5] for (a) San Francisco and (b) Manhattan, on the Google Map [3]. Each marker is associated to a place. It is apparent that many places are in a very close proximity. Distinguishing a building in a city from the others may be challenging for traditional techniques.

identification and 3D model reconstruction to provide users' information of location, viewing directions, and routing for photography suggestions [30]. Recently, Rischka and Conrad performed visual-clustering using hierarchical k -means trees for high-dimensional visual features [40]. Feng *et al.* discovered image descriptors and attempted to extract effective color and texture features [17].

Most previous works on landmark identification were based on image clustering of visual features and geo-tags. The performance of these works may depend on the setting of the number of clusters or the radius of a cluster. Moreover, GPS location (in geo-tags) that has been derived from positioning systems may not be accurate enough. Consequently, we investigate the addition of check-in data as a complement to geo-tags for city-view image location identification, where check-in data are used to guide image clustering in our work.

As a result, we unify visual features, geo-tags, and check-in data of photos, from cross-domain social media. Check-in data offer status messages for users of social networks, mostly aiming to provide the current location (usually, a venue)

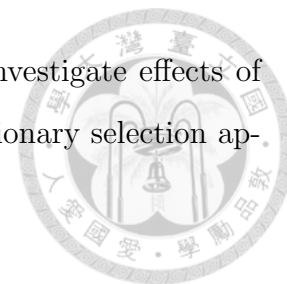


of someone for his or her friends [31]. That is, check-in data also provide location information of photos. In contrast to geo-tags, check-in data can be regarded as complementary user-contributed data to photos. Consequently, we combined both geo-tags and check-in locations acquired from Foursquare [2] to enhance the accuracy of location information for location identification, where check-in data are used to generate initial clusters of images in this work. Further, we add photos from Flickr [1] to increase diversity and performance.

In addition, we observed that some photos might not be closely related to the places these photos were taken at, but related to other places. For example, image contents of photos taken at a skyscraper may be more closely related to buildings near the skyscraper than the skyscraper itself. Intuitively, it is desirable to make the photos associated to the places closely related to themselves. As a result, this chapter presents a location-aware graph-based regrouping approach, leveraging the construction of spanning graphs [54, 58], to optimize the initial clusters for the improvement of search performance. We further extend to include the places of interest in Manhattan for the experimental setup, and conduct experiments to investigate effects of using cross-domain social media and effects of using regrouping on the search performance. In summary, this chapter has five main contributions:

- We present an effective approach that unifies visual features, geo-tags, and check-in data of photos to confront the challenges of city-view image location identification.
- We show a way to circumvent the problem of determining the number of clusters or the radius of a cluster in conventional clustering-based techniques.
- We propose a location-aware graph-based regrouping approach on clusters of photos, leveraging spanning graph construction, to refine clustering results.
- We implement our location identification technique based on sparse coding, where a graph-based approach is proposed for dictionary selection.

- We conduct experiments on large-scale real datasets to investigate effects of using cross-domain social media, regrouping, and the dictionary selection approach.



The rest of the chapter is organized as follows. Section 2.2 outlines our image location identification technique. Section 2.3 presents our regrouping approach. Section 2.4 reviews the technique of sparse coding, followed by presenting our dictionary selection approach. Finally, Section 2.5 reports the experimental results.

2.2 City-View Image Location Identification

This section introduces our location identification technique. For readability, we first explain the function of sparse coding for this work, and leave the details of sparse coding to Section 2.4.

For our implementation of location identification, measuring the visual similarity between two images (based on their visual features) is a frequently used process. However, the dimension of the visual feature (e.g., bag-of-words) of an image could be set to be very large in order to cover as many feature dimensions as possible, e.g., 1 million in this work. Although the visual features are typically sparse, it could be relatively time-consuming to iterate the process during location identification for many times. Moreover, storing visual features of a large amount of images could be memory-consuming. Consequently, we adopt a two-phase strategy. We first extract a high-dimensional visual feature for each image on purpose because it is unclear which feature dimensions are truly important to images in our image datasets. Sparse coding is then used to transform the visual feature of each image into a low-dimensional vector based on a *dictionary* associated to the image.

Ideally, the transformation from sparse coding preserves the similarity relations between images. We can measure the visual similarity between two images based on their vectors derived from sparse coding instead of visual features, if the vectors were from the same dictionary. In summary, we do not directly extract a

low-dimensional visual feature for each image but rely on sparse coding to identify important feature dimensions for the image. To this end, we will create dictionaries for sets of images, so that we can extract appropriate information from the given visual feature to form a low-dimensional vector (based on sparse coding) for each image. In the sequel, we will use the terms *sparse vector* and *dictionary* when regarding to sparse coding.

The implementation of our image location identification technique consists of two procedures, including data collection and location identification. Figures 2.2 and 2.3 show the procedures of data collection and location identification, respectively¹. In the following, Section 2.2.1 details the procedure of data collection, and Section 2.2.2 details the procedure of location identification.

2.2.1 Data Collection

For data testing, we are given a set of places from a city or a zone. In this work, we extracted a name list of places of interest from Yahoo! Travel [5] for San Francisco and a name list for Manhattan. Note that the lists of places of interest can be also acquired from other sources and for other cities or zones. Overall, there are 387 and 350 places for San Francisco and Manhattan, respectively. Figure 2.1 sketches the distribution of these places. Among these places, most of them are buildings, and some might not be easily recognizable. In the sequel, for clarity, we consider only a list of places of interest (same methods were used for the other list).

Given a set of places, we then acquired check-in data from Foursquare [2]. Foursquare is a location-based social networking website for mobile devices. Foursquare allows users to upload photos and *check in* to a specific location, *i.e.*, *venue* in Foursquare, that has a specific geo-tag consisting of the latitude and longitude of the location. We thought that the collections of the check-in data are accurate to

¹To make the source of photos clear, all of the photos shown in the two figures were obtained from the same source, *i.e.*, Foursquare [2]. Nevertheless, we will use photos from Foursquare and photos from Flickr [1] for practical implementation. Moreover, the query photo can be any photo provided by users, not limited to a photo from Foursquare.

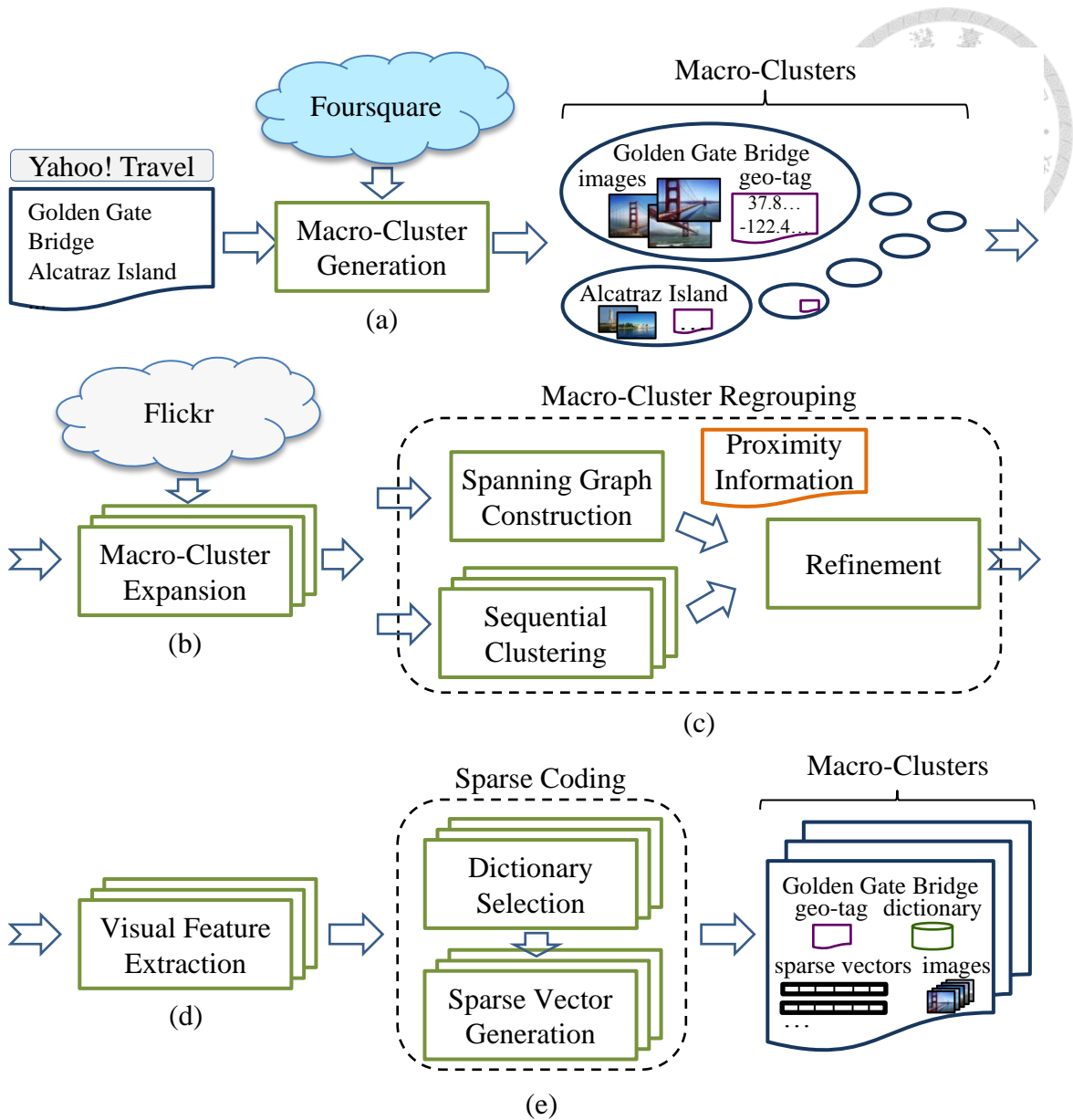
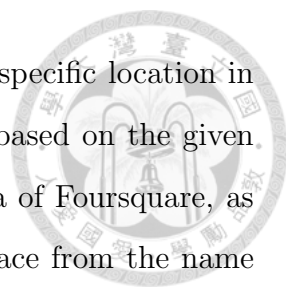


Figure 2.2: Overall flow of our data collection procedure. The procedure is mainly composed of five steps, including (a) macro-cluster generation, (b) macro-cluster expansion, (c) macro-cluster regrouping, (d) visual feature extraction, and (e) sparse coding. Given a name list of place of interest, we first generate a set of macro clusters, where each macro cluster consists of a text-tag, a geo-tag, and a set of images. For better diversity and performance, we then add images from Flickr to each of the macro clusters. (This step can easily be extended to consider other media streams for cross-domain media combination.) Afterwards, the set of macro clusters are regrouped for better search performance. We then extract visual features of all images in the macro clusters. Finally, sparse coding is used to create sparse vectors and a dictionary for each macro cluster, based on the visual features of images of the macro cluster. (Note that we do not create sparse vectors for all images at once, but cluster-by-cluster, aiming to find appropriate vectors and dictionaries for the macro clusters of images.)



some extent. That is, the geo-tag and most photos linked to a specific location in Foursquare are indeed associated to the location. As a result, based on the given name list, we generated a set of clusters using the check-in data of Foursquare, as shown in Figure 2.2(a), where each cluster is associated to a place from the name list and a specific location in Foursquare. More specifically, each cluster consists of (1) a name tag (*i.e.*, a text-tag), (2) a geo-tag, and (3) a set of images. We define each of the clusters as a *macro cluster*, in order to differentiate from *micro clusters* that will be mentioned later in Section 2.3. To increase the diversity of each macro cluster, we then added images from Flickr [1] to every macro cluster with keyword-based search, as shown in Figure 2.2(b). We will show in Section 2.5.1 that adding the images can also increase the search performance.

Empirically, an image is potentially harmful to our search performance if (1) human faces cover more than a quarter of the area of the image, (2) the image consists of several images, *e.g.*, an image collage, or (3) the image has been pre-processed, such as color filtering and blurring. This is because any of the three cases might reduce the *useful* feature dimensions of images to our location identification system. Therefore, we automatically removed those images from all the macro clusters. Totally, 420,123 images remain in the macro clusters for San Francisco, and 242,360 images remain in the macro clusters for Manhattan.

We then refined the macro clusters with a regrouping process that moves some images from one macro cluster to another, as shown in Figure 2.2(c). This is because we observed that some images inside a macro cluster sometimes are not closely related to the name tag of the macro cluster. For example, the macro cluster associated to a tower or a skyscraper may contain images that were taken for other places. Regrouping is used to redistribute images among the macro clusters for better intra-cluster similarity. (See Section 2.3 for more details). Subsequently, as shown in Figure 2.2(d), visual features were extracted from all the images (see Section 2.5). Finally, for each macro cluster, a dictionary for sparse coding was selected (based on the images in the macro cluster), and then sparse vectors of all

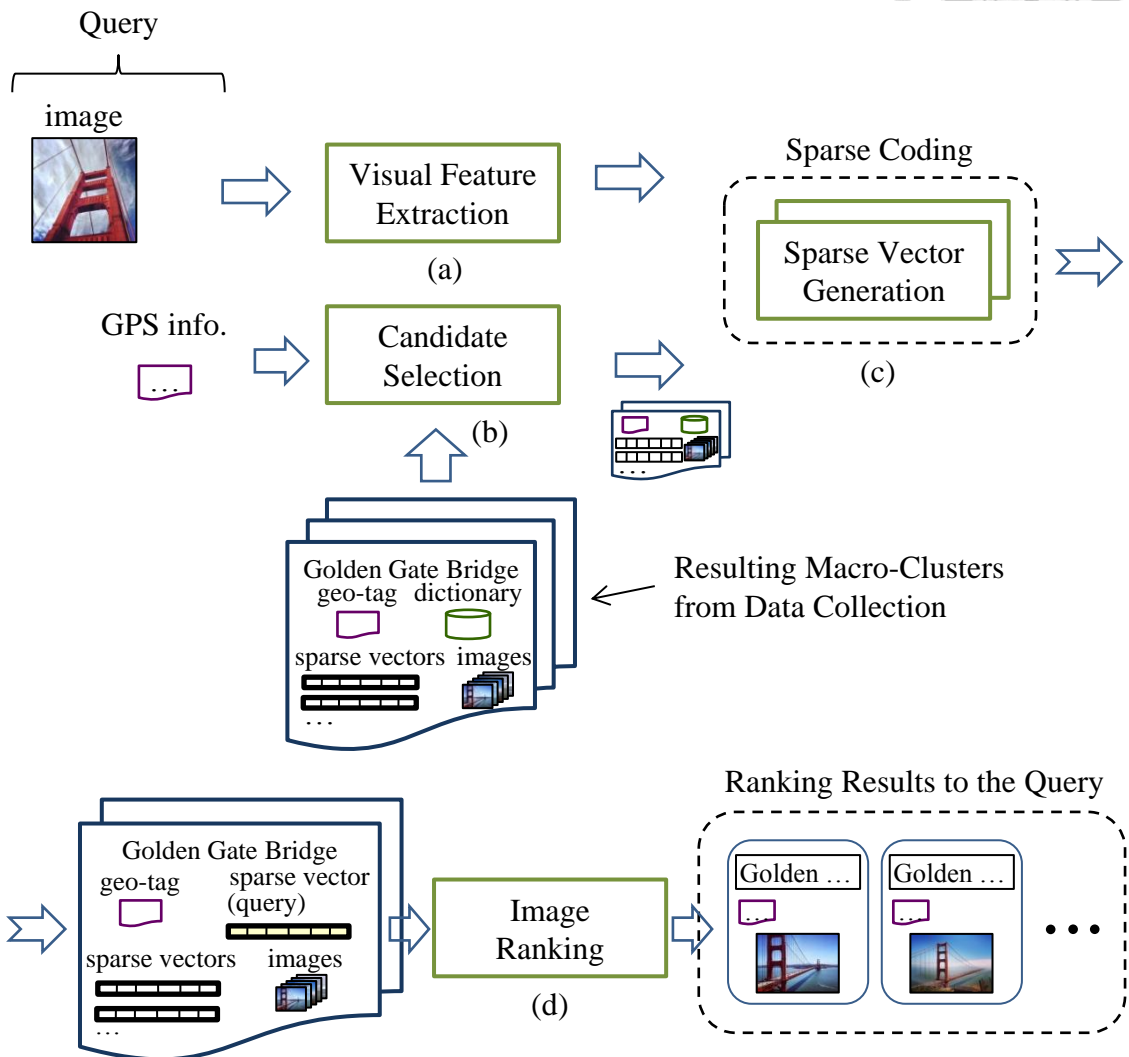


Figure 2.3: Overall flow of our location identification procedure. The procedure is mainly composed of four steps, including (a) visual feature extraction, (b) candidate selection, (c) sparse vector generation, and (d) image ranking. We are given a query consisting of an image and GPS location, and the macro clusters obtained from the data collection procedure (Figure 2.2). We first extract the visual feature of the query image, followed by removing the macro clusters that are far from the GPS location. Note that each macro cluster has its own dictionary, and two sparse vectors can make a fair comparison only if they were derived from the same dictionary. We then generate sparse vectors for the query image, one for each of the remaining macro clusters. Finally, we will generate a ranked list for the images in the remaining macro clusters.

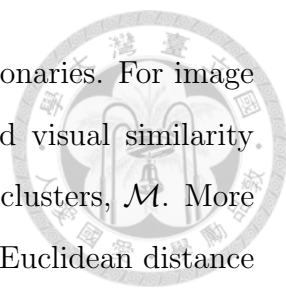
images inside the macro cluster were generated based on the dictionary, as illustrated in Figure 2.2(e).

So far, we have a set of macro clusters, where each macro cluster consists of a name tag, a geo-tag, a dictionary, a set of images, and a set of sparse vectors.

2.2.2 Location Identification

For location identification, a user may submit a photo and current GPS location from a mobile device as a query to our location identification system. For easier presentation, the photo and the GPS location provided by the user are called the *input image* and the *input location*, respectively. We first extract the visual feature of the input image, as shown in Figure 2.3(a). Then, we use the input location to filter some of the macro clusters (obtained from Section 2.2.1) that are unlikely to be relevant to what the image was taken for, as shown in Figure 2.3(b). Note that the input location is not essential for a query to our system, but would benefit to the search performance. We preserve only the macro clusters whose corresponding locations (in their geo-tags) are 21.6 meters near the input location because the input location may have an RMSE of 21.6 meters (see Section 2.1). Obviously, the length, 21.6 meters, is adjustable. The setting of the length provides a trade-off between the search performance and the execution time of our system. For clarity, let the set of the remaining macro clusters be \mathcal{M} . However, if the cardinality of \mathcal{M} (*i.e.*, $|\mathcal{M}|$) is less than a positive integer α , the nearest $\alpha - |\mathcal{M}|$ macro clusters (except any macro cluster in \mathcal{M}) to the input location will be added into \mathcal{M} . Empirically, α is set to 10. By doing so, we may achieve a good balance between the search performance and the execution time because rarely check-in location could be with significant errors [43].

Subsequently, for each macro cluster $C \in \mathcal{M}$, based on the dictionary in C , the visual feature of the input image is used to generate a sparse vector, as shown in Figure 2.3(c). Totally, there will be $|\mathcal{M}|$ sparse vectors associated to the input image. Note that the sparse vectors associated to the input image would be different



because mostly these vectors are derived based on different dictionaries. For image ranking, see Figure 2.3(d), we consider both geo-similarity and visual similarity between the input image and images from the remaining macro clusters, \mathcal{M} . More specifically, for each macro cluster $C \in \mathcal{M}$, we calculate (1) the Euclidean distance d_g between the input location and the corresponding location of C , and (2) the Euclidean distance d_v^i between the sparse vector of the input image and the sparse vector of each image with an index i in C . The score for each image with an index i in C is then set to the inverse of the product of d_g and d_v^i .

