Convolution Neural Network-Based Expert Recommendation for Alert Processing

Haidong Huang[1], Liming Wang[1], Rao Fu[2*], Jing Yu[1], Ding Yuan[2] and Danqi Li[2]
[1]*State Grid Jiangsu Electric Power CO., LTD. Nanjing, China*
[2]*Xuzhou Supply Company, State Grid Jiangsu Electric Power CO., LTD. Xuzhou, China*

Keywords: Trouble Ticket, Expert Recommendation, Expert Profile, Convolution Neural Network.

Abstract: Alerts and associated trouble tickets provide extremely useful information for IT system maintenance. However, the continuously occurrence of thousands of tickets also leads to a big challenge for accurately and effectively dispatching them to skilled experts for quick problem-solving. To cope with such a challenge, this paper develops a convolution neural network-based expert recommendation approach for in-time alert processing. First, expert profile is built by extracting domain words from historical tickets, and is encoded into a sentence like problem description. Second, an attention-based convolution neural network (CNN) is developed not only to learn a unified representation for both problem description and expert profile but also measure the semantic similarity between them. Finally, an ordered expert list is outputted. We evaluate our approach on a real-world data set. Experimental results show that compared to the best baseline approach, our approach can not only improve 2.8% in terms of p@1 but also shorten 11.7% in terms of the mean number of steps to resolve (MSTR).

1 INTRODUCTION

Trouble tickets play a very important role in the complex IT system maintenance. When an event happens, or when special situations, errors, even faults occur during the IT service consumption, a trouble ticket, also called issue ticket, is generated, which records the detailed problem symptom. And then, the ticket management employed by the IT system automatically dispatches the ticket to domain experts for problem-solving. Once the problem is fixed, the ticket is closed. System maintenance staffs always attempt to quickly bring an abnormal service back to normal by assigning skilled and well-matched experts using the deployed expert recommendation module. However, several real-world situations, such as diverse troubles, vague problem descriptions written in a natural language way, huge size of tickets and low efficiency of manually assigning experts, pose great challenges on the expert recommendation module to avoid violating the signed Service Level Agreement (SLA) with users. Therefore, rapid problem-solving strongly depends on efficiently and accurately expert recommendation, which motivates us to focus on expert recommendation for trouble tickets.

Typically, a trouble ticket contains at least five fields, including ‘problem description’, ‘problem type’, ‘expert name/id’, ‘resolution’ and ‘status’, as shown in Figure 1. The problem description presents the detailed symptom occurring at system runtime. Each ticket is assigned to a specific problem type belonging to the problem category. The expert name is used to identify a specific expert who attempts to solve the problem. The resolution records the detailed approaches to fix the problem. The status is setting to ‘closed’ if the problem has been fixed, otherwise it is setting to ‘open’. The ticket resolving process is regarded as a ticket delivery sequence starting from an initial expert to the final resolver. Initially, an incoming ticket was assigned to an expert. If the problem is fixed, the ticket is closed. Otherwise, the ticket dispatching system delivers the ticket to another expert. Such a delivery process is repeated until the ticket is closed. The last expert who resolved it is called a resolver.

Although a few studies (Shao Q, Shao Q, Agarwal S, Botezatu M. M., Xu J, Zhou W) have been reported to deal with ticket dispatching or expert recommendation, there are still many limitations in existing methods that deserve further investigation. The traditional machine learning technology, such as logistic regression (LR)(R. Qamili), decision tree

Huang, H., Wang, L., Fu, R., Yu, J., Yuan, D. and Li, D.
Convolution Neural Network-Based Expert Recommendation for Alert Processing.
DOI: 10.5220/0012284900003807

In *Proceedings of the 2nd International Seminar on Artificial Intelligence, Networking and Information Technology (ANIT 2023)*, pages 401-408
ISBN: 978-989-758-677-4

