Resolution Recommendation for Event Tickets in Service Management

Wubai Zhou, Liang Tang and Tao Li
School of Computer Science
Florida International University
Miami, FL, USA
Email:{wzhou005,ltang002, taoli}@cs.fiu.edu

Larisa Shwartz
Operational Innovations
IBM T.J. Watson Research Center
Yorktown Heights, NY, USA
Email: lshwart@us.ibm.com

Genady Ya. Grabarnik
Dept. of Math & Computer Science
St. John's University
Queens, NY, USA
Email:grabarng@stjohns.edu

***Abstract*—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 paper 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 paper, 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.

***Index Terms*—IT Service Management, Recommender System, LDA (Latent Dirichlet Allocation), Metric Learning**

I. 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 [1], [2], [3], [4], [5]. Automatic problem detection is typically realized by system monitoring software, such as IBM Tivoli Monitoring [6] and HP OpenView [7]. 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 paper 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 [8] 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 [8] only considers the event information, and ignores the related resolutions. In addition, each attribute is

978-3-901882-76-0 @2015 IFIP

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 paper, 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 paper is organized as follows: Section II briefly introduces the workflow of the infrastructure management of an automated service and shares our observations on three sets of monitoring tickets. In Section III, we present resolution recommendation algorithms for monitoring tickets. Section IV discusses some detailed implementation issues. In Section V, we present experimental studies on real monitoring tickets. Section VI describes related work about service infrastructure management and recommendation systems. Finally, Section VII concludes our paper and discusses our future work.

II. 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.

A. 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 [9]. 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 paper, we consider tickets generated by a service management system (see Figure 1).

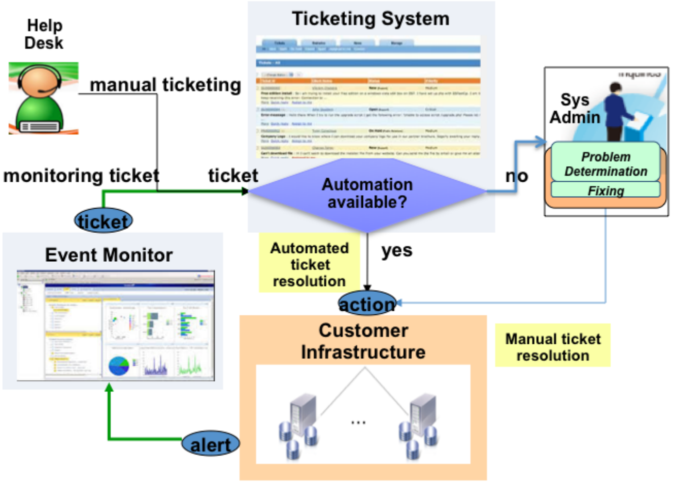


Fig. 1: Service Management System

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 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.

B. 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.

TABLE I: Data Summary

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

The collected ticket sets from the three accounts are denoted by “account1”, “account2” and “account3”, respectively. Table I summarizes the three data sets. Figure 2 shows the numbers

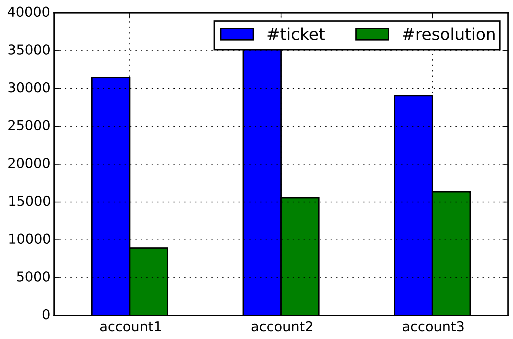


Fig. 2: Numbers of Tickets and Distinct Resolutions

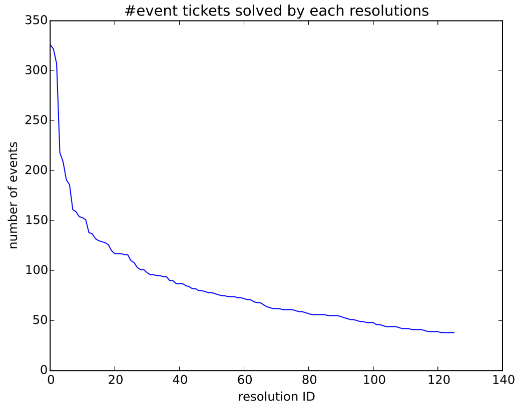


Fig. 3: number of monitoring tickets resolved by each resolutions denoted by "resolution ID" in account1

of tickets and distinct resolutions and Figure 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.

