Leveraging AI in Service Automation Modeling: from Classical AI Through Deep Learning to Combination Models

Qing Wang[1][0000−1111−5421−5515], Larisa Shwartz[1][0000−0001−5878−0765], Genady Ya. Grabarnik[2][0000−0001−8068−0920], Michael Nidd[3][0000−0001−7379−2852], and Jinho Hwang[1][0000−0001−8595−7431]

[1] IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA
{qing.wang1@ibm.com}, {lshwart, jinho}@us.ibm.com
[2] Dept. Math & Computer Science, St. John's University, Queens, NY 11439, USA
grabarng@stjohns.edu
[3] IBM Research – Zurich, 8803 Rueschlikon, Switzerland
mni@zurich.ibm.com

Abstract. With the advent of cloud, new generations of digital services are being conceived to respond to the ever-growing demands and expectations of the market place. In parallel, automations are becoming an essential enabler for successful management of these services. With such importance being placed on digital services, automated management of these services – in particular, automated incident resolution – becomes a key issue for both the provider and the users. The challenge facing automation providers lies in variability and the frequently changing nature of the monitoring tickets that provide the primary input to automation. Despite the value of the correct automation at the correct time, it is also important to remember that triggering an incorrect automation may damage the smooth operation of the business. In this paper, we discuss AI modeling for automation recommendations. We describe a wide range of experiments which allowed us to conclude an optimal method with respect to accuracy and speed acceptable to service providers.

Keywords: Classical and deep learning models · Combination models · Multiclass text classification · AI for service automation.

1 Introduction

Providing Service Management at scale requires automation. For decades, system administrators have had scripts to automate routine repairs, and have also employed automated monitoring systems to raise alerts when repair is called for. While some of these alert conditions require individual expert attention, service delivery professionals have confirmed that many of these conditions can be handled with the execution of the correct well-written script. These scripts are combined into automation library for IBM Automation service. The first step in managing alert conditions is to filter and collect the alerts into the "trouble

tickets" that act as work orders for repair. IBM service delivery teams have libraries of regular expressions that match ticket details to automation that may solve the problem. These regular expressions are applied to ticket raised by monitoring systems, which encode both specific fields (machine address, timestamp, severity, etc.) and also "free text" summary information.

In the normal course of business, new monitor systems are added, either through organic growth or by mergers and acquisitions. These new systems will usually be generating tickets about similar issues, so existing automation would often be appropriate for its resolution, but ticket formats may be sufficiently different to not match the corresponding regular expression. Figure 1 shows an example of two different ticket problems that were detected and auto-ticketed by the monitoring system, and subsequently successfully closed by the same automation. Over the course of several years IBM Global Services established around 25,000 regular expressions. This approach is very effective, but it is difficult to maintain. To assist IBM automation services we created "matching service" that utilizes artificial intelligence for choosing an automation for a monitoring ticket with ensuring the following challenge:

How does the matcher service effectively achieve and maintain high accuracy on noisy tickets while automatically adapting to an introduction of a new or changed ticket contents? Since executing an inappropriate automation on a server may cause damage, our recommendation has to be highly accurate. Informative, discriminating and independent features can greatly promote the performance of classification or regression algorithms, while irrelevant features decrease it. Unfortunately, real-world tickets (seen in Figure 1) contain various time-format, numeric and domain-specific terms, and unknown text snippets that make them too noisy to be interpretable. Thus, effective feature-selection [1] often involving immense efforts on text preprocessing becomes a crucial step to make this learning task more efficient and accurate. Feature selection is performed manually for classical classification solutions [2], but the changes in ticket style and content over time means that optimal feature-selection will also change, leading to degraded performance. Reasonable techniques to address this issue include the direct use of deep neural networks [3–5] without manual feature-selection, or using convolutional neural networks as input for classical AI models (i.e., combination models) to automatically learn a generic feature representation.

In the course of our work with service delivery teams to improve automation, we have studied the use of Machine Learning (ML) to identify methodology that would help us to address the tickets that were not recognized by the existing rules. We have conducted a comparative study on a wide range of machine learning approaches including classical AI, deep learning and combination methods. They are described in details in Section 3. We have large volume of tickets from IBM Services which were successfully and automatically remediated, so we have large amount of task specific labeled dataset for our experiments.

As we present in this paper, our initial solutions have performed extremely well on the labeled test data. A significant challenge when evaluating recom-

ALERT_KEY	XXX_logalrt_x072_aix		AUTOMATION	Disk Path Checker	
AGENT	CUSTOMER_CODE	ALERT_GROUP	COMPONENT		OSTYPE
EIF Probe on xxxxxx	XXX	ITM_XXX_LOGFILEMONITOR_LOG_FILE	Computer System		Generic
TICKET_GROUP	PRIORITY	HOSTNAME	IP_ADDRESS		SUBCOMPONENT
I-XXX-XXX-DS	PX	XXX	XXX.XXX.XXX.XXX		Log
TICKET SUMMARY	LogEvent: Thu Apr 4 02:00:01 EDT2019,xxxxxx,pcmpath,disk,fiber adapters missing or failed on server		TICKET DESCRIPTION	4 xxxxxx GENERIC LOG /TMP/xxxxxx.LOG AIX is CRITICAL **	