(DT) (R. Qamili) and support vector machine (SVM) (Agarwal S), has been applied to conduct ticket dispatching. An obvious advantage is that these approaches can directly work on trouble tickets with a little effort. However, the characteristics of ticket problem descriptions, such as unformatted, large vocabulary size and short texts, make expert recommendation suffer from low accuracy. The main reason lies in that the representation models, such as the n-gram (R. Kallis), TF-IDF (R. Qamili) and LDA (Zhou W), generally used by these approaches cannot characterize trouble tickets well. To solve this issue, several approaches based on the deep learning technology has been proposed and shown its potential in meeting the need of trouble ticket expert recommendation and improving recommendation performance. In this paper, to leverage the characteristics of tickets well to further improve recommendation accuracy, we propose a deep neural ranking model-based expert recommendation approach for trouble tickets by combining expert profiling, vector-based ticket representation, the attention-based convolution neural network. Further, we evaluate the effectiveness of our model on a real trouble ticket dataset.

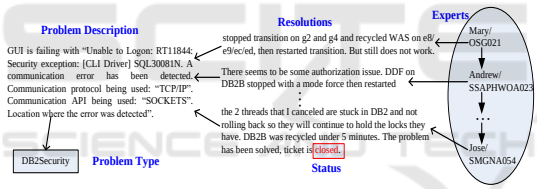

Figure 1. An instance of the trouble tickets.

In summary, our contributions are the followings:

1) An expert profiling component is designed by making full use of ticket problem descriptions and resolutions to characterize expert's professional knowledge with domain words, which is helpful to improve the efficiency and accuracy of ticket assignment.

2) Two attention-based sentence models characterizing problem description and expert profile, respectively, are integrated into our recommendation approach and mapped into the same vector, which enables the semantic similarity measure between the problem description and the expert profile and benefit to improve the recommendation accuracy.

The rest of the paper is organized as follows. In Section 2, a deep learning-based expert recommendation approach is proposed. The experiment settings are explained and the performance evaluation results are discussed in Section 3. At last, we conclude our work in Section 4.

2 THE PROPOSED APPROACH

2.1 Problem Formulation

Assume that $\mathcal{T} = \{t_1, .., t_i, ..., t_N\}$ is a set of tickets from a given IT system, where each ticket can be represented by a quintuple, denoted as $t =< id, c, \tau, r, rp_{(k)} >$, $t \in \mathcal{T}$. The first component denotes the unique identifier of a ticket. The components c, τ and r are the problem type, the problem description and the problem resolution, respectively. We assume that the number of problem types is M. Resolutions from multiple experts involved in the ticket routing sequence are merged into the final problem resolution. The last component $rp_{(k)}$ denotes a ticket routing sequence containing k ordered experts. We operate on historical tickets to output a set of instances, denoted as $I = \{< \tau_i, e_j, s_{ij} >\}$, where τ_i is the problem description for the ith ticket, e_j is the ith expert involving in its ticket routing sequence, and s_{ij} is a competency score of expert e_j. If expert e_j is a resolver for the ith ticket, $s_{ij} = 1$, otherwise $s_{ij} = 0$. Therefore, given the set I, our purpose is to construct a ranking model that calculates an optimal score s_{ij} for each pair $< \tau_i, e_j >$, *s.t.* an expert with a strong competency has a high score. Formally, the expert recommendation task is to learn a ranking function from historical pairs, as shown in Equation (1),

$$h(w, \psi(\tau_i, e_j)) \rightarrow s_{ij} \quad (1)$$

where function $\psi(\cdot)$ maps a pair $< \tau_i, e_j >$ to a similarity vector, where each component reflects a certain type of similarity, e.g., lexical, syntactic, or semantic. The weight vector w is a parameter of the ranking model and is learned during the training.

2.2 Framework

The overview of our expert recommendation approach is illustrated in Figure 2. Our approach consists of four components, including data preprocessing, expert profiling, vector-based representation, and convolution neural network based ranking model. Since problem descriptions record textual information, the data preprocessing component is essential to do some text preprocessing operations by applying the natural language techniques. And then, each problem description is represented as a sentence vector using the learned word vector from historical tickets. On the other hand, domain words are extracted from problem descriptions and resolutions to characterize expert profile. Each expert is also represented as a sentence

vector. Further, each ticket is transformed into at least one triplet containing a problem description vector, an expert profile vector and a competency score. Finally, an attention-based convolution neural network ranking model is trained on these triplets. When an incoming ticket arrives, the resulting ranking model outputs an ordered expert list for the ticket based on the matching score. Experts with the top-*N* competency score are recommended to resolve the incoming ticket one by one until it is resolved.

