Ticket-BERT: Labeling Incident Management Tickets with Language Models

Zhexiong Liu, Microsoft, zhexiong@cs.pitt.edu Cris Benge, Microsoft, crbenge@microsoft.com
Stone Jiang, Microsoft, stonejiang@microsoft.com

Abstract

Software and hardware incident management has gained notable attention in enterprise-level products, where efficient ticket resolution services are critically needed due to large-scale ticket resolution demands. An essential aspect of prioritizing incident tickets for resolution is efficiently labeling tickets with fine-grained categories. This facilitates the accurate triage of incidents to responsible teams and ultimately accelerates ticket resolution. However, ticket data is often complex and poses several unique challenges for modern machine learning methods: (1) tickets are created and updated either by machines with pre-defined algorithms or by engineers with domain expertise that share different protocols, (2) tickets receive frequent revisions that update ticket status by modifying all or parts of ticket descriptions, and (3) ticket labeling is time-sensitive and requires knowledge updates and new labels per the rapid software and hardware improvement lifecycle. To handle these issues, we introduce Ticket-BERT, which trains a simple yet robust language model for labeling tickets using the proposed ticket datasets (i.e., D-Human, D-Machine, D-Mixture). Our extensive experiments demonstrate the superiority of Ticket-BERT over a diverse set of baselines and state-of-the-art text classifiers in Microsoft Cognitive Services. We further evaluate Ticket-BERT using human input on a set of hard-to-identify tickets and achieve outstanding performance. We deploy the proposed models in an active learning cycle that makes the model adapt to newly collected data and achieve greater accuracy with little annotating burden.

Keywords: Incident Management, Ticket Labeling, Ticket Triage, Ticket Dataset, Language Models, Active Learning.

1. Introduction

The explosive growth of digitization in enterprise-level products emphasizes the need for highly-reliable and functional services with efficient incident maintenance (e.g., efficiently resolving system interruptions or outrages). These problematic incidents are typically governed by incident management systems [1, 2], where hardware and software failures are documented as tickets using a series of formative descriptions and arrays of structured fields (e.g., date, titles, keywords, severity). The essential aspect of processing incident tickets lies in labeling issues with specific tags that precisely exhibit incident categories; however, this is often complicated by highly variable content, which makes it challenging to transfer knowledge among diverse environments. For example, the tickets could be automatically created by machines that monitor fatal errors or initiated by engineers. The mixture of human and machine-generated tickets have highly variable, issue-specific vocabularies. Moreover, tickets may receive frequent revisions on the status and descriptions based on newly updated information (e.g., ticket acknowledgment, mitigation, remediation). New knowledge and labels may also be needed when software and hardware upgrades to new versions or when new issues occur. Therefore, developing a domain-adaptive ticket labeling system is necessary for achieving acceptable performance.

Ticket classification has been widely explored in machine learning tasks, such as ticket assignment [3, 4], prioritization [5, 6], resolution [7], and duplicate identification [8]. These methods generally analyze an incident ticket solely by using its cleaned content (e.g., title, description, structured data) rather than focusing on the raw process of ticket creation, updates, and resolution that convey rich and valuable ticket information. Typically, tickets undergo multiple updates throughout their lifecycle; details are added or corrected by potentially various authors over time. However, little work has been done leveraging this update-related information to address the task of ticket labeling. Moreover, the existing machine learning approaches [5, 9, 10, 11] train ticket classification models with small-size domain-specific datasets that are not transferable and generalizable to large-scale real-world problems. To bridge these gaps, we develop new datasets using 76K raw tickets and ten fine-grained issue-specific taxonomies. Specifically, each ticket receives dozens of updates in its lifecycle, and we utilize a few of those to create D-Machine, D-Human, and D-Mixture datasets containing ticket titles, descriptions, and summaries generated by machines, engineers, and both, respectively. Afterward, we develop language models to identify issue-related ticket labels on these datasets. Note that our datasets are developed based on the first five ticket updates; thus, our trained model can label tickets as soon as they are created. This significantly helps identify new ticket labels in their early stage and ultimately provides formative information for engineers to accelerate ticket resolution.

Adapting machine learning models to text classification tasks has gained unprecedented interest in recent research [12, 13]. Multiple text representation techniques and classification methods have been proposed. Typically, text classification involves three stages: (1) feature extraction that utilizes salient information in raw text to present text semantics, such as TF-IDF and Bag-of-Words (BoW) features, (2) data down-sampling that aims to reduce the dimensions of text vector representations or leverage fewer samples to train lightweight classifiers that can be efficiently deployed in low memory systems, and (3) supervised or unsupervised model training that either leverages many class annotations to conduct text classification or employs cluster-based methods to group similar text [14]. Generally, these methods often perform poorly when the text data (e.g., raw tickets) are not clean or have large variability. For example, the ticket descriptions regarding software incidences would be significantly different from those that address hardware incidences in terms of text semantics and syntactic structures. To address this variability, end-to-end and domain-adaptive methods are needed. In a previous study, transformer-based language models have demonstrated strong robustness in learning context-aware text information from diverse environments [15, 16, 17], but their application in ticket labeling tasks remains relatively unexplored. A scarcity of labeled data for issue-specific incidences in real-world ticketing systems has limited the ability to develop effective deep learning models for classification. We overcome this challenge by introducing large-scale datasets and the first high-performance Ticket-BERT models.

Against this backdrop, we study various ticket labeling classification methods and demonstrate our language models' superiority over a set of strong baselines on all the proposed ticket datasets. Our contribution is fourfold: (1) We develop the first large-scale real-word ticket dataset that contains ticket titles, summaries, and multiple updates of ticket descriptions, (2) We develop highly-effective language models that gain state-of-the-art performance on all the proposed datasets over multiple strong baselines, including Microsoft Cognitive Service, (3) We develop a novel text sampling strategy that leverages auxiliary text to greatly boost ticket labeling performance on the difficult D-Human dataset, and (4) we proposed an active-learning pipeline in the real incident management systems that iteratively learns models from new data. This ensures domain-adaptation for the Ticket-BERT when handling diverse incident issues.

2. Related Work

2.1. Text Representation

Text representation leverages computational methods to transform text into vectors that serve as features for machine learning [18, 19]. Work focusing on extracting useful information from text comes primarily from three fields: weighted words, word embedding, and linguistic features [6]. The most common weighted features are Bag-of-Words (BoW) [20] and TF-IDF [21], which represent words as explainable one-hot-encoded vectors; unfortunately, these representations cannot fully capture the word semantics in the context. Word embedding [22] addresses this limitation by considering word occurrence and co-occurrence. Specifically, each word is represented as a learned vector that conveys information from surrounding words. The representative word embedding methods include Word2Vec [23], GloVe [24], FastText [25], and contextualized word representations [26, 27]. In addition to word embedding, linguistic features are also useful. For example, synonymy and hypernymy are effective methods to increase variants of the text; thus, they can be used as features to measure the text diversity [28]. Recent research on linguistic features are described in [6], such as [29] that uses parts-of-speech (PoS) to count tags and conducts ticket classification. However, existing word embedding and linguistic feature methods are not applicable to ticket labeling tasks for the following reasons: (1) pre-trained word embedding models are not trained on the ticket-related vocabulary containing issue-specific words, and (2) extracting linguistic features requires input text to be either well-structured or have rich semantic indicators [30] whereas ticket descriptions usually have a large amount of unstructured text (e.g., code comments, database queries, cornerstones from different engineers regarding the ticket issues). To address these issues, we apply simple yet effective TF-IDF and BoW features in the baseline models and only use clean text as input for Ticket-BERT models.

