STAR: A System for Ticket Analysis and Resolution

Wubai Zhou, Wei Xue
Computer Science
Florida International University
Miami, USA
{wzhou005,wxue004}@cs.fiu.edu

Ramesh Baral
Computer Science
Florida International University
Miami, USA
rbara012@cs.fiu.edu

Qing Wang
Computer Science
Florida International University
Miami, USA
qwang028@cs.fiu.edu

Chunqiu Zeng
Computer Science
Florida International University
Miami, USA
czeng001@cs.fiu.edu

Tao Li
Florida International University
Nanjing University of Posts and Telecommunications
taoli@cs.fiu.edu

Jian Xu
Computer Science & Engineering
Nanjing University of Science and Technology
dolphin.xu@njust.edu.cn

Zheng Liu
Nanjing University of Posts and Telecommunications
zliu@njupt.edu.cn

Larisa Shwartz
IBM T.J. Watson Research Center
New York, USA
lshwart@us.ibm.com

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

ABSTRACT

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 paper, 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.

KDD '17, August 13-17, 2017, Halifax, NS, Canada

ACM ISBN 978-1-4503-4887-4/17/08…$15.00
https://doi.org/10.1145/3097983.3098190

CCS CONCEPTS

• **Information systems** → **Learning to rank**; *Language models*; *Similarity measures*; • **Networks** → *Network monitoring*; • **Computing methodologies** → *Machine translation*;

KEYWORDS

IT Service Management, Ticket Resolution, Sentence Model, Convolutional Neural Network

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 procedures (for instance, problem detection, determination and resolution) for the service infrastructure [4, 19, 34]. The automated problem detection has been realized by some system monitoring softwares, such as HP OpenView [1] and IBM Tivoli Monitoring [2]. 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 [3] and is illustrated in Fig. 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

[1] http://www8.hp.com/us/en/software/enterprise-software.html
[2] http://ibm.com/software/tivoli/
[3] http://www.itil-officialsite.com/home/home.aspx

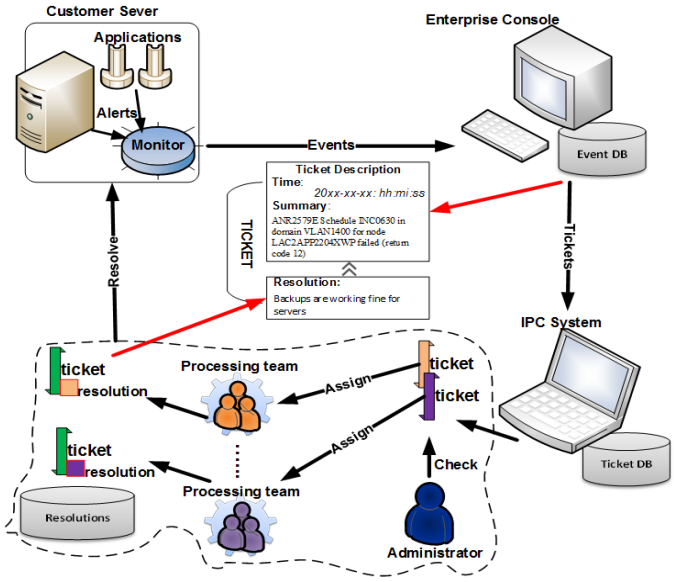


Figure 1: Information Technology Infrastructure Library (ITIL) Service Management System

or knowledge-based engines to analyze the events and determines whether or not to create an incident ticket in the IPC system [16].

Each ticket is stored as a database record that consists of several related attributes (see Table 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 large IT service providers that handle many tickets every day. This is one of the major motivations behind the automated analysis of ticket resolution.

Table 1: A sample ticket

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

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 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. Based on our preliminary studies [36, 39], 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 [39] demonstrated that not all of the resolutions are equally worthy. For example, as shown in Table 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 [39], 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.

1.2 Related Research

There have been some studies that adopted data mining techniques to facilitate IT service management. Most of them have focused on system behaviors. The text mining technique has been used to capture the system events from the raw textual logs [32]. In [37], the authors have analyzed the historical events to better understand system behaviors. The analysis of the system log files and monitoring events [10, 24], the identification of actionable patterns of events and misses, or false negatives, by the monitoring system [39] have been mainly geared towards the study of system events.