ALERT_KEY	XXX_erp_xlo2_std		AUTOMATION	Disk Path Checker	
AGENT	CUSTOMER_CODE	ALERT_GROUP	COMPONENT		OSTYPE
EIF Probe on xxxxxx	XXX	ITM_XXX_LOGFILEEVENTS	Operating System		AIX
TICKET_GROUP	PRIORITY	HOSTNAME	IP_ADDRESS		SUBCOMPONENT
NUSN_XXDISTOPS	PX	XXX	XXX.XXX.XXX.XXX		ErrorReport
TICKET SUMMARY	Errpt log entry: xxxxxx 0404051519 P H - PATH HAS FAILED - hdisk21		TICKET DESCRIPTION	xxx.xxx.xxx.IBM.COM AIX ERRORREPORT /VAR/ADM/RAS/ERRLOG.HDISK21 UNIX is CRITICAL **	

Fig. 1. Two different monitoring tickets and the same matching automation.

mendations, however, is that this data is all taken from the tickets that were automated through regular expression match. When we apply these models to tickets that did not match any of those expressions, the only way to evaluate the recommendation is for a subject matter expert (SME) to review it. We have done this and, as we will present here, the results are very promising.

We have incorporated these models into "matching service" for IBM service delivery: automation service makes a call to matching service for the tickets that do not match any of the handcrafted rules. If our system identifies a ticket that our model strongly associates with an existing automation (for an audited set of automations that are deemed "safe" enough to run prospectively), that automation will be run. After it is run, the ticket condition is reevaluated, and the ticket is either resolved automatically or escalated.

This "matching service" is part of Cognitive Event Automation (CEA) framework [6] that focuses on service management optimization and automation with the goal of transforming the service management lifecycle to deliver better business outcomes through a data-driven and knowledge-based approach. The framework relies on novel domain specific techniques in data mining and machine learning to generate insights from operational context, among them generation of predictive rules, deep neural ranking, hierarchical multi-armed bandit algorithms, and combination models for the automation matching service that is the focus of this paper.

This paper presents the comprehensive performance comparison on a wide range of popular classical AI, deep learning and combination classifiers that has guided us in model selection for an IBM automation service implementation. We have established that under the time constraint, classical AI models perform best when size of training data is small but with the drawback that features must be hand-crafted. Deep learning models will also perform well when the training data is large enough, but combination models have the best performance for large dataset size and without a need for manual feature-engineering.

The remainder of this paper is organized as follows. An overview of the CEA system is presented in Section 2. In Section 3, we provide the mathematical for-

malization of the problem and methodologies. Section 4 describes a comparative study conducted over real-world ticket data to show the performance of the proposed methodologies. Finally, a systematic review of related work is presented in Section 5, and Section 6 concludes the work and provides directions for future research.

2 System Overview

2.1 Service Management workflow

A typical workflow for service management usually includes six steps. (1) An anomaly is detected, causing the monitoring system to emit an event if the anomaly persists beyond a predefined duration. (2) Events from an entire environment are consolidated in an enterprise event management system, which makes the decision whether or not to create an alert and subsequently an incident ticket. (3) Tickets are collected by an IPC (Incident, Problem, and Change) system [7]. (4) A monitoring ticket is identified by automation services for potential automation (i.e., scripted resolution) based on the ticket description. If the automation does not completely resolve the issue, this ticket is then escalated to human engineers. (5) In order to improve the performance of automation services and reduce human effort on escalated tickets, the workflow incorporates an enrichment system like CEA that uses Machine Learning techniques [8–10] for continuous enhancement of automation services. Additionally, the information is added to a knowledge base, which is used by automation services as well as in resolution recommendation for tickets escalated to human engineers. (6) Manually created and escalated tickets are forwarded to human engineers for problem determination, diagnosis, and resolution, which is a very labor-intensive process.

2.2 System Architecture

The microservice is a new computing paradigm that overcomes the limitations of the monolithic architectural style. The microservice architecture consists of a collection of loosely coupled services, each of which implements a business function. The microservice architecture enables scalability, flexibility, and also continuous devops (development and operations) of large, complex applications. We use the microservices framework to support our data processing components.

Figure 2 illustrates the CEA system architecture. In general, the system consists of two types of services: offline processing services build a knowledge corpus and models; and inline processing services apply reasoning to gathered runtime data, using the models and knowledge corpus built offline. The offline processing services take advantage of AI that incorporates feedback loop analysis, automatically re-training models periodically or on demand if the monitoring system undergoes a significant change. The system continuously improves recommendations for automated resolution of monitoring tickets. The inline system has a number of services: Correlation and Localization Service (CLS) identifies clusters