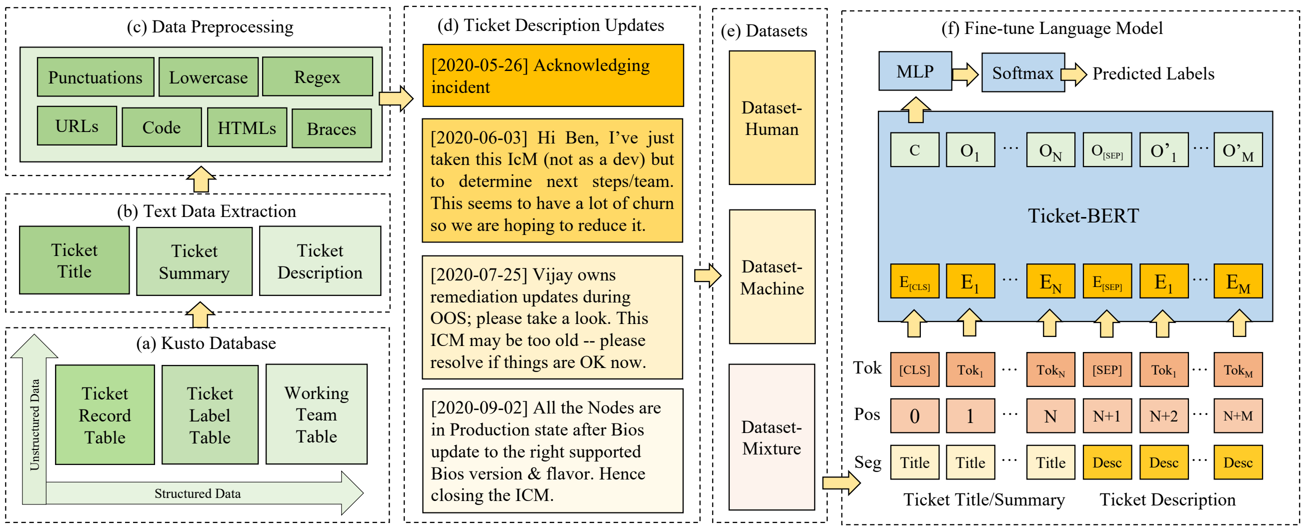

(c) Data Preprocessing
Punctuations
Lowercase
Regex
URLs
Code
HTMLs
Braces
(b) Text Data Extraction
Ticket Title
Ticket Summary
Ticket Description
(a) Kusto Database
Unstructured Data
Ticket Record Table
Ticket Label Table
Working Team Table
Structured Data
(d) Ticket Description Updates
[2020-05-26] Acknowledging incident
[2020-06-03] Hi Ben, I've just taken this IcM (not as a dev) but to determine next steps/team. This seems to have a lot of churn so we are hoping to reduce it.
[2020-07-25] Vijay owns remediation updates during OOS; please take a look. This ICM may be too old -- please resolve if things are OK now.
[2020-09-02] All the Nodes are in Production state after Bios update to the right supported Bios version & flavor. Hence closing the ICM.
(e) Datasets
Dataset-Human
Dataset-Machine
Dataset-Mixture
(f) Fine-tune Language Model
MLP
Softmax
Predicted Labels
Ticket-BERT
Tok
Pos
Seg
Ticket Title/Summary
Ticket Description

Figure 1. Ticket labeling framework: a) Kusto database that stores structured and unstructured data, including ticket records, incident labels, on-call teams, etc. (b) We pulled 76K raw tickets from Kusto, including ticket titles, summaries, and descriptions. (c) The tickets are preprocessed and cleaned using predefined rules. The clean tickets in (d) show four out of dozens of ticket updates that reflect the new ticket status. (e) We build three datasets using ticket descriptions: D-Human where ticket descriptions/updates are entered by humans, D-Machine where ticket descriptions/updates are generated by machines, D-Mixture where it is a mixture of D-Human and D-Machine (Sec. 3). (f) Finally, we fine-tune a Ticket-BERT using ticket title/summary as an auxiliary prompt-prefix that concatenates with ticket descriptions (Sec. 4).

2.2. Ticket Labeling

Ticket labeling is the task of assigning specific labels to a given ticket which provides an informative overview of the incident issues therein. General ticket labeling can be done with rule-based methods and machine learning methods: (1) rule-based methods develop a set of rules to determine ticket labels, such as keyword searching [31] or regular expression matching [32]. These methods are favored by their transparency, explainability, and simplicity [33, 34]. Additionally, rule-based methods can be integrated into machine learning approaches, such as decision trees that develop multiple rules to determine tree splitting [35, 36]. Note that rule-based ticket labeling may work well on a small set of rules, such as the keywords that address one incident may also apply to another, but could have reduced efficiency when handling a large number of diverse tickets governed by more complex rules. (2) machine learning methods have demonstrated effectiveness on multiple classification tasks [37]. For example, [38] shows that machine learning methods have advantages to discriminate bugs from incident issues. [39] develops machine learning classifiers for identifying incident issues and reveals that developers and maintainers often assign wrong categories to the tickets due to potential intervention from implicit unstructured data (e.g., implicit ticket description addressing an incident issue). To improve this, [40] introduces an ensemble method that leverages both structured and unstructured data to reduce the noise of misclassification via a multi-stage classification approach. However, it is challenging to determine the best machine learning algorithm for a particular application [6] in real-world scenarios. To address these issues, we elected to build end-to-end deep learning language models that avoid the manual creation of complicated rules and feature engineering preprocessing yet achieve state-of-the-art performance for the ticket labeling task.