As a result, each image in the macro clusters of \mathcal{M} has a score that represents its similarity to the input image considering their geo-similarity and visual similarity. A high score represents a high similarity. Finally, for all the images in \mathcal{M} , our system returns a ranked list of images, where each image in the ranked list is accompanied by a name tag and a geo-tag. The name tags indicate what the user might take for, while geo-tags show where the user might be close to. If necessary, the geo-tags and the input image can also be used to perform a new query for to increase the search performance.

2.3 Macro-Cluster Regrouping

This section introduces our macro-cluster regrouping approach. During the process of data collection, we found that some images are not closely related to check-in locations they were uploaded for, but to nearby locations. It is desirable to assign these images to locations that are more closely related to them if there are. More specifically, given macro clusters of images, the regrouping approach aims to redistribute images among the macro clusters such that our location identification system is capable of achieving high search performance. Consider the macro clusters of images which are ready for our regrouping process (mentioned in Section 2.2.1). We assume that the majority of the images have already been in the macro clusters which are closely related to themselves because the macro clusters were immediately derived from Foursquare or Flickr. Therefore, the regrouping approach shall not

drastically change the distribution of the images.

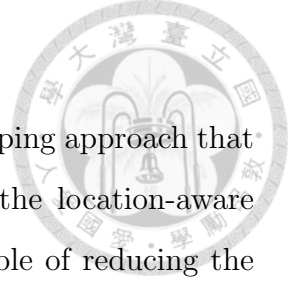
In the following, Section 2.3.1 presents a naive regrouping approach, and Section 2.3.2 presents a location-aware regrouping approach used in our system.



2.3.1 A Naive Approach

For macro-cluster regrouping, a naive approach may process images one at a time. Whenever an image is considered, for each macro cluster, we calculate a visual similarity score between the image and the macro cluster. Afterwards, the image is moved to the macro cluster that is with the highest similarity score for the image. Note that it is preferable not to move an image from the macro cluster (derived from Foursquare or Flickr) that initially contains the image, unless the movement is convincing to be beneficial. Thus, we may define a threshold value, says β , and then move an image to a macro cluster only if the similarity score of the image is higher than β . However, this approach has two drawbacks.

First, a macro cluster actually may consist of several groups of images, where each group is associated to a particular scene of the macro cluster. For example, images of a building (*i.e.*, a macro cluster) may be classified into two groups depending on whether the images were taken indoors or outdoors. Suppose that a given image should be added to a group in a macro cluster C . The presence of images from the other groups in C may bias the similarity score between the image and C , if all of the images in C are considered. Consequently, the image may be moved to some other macro cluster. For example, consider a macro cluster with two groups of images, indoor and outdoor. Given an image that was taken for an outdoor scene, this image may not be with a high similarity score if for the macro cluster, the majority of images were taken for indoor scenes. Second, images are processed based on a pre-determined order; therefore, the results of the naive approach would be sensitive to the order of processing the images.



2.3.2 Regrouping by Proximity

This section presents a location-aware graph-based regrouping approach that improves the naive approach described in Section 2.3.1. For the location-aware regrouping approach, we first present an approach that is capable of reducing the risk of moving images improperly based on the geo-tags of macro clusters, followed by presenting two techniques used to address the two drawbacks of the naive approach. Finally, we will summarize the whole location-aware regrouping approach.

For macro-cluster regrouping, the setting of a threshold value (*i.e.*, β in Section 2.3.1) aims to reduce the risk of moving images improperly. Here, we present an approach aiming to further reduce the risk by reducing the number of macro clusters which each image requires to consider. Intuitively, for a macro cluster C , it is not necessary for images in C to consider (1) macro clusters which are far away from C , and (2) macro clusters where the views associated to the macro clusters are *blocked* by the view(s) associated to some other macro cluster(s).

For example, a skyscraper (*e.g.*, C_2 in Figure 2.4(a)) may block the views just behind the skyscraper. It might not be necessary to consider moving images in a macro cluster associated to a place in front of the skyscraper (*e.g.*, C_1 in Figure 2.4(a)) to those macro clusters associated to the places just behind the skyscraper (*e.g.*, C_3 in Figure 2.4(a)). Even if C_2 is not a skyscraper, we thought that taking photos for views behind some other views are relatively uncommon. However, it is challenging to define a specific distance so as to determine whether one macro cluster is far away from another macro cluster or not, because the maximum distance that a camera can produce images depends on several factors, such as camera specifications and lighting conditions. It is also a hard work to identify whether a view is blocked or not.

Consequently, we implement a spanning graph [54, 58]. For each macro cluster, the spanning graph helps us determine which macro clusters should be considered for. Originally, constructing a spanning graph was for efficiently finding a minimum spanning tree. Given a set of vertices, a spanning graph with the vertices

is a simple graph that contains a sufficient number of edges between the vertices such that a minimum spanning tree can be constructed by using some of the edges of the spanning graph. Recently, for some studies, constructing spanning graphs has been considered as a technique of finding critical neighborhoods of vertices in a plane, *e.g.*, [21].

A spanning graph can be constructed as follows. Consider a set of vertices in a plane. We traverse the vertices in any order. Whenever a vertex is visited, we divide the plane evenly into eight regions with respect to the vertex, followed by connecting the vertex to the nearest vertex in each region. Eventually, we can obtain a spanning graph for the set of vertices, once all of the given vertices have been visited. For our regrouping approach, each vertex of a spanning graph is associated to a macro cluster, where the locations of the vertex is determined by the geo-tag of the macro cluster. Figure 2.4 illustrates the idea of constructing a spanning graph for macro clusters C_1, C_2, C_3 , and C_4 , in a plane. As shown in Figure 2.4(a), initially, the macro clusters are disconnected. Assume that C_1 is first visited. The plane is divided into eight regions with respect to C_1 . As shown in Figure 2.4(b), C_1 is then connected to C_2 and C_4 , because either of C_2 and C_4 is the nearest macro cluster to C_1 in a region. Sequentially, C_2, C_3 , and C_4 will be visited and edges will be added accordingly. As a result, Figure 2.4(c) shows the resultant spanning graph for the given four macro clusters.

Given a spanning graph, images of a macro cluster C will not consider the macro clusters which are not adjacent to C in the spanning graph. The rationale behind is that those macro clusters might be far away from C . Besides, the views associated to those macro clusters could be *blocked* by the view(s) associated to some other macro cluster(s), because intuitively taking photos for views behind some other views are relatively uncommon. That is, for regrouping, images of a macro cluster C consider only the macro clusters which are adjacent to C in the spanning graph. Take Figure 2.4(c) as an example. Images in C_1 will not consider C_3 , because C_1 and C_3 are not adjacent to each other. The idea is that, for C_1 ,

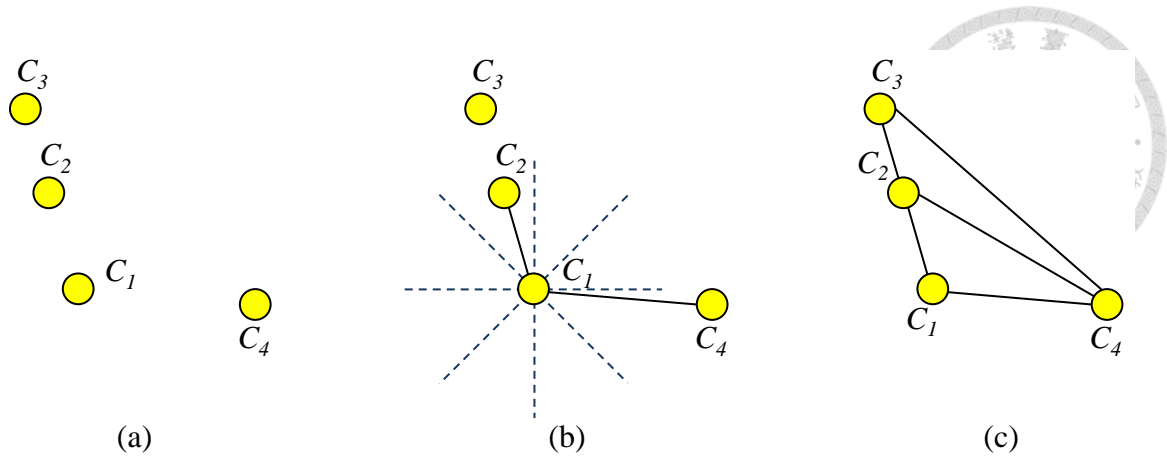


Figure 2.4: Illustration of constructing a spanning graph for four macro clusters, C_1 , C_2 , C_3 , and C_4 , in a plane. (a) The macro clusters are initially disconnected. (b) Assume that C_1 is first visited. The plane is divided into eight regions with respect to C_1 , and C_1 is connected to the macro cluster that is the nearest one of C_1 in each region, *i.e.*, C_2 and C_4 . (c) The resultant spanning graph for C_1 , C_2 , C_3 , and C_4 , after the four macro clusters have been visited. (Details of constructing a spanning graph can be found in Section 2.3.2.) With the spanning graph, images of a macro cluster will not consider macro clusters which are not adjacent to it for regrouping, because the graph is expected to find critical neighborhoods for these macro clusters.

C_3 is farther than C_2 , and the view associated to C_3 could be blocked by the view associated to C_2 .

Now we consider the two issues of the simple approach mentioned in the last paragraph of Section 2.3.1. The first issue comes from the fact that a macro cluster may consist of several groups of images. The second issue is that the results of the simple approach could be highly sensitive to the order of processing images. For the first issue, we group the images in each macro cluster into a set of *micro clusters*, where each micro cluster is associated to a particular scene of the macro cluster. We take sequential clustering [44] as the clustering approach to generate micro clusters because it is difficult to determine an appropriate number of scenes that a macro cluster contains in advance, while sequential clustering can determine the number naturally. Once all macro clusters have been processed, we begin to calculate similarity scores. In contrast to the simple approach, for each image being considered, here we calculate visual similarity scores between the image and micro

clusters, instead of macro clusters.

By doing so, for each image, the influence of irrelevant images on similarity scores could be minimized. We then consider moving each image to the micro cluster that is with the highest similarity score to the image. Although our objective is to redistribute images among macro clusters, not micro clusters, moving an image to a micro cluster H is equivalent to move the image to the macro cluster that contains H . For the second issue, we calculate similarity scores of all images to micro clusters, followed by moving the images to their desired macro clusters concurrently. Therefore, the clustering results of our regrouping approach are independent of the order of processing the images.

Figures 2.5 and 2.6 summarize the whole process of our location-aware graph-based regrouping approach. In Line 3, we construct a spanning graph for the given macro clusters. In Lines 4–6, we generate micro clusters for each macro cluster. In Lines 7–15, we determine a desired macro cluster for each image. In Lines 16–24, we consider moving images to their desired macro clusters.

2.4 Sparse Coding for Macro Clusters

This section describes how sparse coding is used for images of each macro cluster. In the following, Section 2.4.1 reviews the technique of sparse coding, and Section 2.4.2 presents our dictionary selection approach.

2.4.1 Formulation

Sparse coding has been shown success in various fields, such as machine learning, signal processing, and statistics [35]. For image location identification, we intend to transform visual features of images into sparse vectors. The transformation is applied cluster-by-cluster to the macro clusters obtained in Section 2.3.2. Formally, consider a macro cluster C . Let n be the number of images in C . Let m be the dimension of the visual feature of an image. Let $V = (v^{(1)}, v^{(2)}, \dots, v^{(n)})$, $V \in \mathbb{R}^{m \times n}$ be a matrix consisting of visual features of images in C , where $v^{(i)} \in \mathbb{R}^m$ is a vec-



```

1: Procedure Regrouping(macro clusters  $\mathcal{M}$ )
2: Begin
3:    $G = \text{CreateSpanningGraph}(\mathcal{M});$ 
4:   foreach macro cluster  $C \in \mathcal{M}$ 
5:      $\text{GenerateMicroClusters}(C);$ 
6:   end foreach
7:   foreach macro cluster  $C \in \mathcal{M}$ 
8:     foreach micro cluster  $H \in C$ 
9:       foreach image  $q \in H$ 
10:         $q.\text{maxSimilarity} = \text{CalcSimilarity}(H, q);$ 
11:         $q.\text{bestMacroCluster} = C;$ 
12:         $\text{FindDesiredCluster}(G, C, q);$ 
13:       end foreach
14:     end foreach
15:   end foreach
16:   foreach macro cluster  $C \in \mathcal{M}$ 
17:     foreach image  $q \in C$ 
18:        $T = q.\text{bestMacroCluster};$ 
19:       if  $(T \neq C) \wedge (q.\text{maxSimilarity} > \beta)$ 
20:          $C = C \setminus \{q\};$ 
21:          $T = T \cup \{q\};$ 
22:       end if
23:     end foreach
24:   end foreach
25: End

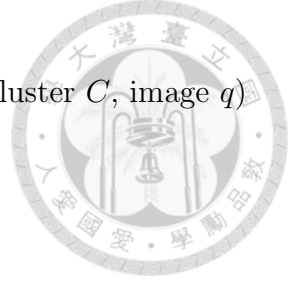
```

Figure 2.5: Algorithm of the proposed graph-based regrouping process. Given macro clusters, the process moves images to their desired macro clusters.

tor of the visual feature of the image with an index i in C for $1 \leq i \leq n$. Let $S = (s^{(1)}, s^{(2)}, \dots, s^{(n)})$, $S \in \mathbb{R}^{r \times n}$ be a matrix consisting of sparse vectors, where $s^{(i)} \in \mathbb{R}^r$ is a sparse vector associated to the image with an index i in C . Motivated by [29], we then consider the following optimization problem.

$$\begin{aligned}
& \underset{D, S}{\text{minimize}} && \sum_{i=1}^n \left(\|v^{(i)} - Ds^{(i)}\|_2^2 + \gamma \|s^{(i)}\|_1 \right), \\
& \text{subject to} && \sum_{i=1}^m (D_{ij})^2 = 1, 1 \leq j \leq r,
\end{aligned}$$

where $D \in \mathbb{R}^{m \times r}$ is often called a dictionary, consisting of r basis vectors in \mathbb{R}^m , and $\gamma \geq 0$ is a constant. Among the objective function, the left term represents an



```

1: Procedure FindDesiredCluster(spanning graph  $G$ , macro cluster  $C$ , image  $q$ )
2: Begin
3:   foreach macro cluster  $T$  adjacent to  $C$  in  $G$ 
4:     foreach micro cluster  $W \in T$ 
5:       curSimilarity = CalcSimilarity( $W$ ,  $q$ );
6:       if curSimilarity >  $q$ .maxSimilarity
7:          $q$ .maxSimilarity = curSimilarity;
8:          $q$ .bestMacroCluster =  $T$ ;
9:       end if
10:    end foreach
11:  end foreach
12: End

```

Figure 2.6: Algorithm of finding desired cluster for the proposed graph-based re-grouping process. Given a spanning graph, a macro cluster, and an image, the process finds the desired macro cluster with the associated similarity between the image and a micro cluster of the macro cluster for the image.

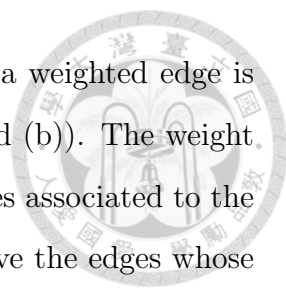
error to reproduce the given visual feature of an image using a linear combination of the basis vectors from the dictionary ($s^{(i)}$ is a vector of weights for $1 \leq i \leq n$), while the right term is for sparsity. γ is used to control the relative importance of the two terms. The addition of the constraint is to normalize the basis vectors in D .

We will show how to find D (and r) in Section 2.4.2. Once D has been determined, the optimization problem can be reduced to an L1-regularized least square problem, and then efficiently solved by the least angle regression algorithm [15].

2.4.2 Dictionary Selection within a Macro Cluster

Given a set of images, this section describes how to create a dictionary from the images for sparse coding. Similar to random sampling-based approaches [12,38], our idea is to find a subset of the given images for dictionary creation. Different from random sampling, we intend to find the subset based on visual features of images.

The proposed approach consists of three steps: (1) graph construction, (2) edge removal, and (3) image removal. For the first step, we first construct a visual



similarity graph, where each image is modeled as a vertex and a weighted edge is added between every pair of vertices, (*e.g.*, see Figure 2.7(a) and (b)). The weight of each edge is set to the visual similarity between the two images associated to the two terminal vertices of the edge. For the second step, we remove the edges whose weights are too small from the visual similarity graph by thresholding, (*e.g.*, see Figure 2.7(c)). As for the final step, this step concentrates on reducing the size of the visual similarity graph by removing the vertices that might not be *important* (see, next paragraph). Once we have removed those vertices, the images associated to the vertices of the resulting graph are used to create a dictionary D for sparse coding. More specifically, each basis vector in D will be the visual feature of an image from the resulting graph.

However, it is difficult to define the so-called *importance* of a vertex, *i.e.*, an image. Intuitively, the importance of a vertex of a graph is closely related to the topology of the graph. We could remove a vertex u from a visual similarity graph while *preserving* the similarity relations of the graph well if all vertices strongly correlated (*i.e.*, adjacent) to u are preserved. Moreover, it is desirable to preserve as small number of vertices as possible because the number of remaining vertices after the step of image removal will be the dimension of sparse vectors generated by sparse coding. Consequently, we consider the minimum vertex cover problem, which is to find the smallest set of vertices from a graph such that every edge of the graph must be incident to at least one vertex of the set. We can remove the vertices not belonging to the resulting vertex set of the minimum vertex cover problem because the similarity relations of the graph should be well-preserved by vertices of the resulting vertex set. Although it is unlikely to efficiently solve the minimum vertex cover problem, lots of efficient approximation algorithms have been developed, *e.g.*, [24].

In this work, as suggested in [47], the minimum vertex cover problem was formulated as a 0-1 integer linear programming. As already defined, n is the number of images in a macro cluster. Further, we define Z as the given graph for the mini-

minimum vertex cover problem; E as the edge set of Z . Each vertex of Z is associated to a variable, $x_i \in \{0, 1\}$, for $1 \leq i \leq n$. The minimum vertex cover problem was formulated as follows.

$$\begin{aligned} & \text{minimize} && \sum_{i=1}^n x_i, \\ & \text{subject to} && x_i + x_j \geq 1, \forall (i, j) \in E, \end{aligned}$$

where a vertex is in the resulting vertex set of the minimum vertex cover problem if the variable associated to the vertex is set to one; otherwise, the vertex is not. We can see that the objective is to minimize the number of vertices in the resulting vertex set, while the constraint is to ensure that every edge is covered by at least one vertex of the set. We directly solve the problem if the number of vertices is small. Otherwise, an approximate solution is achieved through the linear programming relaxation of the formulation.

Figure 2.7 summarizes the overall process of our graph-based selection approach². Given a macro cluster with images, we construct a visual similarity graph based on the images, and then we remove the edges whose weights are too small from the graph. Finally, we formulate a minimum vertex cover problem, followed by removing the vertices not belonging to the resulting vertex set of the problem.

2.5 Experiments

This section evaluates the proposed approaches. We implemented our approaches using the Matlab programming language. All of the evaluations were based on the two image datasets described in Section 2.2.1. Overall, for San Francisco, there are 387 macro clusters (*i.e.*, 387 query categories) and 420,123 images in the associated image dataset. For Manhattan, there are 350 macro clusters and 242,360 images in the associated image dataset. We extracted visual features using the following way. We took *visual words* (*i.e.*, bag-of-words) as visual features. For each image, we applied the difference-of-Gaussian approach to detect feature

²All of the photos shown in this figure were obtained from Foursquare [2].

