10-4-2017

Data Mining Techniques to Understand Textual Data

Wubai Zhou
Florida International University, wzhou005@fiu.edu

DOI: 10.25148/etd.FIDC003998

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

DATA MINING TECHNIQUES TO UNDERSTAND TEXTUAL DATA

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Wubai Zhou

2017

To: John L. Volakis
College of Engineering and Computing

This dissertation, written by Wubai Zhou, and entitled Data Mining Techniques to Understand Textual Data, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Sundaraja Sitharama Iyengar

Ning Xie

Debra VanderMeer

Shu-Ching Chen, Co-Major Professor

Tao Li, Co-Major Professor

Date of Defense: October 04, 2017

The dissertation of Wubai Zhou is approved.

John L. Volakis
College of Engineering and Computing

Andrés G. Gil
Vice President for Research and Economic Development
and Dean of University Graduate School

Florida International University, 2017

DEDICATION

I dedicate this dissertation work to my beloved family, especially my parents. Without their patience, understanding, support, or love, the completion of this work would not have been possible.

ACKNOWLEDGMENTS

It is the support from many people that brings me with the possibility to complete the dissertation for concluding my Ph.D. study.

First and foremost, I would like to express my sincerest thanks and appreciation to my advisor Dr. Tao Li, for his inimitable support, meticulous guidance, insightful advice, and getting me immersed with an excellent research atmosphere. Professor Tao Li, an eminent researcher in the community of data mining, is recognized as a knowledgeable and dedicated mentor. It is his great passion and endless patience that help me to locate my research interest and develop the related research skills in data mining area.

Second, I want to extend my gratitude to my co-advisor, Dr. Shu-Ching Chen, who has not only provided valuable guidance for me doing BCIN research project, but also given me constructive suggestion in developing my Ph.D. career.

Third, my thanks goes to all my dissertation committee members: Dr. S.S. Iyengar, Dr. Ning Xie, and Dr. Debra VanderMeer, for their helpful advices, insightful comments on my dissertation research and future research career plans.

Fourth, I'd like to thank all my mentors including Dr. Larisa Shwartz, Dr. Genady Ya. Grabarnik, Mr. Jinsong Tan for my three internships: twice at IBM and once at Uber. The patient guidance and help from them let me accumulate valuable experience and benefit me a lot in my Ph.D. study.

Moreover, I would like extend my thanks to our department staffs for assisting me with the administrative tasks necessary during my doctoral study: Olga Carbonell, Carlos Cabrera, Steven Luis, Luis Rivera, etc.

Additionally, it's extremely fortunate for me to join Knowledge Discovery and Research Group (KDRG) where I have built up my research experience and enhanced my knowledge. I am very grateful to all my colleagues of KDRG including Dr. Liang Tang, Dr. Yexi Jiang, Dr. Li Zheng, Dr. Lei Li, Dr. Jingxuan Li, Dr. Chao Shen,

Dr. Longhui Zhang, Dr. Chunqiu Zeng, Hongtai Li, Wei Xue, Qing Wang, Shekoofeh Mokhtari, Wentao Wang, Ramesh Baral, Xiaolong Zhu and Boyuan Guan. Both valuable discussions and helpful suggestions from them continuously enlighten me with new insights into my research problems.

Finally, I would like to express my utmost gratitude to my parents and family, whose endless love and understanding are with me in whatever I pursue. Without the unlimited support from them, I would never go through any tough times in my life.

ABSTRACT OF THE DISSERTATION

DATA MINING TECHNIQUES TO UNDERSTAND TEXTUAL DATA

by

Wubai Zhou

Florida International University, 2017

Miami, Florida

Professor Tao Li, Co-Major Professor

Professor Shu-Ching Chen, Co-Major Professor

More than ever, information delivery online and storage heavily rely on text. Billions of texts are produced every day in the form of documents, news, logs, search queries, ad keywords, tags, tweets, messenger conversations, social network posts, etc. Text understanding is a fundamental and essential task involving broad research topics, and contributes to many applications in the areas text summarization, search engine, recommendation systems, online advertising, conversational bot and so on. However, understanding text for computers is never a trivial task, especially for noisy and ambiguous text such as logs, search queries. This dissertation mainly focuses on textual understanding tasks derived from the two domains, i.e., disaster management and IT service management that mainly utilizing textual data as an information carrier.

Improving situation awareness in disaster management and alleviating human efforts involved in IT service management dictates more intelligent and efficient solutions to understand the textual data acting as the main information carrier in the two domains. From the perspective of data mining, four directions are identified: (1) Intelligently generate a storyline summarizing the evolution of a hurricane from relevant online corpus; (2) Automatically recommending resolutions according to the textual symptom description in a ticket; (3) Gradually adapting the resolution rec-

ommendation system for time correlated features derived from text; (4) Efficiently learning distributed representation for short and lousy ticket symptom descriptions and resolutions. Provided with different types of textual data, data mining techniques proposed in those four research directions successfully address our tasks to understand and extract valuable knowledge from those textual data.

My dissertation will address the research topics outlined above. Concretely, I will focus on designing and developing data mining methodologies to better understand textual information, including (1) a storyline generation method for efficient summarization of natural hurricanes based on crawled online corpus; (2) a recommendation framework for automated ticket resolution in IT service management; (3) an adaptive recommendation system on time-varying temporal correlated features derived from text; (4) a deep neural ranking model not only successfully recommending resolutions but also efficiently outputting distributed representation for ticket descriptions and resolutions.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

More than ever, information delivery online and storage heavily rely on text. Billions of texts are produced every day in the form of documents, news, logs, search queries, ad keywords, tags, tweets, messenger conversations, social network posts, etc. Text understanding is a fundamental and essential task involving broad research topics, and contributes to many applications in the areas text summarization, search engine, recommendation systems, online advertising, conversational bot and so on as shown in Figure 1.1. However understanding text for computers is never a trivial task, especially for noisy and ambiguous text such as logs, search queries.

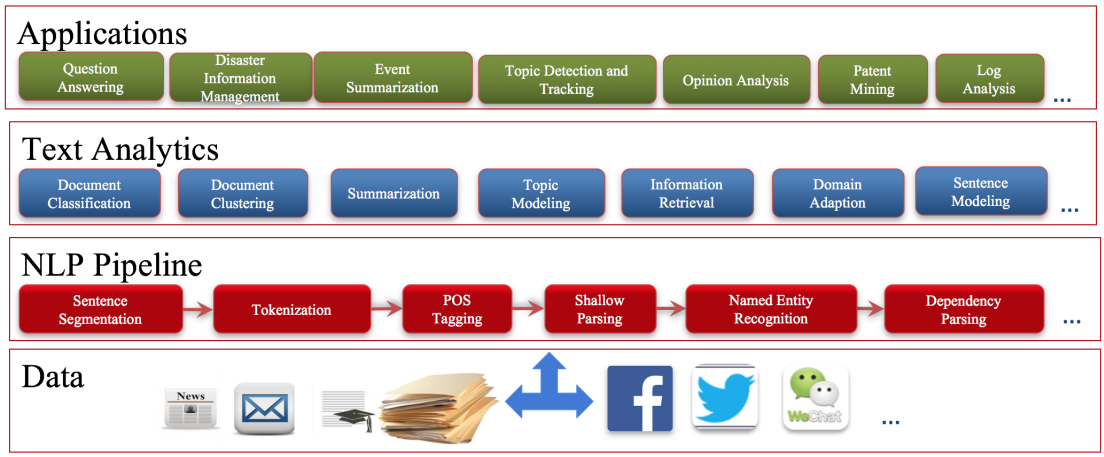

Figure 1.1: General architectures of text analysis.

Figure 1.1 shows the general architecture of text analysis that consists of four layers. 1) The data layer provides the potential sources for textual data collection; 2) the NLP pipeline layer presents a classical, but not limited to, pre-processing pipeline for text analytics; 3) the text analytics layer is the core part of text analysis listing the existing categories of data mining techniques to text analysis, such as summarization,

sentence modeling and so on; 4) the application layer finally specifies the real-world applications benefiting from text analytics. The breadth and depth of text analytics indicates the impossibility for me to coverage all related topics, and this thesis mainly focus on textual analytical tasks derived from the two domains I work on during my Ph.D. program, i.e., disaster management and IT service management that mainly utilizing textual data as an information carrier.

Disaster management aims to prevent natural disaster from occurring and failing, and should develop a good action plan to mitigate the results and effects of any natural disasters. Natural disasters such as hurricanes, earthquakes and tsunamis cause inestimable physical destruction, loss of life and property around the world every year. For example, Hurricane Sandy affected the east coast of U.S. in 2012 and posed immense threats to businesses, human lives, and properties. In order to minimize the consequent loss of the disasters, a essential task in disaster management is to efficiently analyze and understand the disaster-related situation updates which usually can be gathered and extracted from a myriad of web documents, e.g., news and reports that are related to the disasters.

Todays competitive business climate, and the complexity of service environments, dictate efficient and cost-effective service delivery and support. This is largely achieved through service-providing facilities to collaborate with system management tools, combined with automation of routine maintenance procedures that includes problem detection, determination, and resolution for IT service providers, and is prescribed by the Information Technology Infrastructure Library (ITIL) specification [urld]. A typical workflow of the IT routine maintenance is illustrated Figure 1.2, where five major stages are defined.

At the first stage, problem detection in IT environment is realized by system monitoring. Some popular system monitoring softwares encompass IBM Tivoli Mon-

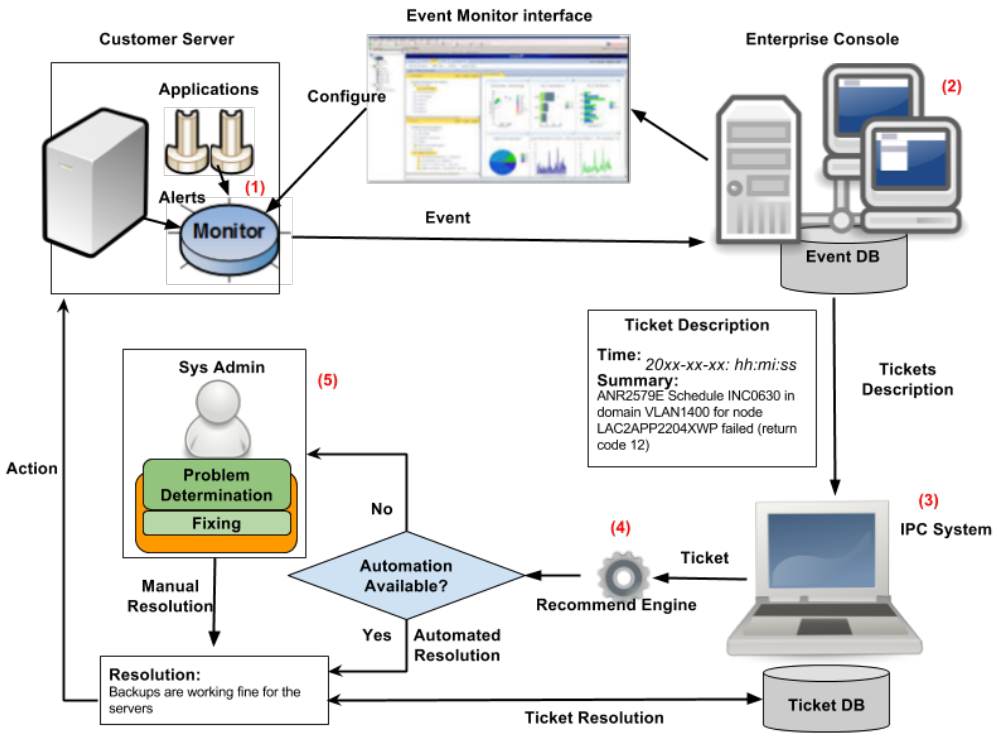

Figure 1.2: Service management system.

itoring [IBM16], HP Open View [urla], LogicMonitor [log16], Zenoss [zen16], ManageEngine [Man16], etc. System monitoring, one important component in service management, is capable of tracking the states of a system by collecting system statistics information such as the CPU utilization, the memory usage, the number of data bytes written and read on the disk, the sequence of request and responses processed on an application server, etc. The system monitoring computes metrics based on the regularly collected system performance data and compares those metrics with some predefined acceptable thresholds, referred to as monitoring situations as well. Any violation after comparison raises an alert. If the alert persists beyond a certain period specified in the situation, the monitor emits an event.

At the second stage, the generated events from the entire IT environment are consolidated in an enterprise console and stored in an event database. The console

also employs rule, case or knowledge based engine to analyze the events, and the results are reported to system administrators by the event business intelligence system through which system administrators is able to adjust or add new configuration to the system monitoring. The console analyzes the events and decides whether to report problems with a service ticket in the Incident, Problem, Change (IPC) system.

At the third stage, the reported tickets are stored in the ticket database of IPC system. The information accumulated in the newly created ticket, which only describes the symptoms of the corresponding problem, is used for problem diagnosis, determination and resolution by system administrators. A newly created ticket with only ticket description is illustrated in Figure 1.2.

At the forth stage, the resolution recommendation engine proposes several resolutions from historical tickets to system administrators. The system administrators apply those resolutions to the problematic servers. If problems still persist, the ticket then is assigned and passed to system administrators for manual diagnosis and resolution.

SEVERITY	**FIRST-OCCURRENCE**	**LAST-OCCURRENCE**
0	2014-03-29 05:50:39	2014-03-31 05:36:01
SUMMARY	ANR2579E Schedule INC0630 in domain VLAN1400 for node LAC2APP2204XWP failed (return code 12)	
RESOLUTION	Backups are working fine for the server.	
CAUSE	**ACTIONABLE**	**LAST-UPDATE**
Maintenance	Actionable	2014-04-29 23:19:25

Table 1.1: A sample ticket.

At the fifth stage, as a new ticket arrives, the system administrators inspect the ticket description, and infer the possible categories of the underlying IT problem based on their domain knowledge. The problem category inference further directs the ticket being assigned to proper processing teams for problem resolution, where different processing teams typically specialize in diverse IT problem categories. In general, the

system administrators' role is limited to help triage tickets to the processing teams for problem resolving, while the processing teams are responsible to perform complex root cause analysis with respect to the related system performance statistics, event and ticket data. Finally, the service returns to be normal after problem resolving. The intensive human labor involving in this stage dictates the efficiency of the forth stage, which requires intelligent understanding on ticket data whose most important and informative attributes are in textual form, i.e., ticket summary and resolution. A complete sample ticket is illustrated in Table 1.1.

In summary, intelligent understanding of these text resources is critical for a high quality service delivery. In disaster management, a domain expert may be interested in how a hurricane evolves over different geo-spatial regions and how the topic changes over time. How can the computer understand those text and extract the gist from the large amount of corpus? In IT service management, once a large amount of historical resolving tickets are accumulated and collected in which potential relevant resolutions might exist for an incoming unresolved ticket, many IT service providers rely on automatically retrieving relevant resolutions from historical tickets to alleviate human efforts involved in IT service management. However, how can the computer understand those text and retrieve the relevant resolutions might be a challenging task.

1.2 Motivation and Problem Statement

Many data mining techniques can be utilized to achieve the goal of various intelligent textual understanding tasks. However, it is not a trivial task for computers, sometimes not even for humans to understand them. Some textual understanding challenges in the disaster management and the IT service management domains are identified and listed as below.

- Natural disasters such as hurricanes, earthquakes and tsunamis cause inestimable physical destruction, loss of life and property around the world every year. For example, Hurricane Sandy affected the east coast of U.S. in 2012 and posed immense threats to businesses, human lives, and properties. In order to minimize the consequent loss of a catastrophe like hurricanes, it is critical to instantly realize situation updates on the disaster from a large number of easily accessible disaster-related documents. The domain experts expect to obtain condensed information about the detailed disaster event description, e.g., the evolutionary tendency of the disaster with respect to different locations. However, it is often a non-trivial task to generate a big picture of the disaster events due to the flood of web documents.

- Todays competitive business climate, and the complexity of service environments, dictate efficient and cost-effective service delivery and support. This is largely achieved through service-providing facilities to collaborate with system management tools, combined with automation of routine maintenance procedures including problem detection, determination and resolution for the service infrastructure [MSGL09, TLP+12, ABD+07, WE11, YPZ10]. Most service providers keep track of a large amount of historical tickets with resolutions. The resolution is usually stored as a plain text which describes how this ticketed incident has been resolved. We analyzed historical tickets collected from several accounts (an aggregate of services using a common infrastructure) managed by IBM Global Services. One observation is that many tickets share the same resolutions. We noticed that there are many repeating resolutions for tickets within an account. It is natural to expect that if ticket summary are similar, then their respective tickets probably have the same resolution. Therefore, we can recommend a resolution for an incoming ticket based on the ticket

summary information and historical tickets. However, it is a challenging task to match semantically similar ticket with short and noisy textual information automatically generated from IT servers.

- Massive heterogeneous applications as well as various monitoring software are running on clients' servers to accomplish required tasks and to monitor system health via different metrics. It leads to dynamic change in tickets' representation that they have different symptom descriptions but similar resolutions. The time-varying representation cause performance degradation in ticket resolution recommendation. Therefore, mining those temporal correlated features is critical for adapting the resolution recommendation system in the complex IT environment and thus improving the service quality.
- The scale and complexity of these system probably causes a large number of unexpected behaviors during failures, system perturbations and even normal operations which leads huge amounts of tickets. However, previous approaches for ticket resolution recommendation are hard to scale and limited to learn semantical meanings from historical tickets data, such as the distributed representation for tickets. A sample of real-world tickets (see Table 1.1 for the contents of tickets that are not easily interpretable) illustrate the unique ticket characteristics that are less intuitive and lead to challenges in many basic ticket analysis tasks. Therefore, it is an essential task in IT service management to accurately represent the ticket summary and resolution. The classical techniques such as the n-gram, TF-IDF, and LDA are not effective in representing tickets as the ticket summary and resolution are generally not well formatted.

Driven by the challenges above, efficient data mining techniques for intelligent understanding on textual data are pressingly demanded. In recent years, data mining techniques have acquired great success to address issues in textual understand-

ing tasks, such as text summarization [Man01, RHM02, ER04, SGH12], distributed representation learning for text [DDF+90, BTWDR13], question answering [CKC05, Sas05, RCD11] and so on. These techniques are employed for efficiently extracting gist from a myriad of documents and building semantical relationship among words, phrases, sentences and even documents. Our work focuses on designing and implementing solutions based on textual data to alleviate human efforts and facilitate problem determination, diagnosis and resolution in disaster management and IT service management.

From the perspective of data mining, four research directions are identified and considered to better understand text and thus benefit solving our real world problems.

1. ***Intelligently generate a storyline summarizing the evolution of a hurricane from relevant online corpus.*** In order to minimize the consequent loss from a catastrophe like hurricanes, it is critical to instantly realize situation updates on the disaster from a large number of easily accessible disaster-related documents. The domain experts expect to obtain condensed information about the detailed disaster event description. This task is often recognized as a text summarization problem. To tackle this problem, various type of document understanding system have been developed over the last decade.These systems include (1) summarization-based systems [LLL12a, RJST04, SBC03, SL10, WLLH08] that choose from multiple documents a subset of sentences conveying the principle idea; (2) topic detection and tracking systems [All12] aiming to group documents into different clusters as events and monitor future events related to the corresponding topic; and (3) timeline generation systems [SGH12, WLO12] that create summaries to present the evolution of an event by leveraging temporal information attached to or extracted from the documents. These systems are able to alleviate the so-called problem to

some extent; however, they suffer from several limitations that may affect the quality of the summarized results. First, most of them focus on summarizing an event via topic evolution over the time, but ignore the spatial information which is important especially for large-scale disaster events. Second, these systems usually generate a single layer summarization or storyline to reflect topic changes over the entire event. However, due to the spatial factor, the information evolution over a disaster event is intrinsically hierarchical. In most cases, domain experts are often interested in not only the general picture of a disaster, but also how it affects a particular region. Our work fits into the third category, i.e., timeline generation systems. Specifically, in our framework, a disaster event is initially summarized from a large set of documents (e.g., news and reports) with a big picture showing how the disaster affects different regions. It can then be zoomed into a specific location for more detailed location-specific event summarization.

2. ***Automatically recommending resolutions according to the symptom description in a ticket.*** The symptom description of an IT problem is typically accumulated as a short text message, which is a machine generated text with a very problem-specific vocabulary. In traditional IT maintenance procedure, the system administrators utilize their domain knowledge to diagnosis the problem and propose approaches to restore the service according to the symptom description in a ticket. However, this diagnosis and resolving process is extremely labor extensive. Fortunately most service providers keep years' worth of historical tickets with their resolutions. The resolution is usually collected as a free-form text and describes steps taken to remediate the issue described in the ticket. Repeating events generate similar tickets, which in turn have a vast number of repeated problem resolutions likely to be found

in earlier tickets. Therefore, developing a recommendation system to recommend relevant resolutions in historical tickets becomes a appropriate approach towards an automated delivery of a service in ticket resolving, and thus reduce cost and maintain quality. A substantial amount of research has been devoted to recommendation systems. These recommendation systems determine items or products to be recommended based on prior behavior of the user or similar users and on the item itself. An increasing amount of user interactions have provided these applications with a vast amount of historical information that can be converted into practical knowledge. A similar approach and methodology can be developed that finds a resolution for an ticket by making use of similarities between the symptom description and previous symptom descriptions of monitoring tickets. However, measuring the similarities between the symptom descriptions is a very challenging task considering the symptom descriptions are very short, noisy and written in a very domain-specific vocabulary.

3. ***Gradually adapting the resolution recommendation system for time correlated features derived from text.*** Massive heterogeneous applications as well as various monitoring software are running on clients' servers to accomplish required tasks and to monitor system health via different metrics. It leads to generation of tickets that have different symptom descriptions but similar resolutions. Furthermore, change of server's environments can also induce similar situations in which ticket descriptions differ but could have similar resolutions. This phenomenon causes performance degradation in ticket resolution recommendation. The root cause of performance degradation is the fact that features derived from textual description are evolving gradually due to the heterogeneous environment aforementioned. Training and applying a model in data with varying feature space is considered as a domain adaptation task.

4. ***Efficiently learning distributed representation for text using scalable deep neural ranking model.*** Previous approaches for ticket resolution recommendation are hard to scale and limited to learn semantical meanings from historical tickets data, such as the distributed representation for tickets. However, it is an essential task in IT service management to accurately represent the ticket summary and resolution. The classical techniques such as the n-gram, TF-IDF, and LDA are not effective in representing tickets as the ticket summary and resolution are generally not well formatted. In this work, we formulate the ticket resolution recommendation as a learning to rank task in which, given a ticket summary, the model ranks historical resolutions according to their estimated matching scores. Specifically, we propose a deep neural network ranking model capable of outputting effective distributed representation for ticket summary and resolution as an intermediate result. These representations can be used in other ticket analysis tasks, such as ticket classification and clustering. Furthermore, earlier studies generally assumed that the tickets with similar descriptions should have similar resolutions, and often treated all such ticket resolutions equally. However, the study [ZTZ+16] demonstrated that not all of the resolutions are equally worthy recommending.

1.3 Contribution

My dissertation will address the research topics outlined above. Concretely, I will focus on designing and developing data mining solutions to better understand text in different domains, including (1) a storyline generation method for natural hurricanes based on crawled online corpus which summarizing the evolution of the hurricane with temporal and spatial information; (2) a recommendation framework for automated ticket resolution in IT service management; (3) an adaptive recommendation system

on time-varying temporal correlated features derived from text; (4) a deep neural ranking model not only successfully recommending resolutions but also efficiently outputting distributed representation for ticket descriptions and resolutions.

1.3.1 Intelligent storyline generation

Hurricane Sandy affected the east coast of U.S. in 2012 and posed immense threats to businesses, human lives and properties. In order to minimize the consequent loss of a catastrophe like this, a critical task in disaster management is to understand situation updates about the disaster from a large number of disaster-related documents, and obtain a big picture of the disasters trends and how it affects different areas. Intelligent storyline generation about the evolution of natural disaster acts as a highly efficient approach to improve situation awareness in disaster management. The contribution of our work is summarized as below.

1. We present a novel two-layer summarization framework to summarize multiple disaster-related documents. The first layer provides an overall summary of the disaster events, while the second layer gives condensed information on how specific locations/regions were affected by the disaster.
2. We consider both temporal and spatial factors when generating summaries for the disaster events, and these two factors enable us to reason on the evolution of events over time and locations. The generated summaries can be naturally represented as a storyline.
3. We conduct quantitative experiments and case studies on crawled web documents related to three major hurricane disasters, and the results demonstrate the efficacy of our proposed framework in generating readable and understandable summaries.

1.3.2 Automatic ticket resolution

Maximal automation of routine IT maintenance procedures is an ultimate goal of IT service management. System monitoring, an effective and reliable means for IT problem detection, generates monitoring ticket. In light of the ticket description, system administrators determine the root cause of the IT problem and resolve the problem with solutions recorded in ticket as unstructured text. Automatic IT problem resolution acts as a critical part during the routine IT maintenance procedures. The contribution of our work is summarized as below.

1. We analyze historical monitoring tickets from three production accounts and observe that their resolutions are recommendable for current monitoring tickets on the basis of event information.
2. We propose a feature extraction approach capable of representing both the event and resolution information using topic-level features obtained via the LDA model.
3. We propose to further improve the similarity measurement using metric learning when resolution categories are available.
4. We conducted extensive experiments for our proposed algorithms on real ticket datasets, and experimental results demonstrate the effectiveness and efficiency of the proposed approaches.

1.3.3 Adaptive recommendation system on time varying features

In current service environments, massive heterogeneous applications, as well as various monitoring software, running on customers servers to accomplish complex tasks

and to monitor system health via different metrics, lead to generation of correlated tickets that have different symptom descriptions but similar resolutions. Furthermore, evolving over time, service environments cause a further discrepancy. The description of tickets generated before and after change differ but might have similar resolutions since root causes remain unchanged. This heterogeneous IT environment require an adaptive recommendation system since the features derived from symptom description are changing. Our research is based on the domain adaption methodology and the contribution of this work is summarized as below.

1. Based on our observation and initial experiments, we find out that features derived from ticket symptom descriptions are changing and shifting over time but interesting mappings exist in those features.
2. We adopts structural corresponding learning (SCL) to discover the features' mapping and apply it to our ticket resolution recommendation system.
3. Extensive empirical studies on real application ticket data are conducted to demonstrate the effectiveness and the efficiency of the proposed method.

1.3.4 Learn distributed representation via deep neural ranking model

It is an essential task in IT service management to accurately represent the ticket summary and resolution which can be used in many ticket analysis tasks such as ticket classification and clustering. The classical techniques such as the n-gram, TF-IDF, and LDA are not effective in representing tickets as the ticket summary and resolution are generally not well formatted. Learning distributed representation for textual data have been explored in many studies [DDF$^+$90, BTWDR13, CW08]. My

research is based on the combination of deep neural ranking model and sentence model, and the related contributions are listed as below.

1. Carefully identify relevant features and then build a regression model to quantify ticket resolution quality to develop an effective resolution recommendation model, such low-quality resolutions should be ranked lower than high-quality resolutions.
2. Formulate the ticket resolution problem as an integrated deep neural network-based ranking framework and efficient handling those challenges.
3. Generalize the ticket representation and successfully apply to other ticket analysis tasks, such as, ticket classification and clustering.
4. Extensively evaluate on the proposed model against a large real-world dataset. The experimental results show its supremacy to other traditional representations for textual data.

1.4 Summary and Roadmap

More than ever, information delivery online and storage heavily rely on text. Billions of texts are produced every day in the form of documents, news, logs, search queries, ad keywords, tags, tweets, messenger conversations, social network posts, etc. Text understanding is a fundamental and essential task involving broad research topics, and contributes to many applications in the areas text summarization, search engine, recommendation systems, online advertising, conversational bot and so on. However understanding text for computers is never a trivial task, especially for noisy and ambiguous text such as logs, search queries. In my proposal, I present several data mining techniques on understanding textual data to facilitate the knowledge absorption in disaster management and ticket resolving in IT service management.

Herein, the organization of my proposal is outlined to facilitate the reading and understanding the research problems presented in this proposal. First, we briefly presents the preliminaries and related work of the aforementioned research directions in Chapter 2. To be continue, we study the problems related to these research directions in Chapter 3, Chapter 4, Chapter 5, Chapter 6, respectively. Particularly, in Chapter 3, the storyline generation task is studied, where both the temporal and spatial information are considered. In Chapter 4, we focus on the automated ticket resolution task based on historical ticket data, and particular issues are studied on how to measure the relevance in symptom descriptions in the form of text. In Chapter 5, we study the problem about how to adapt the ticket resolution recommendation system on the time-varying temporal correlated features due to the heterogeneous IT service environment. The correlation between features guides the recommendation system to be adaptive. In Chapter 6, we study the ticket resolution recommendation in the perspective of learning to rank using deep neural network. Moreover, efficient representation for tickets can be learned by incorporating sentence model into the ranking model. Finally, in Chapter 7, we conclude the work of this dissertation and discuss the future work along our research.

CHAPTER 2

PRELIMINARIES AND RELATED WORK

This dissertation studies the concrete problems along the aforementioned research directions in applying data mining techniques to understand textual data, specifically online news documents in disaster management and tickets in IT service management, and the corresponding solutions are exhaustively discussed as well. In this chapter, we highlight existing literature studies that are related to our work in this dissertation. In particular, Section 2.1 reviews the existing work related to storyline generation as well as relevant techniques such as text summarization, topic detection and tracking, and existing disaster management tools. Section 2.2 introduces the priori studies related to IT system monitoring and recommendation system. Section 2.3 presents existing literature of transfer learning and its subclass research area domain adaptation what our proposed approach is categorized as. In Section 2.4, we first highlight studies on learning to rank model and surveys relevant work on learning distributed representation and question answer task.