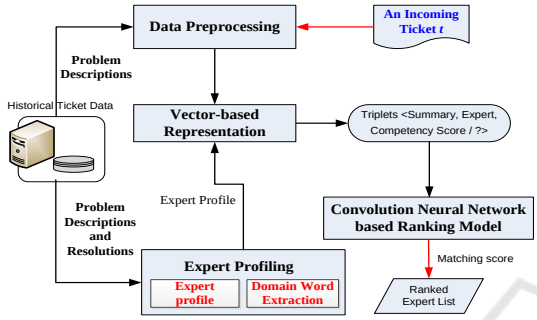

Figure 2. Overview of the proposed Approach.

2.3 Vector-Based Ticket Representation

A proper ticket representation approach is crucial to the task of expert recommendation. Recently, the word vector technology (Zhou W), also known as word embedding, has been successfully applied in the domain of text representation and has been demonstrated that it has a good effect in charactering text semantic information. We argue that deriving an effective representation for problem descriptions plays an importance role in automating IT service management. Thus, in this paper, the word vector technology is also applied to ticket representation.

Assume that V is the word vocabulary of tickets. Let $\mathrm{w} \in \mathbb{R}^{\mathrm{d}}$ be a *d*-dimensional word vector to represent any word from ticket problem descriptions or resolutions, and w belongs to a word vector matrix $\mathrm{W} \in \mathbb{R}^{\mathrm{d}\times|\mathrm{V}|}$. Further, a ticket is represented as a sentence combining its problem description and resolution. Using the word vector matrix, each ticket is mapped to their vector representation. Assume that ticket t contains l words. The sentence for ticket t is represented as a matrix $\mathrm{s} \in \mathbb{R}^{\mathrm{d}\times l} = [\mathrm{w}_1, \ldots, \mathrm{w}_l]$.

Although there are some publicly pre-trained word vector models, such as word2vec (Aggarwal V), FastText (Athiwaratkun), and Bert. However, due to the wide existence of non-dictionary words in tickets, these pre-trained word vector models cannot be applied to our work. According to the common experience that a minimal size of the corpora required for learning word vectors should be at least in the order of hundreds of thousands, we use sufficient historical tickets in our database as text corpora to learn and extend the existing word vectors applicable to IT service management.

2.4 Expert Profile and Representation

Expert profiling is a critical factor impacting the efficiency and accuracy of ticket assignment. Here, we use the professional knowledge or skills owned by experts to characterize expert profile, where the professional knowledge indicates his/her expertise to resolve the ticket. For a given ticket, it is difficult to determine whether an expert's profile matches with the ticket problem description. Thus, we represent each expert profile as a group of domain words extracted from problem descriptions and resolutions to indicate general interests and activities of an expert, defined as Definition 1.

Definition 1 (Expert Profile). A expert profile of expert e_i is represented as a set of words, $ep_i = \{dw_r^i|_{r=1,\ldots,K}\}$, where $\{dw_1^i, \ldots, dw_r^i, \ldots, dw_K^i\}$ denotes the set of *K* domain words describing general interests and activities of expert e_i.

Different with the traditional way, we attempt to automatically extract domain words from historical tickets without human intervention. Using historical tickets resolved by an expert, we think that those words frequently occurring in problem descriptions and resolutions of those tickets have a higher probability to be domain words of the expert. For any expert, we extract the same number of words to characterize the expert.