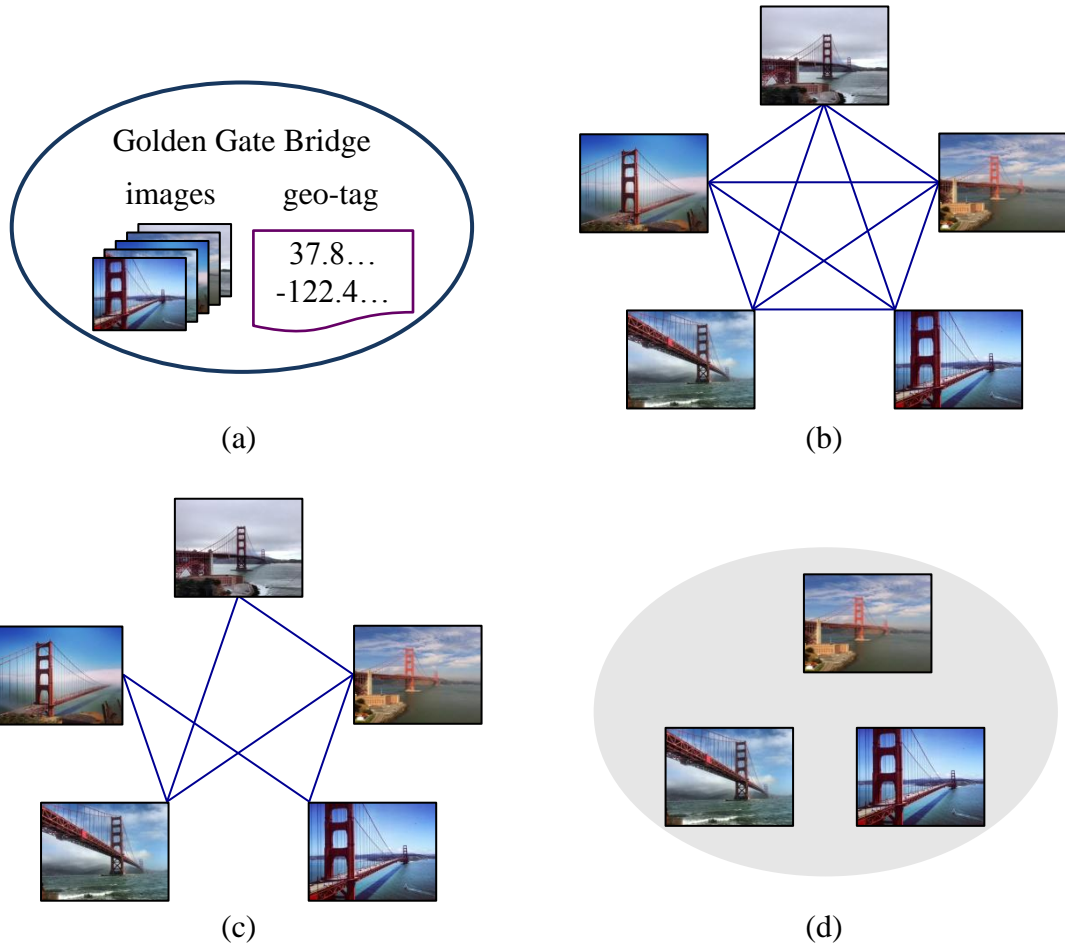


Figure 2.7: Illustration of our graph-based selection approach: (a) consider a macro cluster with images; (b) construct a visual similarity graph based on the visual similarities of images; (c) remove the edges whose weights are too small; (d) remove images while retaining representative images by solving the vertex cover problem. Finally, the images associated to the vertices of the resulting graph are used to create a dictionary. (*cf.* Section 2.4.2.)

points, followed by describing them with scale invariant feature transform (SIFT). The SIFT descriptors were then quantized into 1 million clusters by k -means clustering. Eventually, each cluster generated a visual word containing the feature descriptors (feature points) of the cluster.

Because there is no ground truth in the image datasets, we determined ground truth in two ways: (1) manual annotation: images (of our image datasets) were labeled manually, and (2) automatic annotation: images from Foursquare were regarded as ground truth because generally images uploaded to a location (*i.e.*, venue) in Foursquare were taken at this location. All performances were evaluated by $P@k$, which is the precision of top- k images in ranking results. In the sequel, Section 2.5.1 evaluates our approaches based on manual-annotated ground truth, while Section 2.5.2 evaluates our approaches based on automatic-annotated ground truth.

Note that the terms, “Ours,” for all tables of Section 2.5, are exactly the same. “Ours” combines visual features, GPS locations, and check-in data, selecting images for dictionaries with minimum vertex covering, and using graph-based regrouping, for image location identification.

2.5.1 Results with Manual-Annotated Ground Truth

In this section, we conducted three suites of experiments to show the effectiveness of the use of (1) cross-domain social media (*i.e.*, Foursquare and Flickr), (2) the location-aware graph-based regrouping approach, and (3) the proposed dictionary selection approach. For ground truth, we sampled 12 locations from each image dataset and manually added tags for images of the locations.

First, we investigated the effect of using cross-domain social media on search performance. Table 2.1 shows the results of using Foursquare only and the results of combining Foursquare and Flickr as Metadata, for the images of San Francisco. Similarly, Table 2.2 shows the results of using Foursquare only and the results of combining Foursquare and Flickr as Metadata, for the images associated to Manhattan. Note that the experiments also compared our approach to three approaches,

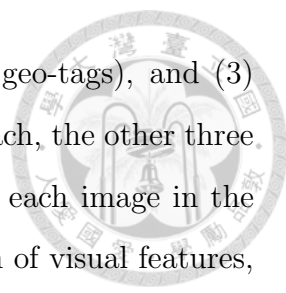


Table 2.1: Comparison between visual-based, location-based, visual and location-based, and our approach, on the Foursquare-based image dataset of San Francisco, and on the Foursquare and Flickr-based image dataset of San Francisco. Note that “Ours” combines visual features, GPS locations, and check-in data.

Dataset: Foursquare for San Francisco				
	P@1	P@2	P@5	P@10
Visual Only	0.1243	0.1631	0.1687	0.1923
Location Only	0.1744	0.1859	0.2344	0.2523
Visual + Location	0.2009	0.2056	0.2135	0.2482
Ours	0.2149	0.2417	0.2414	0.2648
Dataset: Foursquare+Flickr for San Francisco				
	P@1	P@2	P@5	P@10
Visual Only	0.1401	0.1459	0.1933	0.2407
Location Only	0.2380	0.2912	0.3399	0.3449
Visual + Location	0.3230	0.3797	0.4128	0.4405
Ours	0.3642	0.3793	0.4317	0.4682

Table 2.2: Comparison between visual-based, location-based, visual and location-based, and our approach, on the Foursquare-based image dataset of Manhattan, and on the Foursquare and Flickr-based image dataset of Manhattan. Note that “Ours” combines visual features, GPS locations, and check-in data.

Dataset: Foursquare for Manhattan				
	P@1	P@2	P@5	P@10
Visual Only	0.1547	0.1915	0.2032	0.2207
Location Only	0.1983	0.2011	0.2297	0.2375
Visual + Location	0.2508	0.2573	0.2627	0.2814
Ours	0.2577	0.2819	0.3029	0.3317
Dataset: Foursquare+Flickr for Manhattan				
	P@1	P@2	P@5	P@10
Visual Only	0.1408	0.1607	0.1792	0.2023
Location Only	0.1943	0.1988	0.2145	0.2355
Visual + Location	0.2595	0.2693	0.2999	0.3305
Ours	0.2711	0.2908	0.3312	0.3528



including (1) only visual features, (2) only GPS locations (in geo-tags), and (3) both visual features and GPS locations, where except our approach, the other three approaches successively measured similarity between query and each image in the associated image dataset. Our overall approach is a combination of visual features, GPS locations, and check-in data (*i.e.*, macro clusters).

As revealed in Tables 2.1 and 2.2, we found that combing both visual features and GPS locations (*i.e.*, Visual + Location) is beneficial for performance enhancement. On top of them, adding check-in data is beneficial to further increase their performance. In addition, as shown in the tables, it is interesting to note that combining Foursquare and Flickr is beneficial for performance enhancement. At first, we thought that the search performance of an image dataset with two image sources might be worse than the search performance of the image dataset with any one of the two image sources, because the noise to the image dataset might increase accordingly. While based on the results, we found that combining the two image sources may result in better search performance than the performance by using any of them alone. The reason why the two image sources could benefit from each other may be due to the averaging of the noise from different image sources.

Second, we investigated the effect of using our regrouping approach on search performance. Table 2.3 compares our overall approach and two variants derived from our overall approach, based on the image dataset obtained from combining datasets of San Francisco and Manhattan. Refer to Section 2.3.2. The first variant did not activate the process of macro-cluster regrouping. The second variant simplified the process of macro-cluster regrouping. The second variant does not move images among macro clusters, but remove images in macro clusters. The second variant removes images if their desired macro clusters are not exactly the same as the macro clusters they are in. Based on the results, we concluded that macro-cluster regrouping (ours) is capable of increasing search performance. Moreover, observing the results of not using regrouping and those of using the modified regrouping approach, we thought that removing an images from a macro cluster may lose a small

number of important features for the macro cluster, even if the image is more closely related to another macro cluster. Observing the results of using the simplified regrouping approach and those of using our regrouping approach (ours), we concluded that the amount of the loss (from moving out images from macro clusters) could be much smaller than the amount of added features through regrouping images.

Table 2.3: Comparison between our overall approach and its two variants for the effects of using our regrouping approach on search performance. The two variants include our approach without macro-cluster regrouping and our approach with simplified regrouping (*i.e.*, removing ambiguous images in macro clusters). Note that “Ours” uses macro-cluster regrouping.

Dataset: Foursquare+Flickr for San Francisco+Manhattan				
	P@1	P@2	P@5	P@10
Ours w/o regrouping	0.3592	0.3601	0.3777	0.4052
Ours w/ simple regrouping	0.3566	0.3738	0.3798	0.3982
Ours	0.3983	0.4055	0.4235	0.4512

Third, we investigated the effect of using our dictionary selection approach. Table 2.4 compares our overall approach and its variant derived from our overall approach, based on the image dataset obtained from combining datasets of San Francisco and Manhattan. The only difference between our overall approach and its variant is the composition of the dictionary of each macro cluster for sparse coding. For the variant, the dictionary of each macro cluster consists of all images in the macro cluster, while for our overall approach, the dictionary of each macro cluster was selected by the approach introduced in Section 2.4.2 from all images in the macro cluster.

Table 2.4: Comparison between our overall approach and its variant for the effect of using the proposed dictionary selection approach (see Section 2.4.2) or not. “VC” denotes the vertex-cover-based approach, which is the proposed dictionary selection approach. Note that “Ours” uses the dictionary selection approach.

Dataset: Foursquare+Flickr for San Francisco+Manhattan				
	P@1	P@2	P@5	P@10
Ours w/o VC	0.2593	0.2694	0.3020	0.3385
Ours	0.3983	0.4055	0.4235	0.4512

Based on the results, we found that for an image cluster, there might be

a number of noisy images inside, and some noisy images can be removed by the dictionary selection approach. More specifically, the second step of our dictionary selection approach eliminated weak similarity relations between noisy images and *good* images, and then the third step (vertex cover) implicitly removed vertices incident to few edges, which probably, were noisy images.

2.5.2 Results with Automatic-Annotated Ground Truth

In this section, we conducted an experiment to show the effectiveness of our approach, where the images (of our image dataset) from Foursquare were regarded as ground truth, *i.e.*, automatic annotation. Note that we did not activate the process of macro-cluster regrouping at this time. This is because automatic annotation was based on the assumption that images from Foursquare were regarded as ground truth, while macro-cluster regrouping was motivated by the observation that some images might not be closely related to the macro clusters they are in. As a reference, Table 2.5 compares our approach (without macro-cluster regrouping) and three approaches, based on the image dataset obtained from combining datasets of San Francisco and Manhattan, where the three approaches are the same as those compared in Tables 2.1 and 2.2. More specifically, the three approaches are (1) considering only visual features, (2) considering only GPS locations, and (3) considering both visual features and GPS locations. Similar to the results in Tables 2.1 and 2.2, as can be seen in Table 2.5, our approaches achieved higher performance than the three approaches.



Table 2.5: Comparison between visual-based, location-based, visual and location-based, and our approach. Note that here the same dataset as that for Tables 2.3 and 2.4 was used, but this experiment was based on automatic annotation (the prior experiments were based on manual-annotated ground truth). We did not show “Ours”, but “Ours w/o regrouping” because automatic annotation assumes that Foursquare provides the golden image clustering results, while the motivation of regrouping is that the clustering results might not be the best at all aspects.

Dataset: Foursquare+Flickr for San Francisco+Manhattan				
	P@1	P@2	P@5	P@10
Visual Only	0.1322	0.1734	0.2017	0.2351
Location Only	0.1929	0.2313	0.2916	0.3205
Visual + Location	0.2858	0.3259	0.3467	0.3519
Ours w/o regrouping	0.3377	0.3627	0.4016	0.4463



Chapter 3

Graph-based Semi-Supervised Learning for Image Annotation

3.1 Introduction

Image annotation aims to generate proper tags for the images of a given image dataset, where semi-supervised learning (SSL) is widely-used learning technique that explores lots of untagged images in the presence of a small amount of tagged images. Among the existing SSL approaches, graph-based approaches are quite popular due to their high efficiency [60]. Typically, a graph-based approach will model both tagged and untagged images as vertices followed by adding a weighted edge between each pair of vertices, where the weight of an edge is the similarity between its two terminal vertices (*i.e.*, images). An edge weight is either positive or zero. In practical implementation, the zero-weight edges are often removed for complexity reduction. Then, a process called label propagation is activated to add tags to images (*i.e.*, to label images) accordingly. In this chapter, we focus on the development of a *multi-label* propagation approach for graph-based SSL for image annotation. Note that in this chapter the term, “label,” is equivalent to “add tag(s)” or “tag,” depending that it is a verb or a noun, respectively.

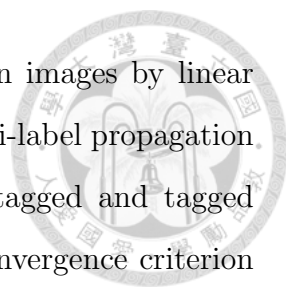
So far, there have been lots of studies on label propagation, where [19, 23, 33, 37, 39, 45, 46, 48, 57, 62] are closer to ours. Zhu *et al.* proposed to use Gaussian random field model [62]. Later, Zhou *et al.* presented an iterative algorithm followed by deriving a simple while effective closed-form solution [57]. After that, He *et al.* improved the graph structure in [57] and indicated that the closed form solution in [57] may not be applicable for large-scale label propagation [19]. Moreover, He *et al.* combined relevance feedback with label propagation. As a counterpart of [19],

which focused on the scenario of query by example, Tong *et al.* focused on the scenario of query by keyword and proposed the idea of multi-label propagation [45]. Unlike the prior works, which are deterministic methods, Pan *et al.* proposed a probabilistic solution based on random walk with restart [37].

On the other hand, Tong *et al.* presented a linear fusion and a sequential fusion schemes to fuse multi-modal features for learning quality enhancement [46]. In addition, Wang *et al.* argued to integrate *visual* and *textual* information for label propagation [48]. Liu *et al.* presented to integrate similarities immediately on different graphs for label propagation [33]. Recently, Liu *et al.* observed that most label-propagation approaches have to perform at least an inverse operation on a n -by- n graph Laplacian, where n is the number of vertices. The inverse operation often requires a cubic time (*i.e.*, $O(n^3)$) complexity, which becomes prohibitive as the number of vertices (*i.e.*, images) increases [32]. For the addressed problem, Rao and Yarowsky focused on text data to develop a parallel label-propagation algorithm [39]; unfortunately, such a text-oriented algorithm may not handle more complicated data (*e.g.*, images) well. More recently, Kang *et al.* presented a matrix-vector multiplication approach by MapReduce [23].

This chapter focuses on label propagation for graph-based SSL, where two important issues raised in recently years. One is the large-scale issue: the proposed algorithm has to be useful to high-dimensional data and large-scale datasets. The other is the multi-labeling issue: the proposed algorithm has to handle several tags on an image, so as to explore (improve) multiple tags for each untagged (tagged) image, while most previous works focused on single tag issue. Consequently, we unify the promising ideas of (a) graph-based SSL [19, 37, 57], (b) multi-label propagation [45], (c) linear fusion [33, 46], and (d) matrix-vector multiplication [23], to develop a multi-layer multi-label propagation framework for large-scale image datasets. Note that for multi-label propagation, we generalize the matrix-vector multiplication in [23] to a matrix-matrix multiplication, *i.e.*, matrix multiplication.

Overall, in this chapter, (1) we implement a flexible multi-layer learning



structure, which unifies the visual and textual features of given images by linear fusion to enhance the learning performance; (2) we consider multi-label propagation to enable the addition and refinement of multiple tags to untagged and tagged images, respectively; (3) we introduce a technique to set the convergence criterion of our multi-label propagation approach automatically instead of using an ad-hoc assignment; (4) we present a practical implementation of our multi-layer multi-label propagation approach by MapReduce, which is capable of handling large-scale image datasets; (5) we conduct experiments on a medium-scale and a large-scale image datasets to evaluate the effectiveness of the proposed methods.

The rest of this chapter is organized as follows. Section 3.2 presents the overview of our graph-based SSL system. Section 3.3 introduces the proposed algorithms for label propagation. Finally, Section 3.4 presents the experimental evaluation.

3.2 Learning System Overview

Before turning to the details of our label-propagation algorithm, this section gives an overview of our graph-based SSL system. The overall system flow is illustrated in Figure 3.1¹. Given a set of tagged and untagged images, for each image we extract both the visual and textual features. Note that the textual features might be missing. With the extracted features, we then construct a sparse visual graph and a sparse textual graph accordingly. Each vertex of the visual (textual) graph is associated with an image, while the weight of an edge between two images is the visual (textual) similarity of the two image contents. Subsequently, our label-propagation algorithm is activated to add (improve) tags to the given untagged (tagged) images. Finally, we suppress noisy tags by tag refinement and output the results.

As aforementioned, we construct two sparse graphs before the activation of label propagation. In fact, our label-propagation algorithm can also work on a dense

¹All of the photos shown in this figure were obtained from Flickr [1].