2.1 Related Work of Storyline Generation

The storyline generation problem aims to obtain a sequence of summaries that describe how an event evolves over time from a myriad of web documents. Therefore, the storyline generation problem is typically categorized as a text summarization task. In this section, we highlight some previous research results that are most relevant to this work in the following three directions: multi-document summarization, topic detection and tracking, and storyline generation. We will also discuss several useful disaster situation-specific tools.

2.1.1 Multi-document Summarization

Multi-document summarization is a mechanism which addresses the information overload problem by compressing a given collection of documents into a concise summary. In general, it can be categorized into extractive and abstractive summarization [Man01]. Extractive summarization [RHM02] selects important sentences from the original documents to form a summary, while abstractive summarization [RHM02] paraphrases the corpus using new sentences. The latter usually employs natural language generation techniques such as information fusion, sentence compression and reformulation. Our work is more related to extractive summarization. Various multi-document summarization methods have been proposed over the last decade, including centroid-based [RJB00], graph-based [ER04, SL10], knowledge-based [LL14, LWSL10], and etc. Other methods, such as non-negative matrix factorization, latent semantic analysis, and sentence-based topic models, have also been applied to generate the summaries by selecting semantically important sentences in the documents [WLZD08, SLD11]. Most existing extractive summarization methods generate short summaries by selecting sentence from the input; however, they often ignore the implicit temporal, spatial and structural information possibly presented in the documents.

2.1.2 Topic Detection and Tracking

Topic detection and tracking (TDT) is a research program initiated by DARPA (Defense Advanced Research Projects Agency) for finding and following the new events in streams that broadcast news stories. It consists of three major technical tasks: tracking known events, detecting unknown events, and segmenting a news source into stories. Many promising approaches have been proposed and identified during the TDT evaluation, in particular within the information retrieval and natural language

processing communities [All12, LAD+02, MAMS04]. However, previous research efforts only focused on detecting the flat structure of events, and fail to consider the hidden hierarchies of topics.

2.1.3 Storyline Generation

Storyline generation aims to obtain a sequence of summaries that describe how an event evolves over time, and has attracted great attention recently. For example, Google News Timeline clusters incoming articles into groups based on topics and lists the generated groups in chronological order. Alonso et al. [ABYG09] proposed a framework for generating temporal snippets to improve user search experience. These methods consider the temporal information as references and represent the results in chronological order. Recently, Wang et al. [WLO12] proposed a framework that integrates text, image, and temporal information to generate storyline-based summaries to reflect the evolution of the given topic. Lin et al. [LLL+12b] presents a framework for generating storylines from microblogs for user input queries. Shahaf et al. [SGH12] proposed a methodology called *metro map* for creating structural summaries of documents by optimizing several objectives (e.g., relevance, coherence, coverage and connectivity) simultaneously. Jiang et al. [JPL11] proposed an temporal event summarization solution to summarize the temporal dynamics of the event sequences using the inter-arrival information. Unlike these existing systems, our framework takes into account the spatial information and generates storyline-based summaries to reflect the evolution of a given topic over different geo-spatial regions.

2.1.4 Disaster Situation-specific Tools

Disaster Situation-specific Tools: Commercial systems such as Web EOC and E-Team are usually used by Emergency Management departments located in urban areas [Inc12, NC412]. Recently Ushahidi provides a platform to crowd source news stories and crisis information using multiple channels and prepares visualization and interactive maps [ush12] and GeoVISTA monitors tweets to form situation alerts on a map-based user interface according to the geo-locations associated with the tweets [Geo10]. These situation-specific tools provide query interfaces, GIS and visualization capabilities to support user interaction and query [ZST+13]. However, they do not generate textual storylines to improve the situation awareness.

2.2 Related Work of Ticket Resolution Recommendation

In IT service management, the ticket resolution is usually collected as a free-form text and describes steps taken to remediate the issue described in the ticket. We analyzed historical incident tickets collected from one of the large service providers and noticed that there are many repeating resolutions for tickets within an account. It is natural to expect that if ticket summary are similar, then their respective tickets probably have the same resolution. Therefore, we can recommend a resolution for an incoming ticket based on the ticket summary information and historical tickets.

This section reviews prior research studies related to the automated IT service management and the recommendation system. System monitoring, as part of the automated Service management, has become a significant research area of the IT industry in the past few years.

2.2.1 IT monitoring system

Numerous studies [KRRS08, ADNR07, MJ93, XZB05, ESV03, RLS+98] focus on monitoring that is critical for a distributed network. There are also many commercial products, such as IBM Tivoli [urlb], HP OpenView [urla] and Splunk [urle] that focuses on system monitoring. The monitoring targets include the components or subsystems of IT infrastructures, such as the hardware of the system (CPU, hard disk) or the software (a database engine, a web server). Once certain system alarms are captured, the system monitoring software will generate the monitoring tickets into the ticketing system. Automated ticket resolution is much harder than automated system monitoring because it requires vast domain knowledge about the target infrastructure. Some prior studies apply approaches in text mining to explore the related ticket resolutions from the ticketing database [SCT+08, WLZG11]. Other works propose methods for refining the work order of resolving tickets [SCT+08, MMY+10, ZLSG14a] and discovering the dependency of tickets [TLS12].

A number of studies focused on the analysis of historical events with the goal of improving an understanding of system behaviors. A significant amount of work was done on analysis of system log files and monitoring events. See example, [HMP02, PTG+03, GSSM04]. Another area of interest is the identification of actionable patterns of events and misses, or false negatives, by the monitoring system. False negatives are indications of a problem in the monitoring software configuration, wherein a faulty state of the system does not cause monitoring alerts.

Labor cost is one of the largest costs of IT service providers. Large service providers staff their service centers with hundreds of IT experts who are responsible for resolving various incident tickets every day. Therefore, service providers heavily rely on human efficiency for tasks such as root cause analysis and incident ticket resolution. Automatic techniques of recommending relevant historical tickets

with resolutions can significantly improve the efficiency of technical support in this task. Based on the relevant tickets, a person can correlate related system problems that happened earlier and perform a deeper system diagnosis. Solutions described in relevant historical tickets also provide best practices for solving similar issues.

2.2.2 Recommendation System

With the development of e-commerce, a substantial amount of research has been devoted to the recommendation system. Lots of recommendation algorithms are proposed for promoting products to online users [BK07, DL05, Kor10, Kar01, LMX11]. Recommendation algorithms can be categorized as item-based [Kar01, NK11, SKKR00] and user-based algorithms [BK07, DL05, Kor10, Kar01, TH01]. The difference with our work is that, in e-commerce, products are maintained by reliable sellers. The recommendation algorithms usually do not need to consider the problem of fake or low quality products. But in service management, false tickets are unavoidable. The traditional recommendation algorithms do not take into account the types of tickets and as a result would recommend misleading resolutions.

2.3 Related Work of Domain Adaptation

Massive heterogeneous applications as well as various monitoring software are running on clients servers to accomplish required tasks and to monitor system health in different metrics. It leads to generation of correlated tickets that have different symptom descriptions but similar resolutions. Furthermore, change of servers environments can also bring similar situation in which tickets description differ but might have similar resolutions. These correlated tickets cause performance degradation in ticket resolution recommendation.

The degradation can be alleviated by applying domain adaptation techniques to ensure that the recommendation system can efficiently work in a dynamic environment. In this section, we highlight the existing literature studies on domain adaptation and its superclass transfer learning.

2.3.1 Transfer Learning

Traditional data mining and machine learning algorithms make predictions on the future data using statistical models that are trained on previously collected labeled or unlabeled training data [YHYY06, KR07]. Semisupervised classification [Zhu05, NMTM00, BM98] addresses the problem that the labeled data may be too few to build a good classifier, by making use of a large amount of unlabeled data and a small amount of labeled data. Variations of supervised and semisupervised learning for imperfect data sets have been studied; for example, work [ZW06] have studied how to deal with the noisy class label problems. Nevertheless, most of them assume that the distributions of the labeled and unlabeled data are the same. Transfer learning [PY10], in contrast, allows the domains, tasks, and distributions used in training and testing to be different. In the real world, we observe many examples of transfer learning. For example, we may find that learning to recognize apples might help to recognize pears. Similarly, learning to play the electronic organ may help facilitate learning the piano. The study of Transfer learning is motivated by the fact that people can intelligently apply knowledge learned previously to solve new problems faster or with better solutions.

In transfer learning, we have the following three transfer learning settings: 1) inductive transfer learning, 2) transductive transfer learning, 3) unsupervised transfer learning. In the inductive transfer learning setting, the target task is different from the source task, no matter when the source and target domains are the same or

not. In this case, some labeled data in the target domain are required to induce an objective predictive model $f_T(\cdot)$ for use in the target domain. Research areas Multi-task learning [DYXY07, BH03] and self-taught learning [RBL$^+$07, DYXY08] fit into this category. In the transductive transfer learning setting, the source and target tasks are the same, while the source and target domains are different. In this situation, no labeled data in the target domain are available while a lot of labeled data in the source domain are available. It includes research areas such as domian adaptation, sample selection bias, co-variate shift [DIM06, Zad04, Shi00]. Finally, in the unsupervised transfer learning setting, similar to inductive transfer learning setting, the target task is different from but related to the source task. However, the unsupervised transfer learning focus on solving unsupervised learning tasks in the target domain, such as clustering, dimensionality reduction, and density estimation [WSZ08, DYXY08].

2.3.2 Domain Adaptation

Domain adaptation is well studied area. Roark and Bacchiani [RB03] use a Dirichlet prior on the multinomial parameters of a generative parsing model to combine a large amount of training data from a source corpus and a small amount of training data from a target corpus. Several authors have also given techniques for adapting classification to new domains. Chelba et al. [CA06] first train a classifier on the source data and then apply the maximum a posteriori estimation of the weights of a maximum entropy on a target domain classifier in which the Gaussian prior has a mean equal to the weights of the source domain classifier. Daumé III and Marcu [DIM06] use an empirical Bayes model to estimate a latent variable model grouping instances into domain-specific or common across both domains. Our work focuses on applying structural corresponding learning (SCL) to find a common representation for features from different tickets to favor ticket resolution recommendation.

Finally we note that SCL is first introduced in the work of Ando et al. [AZ05], and later Blitzer combines SCL with labeled target domain data, they compared the two using the label of SCL or non-SCL source classifiers as features. Several applications of SCL have been studied in papers [BMP06, BDP+07]. Unlike these applications, we apply SCL to our ticket resolution recommendation task and pick up the pivot features from both source and target labeled tickets. We show that we can make better use of SCL to discover a useful feature mapping in our real-work ticket data and improve performance of our ticket resolution recommendation task.

2.4 Related Work of Deep Neural Ranking Model

In our previous work, we formulate the automated ticket resolution task as a recommendation problem. However, with potential scalability and efficacy of deep neural ranking model, it is natural to model it as a learning to rank task utilizing deep neural network. Furthermore, it is an essential task in IT service management to accurately represent the ticket summary and resolution which can be used in many ticket analysis tasks such as ticket classification and clustering. The classical techniques such as the n-gram, TF-IDF, and LDA are not effective in representing tickets as the ticket summary and resolution are generally not well formatted. Learning distributed representation for textual data have been explored in many studies [DDF+90, BTWDR13, CW08]. This section mainly studies the learning to rank model and learning distributed representation for sentences.

2.4.1 Learning to Rank

Our learning to rank method is based on a deep learning model for advanced text representations using distributional word embeddings. Distributional representations

have a long tradition in IR, e.g., Latent Semantic Analysis [DDF+90], which more recently has also been characterized by studies on distributional models based on word similarities. Their main properties is to alleviate the problem of data sparseness. In particular, such representations can be derived with several methods, e.g., by counting the frequencies of co-occurring words around a given token in large corpora. Such distributed representations can be obtained by applying neural language models that learn word embeddings [BTWDR13] and more recently using recursive autoencoders [VLL+10], and convolutional neural networks [CW08].

Our work more directly targets the task of answer sentence selection, i.e., the task of selecting a sentence that contains the information required to answer a given question from a set of candidates (for example, provided by a search engine). In particular, the state of the art in answer sentence selection is given by work [WSM07], that use quasi-synchronous grammar to model relations between a question and a candidate answer with the syntactic transformations. The model of Yao et al., 2013 [YVDCBC13] applies linear chain CRFs with features derived from TED to automatically learn associations between questions and candidate answers. Severyn and Moschitti [SM13] applied SVM with tree kernels to shallow syntactic representation, which provide automatic feature engineering. Yih et al. [YCMP13] use distributional models based on lexical semantics to match semantic relations of aligned words in QA pairs.

The work closest to ours is [YHBP14], where they apply deep learning to learn to match question and answer sentences. However, their sentence model to map questions and answers to vectors operates only on unigrams or bigrams. Our sentence model is based on a convolutional neural network with the state-of-the-art architecture, we use a relatively large width of the convolution filter, thus allowing the network to capture longer range dependencies. Moreover, the architecture of deep

learning model along with the question-answer similarity score also encodes question and answer vector representations in the model. Hence, our model constructs and learns a richer representation of the question-answer pairs, which results in superior results on the answer sentence selection dataset.

Language models were applied to definitional QA in [CKC05, Sas05, SSM02]. Regarding solve QA task in the perspective of language translation, Ritter et al. [RCD11] have investigated the feasibility of conducting short text conversation by using statistical machine translation (SMT) techniques, as well as millions of naturally occurring conversation data in Twitter. In the approach, a response is generated from a model, not retrieved from a repository, and thus it cannot be guaranteed to be a legitimate natural language text.

2.4.2 Summary

This chapter highlights the existing works in the literature, which are highly related to the four research directions of my dissertation, i.e., storyline generation, resolution recommendation, domain adaptation and learning distributed representation from a learning to rank model. For each research direction, both the related approaches and evaluation metrics are exhaustively surveyed.

CHAPTER 3

GENERATING TEXTUAL STORYLINE FOR DISASTER

Hurricane Sandy affected the east coast of U.S. in 2012 and posed immense threats to businesses, human lives and properties. In order to minimize the consequent loss of a catastrophe like this, a critical task in disaster management is to understand situation updates about the disaster from a large number of disaster-related documents, and obtain a big picture of the disaster's trends and how it affects different areas. In this chapter, we present a novel two-layer storyline generation framework which generates an overall or a global storyline of the disaster events in the first layer, and provides condensed information about specific regions affected by the disaster (i.e., a location-specific storyline) in the second layer. To generate the overall storyline of a disaster, we consider both temporal and spatial factors, which are encoded using integer linear programming. While for location-specific storylines, we employ a Steiner tree based method. Compared with the previous work of storyline generation, which generates flat storylines without considering spatial information, our framework is more suitable for large-scale disaster events. We further demonstrate the efficacy of our proposed framework through the evaluation on the datasets of three major hurricane disasters.

3.1 Introduction

Natural disasters such as hurricanes, earthquakes and tsunamis cause inestimable physical destruction, loss of life and property around the world every year. For example, Hurricane Sandy affected the east coast of U.S. in 2012 and posed immense threats to businesses, human lives, and properties. In order to minimize the consequent loss of the disasters, a critical task in disaster management is to efficiently analyze and understand the disaster-related situation updates. This requires effective

information gathering methods to operate on a myriad of web documents, e.g., news and reports that are related to the disasters. The domain experts expect to obtain condensed information about the detailed disaster event description, e.g., the evolutionary tendency of the disaster with respect to different locations [LL14]. However, it is often a non-trivial task to generate a big picture of the disaster events due to the flood of web documents.

To tackle this problem, various types of document understanding systems have been developed over the last decade. These systems include (1) summarization-based systems [LLL12a, RJST04, SBC03, SL10, WLLH08] that choose from multiple documents a subset of sentences conveying the principle idea; (2) topic detection and tracking systems [All12] aiming to group documents into different clusters as events and monitor future events related to the corresponding topic; and (3) timeline generation systems [SGH12, WLO12] that create summaries to present the evolution of an event by leveraging temporal information attached to or extracted from the documents. These systems are able to alleviate the so-called *information overload* problem to some extent; however, they suffer from several limitations that may affect the quality of the summarized results. First, most of them focus on summarizing an event via topic evolution over the time, but ignore the spatial information which is important especially for large-scale disaster events. For instance, for a hurricane which affects several states of U.S., a domain expert may be interested in how these regions are affected, and how the hurricane evolves over different geo-spatial regions. Second, these systems usually generate a single layer summarization or storyline to reflect topic changes over the entire event. However, due to the spatial factor, the information evolution over a disaster event is intrinsically hierarchical. In most cases, domain experts are often interested in not only the general picture of a disaster, but also how it affects a particular region.

3.2 Research Objective

In this chapter, we propose a storyline generation framework that addresses the aforementioned limitations by generating a two-layer storyline that consists of global storylines for cross-location disaster events on the first layer and location-specific storylines for individual events on the second layer. Specifically, in our framework, a disaster event is initially summarized from a large set of documents (e.g., news and reports) with a big picture showing how the disaster affects different regions. It can then be zoomed into a specific location for more detailed location-specific event summarization. In the cross-location layer, integer linear programming is employed to summarize the event via a list of representative locations, each of which is associated with a short description. On the location-specific layer, a Steiner-tree based approach is applied to generate a storyline for each specific location.

In summary, the contributions of this work are three-fold:

- We present a novel two-layer summarization framework to summarize multiple disaster-related documents. The first layer provides an overall summary of the disaster events, while the second layer gives condensed information on how specific locations/regions were affected by the disaster.
- We consider both temporal and spatial factors when generating summaries for the disaster events, and these two factors enable us to reason on the evolution of events over time and locations. The generated summaries can be naturally represented as a storyline.
- We conduct quantitative experiments and case studies on crawled web documents related to three major hurricane disasters, and the results demonstrate the efficacy of our proposed framework in generating readable and understandable summaries.

The rest of the chapter is organized as follows. We first define our problem in Section 3.3. In Section 3.4, an overview of our proposed framework is introduced. Detailed descriptions of how to generate a global storyline and a local storyline are presented in Section 3.5 and Section 3.6, respectively. We evaluate our system in Section 3.7 and finally conclude this chapter and discuss potential extensions of the proposed framework in Section 3.8.

3.3 Problem Definition

To summarize what is happening in the vicinity of a given disaster, we present a storyline of the disaster in the form of a two-layer graph of events.

Definition 3.3.1 *An* event *is represented by a tuple* (t, l, s) *where* t *is the time that the event occurs,* l *is the location and* s *is the textual description about the event. For example, (08/27/2011, New York City, "The five main New York City-area airports will be closed to arriving flights") represents an event in Hurricane Sandy.*

The problem of generating a storyline can be defined as follows:

Input: A collection of documents related to a disaster.

Output: A two-layer storyline consists of the most representative events summarizing the evolution of disaster-relevant topics. The **first layer** (or the upper layer) is a chain of events $(o_1, \ldots, o_n)$, as the global temporal and spatial evolution of a disaster, therefore also referred as the global storyline. An event of the upper layer o_i can be further expanded in the **second layer** (or the lower layer) to a connected tree of events as the temporal and topic evolution locally for a specific location of o_i.

A global storyline, which is a chain of events, describes how the disaster moves over time by the location attribute of the events and how the disaster affects different areas by the description attributes. The chain structure is used under the assumption

that a disaster at any time should have only one geo-spatial center, which should move continuously over time. Such an assumption is valid for most of the natural disasters like hurricanes, storms, and blizzards, but not for the man-made disasters like cyber attacks. In our future work, we will explore more complicated evolution structures of different disaster types. For local storyline generation, we follow previous work of storyline generation [WLO12] to use a tree structure as the storyline to capture more topics in the topic evolution, allowing multiple topics to coexist at the same time.

3.4 System Framework

Figure 3.1 shows our system framework. Given a collection of documents related to a disaster, we first extract text snippets as sentences with time and location phrases, which are identified by Stanford NER [FGM05]. Time phrases are normalized by SUTime [CM12] to timestamps and location phrases are mapped to geocodes by Google API Together with its timestamp and geocode, a snippet approximately describes an event.

In our framework, the extracted text snippets are first organized as a similarity graph, followed by two layers of processing, corresponding to the two layers of the output. In the first layer, a minimum dominating set algorithm is employed on the snippet graph to find several representative events, on top of which an integer linear programming method is then proposed to find a chain of events reflecting the overall spatial evolution of the disaster as the global storyline. We visualize the global storyline on a map using Google map APIs.

If a user is interested in certain area and click it on the map, the map will be zoomed-in the clicked area and display the local storyline of the area. To do this, a sub-graph of the overall similarity graph is first induced and augmented to a multi-view graph. The same minimum dominating set algorithm is first applied to the sub-

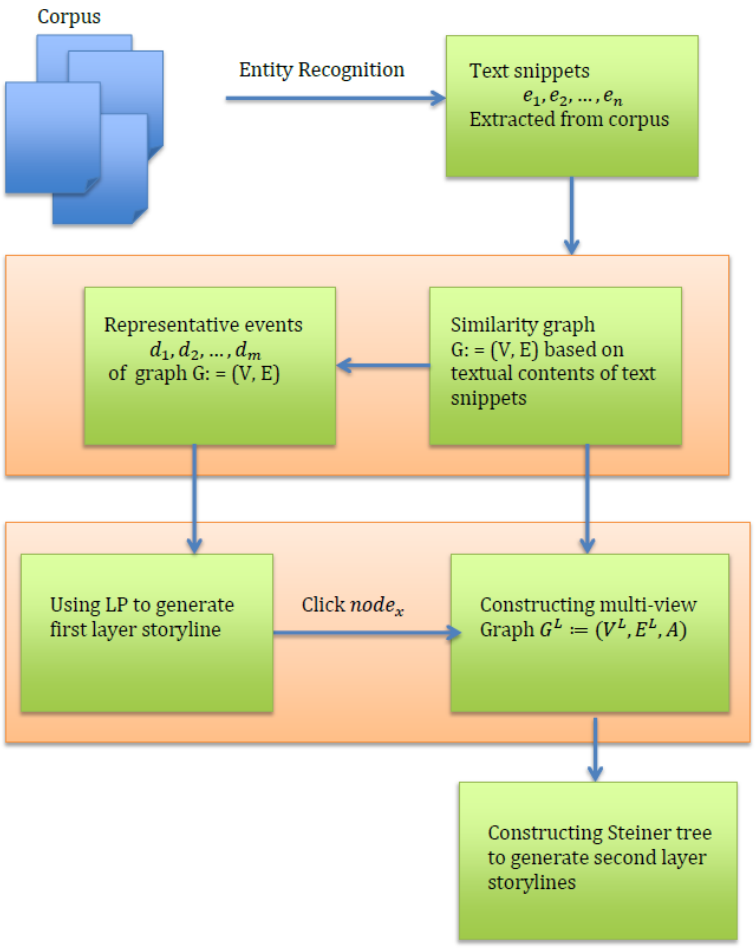

Figure 3.1: The high-level system overview.

graph for finding representative events, and then followed by a Steiner tree algorithm to make the selected events temporally smooth and coherent.

3.5 Global Storyline Generation

3.5.1 Text Snippet Graph Construction

Although each text snippet can be considered as an event, many of those are redundant. To remove the redundancy and obtain a set of representative events, we

construct a graph $G = (V, E)$ with the given text snippets as the vertex set V, and add an edge between each pair of snippets which are likely to refer to the same event. Specifically, for two nodes $v_i, v_j \in V$, we first convert these two text snippets into two feature vectors as n-gram bags-of-words, then compute the cosine similarity between these two feature vectors. $e_{ij} = (v_i, v_j) \in E$ if and only if both the similarity of v_i and v_j is greater than a similarity threshold parameter α, and their distance calculated by their geocode is less than a distance threshold parameter *radius*. Note that the latter constraint takes the spatial smoothness of events into consideration.

3.5.2 Identifying Events via Dominating Set

We identify the set of representative events in the original snippets with minimum redundancy by solving the minimum dominating set problem. A vertex u of a graph dominates another vertex v of the graph, if u and v are joined by an edge in the graph. A subset of S of the vertex set of an undirected graph is a dominating set if for each vertex u, either u is in S or a vertex in S dominates u. The *Minimum Dominating Set* (MDS) problem is to find a dominating set with minimum size. MDS has been previously used to model multi-document summarization problem [SL10]. In our case, we use the MDS of text snippets to capture the representative events from the text snippets of disaster event descriptions.

The MDS problem is known to be NP-hard but an efficient greedy algorithm by Johnson [Joh74] is known to achieve an approximation ratio of $H(d + 1)$, where d is the maximum degree of the graph and $H(n) = \sum_{i=1}^{n} \frac{1}{i}$ is the harmonic function. Johnson's greedy algorithm was initially designed for the Set Cover problem, but it is well-known that there is an L-reduction between MDS and Set Cover. The greedy algorithm is described in Algorithm 1 and was also used in [SL10].

Algorithm 1 Greedy MDS Approximation Algorithm.

INPUT: Graph $G = (V, E)$, MDS upper bound W
OUTPUT: dominating set S

$S \leftarrow \emptyset$
$T \leftarrow \emptyset$
while $|S| < W$ and $S \neq V(G)$ **do**
  for $v \in V(G) - S$ **do**
    $s(v) \leftarrow |N(v) \setminus T|$
  end for
  $v^* \leftarrow \arg\max_v s(v)$
  $S \leftarrow S \cup \{v^*\}$
  $T \leftarrow T \cup N(v^*)$
end while

3.5.3 Storyline Generation by Connecting Dominating Objects via Linear Programming (LP)

Using Algorithm 1, we generate the dominating set of $G(V, E)$, m text snippets $d_1, \ldots, d_m$, as the representative events. Without loss of generality, the set of events are assumed to be in chronological order. To generate a global storyline capturing the major location change of the disaster, we select a sequence of nodes $o_1, o_2, \ldots, o_l$ from the representative events in chronological order. Intuitively, the generated storyline should also be in spatial coherence, reflecting the continuous location change of the disaster over time. Since a disaster is likely to affect adjacent areas in a similar fashion, the storyline should be coherent in content as well.

Based on the above discussions, we model the storyline generation problem using integer linear programming. To select a chain of nodes from $d_1, \ldots, d_m$, we use variables $\textit{node-active}_i \in \{0, 1\}, i = 1 \ldots m$ to indicate whether d_i is included in the selected chain, and $\textit{next-node}_{ij} \in \{0, 1\}, i, j = 1 \ldots m$ to indicate that d_i and d_j are two successive nodes (i.e., a transition) in the chain. The objective function aims to maximize the storyline's content coherence which is defined as the minimal similarity

between two successive nodes along the storyline as shown below:

$$Coherence(o_1, o_2, \ldots, o_n) = \min_{i=1,2,\ldots,n-1} similarity(o_i, o_{i+1}).$$

We further impose the following set of constraints to model storyline's spatial coherence.

Chain Constraints: It should be guaranteed the consistency of variables $node\text{-}active_i$ and $next\text{-}node_{ij}$, and that the selected nodes should compose a chain in chronological order.

// A node has at most one in-edge and at most one

// out-edge

$$\forall_j : \sum_i next\text{-}node_{i,j} \leq node\text{-}active_j, \tag{3.1}$$

$$\forall_i : \sum_j next\text{-}node_{i,j} \leq node\text{-}active_i. \tag{3.2}$$

// The number of active transitions is equal to the

// number of active nodes minus one

$$\sum_i node\text{-}active_i - \sum_{i,j} next\text{-}node_{i,j} = 1. \tag{3.3}$$

// The chain is ordered chronologically:

$$\forall_{i>j} : next\text{-}node_{i,j} = 0. \tag{3.4}$$

// A transition of two node can not be active if

// there exists an active node between them.

$$\forall_{i<k<j} : next\text{-}node_{i,j} \leq 1 - node\text{-}active_k. \tag{3.5}$$

Length Constraints: The selected chain should be in a reasonable length ranged between pre-defined minimum length threshold $\mathcal{L}_{min}$ and maximum length threshold

$\mathcal{L}_{max}$.

$$\mathcal{L}_{min} \leq \sum_i node\text{-}active_i \leq \mathcal{L}_{max}. \tag{3.6}$$

Location Smoothness Constraints: We require both pairwise and triple-wise smoothness of location change on the selected chain. Let $\mathcal{D}_{i,j}, i, j = 1, \ldots, m$ be the distance based pairwise location relationship between d_i and d_j, and $\mathcal{D}_{i,j} = 1$ if distance between d_i and d_j is less than a pre-defined distance parameter, $\mathcal{D}_{i,j} = 0$ otherwise. For triple-wise smoothness, let $\mathcal{A}_{i,j,k}$ be the angle based triple-wise location relationship, and $\mathcal{A}_{i,j,k} = 1$ indicates the angle constructed by three successive nodes d_i, d_j and event k is not an acute one, otherwise $\mathcal{A}_{i,j,k} = 0$. By not including in the chain three successive nodes of which the angle is acute, we excludes the back-and-forth events from the storyline and smooth the location change.

$$\begin{aligned}
&// \text{ Distance of two successive nodes should be} \\
&// \text{ within some range} \\
&\forall_i : \sum_j (1 - \mathcal{D}_{i,j}) \cdot next\text{-}node_{i,j} \leq 0. \qquad (3.7) \\
&// \text{ Three successive nodes can not construct} \\
&// \text{ an acute angle} \\
&\forall_{i,j,k} : next\text{-}node_{i,j} + next\text{-}node_{j,k} \leq 1 + \mathcal{A}_{i,j,k}. \qquad (3.8)
\end{aligned}$$

Minimal Similarity Constraints: Let $\mathcal{S}_{ij}, i, j = 1 \ldots, m$ be the cosine similarity between d_i and d_j. we can use the following constraints to find the similarity of the minimum similar transition *min-edge* among active transitions.

$$\forall_{i,j} : min\text{-}edge \leq 1 - (1 - \mathcal{S}_{i,j}) \cdot next\text{-}node_{i,j} \tag{3.9}$$

The Objective Function: Besides to maximize minimal similarity between two successive nodes along the storyline, we also try to make storyline as long as possible,

so the objective function has the following form

$$\text{Maximize: } \textit{min-edge} + \lambda \cdot l, \tag{3.10}$$

where λ is a coefficient parameter.