III. Resolution Recommendation

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

A. *Workflow*

Figure 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 [8] and the last three algorithms are proposed in this paper. Figure 4 clearly illustrates the differences among these four recommendation methods. The details of the three proposed algorithms will be described in detail in Section III-C, Section III-D, and Section III-E, respectively.

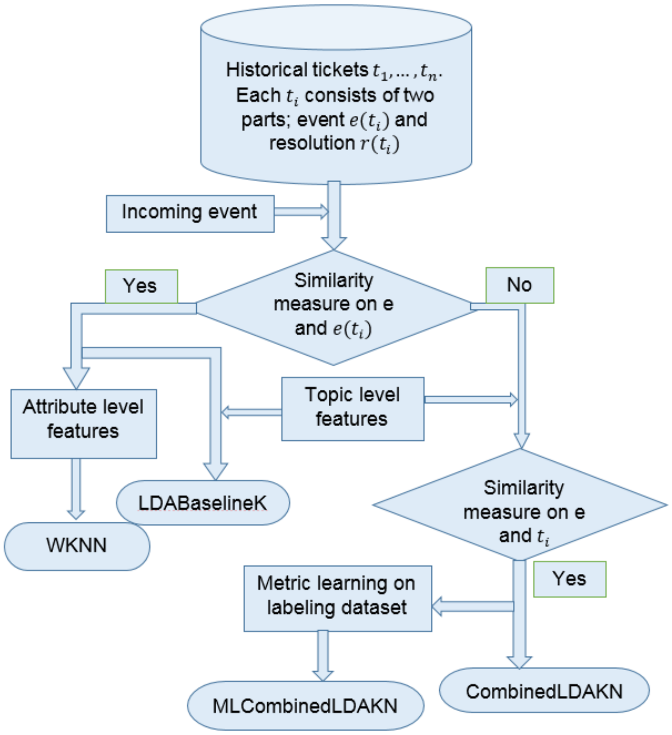

Historical tickets $t_1, \ldots, t_n$. Each t_i consists of two parts; event $e(t_i)$ and resolution $r(t_i)$
Incoming event
Yes
Similarity measure on e and $e(t_i)$
No
Attribute level features
Topic level features
LDABaselineK
WKNN
Similarity measure on e and t_i
Metric learning on labeling dataset
Yes
MLCombinedLDAKN
CombinedLDAKN

Fig. 4: Algorithms' workflow

B. *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 parameter), and recommends the top $k \leq K$ representative resolutions among them [10], [11]. The nearest neighbors are indicated by similarities of the associated events of the tickets. In this paper, the representativeness is measured by the number of occurrences in the K neighbors.

TABLE II: Notations

Notation	Description
D	Set of historical tickets
$\lvert \cdot \rvert$	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 II lists the notations used in this paper. 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 attributes. There are three types of event attributes: categorical, numeric and textual (shown by Table III). 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

TABLE III: Event Attribute Types

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

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

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* [12], 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

$$sim(e_1, e_2) = \frac{\sum_{a \in A(e_1) \cap A(e_2)} sim_a(e_1, e_2)}{|A(e_1) \cup A(e_2)|}. \tag{1}$$

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.

C. Representation of Monitoring Tickets

As shown in Section III-B, 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 IV 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 paper, we propose to apply Latent Dirichlet Allocation [13](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 [13]. Figure 5 shows the graphical model representation of LDA.

The w_{ij}'s are the only observable variables. Following [13], 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:

$$\prod_{d=1}^{M} \int p(\theta_d|\alpha) \Big(\prod_{n=1}^{N_d} \sum_{z_{d_n}} p(z_{d_n|\theta_d}) p(w_{d_n})|z_{d_n}, \beta \Big) d\theta_d \tag{2}$$

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 [13]. Topic probability distribution of a document is commonly used as its feature vector.

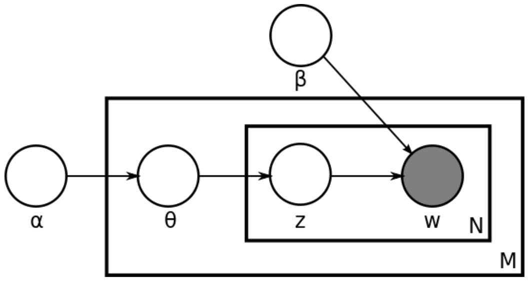


Fig. 5: Plate notation representing the LDA model. α is the parameter of the Dirichlet prior on the per-document topic distribution; β is the parameter of the Dirichlet prior on the per-topic word distribution; θ_i is the topic distribution for document i; ϕ_k is the word distribution for topic k; z_{ij} is the topic for the j-th word in document i, and w_{ij} is the specific word.