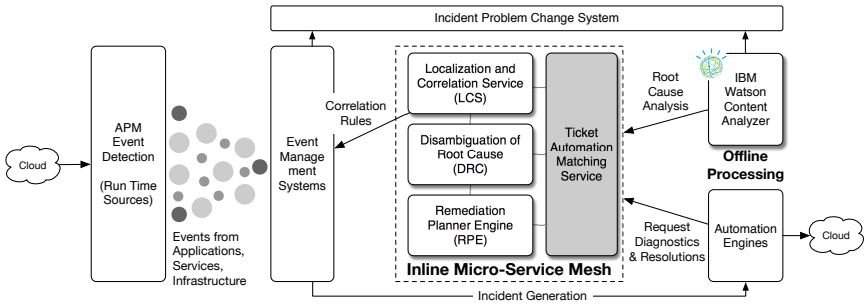

Fig. 2. CEA System Architecture

of symptoms attributable to a complex incident; Disambiguation of Root Causes (DRC) is built as a recommender system that enriches ticket data with possible root causes, identifies steps necessary for full diagnosis and provides optimal sequence of diagnostics and remediation steps using AI planning service. Finally, the ticket-automation matching service provides a service to identify the correct automation for a given ticket. The model built in this work allows the system to trigger the correct automation, despite the challenge of external influences that change ticket content and style over time.

3 Problem Definition & Methodology

In this section, we provide a mathematical formulation of the service automation modeling problem, followed by the description of proposed methodologies.

Based on a ticket's content, finding the best automation can be intuitively viewed as a multiclass text classification problem. A general framework for statistical learning (in particular for supervised learning) is as follows: a set of labeled training data $\mathcal{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^{N}$ is drawn independently according to a distribution $P(\mathbf{x}, y)$ on $(\mathbb{R}^d, Y)$, where $\mathbf{x},\ \mathbf{x}_t \in \mathbb{R}^d,\ y,\ y_t \in Y = \{1, 2, \cdots, K\}$ is the ground truth class label for $\mathbf{x}_t$. For example, a ticket (seen in Figure 1) can be represented by a set of features $\mathbf{x}$ using NLP techniques [11] and a known class label $y = \{Disk\ Path\ Checker\}$. We assume that $g(\mathbf{x}) : \mathbb{R}^d \mapsto Y$ be a prediction function.

A loss function $\ell : Y \times Y \mapsto \mathbb{R}$, usually a positive function, measures error between prediction and actual outcomes. Cross entropy loss (log loss), mean absolute error (L1 loss), and mean squared error (L2 loss) are three popular loss functions that calculate the error in different ways. Expectation L of ℓ by measure P is called expected loss:

$$L(g) = E_{(\mathbf{x},y)\sim P}[\ell(g(\mathbf{x}), y)].$$

The prediction function g is usually found from minimization of expected loss L:

$$g = \arg\min_{g} L(g). \tag{1}$$

Sometimes we choose g from a family of functions $g(\Theta)$, where Θ is a set of parameters. Equation 1 in this case becomes $\Theta^* = \arg\min_\Theta L(g(\Theta))$.

Let $p(k|\mathbf{x}) = P(Y = k|X = \mathbf{x})$ be the conditional probability of getting label k given $X = \mathbf{x}, k = 1, \ldots, K$. For equal misclassification costs the loss function $\ell(y, g(\mathbf{x})) = \chi(y \neq g(\mathbf{x}))$, with χ being indicator function of the set $\{y \neq g(\mathbf{x})\}$. Then the best classification (Bayes) rule minimizes expected misclassification

$$g_B(\mathbf{x}) = \underset{k=1,\ldots,K}{\arg\min}[1 - p(k|\mathbf{x})] = \underset{k=1,\ldots,K}{\arg\max}\, p(k|\mathbf{x}).$$

In order to solve the real-world challenge, we have conducted a comparative study on a wide range of machine learning approaches including classical AI, deep learning and combination methods. Figure 3 shows an overview of how different models are used to address multiclass text classification problem. In order to use classical AI models for this classification task, a feature extraction step must first transform the raw text data into informative feature vectors. In comparison, deep learning models can automatically perform feature engineering and classification tasks. Combination models unite the broad applicability of classical AI models with the automatic feature engineering of deep learning models to improve the performance. As background for the experimental design of Section 4, we will first outline a few classical AI learning algorithms, like support vector machines, ensemble methods, and deep learning approaches.

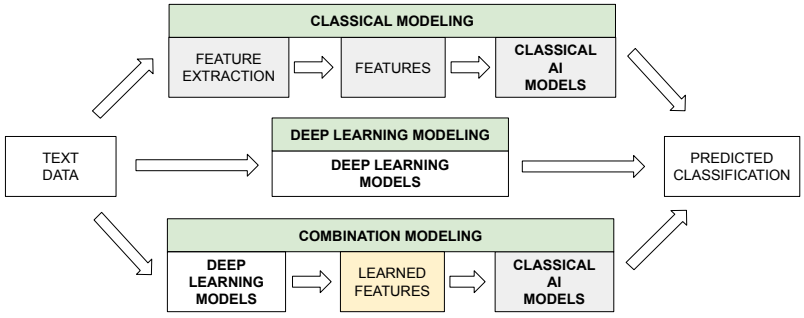

Fig. 3. Using modeling for the multiclass text classification: classical AI vs deep learning vs combination.