Although integer linear programming is an NP-hard problem, there are efficient approximation algorithms and implementations such as IBM CPLEX [CPL09] , which is used for optimization in this chapter.

3.6 Local Storyline Generation

A global storyline presents a general high-level picture of how a disaster affects different areas when it hits these areas. To show how the disaster affects a specific area locally for a longer time period during preparation and recovery, we allow users to zoom-in to a node $node_x$ of the global storyline. Once a user clicks the node $node_x$, a new graph $G^L(V^L, E^L)$ will be constructed, which is an induced sub-graph of $G(V, E)$, where V^L includes all text snippet nodes which are close to $node_x$ according to their associated geocodes. For the graph $G^L(V^L, E^L)$, we employ the storyline generation method proposed in [WLO12] to generate a storyline for the selected area.

3.6.1 Augmented Multi-view Graph Construction

Definition 3.6.1 *A* multi-view graph *is a triple $G = (V, E, A)$, where V is a set of vertices, E is a set of undirected edges, and A is a set of directed edges.*

Different from the global storyline generation where the temporal and spatial information of text snippets are modeled by integer linear programming, here we incorporate temporal information in an augmented multi-view graph $G^L = (V^L, E^L, A)$ from $G^L = (V^L, E^L)$, where A is a set of directed edges for temporal relationship

between events. To define edges in A, we introduce two additional parameters $\tau_1, \tau_2, 0 < \tau_1 < \tau_2$. For every pair of nodes o_i, o_j in V, we draw an arc from o_i to o_j if $\tau_1 < t_j - t_i < \tau_2$, where t_i, t_j are the timestamps of o_i and o_j, respectively.

3.6.2 Generating Storylines via Directed Steiner Tree

Similar to generating global storylines, after extracting a dominating set of $G^L = (V^L, E^L)$ which represent the main content topics, we need to generate a storyline capturing the temporal and structural information of the local event descriptions. To tackle this problem, we use the concept of Steiner Tree. A *Steiner tree* of a graph G with respect to a vertex subset X is the edge-induced subtree of G that contains all the vertices in X with minimum cost, where the cost is often measured by the size of the tree.

Problem 1 *Given a directed graph $G = (V, A)$, a set X of vertices (called* terminals*), and a root $v_0 \in X$ from which every vertex of X is reachable in G, find the subtree of G rooted at v_0 containing X with the smallest total vertex weight.*

This problem is known to be NP-hard since the undirected version is already NP-hard. While the undirected version has been well studied, much less work has been done on directed version [CCC+99]. An intuitive solution for this problem is to find the shortest path from the root to each of the terminal and then merge the paths. Of course, this does not guarantee the optimal solution.

We make use of an algorithm due to Charika et al. [CCC+99]. The algorithm takes a level parameter $i \geq 1$. In addition, it takes as input the target terminal set Y, the root r, and the required number of nodes to cover, k. When $i = 1$, it leads to the intuitive solution: i.e., selecting the top k shortest path from the root to k nodes and return the union of those paths. Let the length of every arc $(u, v) \in A$ is 1. We will

make initial call of $A_i(k, v_0, X)$ with X is the dominating set calculated by Algorithm 1 based on graph G, v_0 is the event among X with the earliest timestamp, and k is $|X|$, the size of X. We interpret the output tree as a local storyline evolving from the root event to all the other dominating events. For a constant i, the algorithm is known to run in polynomial time and produces an $O(k^{\frac{1}{i}})$-approximate solution [CCC+99].

Algorithm 2 $A_i(G, k, r, X)$

INPUT: $G = (V, A)$: directed multi-view graph
X : target vertex set X
$r \in X$: the root X
$k \geq 1$: the target size X
OUTPUT: T: a Steiner tree rooted at r covering at least k vertices in X

```
T = ∅
while k > 0 do
    T_best ← ∅
    cost(T_best) ← ∞
    for each vertex v, (v_0, v) ∈ A, and k', 1 ≤ k' ≤ k do
        T' ← A_{i-1}(k', v, X) ∪ {(v_0, v)}
        if cost(T_best) > cost(T') then
            T_best ← T'
        end if
        T ← T ∪ T_best
        k ← k − |X ∩ V(T_best)|
        X ← X \ V(T_best)
    end for
end while
return T
```

3.7 System Evaluation

3.7.1 Datasets

We collect datasets from Bing News Search using keywords about three major hurricanes in the last ten years (i.e., Hurricane Katrina, Hurricane Irene, and Hurricane

keyword	#documents	#text snippets
Hurricane Katrina	800	1572
Hurricane Sandy	795	2253
Hurricane Irene	691	2186

Table 3.1: Statistics of the datasets.

content	time	location
This photo made available by the New Jersey governor's office shows flooding and damage in Seaside Heights, N.J. on Oct. 30, 2012 after super-storm Sandy made landfall in the state.	2012-10-30	New Jersey — Seaside Heights N.J.
October 22, 2012 - Sandy develops into a tropical storm in the Caribbean Sea.	2012-10-22	Caribbean Sea
October 24, 2012 - Hurricane Sandy makes landfall near Kingston, Jamaica, with winds of 80 mph.	2012-10-24	Kingston Jamaica
By Patrick Clark September 26, 2013 Business owners pile muddy furniture outside their building off Canon Avenue in Manitou Springs, Colo.	2013-09-26	Manitou Springs Colo.

Table 3.2: Events example extracted from document using entity recognition.

Sandy) to evaluate our storyline generation system. For the search results returned from Bing News Search, we extract the text content from the corresponding web pages. Basic statistics about the datasets are shown in Table 3.1, and some examples of extracted text snippets are shown in Table 3.2.

3.7.2 Summarization Performance of Global Storylines

To evaluate the quality of global storylines generated by our proposed framework, a human labeler manually composed global storylines for the three hurricane disasters, which are compared with system-generated storylines using ROUGE [LH03] toolkit (version 1.5.5). ROUGE is widely applied by DUC for summarization performance

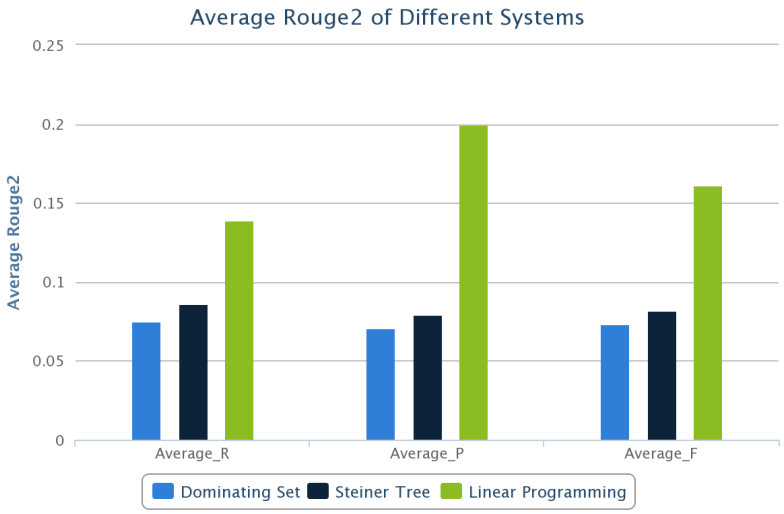

Figure 3.2: Average recall, precision, F-1 of ROUGE-2.

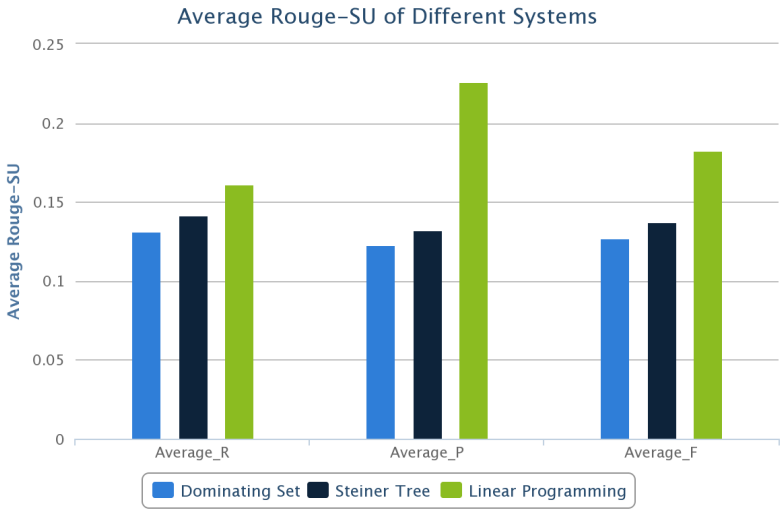

Figure 3.3: Average recall, precision, F-1 of ROUGE-SU4.

evaluation. It measures the quality of a summary by counting the unit overlaps between the candidate summary and a set of reference summaries. Several automatic evaluation methods are implemented in ROUGE, such as ROUGE-N, ROUGE-L, ROUGE-W and ROUGE-SU. ROUGE-N is an n-gram recall computed as follows:

$$\text{ROUGE-N} = \frac{\sum_{S \in \text{ref}} \sum_{\text{gram}_n \in S} \text{Count}_{\text{match}}(\text{gram}_n)}{\sum_{S \in \text{ref}} \sum_{\text{gram}_n \in S} \text{Count}(\text{gram}_n)}, \tag{3.11}$$

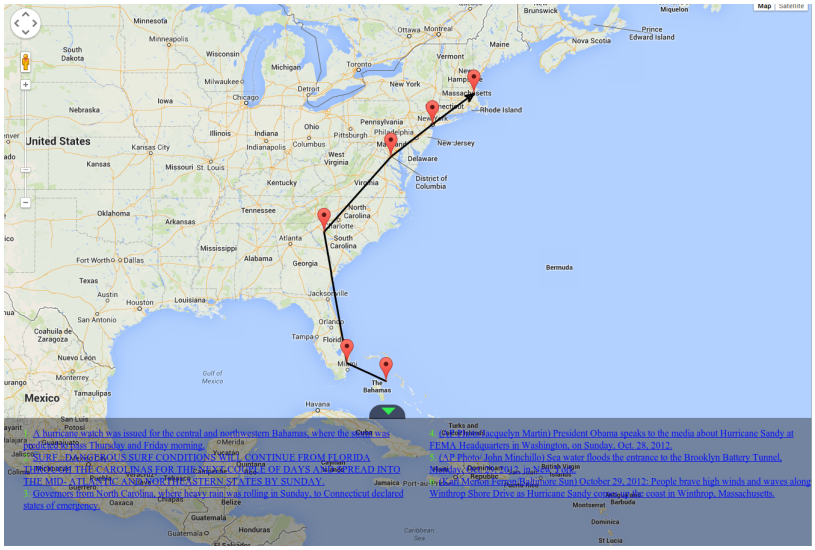

Figure 3.4: Experimental result for hurricane sandy.

Figure 3.5: Wikipeadia ground truth for hurricane sandy.

where n is the length of the n-gram, and ref stands for the reference summaries. $\text{Count}_{\text{match}}(\text{gram}_n)$ is the maximum number of n-grams co-occurring in a candidate summary and the reference summaries, and $\text{Count}(\text{gram}_n)$ is the number of n-grams

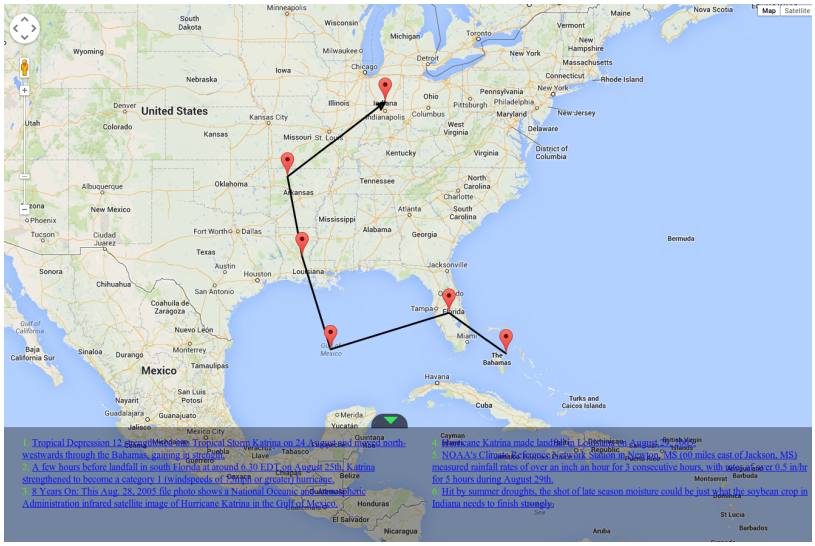

Figure 3.6: Experimental result for hurricane katrina.

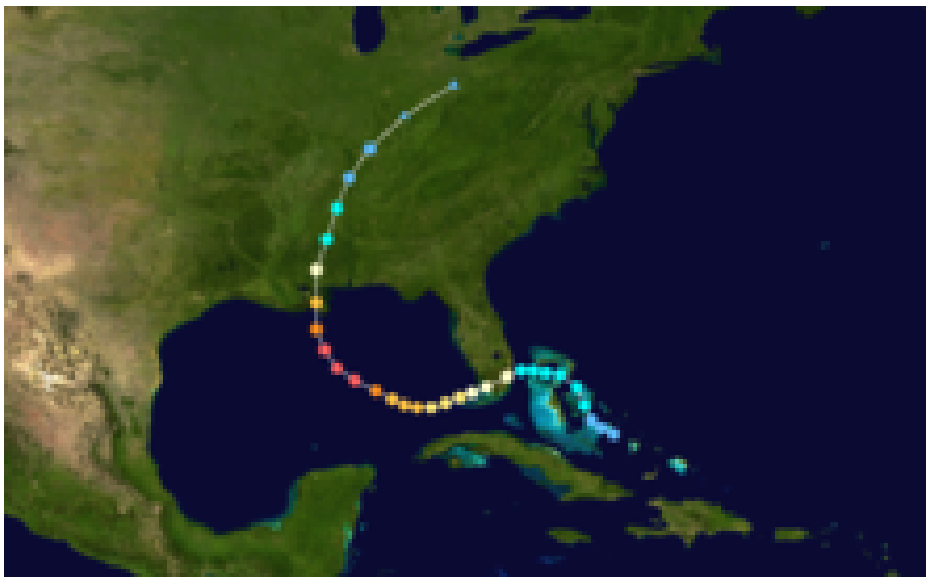

Figure 3.7: Wikipeadia ground truth hurricane katrina.

in the reference summaries. ROUGE-SU4 is based on skip-bigram plus unigram, where skip length is 4.

We compare the global storylines generated by our proposed method considering geo-spatial information with the results from the following methods:

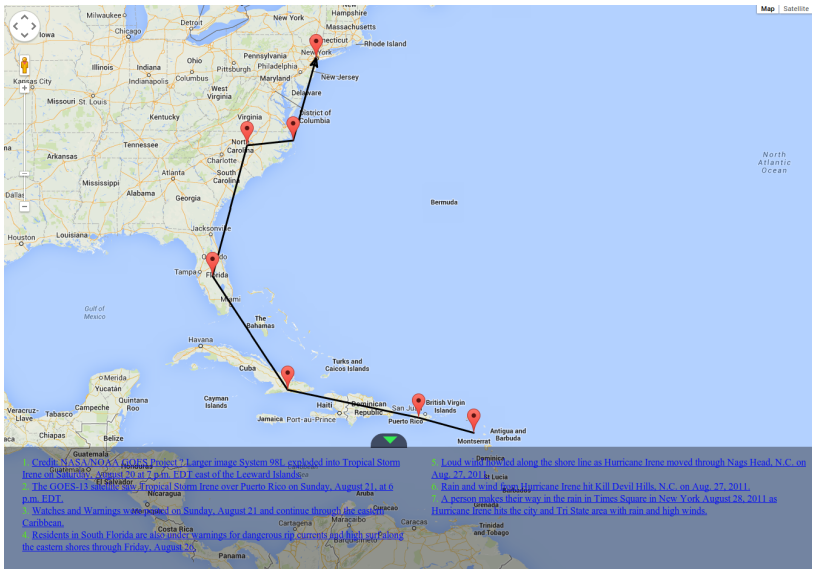

Figure 3.8: Experimental result for hurricane irene.

Figure 3.9: Wikipeadia ground truth hurricane irene.

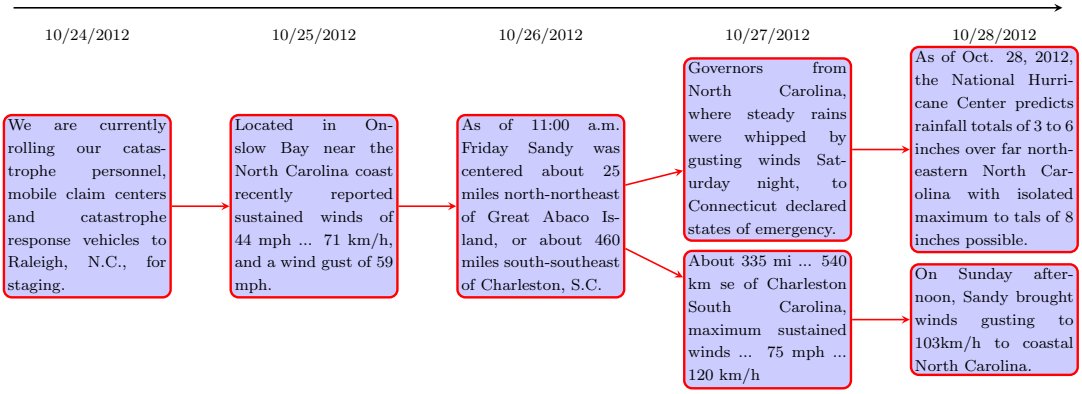

Figure 3.10: An illustrative example of the local storyline for the area of the Carolinas during Hurricane Sandy.

1. Steiner tree based storyline generation [WLO12], which does not consider geospatial information;

2. dominating set based summarization method [SL10], which is a standard multi-document summarization method.

Figure 3.2 and Figure 3.3 show the performance comparison of the three methods using ROUGE-2 and ROUGE-SU4, respectively.

We can observe that the Streiner tree based storyline generation method outperforms the pure multi-document summarization method that does not incorporate the temporal information. Our proposed storyline generation method, which considers both the temporal and spatial information, performs the best among all three methods.

3.7.3 A Case Study

A case study is conducted to demonstrate the effectiveness of the storylines generated using our proposed method. We draw the global storyline generated by our proposed method using Google Map API (shown in Figure 3.4, 3.6, 3.8) and compare

it with the storm paths downloaded from Wikipedia (shown on the right sub-figures in Figure 3.5, 3.7, 3.9).

We can observe that the paths in our generated storylines are similar with the ground truth. The differences are: 1) in addition to show the real paths, our generated storylines can reflect more information about how the hurricanes affect different areas; and 2) the generated storylines not only shows how hurricanes move but also present text descriptions about the status updates and damages they cause along the movement. With the geo-temporal storyline, users can easily capture the overall situation evolution of a disaster.

Figure 3.10 shows an illustrative example of a local storyline when we are interested in a specific area like Carolina during Hurricane Sandy. We can see how Hurricane Sandy affects the area during the period of time and covering different topics like wind and rain.

3.8 Summary

In this chapter, we present a novel storyline framework for summarizing multiple disaster-related documents to generate a two-layer hierarchical storyline to improve situation awareness during or after disasters. We organize the storyline as a two layer hierarchical structure to naturally describe a large-scale disaster. Especially both temporal and spatial factors are considered in the global storyline generation capturing spatial evolution of the disaster over time.

In our future work, we will first explore more complicated evolution structures of different disaster types for storyline generation. We will also extend our framework to incorporate more disaster types like earthquakes and other man-made disasters.

CHAPTER 4

AUTOMATE TEXTUAL RESOLUTION RECOMMENDATION

In recent years, IT Service Providers have been rapidly transforming to an automated service delivery model. This is due to advances in technology and driven by the unrelenting market pressure to reduce cost and maintain quality. Tremendous progress has been made to date towards attainment of truly automated service delivery; that is, the ability to deliver the same service automatically using the same process with the same quality. However, automating Incident and Problem Management continuous to be a difficult problem, particularly due to the growing complexity of IT environments.

Software monitoring systems are designed to actively collect and signal event occurrances and, when necessary, automatically generate incident tickets. Repeating events generate similar tickets, which in turn have a vast number of repeated problem resolutions likely to be found in earlier tickets. In this work, we find an appropriate resolution by making use of similarities between the events and previous resolutions of similar events. Traditional KNN (K Nearest Neighbor) algorithm has been used to recommend resolutions for incoming tickets. However, the effectiveness of recommendation heavily relies on the underlying similarity measure in KNN. In this work, we significantly improve the similarity measure used in KNN by utilizing both the event and resolution information in historical tickets via a topic-level feature extraction using the LDA (Latent Dirichlet Allocation) model. In addition, when resolution categories are available, we propose to learn a more effective similarity measure using

metric learning. Extensive empirical evaluations on three ticket data sets demonstrate the effectiveness and efficiency of our proposed methods.

4.1 Introduction

Today's competitive business climate, as well as the complexity of service environments, dictate the need for efficient and cost-effective service delivery and support. This is largely achieved through service-providing facilities that collaborate with system management tools, combined with automation of routine maintenance procedures such as problem detection, determination and resolution for the service infrastructure [MSGL09, TLP+12, ABD+07, WE11, YPZ10]. Automatic problem detection is typically realized by system monitoring software, such as IBM Tivoli Monitoring [urlc] and HP OpenView [urla]. Monitoring continuously captures the events and generates incident tickets when alerts are raised. Deployment of monitoring solutions is a first step towards fully automated delivery of a service. Automated problem resolution, however, is a hard problem.

With the development of e-commerce, a substantial amount of research has been devoted to recommendation systems. These recommendation systems determine items or products to be recommended based on prior behavior of the user or similar users and on the item itself. An increasing amount of user interactions have provided these applications with a vast amount of historical information that can be converted into practical knowledge. In this chapter we apply a similar approach and develop a methodology that finds a resolution for an event by making use of similarities between the events and previous resolutions of monitoring tickets. Most service providers keep years' worth of historical tickets with their resolutions. The resolution is usually collected as a free-form text and describes steps taken to remediate the issue described in the ticket. We analyzed historical monitoring tickets collected from

three different accounts managed by one of the large service providers (an account is as an aggregate of services that use common infrastructure). We noticed that there are many repeating resolutions for monitoring tickets within an account. It is natural to expect that if events are similar, then their respective tickets probably have the same resolution. Therefore, we can recommend a resolution for an incoming ticket based on the event information and historical tickets.

A KNN-based approach has been proposed in [TLSG13] to provide resolution recommendations for incoming tickets in service management. Although the approach has been successfully used in practice, it has the following two major limitations:

- **Representation of monitoring tickets:** In the KNN-based approach, attribute-based features are used to represent monitoring tickets. However, attribute-level feature representation is not interpretable and often contains lots of noise. In practice, each monitoring ticket describes the existing problems (e.g., low capacity, high CPU utilization) in service and the associated ticket resolutions should be highly relevant to the problems. Therefore, it is better to use features semantically capturing these problems, instead of attribute-level features, to represent monitoring tickets.
- **Similarity Measurement:** The similarity measure used in [TLSG13] only considers the event information, and ignores the related resolutions. In addition, each attribute is treated equally when computing the similarity measure. However, the resolutions often reveal their prevalence in historical tickets and contain important information about the events, which can be used to improve the recommendation performance. Moreover, different attributes should have different weights in computing the similarity measure as they often play different roles in representing the tickets.

In this work, we propose an approach to address the aforementioned limitations in recommending ticket resolutions for service management. In particular, we make the following contributions:

- We analyze historical monitoring tickets from three production accounts and observe that their resolutions are recommendable for current monitoring tickets on the basis of event information.
- We propose a feature extraction approach capable of representing both the event and resolution information using topic-level features obtained via the LDA model.
- We propose to further improve the similarity measurement using metric learning when resolution categories are available.
- We conducted extensive experiments for our proposed algorithms on real ticket datasets, and experimental results demonstrate the effectiveness and efficiency of the proposed approaches.

The rest of the chapter is organized as follows: Section 4.2 briefly introduces the workflow of the infrastructure management of an automated service and shares our observations on three sets of monitoring tickets. In Section 4.3, we present resolution recommendation algorithms for monitoring tickets. Section 4.4 discusses some detailed implementation issues. In Section 4.5, we present experimental studies on real monitoring tickets. Finally, Section 4.6 concludes our chapter and discusses our future work.

4.2 Background

In this section, we first provide an overview of automated service infrastructure monitoring with ticket generation and resolution. Then we present our analysis on real ticket data sets.

4.2.1 Automated Services Infrastructure Monitoring and Event Tickets

The typical workflow of problem detection, determination, and resolution in services infrastructure management is prescribed by the ITIL specification [urld]. Problem detection is usually provided by monitoring software, which computes metrics for hardware and software performance at regular intervals. The metrics are then matched against acceptable thresholds. A violation induces an alert. If the violation persists beyond a specified period, the monitor emits an event. Events from the entire service infrastructure are accumulated in an enterprise console that uses rule-, case- or knowledge-based engines to analyze the monitoring events and decide whether to open an incident ticket in the ticketing system. The incident tickets created from the monitoring events are called monitoring tickets. Additional tickets are created upon customer request. The information accumulated in the ticket is used by technical support for problem determination and resolution. In this chapter, we consider tickets generated by a service management system (see Figure 4.1).

Each event is stored as a database record that consists of several related attributes with values describing the system status at the time this event was generated. For example, a CPU-related event usually contains the CPU utilization and paging utilization information. A capacity-related event usually contains the disk name and the size of disk used/free space. Typically, different types of events have different sets

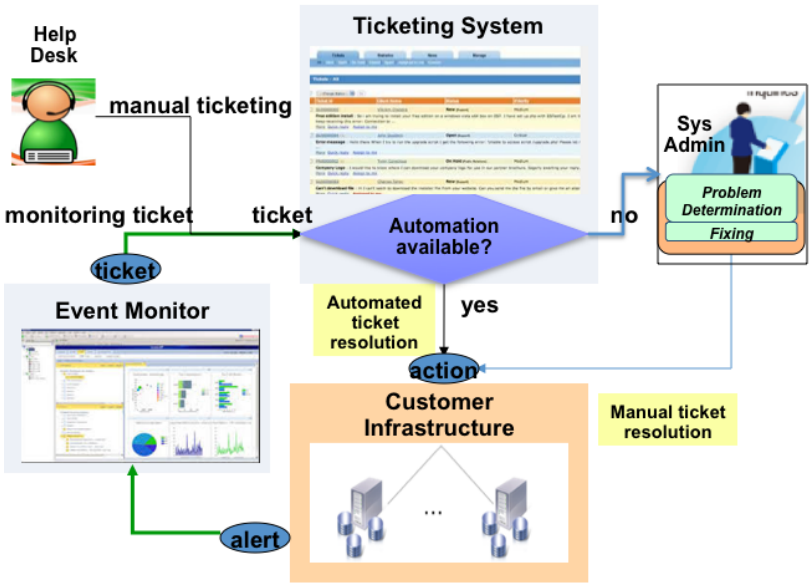

Figure 4.1: Service management system.

of related attributes. The problem resolution of every ticket is stored as a textual description of the steps taken by the system administrator to resolve this problem.

4.2.2 Repeated Resolutions of Monitoring Tickets

We analyzed ticket data from three different accounts managed by IBM Global Services. Many ticket resolutions repeatedly appear in the ticket database. For example, for a low disk capacity ticket, usual resolutions are deletion of temporal files, backup data, or addition of a new disk. Unusual resolutions are very rare.

Data set	Num. of Tickets	Time Frame
account1	31,447	1 month
account2	37,482	4 months
account2	29,057	5 months

Table 4.1: Data summary.

The collected ticket sets from the three accounts are denoted by "account1", "account2" and "account3", respectively. Table 4.1 summarizes the three data sets.

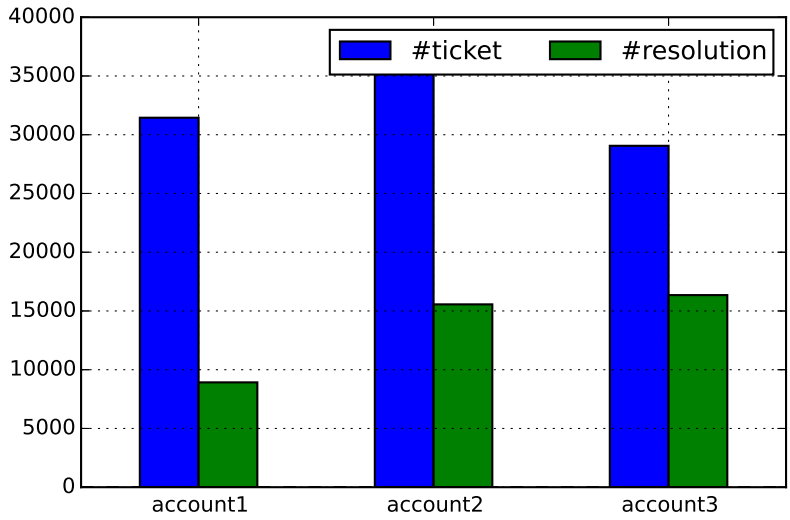

Figure 4.2: Numbers of tickets and distinct resolutions.