2.3. Deep Language Models

Recently, deep learning methods have gained unprecedented prevalence in natural language processing (NLP) [37, 41]. For example, Long short-term memory (LSTM) models [42] have been trained to solve sequence tagging [43], semantic parsing [44], sentiment analysis [45], and text generation [46]. With regard to text representation, [27] proposes the Embedding from Language Models (ELMo) that learns word vectors by a bidirectional LSTM, where weighted layers are combined as embedding to represent text. Although ELMo has bidirectional neurons that look forward and backward in the text sequence, it is unable to take advantage of both contexts simultaneously, thus its ability to understand text content is limited. To improve this, [47] proposes another Bidirectional Encoder Representations based on Transformers (BERT) that addresses Masked Language Model (MLM) and Next Sentence Prediction (NSP) tasks for training language models. Previous work leverages BERT to conduct classification in computer vision and NLP [48, 49, 50, 51], and achieved excellent results; however, there is no prior work, to the best of our knowledge, that employs pre-trained BERT models to conduct ticket labeling because ticket data is often domain-specific, complex, and there exists no large-scale labeled datasets publicly available. Smaller datasets may be scraped in individual cases, but several reasons exist as to why these are often not enough to fine-tune a large transformer like BERT. First, ticket data is combination of machine-generated text and natural language. BERT was pre-trained on the latter, and may require more data to learn the nuisances of fragmented text (sometimes even in table format) generated by machines. Second, we observe that some tickets are information sparse, which means that we have a set of hard-to-identify labels. This is one of the reasons why we believe the model would need an active learning cycle [3]. In this work, we develop the first TicketBERT for ticket labeling using our proposed datasets. Nevertheless, directly fine-tuning large language models on the dataset (i,e, D-Machine, D-Human, and D-Mixture) that share different vocabulary (e.g., ticket-related tokens) may not perform well. Ticket text uses domain-specific words and/or abbreviations to present incident issues. To solve for this, we borrow the idea from prompt learning [52] that helps language models learn factual knowledge [53], and develop novel training strategies to improve ticket labeling performance with language models. Specifically, we take the title and summary as prompt-prefix and concatenate to ticket description. This simple yet effective strategy greatly boosts ticket labeling performance on the difficult D-Human dataset.

3. Dataset

3.1. Data Source

Presently, we are not aware of a labeled large-scale dataset for incident management. As a result, we curated the first ticket dataset using 76K raw tickets pulled from Microsoft Kusto [1] between June 2020 and June 2022. The raw data contains 5 fields, including Title, Description, Summary, Edited by, Timestamp. To preprocess the data, we cleaned URLs, Code, HTML Tags, Tables, and other metadata, in order to parse out plain text in the ticket in titles, descriptions, and summaries. Given that each ticket T receives multiple (m) modifications in their lifecycles, as shown in Figure 1 (d), either from machines or from engineer, we define the ticket update T_i as the i-th record of modification on the ticket description. For example, Table 2 shows the parts of ticket updates, where the ticket lifecycle range from May 2020 to September 2020 which suggests it is a complex incident that costs months to resolve. Typically, a human-entered/updated T_i has high variability and often addresses diverse incident issues with specific details. Machine-generated/updated T_i focuses on a smaller number of incident failures that results from automation. Table 1 shows that the human entered T_i are close to natural language, full of transition and reasoning while machine-generated T_i shares similar patterns but has low variability.

[1] https://docs.microsoft.com/en-us/azure/data-explorer/kusto/query/

3.2. Dataset Construction

Note that each ticket has multiple description updates T_i where $i = 1, 2, \dots, m$, and all the updates describe the same incident issues in different timelines. For simplicity, we draw one out of m updates as a representative description of each ticket to build our datasets. Specifically, we drop short ticket descriptions that have no salient semantic information about incident issues, such as *"incident acknowledged"*. Here the short ticket descriptions are defined as text with less than n characters. To investigate the best value of n, we show the frequency distribution of T_i with respect to its length (n characters) in Table 3. Interpreting this table, 99.7% of data is drawn from the first update T_1 (0.1% from the second update T_2) of the ticket descriptions if we draw the first long text with more than $n = 10$ characters. In dataset construction, we develop D-Human dataset that only draws ticket descriptions entered by humans, D-Machine that only draws ticket descriptions generated by machines, and D-Mixture that is a hybrid version of D-Human and D-Machine. Specifically, we set $n = 50$ which ensures the collected ticket descriptions are mostly from the first 3 descriptions (T_1, T_2, T_3). This way, the models trained on these datasets are able to label tickets as soon as the tickets are created, which significantly helps identify new ticket labels in their early stage (e.g., first 5 updates) and ultimately help accelerate ticket resolution.

3.3. Data Distribution

In order to specifically identify incident issues, we develop 10 fine-grained ticket labels shown in Table 4. Figure 2 shows label distribution in D-Human dataset, where *Cloud-Net FPGA Alerts* issues are the most. These labels were developed based on feedback from subject matter experts within the Azure Hardware space. Primarily, the focus was on identifying systematic issues, and then bucketing these issues based on previous root cause, or potential path to resolution. The ticket labels shown are the best estimates for potential problems underlying our IcM tickets. In this dataset, we only have 10 labels total for all tickets. Statistically, D-Human contains 20,142 tickets for training, 2,239 for validation, and 5,596 for test. D-Machine contains 39,391 tickets for training, 4,377 for validation, and 10,942 for test. D-Mixture contains 40,611 tickets for training, 4,513 for validation, and 11,282 for test.

Update	Human entered/updated ticket descriptions	Machine generated/updated ticket descriptions
T_1	@NAMEMASKED - Assigning to you to begin investigation into this issue.	[automated] Update: Node is currently out-for-repair. Need to validate proper repair action.
T_2	If the goal is to move the from AU (legacy boot mode) to AO flavor (UEFI boot mode), there should be ...	[automated] Update: This IcM has not been updated in the past 7 days.
T_3	All the Nodes are in Production state after Bios update to the right supported Bios version & flavor. Hence closing the ICM.	The severity in this IcM incident is inherited from the severity of the GDCO Ticket

Table 1. Human-entered ticket descriptions (left) v.s. machine-generated ticket descriptions (right).

Time	Ticket Descriptions
2020-05-26	Acknowledging incident
2020-06-03	Hi Ben, I've just taken this IcM (not as a dev) but to determine next steps/team. This seems to have a lot of churn so we are hoping to reduce it...
2020-06-15	If the goal is to move the from AU (legacy boot mode) to AO flavor (UEFI boot mode), there should be an issue...
2020-07-25	Vijay owns remediation updates during OOS; please take a look. This ICM may be too old - please resolve if things are OK now. Thanks!

Table 2. An example of parts of ticket description updates.

Text	Ticket description updates (T_i)					
Length (n)	T_1	T_2	T_3	T_4	T_5	others
10 (char)	99.7%	0.1%	0.0%	0.0%	0.0%	0.2%
20 (char)	30.7%	68.8%	0.1%	0.0%	0.0%	0.2%
50 (char)	29.3%	65.6%	3.1%	0.7%	0.2%	1.0%
100 (char)	27.1%	64.9%	3.1%	1.5%	0.6%	2.8%
200 (char)	25.7%	64.1%	2.3%	1.4%	0.9%	5.6%

Table 3. The percentage frequency distribution of drawn examples that are from T_i-th update of the ticket description with longer than n-character.