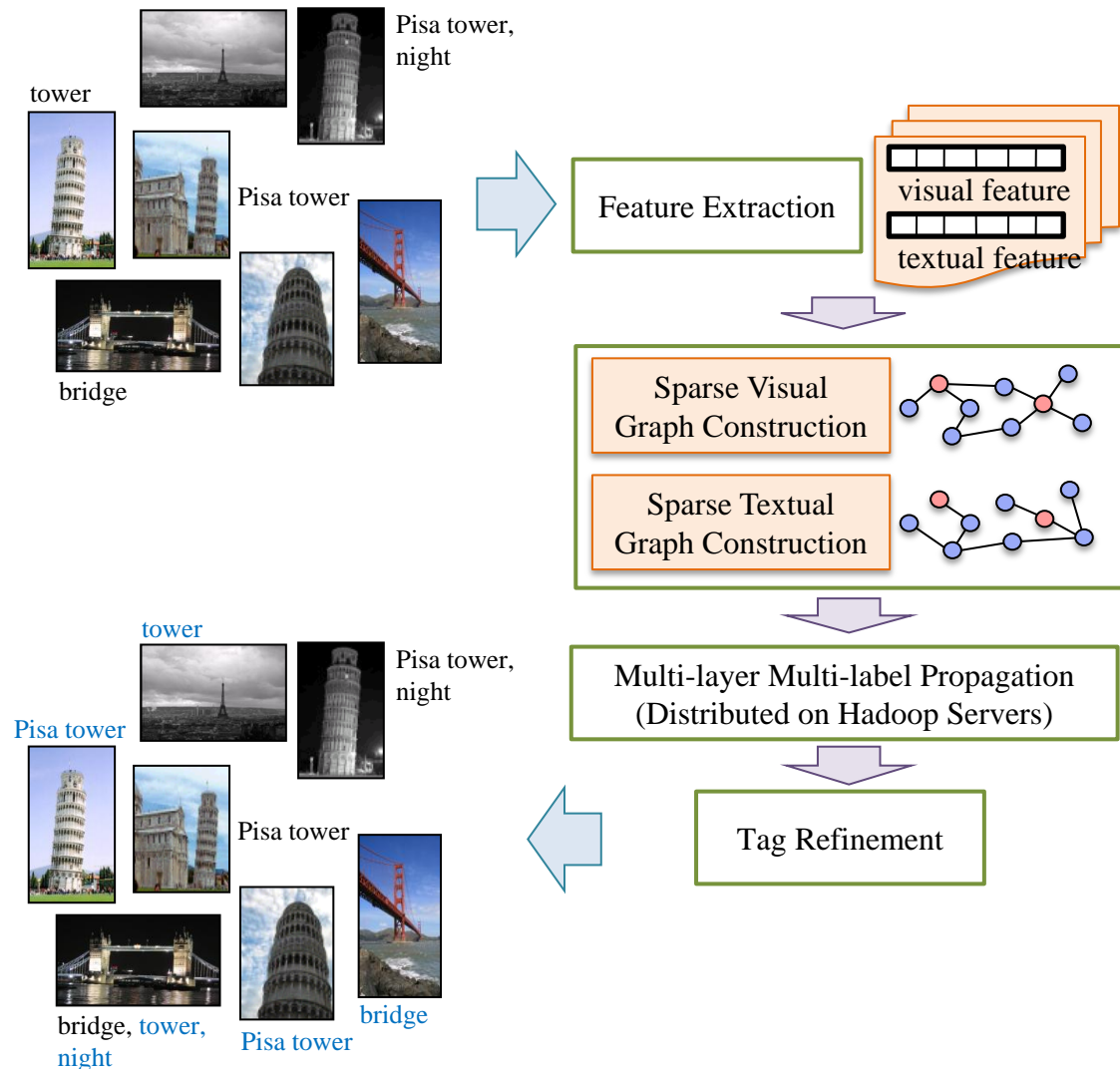
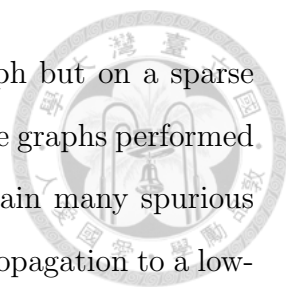


Figure 3.1: Illustration of our graph-based SSL system. Given an image dataset, we first perform feature extraction to obtain both visual and textual features. Then a sparse visual graph and a sparse textual graph are constructed to enable multi-layer label propagation. Based on the two graphs, we then perform multi-layer multi-label propagation. Finally, we apply tag refinement to improve the tag results.



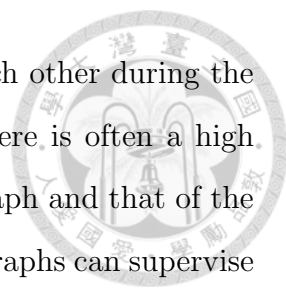
graph, *e.g.*, a complete graph. We do not work on a dense graph but on a sparse graph because it has been shown in [61] that fully-connected dense graphs performed worse than sparse graphs empirically. A dense graph may contain many spurious edges between dissimilar vertices, and therefore misleads label propagation to a low-quality result. Since graph construction is not the main focus of this chapter, we shall not give the details here, but a brief description instead, as follows. Overall, our in-house tool for graph construction consists of two stages. For large-scale issues, both the two stages were implemented based on the MapReduce programming model [14]. In the first stage, we partition the given images into overlapped groups called image pools by MinHash [9], while in the second stage, for each image pool we compute the pairwise similarities and remove image-pool boundaries afterwards. Note that we do not build any zero-weight edge to reduce the graph complexity. The resultant graph is our sparse similarity graph. For simplicity, in the rest of the chapter, we will use the terms “visual graph” and “textual graph” to be as shorthand for “sparse visual graph” and “sparse textual graph,” respectively.

3.3 Methodologies

This subsection presents our proposed methods for multi-layer multi-label propagation. Note that all of the distributed algorithms mentioned in the rest of chapter are developed based on the MapReduce model. Section 3.3.1 introduces our multi-layer learning structure followed by the overview of our proposed algorithm. Section 3.3.2 gives a formal description of our multi-layer multi-label propagation approach. Section 3.3.3 presents the convergence criterion setting of our label propagation approach. Finally, Section 3.3.4 describes the tag refinement approach.

3.3.1 Multi-Layer Learning Structure

Given a visual and a textual graph, we construct a multi-layer learning structure, which is sketched in Figure 3.2. In this structure, the top layer is a visual graph, the bottom layer is a textual graph, and the middle layer shows an abstract fusion



layer for the visual and textual graphs to communicate with each other during the learning process. The reason behind is that empirically there is often a high correlation between the similarity of two images in the visual graph and that of the two images in the textual graph. With the fusion layer, the two graphs can supervise and guide each other to achieve a high-quality learning result.

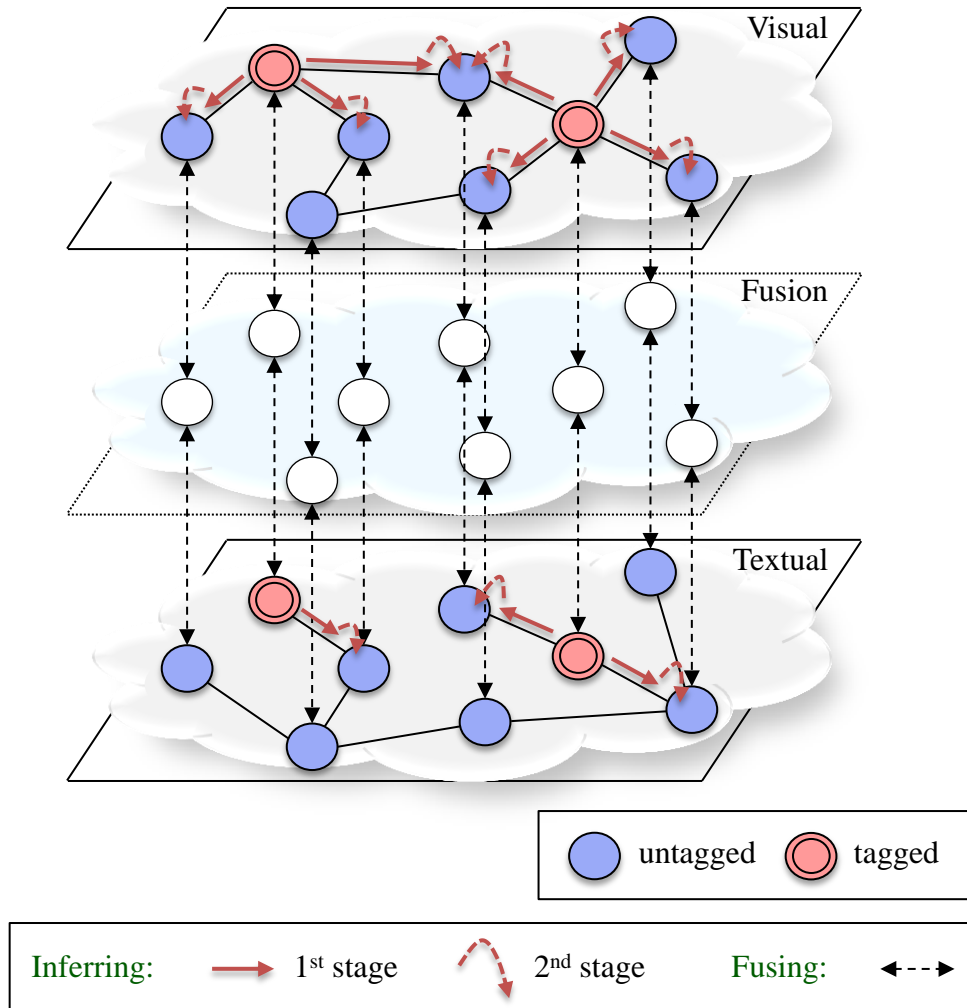
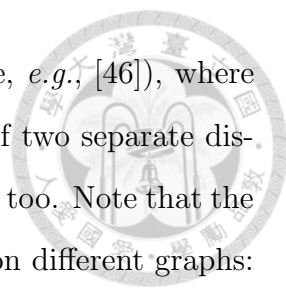


Figure 3.2: Multi-layer learning structure. The top layer is a visual graph, the bottom layer is a textual graph, and the middle layer shows an abstract fusion layer for the communication of the visual and textual graphs. Note that initial labels are shown in double circles (light red). Graph structure can also benefit from the MapReduce model by only propagating and receiving information from the neighbors.

Our multi-layer multi-label propagation algorithm is an iterative process. We assign some initial tags as ground truth in our algorithm. Each iteration consists of



two inferring processes and one fusion process (linear fusion, see, *e.g.*, [46]), where an inferring process is a two-stage process, which is composed of two separate distributed algorithms; the fusion process is a distributed algorithm, too. Note that the actions of the two inferring processes are the same but applied on different graphs: one is on the visual graph, and the other is on the textual graph. An inferring process is activated to propagate the label information (*e.g.*, a value) from each labeled vertex to its adjacent vertices. More precisely, the first stage of an inferring process weights the adjacent edges of each vertex using the label information, while the second stage collects the weighted value(s) to the vertices pointed to (*cf.* Figure 3.2). After the inferring processes, the fusion process is activated. The fusion process fuses every two vertices in the different graphs but referring to the same image, in a weighted fashion with a user-specified parameter $\beta \in (0, 1)$ (*cf.* Eq. (3.2), *e.g.*, $\beta = 0.5$ for averaging) so that the two graphs can communicate with each other. Finally, both visual and textual graphs are updated by assigning each fused value back to the two corresponding vertices. The overall learning process is repeated until a certain number of iterations have been reached.

3.3.2 Multi-Layer Multi-label Propagation Algorithm

This subsection details our multi-layer multi-label propagation algorithm. Our algorithm is extended from [57], targeting at multi-label propagation for large-scale image datasets. For simplicity, we shall concentrate on the operations in the visual graph, and shall not mention those in the textual graph, except when we explain the fusion process. Note that each vertex may have tags or no tag at all.

In the sequel, let G be the visual graph with vertex set $\mathcal{X} = \{x_1, \dots, x_n\}$, where n is the vertex number; let $W \in \mathbb{R}^{n \times n}$ be the similarity matrix of G , where W_{ij} denotes the edge weight between vertices x_i and x_j ; let $D \in \mathbb{R}^{n \times n}$ be a diagonal matrix with D_{ii} equaling the inverse square root of the sum of the i th-row of W ; let $\mathcal{L} = \{1, \dots, c\}$ be the label set of G ; let $Y \in \mathbb{R}^{n \times c}$ be a matrix with $Y_{ij} = 1$ if x_i is initially labeled as j , and $Y_{ij} = 0$ otherwise; let $F \in \mathbb{R}^{n \times c}$ be a classification

matrix, where eventually each vertex x_i will be assigned tag(s) by tag refinement (see Section 3.3.4) based on the i -row of F ; let $\alpha \in (0, 1)$ be a user-specified parameter. Without loss of generality, assuming that initially the first l (typically, $l \ll n$) vertices $\{x_1, \dots, x_l\}$ are individually labeled, and the other $(n - l)$ vertices $\{x_{(l+1)}, \dots, x_n\}$ are unlabeled. Our objective is to infer the labels of the unlabeled vertices. Note that in general both S and Y are sparse.

The propagation process is as follows. Initially, we assign Y to F . Then we create a normalized weight matrix S , which equals the product of D times W times D (*i.e.*, $S = D \times W \times D$). Next, we activate an iterative process, iteratively performing the following two operations.

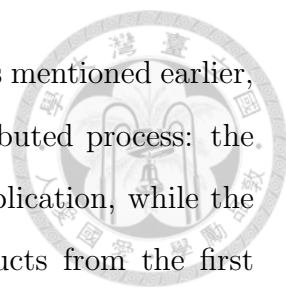
$$F^{((i+1),vis)} = \alpha S \times F^{(i,vis)} + (1 - \alpha)Y, \quad (3.1)$$

$$F^{((i+1),vis)} = \beta F^{((i+1),vis)} + (1 - \beta)F^{((i+1),txt)}, \quad (3.2)$$

where the meaning of a matrix with the superscript “ (i, vis) ” is twofold: obtained in the i -th iteration, and used for learning on the *visual* graph (*i.e.*, $F^{(0,vis)} = F$). Similarly, $F^{((i+1),txt)}$ is the classification matrix for the *textual* graph obtained in the $(i + 1)$ -th iteration. Note that β is the fused parameter (*cf.* Section 3.3.1). As aforementioned, the physical meanings of the two are to infer labels and then to fuse with the textual graph. Note that fusion (Eq. (3.2)) in the i -th iteration can only be done after inferring labels (Eq. (3.1)) in both of the visual graph and textual graphs in the i -th iteration. The iterative process is repeated until a certain number of iterations is reached. We will show how to decide the iteration number in Section 3.3.3.

Now we present our implementation for Eq. (3.1), *i.e.*, the inferring process. Note that both S and Y are invariant during the whole propagation process. We substitute S' for αS and Y' for $(1 - \alpha)Y$, *i.e.*, $S' = \alpha S$ and $Y' = (1 - \alpha)Y$. Clearly, S' (Y') is sparse if S (Y) is sparse. Therefore, we rewrite Eq. (3.1) as

$$F^{((t+1),vis)} = S' \times F^{(t,vis)} + Y', \quad (3.3)$$



which contains a matrix multiplication and a matrix addition. As mentioned earlier, we implemented such an inferring process by a two-stage distributed process: the first stage performs the multiplication part of the matrix multiplication, while the second stage performs the addition concurrently for the products from the first stage and for the matrix addition. The two stages are described in Figure 3.3 and Figure 3.4, sequentially. Note that we store each matrix based on a coordinate list representation: each line is associated to a row-column index pair and a nonzero weight.

The rationale behind Figure 3.3 is that given an operation, S' times F , each entry of the i -th column of S' should and would be multiplied by each entry of the i -th row of F exactly once. This can be done by assigning proper key values in the *map* procedure followed by performing pairwise multiplications in the *reduce* procedure. Subsequently, in Figure 3.4 we add the resultant entries from Figure 3.3 and the entries of Y' together according to their row and column indexes. (Two entries can only be added if they have the same row and column indexes.) Note that in practical implementation, the *reduce* procedure can also work as a combiner of the MapReduce programming model. In addition, it is instructive to note that if in Figure 3.4 we consider only the resultant entries from Figure 3.3, the two-stage process will become a generalized version of the matrix-vector multiplication approach [23] for matrix-matrix multiplication. So far we have presented the implementation of the inferring process. In particular, the fusion process can also be achieved in a similar way as in Figure 3.3, where the fusion parameter β is injected in the *map* procedure.

3.3.3 Convergence Criterion of Label Propagation

The issue remains in label propagation is to decide the iteration number as the criterion of convergence. This is important for a distributed environment because it is hard to inspect each subgraph individually. We will also show experimentally in Section 3.4.5 the necessity to determine the iteration number automatically and globally. Intuitively, this number must be large enough such that each image can



```
1: Procedure Map(String key, String value)
2: Input:
   key: file name of S', file name of F;
   value: a matrix entry;
3: Begin
4:   (rowId, colId, weight) = ParseLine(value);
5:   if IsSbarFile(key) is true
6:     newKey = colId;
7:     newValue.setTriple('S', rowId, weight);
8:   else
9:     newKey = rowId;
10:    newValue.setTriple('F', colId, weight);
11:  end if
12:  Emit(newKey, newValue);
13: End

14: Procedure Reduce(String key, Iterator values)
15: Begin
16:   ArrayList arrayListS, arrayListF;
17:   foreach val in values
18:     if val.first is 'S'
19:       arrayListS.addPair(val.second, val.third);
20:     else
21:       arrayListF.addPair(val.second, val.third);
22:     end if
23:   end foreach
24:   foreach dataS in arrayListS
25:     (rowId, weightS) = dataS.getPairData();
26:     foreach dataF in arrayListF
27:       (colId, weightF) = dataF.getPairData();
28:       Emit(rowId+" "+colId, weightS×weightF);
29:     end foreach
30:   end foreach
31: End
```

Figure 3.3: The first stage of an inferring process with MapReduce.

propagate its label information to those images related to it. That is, a vertex u should propagate its label information to every vertex v during label propagation if there is a path from u to v . To this end, we first replace each edge weight by one. Note that our graph did not reserve the zero-weight edges before; the edges


```

1: Procedure Map(String key, String value)
2: Input:
   key: file names;
   value: a matrix entry;
3: Begin
4:   (rowId, colId, weight) = ParseLine(value);
5:   Emit(key, weight);
6: End

7: Procedure Reduce(String key, Iterator values)
8: Begin
9:   sum=0.0;
10:  foreach val in values
11:    sum += val;
12:  end foreach
13:  Emit(key, sum);
14: End

```

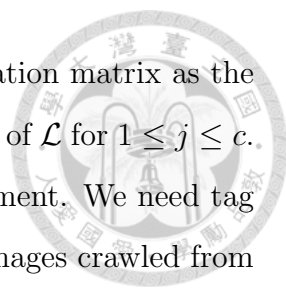


Figure 3.4: The second stage of an inferring process with MapReduce.

considered here are all positive-weight edges. By doing so, the original problem can then be transformed into the finding of maximum path length from the shortest paths between all pairs of vertices in the graph, where each missing edge represents an infinite distance. Since the inferring process introduced before in nature realizes a matrix multiplication, we thus find the all-pairs shortest paths using dynamic programming, *i.e.*, the matrix multiplication-based approach [13]. Once the all-pairs shortest paths have been identified, we can find the maximum path length accordingly. Note that the processes described above can be implemented following the ideas of Figure 3.3 and Figure 3.4 with small modifications.

3.3.4 Tag Refinement

In this subsection, we present the tag refinement approach used in this chapter, which is applied after label propagation. Note that at this moment, the classification matrices in both the visual graph and the textual graph are the same. We take any of them as the resultant classification matrix. In the sequel, we define the



transpose matrix of the i th-row matrix of the resultant classification matrix as the tag matrix $T_i \in \mathbb{R}^{c \times 1}$, where $(T_i)_{j1}$ is the score of image i to tag j of \mathcal{L} for $1 \leq j \leq c$. Before proceeding, let us explain the reason for using tag refinement. We need tag refinement to improve image tags learned because the original images crawled from an image dataset, such as Flickr, without any pre-processing may have some noisy tags. In other words, some tags may not properly (or even not correctly) interpret the images. Undoubtedly, such noisy tags may introduce noises into the learning results after label propagation. As a solution, the main objective of tag refinement is to effectively suppress the noisy tags while emphasize the others. That is, given an image with a tag list, we would like to reorder the list such that tag i precedes (*i.e.*, gets a higher rank) tag j if tag i is more relevant to the image than tag j . Figure 3.5 gives two examples². Each image contains two list of tags with different orders, where the upper (lower) list is the tag list before (after) tag refinement. Consider the left image. We can see that after tag refinement, the tag list is sorted according to their relevance to the image. For example, tag “Tower of Pisa” is placed at the first in the tag list. This means that “Tower of Pisa” is the most-relevant tag to the image here. The right image is the same. The most-relevant tag, “Golden Gate Locale,” in the tag list is placed at the first in the tag list after tag refinement.

Recently, a promising work for tag refinement was presented in [59], where Zhu *et al.* proposed to decompose a user-provided tag matrix into a low-rank refined matrix and a sparse error matrix. The work in [59] is effective for a global tag refinement; however, it may not be appropriate to our problem here as each entry in our tag matrices represents a score and our objective is to tune the scores locally (a global tag refinement may obscure the experimental results of our proposed approaches). As a result, we modified the approach of learning class-specific weights for the searching of visually similar images in [22] to fit our needs. Without loss of generality, assume there is a union set of vertices $\{x_1, x_2, \dots, x_{(k-1)}\}$ which are adjacent to a target vertex x_k in the visual or textual graphs. Let $\text{adjVis}(x_k)$ and

²All of the photos shown in this figure were obtained from Flickr [1].