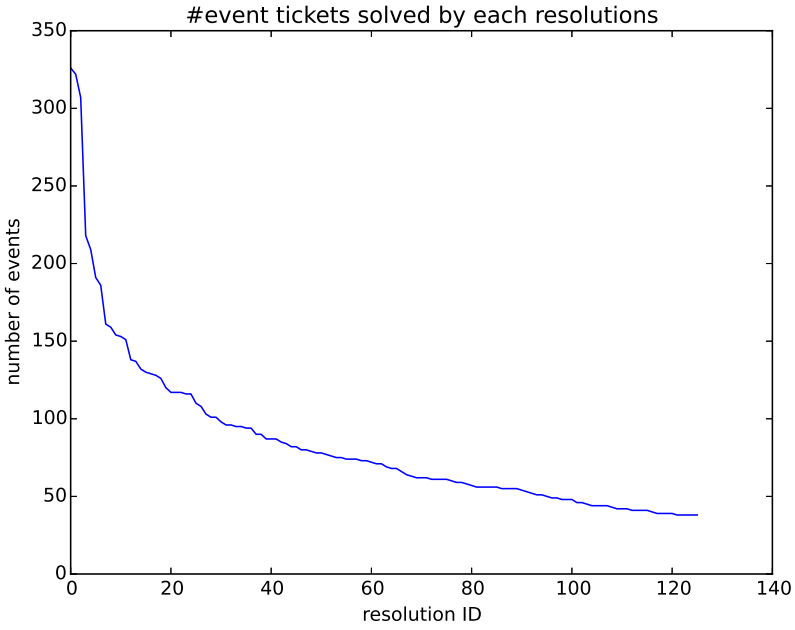

Figure 4.3: Number of monitoring tickets resolved by each resolutions denoted by "resolution ID" in account1.

Figure 4.2 shows the numbers of tickets and distinct resolutions and Figure 4.3 shows the top repeated resolutions in "account1" denoted by "resolution ID". We observe that a single resolution can resolve multiple monitoring tickets. In other words, multiple tickets share the same resolutions.

4.3 Preliminary Work

In this section, we first introduce the basic KNN-based recommendation algorithm, and then present our improved algorithms.

4.3.1 Workflow

Figure 4.4 shows the workflow of resolution recommendation. Four different algorithms are included in the workflow:

- KNN: the algorithm using attribute-level features
- LDABaselineKNN: the algorithm using topic-level features obtained via LDA
- CombinedLDAKNN: the algorithm incorporating both the event and resolution information with top-level features
- MLCombinedLDAKNN: the algorithm using the similarity measure obtained using metric learning (when resolution categories are available)

The first algorithm was used in [TLSG13] and the last three algorithms are proposed in this chapter. Figure 4.4 clearly illustrates the differences among these four recommendation methods. The details of the three proposed algorithms will be described in detail in Section 4.3.3, Section 4.3.4, and Section 4.3.5, respectively.

4.3.2 Basic KNN-based Recommendation

Given an incoming monitoring ticket, the objective of the resolution recommendation is to find k resolutions as close as possible to the true one for some user-specified parameter k. The recommendation problem is often related to that of predicting the top k possible resolutions. A straightforward approach is to apply the KNN algorithm, which searches the K nearest neighbors of the given ticket (K is a predefined

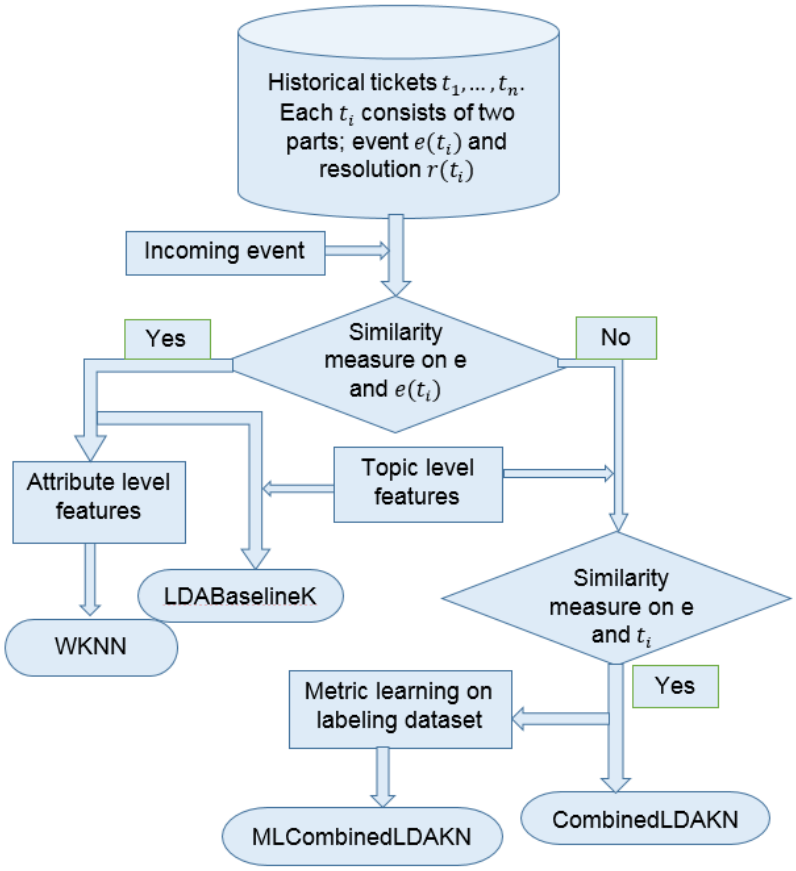

Figure 4.4: Algorithms' workflow.

parameter), and recommends the top $k \leq K$ representative resolutions among them [SKKR00, T$^+$06]. The nearest neighbors are indicated by similarities of the associated events of the tickets. In this chapter, the representativeness is measured by the number of occurrences in the K neighbors.

Table 4.2 lists the notations used in this chapter. Let $D = \{t_1, ..., t_n\}$ be the set of historical monitoring tickets and t_i be the i-th ticket in D, $i = 1, ..., n$. Given a monitoring ticket t, the nearest neighbor of t is the ticket t_i which maximizes $sim(e(t), e(t_i))$, $t_i \in D$, where $sim(\cdot, \cdot)$ is a similarity function for events. Each event consists of event attributes with values. Let $A(e)$ denote the set of attributes of event e. The similarity for events is computed as the summation of the similarities for all

Notation	Description
D	Set of historical tickets
$\|\cdot\|$	Size of a set
t_i	i-th monitoring ticket
$r(t_i)$	Resolution description of t_i
$e(t_i)$	The associated event of ticket t_i
$A(e)$	Set of attributes of event e
$sim(e_1, e_2)$	Similarity of events e_1 and e_2
$sim_a(e_1, e_2)$	Similarity of a values of event e_1 and e_2
K	Number of nearest neighbors in the KNN algorithm
k	Number of recommended resolutions for a ticket, $k \leq K$

Table 4.2: Notations.

attributes. There are three types of event attributes: categorical, numeric and textual (shown by Table 4.3).

Type	Example
Categorical	OSTYPE, NODE, ALERTKEY,...
Numeric	SERVERITY, LASTUPDATE, ...
Textual	SUMMARY,...

Table 4.3: Event attribute types.

Given an attribute a and two events e_1 and e_2, $a \in A(e_1)$ and $a \in A(e_2)$, the values of a in e_1 and e_2 are denoted by $a(e_1)$ and $a(e_2)$. The similarity of e_1 and e_2 with respect to a is

$$sim_a(e_1, e_2) = \begin{cases} I[a(e_1) = a(e_2)], & \text{if } a \text{ is categorical,} \\ \frac{|a(e_1) - a(e_2)|}{max|a(e_i) - a(e_j)|}, & \text{if } a \text{ is numeric,} \\ Jaccard(a(e_1), a(e_2)), & \text{if } a \text{ is textual,} \end{cases}$$

where $I(\cdot)$ is the indicator function returning 1 if the input condition holds, and 0 otherwise. Let $max|a(e_i) - a(e_j)|$ be the size of the value range of a. $Jaccard(\cdot, \cdot)$ is the Jaccard index for *bag of words model* [Cho10], frequently used to compute the similarity of two texts. Its value is the proportion of common words in the two texts. Note that for any type of attribute, inequality $0 \leq sim_a(e_1, e_2) \leq 1$ holds. Then, the

similarity for two events e_1 and e_2 is computed as

$$sim(e_1, e_2) = \frac{\sum_{a \in A(e_1) \cap A(e_2)} sim_a(e_1, e_2)}{|A(e_1) \cup A(e_2)|}. \tag{4.1}$$

Clearly, $0 \leq sim(e_1, e_2) \leq 1$. To identify the type of attribute a, we only need to scan all appearing values of a. If all values are composed of digits and a dot, a is numeric. If some value of a contains a sentence or phrase, then a is textual. Otherwise, a is categorical.

4.3.3 Representation of Monitoring Tickets

As shown in Section 4.3.2, attribute level features are used in the traditional KNN algorithm for recommendation. However, attribute-level feature representation is not interpretable and often contains a lot of noise.

Our observation indicates that each monitoring ticket describes the existing problems (e.g., low capacity, high CPU, utilization) in service, and the associated ticket resolution should be highly relevant to the problems. For example, Table 4.4 presents some sample monitoring tickets for "low free space" and their corresponding resolutions. The problems in these tickets are described by the "SUMMARY" attribute and they all share the similar semantic meaning "low free space". Therefore, it is better to use features semantically capturing these problems, instead of attribute-level features, to represent monitoring tickets.

In this chapter, we propose to apply Latent Dirichlet Allocation [BNJ03](LDA) to perform feature extraction, which can first extract hidden topics and then encode monitoring tickets using topic level features.

LDA is a generative probabilistic model of a document corpus. Its basic idea is that documents are represented as random mixtures over latent topics, where each

topic is characterized by a distribution over words [BNJ03]. Figure 4.5 shows the graphical model representation of LDA.

The w_{ij}'s are the only observable variables. Following [BNJ03], LDA assumes the following generative process for each document w in a corpus D of length M:

1. Choose $\theta \sim Dir(\alpha)$, where $Dir(\alpha)$ is the *Dirchlet distribution* for parameter α
2. For each of the N words w_n:
   (a) Choose a topic $z_n \sim Multinomial(\theta)$.
   (b) Choose a word w_n from $p(w_n|z_n, \beta)$, a multinomial probability conditioned on the topic z_n.

According to the graphical model, the total probability $P(D|\alpha, \beta)$ of a corpus D is given by:

$$\prod_{d=1}^{M} \int p(\theta_d|\alpha)\Big(\prod_{n=1}^{N_d} \sum_{z_{d_n}} p(z_{d_n|\theta_d}) p(w_{d_n})|z_{d_n}, \beta\Big) d\theta_d \tag{4.2}$$

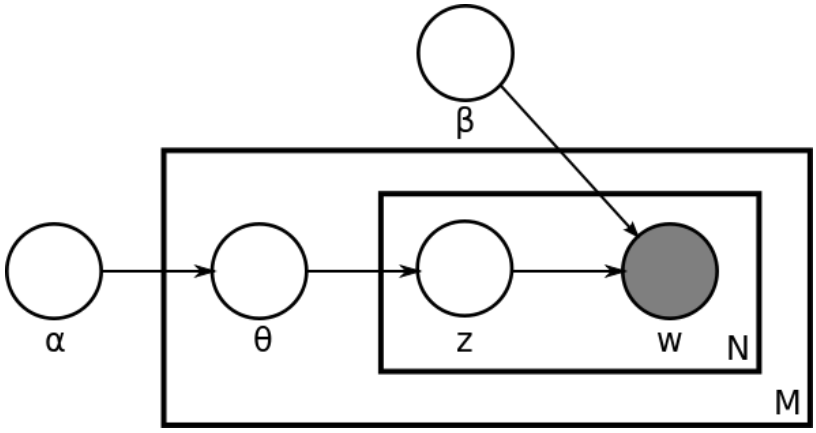

Figure 4.5: Plate notation representing the LDA model. α is the parameter of the Dirichlet prior on the per-document topic distribution; β is the parameter of the Dirichlet prior on the per-topic word distribution; θ_i is the topic distribution for document i; ϕ_k is the word distribution for topic k; z_{ij} is the topic for the j-th word in document i, and w_{ij} is the specific word.

Learning the various distribution (the set of topics, their associated word probabilities, the topic of each word, and the topic probabilities of each document) is a problem of Bayesian Inference [BNJ03]. Topic probability distribution of a document is commonly used as its feature vector.

Following are steps for using LDA for feature extraction in our work:

- Represent each monitoring ticket as a document by concatenating each attribute after stop words removal and tokenization
- Using historical tickets to train a LDA model
- Inference feature vectors using the trained LDA model for both incoming events and historical monitoring tickets.

After those steps, monitoring tickets can be encoded as feature vectors and the cosine similarity can then be applied to measure their similarities. Experiments demonstrate that the algorithm performance based on topic level features is better than that on attribute level features.

4.3.4 Incorporating the Resolution Information

In previous KNN-based recommendation approaches, resolutions are ranked according to the similarity measurement using the event information only. However, the resolutions often reveal their prevalence in historical tickets and contain important information about the events, which can be used to improve the recommendation performance. There are two practical motivations for incorporating the resolution information:

1. In a K nearest neighbor search, historical tickets with resolutions that are highly relevant to an incoming event should be ranked higher than those tickets having similar event descriptions, but with less related resolutions.

2. In a K nearest neighbor search, those tickets with resolutions that are more prevalent should be ranked higher than those with less prevalent resolution, even if their event descriptions are similar.

Table 4.4 presents four tickets having similar event descriptions (shown in the "SUMMARY" attribute) from account1. All four tickets are describing a "low free space" problem. In practice, however, the resolution from Ticket 1 should have a higher rank than the one from Ticket 4 since the resolution from Ticket 1 is more informative. Similarly, resolutions from Ticket 1 and Ticket 2 should have higher ranks than the one from Ticket 3 because of their higher prevalence.

ticketID	**SUMMARY**	**RESOLUTION**
1	The logical disk has a low amount of free space. Percent available: 2 Threshold: 5	After deleting old uninstall files, the logical disk has now over 10% of free disk space.
2	The percentage of used space in the logic disk is 90 percent. Threshold: 90 percent	After deleting old uninstall files, the logical disk has now over 15% of free disk space.
3	File system is low. The percentage of available space in the file system is 10 percent. Threshold: 90 percent	After delprof run, the server now has more than 4gb of free space
4	The logical disk has a low amount of free space. Percent available: 3 Threshold: 5	No trouble was found, situation no longer persists.

Table 4.4: Tickets for explaining motivation of incorporating resolution information.

In Section 4.3.2, $sim(e, e(t_i))$ is computed to find the K nearest neighbors of an incoming event e, in which $e(t_i)$ is the event information associated with the i-th ticket. To incorporate the resolution information, $sim(e, t_i)$ (i.e., similarity between an incoming event and the i-th ticket), rather than $sim(e, e(t_i))$, is used in the algorithm. $sim(e, t_i)$ can be easily computed since e and t_i can be vectorized with the same dimensions after using topic-level features. Experiments demonstrate the effectiveness of this proposed approach.

4.3.5 Metric Learning

In previous sections, we improve the recommendation algorithm by using topic-level features and incorporating resolution information into a K nearest neighbor search. However, we still treat each feature equally in computing the similarity measure. According to our observation, topics extracted from the LDA model should have different contributions to the similarity measurement since some topics contain the major descriptive words about events while the others may consist of less meaningful words. For example, Table 4.5 lists two topics for illustration. Apparently Topic 30 contains more descriptive words than Topic 14 and thus we should assign a larger weight to Topic 30 in the similarity measurement. We adopt metric learning [Kul12] to achieve this goal.

topicID	**SUMMARY**
14	server wsfpp1 lppza0 lppzi0 nalac application
30	server hung condition responding application apps

Table 4.5: First 6 words are extracted to represent topics trained from LDA.

The metric learning problem aims at learning a distance function tuned to a particular task, and has been shown to be useful when used in conjunction with nearest-neighbor methods and other techniques that rely on distances or similarities [FSSM07]. Mahalanobis Distance is commonly used for vectorized inputs, which can avoid the scenario in which one feature dominates in the computation of the Euclidean distance. In the metric learning literature, the term "Mahalanobis distance" is often used to denote any distance function of the following form:

$$d_A(x, y) = (x - y)^T A(x - y), \tag{4.3}$$

where A is some positive semi-definite (PSD) matrix, and x, y are the feature vectors. To facilitate the learning process, in metric learning, a slightly modified form of

distance function is commonly used, as described below [Kul12]:

$$d_A(x, y) = x^T A y. \tag{4.4}$$

In our work, we have n historical tickets $t_1, t_2, \ldots, t_n$ and n corresponding resolutions $r(t_1), r(t_2), \ldots, r(t_n)$. We consider the resolution categories as supervision for metric learning since intuitively similar resolutions solve similar issues. We precalculate matrix $R \in R^{n*n}$ in which $R_{i,j} = sim(r(t_i), r(t_j))$. Our goal is to learn a similarity function $S_A(\vec{t_i}, \vec{t_j})$ by solving following an optimization problem:

$$\begin{aligned} f(A) &= min \sum_{i=1}^{n} \sum_{j=1}^{n} ||R_{i,j} - S_A(\vec{t_i}, \vec{t_j})||^2 \\ &= min||R - SAS^T||^2, \end{aligned} \tag{4.5}$$

in which we use $S_A(\vec{t_i}, \vec{t_j}) = \vec{t_i}^T * A * \vec{t_j}$ ($\vec{t_i}$ and $\vec{t_j}$ are feature vector for ticket t_i and t_j) instead of $S_A(e(\vec{t_i}), e(\vec{t_j}))$ as we want to keep benefits of incorporating the resolution information into K nearest search. Since matrix A is constrained to be a PSD matrix, the projected gradient descent algorithm can be directly applied to solve the optimization problem in Equation 4.5. In each iteration of gradient descent, the new updated matrix A will be projected into a PSD matrix as the initial value for the next iteration. The singular value thresholding [CCS10] has been applied to project A into a PSD matrix by setting all A's negative eigenvalues to be zero.

The following is the gradient for Equation 4.5:

$$\begin{aligned} \frac{\partial f(A)}{\partial A} &= \frac{\partial((R - SAS^T)^T(R - SRS^T))}{\partial A} \\ &= 2S^T SAS^T S - 2S^T RS \end{aligned} \tag{4.6}$$

The resolution categories are usually provided by system administrators. With the available category information, the similarity between two resolutions is computed as follows:

$$sim(r(t_i), r(t_j)) = \begin{cases} 1, & \text{if } r(t_i), r(t_j) \text{ are in same category,} \\ 0, & \text{otherwise.} \end{cases}$$

4.4 implementation

In this section, we discuss several issues in implementing the resolution recommendation system.

Redundancy Removal in Recommendation

KNN-based recommendation algorithms recommend the top k representative resolutions in the K nearest tickets. However, since all of these are similar to the incoming ticket, the resolutions of the K tickets may also be similar to each other, so that there may be some redundancy in the recommended results. To avoid this, another validation step is applied. First, the K nearest tickets' resolutions are sorted according to their representativeness in descending order. Then, we go through all K resolutions and check whether or not each of them is redundant to any previously selected resolution. If it is, we skip this resolution and jump to the next one; otherwise, we add it to the selection. Since the resolutions are textual descriptions, the redundancy of two resolutions is measured by the Jaccard index, $Jaccard(\cdot, \cdot)$, introduced in Section 4.3.2. In practice, if the descriptions of two resolutions $r(t_1)$ and $r(t_2)$ have more than one half common words (i.e. $Jaccard(r(t_1), r(t_2)) > 0.5$), the two resolutions are quite likely to be the same.

Finding Nearest Neighbors

Finding the K nearest neighbors in a large collection of historical tickets is time-consuming. There are many standard indexing search methods, such as k-d Tree [Ben75], R-Tree [Gut84], VP-Tree [Yia93], cover tree [BKL06]. But the search space of our monitoring tickets is not metric and the dimensionality is high. Therefore, locality sensitive hashing [GIM$^+$99] is more practical. Another heuristic solution is the attribute clustering based method. Different system events have different system

attributes, and the clustering algorithm can easily separate all tickets into categories based on their attribute names. If two events share very few common attributes, their similarity cannot be high. Therefore, in most cases, the nearest neighbors search only needs to access these tickets in the same category.

4.5 Evaluation

4.5.1 Algorithms

We implemented four algorithms: Weighted KNN [Dud76] using attribute level feature, the Weighted KNN method using topic level feature, the method incorporating historical resolutions information and the method using improved similarity metric after applying metric learning, which are denoted by "WKNN", "LDABaselineKNN", "CombinedLDAKNN" and "MLCombinedLDAKNN" respectively. Those algorithms, "WKNN", "LDABaselineKNN", "CombinedLDAKNN" and "MLCombinedLDAKNN", are all based on the weighted KNN algorithm framework. We still show experimental results between "WKNN" and "LDABaselineKNN" since they prove that topic level features do not cause information loss compared to attribute level features. The "LDABaselineKNN" algorithm is the baseline for "CombinedLDAKNN", which itself is the baseline for "MLCombinedLDAKNN". We use the Weighted KNN algorithm as the underlying algorithm because it is the most widely used Top-N item-based recommendation algorithm.

4.5.2 Experimental Data

Experimental monitoring tickets are collected from three accounts managed by IBM Global Services, denoted later "account1", "account2" and "account3". The mon-

itoring events are captured by IBM Tivoli Monitoring [urlb]. The ticket sets are summarized in Table 4.1. To evaluate metric learning, 1000 labeled tickets with resolution categories are obtained from "account1". Table 4.6 shows three sample categories of resolutions [BGL+14].

resolution class	**resolved event key words**
Server Unavailable	Server unavailable due to unexpected shutdown, reboot, defect hardware, system hanging
Disk/FS Capacity shortage	Disk or file system capacity problems and disk failure
Performance inefficiency	Performance and capacity problems of CPU or memory

Table 4.6: Three resolution types with the event description they resolved.

4.5.3 Evaluation Metric

The following evaluation measures are used in our experiments.

Average Similarity

In general, several resolutions can be recommended for a single testing instance. To consider the relativeness of all recommended resolutions, the *average similarity* (avgSim) is used as one evaluation metric which is given by the following equation:

$$\text{avgSim} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{n_i} sim(r_{io}, r_j)/n_i,$$

in which N is the number of testing instances, and n_i is the number of recommended resolutions for testing instance i and r_{io} is its original resolution, and r_j is its jth recommended resolution. Jaccard Similarity is used to calculate $sim(r_{io}, r_j)$.

Mean Average Precision

Mean Average Precision (MAP) [Zhu04] is widely used for recommendation evaluation. It considers not only the relativeness of all recommended results, but also the ranks of the recommended results.

$$\text{MAP@n} = \sum_{i=1}^{N} ap@n_i/N,$$

N is the number of a testing instance, $ap@n$ is given by the following equation:

$$\text{ap@n} = \sum_{k=1}^{n} p(k)\delta r(k),$$

where k is the rank in the sequence of retrieved resolutions, n is the number of retrieved resolutions, $p(k)$ is the precision at cut-off k in the list, and $\delta r(k)$ is the change in recall from items $k-1$ to k.

4.5.4 Choosing the Number of Topics

Figure 4.6 shows the experimental results of choosing the proper number of topics for training the LDA model using data set "account1". The results show that $numTopics = 300$ is a proper setup for the number of topics. Thus, we choose $numTopics = 300$ for all the following experiments.

4.5.5 Overall Recommendation Performance

The *average similarity* is used for comparing the performance among "WKNN", "LDABaselineKNN" and "CombinedLDAKNN". When resolution categories are available, *MAP@n* is used for comparing the performance between "CombinedLDAKNN" and "MLCombinedLDAKNN" since it explicitly considers the relativeness of the recommended results.

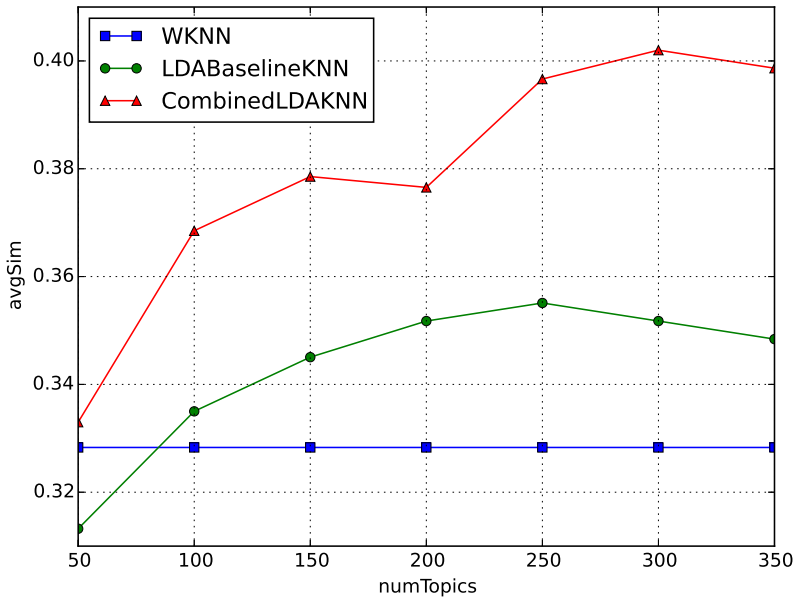

Figure 4.6: Accuracy varies for different *numTopics* for dataset "account1".

To compare the results of each algorithm, we vary the number of recommended resolutions, k. Figures 4.74.12 show the *average similarity* scores by setting $k = 1, 3, 5, 7$ separately, with $K = 8$ and $K = 16$, for data from three accounts. As shown by Figure4.74.12, topic level features are better than attribute level features for account1 and account2 and slightly worse for account3 by comparing algorithm "WKNN" and "LDABaselineKNN". "CombinedLDAKNN" always outperforms "LDABaselineKNN", which proves the effectiveness of incorporating the resolution information into K nearest neighbor search.

Metric Learning Performance

Figures 4.134.16 and Figure 4.17 are used to illustrate the usefulness of metric learning. In these figures, X-axis and Y-axis are the event id's ordered by the resolution categories, and the color indicates the similarity score. As shown in Figures 4.13, 4.13 and Figures 4.15, 4.16 similarity scores between monitoring tickets with resolutions from the same category will be enhanced while similarity scores between monitoring

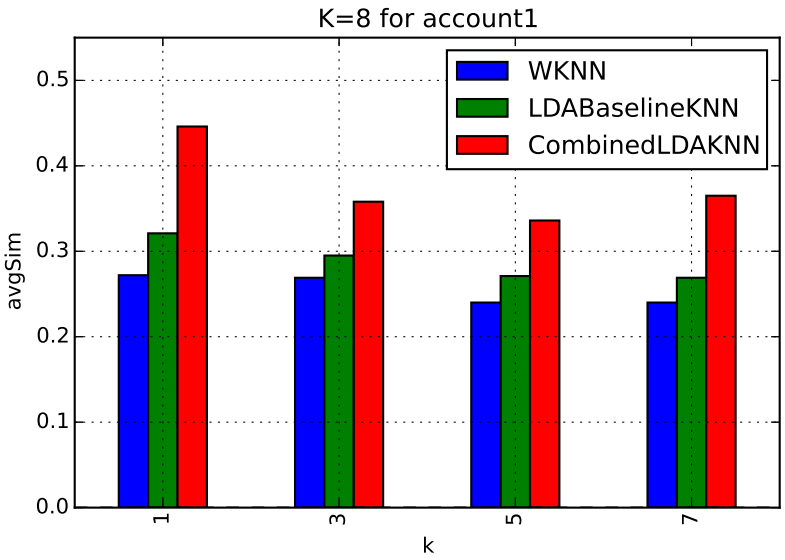

Figure 4.7: Test results for account1 by by varying k for $K = 8$.

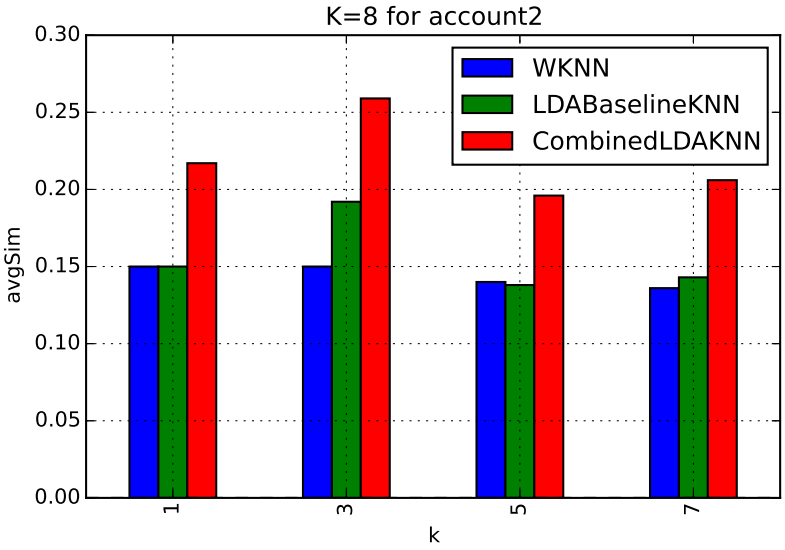

Figure 4.8: Test results for account2 by varying k for $K = 8$.

tickets with resolutions from different categories will be reduced. Therefore, for example, for a testing instance whose original resolution belongs to category i, more resolutions from category i will be retrieved first after applying metric learning.