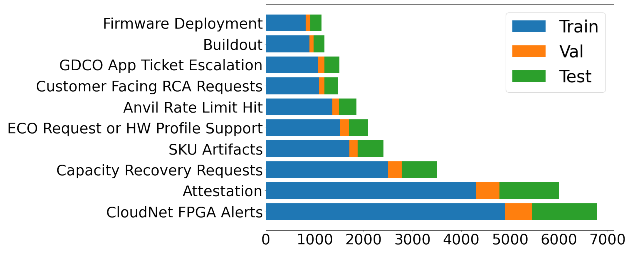


Figure 2. Incident label distribution (numbers) in D-human dataset.

4. Methods

In this section, we introduce the ticket labeling framework shown in Figure 1, where we develop new ticket datasets and fine-tune Ticket-BERT on proposed dataset and achieve state-of-the-art performance. In particular, We fine-tune the BERT-base [47] models with additional multi-layer perception and a softmax layer for multiclass classification, as shown in Figure 1 (f), on the proposed datasets for multi-class classification. For baselines, we leverage TF-IDF and BoW features to train Naive Bayes and Logistic Regression models, as well as other competitive baseline classifiers on the Language Studio at Microsoft Cognitive Science[2].

[2]Microsoft Language Studio `https://docs.microsoft.com/en-us/azure/cognitive-services/language-service/language-studio`

4.1. Baseline Models

NB-BoW: Naive Bayes with Bow features. In the Bag-of-Word (BoW) representation, we construct a vocabulary containing all unique words found in the training ticket text and use a binary encoder to represent the word presence or absence. In particular, a word is assigned one if in the vocabulary; otherwise, zero. We use Naive Bayes, which is demonstrated as an effective IT ticket classifier in [6], as our ticket labeling baseline.

NB-TF-IDF: Naive Bayes with TF-IDF features. The Term Frequency-Inverse Document Frequency (TF-IDF) is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. The first metric is word frequency in a document and the second one is the inverse document frequency of the word. In the ticket labeling settings, the document is a ticket description. The TF-IDF features are fed to a Naive Bayes model.

LG-BoW:Logistic Regression with BoW features. Logistic Regression is often used in ticket classification [6, 54, 55] due to its light and explainable model structure. We use BoW features as the model inputs. Please note that we use one-against-the-rest strategy [56] to perform multi-class classification on the logistic regression model.

LG-TF-IDF: Logistic Regression with TF-IDF features. For a fair comparison, we use the same TF-IDF features in NB-TF-IDF and Logistic Regression as another baseline.

LS-Model: Text classifiers on Microsoft Language Studio. To the best of our knowledge, the text classifiers on the Microsoft Language Studio are advanced neural-network

Labels	Explanations
CloudNet FPGA Alerts	The FPGA board on the cloud network triggers an alert
Attestation	The attestation error happened
Capacity Recovery Requests	The capacity recovery incident
SKU Artifacts	The SKU related issues
ECO Request or HW Profile Support	The hardware or ECO related support
Anvil Rate Limit Hit	The limitation of the anvil achieved
GDCOApp Ticket Escalation	The request to escalate ticket about GDCOApp
Customer Facing RCA Requests	The request about RCA instance
Buildout	The request to buildout the system
Firmware Deployment	The firmware related issues

Table 4. The fine-grained ticket labels with explanations.

models, which serves as an competitive baseline. Please note that the reason why we benchmark on LS-Model rather than RNN/LSTM-based models is that (1) the ticket descriptions are often longer than normal sentences (e.g, a ticket description could have more than 50 words), which would make the RNN/LSTM-based models suffering from the issues of gradient vanishing or exploding. (2) LS-Model is an commercialized and state-of-the-art text classification model which serves as a strong baseline.

4.2. Ticket-BERT

We fine-tune BERT-base model for the ticket labeling tasks on the proposed datasets. Instead of training BERT-base from scratch, we fine-tune pretrained BERT-base model which takes much less time to learn and achieve state-of-the-art results with minimal task-specific adjustments. The BERT-base model structure is shown in Figure 1 (f). The inputs are a sequence of ticket descriptions. In order to leverage auxiliary data, such as ticket title and summary, we also develop prompt-prefix strategy to format input text sequences for training baseline and fine-tuning Ticket-BERT models. In particular, we concatenate prompt-prefix (ticket titles/summaries) and ticket descriptions in three templates:

(1) *[CLS] <description>*

(2) *[CLS] <title> [SEP] <description>*

(3) *[CLS] <title> [SEP] <summary> [SEP] <description>*

The formatted texts are inputs for fine-tuning the BERT-base model. These texts are encoded with word-piece tokens, positional embedding, and penitential segment embedding. All the encoded vectors are concatenated to feed the BERT-base model. Afterward, we extract the '[CLS]' token embedding from the last layer in the BERT-base, and feed it to a Multi-layer Perceptron (MLP) to perform multi-class classification. In the fine-tuning stage, we load the initial BERT-base weights and only fine-tune the MLP block.

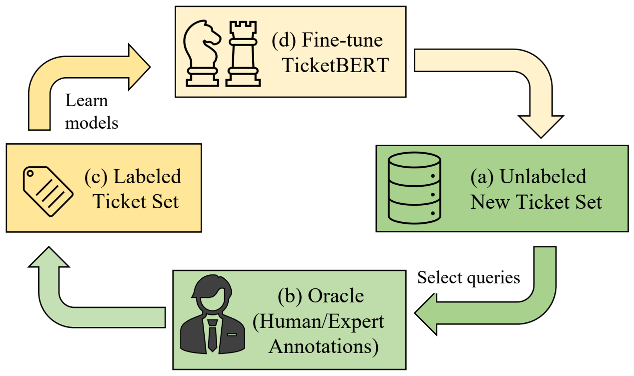


Figure 3. The pool-based active learning cycle. (a) newly unlabeled tickets are collected; (b) a subset of queried unlabeled data requires human/expert annotations; (c) collected/extended the ticket set with annotated labels; (d) fine-tune Ticket-BERT on the annotated data.

4.3. Active Learning

In the real-world ticket labeling task, a large number of unlabeled data can be collected each single day. For example, there is almost 7K new tickets created each day in Microsoft Kusto. These ticket could either cover new incidents that never existed before or address new issues (using new ticket labels) that was not focused on due to the rapid software and hardware improvement. Therefore an active learning [57] framework that could adapt previously learned model to the new data distribution is critically needed. To bridege this gap, we build a pool-based active learning cycle [58], which assumes a small set of data L is annotated but a large number of data U has no labels. With this, we hope to achieve near perfect accuracy as we continue to iterate with our customers.