We find tickets solved by a given expert from historical tickets, and then candidate domain words are fetched from these tickets. Domain words are usually nouns or verbs provided by system administrators or obtained from related documents like the catalog taxonomy of system management. After fetching candidate domain words, we measure the importance of each word using the idea similar to TF-IDF. For a candidate word *w*, we first measure its frequency of appearing in tickets solved by expert e_j, denoted as $tf(\mathrm{w})$. The higher $tf(\mathrm{w})$, the higher the possibility that the word is a terminology is. Second, we measure its expert frequency, denoted as $ef(\mathrm{w})$, which represents the number of experts that have ever solved the tickets containing the word. Third, we measure its inverse expert frequency, denoted as $ief(\mathrm{w})$. The higher the value of $ief(\mathrm{w})$, the higher the differentiation of the word among experts is. Last,

we take these two factors into consideration to get the importance of the word, and sort all candidate domain words by the importance in a descending order and then select the most important top-K words as the domain words.

To accurately measure the semantic similarity between a ticket and an expert, we make the expert representation consistent with ticket representation. An expert is represented as a sentence combining its domain words. Using the same word vector matrix W used in the ticket presentation, each expert is mapped to their vector representation. Assume that expert profile contains K words. The sentence for expert e is represented as a sentence matrix $\mathrm{q} \in \mathbb{R}^{\mathrm{d}\times K} = [\mathrm{w}_1, \ldots, \mathrm{w}_K]$.

2.5 CNN-Based Ranking Model

In this section, we build an attention-based CNN ranking model to recommend experts, as shown in Figure 3. Our model consists of two parts. The first part consists of two attention-based sentence models for mapping the trouble ticket problem and the expert profile to their vector representation, respectively. The second part is an expert ranking model that ranks experts by learning the semantic similarity score between the problem description and the expert profile. We will describe these two parts in the following texts. Generally, an incoming ticket can be resolved by multiple experts with different competency values. Thus, our model will recommend all the candidates in the order that an expert with a high competency ranks first.

In the embedding layer, the input is sentences s and q, treated as sequences of words that represent a ticket problem description and an expert capacity respectively, where each word is drawn from a word vocabulary V. Words are represented by d-dimensional vectors $\mathrm{w} \in \mathbb{R}^d$. Input sentences s and q are represented by matrices $S \in \mathbb{R}^{d\times|t_s|} = (\mathrm{w}_1, \ldots, \mathrm{w}_{|t_s|})$ and $Q \in \mathbb{R}^{d\times|t_q|} = (\mathrm{w}_1, \ldots, \mathrm{w}_{|t_q|})$, respectively, where t_s and t_q are the number of words. To make our model focus on domain words, we introduce the attention mechanism. Note that although a sentence representing the expertise profile is generated by combining those domain words, expert's expertise differs from expert to expert. Thus, it is reasonable to pay close attention to domain words related to query topics. Specifically, an attention coefficient matrix $A = \mathbb{R}^{|t_s|\times|t_q|}$ is generated by calculating the similarity between two matrices S and Q. For any $A_{i,j} \in A$, $\mathrm{A}_{i,j} = attention(S_i, Q_j)$ is a coefficient matrix, where S_i and Q_j denote the ith and jth word vector in S and Q respectively, and $attention(x, y)$ is a transformation function used to calculate the attention coefficient. There are several alternative approaches for the transformation function. Here, the coefficient matrix calculation is defined as (2), where $|\cdot|$ denotes Euclidean distance between two words.

$$attention(S_i, Q_j) = \frac{1}{1+|S_i - Q_j|} = \frac{1}{\sqrt{\sum_{k=1}^{d}(S_{k,i} - Q_{k,j})^2}} \quad (2)$$

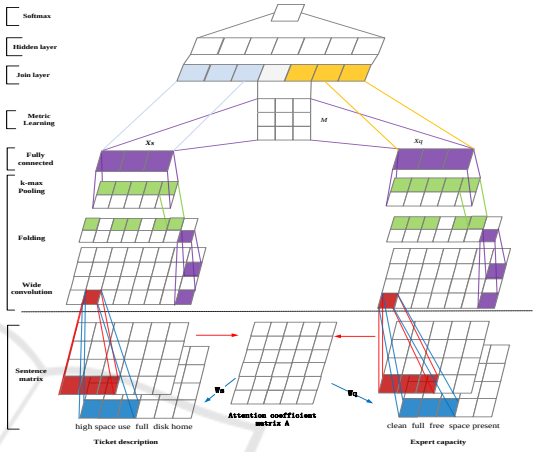

Figure 3. Overview of the CNN-based Ranking Model.