Figure 4.17 uses *MAP* to evaluate the performance of "CombinedLDAKNN" and "MLCombinedLDAKNN". As shown in Figure 4.17, overall MAP scores of "ML-

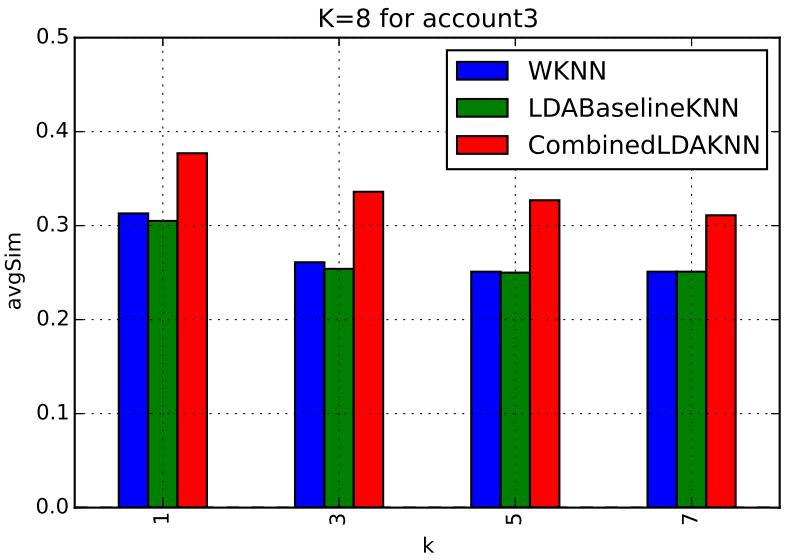

Figure 4.9: Test results for account3 by varying k for $K = 8$.

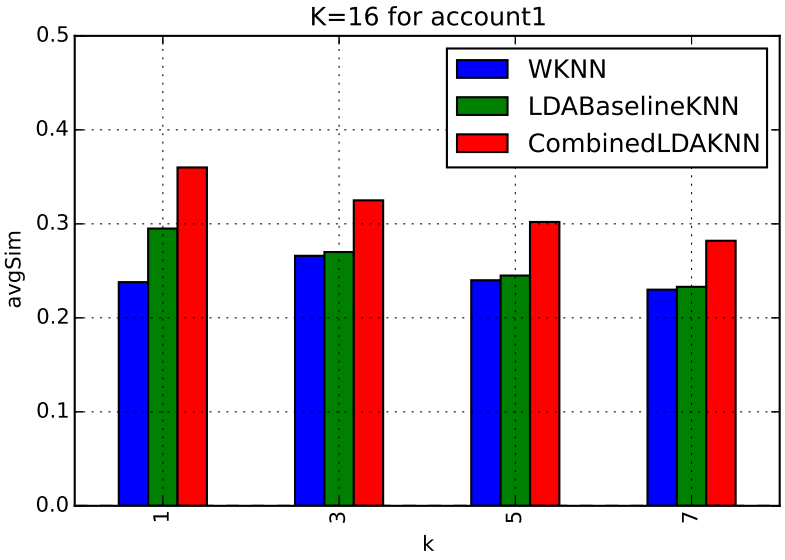

Figure 4.10: Test results for account1 by varying k for $K = 16$.

CombinedLDAKNN" are higher and more stable than "CombinedLDAKNN" when K increases. It indicates that "MLCombinedLDAKNN" can retrieve more related resolutions first and thus is more robust to noisy resolutions compared to "CombinedLDAKNN", which proves the effectiveness of metric learning.

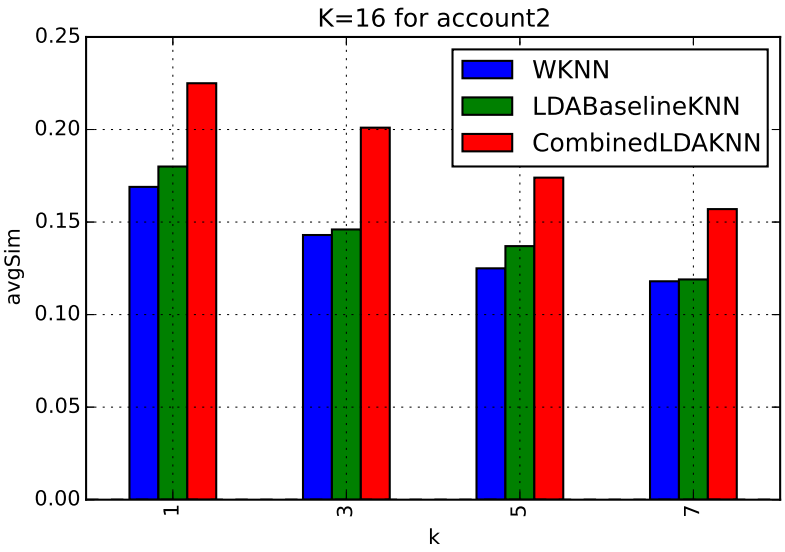

Figure 4.11: Test results for account2 by varying k for $K = 16$.

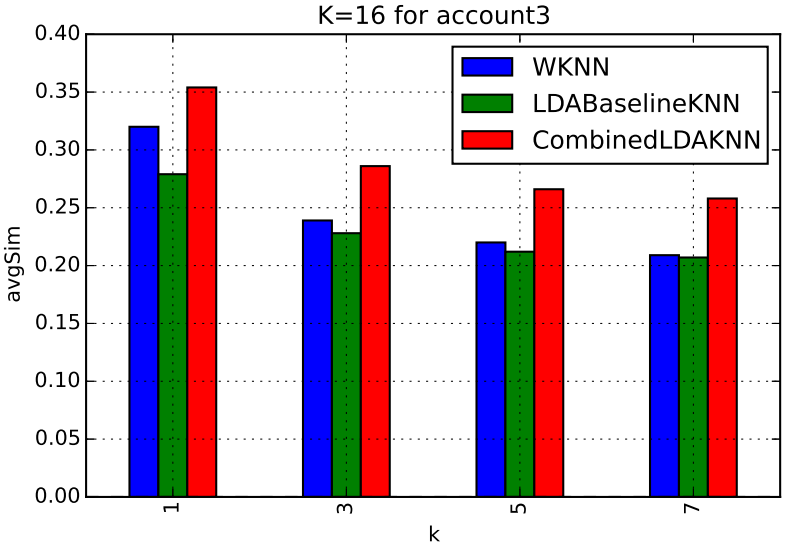

Figure 4.12: Test results for account3 by varying k for $K = 16$.

4.6 Summary

This chapter studies the problem of resolution recommendation for monitoring tickets in an automated service management. We analyze three sets of monitoring tickets collected from a production service infrastructure and identify a vast number of re-

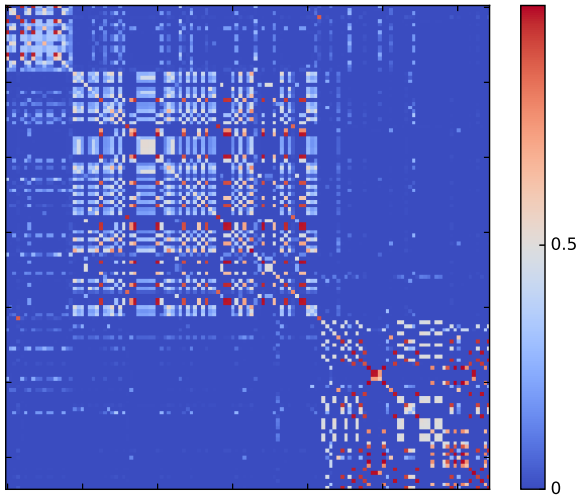

Figure 4.13: Similarity measure before metric learning for training set.

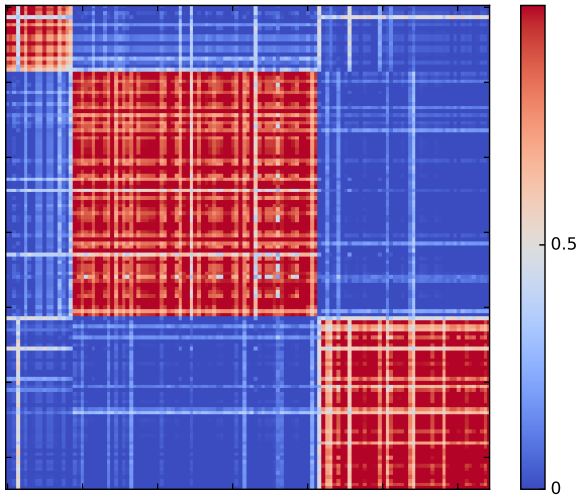

Figure 4.14: Similarity measure after metric learning for training set.

peated resolutions for monitoring tickets. Based on our prior work of KNN-based recommendation, we improve the similarity measure by utilizing both the event and resolution information from historical tickets via a topic-level feature extraction using

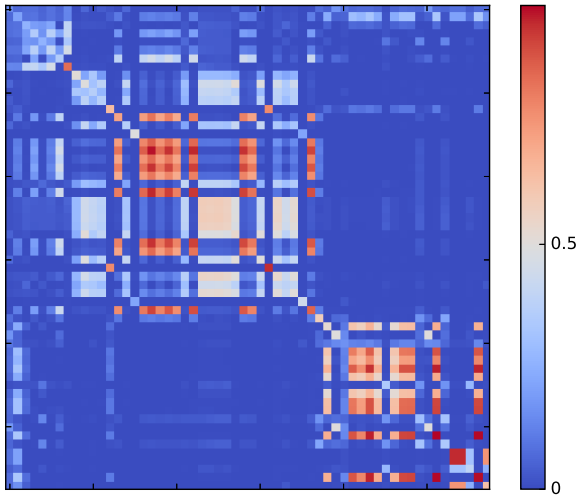

Figure 4.15: Similarity measure before metric learning for testing set.

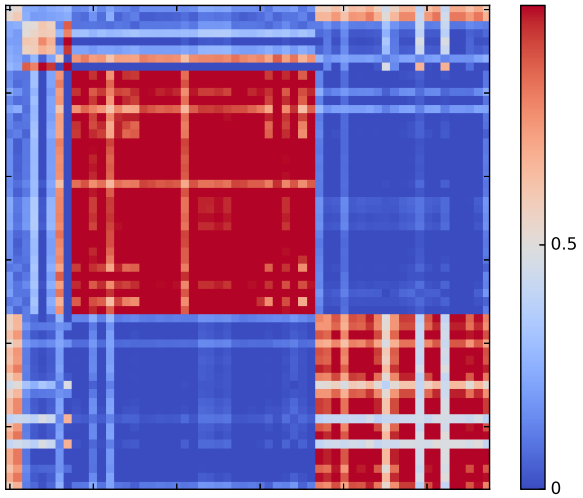

Figure 4.16: Similarity measure after metric learning for testing set.

the LDA (Latent Dirichlet Allocation) model. In addition, a more effective similarity measure is learned using metric learning when resolution categories are available.

There are several avenues for future research. First, we plan to investigate and develop intelligent classification techniques to automatically label resolutions [ZLSG14a,

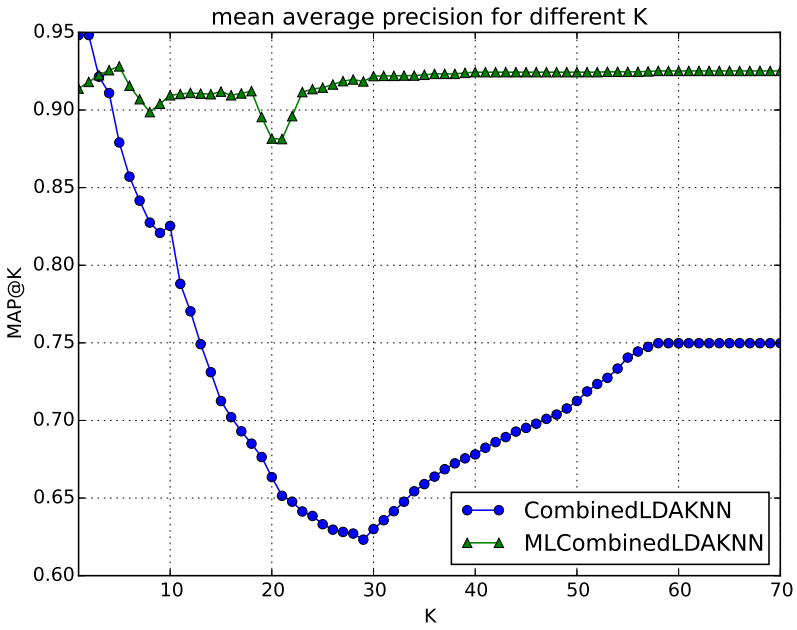

Figure 4.17: Mean average precision (MAP) varying parameter K of underlying KNN algorithm.

CQT+13]. Second, our current recommendation system uses KNN-based algorithms due to their simplicity and efficiency. We will investigate and develop other advanced algorithms to improve the recommendation performance. Finally, we also plan to use an active query strategy to fully automate resolution recommendations.

CHAPTER 5

DOMAIN ADAPTATION FOR TEXTUAL FEATURES

In recent years, IT Service Providers have been rapidly introducing automation to their service delivery model. Driven by market pressure to reduce cost and maintain quality of services, they are looking for technologies that will allow rapid progress towards attainment of truly automated service delivery. Software monitoring systems are designed to actively collect and signal event occurrences and, when necessary, automatically generate incident tickets. Repeating events generate similar tickets, which in turn have a vast number of repeated problem resolutions likely to be found in earlier tickets.

In this chapter, we develop techniques to recommend an appropriate resolution for incoming events by making use of similarities between the events and historical resolutions of similar events. The traditional KNN (K Nearest Neighbor) algorithm has been first applied to recommend resolutions for incoming tickets. Massive heterogeneous applications as well as various monitoring software are running on clients' servers to accomplish required tasks and to monitor system health via different metrics. It leads to generation of correlated tickets that have different symptom descriptions but similar resolutions. Furthermore, change of servers' environments can also induce similar situations in which ticket descriptions differ before and after change but could have similar resolutions. These correlated tickets cause performance degradation in ticket resolution recommendation. Therefore, we propose using SCL (structural corresponding learning) based feature adaptation to uncover feature mapping in different time intervals. Moreover, to put more insights into the periodic regularities existing in our ticket datasets, we apply our algorithm on tickets grouped by different time interval granularities. Extensive empirical evaluations on real-world ticket data sets demonstrate the effectiveness and efficiency of our proposed methods.

5.1 Introduction

The competitive business climate, as well as the complexity of service environments, dictate the need for efficient and cost-effective service delivery and support. These are largely achieved through service-providing facilities integrated with system management tools in combination with automation of routine maintenance procedures such as problem detection, determination and resolution for the service infrastructure [MSGL09, TLP+12, ABD+07, WE11, YPZ10]. Automatic problem detection is typically realized by system monitoring software, such as IBM Tivoli Monitoring [urlc] and HP OpenView [urla]. Monitoring continuously captures the events and generates incident tickets when alerts are raised. Deployment of monitoring solutions is a first step towards fully automated delivery of a service. Automated problem resolution, however, is a hard problem.

However, most service providers keep years' worth of historical tickets with their resolutions. The resolution is usually collected as a free-form text and describes steps taken to remediate the issue described in the ticket. We analyzed historical monitoring tickets collected from three different accounts managed by one of the large service providers (an account is an aggregate of services that uses common infrastructure). We noticed that there are many repeating resolutions for monitoring tickets within an account. It is natural to expect that if events are similar, then their respective tickets probably have the same resolution. Therefore, we can recommend a resolution for an incoming ticket based on the event information and historical tickets.

In our previous work [TLSG13], a KNN-based approach has been first applied to provide resolution recommendations for incoming tickets in service management. Additionally, several improved approaches [TLSG13] have been proposed to resolve various shortcomings of the basic algorithm and thus to make recommended resolu-

tions more relevant and practical. However, a further drawback has been uncovered when our previous methods were applied to system management.

In current service environments, massive heterogeneous applications, as well as various monitoring software, running on customers' servers to accomplish complex tasks and to monitor system health via different metrics, lead to generation of correlated tickets that have different symptom descriptions but similar resolutions. Furthermore, evolving over time, service environments cause a further discrepancy. The description of tickets generated before and after change differ but might have similar resolutions since root causes remain unchanged.

Based on our previous understanding and initial experiments, we find out that vocabularies used in ticket descriptions are changing and shifting over time but interesting mappings exist in those different vocabularies. However, our previous algorithms are not able to discover those mappings and thus their performance degrades over time due to inaccurate ticket similarity. To overcome drawback, we propose structural corresponding learning (SCL) to discover the words' mapping and apply it to our ticket resolution recommendation system.

The traditional KNN-based recommendation methodology was first proposed in our preliminary work [TLSG13]. The details and extended algorithms are fully discussed there. The rest of this chapter is organized as follows: Section 5.2 briefly introduces the workflow of the infrastructure management of an automated service and shares our observations on real-world monitoring tickets. In Section 5.3, we present SCL details and how they are used for finding feature mapping. Notice that phrases "feature mapping" and "feature adaptation" are used interchangeably in the rest of our work. Section 5.4 discusses detailed implementation and application of SCL within resolution recommendation algorithms for monitoring tickets. In Section

5.5, we present experimental studies on real monitoring tickets. Finally, Section 5.6 concludes the paper and discusses our future work.

5.2 Background

In this section, we first provide an overview of automated service infrastructure monitoring with ticket generation and resolution. Then we present our analysis on real ticket data sets.

5.2.1 Automated Services Infrastructure Monitoring and Event Tickets

The typical workflow of problem detection, determination, and resolution in services infrastructure management is prescribed by the ITIL specification [urld]. Problem detection is usually provided by monitoring software, which computes metrics for hardware and software performance at regular intervals. The metrics are then matched against acceptable thresholds. A violation induces an alert. If the violation persists beyond a specified period, the monitor emits an event. Events from the entire service infrastructure are accumulated in an enterprise console that uses rule-, case- or knowledge-based engines to analyze the monitoring events and decide whether to open an incident ticket in the ticketing system. The incident tickets created from the monitoring events are called monitoring tickets. Additional tickets are created upon customer request through Service Management System. The information accumulated in the ticket is used by technical support for problem determination and resolution. In this chapter, we consider tickets generated by a monitoring system (see Figure 5.1).

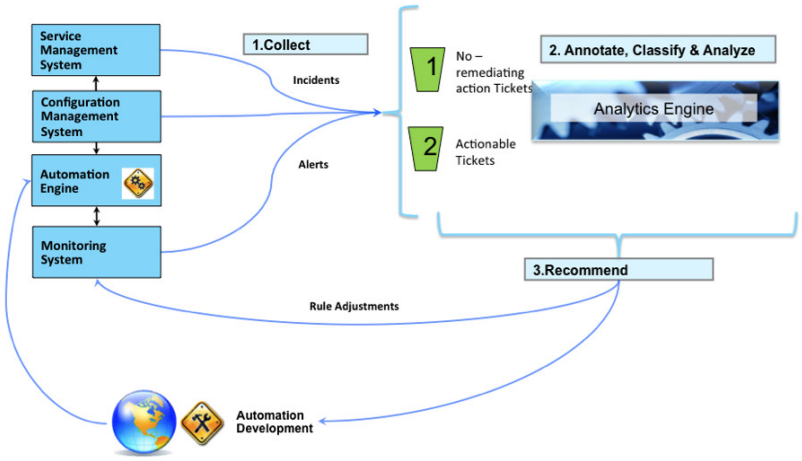

Figure 5.1: Service management system.

Each monitoring ticket is stored as a database record that consists of several related attributes with values describing the system status at the time when monitoring event was generated. For example, a CPU-related ticket usually contains the CPU utilization and paging utilization information. A capacity-related ticket usually contains the disk name and the size of disk used/free space. Typically, different types of monitoring events have different sets of related attributes. The resolution of every ticket is stored as a textual description of steps taken by the system administrator to resolve this problem.

5.2.2 Repeated Resolution of Monitoring Tickets

We analyzed ticket data from three different accounts managed by IBM Global Services. Many ticket resolutions repeatedly appear in the ticket database. For example, for a low disk capacity ticket, usual resolutions mean deletion of temporal files, backup data, or addition of a new disk. Unusual resolutions are very rare.

Collected ticket sets from the three accounts are denoted by "account1," "account2" and "account3," respectively. Table 5.1 summarizes the three data sets. Figure 5.2 shows the numbers of tickets and distinct resolutions. We observe that

Data set	Num. of Tickets	Time Frame
account1	31,447	1 month
account2	37,482	4 months
account3	29,057	5 months

Table 5.1: Data summary.

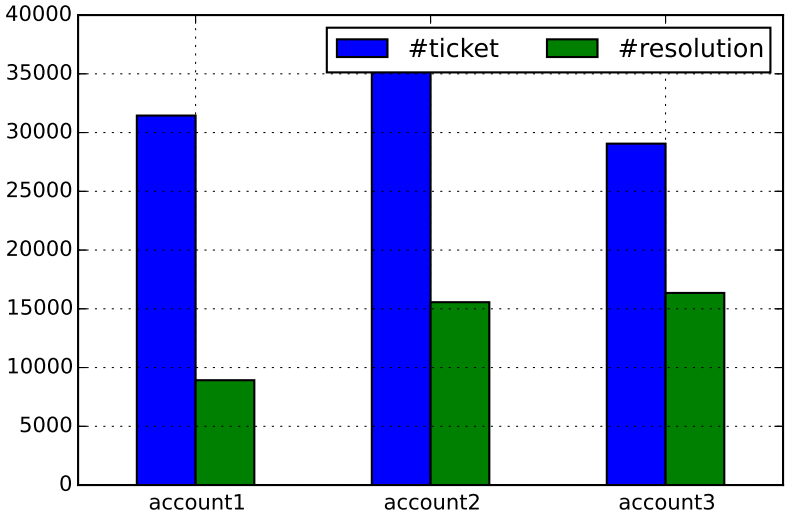

Figure 5.2: Numbers of tickets and distinct resolutions.

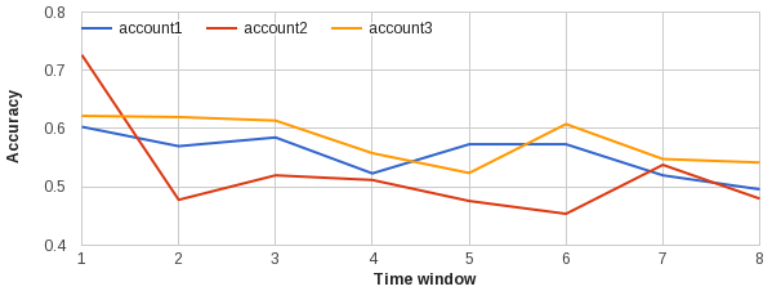

Figure 5.3: Recommendation performance degrading as testing instances coming from different sliding window.

a single resolution can resolve multiple monitoring tickets. In other words, multiple tickets share the same resolutions.

5.3 Feature Adaptation

In this chapter, we propose a solution for ticket resolution recommendation that accommodates vocabularies changing or shifting with time. We evaluate our solution on three real world ticket datasets collected from IBM service management system.

First, we show that feature variation and shift exists and degrades performance of ticket resolution recommendation. Second, we apply a structural correspondence learning (SCL) domain adaptation algorithm [BMP06] for use in ticket resolution recommendation to solve the aforementioned issue. We assume that although features shift with time, there exists some feature mapping, i.e., some correspondence of features for tickets generated in different time intervals. We order and group tickets based on consecutive and disjointed time windows, and consider each time window as one domain. Thus, a feature mapping problem in ticket resolution recommendation become a domain or feature adaptation problem.

In the following section we will briefly review SCL, introduce our new pivot selection strategy and describe how we apply SCL to ticket resolution recommendation problems.

5.3.1 Structural Corresponding Learning

First, we consider a simple example to illustrate application of SCL. Suppose that we have a dataset of historical tickets based on which we need to identify an appropriate resolution for the incoming event. Resolutions for the same root cause could slightly differ, but descriptions of symptoms could vary significantly. For example, two tickets have the same resolution as "archive the logs and thus reduce the space utilization", but their descriptions could be different when diverse vocabularies are used, such as "volume", "capacity" or "harddiskvolume".

Our key intuition is that even if "volume", "capacity" and "harddiskvolume" are literally distinct, they have high correlation with "space" or "utilization" in the historical data set, and thus we can tentatively construct some mapping between them and recommend similar resolutions for incoming events represented by different vocabulary.

5.3.2 Algorithm Overview

Given two consecutive sliding time windows, source tickets are defined as the tickets from the first time window; target domain tickets are the tickets from the second time window. The SCL first chooses a set of m pivot features that occur frequently in both domains. Next, it models the correlation between the pivot features and all other features by training pivot predictors to predict the occurrences of each pivot feature in all training dataset from both domains [AZ05, BMP06]. The coefficients of l-th pivot predictor characterize the correlation between non-pivot features with the l-th pivot feature; positive coefficients indicate that a non-pivot features is highly correlated with the corresponding pivot feature.

We consider the coefficients of each pivot predictor as a column vector. All predictors can then be arranged into a matrix $W = [w_l]_{l=1}^{n}$, where w_l is the lth column coefficient vector and n is the number of pivot features. Let $\theta \in \mathbb{R}^{h \times d}$ be the top h left singular vectors of W, i.e., $[U\ D\ V^T] = SVD(W)$, $\theta = U^T[1:h, :]$. There vectors are the principal predictors for our coefficient space. If these pivot features are well chosen, we expect these principal predictors to distinguish between words leading to similar and different resolutions in both domains.

As we observe a feature vector $x \in S$ at training and testing time, we notice feature space S is different for different domains. We apply the projection θx to obtain k new real-valued features. Now we use augmented feature vector $< x, \theta x >$

for the same instance. If θ contains meaningful correspondences, then we have a mapping of feature vectors from different feature spaces into a shared feature space. The shared features will bring tickets predicting similar pivot features closer using similarity measurement given in following Equation 5.1:

$$sim(xa_s, xa_t) = \frac{\sum_{w \in V} x_s(w) * x_t(w)}{2|x_s| \cdot |x_t|} + \frac{cos(\theta x_s, \theta x_t)}{2} \tag{5.1}$$

Where $xa =< x, \theta x >$ is the augmented feature vector, V is the shared words in two feature space that x_s and x_t belongs to, $x(w)$ is the entry value for word w in vector x and $cos(\cdot, \cdot)$ is the cosine similarity function. Here we assume those pivot features strongly indicate the resolution and we will explain how we extract those pivot features in the next section.

5.3.3 Picking Pivot Features

The efficacy of SCL depends on the choice of pivot features. In [BMP06], frequently-occurring words are chosen as pivot features to resolve domain adaptation in a speech tagging problem. Frequently-occurring words often correspond to function words, such as prepositions and determiners, and are good indicators of parts of speech. With respect to sentiment classification in [BDP+07], those features are chosen as pivot features which have the highest mutual information to the source label. We use a different approach, however, for a ticket resolution recommendation task in picking up pivot features. We require pivot features to be good predictors of resolutions. We attempt following two approaches in our situations.

First, we calculate the term frequency-inverse document frequency (TF-IDF [MRS+08]) scores for all words out of *ticket symptom description* in both domains and choose 1000 words having the highest TF-IDF scores for each domain. Then, we choose the m most frequently-occurring words out of the two sets of 1000 words.This allows us

to eliminate function words, such as prepositions and determiners, while choosing less frequently-occurring words that are strong indicators of resolutions. However, vocabularies used for describing "ticket symptom" and "ticket resolution" could differ, and the first approach only gives us the words that are strong indicators of "symptom" instead of "resolution". Second, we assume that there are some tickets with resolution in target domain and use the strategy shown in Table 5.3 to pick the pivot features from *ticket resolutions* instead of ticket symptoms. The assumption can be easily satisfied in a practical scenario since we define source and target domain via partitioning the tickets and, therefore, we can always assign those latest tickets with resolutions to a target domain. Table 5.2 contains the top pivot features chosen using these two approaches.

TF_1	**TF_3**
app space job high restore status error procedures failed db	incident close copy resolve server found issue action team job clear close file

Table 5.2: Top pivot features chosen by TF_1, TF_3.

We refer throughout the rest of this work to the algorithm using the first approach of picking pivot features as TF_1, and the algorithm using the second one as TF_3. As shown in Table 5.2, the pivot features chosen by TF_1 strongly describe the ticket symptom observed on the server system. At the same time, the pivot features chosen by TF_3 describe the ticket resolution, i.e., how to resolve issues on the server system.

5.4 Implementation

In this section, we discuss implementation and some issues we encountered while building the resolution recommendation system based on feature adaptation.

5.4.1 Pivot Predictors

From each pivot feature we create a binary classification problem of the form "does pivot feature l occur in this ticket?". Then we classify the training set. If we represent our features as a vector x, we can solve these problems using m linear predictors in which we use a linear regression model with l_2 regularization as the underlying classification model.

$$f_l(x) = sgn(\hat{w}_l \cdot x), l = 1 \ldots m \tag{5.2}$$

Notice that these predictors operate on the original feature space and that there are several differences for constructing predictor formulas indicated by Equation 5.2. Table 5.3 summarizes the differences.

