Picking Pearl From Seabed: Extracting Artefacts from Noisy Issue Triaging Collaborative Conversations for Hybrid Cloud Services

Amar Prakash Azad,[1] **Supriyo Ghosh,** [1] **Ajay Gupta,** [1] **Harshit Kumar,**[1] **Prateeti Mohapatra,** [1]
Lena Eckstein, [2] **Leonard Posner,**[2] **Robert Kern** [2]

[1] IBM Research Lab
[2] IBM Data and AI

amarazad@in.ibm.com, supriyog@ibm.com, ajaygupta@in.ibm.com, harshitk@in.ibm.com, pramoh01@in.ibm.com,
lena.eckstein@de.ibm.com, leonard.timo.posner@ibm.com, robkern@de.ibm.com

Abstract

Site Reliability Engineers (SREs) play a key role in identifying the cause of an issue and performing remediation steps to resolve it. After an issue is reported, SREs come together in a virtual room (collaboration platform) to triage the issue. While doing so, they leave behind a wealth of information, in the form of conversations, which can be used later for triaging similar issues. However, usability of these conversations offer challenges due to them being noisy and scarcity of conversation utterance label. This paper presents a novel approach for issue artefact extraction from noisy conversations with minimal labelled data. We propose a combination of unsupervised and supervised models with minimal human intervention that leverages domain knowledge to predict artefacts for a small amount of conversation data and use that for fine-tuning an already pre-trained language model for artefact prediction on a large amount of conversation data. Experimental results on our dataset show that the proposed ensemble of the unsupervised and supervised models is better than using either one of them individually. We also present a deployment case study of the proposed artefact prediction.

Introduction

The deployment of applications using the micro-services architecture has simplified the scope of developers to put the system in production, however, the role of Site Reliability Engineers (SREs) has become more complex. Most times, when services fail, leading to alerts and anomalies, a Site Reliability Engineer comes into play, whose role is to ensure that the services run uninterrupted, i.e. if they fail, they return to normal execution as quickly as possible without impacting any client business. SREs use collaboration platforms such as Slack or Microsoft Teams to communicate with each other while triaging an issue, with a common aim to identify the problem, symptoms, diagnosis and action steps to resolve the issue. Such conversations contain useful artefacts including symptoms, diagnosis and action (key information of SRE's interest) for the issue. These artefacts can be used to find similar conversations from the historical conversations database - this requires grounding historical conversations and the current conversation to a common skeleton structure consisting of the aforementioned artefacts.

Figure 1 depicts an example where key artefacts from noisy SREs' *Conversation* (with chit-chat), referred to as *issue conversation*, are extracted and visualized as a triaging tree which is a linear temporal representation of extracted artefacts.

Identifying key artefacts in an issue conversation is a challenging problem because of the unavailability of ground truth data. Labelled data scarcity is one of the major bottleneck to build an AI system, same is the case here; we do have a lot of conversation data, however, artefacts annotations are not available for the utterances in these conversations. To overcome this problem, we annotated a small conversation dataset using manually defined dictionaries and rules. Further, we get this small labelled data corrected by SREs such that we have a clean labelled data for the next step which involves fine-tuning a pre-trained language model such as BERT for the artefact labelling task. The motivation behind manual correction step is to ensure that utterance labelled with artefacts, albeit small, should be accurate to be used as ground truth data, such that the underlying language model that would be fine-tuned on this data for artefact prediction performs well.

In this paper, we propose a novel approach for artefact prediction from conversations through an ensemble of i) Domain Knowledge Guided unsupervised artefact prediction model ii) A supervised artefact detection approach based on BERT model fused with FastText embedding. We observed that the proposed approach outperforms the artefact prediction in absence of label data. Also, with minimal label verification effort, it outperforms existing supervised models. In addition, we describe a deployment case study of Artefact prediction model.