Further, the attention coefficient matrix A is transformed into an attention feature matrix with the same dimension as sentence matrices, and then is used together with the original sentence matrices as inputs to the convolution operation. The attention feature matrix enables the convolution operation to learn domain words. The attention feature matrices S_a and Q_a can be calculated by Equation (3) and Equation (4), respectively, where the weight matrices $W_s \in \mathbb{R}^{d\times|t_s|}$ and $W_q \in \mathbb{R}^{d\times|t_q|}$ are parameters learned during the training process. We randomly initialize them.

$$S_a = W_s . A^T \quad (3)$$

$$Q_a = W_q . A \quad (4)$$

The convolutional layer aims to obtain interesting patterns of word sequences as ticket features and expert capability features. We apply a one-dimensional convolution operation on the sentence vector $s \in \mathbb{R}^{|s|}$ and the convolution kernel $f \in \mathbb{R}^{|h|}$ in a wide convolution manner. The one-dimensional convolution operation is taken in each h-size window of sentence s to obtain another sequence c:

$$c_j = (s * f) = s_{j-h+1}^T \cdot f = \sum_{k=j}^{j+h-1} s_k f_k \quad (5)$$

where each row vector $c_j \in \mathbb{R}^{|s|+h-1}$ 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 $F \in \mathbb{R}^{d\times h}$, that work in parallel are applied in a deep learning model, producing multiple feature maps $C \in \mathbb{R}^{d\times|s|+h-1}$. A nonlinear feature of the text is extracted using a Rectified Linear Unit (ReLU) as an activation function after each convolutional layer.

The folding layer aims to capture the association information of features between adjacent rows and reduce the dimension of features, which sums up every two rows in the feature map component-wise.

The pooling layer aims to extract the most representative feature from sentences to reduce the representation. We use the k-max pooling strategy to select the top k features from all features according to the sentence input length, and to keep the order information of features, which can effectively preserve the strength of recurring word features, especially for words that may be repeated in problem descriptions and solutions.

Using the fully connected layer, we fetch the resulting representation vectors X_s and X_q of the same dimension for processing tickets and expert profiles.

The second part starts from using these two vectors to be feed into the expert ranking model in order to recommend experts. First, the similarity between a ticket and an expert is calculated using Equation (6) as the expert matching score.

$$sim(x_s, x_q) = x_s^T M^s x_q \quad (6)$$

where $M^s \in \mathbb{R}^{d\times d}$ 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. The similarity matrix M is a parameter of the network and is optimized during the training. And then, the joint vector is passed through a three-layer, full connection, feed-forward neural network, which allows rich interactions between a sentence pair from one of the three components. The joint layer is responsible for connecting these two eigenvector and the similarity matrix using Equation (7).

$$x_{join} = (x_s^T; x_{sim}; x_q^T) \quad (7)$$

The output of the hidden layer is calculated using $\alpha(w_h \cdot x + b)$, where w_h is the weight vector of the hidden layer, and $\alpha(\cdot)$ is a nonlinear activation function. Here, the ReLU function is used as the activation function. Finally, the neurons output by the hidden layer are passed through the softmax layer to get the resulting probability as the expert recommendation score, as shown in Equation (8), where θ_k represents the weight vector of the kth problem area.

$$score(x_s, x_q) = p(y = j|x_{join}) = \frac{e^{x_{join}^T \theta_j}}{\sum_{k=1}^{K} e^{x_{join}^T \theta_k}} \quad (8)$$

Our model is trained to minimize the binary cross-function, as shown in Equation (9).

$$L = -\log \prod_{i=1}^{N} p(y_i|s_i, q_i) = -\sum_{i=1}^{N} y_i log a_i + (1 - y_i)\log(1 - a_i) \quad (9)$$

where y_i and a_i are the ground truth and the prediction result for the ith pair of ticket and expert. The parameters in the neural network are trained by the mini-batch gradient descent approach, and the sample size of each epoch is optimized during the experiment. In order to mitigate the over-fitting issue, we augment the cost function with L2-norm regularization to constrain the parameters, and employ the dropout strategy in the full-connection layer to prevent feature co-adaption by dropping out a portion of hidden units during the forward phrase.

