Cognitive System to Achieve Human-Level Accuracy in Automated Assignment of Helpdesk Email Tickets

Atri Mandal[1(✉)], Nikhil Malhotra[2], Shivali Agarwal[1], Anupama Ray[1], and Giriprasad Sridhara[1]

[1] IBM Research AI, Bengaluru, India
[2] IBM Global Technology Services, Bengaluru, India
{atri.mandal,nikhimal,shivaaga,anupamar,girisrid}@in.ibm.com

Abstract. Ticket assignment/dispatch is a crucial part of service delivery business with lot of scope for automation and optimization. In this paper, we present an end-to-end automated helpdesk email ticket assignment system, which is also offered as a service. The objective of the system is to determine the nature of the problem mentioned in an incoming email ticket and then automatically dispatch it to an appropriate resolver group (or team) for resolution.

The proposed system uses an ensemble classifier augmented with a configurable rule engine. While design of classifier that is accurate is one of the main challenges, we also need to address the need of designing a system that is robust and adaptive to changing business needs. We discuss some of the main design challenges associated with email ticket assignment automation and how we solve them. The design decisions for our system are driven by high accuracy, coverage, business continuity, scalability and optimal usage of computational resources.

Our system has been deployed in production of three major service providers and currently assigning over 40,000 emails per month, on an average, with an accuracy close to 90% and covering at least 90% of email tickets. This translates to achieving human-level accuracy and results in a net saving of about 23000 man-hours of effort per annum.

Keywords: Cognitive email assignment · Helpdesk automation
Ticket resolver group · Smart dispatch · Ensemble classifiers

1 Introduction

The landscape of modern IT service delivery is changing with increased focus on automation and optimization. Most IT vendors today, have service platforms aimed towards end-to-end automation for carrying out mundane, repetitive labor-intensive tasks and even for tasks requiring human cognizance. One such task is ticket assignment/dispatch where the service requests submitted by

C. Pahl et al. (Eds.): ICSOC 2018, LNCS 11236, pp. 332–341, 2018.
https://doi.org/10.1007/978-3-030-03596-9_23

the end-users to the vendor in the form of tickets are reviewed by a centralized dispatch team and assigned to the appropriate service team i.e. resolver group.

The dispatch of a ticket to the correct group of practitioners is a critical step in the speedy resolution of a ticket. Incorrect dispatch decisions can significantly increase the total turnaround time for ticket resolution, as observed in a study of an actual production system [2]. Several factors make the dispatcher's job challenging such as requirement of knowledge of the IT portfolio being managed, roles and responsibilities of the individual groups, ability to quickly parse the ticket text describing the problem and map it to the right group, which is often not straightforward given the heterogeneous and informal nature of the problem description. A number of different approaches have been proposed for automating ticket dispatch [2,7,10,11]. Although automated email assignment may look like a simple text classification problem at first glance it becomes quite complex and challenging when considered at industry scale.

In this paper we present a deployed end-to-end automatic email dispatch system having the following key features:

1. An ensemble based classification engine that uses supervised data in the form of unstructured email text and resolver groups as labels. The choice of ensemble is based on the results of comprehensive study performed with various machine learning and deep learning models as presented in Sect. 4.2.
2. A rule engine with a customer-independent framework for rule specification to ensure business continuity and handle domain specific content missed by the ensemble classifier.
3. Experimental results with real customer data from three different datasets - the largest of them having more than 700,000 emails and 428 resolver groups. We were able to achieve human level accuracy with more than 90% coverage on all the datasets with the proposed system using minimal computational resources.

The remainder of the paper is organized as follows. Section 2 describes the related work. Section 3 gives a system overview, and Sect. 4 discusses the different components of the system in detail. We present our experimental results in Sect. 5 and conclusion in Sect. 6.

2 Related Work

A semi-automated approach based on confidence scores of Support Vector Machines and discriminative keywords had been proposed in [2] for ticket dispatch. We have surpassed their work to (i) reach human level accuracy using advanced ensemble techniques for automated dispatch, (ii) scale it to hundreds of resolver groups and (iii) incorporate retraining strategies to adapt to changing data. Several other researchers have studied different aspects of the problem of routing tickets to resolver groups [7,10,11]. The work in [11] approaches the problem by mining resolution sequence data and does not access ticket description at all. Its objective is to come up with ticket transfer recommendations given

the initial assignment information. The work in [10] mines historical ticket data and develops a probabilistic model of an enterprise social network that represents the functional relationships among various expert groups in ticket routing. Based on this network, the system then provides routing recommendations to new tickets. The work in [7] approaches the problem from a queue perspective, related to the issue of service times and becomes particularly relevant when the ticket that has been dispatched to a group needs to be assigned to an agent. Some works have focused on applying text classification techniques to handle tickets [3,12], by identifying the ticket category helping human dispatchers for faster ticket assignment. The work in [4] attempts to classify the incoming change requests into one of the fine-grained activities in a catalog. Some other works [1,9] talk about a holistic approach of ticket category classification, cause analysis and resolution recommendation. However, they do not automate the process of assignment.