Suppose we have learned an initial Ticket-BERT $M^{(i)}$ model with initial annotated data $L^{(i)}$ and new data $U^{(i)}$ is collected after fine-tuning $M^{(i)}$, where $i = 0, 1, 2, ..n$, we start a cycle of active learning shown in Figure 3. There are four steps: (1) given new collected unlabeled data $U^{(i)}$, we use uncertainty-sampling strategy [58] to query data from $U^{(i)}$. In particular, we use fine-tuned Ticket-BERT model $M^{(i)}$ to predict the possible labels with probabilities across instances in $U^{(i)}$, and query the instance x^* whose best predicted labels has the least confident

$$x^* = \text{argmin}_x P\left(y^* \mid x; \theta\right)$$

where $y^* = \text{argmax}_y P(y \mid x; \theta)$ is the best predicted label, y are the candidate labels, $x \in U^{(i)}$, θ is model weight. This strategy queries a set of the most uncertain or informative tickets $U_q^{(i)}$ from a large set of newly collected unlabeled data $U^{(i+1)}$. (2) an oracle that involves human annotations on new queried data $L_q^{(i+1)}$. (3) the annotated data augments existing labeled data to

$$L^{(i+1)} = L^{(i)} \cup L_q^{(i+1)}.$$

(4) $L^{(i+1)}$ is then used to further fine-tune Ticket-BERT and obtain $M^{(i+1)}$. Thees tickets would be annotated in the next cycle. Please note that the active learning cycles are launched on the Microsoft IcM and the process is on-going, we will report new performance in future work.

5. Experiments and Analysis

5.1. Implementation Details

In the data preprocessing, we develop a set of rules to clean the text, such as remove URLs, Code, HTML Tags, Tables, Braces and other metadata. All the cleaned text are converted to lower cases and stopwords are removed. We conduct extensive experiments on the created datasets and achieve state-of-the-art performance. We implement the multi-class Naive Bayes (NB) and Logistic Regression (LR) with scikit-learn[3] and Ticket-BERT with Pytorch[4]. We fine-tune the Ticket-BERT models on a 10-class classifier using cross-entropy losses on an Nvidia V100 GPU. The batch size is set to 16, and the learning rate is set to 0.0001, with 0.1 decay every 4 out of 12 epochs. In preprocessing, we trunk the text into 512 dimensions for the Ticket-BERT and set 1024 as the maximum dimensions for BoW and TF-IDF vectors. We report the best performance on macro Precision, Recall, F1, and AUC of ROC scores on the test set.

5.2. Evaluation on the Dataset

Table 5 shows the baselines and Ticket-BERT model performance. For the D-Human dataset, the Native Bayes models generally are worse than Logistic Regression models which suggests that discriminative modes are better than generative models in ticket labeling tasks. Moreover, the Logistic regression with BoW features is better than it with TF-IDF features; however, the Naive Bayes with TF-IDF features is better than with BoW features. This suggests that discriminative models favor features that cover the most of text information, such as BoW counting word occurrences, while generative models work well on more salient words rather than the ones with high occurrences. In particular, the Ticket-BERT models show competitive results on D-Human in terms of precision, recall, and F1 scores, which demonstrates the superiority of pretrained language models on ticket labeling. Regarding D-Machine and D-Mixture datasets, the Ticket-BERT models have relatively lower recalls than the strongest baselines LG-BoW but show better scores in terms of precision and F1 scores. This makes sense because D-Machine and D-Mixture datasets contain machine-generated/updated tickets that have relatively different semantic structures compared to human languages, as shown in Table 1.

5.3. Evaluation on Prompt-prefix

The model performance on D-Machine and D-Mixture is generally higher than that of in D-Human in Table 5. Our reasoning is that the Human entered/updated ticket descriptions have rich variation that may confuse the machine learning models and make the task difficult, compared to machine-generated/updated text. To overcome these issues, we proposed the prompt-prefix that leverages auxiliary data (e.g., ticket title and summary) to enrich the ticket context and make the text more issue-specific. Table 6 shows the model performance after concatenating prompt-prefix. We observe strong boosts for all the models on the D-Human dataset and moderate improvement on the D-Machine and D-Mixture datasets if we take title as auxiliary text. While concatenating both summary and title on the D-Human dataset, Ticket-BERT's scores decrease from the one fine-tuned with title-only auxiliary text. This might be the reason that ticket summaries mostly share similar structures to the descriptions so concatenating summaries might introduce duplicate information or even non-salient information. In addition, ”titles” are generated at the creation of the ticket, so performing this would not delay the timing of when our algorithm can be applied in production. In general, the models fine-tuned with the prompt-prefix show strong performance thus demonstrate its effectiveness. Furthermore, we report the breakdown scores of the best model on D-Human dataset in Table 7, which shows outstanding performance.

5.4. Evaluation on Hard-to-Identify Tickets

The performance on the D-Machine, D-Human, and D-Mixture is excellent, which demonstrates that the models learn well on these datasets. We further evaluate Ticket-BERT using human input on a set of hard-to-identify tickets that are difficult for annotators to identify in a short time. In particular, these tickets do not express specific incident issues thus are not easy to label by humans. We test fine-tuned

[3] https://scikit-learn.org/stable/
[4] https://pytorch.org/

Dataset	Models	Precision	Recall	F1-Score	AUC
D-Human	NB-BoW	71.07	73.74	69.87	0.955
	NB-TF-IDF	78.18	76.67	76.74	0.973
	LS-Model	84.36	84.08	83.96	**
	LR-BoW	85.16	84.12	84.58	0.98
	LR-TF-IDF	85.81	82.89	83.98	0.986
	Ticket-BERT	**86.40**	**85.50**	**85.90**	**0.988**
D-Machine	NB-BoW	89.17	91.98	89.71	0.997
	NB-TF-IDF	92.28	93.72	92.69	0.999
	LS-Model	96.01	94.52	95.14	**
	LR-BoW	97.91	**98.07**	97.98	1.000
	LR-TF-IDF	97.15	97.50	97.30	1.000
	Ticket-BERT	**98.42**	97.98	**98.19**	**1.000**
D-Mixture	NB-BoW	88.25	91.48	88.88	0.996
	NB-TF-IDF	92.79	94.85	93.61	0.999
	LS-Model	97.24	96.71	96.91	**
	LR-BoW	98.02	**97.93**	97.97	0.999
	LR-TF-IDF	97.61	97.57	97.58	1.000
	Ticket-BERT	**98.62**	97.91	**98.24**	**1.000**

Table 5. The performance on proposed datasets for baselines and Ticket-BERT models. **Please note that Language Studio does not provide per class probabilities, and as a result, we cannot calculate the AUC for this line.

Ticket-BERT model on dozens of these tickets. Surprisingly, we obtain nearly 90% accuracy, which demonstrate the effectiveness of our proposed models in real industry scenarios. Moreover, we deploy the model on a real incident management system where our models are actively refine-tuned with newly collected data, new developed ticket labels, and learned to deal with a diverse set of incoming tickets.

6. Conclusion and Future Work

Ticket labeling is an essential task that provides an important summary of incident issues in support tickets and is critical to efficient triage to responsible teams for timely resolution. Existing methods are not applicable to this problem as they cannot sufficiently address the unique challenges in modern incident management systems (e.g., tickets are frequently manipulated or updated by machines or engineers who possess intrinsically different semantic structures). To address these shortcomings, we create three ticket datasets (i.e., D-Human, D-Machine, D-Mixture) to train domain-adaptive Ticket-BERT that achieves state-of-the-art predictions on all three. Our validation on a set of hard-to-identify tickets supplementally demonstrates Ticket-BERT's effectiveness. While the model works well for IcM tickets for Azure Hardware, it would not generalize to all tickets within the Microsoft IcM system. However, we would be interested in collaborating and supporting additional efforts around generalizing this method for all IcM tickets. In the future, we will refine Ticket-BERT through an active learning cycle and make it adaptive to newly collected ticket data.