2.6 Ranking-Based Expert Recommendation

For an incoming ticket t and a set of experts D, ticket-expert pairs $\{< t, e_i > |1 \le i \le D\}$ are built and feed into the ranking model. The expert recommendation scores for these pairs are outputted using the trained ranking model, and experts are sorted based on their competency scores in a descending order. A straightforward recommendation policy is to recommend an expert who has not been recommended from the ordered expert list each time until the ticket is resolved.

3 EXPERIMENTS

3.1 Experiment Settings

The ticket dataset used in our experiments was collected from an account of a large IT service provider, which contains over 479079 tickets belonging to 95 problem types and 582 system maintenance experts. Statistically, the average number of tickets resolved by an expert is close to 823. 10% of tickets in the dataset are randomly selected to generate the testing dataset, while the rest is used as the training dataset. After that, the natural language processing technique is used to remove stop words and build part-of-speech tags for ticket problem descriptions and resolutions. The nouns, adjectives and verbs are kept as the signature term candidates and are concatenated into a sentence by keeping the word order in its original texts. Further,

for each ticket in the training data, it is divided into several instances with the form $< t_i, e_j, s_{ij} >$, where t_i denotes a ticket represented by a sentence, e_j is an expert involved in the ticket routing sequence and represented by a sentence of domain words, and s_{ij} denotes whether expert e_j solves ticket τ_i or not, $s_{ij} = 1$ if expert e_j is a resolver, otherwise $s_{ij} = 0$. For each ticket in the testing data, it is transformed into one instance with the form $< t_i, e_j, ? >$ by only considering the resolver of this ticket.

Two common metrics in expert recommendation are used to evaluate our approach, and they are precision@α and Mean Steps to Resolve (MSTR) (Shao Q, Xu J, Xu J). Precision@α (short for p@α) (Miao G, Aggarwal V) relates to precision, where α is a position parameter. For example, p@1 denotes the probability that the first recommended expert is a resolver, which is a natural way to indicate the recommendation quality of the retrieved top-N experts. Besides performance, we also use MSTR to evaluate efficiency by measuring the mean steps of resolving a ticket. Obviously, we prefer to a lower MSTR for an efficient recommendation. Assume that T is a set of tickets, $\forall t_i \in T$, expert e_i is the resolver of ticket t_i and $EList(t_i)$ denotes a ordered expert list recommended by any expert recommendation algorithm. MSTR can be calculated as:

$$MSTR(T) = \frac{\sum_{t_i \in T} I(t_i) \times k_i}{|T|} \quad (10)$$

where $I(t_i) = 1$ if $EList(t_i)$ contains expert e_i and k_i denotes the position of expert e_i in the list, otherwise $I(t_i) = 0$.

Further, the four state-of-the-art approaches are considered for comparison, including logistic regression (LR) (R. Qamili), support vector machine (SVM) (Agarwal S), STAR (Zhou W) and ABCNN-1 (Yin W).

All recommendation models are implemented using Python. For convenience, we name our model as CNN-ATT. The testing machine is Windows 10 equipped with Intel Xeon E5-2699 V4 2.3GHz CPU and 256GB RAM. We evaluate all models on the testing data set using the above mentioned metrics.

3.2 Results

First, we evaluate the impact of different representation models on recommendation performance and efficiency. We compare four common representation models, including TF-IDF, Word2Vec, FastText and Bert, and present the results in Table 1. We can see that the distributed representation achieves a good performance. Specifically, the algorithm using Bert performs better than the algorithms using any other text representation model in terms of P@1 and MSTR. Our expert recommendation algorithm with a combination of Bert and the deep learning technology has a great improvement on P@1 over 22.1% and 6.95% compared to TF-IDF and Word2Vec, respectively. The results using FastText are similar to Word2Vec. On the other hand, our algorithm also has an obvious efficiency improvement in terms of MSTR, over 15.97% and 9.0% compared to TF-IDF and Word2Vec, respectively. The main reason lies in that the Bert model trained from historical tickets can characterize the distribution of features automatically and represent the tickets well, while the traditional approaches highly depend on manual feature engineering and may miss important features.