3 System Overview

Figure 1 shows the system architecture along with the data flow diagram. Historical email ticket data is downloaded from the ticketing tool (e.g. Remedy or ServiceNow) using custom-built adapters. The downloaded emails are passed through two stages of pre-processing for data enrichment. The resolver group level pre-processing module uses techniques like resolver group merging, long tail cutoff etc. to reduce the noise in the email data. The training data is further enriched using text pre-processing methods. The enriched email data is then trained using an ensemble of machine learning classifiers and the trained models are stored in a database.

When a user sends an email to the helpdesk account a ticket is automatically generated and stored in the backend ticketing tool. The newly generated tickets are downloaded by the adapter and classified using the runtime that consists of ensemble classifier and the rule engine. The classification system returns a resolver group along with a confidence score. If the confidence score is above a configured threshold the ticket is routed to the returned resolver group. Otherwise the ticket is assigned back to manual queue for inspection by human agent. The combination of ensemble classifier and rule engine ensures that a high percentage of tickets (more than 90%) are classified automatically by our system with a low error rate.

4 Assignment Engine Components

4.1 Preparation of Training Data

Most large companies nowadays use ticketing tools like Remedy or ServiceNow to maintain tickets obtained from various channels (voice, email, web etc.) by the helpdesk. The ticketing tool organizes the email data into structured fields containing relevant information about the ticket e.g. incident number, incident

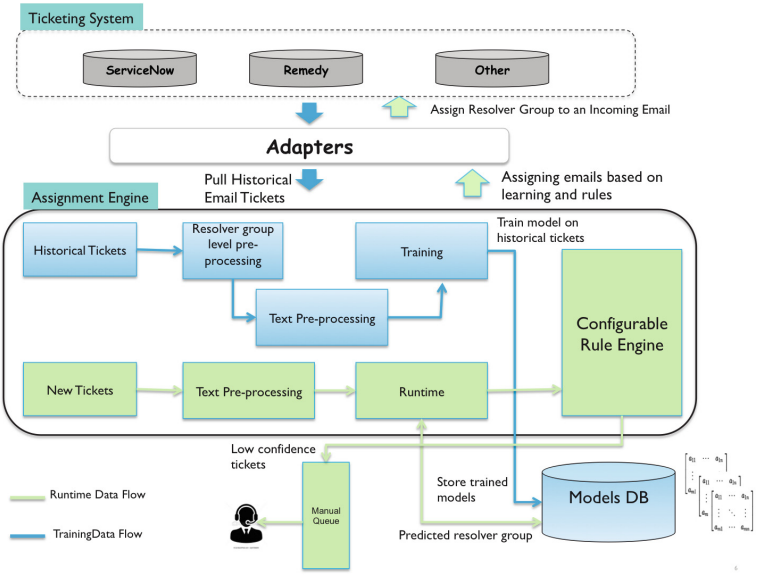


Fig. 1. Architecture of the proposed system

type, date of creation, description, assigned group etc. We use custom adapters to connect to the ticketing tool and extract fields relevant for training. Currently the adapter extracts only the text portion of the email (viz. email subject and body) along with the resolver group for training. The data collected by the adapter is then converted into a format readable by the classifier. The steps involved in training data preparation are described below.

Resolver Group Level Pre-processing. This type of pre-processing is a one-time effort required during customer on-boarding phase. The purpose of this pre-processing is to reduce noise in the training data. We reduce noise and enrich training data for the resolver groups using the following techniques:

Merging Related Resolver Groups. Some of the resolver group labels in the training data can be merged. Merging increases the size of the training data and at the same time reduces the number of unique labels thus improving training accuracy. We found that there are at least two types of resolver groups that can be merged for assignment purpose viz. **(a) Resolver groups with varying escalation levels** and **(b) Region (or zone) specific resolver groups**.