Proposed Method

The proposed framework, depicted in Figure 2, consists of the following modules. The *Conversation Disentanglement* module separates intermingled multi-interlocutor utterances into coherent conversations with clear start and stop boundaries. The *Artefact Prediction* module labels each utterance in an issue conversation with one of the three artefacts (Symptom, Action, and Diagnostic) and chit-chat (CC) using a combination of unsupervised and supervised learning approaches. Finally, the extracted artefacts are stored in a database for later consumption to find similar issue conversations at run-

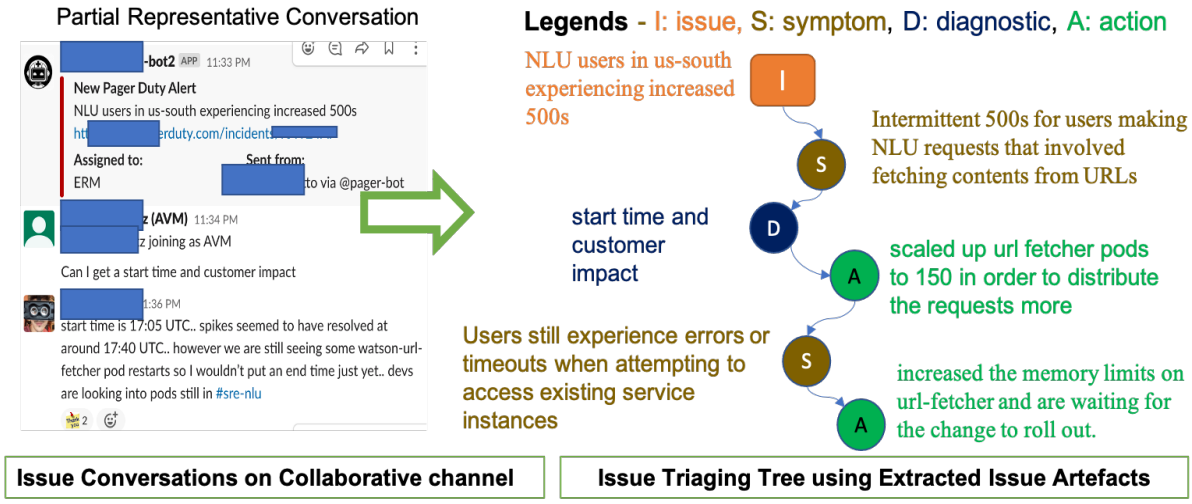

Figure 1: An Issue Triaging Tree example from Collaborative Channel Issue conversations (sensitive information blurred).

time for issue triaging.

Conversation Disentanglement

Conversation data from collaboration channels consists of message exchanges among SREs. These messages may consist of several different conversation threads. The conversation disentanglement module extracts messages which are part of the same thread for further analysis. Lowe et al. (2017) and Lowe et al. (2015) proposed a heuristic approach based on time-difference and direct message to extract threads from the Ubuntu corpus. Conversations contained in collaboration platforms, that SREs use, are multi-interlocutor in nature, whereas Ubuntu conversation is a two-way (or dyadic) conversation. SRE's collaboration platforms also have a feature that allows participants to discuss in threaded structure and these native threads can be extracted using the channel's meta-features. But since several participants may not use this feature all the time, messages may also be written outside these threads as well. These messages can be called *contextual messages*. This module identifies all contextual messages and merges/links them with relevant threads. Our approach extracts all native threads and extracts potential contextual messages before and after the thread. It consists of following rules to identify contextual messages: *Temporal window:* extract a set of messages within a certain temporal window as potential contextual messages, M_{cm}; *User overlap:* extract a set of participant users U_t from the thread and the set of all users U_c from potential contextual messages. All messages from the set M_{cm} written by $U_t \cap U_c$ are considered part of the thread, and are merged together to form one conversation.