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 in Section V demonstrate that the algorithm performance based on topic level features is better than that on attribute level features.

D. 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 IV 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.

TABLE IV: Tickets for explaining motivation of incorporating resolution information

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.

In Section III-B, $sim(e, e(t_i))$ 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 in Section V demonstrate the effectiveness of this proposed approach.

E. 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 V 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 [14] to achieve this goal.

TABLE V: First 6 words are extracted to represent topics trained from LDA

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

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 [15]. 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:

$$d_A(x, y) = (x - y)^T A(x - y), \tag{3}$$

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 [14]:

$$d_A(x, y) = x^T Ay. \tag{4}$$

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 pre-calculate 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:

$$\begin{aligned} f(A) &= min \sum_{i=1}^{n} \sum_{j=1}^{n} ||R_{i,j} - S_A(\vec{t_i}, \vec{t_j})||^2 \\ &= min||R - SAS^T||^2, \end{aligned} \tag{5}$$

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 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 [16] 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 5:

$$\begin{aligned}\frac{\partial f(A)}{\partial A} &= \frac{\partial((R-SAS^T)^T(R-SAS^T))}{\partial A} \\ &= 2S^TSAS^TS - 2S^TAS \end{aligned} \tag{6}$$

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

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

IV. Implementation

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

A. 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 III-B. 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.

B. 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 [17], R-Tree [18], VP-Tree [19], cover tree [20]. But the search space of our monitoring tickets is not metric and the dimensionality is high. Therefore, locality sensitive hashing [21] 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.

V. Evaluation

A. Implementation

We implemented four algorithms: Weighted KNN [22] 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.

B. Experimental Data

Experimental monitoring tickets are collected from three accounts managed by IBM Global Services, denoted later "account1", "account2" and "account3". The monitoring events are captured by IBM Tivoli Monitoring [23]. The ticket sets are summarized in Table I. To evaluate metric learning, 1000 labeled tickets with resolution categories are obtained from "account1". Table VI shows three sample categories of resolutions [24].

TABLE VI: Three resolution types with the event description they resolved

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

C. Evaluation Metric

The following evaluation measures are used in our experiments.

1) 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 $sim(r_{io}, r_j)$.

2) mean average precision: Mean Average Precision (MAP) [25] 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.

D. Choosing the Number of Topics

Figure 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.

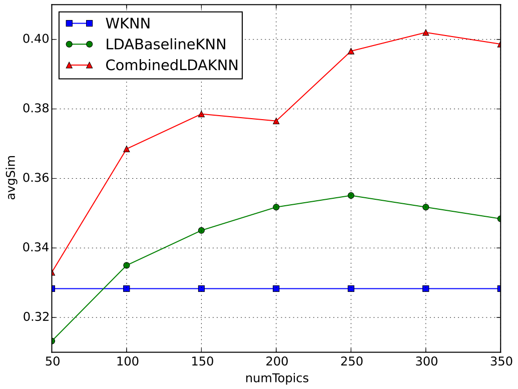


Fig. 6: Accuracy varies for different $numTopics$ for dataset "account1"

E. 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.

To compare the results of each algorithm, we vary the number of recommended resolutions, k. Figures 7, and 8 show the *average similarity* scores by setting $k = 1, 3, 5, 7$ separately, with $K = 8$ and $K = 16$. As shown by Figure 7 and Figure 8, 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.

1) Metric Learning Performance: Figure 9, Figure 10 and Figure 11 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 Figure 9 and Figure 10, similarity scores between monitoring tickets with resolutions from the same category will be enhanced while similarity scores between monitoring 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.