Long Tail Cutoff. We observed that in most of the datasets there are a large number of resolver groups with very few samples. If we plot a histogram of frequencies these groups will constitute more than 80% of the resolver groups but less than 5% of training data. Our studies indicate that, if the long tail is included in training, the overall accuracy of classification goes down along with a significant increase in training time and model size. By restricting the number of resolver groups in training we reduce noise significantly and also avoid

class imbalance. Additionally, the resolver groups, which fall in the long tail, can often be better predicted using the rule engine and using some augmentation techniques. As such our strategy was to divide the downloaded historical data into 2 parts viz. IH = IT + IL where IH is the complete data downloaded for training, IT is the data used for training classifiers and IL is the long tail. Resolver groups belonging to IT will be classified using trained models while those belonging to IL will be handled exclusively by the rule engine. In our system we use the above strategy to retain at least 98% of data while cutting down the resolver group count to less than 20%.

4.2 Classification Models

This section presents our study on the performance of various machine-learning classifiers in classification of email data, in terms of accuracy and training time, although the training is offline. For training the classification models, we concatenate the subject and the body of the email(description) with a space in between and use the resulting string as our training data. The resolver group acts as the label for our training data. Tables 1 and 2 show the impact of various traditional machine learning models [6] and deep neural network models that were used. In order to improve accuracy and coverage of the overall service, we use an ensemble [5]. Each pair of models were combined, and the final ensemble classifier was chosen based on the accuracy and coverage. As explained in Sect. 4.1, rule engine is important to handle the long tail in class distribution and the final chosen ensemble classifier in combination with the rule engine forms the classification module of the service.

Table 1. Comparison of various machine learning algorithms w.r.t. accuracy and training time

		LinearSVM	KNN	LR	m-NB	RF	Adaboost	Gradient boosting
Dataset A	Accuracy (%)	87.3	80.12	79.48	72.68	81.41	31.5	75.6
	Train-time (s)	7.8	260.5	43	17.3	363.75	4561	8612
Dataset B	Accuracy (%)	83.42	72.58	79.95	64.19	74.91	32.98	65.1
	Train-time (s)	76.12	2218.65	404.05	22.18	7190.16	332.97	95320.1
Dataset C	Accuracy (%)	86.339	67.57	84.29	63.97	76.99	30.43	61.47
	Train-time (s)	1001.06	1921.7	2992	167.5	20799.6	1288.63	126960

Training the Classifiers. We convert the training data samples into word vector representation before applying machine-learning algorithms. We observed that using tf-idf representation increased the accuracy of traditional machine learning algorithms for all datasets by at least 3–4%. Another observation was that using bigrams also improved the accuracy for some datasets. Intuitively we can argue that this is so because some bigrams like 'account creation', 'account deletion', 'password reset' etc. are useful indicators in deciding the

Table 2. Comparison of various deep neural networks w.r.t. accuracy and training time

		MLP	CNN-WE	LSTM-WE	CNN-G	LSTM-G	CNNLSTM-G
Dataset A	Accuracy (%)	85.8	74	76.94	74.01	71.64	73.24
	Train-time (s)	184.12	183.75	5546.6	160.56	9833.7	1844.8
Dataset B	Accuracy (%)	80.87	77.75	79.35	76.23	80.37	77.7
	Train-time (s)	10858.15	8680.35	86651.57	1926	89280.94	23229.47
Dataset C	Accuracy (%)	83.3	78.8	78.14	79.1	83.51	81.33
	Train-time (s)	2779	4000.9	90149.9	9522.12	687483	116583.22

resolver group. The hyperparameters were chosen experimentally over 10-fold cross-validation on the datasets.

However, for learning deep neural networks, tf-idf representation being extremely sparse is not useful and hence we used word embeddings. There are primarily two methods of learning the word embeddings: one in which word embeddings are learnt while training the neural network; and second using pre-trained word vectors. We experimented with both methods for classification (models learning word embeddings being referred to CNN-WE, LSTM-WE, and CNN-LSTM-WE in Table 2), and pretrained word-vector representations (100-d GloVe vectors) [8] referred to as CNN-G, LSTM-G and CNN-LSTM-G.

4.3 Rule Engine

The rule engine is one of the key components of our end-to-end system and is used to capture domain specific elements as well as to ensure business continuity. Also, the resolver groups belonging to IT can only be predicted using the rule engine as discussed in Sect. 4.1.