3.1 Classical AI: Support Vector Machines

Support Vector Machines (SVMs) are often considered as an efficient, theoretically solid and strong baseline for text classification problems [2]. SVMs were designed for binary classification. There are two main approaches for multiclass classification: one-vs.-all classifiers (OVA) [12] and multiclass SVMs [13]. One-vs.-all classifications simply construct K SVMs, where K is the number of classes, training k-th SVM with all of the training examples in the k-th class with positive labels, and all other examples with negative labels. In other words, the k-th SVM tries to find a hyperplane that satisfies the following constrained

optimization problem:

$$\min_{\omega^k, b^k, \xi^k} \frac{1}{2}(\omega^k)^T(\omega^k) + C\sum_{t=1}^{N}\xi_t^k,$$

subject to:

$$(\omega^k)^T\phi(\mathbf{x}_t) + b^k \geq 1 - \xi_t^k, \text{ if } y_t = k,$$
$$(\omega^k)^T\phi(\mathbf{x}_t) + b^k \leq -1 + \xi_t^k, \text{ if } y_t \neq k,$$
$$\xi_t^k \geq 0, t = 1, 2, \cdots, n,$$

where ω is the weight vector, b is the intercept of the hyperplane, $\phi(\bullet)$ is the function mapping the feature vector $\mathbf{x}_t$ to a higher dimensional space and C is the penalty parameter. For a new text $\mathbf{x}$, the predicted $\hat{y}$ can be calculated as follows:

$$\hat{y} = \underset{k=1,2,\cdots,K}{\arg\max}\ ((\omega^k)^T\phi(\mathbf{x}) + b^k).$$

3.2 Classical AI: Ensemble Methods

Ensemble methods [14] are learning algorithms that train multiple classifiers, and then typically apply voting (weighted or unweighted) to make predictions for new data. It is well known that an ensemble method is generally more accurate than any single classifier. Useful categories of ensemble methods include *Bagging* and *Boosting*. In this review, we are considering *Random Forests* as an example of Bagging, and *eXtreme Gradient Boosting* as the one of Boosting.

Random Forests [15–17] is a highly accurate and robust machine learning algorithm, capable of modeling large feature spaces. A random forests is an ensemble of H decision trees $\{f_1, f_2, \cdots, f_H\}$, with each tree grown by randomly subsampling with replacement of the entire forest training set $\mathcal{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^N$, $\mathbf{x}_t \in \mathbb{R}^d$, and $y_t \in \{1, 2, \cdots, K\}$.

There are two types of nodes in a binary decision tree [18]. The leaf nodes of each tree are the estimates of posterior distribution p_k for all classes. Each internal node (i.e., *split node*) is associated with a test function Θ that best splits the space of training data. Often, Gini-impurity is used to choose the best test function Θ^*. During the training, each tree selects appropriate test functions and labels leaf node probabilities. For the evaluation, a test sample $\mathbf{x}$ is propagated through each tree leading to a classification probability $p_t(k|\mathbf{x})$ of the t-th tree. A forest's joint probability can be represented as follows:

$$p(k|\mathbf{x}) = \frac{1}{H}\sum_{h=1}^{H} p_h(k|\mathbf{x})$$

Therefore, given $\mathbf{x}$, the predicted class $\hat{y}$ is:

$$\underset{k=1,2,\cdots,K}{\arg\max}\ p(k|\mathbf{x})$$

eXtreme Gradient Boosting [19] has been widely recognized in many machine learning and data mining challenges and provided state-of-art results on many standard classification benchmarks. It is a tree ensemble model based on Gradient Boosting Machine and a set of Classification and Regression Trees (CARTs). Similar to RF, the final output is the sum of prediction of each tree with the given dataset $\mathcal{D}$.

$$\hat{y} = \sum_{h=1}^{H} f_h(\mathbf{x}), f_h \in \mathcal{F},$$

where K is the number of trees, $\mathcal{F}$ is the space of all possible CARTs, and f is an additive function in the functional-sapce $\mathcal{F}$. In order to learn the set of functions in the model, the objective function with a regulation term can be written as:

$$L = \sum_{t=1}^{N} l(y_t, \hat{y_t}) + \sum_{h=1}^{H} \Omega(f_h). \tag{2}$$

The loss function $l(\bullet)$ measures the difference between the target y_t and the prediction $\hat{y_t}$. The regularization function $\Omega(\bullet)$ penalizes the complexity of the model to avoid overfitting. Since the tree ensemble model including these functions (See Equation (2)) cannot be easily solved by traditional optimization methods, XGboost is trained in an additive manner. For the multiclass classification problem, we construct K binary classifiers using XGBoost model [20] and subsequently apply OvA.

3.3 Deep Learning: Convolutional Neural Networks

In recent years, deep neural networks have brought about a striking revolution in computer vision, speech recognition and natural language processing.

Convolutional neural networks (CNNs), one of the most promising deep learning network methods, has achieved remarkable results in computer vision. It also has been shown to be effective in many NLP tasks, such as text categorization, spam detection, and sentiment analysis [4]. CNN performs well feature extraction and classification tasks without any preconfiguration (i.e., without selecting features manually).