Tags (before):
Italy, tower, Tower of Pisa, Pisa,
Roman, Pisa tower,

Tags (after):
Tower of Pisa, Pisa tower, Pisa,
tower, Roman, Italy

(a)



Tags (before):
California and Marin County, traveling,
tourists, Golden Gate Locale, SFO,
bridge, California, San Francisco,
vacation

Tags (after):
Golden Gate Locale, San Francisco,
SFO, bridge, California and Marin
County, California, traveling, tourists,
vacation

(b)

Figure 3.5: Illustration of tag refinement. After tag refinement, the tag order of a tag list is changed while the number of tags is the same. A tag in the tag list of an image precedes if it is more relevant to the image than the tag(s) placed behind.

$\text{adjTxt}(x_k)$ be the vertices adjacent to x_k in the visual and textual graph, respectively, *i.e.*, $\text{adjVis}(x_k) \cup \text{adjTxt}(x_k) = \{x_1, x_2, \dots, x_{(k-1)}\}$. Let A_1, A_2, \dots, A_k be the weight vectors (*i.e.*, column matrices), where $A_i \in \mathbb{R}^{c \times 1}$, $1 \leq i \leq k$. We consider the proximity between x_i and x_j by the similarity between them, for $1 \leq i, j \leq k$ and $i \neq j$, to preserve the inter-image relationship. Finally, our formulation is as follows (*cf.* Eqs. (2)–(5) of [22]).

$$\begin{aligned} \text{Minimize} \quad & \gamma \sum_{x_i \in \text{adjVis}(x_k)} (W_{ik}^{\text{vis}} \|A_i \circ T_i - A_k \circ T_k\|^2) + \\ & (1 - \gamma) \sum_{x_i \in \text{adjTxt}(x_k)} (W_{ik}^{\text{txt}} \|A_i \circ T_i - A_k \circ T_k\|^2) \end{aligned}$$

$$\text{subject to} \quad A_i^\top \times \mathbf{1} = 1, i = 1, \dots, k, \quad (3.4)$$

$$A_i \geq \mathbf{0}, i = 1, \dots, k, \quad (3.5)$$

where $\gamma \in (0, 1)$ is a parameter that controls the balance of the visual and textual graphs, W_{ik}^{vis} (W_{ik}^{txt}) is the edge weight between x_i and x_k in the visual (textual) graph, the symbol “ \circ ” denotes element-wise (Hadamard) product, the superscript “ \top ” denotes the transpose operation, and $\mathbf{1}$ ($\mathbf{0}$) denotes a $c \times 1$ matrix filled with ones (zeros). For the objective function to one graph, the term $\|\cdot\|^2$ shows a weighted distance between two images, which is further weighted by the similarity between the two corresponded images. For the constraints, Constraint (3.4) enforces the summation of the entries of each weight vector to be one, while Constraint (3.5) enforces every entry of a weight vector to be nonnegative. As a result, we can learn k weight vectors, in which $A^{(k)}$ is used to concurrently suppress the noisy tags and emphasize the other tags of our target image, x_k , by weighting $T^{(k)}$ in element-wise. Note that if a normalized result is preferable, we may rewrite Constraint (3.4) as follows:

$$A_i^\top \times T_i = 1, i = 1, \dots, k. \quad (3.6)$$



3.4 Experiments

This section evaluates the effectiveness of the proposed multi-label graph-based SSL approach and the necessity of tag refinement. We implemented our approaches using the Java programming language. The experiments were conducted on a middle Hadoop cluster (version 0.20.1) consisting of 24 commodity machines. The rest of this section is organized as follows. Section 3.4.1 presents the input datasets and explains the basic experiment setup. Section 3.4.2 introduces the evaluation criteria. Section 3.4.3 compares our multi-layer multi-label propagation with four traditional approaches. Section 3.4.4 investigates the effectiveness of tag refinement. Section 3.4.5 evaluates the proposed convergence criterion for label propagation. Finally, Section 3.4.6 performs the sensitivity tests for the three parameters used in this chapter.

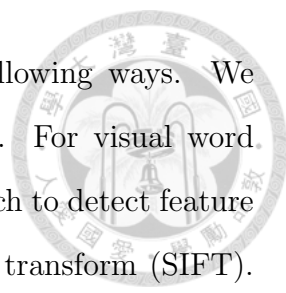
3.4.1 Experimental Setup

All of the experiments were based on the following two image datasets.

Flickr550K: Flickr550K contains 540,321 images with 9,360 manually-annotated ground truth images in 21 query categories [53]. (Note that in [53] this dataset was named Flickr550. We use Flickr550K instead to be consistent with Flickr11K introduced in the following.)

Flickr11K: Flickr11K is made as a specific subset of Flickr550K [26]. It contains 11,277 medium resolution (500×360) images with 1,282 ground truth images in seven query categories.

For both Flickr11K and Flickr550K, each image is presented in visual and textual high-dimensional features. For each dataset, we consider the 1,282 ground truth images in the seven query categories, which are colosseum, eiffel tower, golden, starbucks, torre pendente di pisa, tower bridge, and triomphe. Note that although Flickr550K is with 21 query categories, we focus on the seven out of the 21 query categories.



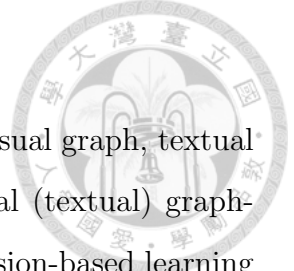
We extracted the visual and textual features in the following ways. We took *visual word* as visual features for similarity computation. For visual word generation, we adopted the difference-of-Gaussian (DoG) approach to detect feature points followed by describing them with scale invariant feature transform (SIFT). The SIFT descriptors were then quantized into 10k clusters using k -means clustering, where each cluster defined a visual word containing the feature descriptors (feature points) in this cluster. For the textural features, we adopted the expanded Google snippet from the Google search engine to perform query expansion to represent associated (noisy) tags in textual features in 91,004 dimensions.

To be close to the real world image data (*i.e.*, relatively few tagged images), for all of the textual graphs used, we reserved only a portion of tagged images by randomly sampling, while removed the tags from the other tagged images. For graph construction, we reserved only 30% tagged images by randomly sampling from each dataset to fit the real world condition. For label propagation, the focus of this chapter, we further reduced the number to 100 to show the effectiveness of our propagation method. That is, Flickr11K and Flickr550K reserved only 0.89% and 0.02% tagged images (randomly sampled), respectively.

Since general graph-based SSL is sensitive to the initial labels, for each experiment we repeated the learning process ten times and average the results for a more accurate test result.

3.4.2 Evaluation Criteria

We evaluate the performance by *Mean Average Precision (MAP)*, which is the mean of the average precision of the tags for each image. That is, a higher MAP indicates a better retrieval results. For tag refinement, we focus on the precision of the top- n labels of the ranked label set of each image, *i.e.*, P@ n . Empirically, parameters α , β , and γ were set to 0.9, 0.5, and 0.6, respectively. Note that the sensitivity test of each of them can be found in Section 3.4.6.



3.4.3 Multi-Layer Multi-Label Propagation

This subsection compares our multi-layer learning with visual graph, textual graph, early fusion and late fusion -based learning, where visual (textual) graph-based learning uses no textual (visual) graph, and early (late) fusion-based learning considers both visual and textual graph but fuses only at the beginning (end). All of the label propagations here were implemented based on our multi-label propagation approach described in Section 3.3.2 to tackle the large-scale dataset, *i.e.*, Flickr550K. The results are shown in Table 3.1. As revealed in the table, our multi-layer learning can significantly improve the baselines by aggregating visual and textual contexts during message passing among the graphs. Moreover, the results also verify that our multi-label propagation approach is practical and useful to a large-scale image dataset.

Table 3.1: Compare the visual graph, textual graph, early-fusion [48], late-fusion [48], and multi-layer (ours) -based learning methods in MAP, where the percentages (%’s) are the improvement ratios between the visual graph only and the multi-layer learning methods. We can see that the multi-label method can achieve better results than the others.

	Visual Only	Textual Only	Early Fusion	Late Fusion	Multi-layer (Ours)
Flickr11K	0.2859	0.3083	0.3114	0.3418	0.3611 (26%)
Flickr550K	0.1632	0.1994	0.2149	0.2342	0.2883 (77%)

After multi-label propagation, each image contains a set of tags, where each tag is with a score. With this information, we then do tag refinement to suppress noisy tags.

3.4.4 Tag Refinement

As mentioned in Section 3.4.2, here we investigate the precision of the top- n labels of the ranked label set of each image, *i.e.*, P@n. Table 3.2 shows the experimental results. We consider the top-1, 2, 3, 4, 5, and 10 labels. As can be seen, the results of using tag refinement are consistently better than using no tag

refinement. Based on the results, we further believe that the noisy tags indeed may lower the learning quality a lot. With tag refinement, such noises are effectively suppressed by considering the neighbor information of an image graph.

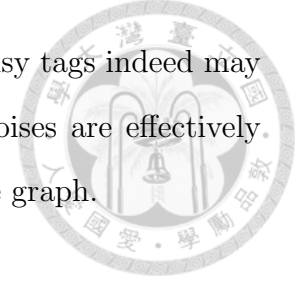


Table 3.2: Compare the use of tag refinement and not in MAP, where the resultant top-1, 2, 3, 4, 5, and 10 labels are considered.

Flick11K	P@1	P@2	P@3	P@4	P@5	P@10
Multi-label w/o Refinement	0.0016	0.0059	0.1059	0.1809	0.3029	0.3548
Multi-label w/ Refinement	0.0842	0.1033	0.1348	0.2948	0.3541	0.3611
Flick550K	P@1	P@2	P@3	P@4	P@5	P@10
Multi-label w/o Refinement	0.0003	0.0189	0.1093	0.1777	0.2574	0.2680
Multi-label w/ Refinement	0.0546	0.1098	0.1357	0.1867	0.2883	0.3271

3.4.5 Convergence Criterion - Shortest Path

As described in Section 3.3.2, we set a maximum number of iterations as the criterion of convergence for label propagation. Such a number is defined as the maximum path length from the shortest paths between all pairs of vertices in the graph such that each image can propagate its label information to any images related to it. At this moment, we compare the shortest path-based approach and an ad-hoc approach which arbitrarily sets the maximum number of iterations. The experimental results are shown in Table 3.3. Each of the numbers, 1, 5, 10, 50, and 100, denotes the maximum number of iterations adopted. Note that the maximum numbers of iterations to Flickr11K and Flickr550K are 13 and 15, respectively. As we can see, our shortest path-based approach can achieve the best results. From the results, it is interesting to notice that using more than enough iteration may degrade the performance. This may be because of the magnification of the influence of the potential noisy tags.

Table 3.3: Compare the shortest path-based approach and an ad-hoc approach in MAP. Based on the results, we can see that the shortest path-based results are comparable to those by the ad-hoc approach.

	Ad-Hoc Approach					Shortest Path-Based Approach
	1	5	10	50	100	
Flickr11K	0.3297	0.3330	0.3504	0.3473	0.3455	0.3611
Flickr550K	0.2544	0.2629	0.2680	0.2643	0.2697	0.2883

3.4.6 Sensitivity Test

In this subsection, we conduct experiments to decide the three user-specified parameters used in this work, *i.e.*, α , β , and γ . They are for the inferring process, the fusion process of label propagation, and the balance control of tag refinement, respectively. The sensitivity tests of them are as follows. Table 3.4 investigates the use of different α in the inferring process. As can be seen, when α equals to 0.9, we can achieve the best MAP. Thus we set α to 0.9. Table 3.5 investigates the use of different β in the fusion process. As can be seen, the best setting of β is 0.5. This result also reflects the necessity of fusing the visual and textual graphs to achieve a better MAP. They are both important. No matter we bias toward any of them, the resultant MAP would get worse. Table 3.6 investigates the use of different γ in balance control. As revealed in the table, the best setting of γ is 0.6. Consequently, parameters α , β , and γ were set to 0.9, 0.5, and 0.6, respectively.

Table 3.4: Consider the setting of α (*cf.* Eq. (3.1)), *i.e.*, the inferring parameter in the graph-based multi-layer multi-label propagation in MAP.

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Flickr11K	0.0097	0.1044	0.1504	0.1695	0.1766	0.2112	0.2373	0.2501	0.2554
Flickr550K	0.0999	0.1203	0.1400	0.1504	0.1655	0.1799	0.1867	0.2003	0.2237



Table 3.5: Consider the setting of β (cf. Eq. (3.2)), *i.e.*, the fusion parameter in the graph-based multi-layer multi-label propagation in MAP.

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Flickr11K	0.2113	0.2367	0.2548	0.2481	0.2677	0.2499	0.2410	0.2158	0.2035
Flickr550K	0.2044	0.2130	0.2205	0.2311	0.2359	0.2245	0.2139	0.2008	0.1917

Table 3.6: Consider the setting of γ (cf. Eq. (3.4)), *i.e.*, the balance control parameter in tag refinement in MAP.

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Flickr11K	0.2572	0.2628	0.2848	0.3066	0.3244	0.3620	0.3502	0.3115	0.2753
Flickr550K	0.1977	0.2046	0.2283	0.2380	0.2508	0.2737	0.2494	0.2229	0.2002



Chapter 4

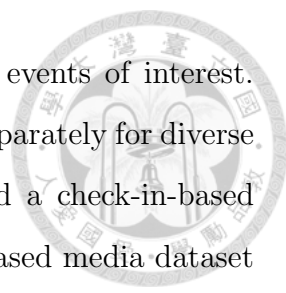
Cross-Domain Media Learning for Event Discovery

4.1 Introduction

Every day, millions (and even more) of media data are uploaded to various social-sharing websites. As the volume of media data increases, there have been many studies in the literature that aim to discover and/or summarize media datasets for useful information. Most previous works focused on the mining of a single media dataset for specific applications.

Previously, based on Twitter tweets, Sankaranarayanan *et al.* build a news processing system to capture late breaking news [42]. Sakaki *et al.* build a reporting system for earthquake detection [41]. Weng *et al.* then build a system, called Voters' Voice, that effectively summarized netizens' discussion for Singapore General Election [49]. Recently, Meladianos *et al.* extracted sub-events for evolving events, *e.g.*, natural disasters [36]. Based on taxis traces, Zhang *et al.* identified the place and time of events that happened and the scale of events [56]. For complex events in videos, Yan *et al.* discovered videos of particular events from internet video archives [52]. Chang *et al.* studied video ranking for specified events, where no training data is required [10].

Intuitively, different media datasets have their own features and information. For example, Kuo *et al.* observed that most Instagram users like to share information of food and travel, while many Twitter users like to share more for sports and news [27]. Similarly, Becker *et al.* also considered multiple media datasets, including Twitter, YouTube, and Flickr, and then provided diverse media contents for target events [8].

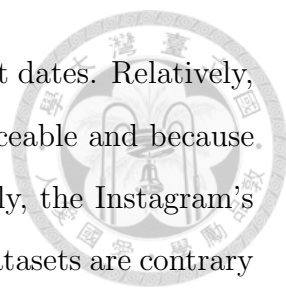


In contrast, this chapter unifies two media datasets for events of interest. Unlike related works, *e.g.*, [27], that explored multiple datasets separately for diverse aspects, this chapter combines a flow-based media dataset and a check-in-based media dataset seamlessly for high search performance. A flow-based media dataset has flow information for some locations. The dataset can offer the numerical value of something per unit of time for each of the locations (*c.f.*, Section 4.2). A check-in-based media dataset has user-contributed information for some locations, *e.g.*, comments and images. Practically, we used a flow dataset of taxis and a check-in dataset from Instagram. We collected data for the City of Sapporo from March to November in 2014, where the taxi’s dataset was created using GPS data from about 10,000 taxis. Note that we did not use another check-in-based media dataset, *e.g.*, Twitter, because Instagram not only has textual data but only a large number of images for future studies. Further, it is more convenient to us for long-term data collection. However, the methods presented in this chapter can be easily extended to many other check-in-based media datasets with textual data. Table 4.1 compares the two media datasets used throughout this chapter.

Table 4.1: Comparison of a flow dataset of taxis and an Instagram’s check-in dataset for the City of Sapporo.

	Taxis (flow dataset)	Instagram (check-in dataset)
Information	time, places, values	time, places, texts images, values
Hotspots / Weakspots	weekdays / holidays	holidays / weekdays
Reaction	relatively instantaneous	relatively non-instantaneous
Distribution	relatively gathered	relatively scattered
Coverage from the other dataset	61%	47%

As can be seen, the strengths of the two datasets are different but complementary. Both the taxi’s dataset and the Instagram’s dataset have numerical data, *i.e.*, number of boarding and alighting for the taxi’s dataset, and number of check-

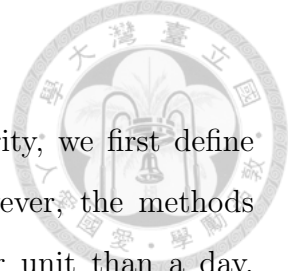


ins for the Instagram’s dataset, for different locations for different dates. Relatively, the numerical data for the taxi’s dataset is more *steady* and traceable and because they were collected from the same set of taxis, while additionally, the Instagram’s dataset have textual data and images. The hotspots of the two datasets are contrary to each other, and thus their data volumes are complementary. Moreover, the reaction to events is instantaneous for the taxi’s dataset, and the distribution of taxis’s boarding and alighting locations are gathered. Compared with the taxi’s dataset, the reaction to events for the Instagram’s dataset are not instantaneous enough (*c.f.*, Section 4.4.3.1), and the distribution of check-in locations are scattered. Further, about half of the boarding and alighting locations and the check-in locations are in a close proximity (*c.f.*, Section 4.4.2).

As the two datasets are complementary, we intend to unify them for high search performance in finding events of interest. Overall, the major contributions of this chapter are summarized as follows:

- To the best of our knowledge, this chapter presents the first work to combine a flow-based media dataset and a check-in-based media dataset seamlessly for events of interest.
- This chapter presents a generic two-stage framework that normalizes data from different sources and then combines the normalized data by effective graph algorithms.
- Experimental results show that our approach is effective to normalize data from different datasets, and is able to achieve high performance for a ranked list of events.

The rest of this chapter is organized as follows. Section 4.2 formulates the problem of finding events of interest. Section 4.3 details the proposed approach. Finally, Section 4.4 evaluates the performance of the proposed approach.



4.2 Problem Formulation

This section presents the problem formulation. For clarity, we first define *a day* as the smallest unit of time for event discovery. (However, the methods presented in this chapter can be easily extended for a smaller unit than a day, *e.g.*, an hour.) With the definition, we then give formal descriptions of two terms, *i.e.*, *flow data* and *check-in data*, as follows.

Definition 1 (*Flow Data*) *Given a location for a period of time, flow data of the location can offer the numerical value of something per day at the location.*

For example, for flow dataset of taxis, flow data may offer the number of boarding and alighting from taxis per day at a location.

Definition 2 (*Check-in Data*) *Given a location for a period of time, check-in data of the location can offer a set of tags and/or comments per day at the location from users of social networks.*

Note that check-in data may offer more information than tags and comments, *e.g.*, images. This chapter will focus on textual information from tags and comments (although textual information could be extracted from images). Finally, our problem of finding events of interest of a region can be stated as follows.

Problem 1 *Given a set of flow data of locations and a set of check-in data of locations for a period of time of a region, the objective of the problem of finding events of interest of a region is to create a ranked list of events of the region, based on the given sets of data.*

Practically, we used (1) a flow dataset of taxis and (2) an Instagram's check-in dataset, for events of interest in the City of Sapporo. Most of events in the experiments are either sports or local festivals for the datasets (despite the fact that the application of our approach is not limited to sports and festivals). In fact, the two

datasets are not the original datasets, but the resultant datasets after pre-processing for ease of use. More specifically, the original flow dataset of taxis consists of a list. Each item of the list keeps (a) a boarding location, (b) an alighting location, and (c) the timing information *e.g.*, the date. The original Instagram’s check-in dataset also consists of a list. Each item of the list keeps (a) a check-in location, (b) a comment from a user, and (c) the timing information of the check-in. Figure 4.1 gives an illustration, where there are two items for the original flow dataset and two items for the original Instagram’s dataset.