Table 2: Illustration of ticket samples from an account. Only ticket summary and resolution are displayed for the sake of simplicity

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

Only a few recent studies have focused on ticket resolution [39] . They have adapted the techniques, such as n-gram, Jaccard similarity and LDA [3, 6], 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 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 paper 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.

1.3 Road Map

The rest of this paper is organized as follows. Section 2 gives an overview of framework. Section 3 describes the pre-process on tickets and the features used to train the model for quantifying the quality of ticket resolution. In Section 4, we introduce our proposed deep neural ranking model. Automation of resolution recommendation is studied in Section 5 and ticket clustering and classification are evaluated in 6. Finally, Section 7 concludes the paper.

2 OVERVIEW

In this section, we provide a high-level description of the system. As illustrated in Figure 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.

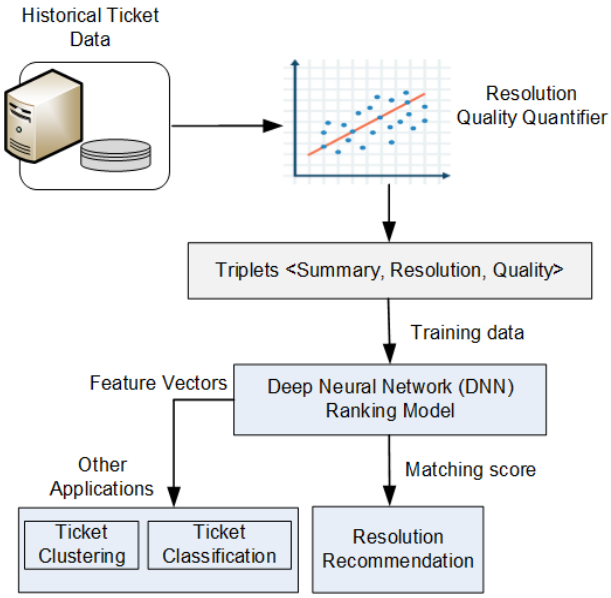


Figure 2: Overview of the proposed system

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 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 [39], 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.

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.

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) [25] 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 [5]. 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 [26] and defined the corresponding PoS tag pattern, as shown in Table 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.

3.2 Findings

We evaluated three of the most popular regression models (logistic regression, gradient boosting tree and random forest [3]) 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 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 semantic features have predominant advantages in characterizing and quantifying the ticket resolution quality over the other features.

4 DEEP NEURAL RANKING MODEL

In our preliminary work [39], 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 3, we can definitely improve the automating ticket resolution task by considering the quality of resolutions.

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 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 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:

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

Table 3: PoS tag pattern for concepts *problem, action.* NP refers to noun phrase derived from the PoS tag sequence for each 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 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.

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

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* [18].

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.

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* [13] 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.

4.2.1 Sentence Model. The architecture of our CNN-based model is shown in Fig 3. It is inspired by the CNN model for performing various sentence classifications [13].

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 $\mathbf{c}$:

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

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 componentwise. 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 [21] to build rich feature representations of the input.

4.2.2 Architecture for ranking ticket summary and resolution pairs. The partial network architecture introduced in Section 4.2.1 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. 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:

$$sim(x_s, x_r) = x_s^T M x_r \tag{2}$$

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 [8]. It can also be viewed as a process of learning similarity metric on two vectors drawing from different feature spaces [14]. 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.

4.2.3 Objective Function. The model is trained to minimize the binary cross-function:

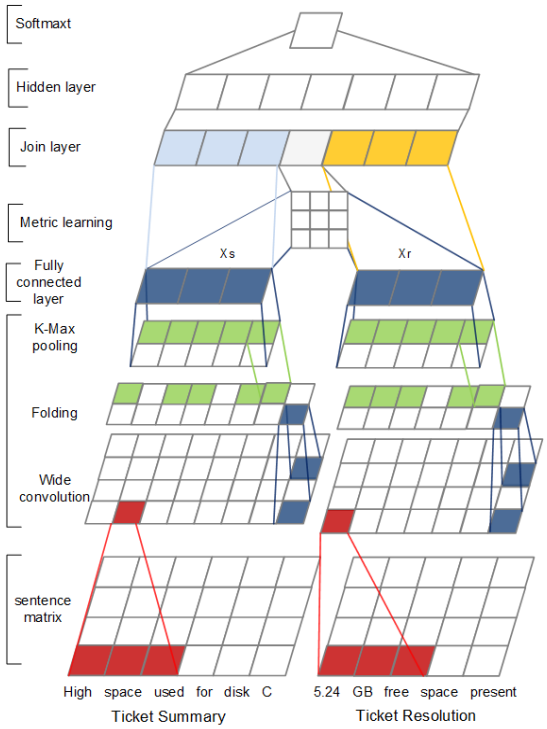


Figure 3: Ranking Model. The character level embedding is not shown for the sake of saving space.