For an m-word input text (padded where necessary) $\mathbf{s}_t = \{w_1, w_2, \cdots, w_m\}$ with a label y_t, $t = 1, 2, \cdots, n$, and $y_t \in \{1, 2, \cdots, K\}$, each word is embedded as a q-dimensional vector, i.e., word vectors $\mathbf{w}_1, \cdots \mathbf{w}_m \in \mathbb{R}^q$. The $m \times q$ representation matrix is fed into a convolutional layer with a filter $\alpha \in \mathbb{R}^{l \times q}$ sliding over the text to produce a feature map. Let $w_{i:i+l-1}$ denotes the concatenation of words $w_i, w_{i+1}, \cdots, w_{i+l-1}$ with a length l. A convolution feature is calculated as follows. $c_i = f(\alpha \cdot w_{i:i+l-1} + \beta)$, where $\beta \in \mathbb{R}$ ia a bias term and $f(\bullet)$ is a non-linear function. This filter α slides over the text $\{w_{1:l}, w_{2:l+1}, \cdots, w_{m-l+1:m}\}$ resulting in a convolution feature map $\mathbf{c}_\alpha = [c_1, c_2, \cdots, c_{m-l+1}]$. A maxpooling layer is followed to capture the most important feature $\hat{c}_\alpha = max\{\mathbf{c}\}$ as the feature corresponding to the particular filter α. In practice, numerous filters with varying

window sizes are used to obtain multiple convolution features. Extracted features are passed to a fully connected softmax layer, whose output is the probability distribution over classification classes $\mathbf{o} = \{o_1, o_2, \cdots, o_K\}$. The predicted class is

$$\hat{y} = \underset{k=1,2,\cdots,K}{\arg\max} \{\mathbf{o}\}.$$

To avoid overfitting, dropout is employed.

In order to learn the parameters in this model, the objective loss function for mutlclass text classification is needed to be defined. Herein, we use the cross-entropy loss function:

$$\min_{\Theta} -\frac{1}{N} \sum_{t=1}^{N} \sum_{k=1}^{K} y_{t,k} \log(p_{t,k}),$$

where Θ denotes the parameters of the CNN model, $y_{t,k}$ is a binary indicator if class k is the correct classification for the t-th text, and $p_{t,k}$ is the predicted probability of text t is of class k through a softmax activation.

$$p_{t,k} = \frac{exp(g(\mathbf{s}_t; \theta_k))}{\sum_{k=1}^{K} exp((g(\mathbf{s}_t; \theta_k))}, \quad \sum_{k=1}^{K} p_{t,k} = 1.$$

3.4 Combination Models

Figure 4 shows the overall architecture of combination models for multiclass text classification tasks. CNN is used for learning feature representation in many applications. Convolution feature filters with varying widths can capture several different semantic classes of n-grams by using different activation patterns [4]. A global maxpooling function induces behavior that separates important n-grams from the rest. We propose a combination model that replaces the softmax layer of CNN with classical AI models for multiclass text classification problems. In this model CNNs perform as an automatic feature extractor to produce the learned (i.e., not hand-crafted) feature vectors from large text data. These feature vectors used in the classical classification models to provide more precise and efficient classification performance [21].

When comparing methods, it is important to remember the preprocessing and feature extraction effort. The classical methods in Subsection 3.1 and 3.2 require considerable effort for text preprocessing and feature extraction [5]. Ensemble methods also require either automatic feature extractors or manual selectors to transform the raw data into suitable internal feature vectors for further pattern recognition and classification [22]. These additional steps are not required for deep learning and combination models. Ability to evade preprocessing steps constitutes an important differentiation from classical methods.

4 Experiments

Executing an incorrect automation is potentially harmful to the service, so a classifier's recommendation has to be highly accurate.

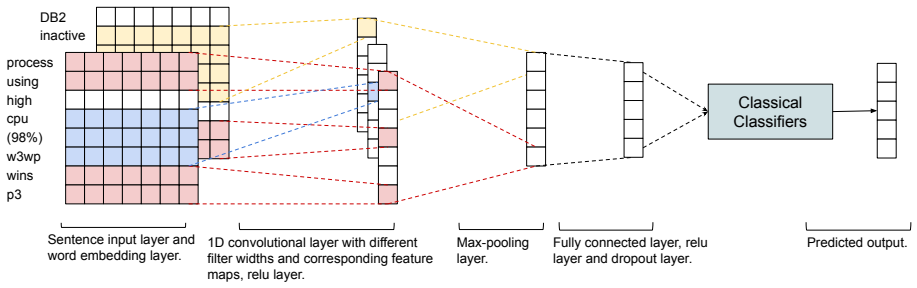

Fig. 4. Architecture of combination models on multiclass text classification tasks.

4.1 Dataset and Experimental Setup

Experimental ticket data is generated by a variety of monitoring systems and stored in the Operational Data Lake. This dataset contains $|\mathcal{D}| = 100,000$ tickets from Jan. 2019 to Apr. 2019, of which 80% is the training dataset, while the remaining are used for validation. There are 114 scripted automations (i.e., 114 classes/labels) in the dataset and a vocabulary V of size $|V| = 184,936$. To ensure validity of training, our ground truth dataset contains only tickets for which an automation was not only selected, but it also run and successfully resolved the ticket. Ticket information together with the automation name associated with the ticket constitutes the labeled dataset for training and testing.