	TF_1	**TF_3**
$sgn(\cdot)$	does pivot feature l occurs in the symptom description of this ticket?	does pivot feature l occurs in the resolution of this ticket?
pivot features	the m most frequently-occurring words shared in the two sets of 1000 words having the top TF-IDF scores in ticket symptom descriptions of both domains	the m most frequently-occurring words shared in the two sets of 1000 words having the top TF-IDF scores in ticket resolutions of both domains
x	1000 words having the highest TF-IDF scores from symptom description but excluding pivot features	1000 words having the highest TF-IDF scores from symptom description
training data	all tickets from both domains	all tickets attached with resolution from both domains

Table 5.3: Differences in constructing predictors for TF_1 and TF_3.

For TF_1, each instance contains features that are totally predictive of the pivot features (the feature itself), so we exclude those features when making the binary prediction as shown in Table 5.3. The pivot predictors are the key element in SCL. The weight vectors $\hat{w}_l$ encode the covariance of the non-pivot features with the pivot

features. If the weight given to the z-th feature by the l-th pivot predictor is positive, then feature z is positively correlated with pivot feature l. Since pivot features strongly indicate resolutions, we expect non-pivot features from both domains to be correlated with them. If two non-pivot features are correlated in the same way with many of the same pivot features, then they have a high degree of correspondence.

5.4.2 Hyper Parameter Tuning

Structural corresponding learning uses the techniques of alternating structural optimization (ASO) to learn correlations among pivot and non-pivot features [AZ05]. There are several free parameters and extensions to ASO, and we briefly address our choices here. As shown in Figure 5.4, setting h between 20 and 40 does not result in large change in performance. With respect to the number of pivot features m, we

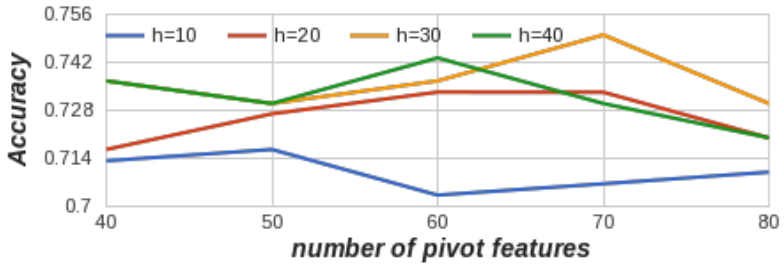

Figure 5.4: Hyper parameter selection. According to experimental results, $h = 30$ and $m = 70$ are chosen for the rest of our work.

observe that m around 65 gives a good performance. Therefore, through the rest of this work, we set $h = 30$ and $m = 70$.

5.5 Evaluation

In this section, we will focus on the dataset, the running environment and discussion of experimental results.

5.5.1 Setup

For each account, we ordered tickets by time and chose various approaches to slide the dataset. Codes are implemented in Java, running on 64-bit Windows 7 operating system residing on a machine equipped with 16 GB RAM, Intel(R) Core(TM) i7-4770 CPU running at 3.40 GHz. Training of pivot predictors is parallelized.

5.5.2 Evaluation of Feature Adaptation

The goal of our first experiment is to verify feature adaptation in which tickets are ordered by time. The first 6000 tickets are chosen for experiments. Then a half of these tickets are considered as source domain and the second half as target domain. We assume the first 1000 tickets in target domain have resolutions. Therefore, pivot features could be extracted from available resolutions of both source and target domain using the approach shown in Table 5.3 . Under this setup, our experiments show that feature mapping between the vocabularies used in both source domain and target domain exists, and it strongly helps in improving ticket recommendation performance.

Figure 5.5 shows the overall performances using three algorithms "No-TF", "TF-1" and "TF-3". "No-TF" is just the basic KNN-based recommendation algorithm with no feature adaptation.

As shown in Figure 5.5, "TF-3" always outperforms the other two algorithms by around 8%, and these two algorithms have similar performances but, "TF-1" never performs better than "No-TF." As we illustrated in Table 5.2, the approach of picking pivot features in "TF-1" provides a strong indication of "symptom description" rather than "resolution description," and therefore is inaccurate and misleading in building correlation between non-pivot features and pivot features.

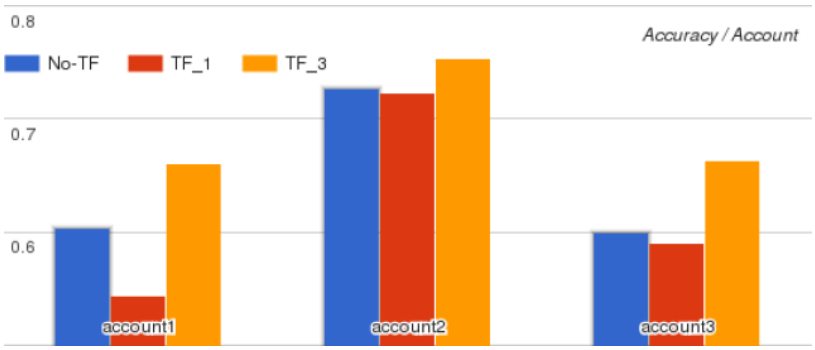

Figure 5.5: Overall performance for three accounts.

We select an event ticket in account1 to illustrate why "TF-3" is better than the basic KNN-based algorithm "No-TF." Table 5.4 shows a list of recommended resolutions given by each algorithm. The testing ticket is a real event ticket triggered by an error when processing a text file. It has the symptom description as "an error in process xxx while processing file xxx.txt, leave the processing" and its true resolution is "connectivity issue, the file has been retransmitted successfully." The general idea of this resolution is to retransfer and reprocess the file.

As shown in Table 5.4, the second resolution recommended by "No-TF" is the most relevant but still a wrong resolution: "as a part of application team testing, file has been successfully repulled." Obviously, it is caused by manual testing. "TF-3," however, recommends 4 resolutions all highly relevant to the true resolution. According to the definition of *HIT* [TLSG13], the first resolution recommended by "TF-3" is regarded as a true resolution, i.e., "the file has been retransmitted successfully." Also as we notice, the word "file" is a pivot feature shown in Table 5.3 that allow us to identify expected resolutions. Therefore, our proposed algorithm "TF-3" indeed can find the hidden feature mapping using SCL and the feature mapping performs better for recommending ticket resolution.

Moreover, we visulalized one row of the project matrix θ for our experiments on general feature adaptation. Table 5.5 illustrates the first row of θ; the features on each

algorithm	recommended resolutions	isHit
No-TF	closing this ticket as its a duplicate of the incident inc0771310	FALSE
	as a part of application team testing, file has been successfully repulled	FALSE
	the file is been decomissioned, no action requried, hence resolving	FALSE
TF-3	the file has been retransmitted successfully	TRUE
	the file was delivered successfully	FALSE
	the file has been successfully repulled	FALSE
	file was pulled successfully from bank of xxx	FALSE

Table 5.4: Case study.

row appear only in the corresponding domain. In a traditional binary classification problem of applying SCL [BDP+07], corresponding features indicate either a positive or negative label. Corresponding features here indicate event tickets caused by the same or similar root cause and thus share similar resolutions. We colored those correpondence feature groups so they could be easily identified visually. For example, features "*sdump, page, harddiskvolume, paging, traps*" colored in red indicate system issues in or similar to paging due to low capacity. "sdump" is an excutable command that tries to dump virtual storage and thus makes space for paging. Without feature mapping, event tickets will be considered having low or no similarity if they contain discriminant features. Once we disovered feature mapping, we can project those discriminant features to shared feature space by applying them to θx. The features will ensure that their corresponding event tickets have higher similarity.

5.5.3 Feature Adaptation for Different Time Granularity

In section 5.5.2, we discussed the experiments for general feature adaptation problems. In this section we consider an adaptability between ticket data sets sliced by different time granularities, i.e., daily, weekly or even monthly. As an example, we choose

type	features
+s	sdump bee idc ami sr included refer read processing queues page read-response
+t	readresponse getacctsummbycustid contingency cli logerror hard-diskvolume paging traps getacctsforgrantee ant
-s	messages wtprocess normal wiptrigger ifscmonitor poa responding dumpcode acctinfo
-t	batjbstrm sm fndstn xmx dbm aelv throw responsestream

Table 5.5: Correspondences discovered by SCL for general feature mapping experiments. Notation "s" corresponds to features coming from source domain, and "t" corresponds to features coming from target domain. The "+" and the "-" symbols indicate positive and negative features in correspondences, respectively.

ticket sets from three consecutive days and use them for the experimental setup for three algorithms listed in Table 5.6.

algorithm	day1	day2	day3
No-TF	training dataset	N/A	testing dataset
TF-1	source tickets	target tickets for training	target tickets for testing
TF-3	source tickets	target tickets for training	testing tickets for testing

Table 5.6: Experimental setup for feature adaptation using daily ticket set.

The goal of this experiment is to understand the feature shifting phenomenon and the shifting of an event type in different time granuarities. Positive results would indicate that monitoring tickets generated daily do not change too much and there still exist stable pivot features for constructing meaningful feature mapping in the feature vocabulary. Otherwise, it would indicate that the daily monitoring tickets shift a lot leading to various daily resolutions and thus noisy pivot features.

Figures 5.6 show the daily ticket number generated for the three different accounts. Here, we remove tickets that are scattered in days with very few tickets, and thus we see fewer tickets compared to Table 5.1. We choose account1 to carry out our daily adaptation experiments since the number of daily tickets in account1 is sufficiently

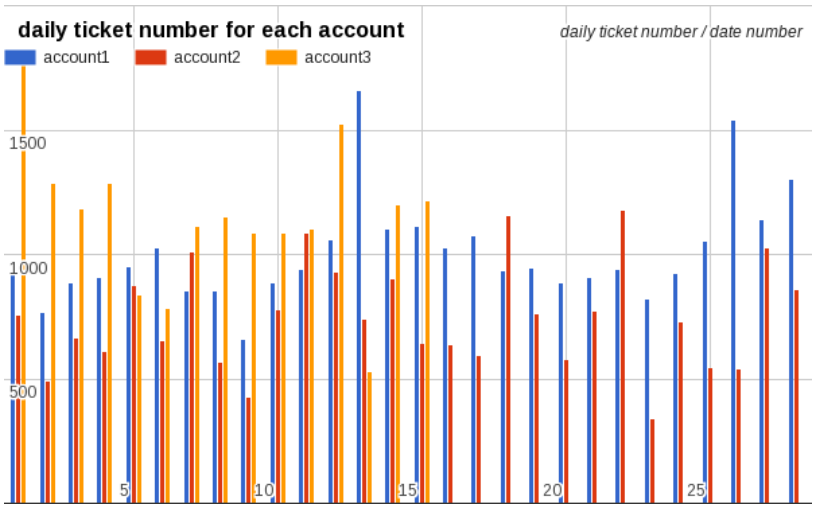

Figure 5.6: Daily ticket number for each account. X-axis indicates the date IDs in which the same ID doesn't necessarily corresponde to the same date for different accounts. Around two weeks data is available in account3 and four weeks' data in account1 and account2 in which we require a sufficient number of tickets generated daily for our experiments.

large and varies the least. Figure 5.7 shows the experimental results for daily feature adaptation. Tickets from three consecutive days are required for one trial. The next trial is based on the tickets from a time window that we get by sliding the start date of time window one time unit later. Weekly consecutive feature adaptation experiments have been carried out in a similar setup and the results are shown in Figure 5.8 and Figure 5.9 for account1 and account2 respectively. Account3 has been ruled out of weekly feature adaptation experiments since it only has two weeks tickets, and the experiment requires ticket data from at least three weeks.

While feature mapping learned from the first two consecutive days' tickets are useful for recommending the last day's event ticket resolutions from the first day, it can also degrade the recommendation performance. This causes problems when resolutions indicated by pivot features are quite commonly shared for the first two days but not for the third day. For example, the event tickets occurred in the first two days mainly caused by "software exception" and "system failure" but in the last

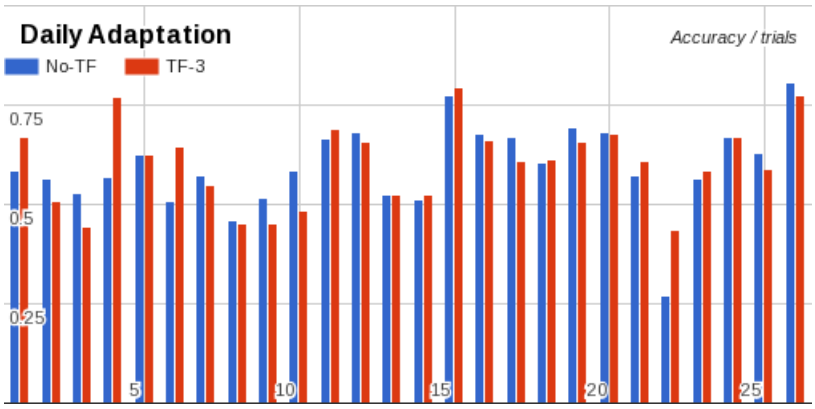

Figure 5.7: Daily adaptation for account1.

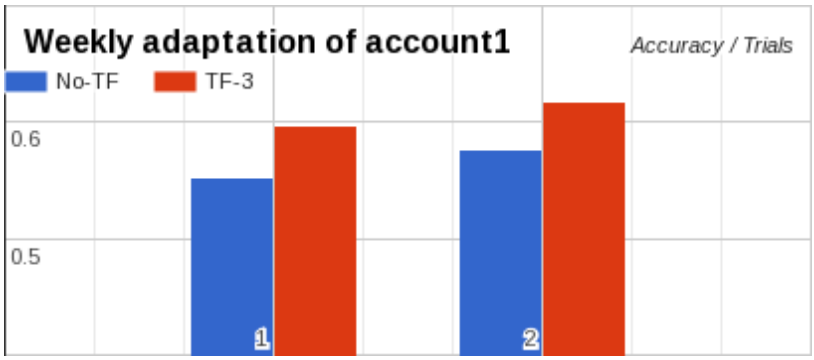

Figure 5.8: Weekly adaptation experimental results on account1's four weeks data. Two trials are carried out since each trial requires three weeks data.

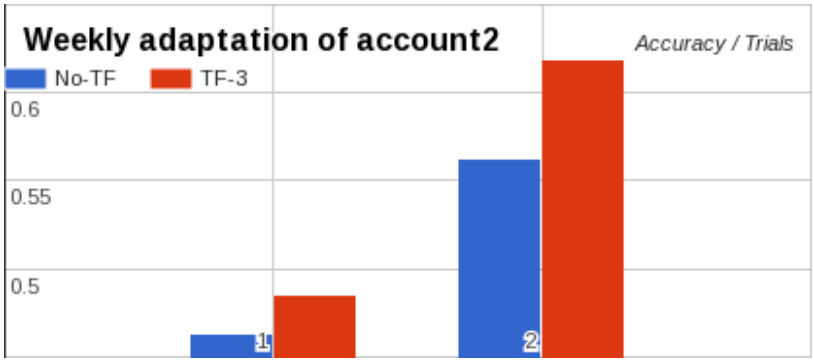

Figure 5.9: Weekly adaptation experimental results on account2's four weeks data. Two trials are carried out since each trial requires three weeks data.

day they occurred because of "low capacity." As shown in Figure 5.7, around half of the trials show performance degradation for our approach "TF-3." These feature mappings, which are not applicable between the first daily tickets and the last daily

ones, are used to project features of the last daily tickets into shared feature space with first daily tickets. They cause noisy and inaccurate similarity calculations by recommending algorithm and degradation of accuracy in resolution recommendation .

Our experiment on weekly ticket datasets achieved positive results as illustrated in Figure 5.8 and Figure 5.9. They indicate that the distribution of event types occurring weekly are similar for these three consecutive weeks. Notice that in our general feature adaptation experiments, around 5 days tickets are used in learning the feature mapping which is nearly one week tickets.

5.6 Summary

This chapter studies the problem of resolution recommendation for monitoring tickets in an automated service management. We analyze several sets of real-world monitoring tickets collected from a production service infrastructure and identify a vast number of repeated resolutions for monitoring tickets. Based on our previous work and some initial experiments, we observe the feature shifting phenomon and the existence of feature mapping in those tickets. In this chapter, we applied structural corresponding learning to the problem of recommending ticket resolution, and conducted extensive experiments on real-world ticket data sets to demonstrate the effectiveness and efficiency of proposed methods.

CHAPTER 6

LEARNING TEXTUAL REPRESENTATION IN RANKING MODEL

In large scale and complex IT service environments, a problematic incident is logged as a ticket and contains the ticket *summary* (system status and problem description). The system administrators log the step-wise *resolution* description when such tickets are resolved. The repeating service events are most likely resolved by inferring similar historical tickets. With the availability of reasonably large ticket datasets, we can have an automated system to recommend the best matching *resolution* for a given ticket *summary*.

In this chapter, we first identify the challenges in real-world ticket analysis and develop an integrated framework to efficiently handle those challenges. The framework first quantifies the quality of ticket resolutions using a regression model built on carefully designed features. The tickets, along with their quality scores obtained from the resolution quality quantification, are then used to train a deep neural network ranking model that outputs the matching scores of ticket summary and resolution pairs. This ranking model allows us to leverage the resolution quality in historical tickets when recommending resolutions for an incoming incident ticket. In addition, the feature vectors derived from the deep neural ranking model can be effectively used in other ticket analysis tasks, such as ticket classification and clustering. The proposed framework is extensively evaluated with a large real-world dataset.

6.1 Introduction

The prominence of efficient and cost-effective service delivery and support is undeniable in the competitive business enterprise and is critical with the growing complexity of service environments. This has motivated service-providing facilities to automate many of their tasks, including system management, and routine maintenance proce-

dures (for instance, problem detection, determination and resolution) for the service infrastructure [ABD+07, MSGL09, WZZ+17]. The automated problem detection has been realized by some system monitoring softwares, such as HP OpenView [urla] and IBM Tivoli Monitoring [urlb] . Such monitoring systems continuously capture system events and generate incident tickets when the alerts are triggered. A typical workflow of problem detection, determination and resolution in IT service management is prescribed by the Information Technology Infrastructure Library (ITIL) specification [urld] and is illustrated in Fig. 6.1. The Incident, Problem, and Change (IPC) system facilitates the tracking, analysis and mitigation of problems and is a requirement for organizations adapting the ITIL framework. A monitoring agent on a server keeps track of the system statistics and triggers an alert when a problem is detected. If an alert persists beyond the specified duration, an event is triggered. Such events are consolidated into an enterprise console, which uses rule-based, case-based or knowledge-based engines to analyze the events and determines whether or not to create an incident ticket in the IPC system [Li15].

Each ticket is stored as a database record that consists of several related attributes (see Table 6.1 for the major attributes) and of their values along with the system status at the time this ticket was generated. Some of the major attributes, such as the **ticket summary** (created by the aggregation of the system status and containing the problem description) and the **ticket resolution** (the textual description of the solution) are critical for diagnosing and resolving similar tickets. Service providers provide an account for every beneficiary that uses the services on a common IT infrastructure. System administrators use the historical tickets and their resolutions from different accounts for problem diagnosis and resolution. The textual description of steps taken to resolve a ticket is logged by the system administrator. Such a human intensive process is quite inefficient in terms of resolution time and cost for

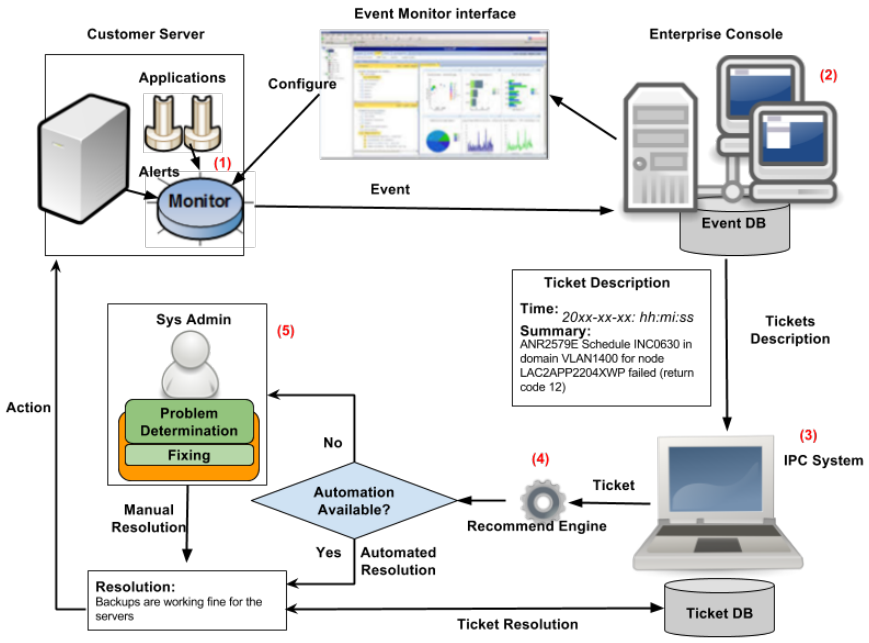

Figure 6.1: Information Technology Infrastructure Library (ITIL) service management system.

large IT service providers that handle many tickets every day. This is one of the major motivations behind the automated analysis of ticket resolution.

SEVERITY	**FIRST-OCCURRENCE**	**LAST-OCCURRENCE**
0	2014-03-29 05:50:39	2014-03-31 05:36:01
SUMMARY	ANR2579E Schedule INC0630 in domain VLAN1400 for node LAC2APP2204XWP failed (return code 12)	
RESOLUTION	Backups are working fine for the server.	
CAUSE	**ACTIONABLE**	**LAST-UPDATE**
Maintenance	Actionable	2014-04-29 23:19:25

Table 6.1: A sample ticket.

6.1.1 Challenges and Proposed Solutions

With the increasing complexity and scalability of IT servers, the necessity of a large-scale efficient workflow in IT service management is undeniable. The samples of real-

world tickets (see Table 6.2 for the contents of tickets that are not easily interpretable) illustrate the unique ticket features that are less intuitive and lead to challenges in IT service management, especially in automated ticket resolution analysis.

ID	Summary	Resolution
1	Box getFolderContents BoxServerException	user doesnt have proper BOX account
2	Box getFolderContents BoxServerException	user should access box terms before access the efile site
3	Box getFolderContents BoxServerException	resolved
4	High space used for logsapp	resolved
5	High space used for disk C	5.24 GB free space present

Table 6.2: Illustration of ticket samples from an account. Only ticket summary and resolution are displayed for the sake of simplicity.

Based on our preliminary studies [ZLSG14b, ZTZ+16], we have identified two key challenges in automating ticket resolution.

Challenge 1 *How to quantify the quality of the ticket resolution?*

Earlier studies generally assumed that the tickets with similar descriptions should have similar resolutions, and often treated all such ticket resolutions equally. However, the study [ZTZ+16] demonstrated that not all of the resolutions are equally worthy. For example, as shown in Table 6.2, the resolution text "resolved" is not useful at all. As a result, the quality of "resolved" is much lower than other resolutions. In order to develop an effective resolution recommendation model, such low-quality resolutions should be ranked lower than high-quality resolutions. In our proposed framework, we first carefully identify relevant features and then build a regression model to quantify ticket resolution quality.

Challenge 2 *How to make use of the historical tickets along with their resolution quality for effective automation of IT service management?*

Although, it might be intuitive to search for historical tickets with the most similar ticket summary, and recommend their resolutions as potential solutions to the target ticket [ZTZ+16], such an approach might not be effective due to 1) the difficulty in representing the ticket summary and resolution, and 2) the avoidance of the resolution quality quantification. It is an essential task in IT service management to accurately represent the ticket summary and resolution. The classical techniques such as the n-gram, TF-IDF, and LDA are not effective in representing tickets as the ticket summary and resolution are generally not well formatted. In our proposed framework, we train a deep neural network ranking model using tickets along with their quality scores obtained from the resolution quality quantification. The ranking model directly outputs the matching scores of ticket summary and resolution pairs. Given an incoming incident, the historical resolutions having top matching scores with its ticket summary can then be recommended. In addition, the feature vectors derived from the ranking model provide effective representations for the tickets and can be used in other ticket analysis tasks, such as ticket classification and clustering.

Recently, only a few studies have focused on ticket resolution [ZTZ+16] . They have adapted the techniques, such as n-gram, Jaccard similarity and LDA [Alp14, BNJ03], which are utilized mostly for processing well-formed text. As the textual attributes of real-world tickets are far from the well-formed natural language (see Table 6.1 for the ticket attributes), the studies relying just on the classical techniques cannot be effective.

To the best of our knowledge, none of the existing studies has attempted to address the aforementioned challenges. The main contributions of this chapter are: (i) Identification and explanation of typical traits of the real-world tickets and the major challenges in their analysis and resolution; (ii) Formulation of the problem as an integrated deep neural network-based ranking framework and efficient handling

those challenges; (iii) Generalization of the ticket representation and successful application to other ticket analysis tasks, such as, ticket classification and clustering; (iv) Extensive evaluation of the proposed model against a large real-world dataset.

6.1.2 Road Map

The rest of this chapter is organized as follows. Section 6.2 gives an overview of framework. Section 6.3 describes the pre-process on tickets and the features used to train the model for quantifying the quality of ticket resolution. In Section 6.4, we introduce our proposed deep neural ranking model. Automation of resolution recommendation is studied in Section 6.5 and ticket clustering and classification are evaluated in 6.6. Comprehensive experiments are conducted and presented in Section 6.5, 6.6. Finally, Section 6.7 concludes the chapter.

6.2 Overview

In this section, we provide a high-level description of the system. As illustrated in Figure 6.2, the training data taken from the historical tickets dataset are first preprocessed in order to quantify and evaluate the quality of the resolution. The preprocessed result is then represented as a triplet of the ticket summary, its resolution text, and the quality score. These triplets are the training data for the proposed deep neural network (DNN) ranking model. The trained DNN model outputs a matching score of a quantified ticket resolution for an incoming ticket summary. The resolutions with the top N highest matching score can be recommended for an incoming ticket. The model's intermediate result is a feature vector for a ticket representation. Such vectors are used in other ticket analysis tasks, such as ticket classification and ticket clustering.

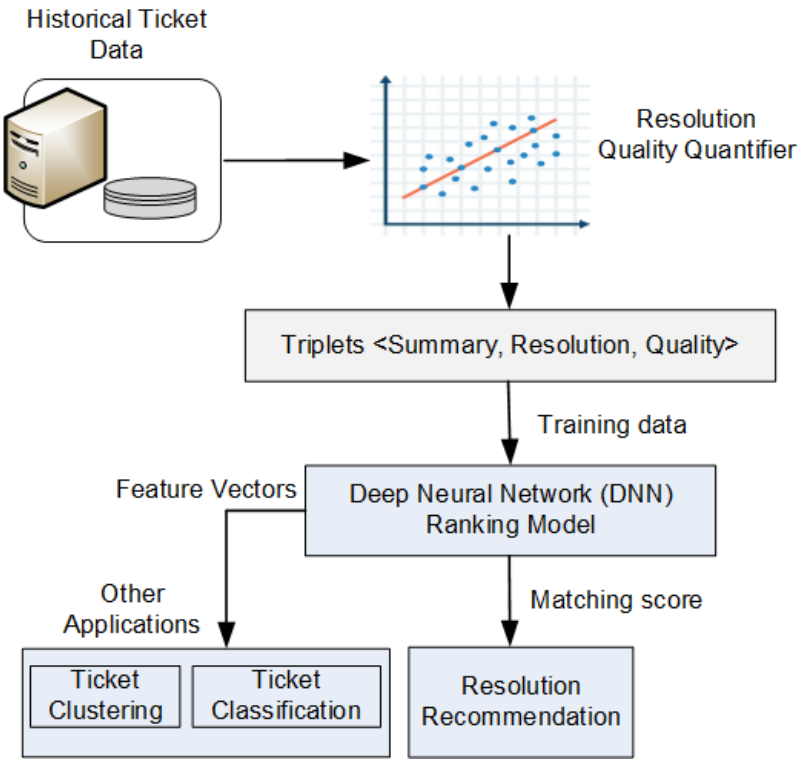

Figure 6.2: Overview of the proposed system.

6.3 Ticket Resolution Quality Quantification

In this section, we describe the features used to quantify the quality of ticket resolutions and present several interesting findings from our experiments.

As shown in Table 6.1, a ticket resolution is a textual attribute of a ticket. A high quality ticket resolution is supposed to be well written and informative enough to describe the detailed actions taken to fix the problem specified in the ticket summary. A low-quality ticket resolution is less or non-informative and is mostly logged by a careless system administrators or when the corresponding issue described in the ticket is negligible. Based on our long preliminary study [ZTZ+16], we've found that for a typical ticket, the ticket resolution quality is driven by the 33 features that can be broadly divided into following four groups:

• **Character-level features:** A low-quality ticket resolution might include a large number of unexpected characters, such as space, wrong or excessive capitalization, and special characters.

• **Entity-level features:** A high-quality ticket resolution is expected to provide information on IT-related entities, such as server name, file path, IP address, and so forth. Because the ticket resolutions are expected to guide system administrators to solve the problem specified in the ticket summary, the presence of the context-relevant entities makes the resolution text more useful.

• **Semantic-level features:** A high-quality ticket resolution typically includes *Verb* and *Noun*, which explicitly guides system administrators on the actions taken to diagnose the problem and to resolve the ticket.

• **Attribute-level features:** A high-quality ticket resolution usually is lengthy enough to carry sufficient information relevant to the problem described in the ticket summary.