Second, we evaluate the impact of sentence length on expert recommendation performance and efficiency. We measure the distribution of the number of words appearing in problem description and characterizing expert profile from the historical tickets, respectively. As for the length of problem description, we can find that about 77% tickets contain 25 to 100 words, and the maximum length is 197. As for the length of sentence representing expert profile, we can find that about 68% tickets contain 10 to 120 domain words, and the maximum length is 237. Hence, we evaluate our algorithm by varying the sentence length, and the results are shown in Figure 4, where the length of sentence model for problem description varies from 20 to 100 while the length of sentence model for expert profile varies from 10 to 120. We can observe that the recommendation performance is optimal when the length of sentence models for problem description and expert profile is 50 and 100, respectively, which means that the longer length of sentence models do not always result in a good recommendation performance, but spends more computing time.

Table 1. Results from the model using different presentation approaches.

Representation	P@1	MSTR
TF-IDF	0.7160	3.13
Word2Vec	0.8175	2.89
FastText	0.8104	2.95
Bert	0.8743	2.63

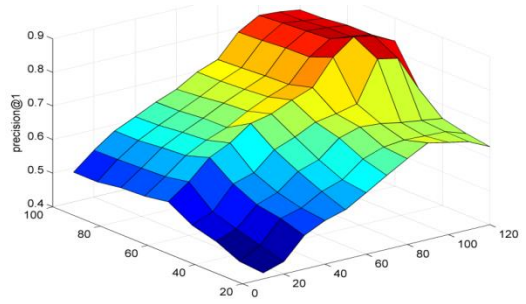

Figure 4. The impact of the sentence length on recommendation performance.

Table 2. Overall performance comparison.

Approaches	P@1	MSTR
LR	0.6672	4.53
SVM	0.6895	4.12
STAR	0.8459	3.01
ABCNN-1	0.8464	2.98
CNN-w/o-ATT(ours)	0.8497	2.97
CNN-ATT(ours)	0.8743	2.63

Automating expert recommendation can be tackled by applying different approaches. Here, we compare several alternative algorithms from aspects of the traditional machine learning and deep learning technique in terms of p@1 and MSTR. Overall performance results are shown in Table 2. We have some interesting observations. First, compared to traditional machine learning approaches, the approaches based on the deep learning technology perform better, which means that our CNN ranking model-based solution is effective by making full use of the semantic information from ticket problem texts. Second, the approach using the attention mechanism performs better than the others without it, which means that the attention mechanism can help our model identify key words in problem descriptions and expert profiles and improve the recommendation performance. Of course, we also see that the optimal precision of the initial expert recommendation is 87.43%, which means that our approach dispatches a trouble ticket to the following expert in the ordered recommendation list until the ticket is resolved.

4 CONCLUSION

To quickly and accurately dispatch trouble tickets from complex IT systems to skilled expert for problem-solving, a deep learning-based expert recommendation approach is proposed in this paper. Our approach takes both problem description and resolution into consideration to characterize expert profile and build recommendation model. The unified distributed representation is first applied to characterize both problem description and expert profile. And then, an attention-based convolution neural network is built to learn a ranked expert recommendation model by taking two sentence models as inputs. For an incoming ticket, an expert list is recommended to dispatch the trouble ticket. The experimental results on real-world tickets show that our approach performs better than both the traditional machine learning-based approaches and the convolution neural network-based approaches in terms of p@1 and MSTR.

ACKNOWLEDGMENTS

The work was supported in part by State Grid Jiangsu Electric Power CO., LTD (J2022014).