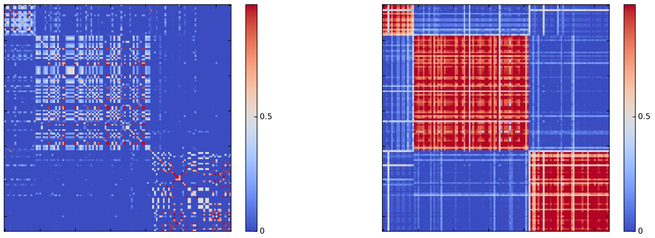


Fig. 9: similarity measure before and after metric learning for training set

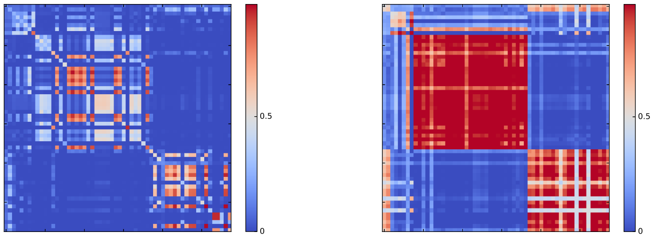
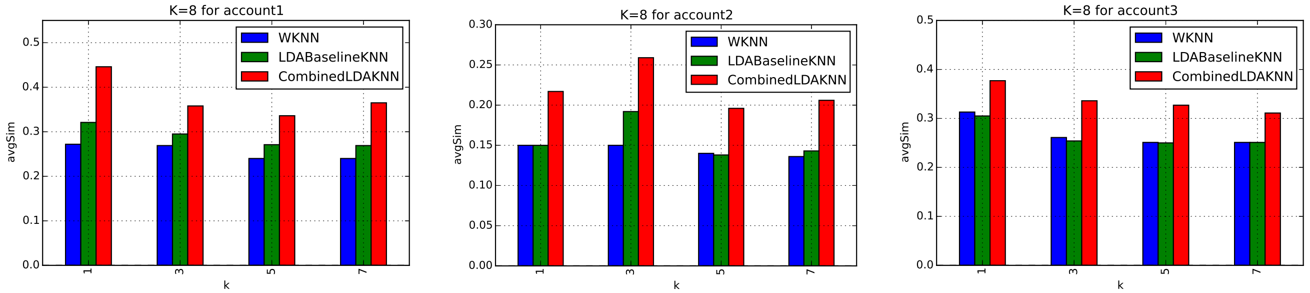


Fig. 10: Similarity measure before and after metric learning for testing set

Figure 11 uses *MAP* to evaluate the performance of "CombinedLDAKNN" and "MLCombinedLDAKNN". As shown in Figure 11, overall MAP scores of "MLCombinedLDAKNN" 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.

VI. Related Work

This section reviews prior research studies related to the automated IT service management and the recommendation

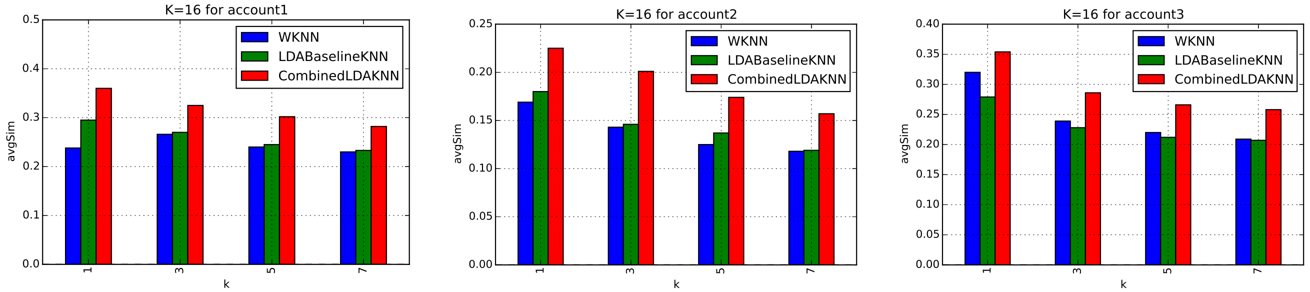


(a) account1

(b) account2

(c) account3

Fig. 7: Test Results for three accounts by varying k for $K = 8$.



(a) account1

(b) account2

(c) account3

Fig. 8: Test Results for three accounts by varying k for $K = 16$.

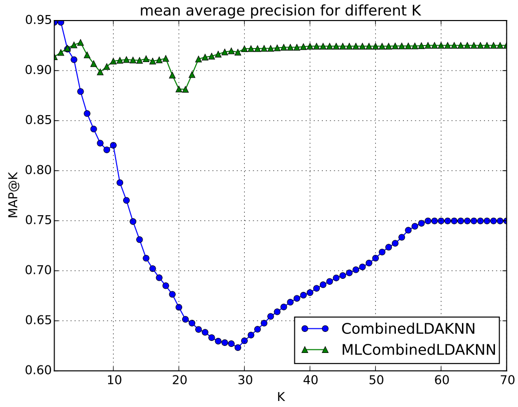


Fig. 11: Mean average precision (MAP) varying parameter K of underlying KNN algorithm

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. There are many commercial products, such as IBM Tivoli [23], HP OpenView [7] and Splunk [26] 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 [27], [28]. Other works propose methods for refining the work order of resolving tickets [27], [29], [30] and discovering the dependency of tickets [31].

VII. Conclusion

This paper 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 repeated 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 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 [30], [32]. 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.