The ticket resolution quality quantifier uses these 4 groups of features and operates on the historical tickets to output a set of triplets $\{< s_1, r_1, q_1 >, < s_2, r_2, q_2 >, \ldots, < s_n, r_n, q_n >\}$ where s_i and r_i are ticket summary and ticket resolution for the i^{th} ticket, and q_i is the quality score assigned by the quantifier.

6.3.1 Feature Description

Character-level features. To quantify the use of character usage, we considered each of the nine character classes (*exclamationRatio, colonRatio, braketRatio, @Ratio, digitRatio, uppercaseRatio, lowercaseRatio, punctuationRatio, whitespaceRatio*) as a feature and then computed their frequency to all the characters within the ticket resolution.

Concept	Pattern	Examples
Action	NOUN/NP preceded/succeeded by VERB	(file) is (deleted)
Problem	NOUN/NP preceded/succeeded by ADJ/VERB	(capacity) is (full)

Table 6.3: PoS tag pattern for concepts *problem, action*. NP refers to noun phrase derived from the PoS tag sequence for each resolution.

Entity-level features. To quantify the usage of IT related entities, we considered each of the eight entity classes (*numericalNumber, percentNumber, filepathNumber, dateNumber, timeNumber, ipNumber, servernameNumber, classNumber*) as a feature and computed their frequency to all the words within the ticket resolution. The occurrence of these entities was captured using the regular expressions. For the *filepathNumber*, it refers the total occurrence of Linux and Window file path in the ticket resolution. For the *classNumber*, we considered the total occurrence of class names or functions in the programming languages, such as Java, Python, and so forth. We also explored some other entities, but in comparison to other features, their contribution to overall model performance was negligible.

Semantic-level features. To quantify the usage of those specific semantic words, we first preprocessed every ticket resolution into a Part-Of-Speech (PoS) [PDM11] tag sequence and then calculated the ratio of each tag within the tag sequence. There were 17 total tags, including the tag *"X"* for the foreign words, typos and abbreviations, they were reduced to 12 tags in the NTLK implementation [Bir06]. Each of the 12 tags, *VERBRatio, NOUNRatio, PRONRatio, ADJRatio, ADVPNRatio, ADPRatio, CONJRatio, DETRatio, NUMRatio, PRTRatio, PUNCTRatio, XRatio*, were considered as a feature. Furthermore, we borrowed the concepts, such as, *Problem, Activity and Action* in work [PJNR13] and defined the corresponding PoS tag pattern, as shown in Table 6.3. We reduced the three concepts into two by merging the concepts *Activity, Action* into the concept *Action* and then used the regular expressions to calculate the occurrence of each concept feature *problemNum, actionNum*.

Attribute-level features. To quantify the high-quality resolution in ticket, we included two attribute-level features *resolutionLength, interSimilarity* in our model. The first one was used for the ticket resolution length. The second one was used to record the Jaccard similarity between a ticket's summary and its resolution, and was used to define the relevance between them.

6.3.2 Findings

We evaluated three of the most popular regression models (logistic regression, gradient boosting tree and random forest [Alp14]) on the labeled real-world ticket dataset and found that the random forest performed best for the ticket resolution quantification and also for evaluation of the feature importances, as illustrated in the Table 6.4.

Based on our evaluation, we found that the best indicator of a good resolution was the length of the resolution *resolutionLength*, followed by the occurrence of the concept action, i.e., feature *actionNum*. It is also self-intuitive that the long resolution can be more informative. The features *actionNum* and *problemNum* correspond to the problems identified and the actions taken by the system administrators in the process of resolving the ticket.

Another interesting finding was that seven out of the top 15 features belonged to the group *word level semantic features*, and were specifically derived from the PoS tag sequence. The 3^{rd} top-ranked feature was *PRTRatio* related to the ratio of the words tagged as particle or function words. This implied that the resolutions containing the function words such as "for" and "due to" have a high quality. Moreover, high-quality resolutions were usually well written and complied with the natural language syntax, while the low-quality resolutions, on the other hand, were ill-formated and caused great difficulty for the PoS tagger trained on natural languages. In summary, the

Feature Group	**Importance score**		
Feature	Rank	Mean	Variance
Character-features			
uppercaseRatio	12	0.026123	0.008717
lowercaseRatio	10	0.049657	0.008206
punctuationRatio	11	0.036442	0.008710
whitespaceRatio	9	0.049123	0.008610
Entity-level features			
servernameNumber	13	0.018770	0.008553
Semantic-level features			
VERBRatio	7	0.079400	0.009091
NOUNRatio	4	0.088025	0.009420
ADJRatio	14	0.013885	0.009048
ADVRatio	5	0.084971	0.008327
DETRatio	8	0.055133	0.008147
PRTRatio	3	0.090921	0.022932
PUNCTRatio	15	0.008797	0.008228
problemNum	6	0.080322	0.008480
actionNum	2	0.147252	0.038538
Attribute-level features			
resolutionLength	1	0.152234	0.043585
MSE Avg.	0.010269	**MSE Var.**	0.004163

Table 6.4: Illustration of the top 15 ranked features and their rank evaluated by the random forest regression model. To best evaluate the feature importance score, we show the rank of average importance score, its mean and variance. The best performance in the metric of both MSE (mean square error) average and variance is attached of the end.

semantic features have predominant advantages in characterizing and quantifying the ticket resolution quality over the other features.

6.4 Deep Neural Ranking Model

In our preliminary work [ZTZ+16], we model automating ticket resolution task as an information retrieval problem and tackle it from the perspective of finding a similar ticket summary in historical data, in which we treat each ticket resolution equally. However, given the triplets $\{< s_1, r_1, q_1 >, < s_2, r_2, q_2 >, \ldots, < s_n, r_n, q_n >\}$ from

section 6.3, we can definitely improve the automating ticket resolution task by considering the quality of resolutions.

6.4.1 Problem formulation

In this section, we view the automating ticket resolution task as text pair ranking task, which is one of the most popular tasks in the information retrieval (IR) domain.

As shown in Table 6.2, the ticket with the same ticket summary can be resolved by multiple resolutions with different qualities. In automating ticket resolution, we expect the model to recommend all the possible resolutions, but with the order in which high quality resolution ranks first. Therefore, given the triplets $\{< s_1, r_1, q_1 > , < s_2, r_2, q_2 >, \ldots, < s_n, r_n, q_n >\}$ from section 6.3, the goal is to build a model that for ticket summary s_i generates an optimal ranking score y_i for each resolution, s.t. a relevant resolution with a high quality has a high ranking score.

More formally, the task is to learn a ranking function:

$$h(w, \psi(s_i, r_i)) \rightarrow y_i$$

where function $\psi(\cdot)$ maps ¡summary, resolution¿ pairs to a feature vector representation, where each component reflects a certain type of similarity, e.g., lexical, syntactic, or semantic. The weight vector w is a parameter of the model and is learned during the training.

There are three common approaches in information retrieval to learn the ranking function, namely, *pointwise, pairwise* and *listwise* [L$^+$09].

Pairwise and *listwise* approaches yield better performance most of the time since they exploit more information from the ground truth ordering, meanwhile they are more complicated to implement and take more time to train. In this work, the training data naturally comes as *pointwise*, and producing a better representation

$\psi(\cdot)$ that encodes ticket summary, resolution or even whole ticket is one of our goals. Hence, we adopt the simple *pointwise* ranking model and focus more on modeling the representation for a ticket and its components using deep learning techniques.

6.4.2 Deep Neural Ranking Architecture

In this section, we propose a deep neural ranking model to solve the problem. The model consists of two *sentence model* [KGB14] for mapping ticket summary and resolution to their vector representation, respectively. We argue that it plays an important role in automation of IT service management to derive an efficient representation for ticket summary and resolution from the ranking model.

In the following sections, we first describe the *sentence model* for mapping ticket summary and resolution to their distributed vectors and then describe how they can be used to learn semantic similarity metric between ticket summary and resolution for ranking.

Sentence Model

The architecture of our CNN-based model is shown in Fig 6.3. It is inspired by the CNN model for performing various sentence classifications [KGB14].

Our network is composed of a single embedding layer, two repeated composite structures and a final fully connected layer that output the distributed representation. The composite structure consists of one *wide* convolutional layer followed by a non-linearity and k-max pooling. The input to the network includes not only the raw words, but also the raw characters. We will briefly explain the components of our neural network.

Embedding Layer

The input to our sentence model is a sentence s treated as a sequence of words and characters: $[w_1, .., w_{|wl|}, e_1, .., e_{|el|}]$, where each word and character is drawn from a word vocabulary V and a character vocabulary E, respectively. Words are represented by distributional vectors $w \in \mathbb{R}^d$ in a word embeddings matrix $W \in \mathbb{R}^{d \times |V|}$. Characters are represented in a similar way. We set the same dimension for word and character embedding, and merge two vocabularies into one $T = V \cup E$ as well as the embedding matrices $W \in \mathbb{R}^{d \times |T|}$. Each input sentence s is represented by a sentence matrix $S \in \mathbb{R}^{d \times |t|} = [w_1, \ldots, w_{|t|}]$, where t is the total length of words and characters in s.

Convolution Layer

Convolutional layer aims to extract interesting patterns of word and character sequences. Concretely, we harness the *one-dimensional convolution* operation working on two vectors $s \in \mathbb{R}^{|s|}$ and $f \in \mathbb{R}^m$ (a filter of size m) and taking the convolution operation in each m-size window of the sentence s to obtain another sequence $\mathbf{c}$:

$$c_j = (s * f)_j = s^T_{j-m+1:j} \cdot f = \sum_{k=j}^{j+m-1} s_k f_k \tag{6.1}$$

where each row vector $c_j \in \mathbb{R}^{|s|+m-1}$ in C results from a convolutional operation between jth row vector in S and jth row vector in F.

In practice, a set of filters, packed as $F \in \mathbb{R}^{n \times d \times m}$, that work in parallel are applied in a deep learning model, producing multiple feature maps $C \in \mathbb{R}^{n \times d \times (|s|+m-1)}$. To allow the network learn an appropriate threshold, a bias matrix $B \in \mathbb{R}^{n \times d}$ is added to the result the feature maps.

Activation Layer

Following a convolutional layer, activation layer with a rectified function $\alpha(\cdot)$ is applied elementwise to the input, i.e., the output from convolutional layer.

Folding Layer

The dependency between different rows is captured by Folding Layer, which sums up every two rows in a feature map component-wise. For a map of d rows, folding reduces it into a map of $d/2$ rows.

Pooling Layer

The output from the convolutional layer (passed through the activation function) is then passed to the pooling layer, whose goal is to aggregate the information and reduce the representation. We use dynamic k-max pooling [NH10] to build rich feature representations of the input.

Architecture for ranking ticket summary and resolution pairs

The partial network architecture introduced in Section 6.4.2 takes a sentence as input and outputs a distributional vector. Applied to a pair of ticket resolution and summary, it will output two distributional vectors with the same dimension thus, a similarity score can be computed, which together with the two vectors are concatenated into a single representation, shown in Fig. 6.3.

In the following section, we briefly introduce how the intermediate distributed representation produced by the sentence model can be used to compute the matching scores of the ticket summary and resolution pairs.

Representation for ticket summary and resolution pair

Having the output of our sentence model for processing ticket summary and resolution, respectively, the resulting representation vectors x_s and x_r, can be used to compute the ticket summary and resolution similarity score as follows:

$$sim(x_s, x_r) = x_s^T M x_r \tag{6.2}$$

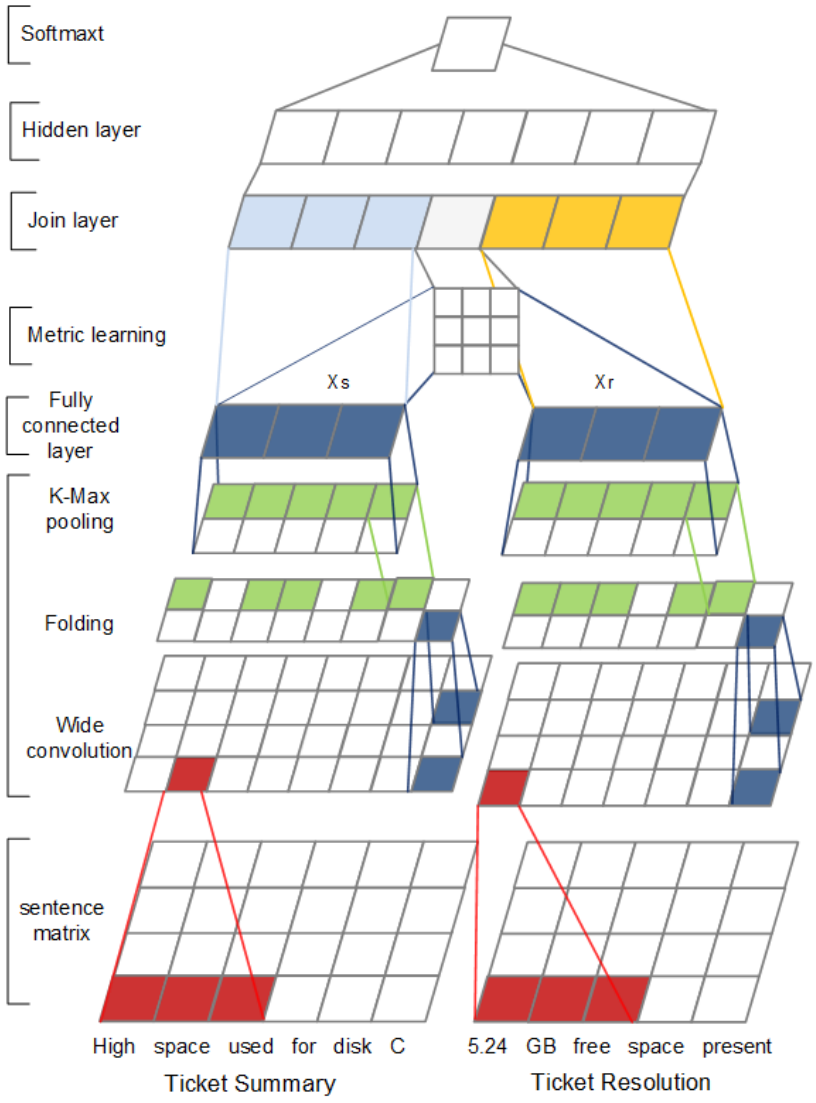

Figure 6.3: Ranking model. The character level embedding is not shown for the sake of saving space.

Where $M \in \mathbb{R}^{d \times d}$ is a similarity matrix, it acts as a model of noisy channel approach for machine learning, which has been commonly adopted as a scoring model in information retrieval and question answer [EM03]. It can also be viewed as a process of learning similarity metric on two vectors drawing from different feature spaces [K$^+$13]. The similarity matrix M is a parameter of the network and is optimized during the training.

Multilayer Perceptron

The joint vector is then passed through a 3-layer, fully-connected, feed-forward neural network, which allows rich interactions between a sentence pair from one of the three components. Finally, a single neuron outputs the score between a query (or the context) and a reply for a linear regression.

Objective Function

The model is trained to minimize the binary cross-function:

$$\begin{aligned} L &= -\log \Pi_{i=1}^{N} p(y_i|\mathbf{s_i}, \mathbf{r_i}) \\ &= -\sum_{i=1}^{N} [y_i \log a_i + (1 - y_i) \log(1 - a_i)] \end{aligned} \tag{6.3}$$

Where y_i is the ground truth for instance i while a_i is the prediction.

The parameters of the network are optimized with Adadelta [Zei12] with the gradients computed by back propagation algorithm.

6.4.3 Regularization

To mitigate the overfitting issue we augment the cost function with L_2-norm regularization terms for the parameters of the network. Also, dropout [SHK+14] is employed to prevent feature co-adaptation by setting to zero (dropping out) a portion of hidden units during the forward phase.

6.4.4 Word Embedding

While our model allows for learning the word embeddings directly for a given task, we initialize the word matrix parameter W from an unsupervised neural language

model [PSM14]. Although according to a common experience that a minimal size of the dataset required for tuning the word embeddings for a given task should be at least in the order of hundred thousands, and in our case the number of ticket summary resolution pairs is sufficient, the wide existence of special words in tickets results in a much larger vocabulary size than in common natural language. We choose the dimensionality of our word embeddings as well as character embeddings to be 50.

This ends the description of our entire ranking model. In the following, we first present experiments on training the deep neural model and its performance on automating ticket resolution.

6.5 Automating Ticket Resolution

This section evaluate the proposed deep neural ranking model on automating ticket resolution against a series of baselines.

6.5.1 Datasets

To keep the consistency of our experiments, we conduct all the experiments on historical tickets from one single ticket account, which consists of a total of 479,079 tickets with more than 30% labeled. Therefore, the only labeling effort is devoted to train the resolution quality quantifier. We summarize the usage of dataset in Table 6.5.

System	Training	Validation	Testing
Resolution Quality Quantifier	5000	–	1000
Ticket Resolution Automation	450,000	20,000	9,000
Ticket Clustering	10,000	–	2,000
Ticket Classification	20,000	–	3,000

Table 6.5: Ticket dataset summary.

6.5.2 Ticket Resolution Automation

Evaluation Metrics

Given the ranking lists based on their resolution quality score for test tickets, we evaluated the performance in terms of the following metrics: precision@1 (p@1), mean average precision (MAP) [Zhu04], and normalized discounted cumulative gain (nDCG) [JK02]. Because the system outputs the best selected resolution, p@1 is the precision at the 1st position, and should be the most natural way to indicate the fraction of suitable resolution among the top-1 results retrieved. Besides, we also provided the top-k ranking list for the test ticket using nDCG and MAP, which test the potential for a system to provide more than one appropriate resolutions as candidates. We aimed at selecting as many appropriate responses as possible into the top-k list and rewarding methods on the top that return suitable replies.

Algorithms for Comparison

Automating ticket resolutions can be tackled from different perspectives, hence this section mainly focuses on implementing potential competing solutions for automating ticket resolution from different perspectives and proving each one's effectiveness. We include several alternative algorithms for comparison. The algorithms can be divided into two big categories, i.e., 1) generation-based methods and 2) retrieval-based methods.

Generation-based method. For this group of algorithms, the system will generate a response from a given input. Hence, we use beam search [TN03] to enable them to search for more than one response.

• Statistical Machine Translation (SMT): SMT [RCD11] is a machine translation paradigm that translates one sentence in the source language to a sentence in the

target language. If we treat ticket summary and resolution as separate languages, we can train a translation model to "translate" summary into resolution.

• LSTM-RNN: LSTM-RNN is a Recurrent Neural Network (RNN) using the Long Short Term Memory (LSTM) architecture. The RNN with LSTM units consists of memory cells in order to store information for extended periods of time. We first use an LSTM-RNN to encode the input sequence (ticket summary) to a vector space, and then use another LSTM-RNN to decode the vector into the output sequence (ticket resolution) [SVL14].

Retrieval-based method. The approaches within this group of baselines are based on retrieval systems, which return the best matched candidate resolution out of the historical ticket data repository given a particular new unresolved ticket.

• Random Shuffle. The method randomly selects replies for each query from the retrieved resolution list obtained from tickets having closest (Jaccard distance) ticket summaries as the query. However we only randomize the order of the retrieved resolution candidates instead of randomly choosing the candidates. The true random match is too weak to be included as a decent baseline.

• CombinedLDAKNN. This is one approach adopted in our previous work [ZTZ+16] on automating ticket resolution task without demanding any labeling efforts. It first trains an LDA model on whole historical tickets. For each new ticket, we retrieve the most relevant resolution, directly applying cosine similarity on the feature vector for tickets inferred from the trained LDA model.

Results

Overall performance results are shown in Table 6.6. We have some interesting observations. The performance of the generative methods is quite moderate, which concurs the judgment from [SLL15]. The automatic resolution generators tend to produce

System	p1	MAP	nDCG5	nDCG10
SMT	0.421	0.324	0.459	0.501
LSTM-RNN	0.563	0.367	0.572	0.628
Random Shuffle	0.343	0.273	0.358	0.420
CombinedLDAKNN	0.482	0.347	0.484	0.536
Our method	**0.742**	**0.506**	**0.628**	**0.791**

Table 6.6: Overall performance comparison. For generative model, we enable beam search for multiple output result, so that MAP and nDCG scores can be computed.

universal, trivial and ambiguous resolutions, which are likely to resolve a wide range of tickets, but not specific enough to conduct a meaningful remediation on faulted servers, i.e., low quality resolutions. This leads to the overwhelming performance of retrieved methods over generative methods.

When it comes to phrase-based SMT, it is very tricky to segment a large part of ticket summaries into meaningful words or phrases since they are automatically generated by machines and can be extremely noisy. In general, generative approaches using deep learning (LSTM-RNN) outperform those without deep learning techniques and more advantage can be gained using input with character level order information.

With respect to retrieval-based methods, they attempt to obtain a ranked list of candidate resolutions, which show great potential to conduct system diagnosis and resolving with diversity. Among retrieval-based methods, Random shuffle is a lower bound for all baselines. As we mentioned, it randomizes the order of the retrieved results. Hence, the result is still promising as the straightforward index approach. CombinedLDAKNN slightly outperform the SMT approach, which is not surprising. The trained LDA model enables the algorithm to learn data statistic information, such as resolution popularity, the correlation between ticket summary and resolution, which benefits retrieving high relevant resolutions. The performance of deep learning based algorithms in general overwhelms that of shallow learning-based algorithms.

6.6 Other Ticket Analysis Applications

Category	Measure	Formula	Note
Surface Matching Similarity	Jaccard [GBJR08]	$S_{JAC}(T_1, T_2) = \frac{\|A \cap B\|}{\|A \cup B\|}$	A and B be sets of words in two ticket descriptions
	N-word overlap [AHS08]	$S_{nwo}(T_1, T_2) = \tanh(\frac{overlap_{phrase}(T_1,T_2)}{n_1+n_2})$	A phrasal n-word overlap
	NLCS [II08]	$S_{NLCS}(T_1, T_2) = \frac{(\|LCS(T_1,T_2)\|)^2}{\|T_1\| \times \|T_2\|}$	Considering the length of both the shorter and the longer string
Semantic Similarity	Leacock & Chodorow [LC98]	$S_{lch}(c_1, c_2) = -\frac{\log len(c_1,c_2)}{2 \times D}$	Path-based method using wordnet
	RES [Res95]	$S_{res}(c_1, c_2) = IC(lcs(c_1, c_2))$	Information content-based method
	Word2vec based [MSC+13]	$S_{w2v}(w_1, w_2)$	Word2vec based word similarity using wikipedia
Hybrid Similarity	ISLAM's measure [II08]	$S_{STS}(T_1, T_2) = \frac{(\delta(1-w_f+w_f S_o)+\sum_{i=1}^{\rho} \rho_i) \times (m+n)}{2mn}$	Combining string similarity, semantic similarity and common-word order similarity
	Li's measure [LMB+06]	$S_{SSI}(T_1, T_2) = \delta \frac{s_1 \cdot s_2}{\|s_1\| \cdot \|s_2\|} + (1-\delta)\frac{\|r_1 - r_2\|}{\|r_1 + r_2\|}$	Considering semantic similarity and word-order similarity
	SyMSS [OSdCI11]	$S_{symss}(T_1, T_2) = \frac{1}{n}\sum_{i=1}^{n} sim(h_{1i}, h_{2i}) - l \times PF$	Considering semantic and syntactic info
	our method	$S_{TicDNN}(s1, s2) = cosine(x_{s1}, x_{s2})$	x_s is the vector representation for ticket summary s

Table 6.7: The evaluated similarity measures including 3 categories and 10 measures. The distributed representation for tickets learned in our model capture both string and semantic similarity, thus we categorize it as hybrid similarity.

In this section, we demonstrate that the vector representation x_s for ticket summary and x_r for ticket resolution, derived from our sentence model play an important role in the automation of IT service management, such as ticket clustering and ticket classification. More specifically, we focus on the vector representation for ticket summary x_s since most tasks in the IT service management are accomplished before resolving tickets. The comprehensive empirical experiments conducted on the real

world ticket data set (see Table 6.5 for details) illustrate the effectiveness of the learned vector representation.

6.6.1 Ticket Clustering

In IT service management, ticket clustering is important for optimal ticket dispatching [BBG+15] to relevant service teams. The role of similarity metrics is crucial for any clustering algorithm. In this section, we compare the performance of the clustering based on the ticket's feature vector with other popular metrics when applied to the k-means clustering method (which assigns the ticket to the closest group). The evaluated similarity measures can be classified into three categories: *surface matching methods*, *semantic similarity methods*, and *hybrid methods* (see Table 6.7 for their formula).

The $F1$ scores [AGK+12] obtained for evaluating the ticket clustering task over different measures are illustrated in Table 6.8. To ensure the fairness of comparisons, the F1 score for each measure was taken as the median from the 10 trials of different testing samples (we used the worst case, the median case, and the best case). Moreover, the value with the bold font in each column denotes the best value for that case corresponding to the column. As shown in the table, the hybrid similarity measure performed better than those using the simple similarity measures, the surface matching similarity measures, and the semantic similarity measures. These findings provide an optimistic insight for the development of a new similarity measure by incorporating information from additional sources. Meanwhile, we also found that the semantic similarity measures performed better than the surface matching similarity measures in most cases. A possible reason could be that most of the words recognized by the surface matching measure can also be recognized by the semantic similarity measures using a well-known knowledge base. However, the semantic

similarity measures do not work well for non-English dictionary words because these words are domain-specific and are still not included in common knowledge bases. This makes the surface matching similarity measure more relevant to our framework even though it has a relatively low contribution to the overall similarity. Our proposed method outperformed all the listed similarity measures, which illustrates its ability to better capture the string similarity, semantic similarity and word order similarity simultaneously (even slightly better than S_{STS}).

Measures	F1 score		
	Worst	Avg.	Best
S_{JAC}	0.4318	0.5677	0.7024
S_{nwo}	0.4763	0.5998	0.7043
S_{NLCS}	0.5325	0.6332	0.7221
S_{lch}	0.6823	0.7427	0.7866
S_{res}	0.6885	0.7576	0.7969
S_{w2v}	0.7538	0.8169	0.8693
S_{STS}	0.8048	0.8553	0.8953
S_{SSI}	0.8035	0.8497	0.8834
S_{symss}	0.8042	0.8503	0.8885
S_{TicDNN}	**0.8103**	**0.8595**	**0.9002**

Table 6.8: Comparisons of F1 scores using different similarity measures.

6.6.2 Ticket Classification

The ticket classification is an important step in the automation of ticket assignment across the processing teams in the IT service management. In this section, we illustrate the efficiency of the ticket summary vector representation x_s by applying it to a hierarchical multi-label classification task [ZZL+17].

Let $x = (x_0, x_1, ..., x_{d-1})$ be an instance from the d-dimensional input feature space χ, and $y = (y_0, y_1, ..., y_{N-1})$ be the N-dimensional output class label vector where $y_i \in \{0, 1\}$. A multi-label classification assigns a multi-label vector y to a given instance x, where $y_i = 1$ if x belongs to the ith class, and $y_i = 0$ otherwise.

The hierarchical multi-label classification is a special type of multi-label classification when a hierarchical relation H is predefined on all class labels. The hierarchy H can be a tree, or an arbitrary DAG.

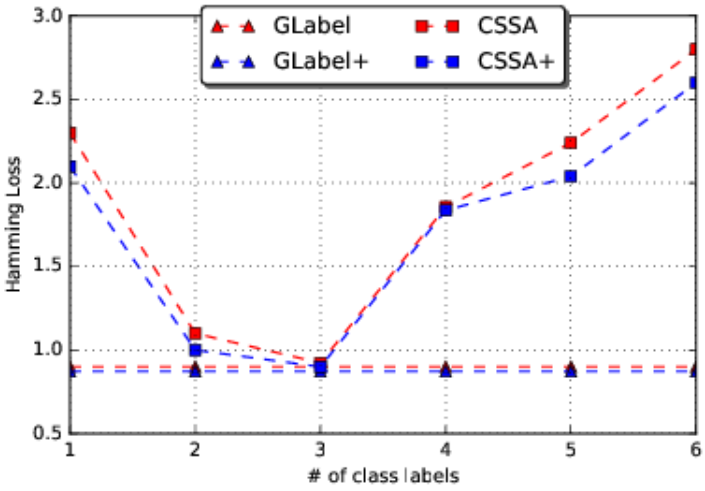

Figure 6.4: The lowest Hamming loss: *GLabel* gets 0.901 and *GLabel*+ 0.872; *CSSA* gets 0.923 and *CSSA*+ 0.901.

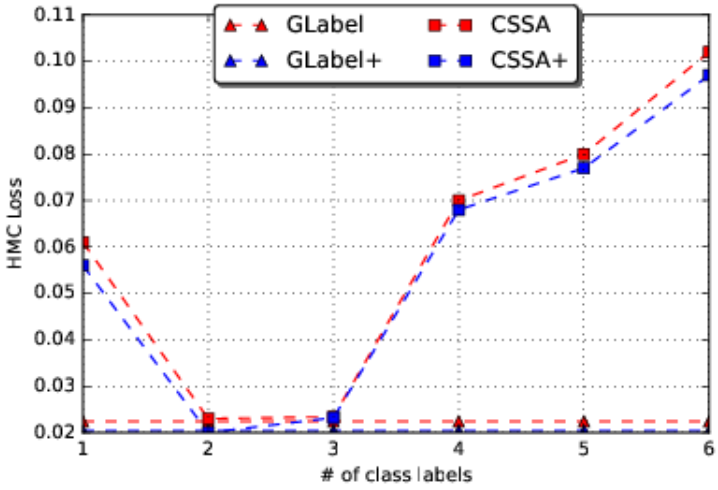

Figure 6.5: The lowest HMC-Loss: *GLabel* gets 0.022 and *GLabel*+ 0.020; *CSSA* gets 0.023 and *CSSA*+ 0.023.