The first step of preparing input data for the classical AI classifiers uses the bag of words method to represent feature vectors of each ticket after text preprocessing (stemming, lemmatization, stop words removal, etc.). Classical AI models usually work with relatively low-dimension attribute spaces, necessitating well-defined and highly informative attributes as coordinates of feature vectors. We use domain experts' assistance to determine such attributes for the dataset.

It is common to initialize deep learning models for NLP by using pre-trained word embeddings. This practice reduces the number of parameters that a neural network needs to discover from scratch. For the deep learning and combination models, it is a prevalent method to initialize pre-trained word vectors from an unsupervised neural language model to improve performance.

A weakness of this method lies with its ability to handle unknown or out-of-vocabulary (OOV) words. Our dataset (see Figure 1) contains critical domain specific information such as various machine address, domain-specific terms, and unknown technical script snippets for which there are no pre-trained data. A multilayer perceptron (MLP) is a deep artificial neural network composed of input layer, output layer and some number of hidden layers in between. In our case the layers are **word embedding layer**, **fully connected layer** and **dropout layer**. The introduction of a **dropout layer** is a regularization technique that reduces overfitting. CNN has an additional **convolutional layer**. Rectified Linear Unit (ReLU) is a commonly used activation function in deep learning models.

After some preliminary testing, we designed our primary experiments to randomly initialize all word vectors with a dimension of 300. We use filter size of

4, 5 with 64 feature maps each (for CNN only), dropout rate of 0.25, mini-batch size of 128, and epoch number of 20.

4.2 Evaluation Metrics

The accuracy (ACC) and F1-score (F1) are widely applied metrics for evaluating multiclass classifiers. We provide expression for the evaluation metrics in terms of Section 3, the problem definition, where $\mathcal{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^{N}$, y_t is one of K classes, and $g(\bullet)$ is the classifier. Multiclass accuracy is defined as an average number of correct predictions: $\text{ACC} = P_{(\mathbf{x},y)\sim\mathcal{D}}[g(\mathbf{x}) = y]$. F1-score for 2 classes of outcome (0, 1), is the harmonic mean of precision and recall

$$F1 = (\frac{1}{2}(\frac{1}{precison} + \frac{1}{recall}))^{-1} = \frac{2C_{1,1}}{C_{1,1} + C_{0,1} + C_{1,1} + C_{1,0}}, \tag{3}$$

where $C_{i,j}$ is a confusion matrix.

There are multiple ways to generalize Formula (3) to a multiclass F1-score. *Macro-averaged* F1-score (F1-macro), which emphasizes each class equally, has been demonstrated to be unbiased and provides an effective performance measure in multiclass settings [23]. For a classifier g, its (multiclass) confusion matrix $C[g] \in [0,1]^{K\times K}$ is defined as $C_{ij}[g] = P(y = i, g(\mathbf{x}) = j)$. Macro-averaged F1-score in terms of the confusion matrix can be written as:

$$F1_{macro}[g] = \frac{1}{K}\sum_{i=1}^{K}\frac{2\times C_{ii}[g]}{\sum_{j=1}^{K} C_{ij}[g] + \sum_{j=1}^{K} C_{ji}[g]}.$$

4.3 Results and Discussions

A wide range of strong classifiers across supervised, unsupervised, deep and combination AI models are evaluated for their performance on a real-world multiclass classification task. To ensure the fairness of comparisons, the accuracy and F1 score for each model are calculated from the average results of 5-fold cross validation (CV). The comparison of performance of ACC, F1-macro and time cost has been shown in Table 1, where time cost is defined as the time required to train a good model on the dataset once for each model. The parameters in XGBoost are learning rate is 0.1, number of estimators is 100, booster is gradient boosting tree, and maximum depth is 4. For Random Forests, the number of estimators is 100 as well. All algorithms are implemented using Python 1.8. All empirical experiments are running on MacOS 10.14 with CPU only.

These results demonstrate that classical classification models such as SVM, Random Forests, and XGBoost have best performance on small datasets, but need handcrafted features. The training time for SVM increases exponentially with data size, while those of Random Forest and XGBoost increase linearly. Clearly, XGBoost has the best accuracy and F1-macro scores on the $4k$ dataset from Table 1.

Table 1. Performance comparison on **Accuracy (ACC(in percent %)), F1-macro (F1(in percent %)), Time Cost (t(in seconds)).**