$$\begin{aligned} L &= -\log \Pi_{i=1}^{N} p(y_i | \mathbf{s_i}, \mathbf{r_i}) \\ &= -\sum_{i=1}^{N} [y_i \log a_i + (1 - y_i) \log(1 - a_i)] \end{aligned} \tag{3}$$

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 [35] with the gradients computed by back propagation algorithm.

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 [30] is employed to prevent feature co-adaptation by setting to zero (dropping out) a portion of hidden units during the forward phase.

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 [23]. 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.

5 AUTOMATING TICKET RESOLUTION

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

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 5.

Table 5: Ticket dataset summary.

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

5.2 Ticket Resolution Automation

5.2.1 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) [40], and normalized discounted cumulative gain (nDCG) [12]. 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.

5.2.2 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 [33] to enable them to search for more than one response.

• Statistical Machine Translation (SMT): SMT [28] 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) [31].

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 [39] 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.

Table 6: Overall performance comparison.

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**

5.2.3 Results. Overall performance results are shown in Table 6. We have some interesting observations. The performance of the generative methods is quite moderate, which concurs the judgment from [29]. The automatic resolution generators tend to produce 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 OTHER TICKET ANALYSIS APPLICATIONS

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 5 for details) illustrate the effectiveness of the learned vector representation.

6.1 Ticket Clustering

In IT service management, ticket clustering is important for optimal ticket dispatching [7] 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 7 for their formula).

The $F1$ scores [2] obtained for evaluating the ticket clustering task over different measures are illustrated in Table 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}).

Table 8: Comparisons of F1 scores using different similarity measures.

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**

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 [38].

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.

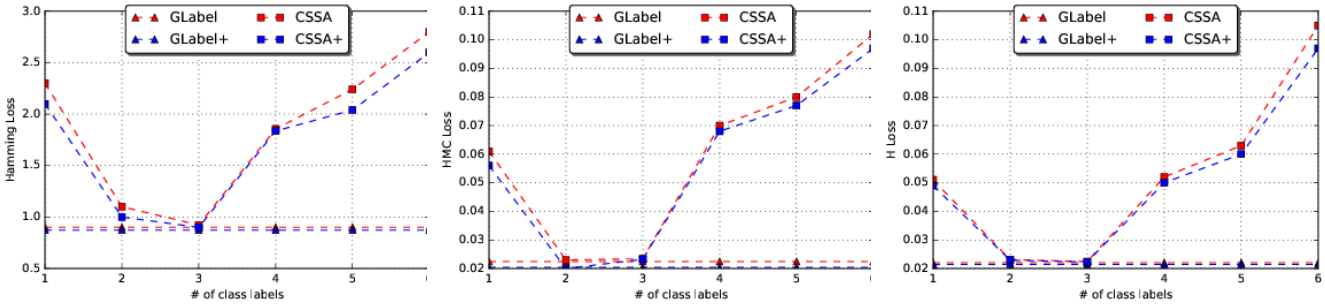
6.2.1 Evaluation Metrics. In order to illustrate the effectiveness of our model, we introduced several metrics to evaluate the hierarchical multi-label classification problem including Hamming-loss, H-loss and HMC-loss [38].

- 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

6.2.2 Algorithms for Comparison. We perform the experiments over the same setup as the previous study [38], 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.

Table 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.