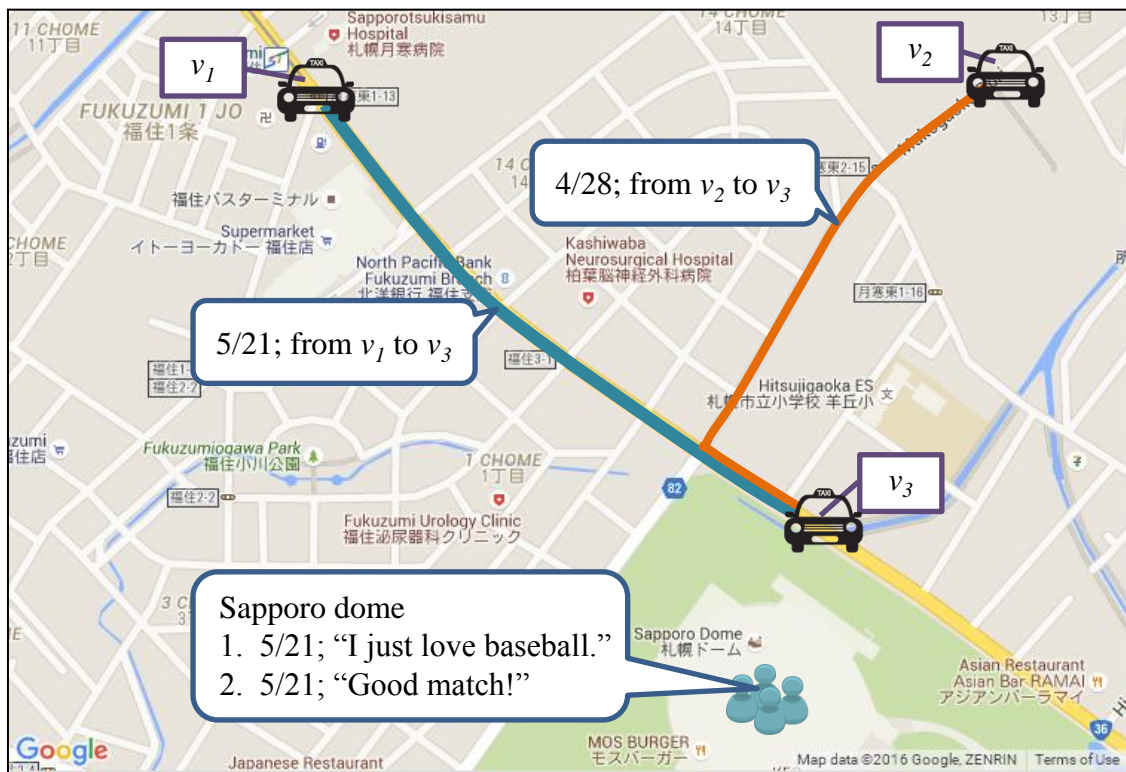


Figure 4.1: Illustration of two items listed in the original flow dataset of taxis, and two items in the original Instagram’s dataset that are associated to Sapporo Dome. The two datasets can be processed for a flow dataset with a set of flow data, and a check-in dataset with a set of check-in data.

We used the datasets that had been pre-processed. That is, the flow dataset can be accessed as a set of flow data, and the Instagram’s check-in dataset can be accessed as a set of check-in data. Note that the locations with flow data and those with check-in data may differ (as indicated in Table 4.1). In addition, our

approach is capable of handling many other flow-based media datasets or check-in-based media datasets, not limited to the taxi’s dataset and the Instagram’s check-in dataset.



Based the given datasets, this work will create a ranked list of events. Table 4.2 illustrates the idea of a ranked list of two events in Sapporo. The list implies that there could be two popular events: one is a display of fireworks in Moerenuma Park on September 20, and the other is a baseball game for Nipponham fighters in Sapporo Dome on May 21. Besides, more information may be obtained by the users’ comments that are associated to the events. In the following, we will describe how to generate the ranked list of events.

Table 4.2: Illustration of a ranked list of events, where “Cmnt” is an abbreviation of “Comment.” Two items (*i.e.*, rows) are listed, where each item is associated to an event. Each item gives the date, the location, the relevant terms, and users’ comments which the relevant terms are derived from, for the event.

Rank	Date	Location	Term	Cmnt
1	9/20	Moerenuma Park	fireworks, art	...
2	5/21	Sapporo Dome	baseball, match Nipponham	...

4.3 Proposed Approach

Figure 4.2 outlines the proposed approach for finding events of interests. Given a set of flow data of locations and a set of check-in data of locations, the proposed approach normalizes the given sets of data, followed by combing the normalized datasets for a ranked list of events. More specifically, the proposed approach uses six techniques, including (a) flow normalization, (b) buzz-score calculation, (c) variability-score calculation, (d) spanning-graph construction, (e) weight determination, and (f) flow propagation. Initially, flow data of each location will be processed by flow normalization and variability-score calculation. Check-in data of each location will be processed by buzz-score calculation and variability-score calculation. Subsequently, the resultant datasets will be linked by spanning-graph construction

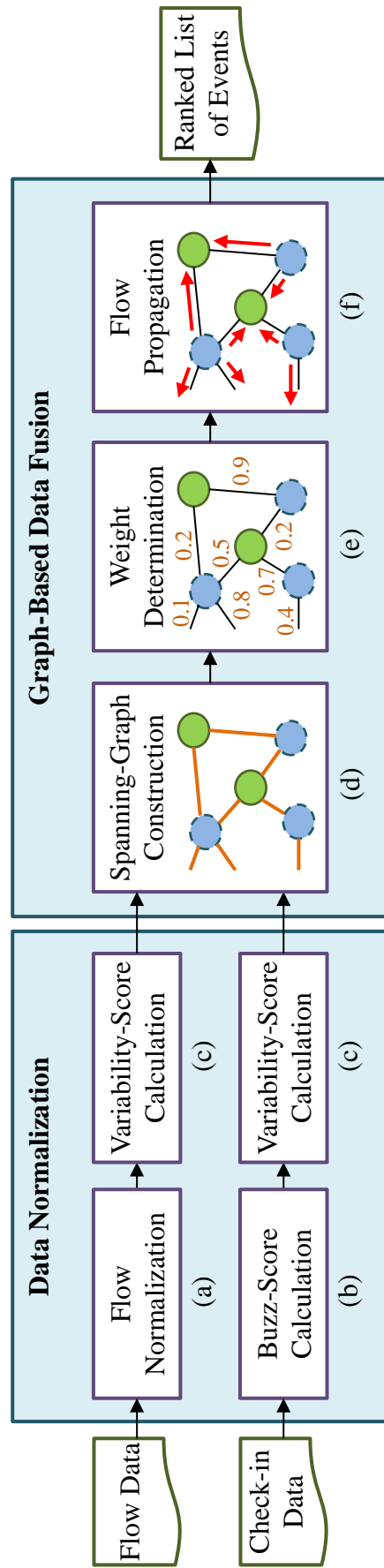


Figure 4.2: Overall flow of the proposed approach for finding events of interest. The framework of the proposed approach has two stages, including data normalization and graph-based data fusion. Note that before data fusion, data normalization is essential because the given datasets are different in nature. Specifically, the approach uses six techniques, including (a) flow normalization, (b) buzz-score calculation, (c) variability-score calculation, (d) spanning-graph construction, (e) weight determination, and (f) flow propagation. For these techniques, the first three are used for data normalization, and the other three are used for data fusion. Overall, the given datasets will be normalized individually, and then combined together by graph-based techniques.



and weight determination. Finally, the datasets will be combined by flow propagation for a ranked list of events.

In the sequel, Section 4.3.1 to Section 4.3.6 will present the aforementioned six techniques, respectively.



4.3.1 Flow Normalization

Given a location, flow data of the location can offer the numerical value of something per day at the location. For the taxi's dataset, flow data can offer the number of boarding and alighting from taxis per day at the location. The number of boarding and alighting from taxis at a location changes with the dates. Therefore, finding events of interest at the location should be achieved by observing the changes of the numbers for the location. In this chapter, calculating variability scores helps us observe the changes of the numbers. Details of variability-score calculation will be presented soon in Section 4.3.3.

However, we found that different periods of time may result in quite different numbers of boarding and alighting from taxis at a location. For example, we observed that the average number of boarding and alighting from taxis at the Sapporo station per day in August is typically more than that in November, probably because of the tourist season in the City of Sapporo (August is, but November is not). With such a bias, it is likely that most high-ranking events found by our approach will be in the tourist season or some other specific periods of time. (See Section 4.3.3 for more details.) Thus, we normalize flow data of a location of a day by

$$f_{i,j}^N = \frac{1}{2a+1} \sum_{s=j-a}^{j+a} \frac{f_{i,j}}{f_{i,s}}, \quad (4.1)$$

where $f_{i,j}^N$ is the flow data of location i for date j after flow normalization, $f_{i,j}$ is the given flow data of location i for date j , and $a \geq 1$ is a user-defined parameter that specifies the data range used for normalization. The summation is from a days before date j to a days after date j . Empirically, a is set to 3, and thus the set of flow data for a week will be involved.



Practically, we will normalize the given set of flow data of each location for each date.

4.3.2 Buzz-Score Calculation

Recently, calculating buzz scores has been shown success in trending search based on query logs, see, *e.g.*, [6, 50]. In this chapter, we develop a strategy similar to that in [50] for trending search on check-in data, where buzz-score calculation helps us identify critical terms from comments and/or tags of users.

Consider a location. Check-in data of the location can offer a set of tags and/or comments at the location per day from users of social networks. Given all check-in data for each location, we first remove stop words appeared in the contents of tags and comments. We then split the contents of tags and comments into terms. Given the terms, we extract the top-frequent terms (top-100 terms in our work), and then calculate their buzz scores individually. Assume that there are n terms, *i.e.*, $\{t_1, t_2, \dots, t_n\}$, have been extracted. The same as that mentioned in [50], the buzz score of term t_k of a location of a date for $1 \leq k \leq n$ can be formulated as

$$c_{i,j,k}^B = \sum_{s=j-1}^{j-b} \left(\frac{1}{j-s} (P(t_k|T_j) - P(t_k|T_s)) \right), \quad (4.2)$$

where $c_{i,j,k}^B$ is the buzz score of term t_k of location i for date j , $b \geq 1$ is a user-defined parameter, and $P(t_k|T_j)$ is the probability of the occurrence of term t_k given the set of terms, T_j , for date j . The buzz score of a term for a date helps us measure the degree of rising for the popularity of the term of the date. A term with a high buzz score for a date implies that the frequency of using the term (mostly, mentioned or discussed) has been increasing up to the date. $b \geq 1$ can be used to determine the number of days that will be involved, before the date. Empirically, b is set to 5. Further, we can consider groups of terms with similar semantics, and then simply modify the formulation, as that suggested in [6, 50].

Note that every location has its own top-frequent terms. Practically, we will calculate the buzz score for each top-frequent term of each location for each date. In

addition, it is worth noting that buzz-score calculation and flow normalization (*c.f.*, Section 4.3.1) are similar in the concepts, because they both consider dates other than the date being processed. We thought that calculating buzz scores can also be regarded as a kind of normalization. Thus we will not further normalize check-in data before variability-score calculation.

4.3.3 Variability-Score Calculation

Consider a period of time. Intuitively, finding events of interest should be achieved by observing the changes of data numbers along the dates. Whenever there is a sharp jump on a date, there might be an event that happened on the date. For example, consider a sequence of five numbers, say $\{10, 11, 20, 10, 11\}$. Assume that each number is associated to a date. It is likely that there was an event that happened on the date associated to the third number of the sequence.

Further, we observed that it is preferred to observe the changes of data numbers along each of the seven days of a week, rather than along the dates. The reason is that the change of data numbers may be considerable for different days of a week. For example, the number of boarding and alighting from taxis on weekdays is typically more than that on holidays, based on our dataset. The change along a day of a week is relatively regular, and thus observing the change is relatively effective. As a result, we decompose the set of data for each location of the resultant dataset from Section 4.3.1 into seven groups, which are associated to the seven days of a week, respectively. Similarly, we also decompose the set of data for each location of each top-frequent term of the resultant dataset from Section 4.3.2 into seven groups, which are associated to the seven days of a week, respectively. Eventually, for the part of the flow-based dataset, each group is associated to a location and one of the seven days of a week. For the part of the check-in-based dataset, each group is associated to a location, a term, and one of the seven days of a week.

Practically, we express each group as a sequence of numbers, sorted along the dates. For example, consider a location and a period of time, *e.g.*, a year. For

the flow-based dataset, *i.e.*, the taxi's dataset, the sequence of Sunday will contain the number (after flow normalization) of boarding and alighting from taxis at the location for the first Sunday of the year, that for the second Sunday of the year, and so on. With all the sequences, we will observe the change of numbers in each sequence for event of interest.

For each sequence, we first calculate the mean and the standard deviation. We then define the variability score associated to flow data as

$$f_{i,j}^V = \frac{f_{i,j}^N - \mu_{i,s}^F}{\sigma_{i,s}^F}, s = d(j), \quad (4.3)$$

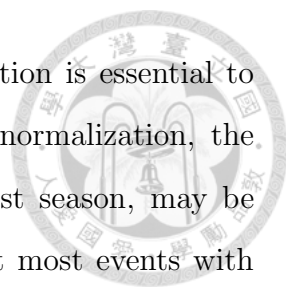
where $f_{i,j}^N$ is the flow data of location i for date j after flow normalization, $f_{i,j}^V$ is the variability score of $f_{i,j}^N$, $d(j)$ gives the day of a week for date j , and $\mu_{i,s}^F$ and $\sigma_{i,s}^F$ are the mean and the standard deviation of the sequence associated to location i of day s of a week for the flow data, respectively. The advantage of calculating variability scores is twofold.

- Every variability score can be used to measure the degree of the variability of the data number for a specific location and a specific date. The larger the absolute value of a score is, the higher the variability of the data is. A positive (negative) score implies that the data number is more (less) than normal.
- Calculating variability scores can be regarded as normalization that eliminates the unit of the given data numbers. It is beneficial for us to combine the flow-based dataset and the check-in-based dataset later.

Similarly, we can define the variability score associated to check-in data as

$$c_{i,j,k}^V = \frac{c_{i,j,k}^B - \mu_{i,s,k}^C}{\sigma_{i,s,k}^C}, s = d(j), \quad (4.4)$$

where $c_{i,j,k}^B$ is the buzz score of term t_k of location i for date j , $c_{i,j,k}^V$ is the variability score of $c_{i,j,k}^B$, $d(j)$ gives the day of a week for date j , and $\mu_{i,s,k}^C$ and $\sigma_{i,s,k}^C$ are the mean and the standard deviation of the sequence associated to term t_k of location i of day s of a week for the check-in data, respectively.



Note that as mentioned in Section 4.3.1, flow normalization is essential to reduce the bias from different periods of time. Without flow normalization, the variability scores of data in some period of time, *e.g.*, a tourist season, may be much higher than those in the other periods of time, such that most events with high rankings may be selected from the period of time (*i.e.*, the tourist season). Similarly, the calculation of buzz scores also helps to reduce such kind of bias.

4.3.4 Spanning-Graph Construction

So far, the set of flow data and the set of check-in data have been normalized. For a high search performance, we intend to bridge the two datasets by graph algorithms. More specifically, consider two types of vertices. Each of the locations to the set of flow data can be modelled as one type of vertex. Each of the locations to the set of check-in data can be modelled as the other type of vertex. Given two vertices with different types, we may add an edge for the two vertices, if the locations associated to them are relevant. We then assign an appropriate weight for each edge. Finally, we may combine the set of flow data and the set of check-in data, by transmitting information along the edges. This section will focus on graph construction, and Sections 4.3.5 and 4.3.6 will present weight determination and information transmission, respectively.

For graph construction, we implement spanning graphs for the two datasets [54, 58]¹. So far, there have been many studies shown that constructing spanning graphs is capable of finding critical neighborhoods of vertices in a plane, see, *e.g.*, [20,21,34]. A spanning graph is an undirected graph that contains no graph loops or multiple edges. Simply, a spanning graph can be constructed as follows. Given a set of vertices in a plane, we may traverse the vertices in any order. Whenever a vertex is visited, we will divide the plane evenly into eight regions with respect to the vertex, followed by connecting the vertex to the nearest vertex to itself in each region.

¹To be self-contained, we introduce the construction of spanning graphs again, despite the fact that the the construction has been presented in Section 2.3.2.

(Note that we may divide the plane evenly into more than eight regions if necessary.) Eventually, we will have a spanning graph for the given set of vertices, when all of the vertices have been visited. Overall, the edge number (*i.e.*, size) of a spanning graph is $O(n)$, if the number of vertices is n . Also, spanning graphs can efficiently be constructed [54]. Thus, constructing spanning graphs is capable of scaling to the case of a large number of vertices.

Figure 4.3 shows an example of spanning-graph construction for four vertices, *i.e.*, v_1 , v_2 , v_3 , and v_4 . Initially, these vertices are disconnected. Assume that v_1 is

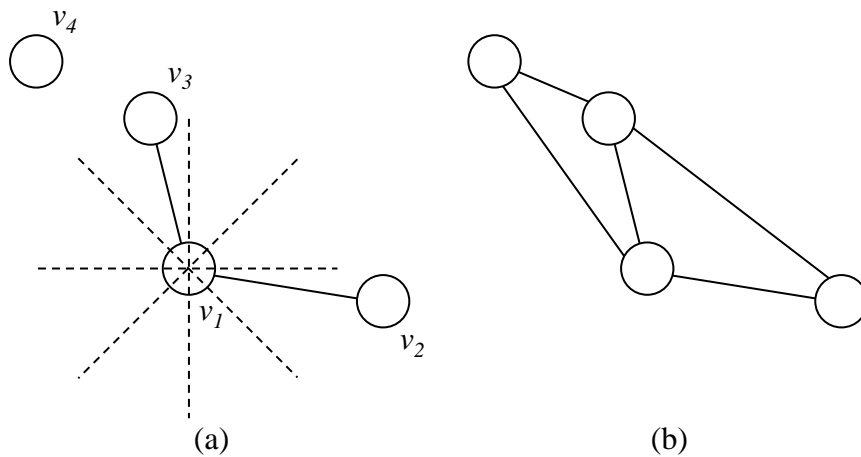


Figure 4.3: Illustration of spanning-graph construction. Consider a set of vertices: v_1 , v_2 , v_3 , and v_4 . (a) Assume that v_1 is first visited. We will divide the plane evenly into eight regions with respect to v_1 . We then add an edge between v_1 and v_2 , and an edge between v_1 and v_3 , because each of v_2 and v_3 is the nearest vertex to v_1 in one of the regions. (b) The spanning graph for the given set of vertices.

first visited. As shown in Figure 4.3(a), v_1 is connected to v_2 and v_3 , because each of v_2 and v_3 is the nearest vertex to v_1 in one of the eight regions with respect to v_1 . Once all of the vertices have been visited, we will have a spanning graph for the four vertices, as shown in Figure 4.3(b).