	$\|\mathcal{D}\| = 4,000$			$\|\mathcal{D}\| = 20,000$			$\|\mathcal{D}\| = 100,000$		
Models	**ACC(%)**	**F1(%)**	**t(s)**	**ACC(%)**	**F1(%)**	**t(s)**	**ACC(%)**	**F1(%)**	**t(s)**
Linear SVM [24]	97.95	88.18	3.60	99.09	92.42	42.81	99.53	93.69	671.97
Decision Tree [25]	97.65	84.71	0.11	98.58	79.96	1.13	98.15	62.74	16.43
KNeighbors [26]	93.75	75.20	0.15	97.39	78.01	3.72	97.80	80.46	99.29
K-Means [27]	< 50.00	–	78.01	< 50.00	–	625.13	< 50.00	–	5960.72
Random Forests [15]	97.65	89.26	1.15	99.05	92.28	13.25	99.29	93.39	251.26
XGBoost [19]	**98.50**	**91.79**	122.06	99.22	89.97	814.90	99.12	79.85	5345.62
MLP [3]	96.37	82.78	2.62	98.85	88.79	18.38	99.23	93.72	251.35
CNN [4]	97.12	81.10	8.65	98.92	88.40	52.87	99.39	93.16	601.11
CNN-SVM [21]	98.77	87.46	145.13	99.48	92.54	403.25	99.79	96.07	3019.69
CNN-Random Forests	98.75	87.92	148.24	99.54	90.01	148.24	**99.80**	**95.90**	1939.16
CNN-XGBoost [28]	93.50	67.41	260.19	97.70	72.15	1804.07	98.75	82.53	14035.91

Deep learning models perform better when the dataset is large, with the additional benefits that the models have a relatively short training time, and do not require feature engineering. Between deep learning models, CNN required more training time than MLP. And this can be attributed to the larger number of its parameters to learn.

Combination models have no need for handcrafted features, which allows them to support evolving sets of ticket templates and styles without the direct intervention of experts. Most of the combination models considered, CNN-SVM and CNN-Random Forests have better accuracy and F1-macro scores than SVM and Random Forests. This confirms that CNN models are good at automatically learning feature representation from a text data. CNN-Random Forest has the best overall performance among all the models including training time on a large dataset.

To summarize, we have explored a wide range of AI models and conducted a comparative study on our real-life data, aiming to provide guidance for model selection. While we find that all methods perform fairly well on different size datasets, the following insights have been learned from the experimental results:

1. Classical AI models perform well when the data size is small but they require handcrafted features.
2. Deep learning models achieve a better performance when the training data is large enough without feature engineerinng.
3. Combination models have the best performance on large dataset with no requirement for engineered features.

5 Related Work

The automation of service management [29] is largely achieved through the automation of subroutine procedures. Automated ticket resolution recommendation presents a significant challenge in service management due to the variability

of services, and the frequently changing styles and formats for monitoring tickets that provide an input to automation.

Traditional recommendation technologies in service management focus on recommending the proper resolutions to a ticket reported by the system's user. Recently, Wang et al. [30] proposed a cognitive framework that enables automation improvements through resolution recommendations utilizing the ontology modeling technique. A deep neural network ranking model [31] was employed to recommend the best top-n matching resolutions by quantifying the qualify of each historical resolution. In [8, 32], the authors leverage a popular reinforcement learning model (specifically, the multi-armed bandit model [9, 33]) to optimize online automation through feedback in automation services.

Text classification, including binary text classification (e.g., sentiment classification and spam detection) and multiclass text classification are the fundamental tasks of Natural Language Processing [34]. The aim of text classification is to assign binary classes or multiple classes $m > 2$ to the input text. Traditional approaches of text classification directly use sparse lexical features, such as bag of words model [35] or n-gram language model [36] to represent each document, and then apply a linear or nonlinear method to classify the inputs. Many machine learning techniques have been developed for the multiclass text classification problem, such as Support Vector Machines (SVM), K-Nearest Neighbors (KNN) and ensemble methods (XGBoost and Random Forests). Most recently, deep learning models have achieved remarkable success in learning document representation from large-scale data. Convolutional neural network (CNN) [4] approach is a very effective to feature extraction, and long short-term memory (LSTM) [37] is powerful in modeling units in sequence. In [28, 21], CNN-XGBoost and CNN-SVM models are used to improve the performance of image classification. We work with raw text data and use CNN in combination models for feature engineering.

Additional related work is provided in line with descriptions of relevant methods in Section 3.

6 Conclusion and Future Work

This paper addresses the automated management of digital services, more specifically an automated resolution of incidents. In the present context of an explosion of AI methods of multiple generations, it is important to choose optimal performing methods when implementing production systems.

In this paper we evaluate the performance of the three main types of the AI models: classical, deep learning and combination. Classical models include regular and ensemble methods. From a vast variety of existing methods, we have chosen those that are most promising in their class. For each model used, we have provided a short description and outlined its benefits and disadvantages.

We run wide range of experiments on real life data to find optimal modeling with respect to a number of metrics: accuracy, F1-macro score (measuring precision to recall ratio), running time and necessity of by-hand processing.

Our experimental results clearly show that under the time constraint, classical AI models perform best when the size of training data is small, and the combination methods are the best performing methods on large datasets of our data and they have no requirement for manual feature engineering. Following this conclusion, a Ticket Automation Matching Service has being implemented for the IBM Services production system.

For future work, we would like to employ the deep reinforcement learning method [22], transforming the backend offline model to an online one. Another important direction will be to build combination services that incorporate both deep learning and classical system together with common optimization problem and find global optimal parameters of the model.