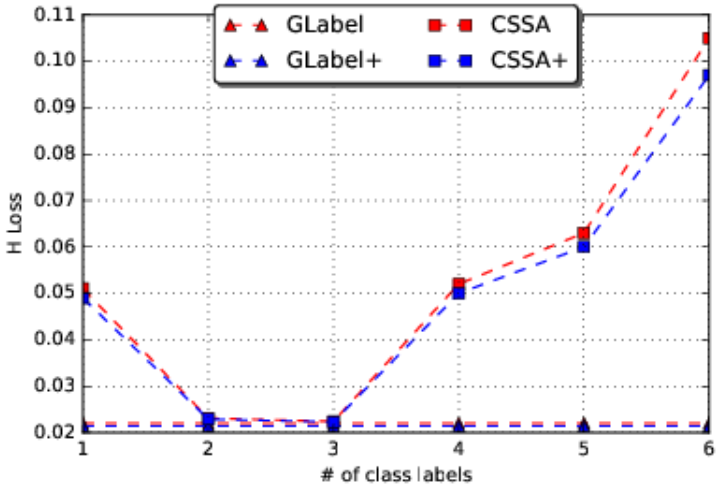

Figure 6.6: The lowest H-Loss: *GLabel* gets 0.022 and *GLabel+* 0.021; *CSSA* gets 0.023 and *CSSA+* 0.21.

Evaluation Metrics

In order to illustrate the effectiveness of our model, we introduced several metrics to evaluate the hierarchical multi-label classification problem including Hamming-loss, H-loss and HMC-loss [ZZL+17].

- Hamming-Loss: calculated by the fraction of the misclassification to the total number of predictions.
- H-Loss: penalized only the first classification mistake along each prediction path.
- HMC-Loss: weighted the misclassification with the hierarchy information while avoiding the deficiencies of the H-loss

Algorithms for Comparison

We perform the experiments over the same setup as the previous study [ZZL+17], where *GLabel*, a hierarchical multi-label classification algorithm, was proposed to classify tickets and has achieved better performance over the state-of-the-arts algorithms. We compared the performance of two classification algorithms on the original feature

representation (*GLabel* and $CSSA$) and the derived feature representation (*GLabel+* and $CSSA+$). We found that the derived feature representation was efficient.

The performance comparison is shown in Figures 6.4- 6.6. *GLabel+* and $CSSA+$ outperformed their counterparts (*GLabel* and $CSSA$) which indicates the effectiveness for our derived feature representation.

6.7 Summary

In this Chapter, we presented the major challenges in ticket resolution, such as quality quantification of ticket resolutions and consideration of resolution quantification in a recommendation problem. We defined a deep neural network-based ticket resolution recommendation framework and evaluated it against a large real-world dataset. The evaluation demonstrated the effectiveness of the proposed model. Moreover, The distributed representation induced by the network is able to capture semantical relations of noisy ticket components, and can be applied to relevant fundamental applications in ticket analysis, such as ticket clustering, ticket classification and so on.

CHAPTER 7

CONCLUSION AND FUTURE WORK

Intelligent textual understanding for computers can alleviate the human effort investment, and thereby reduce the risk of human mistakes involved in many repetitive tasks. Many data mining techniques can be utilized to achieve the goal of various intelligent textual understanding tasks. After meticulously studying, four research directions are identified from data mining perspective, with the purpose to intelligent understanding textual data for computers in the areas of disaster management and IT service management.

The four research directions are highlighted as below:

1. Intelligent generate a storyline summarizing the evolution of a hurricane from relevant online corpus.
2. Automatically recommending resolutions according to the symptom description in a ticket.
3. Gradually adapting the resolution recommendation system for time correlated features.
4. Efficiently learning distributed representation for text using scalable deep neural ranking model.

To follow up with work in my dissertation, some future work along the related directions are provided.

1. We focus on linear storyline in the dissertation, which can be extended to more complicated evolution structures of different disaster types for storyline generation, such as earthquakes and other man-made disasters.
2. There are several avenues for future research in automating ticket resolution. In this dissertation, we recommend the resolution directly. However, it is worthy

investigating and developing intelligent classification techniques to automatically label resolutions [ZLSG14a, CQT+13] first, and then recommend the resolution in related categories. Furthermore, it is possible to use an active query strategy to fully automate resolution recommendations.

BIBLIOGRAPHY

[ABD+07] Naga Ayachitula, Melissa Buco, Yixin Diao, Surendra Maheswaran, Raju Pavuluri, Larisa Shwartz, and Chris Ward. It service management automation-a hybrid methodology to integrate and orchestrate collaborative human centric and automation centric workflows. In *Services Computing, 2007. SCC 2007. IEEE International Conference on*, pages 574–581. IEEE, 2007.

[ABYG09] Omar Alonso, Ricardo Baeza-Yates, and Michael Gertz. Effectiveness of temporal snippets. In *WSSP Workshop at the World Wide Web ConferenceWWW*, volume 9, 2009.

[ADNR07] Shipra Agrawal, Supratim Deb, KVM Naidu, and Rajeev Rastogi. Efficient detection of distributed constraint violations. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 1320–1324. IEEE, 2007.

[AGK+12] Elke Achtert, Sascha Goldhofer, Hans-Peter Kriegel, Erich Schubert, and Arthur Zimek. Evaluation of clusterings–metrics and visual support. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, pages 1285–1288. IEEE, 2012.

[AHS08] Palakorn Achananuparp, Xiaohua Hu, and Xiajiong Shen. The evaluation of sentence similarity measures. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 305–316. Springer, 2008.

[All12] James Allan. *Topic detection and tracking: event-based information organization*, volume 12. Springer Science & Business Media, 2012.

[Alp14] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2014.

[AZ05] Rie Kubota Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *The Journal of Machine Learning Research*, 6:1817–1853, 2005.

[BBG+15] Mirela Madalina Botezatu, Jasmina Bogojeska, Ioana Giurgiu, Hagen Voelzer, and Dorothea Wiesmann. Multi-view incident ticket clustering for optimal ticket dispatching. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1711–1720. ACM, 2015.

[BDP+07] John Blitzer, Mark Dredze, Fernando Pereira, et al. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, volume 7, pages 440–447, 2007.

[Ben75] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[BGL+14] Jasmina Bogojeska, Ioana Giurgiu, David Lanyi, George Stark, and Dorothea Wiesmann. Impact of hw and os type and currency on server availability derived from problem ticket analysis. In *Network Operations and Management Symposium (NOMS), 2014 IEEE*, pages 1–9. IEEE, 2014.

[BH03] Bart Bakker and Tom Heskes. Task clustering and gating for bayesian multitask learning. *Journal of Machine Learning Research*, 4(May):83–99, 2003.

[Bir06] Steven Bird. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics, 2006.

[BK07] Robert M Bell and Yehuda Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 43–52. IEEE, 2007.

[BKL06] Alina Beygelzimer, Sham Kakade, and John Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd international conference on Machine learning*, pages 97–104. ACM, 2006.

[BM98] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.

[BMP06] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics, 2006.

[BNJ03] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[BTWDR13] Richard Berendsen, Manos Tsagkias, Wouter Weerkamp, and Maarten De Rijke. Pseudo test collections for training and tuning microblog rankers. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 53–62. ACM, 2013.

[CA06] Ciprian Chelba and Alex Acero. Adaptation of maximum entropy capitalizer: Little data can help a lot. *Computer Speech & Language*, 20(4):382–399, 2006.

[CCC+99] Moses Charikar, Chandra Chekuri, To-yat Cheung, Zuo Dai, Ashish Goel, Sudipto Guha, and Ming Li. Approximation algorithms for directed steiner problems. *Journal of Algorithms*, 33(1):73–91, 1999.

[CCS10] Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[Cho10] Gobinda G Chowdhury. *Introduction to modern information retrieval*. Facet publishing, 2010.

[CKC05] Hang Cui, Min-Yen Kan, and Tat-Seng Chua. Generic soft pattern models for definitional question answering. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 384–391. ACM, 2005.

[CM12] Angel X Chang and Christopher D Manning. Sutime: A library for recognizing and normalizing time expressions. In *LREC*, pages 3735–3740, 2012.

[CPL09] IBM ILOG CPLEX. V12. 1: Users manual for cplex. *International Business Machines Corporation*, 46(53):157, 2009.

[CQT+13] Shiyu Chang, Guo-Jun Qi, Jinhui Tang, Qi Tian, Yong Rui, and Thomas S Huang. Multimedia lego: Learning structured model by probabilistic logic ontology tree. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 979–984. IEEE, 2013.

[CW08] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[DDF+90] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391, 1990.

[DIM06] Hal Daume III and Daniel Marcu. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, pages 101–126, 2006.

[DL05] Yi Ding and Xue Li. Time weight collaborative filtering. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 485–492. ACM, 2005.

[Dud76] Sahibsingh A Dudani. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, (4):325–327, 1976.

[DYXY07] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *Proceedings of the 24th international conference on Machine learning*, pages 193–200. ACM, 2007.

[DYXY08] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Self-taught clustering. In *Proceedings of the 25th international conference on Machine learning*, pages 200–207. ACM, 2008.

[EM03] Abdessamad Echihabi and Daniel Marcu. A noisy-channel approach to question answering. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 16–23. Association for Computational Linguistics, 2003.

[ER04] Günes Erkan and Dragomir R Radev. Lexpagerank: Prestige in multi-document text summarization. In *EMNLP*, volume 4, pages 365–371, 2004.

[ESV03] Cristian Estan, Stefan Savage, and George Varghese. Automatically inferring patterns of resource consumption in network traffic. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 137–148. ACM, 2003.

[FGM05] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by

gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics, 2005.

[FSSM07] Andrea Frome, Yoram Singer, Fei Sha, and Jitendra Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[GBJR08] CG González, W Bonventi Jr, and AL Vieira Rodrigues. Density of closed balls in real-valued and autometrized boolean spaces for clustering applications. In *Brazilian Symposium on Artificial Intelligence*, pages 8–22. Springer, 2008.

[Geo10] GeoVISTA. GeoVISTA. `http://www.geovista.psu.edu`, 2010.

[GIM+99] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529, 1999.

[GSSM04] Genady Grabarnik, Abdi Salahshour, Balan Subramanian, and Sheng Ma. Generic adapter logging toolkit. In *Autonomic Computing, 2004. Proceedings. International Conference on*, pages 308–309. IEEE, 2004.

[Gut84] Antonin Guttman. *R-trees: A dynamic index structure for spatial searching*, volume 14. ACM, 1984.

[HMP02] Joseph L. Hellerstein, Sheng Ma, and C-S Perng. Discovering actionable patterns in event data. *IBM Systems Journal*, 41(3):475–493, 2002.

[IBM16] IBMTivoli. IBM Tivoli Monitoring. `https://www.ibm.com/software/tivoli`, 2016.

[II08] Aminul Islam and Diana Inkpen. Semantic text similarity using corpus-based word similarity and string similarity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2(2):10, 2008.

[Inc12] E. A. Inc. WebEoc. `http://www.esi911.com/home`, 2012.

[JK02] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[Joh74] David S Johnson. Approximation algorithms for combinatorial problems. *Journal of computer and system sciences*, 9(3):256–278, 1974.

[JPL11] Yexi Jiang, Chang-Shing Perng, and Tao Li. Natural event summarization. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 765–774. ACM, 2011.

[K+13] Brian Kulis et al. Metric learning: A survey. *Foundations and Trends® in Machine Learning*, 5(4):287–364, 2013.

[Kar01] George Karypis. Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 247–254. ACM, 2001.

[KGB14] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[Kor10] Yehuda Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

[KR07] Ludmila I Kuncheva and Juan J Rodriguez. Classifier ensembles with a random linear oracle. *IEEE Transactions on Knowledge and Data Engineering*, 19(4):500–508, 2007.

[KRRS08] Srinivas Kashyap, Jeyashankher Ramamirtham, Rajeev Rastogi, and Pushpraj Shukla. Efficient constraint monitoring using adaptive thresholds. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 526–535. IEEE, 2008.

[Kul12] Brian Kulis. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364, 2012.

[L+09] Tie-Yan Liu et al. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, 2009.

[LAD+02] Victor Lavrenko, James Allan, Edward DeGuzman, Daniel LaFlamme, Veera Pollard, and Stephen Thomas. Relevance models for topic de-

tection and tracking. In *Proceedings of the second international conference on Human Language Technology Research*, pages 115–121. Morgan Kaufmann Publishers Inc., 2002.

[LC98] C Leacock and M Chodorow. Combining local context and wordnet sense similarity for word sense identification. wordnet, an electronic lexical database, 1998.

[LH03] Chin-Yew Lin and Eduard Hovy. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 71–78. Association for Computational Linguistics, 2003.

[Li15] Tao Li. *Event mining: algorithms and applications*. Chapman and Hall/CRC, 2015.

[LL14] Lei Li and Tao Li. An empirical study of ontology-based multi-document summarization in disaster management. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(2):162–171, 2014.

[LLL12a] Jingxuan Li, Lei Li, and Tao Li. Multi-document summarization via submodularity. *Applied Intelligence*, 37(3):420–430, 2012.

[LLL+12b] Chen Lin, Chun Lin, Jingxuan Li, Dingding Wang, Yang Chen, and Tao Li. Generating event storylines from microblogs. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 175–184. ACM, 2012.

[LMB+06] Yuhua Li, David McLean, Zuhair A Bandar, James D O'shea, and Keeley Crockett. Sentence similarity based on semantic nets and corpus statistics. *IEEE transactions on knowledge and data engineering*, 18(8):1138–1150, 2006.

[LMX11] Liwei Liu, Nikolay Mehandjiev, and Dong-Ling Xu. Multi-criteria service recommendation based on user criteria preferences. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 77–84. ACM, 2011.

[log16] logicMonitor. LogicMonitor. `http://www.logicmonitor.com/`, 2016.

[LWSL10] Lei Li, Dingding Wang, Chao Shen, and Tao Li. Ontology-enriched multi-document summarization in disaster management. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 819–820. ACM, 2010.

[MAMS04] Juha Makkonen, Helena Ahonen-Myka, and Marko Salmenkivi. Simple semantics in topic detection and tracking. *Information retrieval*, 7(3-4):347–368, 2004.

[Man01] Inderjeet Mani. *Automatic summarization*, volume 3. John Benjamins Publishing, 2001.

[Man16] ManageEngine. ManageEngine. `https://www.manageengine.com/`, 2016.

[MJ93] Steven McCanne and Van Jacobson. The bsd packet filter: A new architecture for user-level packet capture. In *USENIX winter*, volume 93, 1993.

[MMY+10] Gengxin Miao, Louise E Moser, Xifeng Yan, Shu Tao, Yi Chen, and Nikos Anerousis. Generative models for ticket resolution in expert networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 733–742. ACM, 2010.

[MRS+08] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[MSC+13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[MSGL09] Patricia Marcu, Larisa Shwartz, Genady Grabarnik, and David Loewenstern. Managing faults in the service delivery process of service provider coalitions. In *Services Computing, 2009. SCC'09. IEEE International Conference on*, pages 65–72. IEEE, 2009.

[NC412] NC4. E-teams. `http://www.nc4.us/ETeam.php`, 2012.

[NH10] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[NK11] Xia Ning and George Karypis. Slim: Sparse linear methods for top-n recommender systems. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 497–506. IEEE, 2011.

[NMTM00] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2):103–134, 2000.

[OSdCI11] Jesús Oliva, José Ignacio Serrano, María Dolores del Castillo, and Ángel Iglesias. Symss: A syntax-based measure for short-text semantic similarity. *Data & Knowledge Engineering*, 70(4), 2011.

[PDM11] Slav Petrov, Dipanjan Das, and Ryan McDonald. A universal part-of-speech tagset. *arXiv preprint arXiv:1104.2086*, 2011.

[PJNR13] Rahul Potharaju, Navendu Jain, and Cristina Nita-Rotaru. Juggling the jigsaw: Towards automated problem inference from network trouble tickets. In *NSDI*, pages 127–141, 2013.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[PTG+03] Chang-Shing Perng, David Thoenen, Genady Grabarnik, Sheng Ma, and Joseph Hellerstein. Data-driven validation, completion and construction of event relationship networks. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 729–734. ACM, 2003.

[PY10] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[RB03] Brian Roark and Michiel Bacchiani. Supervised and unsupervised pcfg adaptation to novel domains. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 126–133. Association for Computational Linguistics, 2003.

[RBL+07] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*, pages 759–766. ACM, 2007.

[RCD11] Alan Ritter, Colin Cherry, and William B Dolan. Data-driven response generation in social media. In *Proceedings of the conference on empirical methods in natural language processing*, pages 583–593. Association for Computational Linguistics, 2011.

[Res95] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. *arXiv preprint cmp-lg/9511007*, 1995.

[RHM02] Dragomir R Radev, Eduard Hovy, and Kathleen McKeown. Introduction to the special issue on summarization. *Computational linguistics*, 28(4):399–408, 2002.

[RJB00] Dragomir R Radev, Hongyan Jing, and Malgorzata Budzikowska. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *Proceedings of the 2000 NAACL-ANLP Workshop on Automatic summarization*, pages 21–30. Association for Computational Linguistics, 2000.

[RJST04] Dragomir R Radev, Hongyan Jing, Małgorzata Styś, and Daniel Tam. Centroid-based summarization of multiple documents. *Information Processing & Management*, 40(6):919–938, 2004.

[RLS+98] Marcus J Ranum, Kent Landfield, Mike Stolarchuk, Mark Sienkiewicz, Andrew Lambeth, and Eric Wall. Implementing a generalized tool for network monitoring. *Information Security Technical Report*, 3(4):53–64, 1998.

[Sas05] Yutaka Sasaki. Question answering as question-biased term extraction: a new approach toward multilingual qa. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 215–222. Association for Computational Linguistics, 2005.

[SBC03] Horacio Saggion, Kalina Bontcheva, and Hamish Cunningham. Robust generic and query-based summarisation. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 2*, pages 235–238. Association for Computational Linguistics, 2003.

[SCT+08] Qihong Shao, Yi Chen, Shu Tao, Xifeng Yan, and Nikos Anerousis. Easyticket: a ticket routing recommendation engine for enterprise problem resolution. *Proceedings of the VLDB Endowment*, 1(2):1436–1439, 2008.

[SGH12] Dafna Shahaf, Carlos Guestrin, and Eric Horvitz. Trains of thought: Generating information maps. In *Proceedings of the 21st international conference on World Wide Web*, pages 899–908. ACM, 2012.

[Shi00] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

[SHK+14] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[SKKR00] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, Minnesota Univ Minneapolis Dept of Computer Science, 2000.

[SL10] Chao Shen and Tao Li. Multi-document summarization via the minimum dominating set. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 984–992. Association for Computational Linguistics, 2010.

[SLD11] Chao Shen, Tao Li, and Chris HQ Ding. Integrating clustering and multi-document summarization by bi-mixture probabilistic latent semantic analysis (plsa) with sentence bases. In *AAAI*, 2011.

[SLL15] Lifeng Shang, Zhengdong Lu, and Hang Li. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364*, 2015.

[SM13] Aliaksei Severyn and Alessandro Moschitti. Automatic feature engineering for answer selection and extraction. In *EMNLP*, volume 13, pages 458–467, 2013.

[SSM02] Jun Suzuki, Yutaka Sasaki, and Eisaku Maeda. Svm answer selection for open-domain question answering. In *Proceedings of the 19th in-*

ternational conference on Computational linguistics-Volume 1, pages 1–7. Association for Computational Linguistics, 2002.

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[T+06] Pang-Ning Tan et al. *Introduction to data mining*. Pearson Education India, 2006.

[TH01] Loren Terveen and Will Hill. Beyond recommender systems: Helping people help each other. *HCI in the New Millennium*, 1(2001):487–509, 2001.

[TLP+12] Liang Tang, Tao Li, Florian Pinel, Larisa Shwartz, and Genady Grabarnik. Optimizing system monitoring configurations for non-actionable alerts. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 34–42. IEEE, 2012.

[TLS12] Liang Tang, Tao Li, and Larisa Shwartz. Discovering lag intervals for temporal dependencies. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 633–641. ACM, 2012.

[TLSG13] Liang Tang, Tao Li, Larisa Shwartz, and Genady Grabarnik. Recommending resolutions for problems identified by monitoring. In *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pages 134–142. IEEE, 2013.

[TN03] Christoph Tillmann and Hermann Ney. Word reordering and a dynamic programming beam search algorithm for statistical machine translation. *Computational linguistics*, 29(1):97–133, 2003.

[urla] HP OpenView : Network and Systems Management Products. `http://www8.hp.com/us/en/software/enterprise-software.html`.

[urlb] IBM Tivoli : Integrated Service Management. `http://ibm.com/software/tivoli/`.

[urlc] IBM Tivoli Monitoring. `http://ibm.com/software/tivoli/products/monitor/`.

[urld] ITIL. http://www.itil-officialsite.com/home/home.aspx.

[urle] Splunk: A commerical machine data managment engine. http://www.splunk.com/.

[ush12] ushahidi. ushahidi. http://www.ushahidi.com/, 2012.

[VLL+10] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.

[WE11] Bruno Wassermann and Wolfgang Emmerich. Monere: Monitoring of service compositions for failure diagnosis. *Service-Oriented Computing*, pages 344–358, 2011.

[WLLH08] Furu Wei, Wenjie Li, Qin Lu, and Yanxiang He. Query-sensitive mutual reinforcement chain and its application in query-oriented multi-document summarization. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 283–290. ACM, 2008.

[WLO12] Dingding Wang, Tao Li, and Mitsunori Ogihara. Generating pictorial storylines via minimum-weight connected dominating set approximation in multi-view graphs. In *AAAI*. Citeseer, 2012.

[WLZD08] Dingding Wang, Tao Li, Shenghuo Zhu, and Chris Ding. Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 307–314. ACM, 2008.

[WLZG11] Dingding Wang, Tao Li, Shenghuo Zhu, and Yihong Gong. ihelp: An intelligent online helpdesk system. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(1):173–182, 2011.

[WSM07] Mengqiu Wang, Noah A Smith, and Teruko Mitamura. What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP-CoNLL*, volume 7, pages 22–32, 2007.

[WSZ08] Zheng Wang, Yangqiu Song, and Changshui Zhang. Transferred dimensionality reduction. *Machine learning and knowledge discovery in databases*, pages 550–565, 2008.

[WZZ+17] Qing Wang, Wubai Zhou, Chunqiu Zeng, Tao Li, Larisa Shwartz, and Genady Ya. Grabarnik. Constructing the knowledge base for cognitive it service management. In *Services Computing (SCC), 2017 IEEE International Conference on*. IEEE, 2017.

[XZB05] Kuai Xu, Zhi-Li Zhang, and Supratik Bhattacharyya. Profiling internet backbone traffic: behavior models and applications. In *ACM SIGCOMM Computer Communication Review*, volume 35, pages 169–180. ACM, 2005.

[YCMP13] Scott Wen-tau Yih, Ming-Wei Chang, Chris Meek, and Andrzej Pastusiak. Question answering using enhanced lexical semantic models. 2013.

[YHBP14] Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632*, 2014.

[YHYY06] Xiaoxin Yin, Jiawei Han, Jiong Yang, and Philip S Yu. Efficient classification across multiple database relations: A crossmine approach. *IEEE Transactions on Knowledge and Data Engineering*, 18(6):770–783, 2006.

[Yia93] Peter N Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *SODA*, volume 93, pages 311–321, 1993.

[YPZ10] Yuhong Yan, Pascal Poizat, and Ludeng Zhao. Repair vs. recomposition for broken service compositions. In *International Conference on Service-Oriented Computing*, pages 152–166. Springer, 2010.

[YVDCBC13] Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. Answer extraction as sequence tagging with tree edit distance. In *HLT-NAACL*, pages 858–867, 2013.

[Zad04] Bianca Zadrozny. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the twenty-first international conference on Machine learning*, page 114. ACM, 2004.

[Zei12] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[zen16] zenoss. zenoss. `http://ownit.zenoss.com/`, 2016.

[Zhu04] Mu Zhu. Recall, precision and average precision. *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo*, 2004.

[Zhu05] Xiaojin Zhu. Semi-supervised learning literature survey. 2005.

[ZLSG14a] Chunqiu Zeng, Tao Li, Larisa Shwartz, and Genady Ya Grabarnik. Hierarchical multi-label classification over ticket data using contextual loss. In *Proceedings of IEEE/IFIP Network Operations and Management Symposium*, pages 1–8. IEEE, 2014.

[ZLSG14b] Chunqiu Zeng, Tao Li, Larisa Shwartz, and Genady Ya Grabarnik. Hierarchical multi-label classification over ticket data using contextual loss. In *Network Operations and Management Symposium (NOMS), 2014 IEEE*, pages 1–8. IEEE, 2014.

[ZST+13] Li Zheng, Chao Shen, Liang Tang, Chunqiu Zeng, Tao Li, Steve Luis, and Shu-Ching Chen. Data mining meets the needs of disaster information management. *IEEE Transactions on Human-Machine Systems*, 43(5):451–464, 2013.

[ZTZ+16] Wubai Zhou, Liang Tang, Chunqiu Zeng, Tao Li, Larisa Shwartz, and Genady Ya Grabarnik. Resolution recommendation for event tickets in service management. *IEEE Transactions on Network and Service Management*, 13(4):954–967, 2016.

[ZW06] Xingquan Zhu and Xindong Wu. Class noise handling for effective cost-sensitive learning by cost-guided iterative classification filtering. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1435–1440, 2006.

[ZZL+17] Chunqiu Zeng, Wubai Zhou, Tao Li, Larisa Shwartz, and Genady Y Grabarnik. Knowledge guided hierarchical multi-label classification over ticket data. *IEEE Transactions on Network and Service Management*, 2017.

VITA

WUBAI ZHOU

2012-Present	M.S. and Ph.D., Computer Science Florida International University, Miami, Florida
2008-2012	B.S., Computer Science Wuhan University, Wuhan, P.R. China

PUBLICATIONS

Wubai Zhou, Wei Xue, Ramesh Baral, Qing Wang, Chunqiu Zeng, Tao Li, Jian Xu, Zhen Liu, Larisa Shwartz, Genady Ya. Grabarnik. *STAR: A System for Ticket Analysis and Resolution.* In *Proceedings of the 23th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2017.

Qing Wang, Wubai Zhou, Chunqiu Zeng, Tao Li, Larisa Shwartz, and Genady Ya. Grabarnik. 2017. *Constructing the Knowledge Base for Cognitive IT Service Management.* In *Services Computing (SCC), 2017 IEEE International Conference.*

Tao Li, Chunqiu Zeng, Yexi Jiang, Wubai Zhou, Liang Tang, Zheng Liu, Yue Huang, *Datadriven Techniques in Computing System Management*, *ACM Computing Surveys*, 2017.

Chunqiu Zeng, Wubai Zhou, Tao Li, Larisa Shwartz, and Genady Y Grabarnik. 2017. *Knowledge Guided Hierarchical Multi-Label Classication over Ticket Data. IEEE Transactions on Network and Service Management*, 2017.

Wubai Zhou , Liang Tang, Chunqiu Zeng, Tao Li, Larisa Shwartz, Genady Ya.Grabarnik. *Resolution Recommendation for Event Tickets in Service Management*, in *IEEE Transactions on Network and Service Management(TNSM)*, 2016.

Chunqiu Zeng, Liang Tang, Wubai Zhou, Tao Li, Larisa Shwartz, Genady Ya.Grabarnik. *An Integrated Framework for Mining Temporal Logs from Fluctuating Events*, in *IEEE Transactions on Services Computing(TSC)*, 2016.

Tao Li, Wubai Zhou, Chunqiu Zeng, Qing Wang, Qifeng Zhou, Dingding Wang, Yue

Huang, Jia Xu, Wentao Wang, Minjing Zhang, Steve Luis, Shu-Ching Chen, Naphtali Rishe. *DI-DAP: An Efficient Disaster Information Delivery and Analysis Platform in Disaster Management*, in *Proceedings of the 25th ACM Conference on Information and Knowledge Management(CIKM)*, 2016.

Wubai Zhou, Tao Li, Larisa Shwartz, and Genady Ya Grabarnik. *Recommending ticket resolution using feature adaptation.* In *Network and Service Management (CNSM), 2015 11th International Conference on, pp. 15-21. IEEE*, 2015.

Wubai Zhou, Liang Tang, Tao Li, Larisa Shwartz, Genady Ya. Grabarnik. *Resolution recommendation for event tickets in service management.* In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 287-295. IEEE, 2015.

Wubai Zhou, Chao Shen, Tao Li, Shu-Ching Chen, and Ning Xie. *Generating textual storyline to improve situation awareness in disaster management.* In *Information Reuse and Integration (IRI), 2014 IEEE 15th International Conference* on, pp. 585-592. IEEE, 2014.

Li Zheng, Chunqiu Zeng, Lei Li, Yexi Jiang, Wei Xue, Jingxuan Li, Chao Shen, Wubai Zhou, Hongtai Li, Liang Tang, Tao Li, Bing Duan, Ming Lei, Pengnian Wang. *Applying data mining techniques to address critical process optimization needs in advanced manufacturing.* In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1739-1748. ACM, 2014.

Chunqiu Zeng, Yexi Jiang, Li Zheng, Jingxuan Li, Lei Li, Hongtai Li, Chao Shen, Wubai Zhou, Tao Li, Bing Duan, Ming Lei, Pengnian Wang. *Fiu-miner: a fast, integrated, and user-friendly system for data mining in distributed environment.* In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1506-1509. ACM, 2013.