The rule engine is designed for ease of usage and configuration. Each rule in the rule engine can be expressed by the following equation:

$$(f1 = \phi1) \wedge (f2 = \phi2) \wedge (f3 = \phi3) \wedge (CE = R) \Rightarrow CF \tag{1}$$

where CF is the resolver group predicted by the engine, ϕi is the value of the ticket parameter i and CE is the resolver group predicted by the ensemble classifier. Here ϕi can take a finite value (or regular expression) or the value X (implying DON'T CARE).

4.4 Email Ticket Dispatcher

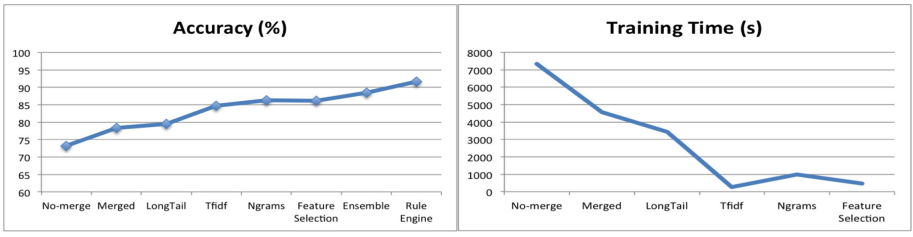
The email ticket dispatcher actually assigns the ticket to a specific resolver group and updates the ticket. The dispatcher selects an ensemble of two classifiers to optimize accuracy and coverage as shown in Fig. 3. It combines the results of the ensemble classifier and rule engine using a dispatch algorithm to output the final prediction and confidence score. If the confidence score of the final result is below the configured threshold the ticket is assigned to the manual queue.

Table 3. Dataset details and results

	Dataset A	Dataset B	Dataset C
Number of email tickets (N)	11562	423343	712320
Number of resolver groups	70	403	428
Duration of the dataset	6 months	12 months	15 months
Ensemble Accuracy (X_{acc})	90.07%	86.17%	89.61%
Ensemble Coverage (X_{cov})	93.67%	92.88%	93.83%
Assignment Engine Accuracy (E_{acc})	92.73%	88.66%	92.13%
Assignment Engine Coverage (E_{cov})	97.84%	93.3%	95.5%

5 Experimental Results

This section enumerates the results of the experimental setup of the assignment engine. For evaluation we have used real datasets from three major helpdesk service providers. The datasets are from two different domains viz. telecom and supply-chain/logistics. To preserve client confidentiality we henceforth refer to these datasets as Dataset A, Dataset B and Dataset C respectively. The datasets were divided into training and test sets with a 90:10 split and we used 10-fold cross-validation on the datasets. All our experiments were run on a NVIDIA Tesla K80 GPU cluster with 4 CUDA-enabled nodes. We use open source machine-learning libraries viz. Python scikit-learn and Keras for our experiments. The dataset statistics as well as the final accuracy numbers achieved by our system are described in Table 3. Please note that the deployment setup is similar to our experimental setup but not identical; so numbers in production may vary slightly. The details about production setup and results are not included to preserve confidentiality.



(a) (b)

Fig. 2. Effect of different optimization techniques (a) Accuracy trend (b) Training time.

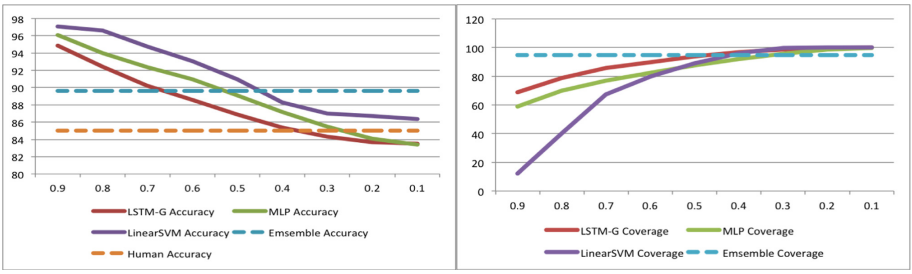
5.1 Accuracy and Training Time

Figure 2 shows how the training techniques and preprocessing affect the accuracy of prediction and training time. It shows the gradual increase in accuracy and corresponding decrease in training time as we apply each technique(shown in X-axis) incrementally. The accuracy and training time charts are shown for only one of the datasets viz. dataset C which is our largest dataset - but the trend is fairly similar across other datasets as well.

5.2 Human Accuracy vs. Assignment Engine Accuracy

We next look at the optimal selection of algorithms that maximize accuracy and coverage. We assert that for business purposes the algorithms need to have at least human-level accuracy with high enough coverage.