References

1. G. Forman, "An extensive empirical study of feature selection metrics for text classification," *Journal of machine learning research*, vol. 3, pp. 1289–1305, 2003.
2. A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *arXiv preprint arXiv:1607.01759*, 2016.
3. D. W. Ruck, S. K. Rogers, and M. Kabrisky, "Feature selection using a multilayer perceptron," *Journal of Neural Network Computing*, vol. 2, no. 2, pp. 40–48, 1990.
4. Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.
5. X. Zhang and Y. LeCun, "Text understanding from scratch," *arXiv preprint arXiv:1502.01710*, 2015.
6. L. Shwartz, J. Hwang, H. Völzer, M. Nidd, M. G. Aguiar, M. V. L. Paraiso, and L. Valero, "Cea: A service for cognitive event automation," in *International Conference on Service-Oriented Computing*. Springer, 2018, pp. 425–429.
7. C. Zeng, T. Li, L. Shwartz, and G. Y. Grabarnik, "Hierarchical multi-label classification over ticket data using contextual loss," in *2014 IEEE NOMS*, 2014.
8. Q. Wang, T. Li, S. Iyengar, L. Shwartz, and G. Y. Grabarnik, "Online it ticket automation recommendation using hierarchical multi-armed bandit algorithms," in *Proceedings of the 2018 SIAM International Conference on Data Mining*. SIAM, 2018, pp. 657–665.
9. Q. Wang, C. Zeng, W. Zhou, T. Li, S. S. Iyengar, L. Shwartz, and G. Grabarnik, "Online interactive collaborative filtering using multi-armed bandit with dependent arms," *IEEE Transactions on Knowledge and Data Engineering*, 2018.
10. C. Zeng, Q. Wang, W. Wang, T. Li, and L. Shwartz, "Online inference for time-varying temporal dependency discovery from time series," in *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 2016, pp. 1281–1290.
11. A. Kao and S. R. Poteet, *Natural language processing and text mining*. Springer Science & Business Media, 2007.
12. L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, L. D. Jackel, Y. LeCun, U. A. Müller, E. Säckinger, P. Y. Simard *et al.*, "Comparison of classifier methods: a case study in handwritten digit recognition," in *International conference on pattern recognition*. IEEE Computer Society Press, 1994, pp. 77–77.
13. C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE transactions on Neural Networks*, vol. 13, pp. 415–425, 2002.
14. T. G. Dietterich, "Machine-learning research," *AI magazine*, vol. 18, no. 4, pp. 97–97, 1997.
15. L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.
16. A. Liaw, M. Wiener *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.
17. A. Prinzie and D. Van den Poel, "Random multiclass classification: Generalizing random forests to random mnl and random nb," in *International Conference on Database and Expert Systems Applications*. Springer, 2007, pp. 349–358.

18. J. Santner, M. Unger, T. Pock, C. Leistner, A. Saffari, and H. Bischof, "Interactive texture segmentation using random forests and total variation." in *BMVC*. Citeseer, 2009, pp. 1–12.
19. T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD*, 2016, pp. 785–794.
20. S. Madisetty and M. S. Desarkar, "An ensemble based method for predicting emotion intensity of tweets," in *International Conference on Mining Intelligence and Knowledge Exploration*. Springer, 2017, pp. 359–370.
21. A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2014, pp. 806–813.
22. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.
23. H. Narasimhan, W. Pan, P. Kar, P. Protopapas, and H. G. Ramaswamy, "Optimizing the multiclass f-measure via biconcave programming," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 2016, pp. 1101–1106.
24. B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*. ACM, 1992, pp. 144–152.
25. J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.
26. T. M. Cover, P. E. Hart *et al.*, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.
27. J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.
28. X. Ren, H. Guo, S. Li, S. Wang, and J. Li, "A novel image classification method with cnn-xgboost model," in *International Workshop on Digital Watermarking*. Springer, 2017, pp. 378–390.
29. Q. Wang, "Intelligent data mining techniques for automatic service management," in *FIU Electronic Theses and Dissertations, https://digitalcommons.fiu.edu/etd/3883*. FIU, 2018.
30. Q. Wang, W. Zhou, C. Zeng, T. Li, L. Shwartz, and G. Y. Grabarnik, "Constructing the knowledge base for cognitive it service management," in *2017 IEEE International Conference on Services Computing (SCC)*. IEEE, 2017, pp. 410–417.
31. W. Zhou, W. Xue, R. Baral, Q. Wang, C. Zeng, T. Li, J. Xu, Z. Liu, L. Shwartz, and G. Ya Grabarnik, "Star: A system for ticket analysis and resolution," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 2181–2190.
32. Q. Wang, C. Zeng, S. Iyengar, T. Li, L. Shwartz, and G. Y. Grabarnik, "Aistar: An intelligent system for online it ticket automation recommendation," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 1875–1884.
33. C. Zeng, Q. Wang, S. Mokhtari, and T. Li, "Online context-aware recommendation with time varying multi-armed bandit," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 2025–2034.
34. Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 1480–1489.
35. Y. Zhang, R. Jin, and Z.-H. Zhou, "Understanding bag-of-words model: a statistical framework," *International Journal of Machine Learning and Cybernetics*, vol. 1, no. 1-4, pp. 43–52, 2010.
36. P. F. Brown, P. V. Desouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai, "Class-based n-gram models of natural language," *Computational linguistics*, vol. 18, no. 4, pp. 467–479, 1992.
37. S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.