Dataset	Models	Precision	Recall	F1-Score	AUC
D-Human +Title	NB-BoW	91.14	94.13	92.09	0.993
	NB-TF-IDF	93.46	94.23	93.77	0.997
	LS-Model	95.56	97.76	96.49	**
	LR-BoW	97.92	97.90	97.91	0.999
	LR-TF-IDF	97.80	97.15	97.46	0.999
	Ticket-BERT	**98.76**	**99.17**	**98.96**	**1.000**
D-Human +Title +Summary	NB-BoW	92.01	94.69	92.82	0.993
	NB-TF-IDF	94.30	95.03	94.61	0.998
	LS-Model	97.81	97.75	97.77	**
	LR-BoW	98.24	98.15	98.19	0.998
	LR-TF-IDF	97.89	97.33	97.60	0.999
	Ticket-BERT	**98.36**	**98.88**	**98.61**	**1.000**
D-Machine +Title	NB-BoW	91.81	94.04	92.27	0.998
	NB-TF-IDF	94.73	96.10	95.16	0.999
	LS-Model	98.98	99.33	99.13	**
	LR-BoW	98.96	99.23	99.09	1.000
	LR-TF-IDF	98.43	98.87	98.64	1.000
	Ticket-BERT	**99.31**	**99.24**	**99.27**	**1.000**
D-Machine +Title +Summary	NB-BoW	93.16	94.86	93.47	0.999
	NB-TF-IDF	95.08	96.84	95.70	1.000
	LS-Model	98.97	99.29	99.11	**
	LR-BoW	99.20	99.20	99.20	1.000
	LR-TF-IDF	98.92	99.02	98.97	1.000
	Ticket-BERT	**99.34**	**99.35**	**99.35**	**1.000**
D-Mixture +Title	NB-BoW	91.87	94.56	92.60	0.998
	NB-TF-IDF	95.76	97.30	96.43	0.999
	LS-Model	98.07	97.82	97.91	**
	LR-BoW	98.93	98.98	98.95	1.000
	LR-TF-IDF	98.67	98.78	98.72	1.000
	Ticket-BERT	**99.31**	**99.38**	**99.34**	**1.000**
D-Mixture +Title +Summary	NB-BoW	93.29	95.52	93.98	0.998
	NB-TF-IDF	95.51	97.45	96.34	0.999
	LS-Model	98.24	97.03	98.06	**
	LR-BoW	99.09	98.94	99.01	1.000
	LR-TF-IDF	98.75	98.67	98.71	1.000
	Ticket-BERT	**99.37**	**99.34**	**99.36**	**1.000**

Table 6. The performance on proposed datasets for baselines and Ticket-BERT models with auxiliary prompt-prefix.**Please note that Language Studio does not provide per class probabilities, and as a result, we cannot calculate the AUC for this line.

Labels	Precision	Recall	F1 Score	AUC
Buildout	93.18	93.18	93.18	1.000
GDCOApp Ticket Escalation	99.67	99.33	99.50	1.000
SKU Artifacts	99.24	99.43	99.33	1.000
Customer Facing RCA Requests	96.40	95.71	96.06	1.000
CloudNet FPGA Alerts	99.48	**99.93**	99.70	1.000
Capacity Recovery Requests	98.05	96.57	97.30	1.000
ECO Request or HW Profile Support	96.55	99.24	97.88	1.000
Anvil Rate Limit Hit	**100.00**	99.73	99.86	1.000
Firmware Deployment	99.11	98.24	98.67	1.000
Attestation	**100.00**	99.92	**99.96**	1.000

Table 7. The breakdown performance on D-Human dataset for the best TicketBERT model. See Appendix for ROC curve and detailed confusion matrix for the best D-Human, D-Machine, D-Mixture Ticket-BERT models.

References

[1] R. Gupta, K. H. Prasad, and M. Mohania, "Automating itsm incident management process," in *2008 International Conference on Autonomic Computing*, pp. 141–150, IEEE, 2008.

[2] R. Gupta, K. H. Prasad, L. Luan, D. Rosu, and C. Ward, "Multi-dimensional knowledge integration for efficient incident management in a services cloud," in *2009 IEEE International Conference on Services Computing*, pp. 57–64, IEEE, 2009.

[3] M. Mukunthan and S. Selvakumar, "Multilevel petri net-based ticket assignment and it management for improved it organization support," *Concurrency and Computation: Practice and Experience*, vol. 31, no. 14, p. e5297, 2019.

[4] L. Feng, J. Senapati, and B. Liu, "Tadaa: real time ticket assignment deep learning auto advisor for customer support, help desk, and issue ticketing systems," *arXiv preprint arXiv:2207.11187*, 2022.

[5] R. Kallis, A. Di Sorbo, G. Canfora, and S. Panichella, "Ticket tagger: Machine learning driven issue classification," in *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 406–409, IEEE, 2019.

[6] A. Revina, K. Buza, and V. G. Meister, "It ticket classification: the simpler, the better," *IEEE Access*, vol. 8, pp. 193380–193395, 2020.

[7] S. S. Ali Zaidi, M. M. Fraz, M. Shahzad, and S. Khan, "A multiapproach generalized framework for automated solution suggestion of support tickets," *International Journal of Intelligent Systems*, vol. 37, no. 6, pp. 3654–3681, 2022.

[8] H. Gu, L. Zhao, and C. Shu, "Analysis of duplicate issue reports for issue tracking system," in *The 3rd International Conference on Data Mining and Intelligent Information Technology Applications*, pp. 86–91, IEEE, 2011.

[9] A. Maksai, J. Bogojeska, and D. Wiesmann, "Hierarchical incident ticket classification with minimal supervision," in *2014 IEEE International Conference on Data Mining*, pp. 923–928, IEEE, 2014.

[10] G. Son, V. Hazlewood, and G. D. Peterson, "On automating xsede user ticket classification," in *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment*, pp. 1–7, 2014.

[11] J. Han and M. Akbari, "Vertical domain text classification: towards understanding it tickets using deep neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[12] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text classification algorithms: A survey," *Information*, vol. 10, no. 4, p. 150, 2019.

[13] A. I. Kadhim, "Survey on supervised machine learning techniques for automatic text classification," *Artificial Intelligence Review*, vol. 52, no. 1, pp. 273–292, 2019.

[14] B. Agarwal and N. Mittal, "Text classification using machine learning methods-a survey," in *Proceedings of the Second International Conference on Soft Computing for Problem Solving (SocProS 2012), December 28-30, 2012*, pp. 701–709, Springer, 2014.