Note that we intend to combine two types of data numbers, *i.e.*, flows and check-ins, instead of data numbers of the same type. We model each of the locations to the set of flow data as one type of vertex, and each of the locations to the set of check-in data as another type of vertex. We then modify a part of spanning-graph

construction method by restriction as follows. Whenever a vertex v is visited, v will only consider those vertices of the other type with respect to v . By doing so, each edge will only be used to connect vertices of different types. Eventually, we can get a graph for the two datasets. Figure 4.4 shows an example. Without loss of generality,

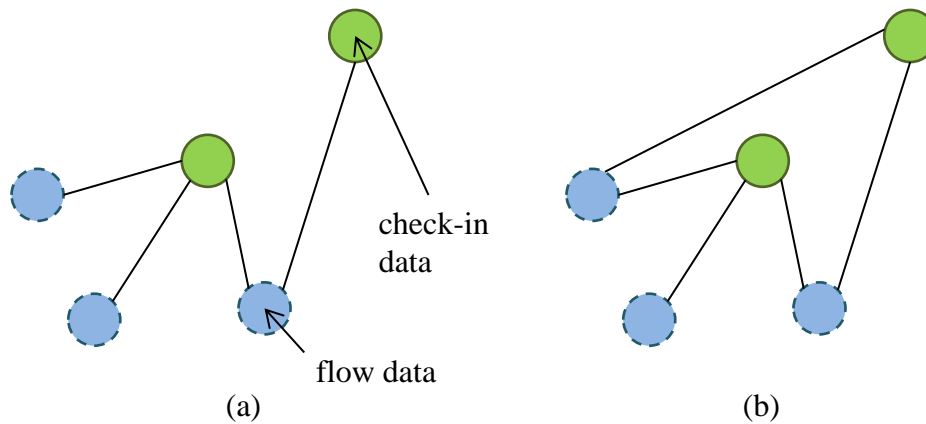
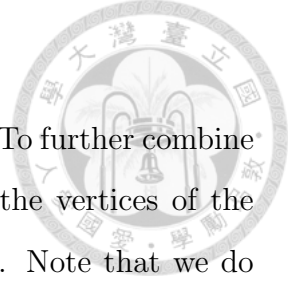


Figure 4.4: Illustration of spanning-graph construction for the two datasets. (a) The resultant graph after the vertices associated to the flow-based dataset have been traversed. (Assume that the vertices associated to the check-in-based dataset have not been traversed.) (b) The resultant spanning graph after all of the vertices have been traversed.

assume that vertices associated to the flow-based dataset will be traversed before those associated to the check-in-based dataset. Figure 4.4(a) shows the resultant graph after the vertices associated to the flow-based dataset have been traversed, and Figure 4.4(b) shows the resultant spanning graph after all of the vertices have been traversed.

Consider the process of connecting a vertex v for flow data to its nearest vertex for check-in data in each of the eight regions with respect to v . Interestingly, the process is somewhat similar to the case that one may walk to a nearby location (in any direction) after alighting from a taxi. From the perspective, we thought that using another method, *e.g.*, the k -nearest neighbors (k NN), for graph construction may work, but the resultant performance might not be better than that of using the spanning graphs (see Section 4.4.3.2 for the evaluation).



4.3.5 Weight Determination

Given a graph, we intend to assign a weight for each edge. To further combine the given two datasets, we plan to transmit information from the vertices of the flow-based dataset to the vertices of the check-in-based dataset. Note that we do not consider the opposite direction for transmission. The reason is that the check-in-based dataset can offer textual information, but our flow-based dataset cannot. Moreover, the information of event locations from the check-in-based dataset is typically more accurate than that from the flow-based dataset. By doing so, the exact event locations will be determined by the locations in the check-in-based dataset.

For weight determination, we consider (1) how the numerical values (*i.e.*, variability scores) of a vertex for flow data to be divided for its adjacent vertices, and (2) the percentage of the input from a vertex v for flow data, a vertex for check-in data that adjacent to v should receive. Accordingly, for each edge, we will assign two values, and then set the product of the two values as the edge weight. Formally, given a graph, let E be the edge set of the graph. Let V^F and V^C be the vertex set to the flow-based dataset of the graph and the vertex set to the check-in-based dataset of the graph, respectively. For each edge, the first value can be set as

$$w_{p,q}^F = \frac{\left(\frac{1}{g(p,q)}\right)^2}{\sum_{(p,r) \in E} \left(\frac{1}{g(p,r)}\right)^2},$$
$$\forall (p,q) \in E, p \in V^F, q \in V^C, \quad (4.5)$$

where $w_{p,q}^F$ is the first value for the edge between vertex p and vertex q , and $g(p,q)$ gives the geographical distance between the two locations that are associated to p and q . An edge will be assigned a large value if both (a) the degree of its vertex for flow data and (b) the geographical distance associated to the edge are small, compared to the other edges connected to the vertex for flow data. Figure 4.5 shows an example, where p is a vertex with a degree of 3 for flow data, and v_1 , v_2 , and v_3 are the vertex for check-in data that are adjacent to p . For each edge, the value

with an underline shows the first value for the edge. Similarly, the second value can

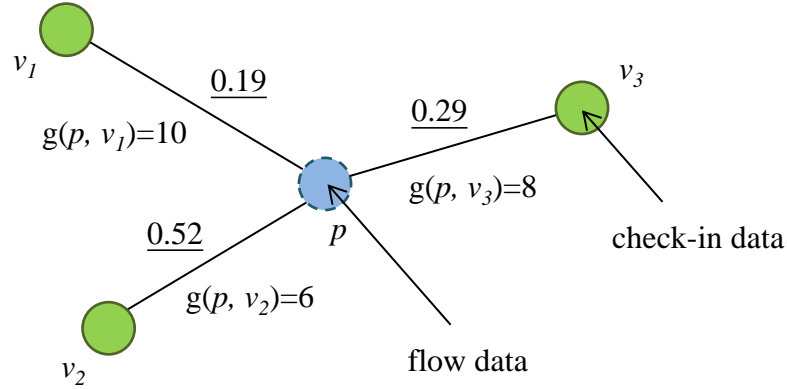


Figure 4.5: Illustration of assigning values for edges (partial values for the desired weights) that are adjacent to a vertex for flow data. Assume that p is a vertex for flow data, and v_1 , v_2 , and v_3 are the vertices for check-in data that are adjacent to p . If $g(\cdot)$ gives the geographical distance for two vertices, the assigned values will be the values with underlines.

be set as

$$w_{p,q}^C = \frac{\left(\frac{1}{g(p,q)}\right)^2}{\sum_{(r,q) \in E} \left(\frac{1}{g(r,q)}\right)^2},$$

$$\forall (p,q) \in E, p \in V^F, q \in V^C, \quad (4.6)$$

where $w_{p,q}^C$ is the second value for the edge between vertex p and vertex q , and $g(p,q)$ gives the geographical distance between the two locations that are associated to p and q . An edge will be assigned a large value if both (a) the degree of its vertex for check-in data and (b) the geographical distance associated to the edge are small, compared to the other edges connected to the vertex for check-in data.

Finally, the weight of the edge between p and q will be set to the product of $w_{p,q}^F$ and $w_{p,q}^C$.

4.3.6 Flow Propagation

So far, we have (1) the variability score of flow data for each location for each date, (2) the variability score for check-in data for each term of each location for

each date, and (3) a graph with weighted edges that connects the vertices associated to the locations of the flow-based dataset and the check-in-based dataset.

Now we can transmit information from the vertices for the flow-based dataset to the vertices for the check-in-based dataset along the weighted edges on the graph. For the check-in-based dataset, a score for each term of each location for each date can be determined by the following formula:

$$c_{q,j,k}^P = c_{q,j,k}^V \sum_{(p,q) \in E} (w_{p,q}^P \cdot f_{p,j}^V), \quad (4.7)$$

where $c_{q,j,k}^P$ is the resultant score for term t_k of location q for date j , $c_{q,j,k}^V$ is the variability score for check-in data for term t_k of location q for date j , $w_{p,q}^P$ is the weight of the edge that connects location p and location q , and $f_{p,j}^V$ is the variability score for the flow data of location p for date j . Finally, we can generate a ranked list of events based on the resultant scores. The higher the score is, the higher the rank is. Note that $c_{q,j,k}^V$ or the summation of the products may be negative. We will remove any event data from the ranked list if both of the values associated to the event data are negative. The reason is that this happens typically when more critical events happened before or after the current date. Although the case of two negative values could imply something happen, it also implies that it is challenging to describe the event based on the textual information from the current datasets.

For the ranked list of events, we will merge the results for the same date and the same location into the result that has a higher rank. As a result, we can use (1) a date, (2) a location (from the check-in-based dataset), and (3) one or more terms, to describe each event of the ranked list. Practically, we still can efficiently extract comments that are related to the terms from users' comments for the location for the date if necessary.

Table 4.3 gives an example of a ranked list of events before merging its items. Totally, five items (*i.e.*, rows) are listed. Comments of each item are associated to the terms of the item (comments are not shown here due to the space limit for the

table). Table 4.2 shows the resultant ranked list of events after merging the items in Table 4.3 that are with the same date and the same location.



Table 4.3: Illustration of an original ranked list of events, where “Cmnt” is an abbreviation of “Comment.” Items of the list will be merged together if they are for the same date and the same location.

Rank	Date	Location	Term	Cmnt
1	9/20	Moerenuma Park	fireworks	...
2	5/21	Sapporo Dome	baseball	...
3	9/20	Moerenuma Park	art	...
4	5/21	Sapporo Dome	match	...
5	5/21	Sapporo Dome	Nipponham	...

4.4 Experiments

To evaluate the proposed approach, we conducted experiments based on a taxi’s dataset, provided by National Institute of Informatics in Japan, from about 10,000 taxis and an Instagram’s dataset for the City of Sapporo from March to November in 2014 (as mentioned in Section 4.1).

The taxi’s dataset was generated based on the GPS data of the taxis. For the Instagram’s dataset, we used the Instagram’s open API [4] for data collection. To use the API, we considered two issues. First, each request has its upper limit to the number of returned photos, and thus the returned result might be incomplete if a large search region is used. Second, we are also limited to 5,000 requests per hour per *access_token*, which makes the use of small search regions for the whole collecting process become impractical. As a result, we followed the similar collection approach presented in [18], to collect as more photos as possible, considering the issues. More specifically, we started from a large search region (*i.e.*, a circle with 5km radius) and a time span (*i.e.*, 1,000 seconds). Whenever the number of returned photos is more

than the limit, we divided the search region into four sub-regions and then called the search API recursively.

Totally, the number of boarding and alighting locations is about 5,500 for the taxi’s dataset, and the number of check-in locations is about 6,300 for the Instagram’s dataset.

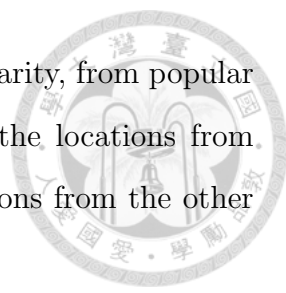
4.4.1 Experimental Setup

Overall, we will show (1) cross-dataset analysis and (2) comparatively studies, in Section 4.4.2 and Section 4.4.3, respectively. For cross-dataset analysis, we focused on data from May to July for detailed analysis. We studied the cross-dataset coverage for the taxi’s dataset and the Instagram’s dataset. For comparatively studies, we evaluated (a) the effect of using cross-domain social media (by comparing our approach with [50], *i.e.*, a state-of-the-art), (b) the effect of using spanning graphs for the connection of the two datasets, and (c) the effect of data normalization. We will show all the results, ranging from March to November.

Note that all performances of the comparatively studies were evaluated by $P@n$, which represents the precision (*i.e.*, correctness) of the top- n events in the ranking results. Because there is no ground truth in the media datasets, we pre-defined 212 events throughout the nine months manually, and then used these events as our ground truth. Most of these events are associated to sports, local festivals, concerts, and exhibitions. In particular, Sapporo has a soccer team and a baseball team. There are many sport events taking place every year. Moreover, Sapporo has many seasonal local festivals, *e.g.*, beer festival and snow festival. Therefore, more than 70% of the pre-defined events are either sports or local festivals.

4.4.2 Cross-Dataset Analysis

We see the correlation between the taxi’s data dataset and the Instagram’s dataset by coverage analysis. We consider the boarding and alighting locations for the taxi’s dataset and the check-in locations for the Instagram’s dataset. For each



dataset, we sorted the locations according to the degree of popularity, from popular locations to non-popular locations. We then see the ratio of the locations from one dataset that are close to (say, within 100 meters) the locations from the other dataset, considering the degree of popularity.

Figure 4.6 shows the ratios of the locations for the taxi's dataset that are close to the locations for the Instagram's dataset, where the numbers listed in the horizontal axis represent the percentages of the boarding and alighting locations for the taxi's dataset, based on the popularity. For example, 5% denotes the top-5% popular locations. The coverage ratio of the top-5% taxi's locations is about 94%, meaning that 94% of the taxi's locations are close to some Instagram's locations. Overall, the change of the coverage ratios implies that most of the popular taxi's

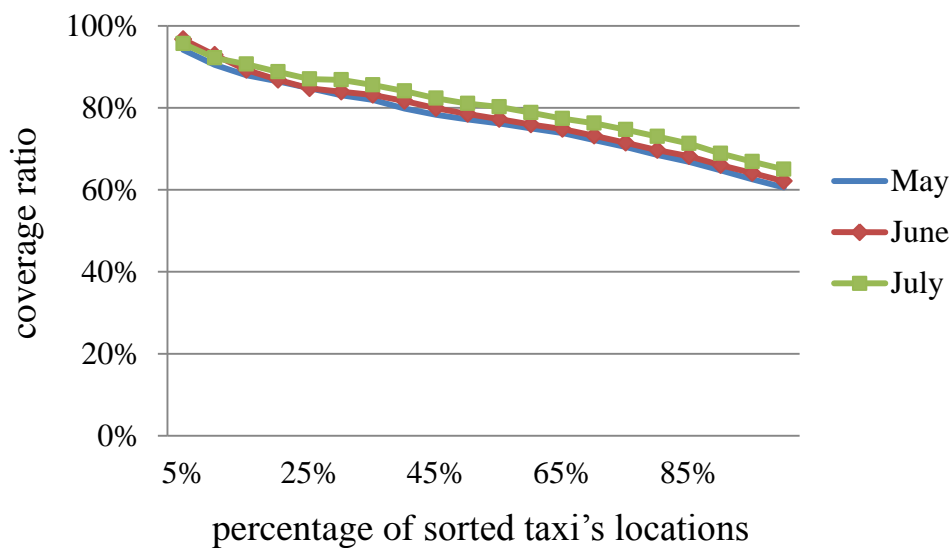
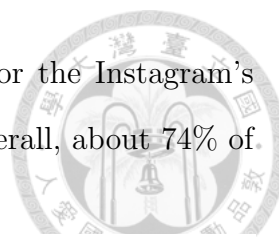


Figure 4.6: Change of coverage ratios from the Instagram's dataset along with changes of percentages of boarding and alighting locations for the taxi's dataset, based on the popularity. Most of the popular taxi's locations are close to some Instagram's locations, but the ratio decreases as more and more non-popular taxi's locations are involved.

locations are close to some Instagram's locations, but the ratio decreases as more and more non-popular taxi's locations are involved. That is, there might be no Instagram's locations that are close to the non-popular taxi's locations.



Similarly, Figure 4.7 shows the ratios of the locations for the Instagram’s dataset that are close to the locations for the taxi’s dataset. Overall, about 74% of

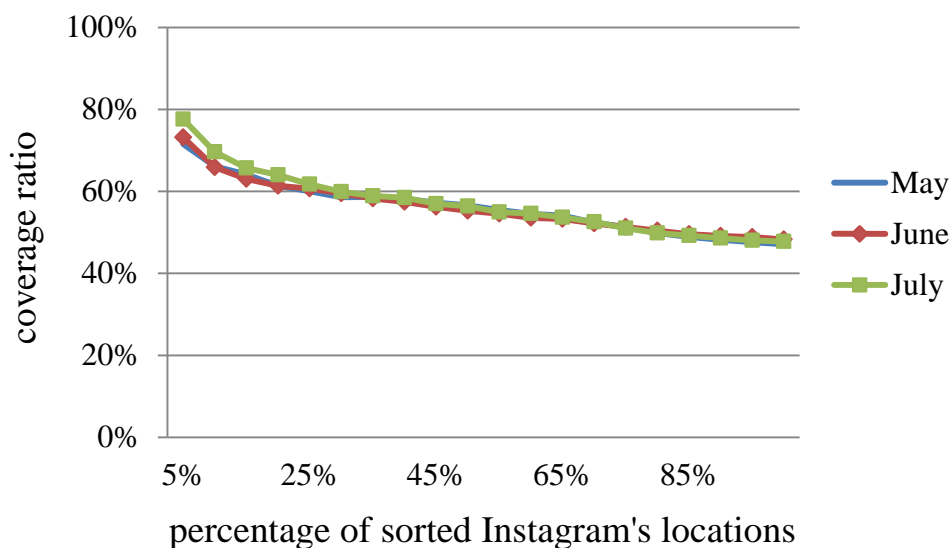


Figure 4.7: Change of coverage ratios from the taxi’s dataset along with changes of percentages of check-in locations for the Instagram’s dataset, based on the popularity. Most of the popular Instagram’s locations are close to some taxi’s locations, but the ratio decreases as more and more non-popular Instagram’s locations are involved.

the top-5% popular Instagram’s locations are close to some taxi’s locations, and the ratio decreases as more and more non-popular Instagram’s locations are involved. We noticed that the coverage ratio, 74%, is smaller than the coverage ratio, 94% (see Figure 4.6). This may be because the distribution of the popular Instagram’s locations is more scattered than that of the popular taxi’s locations. (Note that the coverage ratios shown in Table 4.1 are average values.)

4.4.3 Evaluation of the Proposed Approach

In the sequel, Section 4.4.3.1 shows the effect of cross-domain media mining, Section 4.4.3.2 shows the effect of spanning-graph construction, and finally, Section 4.4.3.3 shows the effect of data normalization.

4.4.3.1 Effect of Cross-Domain Media Mining

Figure 4.8 compares the results of using the Instagram’s dataset alone and the results of using both the taxi’s dataset and the Instagram’s dataset, where the approach of using the Instagram’s dataset alone was implemented based on that in [50]. As can be seen, combining the two datasets achieved better performance for

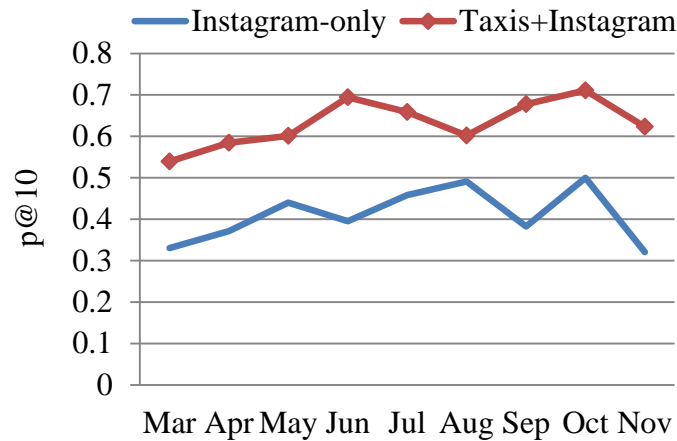


Figure 4.8: Comparison of the results of using the Instagram’s dataset alone and the results of using both the taxi’s dataset and the Instagram’s dataset.

finding events in the nine months. Based on the result, we found that some events can only be detected by using the Instagram’s dataset alone. Despite of that, the information of time for the events might not be accurate enough. The reason is that people might not upload their media data for an event immediately, but a few days after the event. Perhaps, some of them like to remove some photos, add tags for someone, or make some comments. In contrast, the information provided by the taxi’s dataset may be more real-time (as we summarized in Table 4.1). It is thus beneficial to add the taxi’s dataset for good performance.