Category	Measure	Formula	Note
Surface Matching Similarity	Jaccard [9]	$S_{JAC}(T_1, T_2) = \frac{\|A \cap B\|}{\|A \cup B\|}$	A and B be sets of words in two ticket descriptions
	N-word overlap [1]	$S_{nwo}(T_1, T_2) = \tanh(\frac{overlap_{phrase}(T_1, T_2)}{n_1 + n_2})$	A phrasal n-word overlap
	NLCS [11]	$S_{NLCS}(T_1, T_2) = \frac{(\|LCS(T_1, T_2)\|)^2}{\|T_1\| \times \|T_2\|}$	Considering the length of both the shorter and the longer string
Semantic Similarity	Leacock & Chodorow [15]	$S_{lch}(c_1, c_2) = -\frac{\log len(c_1, c_2)}{2 \times D}$	Path-based method using wordnet
	RES [27]	$S_{res}(c_1, c_2) = IC(lcs(c_1, c_2))$	Information content-based method
	Word2vec-based [20]	$S_{w2v}(w_1, w_2)$	Word2vec based word similarity using wikipedia
Hybrid Similarity	ISLAM's measure [11]	$S_{STS}(T_1, T_2) = \frac{(\delta(1 - w_f + w_f S_o) + \sum_{i=1}^{\rho} \rho_i) \times (m+n)}{2mn}$	Combining string similarity, semantic similarity and common-word order similarity
	Li's measure [17]	$S_{SSI}(T_1, T_2) = \delta \frac{s_1 \cdot s_2}{\|s_1\| \cdot \|s_2\|} + (1 - \delta) \frac{\|r_1 - r_2\|}{\|r_1 + r_2\|}$	Considering semantic similarity and word-order similarity
	SyMSS [22]	$S_{symss}(T_1, T_2) = \frac{1}{n} \sum_{i=1}^{n} sim(h_{1i}, h_{2i}) - l \times PF$	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



(a) The lowest Hamming loss: *GLabel* gets 0.901 and *GLabel+* 0.872; *CSSA* gets 0.923 and *CSSA+* 0.901.

(b) The lowest HMC-Loss: *GLabel* gets 0.022 and *GLabel+* 0.020; *CSSA* gets 0.023 and *CSSA+* 0.023.

(c) The lowest H-Loss: *GLabel* gets 0.022 and *GLabel+* 0.021; *CSSA* gets 0.023 and *CSSA+* 0.21.

Figure 4: Experiments involving tickets in terms of Hamming Loss, HMC-Loss and H-Loss.

The performance comparison is shown in Fig. 4. *GLabel+* and *CSSA+* outperformed their counterparts (*GLabel* and *CSSA*) which indicates the effectiveness for our derived feature representation.

7 CONCLUSION

In this paper, 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.

ACKNOWLEDGMENTS

The work was supported in part by the National Science Foundation under Grant Nos. IIS-1213026, CNS-1126619, and CNS-1461926, Chinese National Natural Science Foundation under grant 91646116, Ministry of Education/China Mobile joint research grant under Project No.5-10, and an FIU Dissertation Year Fellowship

REFERENCES

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

[2] Elke Achtert, Sascha Goldhofer, Hans-Peter Kriegel, Erich Schubert, and Arthur Zimek. 2012. Evaluation of Clusterings–Metrics and Visual Support. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*. IEEE, 1285–1288.

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

[4] Naga Ayachitula, Melissa Buco, Yixin Diao, Surendra Maheswaran, Raju Pavuluri, Larisa Shwartz, and Chris Ward. 2007. 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, 574–581.

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

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

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

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

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

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

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

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

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

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

[15] C Leacock and M Chodorow. 1998. Combining local context and WordNet sense similarity for word sense identification. WordNet, An Electronic Lexical Database. (1998).

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

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

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

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

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

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

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

[23] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global Vectors for Word Representation.. In *EMNLP*, Vol. 14. 1532–1543.

[24] Chang-Shing Perng, David Thoenen, Genady Grabarnik, Sheng Ma, and Joseph Hellerstein. 2003. 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.* ACM, 729–734.

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

[26] Rahul Potharaju, Navendu Jain, and Cristina Nita-Rotaru. 2013. Juggling the Jigsaw: Towards Automated Problem Inference from Network Trouble Tickets.. In *NSDI.* 127–141.

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

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

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

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

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

[32] Liang Tang, Tao Li, and Chang-Shing Perng. 2011. LogSig: Generating system events from raw textual logs. In *Proceedings of the 20th ACM international conference on Information and knowledge management.* ACM, 785–794.

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

[34] 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 on.* IEEE.

[35] Matthew D Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* (2012).

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

[37] Chunqiu Zeng, Liang Tang, Wubai Zhou, Tao Li, Larisa Shwartz, Genady Grabarnik, et al. 2016. An integrated framework for mining temporal logs from fluctuating events. *IEEE Transactions on Services Computing* (2016).

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

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

[40] Mu Zhu. 2004. Recall, precision and average precision. *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo* 2 (2004), 30.