[15] A. M. P. Braşoveanu and R. Andonie, "Visualizing transformers for nlp: A brief survey," in *2020 24th International Conference Information Visualisation (IV)*, pp. 270–279, 2020.

[16] K. Jing and J. Xu, "A survey on neural network language models," *arXiv preprint arXiv:1906.03591*, 2019.

[17] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in vision: A survey," *ACM Computing Surveys (CSUR)*, 2021.

[18] A. Hotho, A. Nürnberger, and G. Paaß, "A brief survey of text mining.," in *Ldv Forum*, vol. 20, pp. 19–62, Citeseer, 2005.

[19] K. Babić, S. Martinčić-Ipšić, and A. Meštrović, "Survey of neural text representation models," *Information*, vol. 11, no. 11, p. 511, 2020.

[20] H. M. Wallach, "Topic modeling: beyond bag-of-words," in *Proceedings of the 23rd international conference on Machine learning*, pp. 977–984, 2006.

[21] T. Joachims, "A probabilistic analysis of the rocchio algorithm with tfidf for text categorization.," tech. rep., Carnegie-mellon univ pittsburgh pa dept of computer science, 1996.

[22] S. Selva Birunda and R. Kanniga Devi, "A review on word embedding techniques for text classification," *Innovative Data Communication Technologies and Application*, pp. 267–281, 2021.

[23] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[24] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

[25] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the association for computational linguistics*, vol. 5, pp. 135–146, 2017.

[26] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," *Advances in neural information processing systems*, vol. 27, 2014.

[27] J. Sarzynska-Wawer, A. Wawer, A. Pawlak, J. Szymanowska, I. Stefaniak, M. Jarkiewicz, and L. Okruszek, "Detecting formal thought disorder by deep contextualized word representations," *Psychiatry Research*, vol. 304, p. 114135, 2021.

[28] L. Xiang, X. Sun, G. Luo, and B. Xia, "Linguistic steganalysis using the features derived from synonym frequency," *Multimedia tools and applications*, vol. 71, no. 3, pp. 1893–1911, 2014.

[29] A. Revina, K. Buza, and V. G. Meister, "Designing explainable text classification pipelines: insights from it ticket complexity prediction case study," in *Interpretable Artificial Intelligence: A Perspective of Granular Computing*, pp. 293–332, Springer, 2021.

[30] H. Zhuge, "Interactive semantics," *Artificial Intelligence*, vol. 174, no. 2, pp. 190–204, 2010.

[31] J. Y. Kim and J. Shawe-Taylor, "Fast multiple keyword searching," in *Annual Symposium on Combinatorial Pattern Matching*, pp. 41–51, Springer, 1992.

[32] R. Sidhu and V. K. Prasanna, "Fast regular expression matching using fpgas," in *The 9th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'01)*, pp. 227–238, IEEE, 2001.

[33] W. Hadi, Q. A. Al-Radaideh, and S. Alhawari, "Integrating associative rule-based classification with naïve bayes for text classification," *Applied Soft Computing*, vol. 69, pp. 344–356, 2018.

[34] L. Yao, C. Mao, and Y. Luo, "Clinical text classification with rule-based features and knowledge-guided convolutional neural networks," *BMC medical informatics and decision making*, vol. 19, no. 3, pp. 31–39, 2019.

[35] Y.-D. Ye, X.-Y. Lv, G.-Q. Cai, and L.-M. Jia, "Train ticket predictive analysis based on decision tree induction," in *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 03EX693)*, vol. 4, pp. 2409–2414, IEEE, 2003.

[36] K. Phetrungnapha and T. Senivongse, "Classification of mobile application user reviews for generating tickets on issue tracking system," in *2019 12th International Conference on Information & Communication Technology and System (ICTS)*, pp. 229–234, IEEE, 2019.

[37] M. Ikonomakis, S. Kotsiantis, and V. Tampakas, "Text classification using machine learning techniques.," *WSEAS transactions on computers*, vol. 4, no. 8, pp. 966–974, 2005.

[38] G. Antoniol, K. Ayari, M. Di Penta, F. Khomh, and Y.-G. Guéhéneuc, "Is it a bug or an enhancement? a text-based approach to classify change requests," in *Proceedings of the 2008 conference of the center for advanced studies on collaborative research: meeting of minds*, pp. 304–318, 2008.

[39] K. Herzig, S. Just, and A. Zeller, "It's not a bug, it's a feature: how misclassification impacts bug prediction," in *2013 35th international conference on software engineering (ICSE)*, pp. 392–401, IEEE, 2013.

[40] Y. Zhou, Y. Tong, R. Gu, and H. Gall, "Combining text mining and data mining for bug report classification," *Journal of Software: Evolution and Process*, vol. 28, no. 3, pp. 150–176, 2016.

[41] A. Parvat, J. Chavan, S. Kadam, S. Dev, and V. Pathak, "A survey of deep-learning frameworks," in *2017 International Conference on Inventive Systems and Control (ICISC)*, pp. 1–7, IEEE, 2017.

[42] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[43] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015.

[44] C. Xiao, M. Dymetman, and C. Gardent, "Sequence-based structured prediction for semantic parsing," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1341–1350, 2016.

[45] M. Huang, Y. Cao, and C. Dong, "Modeling rich contexts for sentiment classification with lstm," *arXiv preprint arXiv:1605.01478*, 2016.

[46] S. Santhanam, "Context based text-generation using lstm networks," *arXiv preprint arXiv:2005.00048*, 2020.

[47] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[48] C. Sun, X. Qiu, Y. Xu, and X. Huang, "How to fine-tune bert for text classification?," in *China national conference on Chinese computational linguistics*, pp. 194–206, Springer, 2019.

[49] A. Adhikari, A. Ram, R. Tang, and J. Lin, "Docbert: Bert for document classification," *arXiv preprint arXiv:1904.08398*, 2019.

[50] L. H. Li, M. Yatskar, D. Yin, C.-J. Hsieh, and K.-W. Chang, "What does bert with vision look at?," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 5265–5275, 2020.

[51] H. Bao, L. Dong, and F. Wei, "Beit: Bert pre-training of image transformers," *arXiv preprint arXiv:2106.08254*, 2021.

[52] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *arXiv preprint arXiv:2107.13586*, 2021.

[53] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel, "Language models as knowledge bases?," *arXiv preprint arXiv:1909.01066*, 2019.

[54] K. Khowongprasoed and T. Titijaroonroj, "Automatic thai ticket classification by using machine learning for it infrastructure company," in *2022 19th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pp. 1–6, 2022.

[55] S. Paramesh, C. Ramya, and K. Shreedhara, "Classifying the unstructured it service desk tickets using ensemble of classifiers," in *2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS)*, pp. 221–227, 2018.