To compute human accuracy we mined audit logs of the ticketing systems. Our experiments reveal that across all datasets the accuracy achieved by human agents is about 85%. Therefore we select the confidence threshold such that the expected accuracy of prediction is at least 85%. Figure 3 show the performance of the best three algorithms at different confidence levels (ranging from 0.1 to 0.9). For dataset C a combination of linear SVM (confidence ≥ 0.5) and MLP (confidence ≥ 0.6) gave a slightly higher accuracy (89.61%) than that of LSTM-G (confidence ≥ 0.5) and linear SVM ($X_{acc} = 88.38\%$), although the individual accuracy was marginally higher for LSTM-G compared to MLP. For this reason, as also for other practical considerations like memory and CPU constraints as well as training time our deployment in production uses an ensemble of linear SVM and MLP. For the other two datasets SVM and MLP were clear winners.



(a) (b)

Fig. 3. At different confidence thresholds (a) Assignment accuracy (b) Assignment coverage.

5.3 Observations

There are three main takeaways from our experimental results above. The most important observation is that our assignment engine (ensemble classifier augmented with rule engine) performs better, in terms of accuracy and coverage,

than all traditional machine-learning and deep learning algorithms. Secondly we can see that simple machine learning algorithms like SVM and MLP are often better than more computationally expensive deep learning algorithms in the task of helpdesk email assignment automation. This result is somewhat surprising and unexpected, but is very significant from a product development standpoint as these algorithms are easy to implement, require minimal computational resources and provide better performance at runtime. Last but not the least we observe that LSTM accuracy increases with the size of the dataset and for the largest dataset (dataset C) it outperforms MLP. Thus our results indicate that an ensemble of SVM and MLP will be a good trade-off for most practical purposes but if we have a large enough dataset and infrastructure is not a concern then the best choices are SVM and LSTM-glove.

6 Conclusion and Future Work

We have proposed email ticket assignment engine that uses an ensemble of machine learning techniques, combined with a configurable rule engine, to perform automated dispatch. Our system achieves human-level accuracy and has already been deployed for three customers in production. However, there are still some areas in the system like rule engine which need human intervention and can be automated. In future, we want to solve the problem of automatically extracting rules based on data from misclassified emails. We would also like to handle concept drift in utterances for better retraining. Last but not the least, we need to enhance our assignment algorithm to handle email attachments.

References

1. Agarwal, S., Aggarwal, V., Akula, A.R., Dasgupta, G.B., Sridhara, G.: Automatic problem extraction and analysis from unstructured text in it tickets. IBM J. Res. Dev. **61**, 4–41 (2017)
2. Agarwal, S., Sindhgatta, R., Sengupta, B.: SmartDispatch: enabling efficient ticket dispatch in an it service environment. In: 18th ACM SIGKDD 2012 (2012)
3. Dasgupta, G.B., Nayak, T.K., Akula, A.R., Agarwal, S., Nadgowda, S.J.: Towards auto-remediation in services delivery: context-based classification of noisy and unstructured tickets. In: Franch, X., Ghose, A.K., Lewis, G.A., Bhiri, S. (eds.) ICSOC 2014. LNCS, vol. 8831, pp. 478–485. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45391-9_39
4. Kadar, C., Wiesmann, D., Iria, J., Husemann, D., Lucic, M.: Automatic classification of change requests for improved it service quality. In: 2011 SRII (2011)
5. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms (2004)
6. Mitchell, T.M.: Machine Learning, 1st edn. McGraw-Hill Inc., New York (1997)
7. Parvin, H., Bose, A., Van Oyen, M.P.: Priority-based routing with strict deadlines and server flexibility under uncertainty. In: WSC 2009 (2009)
8. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP) (2014)
9. Potharaju, R., Jain, N., Nita-Rotaru, C.: Juggling the Jigsaw: towards automated problem inference from network trouble tickets. In: USENIX (2013)

10. Shao, Q., Chen, Y., Tao, S., Yan, X., Anerousis, N.: EasyTicket: a ticket routing recommendation engine for enterprise problem resolution. In: VLDB 2008 (2008)
11. Shao, Q., Chen, Y., Tao, S., Yan, X., Anerousis, N.: Efficient ticket routing by resolution sequence mining. In: 14th ACM SIGKDD (2008)
12. Zeng, C., Zhou, W., Li, T., Shwartz, L., Grabarnik, G.Y.: Knowledge guided hierarchical multi-label classification over ticket data. IEEE Trans. Netw. Serv. Manag. **14**(2), 246–260 (2017)