Acknowledgment

The work of W. Zhou, L. Tang and T. Li is partially supported by the National Science Foundation under grants DBI-0850203, CNS-1126619, and IIS-1213026 and the U.S. Department of Homeland Security's VACCINE Center under Award Number 2009-ST-061-CI0001

References

[1] P. Marcu, L. Shwartz, G. Grabarnik, and D. Loewenstern, "Managing faults in the service delivery process of service provider coalitions," in *IEEE SCC*, 2009, pp. 65–72.

[2] L. Tang, T. Li, F. Pinel, L. Shwartz, and G. Grabarnik, "Optimizing system monitoring configurations for non-actionable alerts," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2012.

[3] N. Ayachitula, M. J. Buco, Y. Diao, M. Surendra, R. Pavuluri, L. Shwartz, and C. Ward, "IT service management automation - a hybrid methodology to integrate and orchestrate collaborative human centric and automation centric workflows," in *IEEE SCC*, 2007, pp. 574–581.

[4] B. Wassermann and W. Emmerich, "Monere: Monitoring of service compositions for failure diagnosis," in *ICSOC*, 2011, pp. 344–358.

[5] Y. Yan, P. Poizat, and L. Zhao, "Repair vs. recomposition for broken service compositions," in *ICSOC*, 2010, pp. 152–166.

[6] "IBM Tivoli Monitoring," http://ibm.com/software/tivoli/products/monitor/.

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

[8] L. Tang, T. Li, L. Shwartz, and G. Grabarnik, "Recommending resolutions for problems identified by monitoring," in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*. IEEE, 2013, pp. 134–142.

[9] "ITIL," http://www.itil-officialsite.com/home/home.aspx.

[10] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl, "Application of dimensionality reduction in recommender system – a case study," in *ACM WebKDD Workshop*, 2000.

[11] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Addison Wesley, 2005.

[12] G. Salton and M. McGill, *Introduction to Modern Information Retrieval*. McGraw-Hill, 1984.

[13] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

[14] B. Kulis, "Metric learning: A survey," *Foundations & Trends in Machine Learning*, vol. 5, no. 4, pp. 287–364, 2012.

[15] A. Frome, Y. Singer, F. Sha, and J. 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*. IEEE, 2007, pp. 1–8.

[16] J.-F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 1956–1982, 2010.

[17] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[18] A. Guttman, "R-Trees: A dynamic index structure for spatial searching," in *ACM SIGMOD*, 1984, pp. 47–57.

[19] P. N. Yianilos, "Data structures and algorithms for nearest neighbor search in general metric spaces," in *SODA*, 1993, pp. 311–321.

[20] A. Beygelzimer, S. Kakade, and J. Langford, "Cover trees for nearest neighbor," in *ICML*, 2006, pp. 97–104.

[21] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proceedings of VLDB*, Edinburgh, Scotland, UK, September 1999, pp. 518–529.

[22] S. A. Dudani, "The distance-weighted k-nearest-neighbor rule," *IEEE Transactions on Systems Man and Cybernetics*, vol. SMC-6, no. 4, pp. 325–327, april 1976.

[23] "IBM Tivoli : Integrated Service Management," http://ibm.com/software/tivoli/.

[24] J. Bogojeska, I. Giurgiu, D. Lanyi, G. Stark, and D. 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*. IEEE, 2014, pp. 1–9.

[25] M. Zhu, "Recall, precision and average precision," *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo*, 2004.

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

[27] Q. Shao, Y. Chen, S. Tao, X. Yan, and N. Anerousis, "EasyTicket: a ticket routing recommendation engine for enterprise problem resolution," *PVLDB*, vol. 1, no. 2, pp. 1436–1439, 2008.

[28] D. Wang, T. Li, S. Zhu, and Y. Gong, "iHelp: An intelligent online helpdesk system," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 41, no. 1, pp. 173–182, 2011.

[29] G. Miao, L. E. Moser, X. Yan, S. Tao, Y. Chen, and N. Anerousis, "Generative models for ticket resolution in expert networks," in *KDD*, 2010, pp. 733–742.

[30] C. Zeng, T. Li, L. Shwartz, and G. Y. Grabarnik, "Hierarchical multi-label classification over ticket data using contextual loss," in *Proceedings of IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2014, pp. 1–8.

[31] L. Tang, T. Li, and L. Shwartz, "Discovering lag intervals for temporal dependencies," in *ACM KDD*, 2012, pp. 633–641.

[32] S. Chang, G.-J. Qi, J. Tang, Q. Tian, Y. Rui, and T. S. Huang, "Multimedia lego: Learning structured model by probabilistic logic ontology tree," in *Data Mining (ICDM), 2013 IEEE 13th International Conference on*. IEEE, 2013, pp. 979–984.