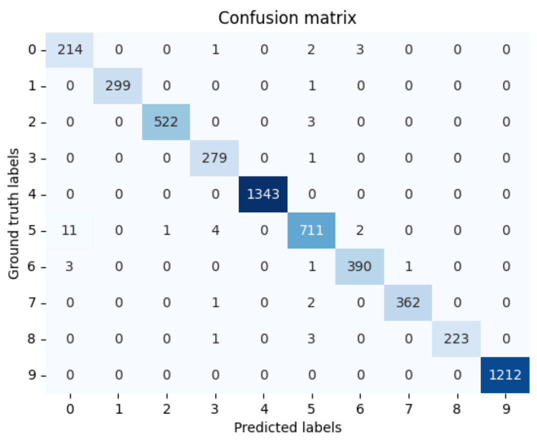
[56] J. Li, Y. Liu, R. Yin, H. Zhang, L. Ding, and W. Wang, "Multi-class learning: From theory to algorithm," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[57] D. D. Lewis, "A sequential algorithm for training text classifiers: Corrigendum and additional data," in *Acm Sigir Forum*, vol. 29, pp. 13–19, ACM New York, NY, USA, 1995.

[58] B. Settles, "Active learning literature survey," 2009.

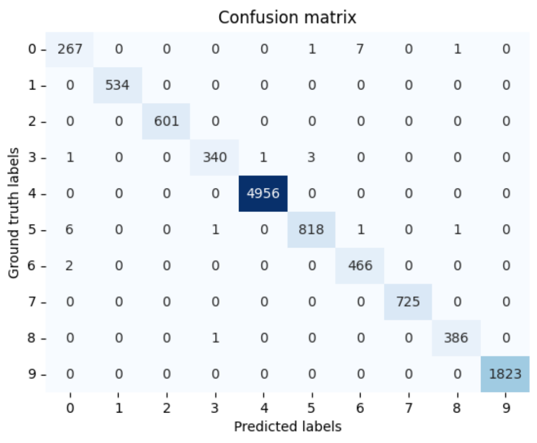
A. Appendices

A.1. Per-class performance for Ticket-BERT


Confusion matrix
Ground truth labels
Predicted labels

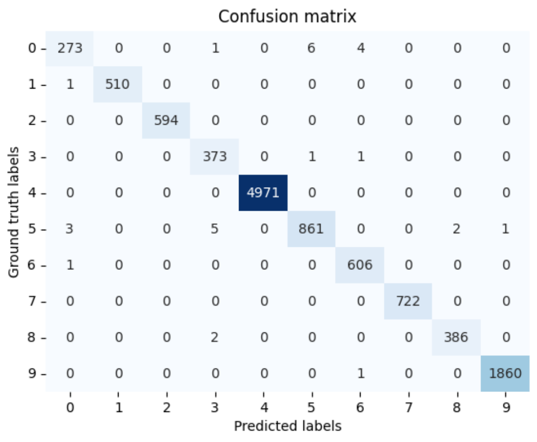
	0	1	2	3	4	5	6	7	8	9
0	214	0	0	1	0	2	3	0	0	0
1	0	299	0	0	0	1	0	0	0	0
2	0	0	522	0	0	3	0	0	0	0
3	0	0	0	279	0	1	0	0	0	0
4	0	0	0	0	1343	0	0	0	0	0
5	11	0	1	4	0	711	2	0	0	0
6	3	0	0	0	0	1	390	1	0	0
7	0	0	0	1	0	2	0	362	0	0
8	0	0	0	1	0	3	0	0	223	0
9	0	0	0	0	0	0	0	0	0	1212

Figure 4. Ticket-BERT with D-human


Confusion matrix
Ground truth labels
Predicted labels

	0	1	2	3	4	5	6	7	8	9
0	267	0	0	0	0	1	7	0	1	0
1	0	534	0	0	0	0	0	0	0	0
2	0	0	601	0	0	0	0	0	0	0
3	1	0	0	340	1	3	0	0	0	0
4	0	0	0	0	4956	0	0	0	0	0
5	6	0	0	1	0	818	1	0	1	0
6	2	0	0	0	0	0	466	0	0	0
7	0	0	0	0	0	0	0	725	0	0
8	0	0	0	1	0	0	0	0	386	0
9	0	0	0	0	0	0	0	0	0	1823

Figure 5. Ticket-BERT with D-machine


Confusion matrix
Ground truth labels
Predicted labels

	0	1	2	3	4	5	6	7	8	9
0	273	0	0	1	0	6	4	0	0	0
1	1	510	0	0	0	0	0	0	0	0
2	0	0	594	0	0	0	0	0	0	0
3	0	0	0	373	0	1	1	0	0	0
4	0	0	0	0	4971	0	0	0	0	0
5	3	0	0	5	0	861	0	0	2	1
6	1	0	0	0	0	0	606	0	0	0
7	0	0	0	0	0	0	0	722	0	0
8	0	0	0	2	0	0	0	0	386	0
9	0	0	0	0	0	0	1	0	0	1860

Figure 6. Ticket-BERT with D-mixture

A.2. ROC-AUC and auPRC curves for Ticket-BERT

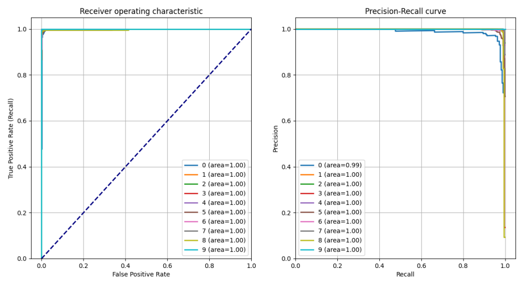


Figure 7. Ticket-BERT with D-human

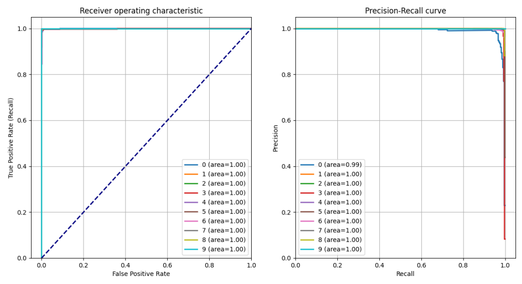


Figure 8. Ticket-BERT with D-machine

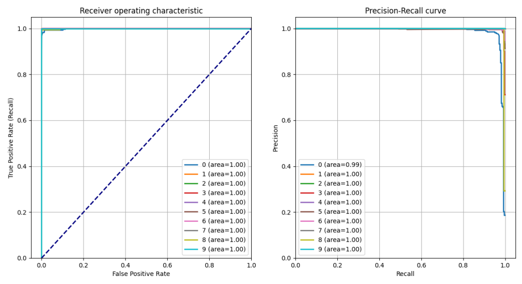


Figure 9. Ticket-BERT with D-mixture

Indicator	**Label**
0	Buildout
1	GDCO Escalation
2	SKU Artifacts
3	Customer Facing RCA
4	CloudNet FPGA Alerts
5	Capacity Recovery Requests
6	ECO Request
7	Anvil Rate
8	Firmware Deployment
9	Attestation