4.4.3.2 Effect of Spanning-Graph Construction

This section considers the use of the k -nearest neighbors (k NN) for graph construction in our work, where the k NN is a classical algorithm that usually works well in practice. For graph construction, the k NN algorithm will connect each vertex to its nearest k vertices. In contrast, for spanning-graph construction, given

a vertex, the vertex connects only the nearest vertex in each of the eight regions with respect to itself (*c.f.*, Section 4.3.4). Figure 4.9 compares the results of using the k NN algorithm for graph construction and the results of using spanning graphs, where the k -value of the k NN algorithm was set to 8 because the spanning graphs used in our work considers only eight regions. (The construction of spanning graphs

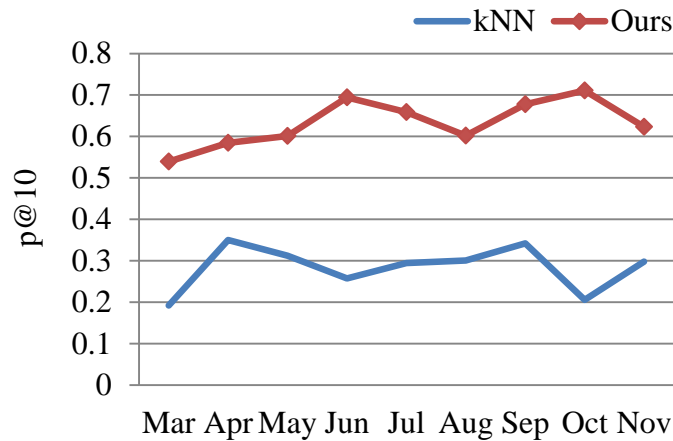


Figure 4.9: Compares the results of using the k -nearest neighbors (k NN) algorithm for graph construction and the results of using spanning graphs (*c.f.*, Section 4.3.4) for our work.

can be easily extended to consider more than eight regions.) Based on the results, we clearly see that the use of spanning graphs achieved better performance. We found that k NN may make a vertex connected to many vertices that are in similar directions to the vertex, and usually, this situation will mislead the information transmission along the edges.

4.4.3.3 Effect of Data Normalization

Finally, the section evaluates the effect of data normalization (*c.f.*, Sections 4.3.1 to 4.3.3). Specifically, we intend to evaluate the effect of the flow normalization and the variability-score calculation, except the buzz-score calculation. Buzz-score calculation is essential for us to extract critical textual information. We consider the correctness, *i.e.*, $P@n$, for the top-10 events, top-20 events, top-50 events, and all events (*i.e.*, 212 events), where the top-10 events refer to the 10

events that have the most data volumes among all the events, and so forth. Table 4.4 compares the results of no normalization (*i.e.*, no flow normalization and no variability-score calculation) and the results of using data normalization. As ex-

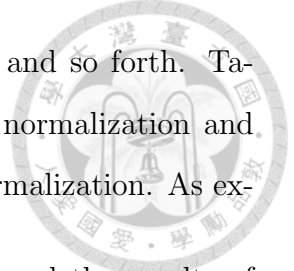


Table 4.4: Comparison of the results of no data normalization and the results of using data normalization. Note that “Ours w/o Normalization” did not use flow normalization and variability-score calculation, but buzz-score calculation.

	Ours w/o Normalization			
	P@1	P@2	P@5	P@10
Pre-defined 10	0.1814	0.1925	0.2144	0.2177
Pre-defined 20	0.1739	0.1911	0.2080	0.2280
Pre-defined 50	0.2004	0.2391	0.2529	0.2643
All pre-defined	0.2123	0.2385	0.2955	0.3149
	Ours			
	P@1	P@2	P@5	P@10
Pre-defined 10	0.4292	0.5452	0.5727	0.6047
Pre-defined 20	0.4385	0.5561	0.5733	0.6126
Pre-defined 50	0.5348	0.5728	0.6517	0.6166
All pre-defined	0.5054	0.5682	0.6419	0.6323

pected, normalization does play an important role to combine datasets with different *units*, *e.g.* the “times” of boarding and alighting from taxis, for high performance. We found that data from the taxi’s dataset dominated the ranking results if no normalization was involved. This is because the data volume of the taxi’s dataset is much more than that of the textual data from the Instagram’s dataset in our work.

Chapter 5

Conclusion and Future Work



In recent years, social media have changed the world and our lives. More and more people like to share their daily life by social media. This dissertation has presented three techniques for an effective social-media mining system, aiming to make our lives better. Given a photo, image location identification provides us geographical information, image annotation provides us people's description or comments, and event discovery provides us events of interest to the nearby places associated to the photo. This chapter concludes our research results and lists some directions for future research.

5.1 Conclusion

For image location identification, we have presented an approach that unifies visual features, geo-tags, and check-in data of images, for the addressed problem. Moreover, we have introduced a location-aware graph-based regrouping approach on clusters of images, where this approach might benefit existing clustering-based techniques. Furthermore, we have integrated sparse coding in our system and developed a graph-based dictionary selection approach for sparse coding. Finally, experimental results have shown that our technique can be applied on daily-life large-scale image datasets to retrieve image locations in a reasonable quality.

For image annotation, we have presented a graph-based multi-layer multi-label SSL method which can effectively unify the visual and textual information for multi-label learning. Our framework does not require to pre-processing the given large-scale dataset but is capable of performing large-scale multi-label propagation. On the other hand, we have also presented a tag refinement technique which can

simultaneously suppress noisy tags and emphasize the other tags. Experimental results have shown that our algorithm can operate on a large-scale image dataset while effectively infer the image labels.

For event discovery, we have presented a two-stage framework, including data normalization and graph-based data fusion, that unifies a flow-based dataset and a check-in-based dataset for ranked events. Data normalization enables the fusion of two datasets, and data fusion combines data from two datasets with graph approaches. Based on a taxi's dataset and an Instagram's dataset, the experiments have shown the effectiveness of the proposed approach. Further, the framework is capable of combing other datasets for high performance.

5.2 Future Work

Two directions for future research are listed as follows.

1. Combing Visual and Textual Information for Event Discovery:

Most approaches for event discovery were based on textual information. Recently, many people like to share photos than textual data (*e.g.*, comments) on social media websites. Generally, photos can provide people a different kind of information, in contrast to textual data. It is desirable to combing visual and textual information for event discovery. Fortunately, there have been lots of studies on image content analysis and/or its application. An idea is to transform information of photos into textual data, and then add the data to existing textual data. By doing so, we can apply conventional approaches for event discovery. However, it is challenging to extract textual data from photos with high performance. Moreover, it could be difficult to find proper weights for different kinds and amounts of information for data fusion.

2. Combing Check-in and Flow Information for Traffic Route Recommendation:

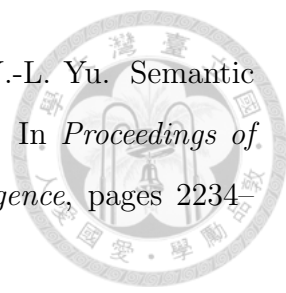
Traffic congestion is an important problem. Traffic congestion not only wastes time and energy resources, but may also endanger our lives. For example, ambulances or fire engines may get stuck in traffic jams. Discovering relation between events and traffic flows is capable of solving the problem. Specifically, check-in data can be used to find events, and flow data can be used to find regions of traffic congestion. Given an event, it is expected to predict the regions of traffic congestion based on the historical data, and if possible, recommend people proper routes to avoid the generation of traffic congestion. Further, it is desirable to perform efficient information update based users' feedback.

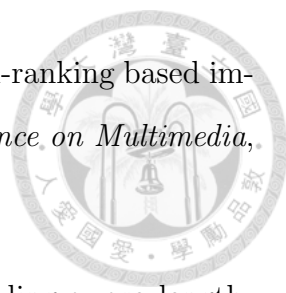


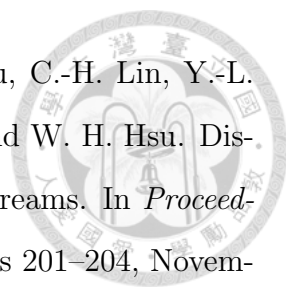
Bibliography

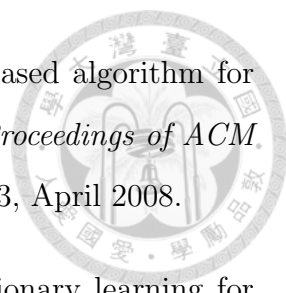



- [1] *Flickr*.
<http://www.flickr.com/>.
- [2] *Foursquare*.
<https://foursquare.com/>.
- [3] *Google maps*.
<https://maps.google.com/>.
- [4] *Instagram API endpoints*.
<http://instagram.com/developer/endpoints/media/>.
- [5] *Yahoo! travel*.
<http://travel.yahoo.com/>.
- [6] Z. Al Bawab, G. H. Mills, and J.-F. Crespo. Finding trending local topics in search queries for personalization of a recommendation system. In *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining*, pages 397–405, August 2012.
- [7] Y. Avrithis, Y. Kalantidis, G. Toliás, and E. Spyrou. Retrieving landmark and non-landmark images from community photo collection. In *Proceedings of ACM International Conference on Multimedia*, pages 153–162, October 2010.
- [8] H. Becker, D. Iter, M. Naaman, and L. Gravano. Identifying content for planned events across social media sites. In *Proceedings of ACM International Conference on Web Search and Data Mining*, pages 533–542, February 2012.
- [9] A. Z. Broder. On the resemblance and containment of documents. In *Proceedings of IEEE Compression and Complexity of Sequences*, pages 21–29, June 1997.

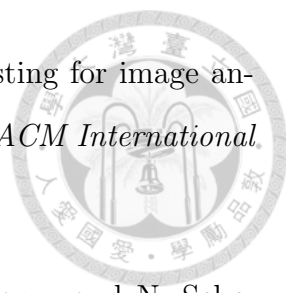
- 
- [10] X. Chang, Y. Yang, A. G. Hauptmann, E. P. Xing, and Y.-L. Yu. Semantic concept discovery for large-scale zero-shot event detection. In *Proceedings of IJCAI International Joint Conference on Artificial Intelligence*, pages 2234–2240, July 2015.
- [11] D. M. Chen, G. Baatz, K. Köser, S. S. Tsai, R. Vedantham, T. Pylvänäinen, K. Roimela, X. Chen, J. Bach, M. Pollefeys, B. Girod, and R. Grzeszczuk. City-scale landmark identification on mobile devices. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 737–744, June 2011.
- [12] A. Coates and A. Y. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *Proceedings of IMLS International Conference on Machine Learning*, pages 921–928, June–July 2011.
- [13] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2009.
- [14] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. *ACM Communications of the ACM*, 51(1):107–113, January 2008.
- [15] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *IMS The Annals of Statistics*, 32(2):407–499, April 2004.
- [16] Facebook. *Form 10-K (annual report)–filed 02/01/13 for the period ending 12/31/12*, 2013.
- [17] L. Feng, J. Wu, S. Liu, and H. Zhang. Global correlation descriptor: a novel image representation for image retrieval. *Elsevier Journal of Visual Communication and Image Representation*, 33(1):104–114, November 2015.
- [18] T. Fujisaka, R. Lee, and K. Sumiya. Discovery of user behavior patterns from geo-tagged micro-blogs. In *Proceedings of ACM International Conference on Ubiquitous Information Management and Communication*, pages 246–255, January 2010.


- 
- [19] J. He, M. Li, H.-J. Zhang, H. Tong, and C. Zhang. Manifold-ranking based image retrieval. In *Proceedings of ACM International Conference on Multimedia*, pages 9–16, October 2004.
- [20] K.-H. Ho, H.-C. Ou, Y.-W. Chang, and H.-F. Tsao. Coupling-aware length-ratio-matching routing for capacitor arrays in analog integrated circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(2):161–172, February 2015.
- [21] S.-L. Huang, C.-A. Wu, K.-F. Tang, C.-H. Hsu, and C.-Y. R. Huang. A robust ECO engine by resource-constraint-aware technology mapping and incremental routing optimization. In *Proceedings of IEEE/ACM Asia and South Pacific Design Automation Conference*, pages 382–387, January 2011.
- [22] Y.-G. Jiang, J. Wang, and S.-F. Chang. Lost in binarization: query-adaptive ranking for similar image search with compact codes. In *Proceedings of ACM International Conference on Multimedia Retrieval*, pages 1–8, April 2011.
- [23] U. Kang, C. E. Tsourakakis, and C. Faloutsos. PEGASUS: mining peta-scale graphs. *Springer Knowledge and Information Systems*, 27(2):303–325, May 2011.
- [24] G. Karakostas. A better approximation ratio for the vertex cover problem. *ACM Transactions on Algorithms*, 5(4):41:1–41:8, October 2009.
- [25] L. Kennedy and M. Naaman. Generating diverse and representative image search results for landmarks. In *Proceedings of ACM International Conference on World Wide Web*, pages 297–306, April 2008.
- [26] Y.-H. Kuo, K.-T. Chen, C.-H. Chiang, and W. H. Hsu. Query expansion for hash-based image object retrieval. In *Proceedings of ACM International Conference on Multimedia*, pages 65–74, October 2009.

- 
- [27] Y.-H. Kuo, Y.-Y. Chen, B.-C. Chen, W.-Y. Lee, C.-C. Wu, C.-H. Lin, Y.-L. Hou, W.-F. Cheng, Y.-C. Tsai, C.-Y. Hung, L.-C. Hsieh, and W. H. Hsu. Discovering the city by mining diverse and multimodal data streams. In *Proceedings of ACM International Conference on Multimedia*, pages 201–204, November 2014.
- [28] Y.-H. Kuo, W.-Y. Lee, W. H. Hsu, and W.-H. Cheng. Augmenting mobile city-view image retrieval with context-rich user-contributed photos. In *Proceedings of ACM International Conference on Multimedia*, pages 687–690, November–December 2011.
- [29] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *Proceedings of NIPS Foundation Advances in Neural Information Processing Systems*, pages 801–808, December 2006.
- [30] H. Liu, T. Mei, J. Luo, H. Li, and S. Li. Finding perfect rendezvous on the go: accurate mobile visual localization and its applications to routing. In *Proceedings of ACM International Conference on Multimedia*, pages 9–18, October–November 2012.
- [31] J. Liu, Z. Huang, L. Chen, H. T. Shen, and Z. Yan. Discovering areas of interest with geo-tagged images and check-ins. In *Proceedings of ACM International Conference on Multimedia*, pages 589–598, October–November 2012.
- [32] W. Liu, J. He, and S.-F. Chang. Large graph construction for scalable semi-supervised learning. In *Proceedings of IMLS International Conference on Machine Learning*, pages 679–686, June 2010.
- [33] Y. Liu, T. Mei, X. Wu, and X.-S. Hua. Multigraph-based query-independent learning for video search. *IEEE Transactions on Circuits and Systems for Video Technology*, 19(12):1841–1850, December 2009.

- 
- [34] J. Long, H. Zhou, and S. O. Memik. An $O(n \log n)$ edge-based algorithm for obstacle-avoiding rectilinear Steiner tree construction. In *Proceedings of ACM International Symposium on Physical Design*, pages 126–133, April 2008.
- [35] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proceedings of IMLS International Conference on Machine Learning*, pages 689–696, June 2009.
- [36] P. Meladianos, G. Nikolentzos, F. Rousseau, Y. Stavrakas, and M. Vazirgianis. Degeneracy-based real-time sub-event detection in Twitter stream. In *Proceedings of AAAI International Conference on Web and Social Media*, pages 248–257, May 2015.
- [37] J.-Y. Pan, H.-J. Yang, C. Faloutsos, and P. Duygulu. Automatic multimedia cross-modal correlation discovery. In *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining*, pages 653–658, August 2004.
- [38] B. Quanz, J. Huan, and M. Mishra. Knowledge transfer with low-quality data: a feature extraction issue. *IEEE Transactions on Knowledge and Data Engineering*, 24(10):1789–1802, October 2012.
- [39] D. Rao and D. Yarowsky. Ranking and semi-supervised classification on large scale graphs using Map-Reduce. In *Proceedings of ACM Workshop on Graph-Based Methods for Natural Language Processing*, pages 58–65, August 2009.
- [40] M. Rischka and S. Conrad. Image landmark recognition with hierarchical k -means tree. In *Proceedings of GI Database Systems for Business, Technology, and Web*, pages 455–464, March 2015.
- [41] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes Twitter users: real-time event detection by social sensors. In *Proceedings of ACM International Conference on World Wide Web*, pages 851–860, April 2010.

- 
- [42] J. Sankaranarayanan, H. Samet, B. E. Teitler, M. D. Lieberman, and J. Spertling. TwitterStand: news in tweets. In *Proceedings of ACM International Conference on Advances in Geographic Information Systems*, pages 42–51, November 2009.
- [43] B. Shaw, J. Shea, S. Sinha, and A. Hogue. Learning to rank for spatiotemporal search. In *Proceedings of ACM International Conference on Web Search and Data Mining*, pages 717–726, February 2013.
- [44] S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. Academic Press, 2008.
- [45] H. Tong, J. He, M. Li, W.-Y. Ma, H.-J. Zhang, and C. Zhang. Manifold-ranking-based keyword propagation for image retrieval. *EURASIP Journal on Advances in Signal Processing*, 2006(1):1–10, January 2006.
- [46] H. Tong, J. He, M. Li, C. Zhang, and W.-Y. Ma. Graph based multi-modality learning. In *Proceedings of ACM International Conference on Multimedia*, pages 862–871, November 2005.
- [47] V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2001.
- [48] M. Wang, X.-S. Hua, X. Yuan, Y. Song, and L.-R. Dai. Optimizing multi-graph learning: towards a unified video annotation scheme. In *Proceedings of ACM International Conference on Multimedia*, pages 862–871, September 2007.
- [49] J. Weng, Y. Yao, E. Leonardi, and B.-S. Lee. Event detection in Twitter. Technical report, HP Laboratories, 2011.
- [50] C.-C. Wu, T. Mei, W. H. Hsu, and Y. Rui. Learning to personalize trending image search suggestion. In *Proceedings of ACM International Conference on Research and Development in Information Retrieval*, pages 727–736, July 2014.

- 
- [51] F. Wu, Y. Han, Q. Tian, and Y. Zhuang. Multi-label boosting for image annotation by structural grouping sparsity. In *Proceedings of ACM International Conference on Multimedia*, pages 15–24, October 2010.
- [52] Y. Yan, Y. Yang, H. Shen, D. Meng, G. Liu, A. Hauptmann, and N. Sebe. Complex event detection via event oriented dictionary learning. In *Proceedings of AAAI Conference on Artificial Intelligence*, pages 3841–3847, January 2015.
- [53] Y.-H. Yang, P.-T. Wu, C.-W. Lee, K.-H. Lin, W. H. Hsu, and H. Chen. ContextSeer: context search and recommendation at query time for shared consumer photos. In *Proceedings of ACM International Conference on Multimedia*, pages 199–208, October 2008.
- [54] A. C.-C. Yao. On constructing minimum spanning trees in k -dimensional spaces and related problems. *SIAM Journal on Computing*, 11(4):721–736, November 1982.
- [55] P. A. Zandbergen and S. J. Barbeau. Positional accuracy of assisted GPS data from high-sensitivity GPS-enabled mobile phones. *RIN Journal of Navigation*, 64(3):381–399, July 2011.
- [56] W. Zhang, G. Qi, G. Pan, H. Lu, S. Li, and Z. Wu. City-scale social event detection and evaluation with taxi traces. *ACM Transactions on Intelligent Systems and Technology*, 6(3):40:1–40:20, May 2015.
- [57] D. Zhou, Q. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Proceedings of NIPS Foundation Advances in Neural Information Processing Systems*, pages 321–328, December 2003.
- [58] H. Zhou, N. Shenoy, and W. Nicholls. Efficient minimum spanning tree construction without Delaunay triangulation. *Elsevier Information Processing Letters*, 81(5):271–276, February 2002.

- 
- [59] G. Zhu, S. Yan, and Y. Ma. Image tag refinement towards low-rank, content-tag prior and error sparsity. In *Proceedings of ACM International Conference on Multimedia*, pages 461–470, October 2010.
- [60] X. Zhu. *Semi-supervised learning literature survey*. Doctoral Dissertation, Carnegie Mellon University, 2006.
- [61] X. Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin Madison, 2008.
- [62] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of IMLS International Conference on Machine Learning*, pages 912–919, August 2003.