REFERENCES

Shao Q, Chen Y, Tao S, Yan X, Anerousis N. EasyTicket: a ticket routing recommendation engine for enterprise problem resolution(C). *Proceedings of the VLDB Endowment*. 2008, 1(2):1436-1439. https://doi.org/10.14778/1454159.1454193

Shao Q, Chen Y, Tao S, Yan X, Anerousis N. Efficient ticket routing by resolution sequence mining(C). *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2008, pp. 605-613. https://doi.org/10.1145/1401890.1401964

Yin W, Schütze, Hinrich, Xiang B, et al. ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs (J).*Transactions of the Association for Computational Linguistics*, 2016, 4:259-272. https://doi.org/10.1162/tacl_a_00097.

Agarwal S, Sindhgatta R, Sengupta B. SmartDispatch: enabling efficient ticket dispatch in an IT service environment(C). *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2012, pp. 1393-1401. https://doi.org/10.1145/2339 530.2339744

Botezatu, M. M., Bogojeska, J., Giurgiu, I., Voelzer, H., & Wiesmann, D. Multi-View Incident Ticket Clustering for Optimal Ticket Dispatching(C). *Proceedings of the 21th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining*. 2015, 1711-1720. https://doi.org/10.1145/2783258.2788607

R. Qamili, S. Shabani and J. Schneider, An Intelligent Framework for Issue Ticketing System Based on Machine Learning(C), *Proceedings of 2018 IEEE 22nd International Enterprise Distributed Object Computing*

Workshop (EDOCW), Stockholm, 2018, pp. 79-86. https://doi.org/10.1109/EDOCW.2018.00022

R. Kallis, A. Di Sorbo, G. Canfora and S. Panichella, Ticket Tagger: Machine Learning Driven Issue Classification(C), *Proceedings of IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Cleveland, OH, USA, 2019, pp. 406-409. https://doi.org/10.1109/ICSME.2019.00070

Zhou W, Tang L, Li T, Shwartz L, Grabarnik GY. Resolution recommendation for event tickets in service management(C). *Proceedings of 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)* 2015 May 11, pp. 287-295. https://doi.org/10.1109/INM.2015.7140303

Xu J, Zhang H, Zhou W, et al. Signature based trouble ticket classification (J). *Future Generation Computer Systems*, 2018, 78:41-58. https://doi.org/10.1016/j.future.2017.07.054

Zhou W, Li T, Shwartz L, Grabarnik GY. Recommending ticket resolution using feature adaptation(C). *Proceedings of 11th International Conference on Network and Service Management (CNSM)*, 2015 Nov 9 (pp. 15-21). IEEE. https://doi.org/10.1109/CNSM.2015.7367333

Zhou W, Xue W, Baral R, et al. STAR: A System for Ticket Analysis and Resolution(C)// *Proceedings of the 23th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp 2181-2190. https://doi.org/10.1145/3097983.3098190

Athiwaratkun, Ben, A. G. Wilson, and A. Anandkumar. Probabilistic FastText for Multi-Sense Word Embeddings(C). *In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. 2018, pp.1-11. https://doi.org/10.18653/v1/P18-1001

Xu J, He R. Expert recommendation for trouble ticket routing (J). *Data & Knowledge Engineering*, 2018, 116(JUL.):205-218. https://doi.org/10.1016/j.datak.2018.06.004

Xu J, He R, Zhou W, et al. Trouble Ticket Routing Models and Their Applications (J). *IEEE Transactions on Network & Service Management*, 2018, 15(2):530-543. https://doi.org/10.1109/TNSM.2018.2790956

Miao G, Moser LE, Yan X, Tao S, Chen Y, Anerousis N. Generative models for ticket resolution in expert networks(C). *In Proceedings of the 16th ACM international conference on Knowledge discovery and data mining (SIGKDD)*. 2010, pp. 733-742. https://doi.org/10.1145/1835804.1835897

Aggarwal V, Agarwal S, Dasgupta GB, et al. ReAct: A System for Recommending Actions for Rapid Resolution of IT Service Incidents(C). *In 2016 IEEE International Conference on Services Computing (SCC)*, 2016, pp. 1-8. https://doi.org/10.1109/COMPSAC.2016.170

Quoc Le, Tomas Mikolov. Distributed representations of sentences and documents(C). *Proceedings of the 31st International Conference on International Conference on Machine Learning*. 2014, 1188-1196. https://dl.acm.org/doi/10.5555/3044805.3045025