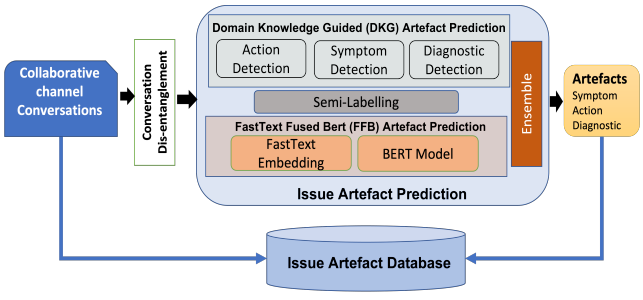

Figure 2: Proposed framework for Artefacts Prediction from Issue Conversations

Artefact Prediction

After obtaining disentangled conversations, the next step is to automatically label utterances in a conversation with artefacts. Each conversation contains a large number of chit-chat utterances besides the utterances that are helpful for issue resolution. The *Artefact Prediction* module assigns relevant artefacts to utterances, in the process, labelling all irrelevant utterances with the artefact chit-chat. We formulate the problem as a multi-class classification problem which includes the artefact label set as Symptom, Action, Diagnostic and chit-chat. The first step is an unsupervised approach, a *Domain Knowledge Guided* (DKG) artefact prediction that uses a set of predefined rules to predict artefact for each utterance in a conversation. In the second step, the labelled utterances are validated by a domain expert. Finally, this set of labelled conversation utterances is used to train the proposed supervised model called as *FastText Fused BERT* (FFB). We further improve the artefact prediction model by taking an ensemble of DKG and FFB, thereby getting the best of both worlds.

Domain Knowledge Guided (DKG) Artefact Prediction

The Domain Knowledge Guided (DKG) artefact prediction module consists of three sub-modules, which predicts utterances of an artefact type, namely, Symptom, Action, Diagnostic in an unsupervised manner. Remaining utterances are labelled as chit-chat.

Action Artefact Detection Action detection is based on the observation that verb-noun pair usually provides natural

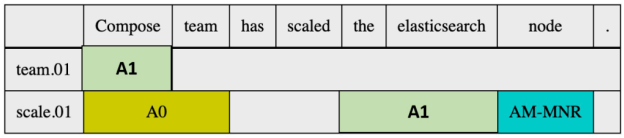

Figure 3: Example of Semantic Role Labelling

and meaningful clusters of IT tickets (Ayachitula and Rohit Khandekar 2020). For a given utterance, key entities are extracted and linked to appropriate actions. The approach for action utterance detection consists of three steps: (i) candidate action utterance selection using action verbs from the domain-specific dictionary, ii) extraction of key entities in these utterances, and (iii) link key entities and action verb using Semantic Role Labelling (Roth and Woodsend 2014) (shallow semantic parser) to filter valid action utterances.

In IT domain, an *action* is defined as a process of performing a change operation by engineers to fix an issue. In particular, action words are those verbs which results in state change of an entity, e.g. *increase*, *reboot*. We curated *action word* dictionary using existing Technical Support and Operations corpus which consists of changes and service request documents. We perform candidate utterance selection in presence of an *action word*.

Extraction of key entities from utterances is inspired by the approach in Mohapatra et al. (2018). The approach uses both linguistic and non-linguistic features to determine an entity. In step (iii), we use Semantic Role Labelling (SRL) (Pradhan et al. 2004; Gildea and Jurafsky 2002) to extract action-entity links from utterances. SRL is a shallow semantic parsing task which provides answers of who did what to whom, when, etc.

We show in Figure 3 the semantic roles with numbered arguments and adjuncts for a sample utterance. Each row in the figure depicts the label of an argument with respect to a particular predicate. In the example sentence, *"team", "scale"* are the two predicates. We use a predicate in a sentence, if it's part-of-speech is Verb and if it is presented in the action dictionary. We also explored the relation between semantic role types and ground truth key phrases annotated for each technical document in (Mohapatra et al. 2018), by deriving the distribution for each role type of a key phrase. We obtained the semantic role of each word in the ground truth key phrase from the corresponding sentence to identify the overall frequency distribution of each role in the corresponding dataset. From this experiment, we found that the semantic role *A1* was the most dominating role. Hence, for each predicate in a sentence, we used only the corresponding text that had *A1* as its semantic role. We extract action utterance if positive action-entity links are present in the utterance.

Symptom Artefact Detection In the support domain, *symptom* artefacts are typically governed by the symptom key-terms, e.g. 500x errors, failure, etc. We curated a *symptom word* dictionary using existing Technical Support and Operations corpus which consists of changes and service request documents. Guided by the presence of dictionary terms in an utterance helps in identification of utterances of artefact type Symptom, and the phrase surrounding the matching term is the actual artefact for the symptom.

Diagnostic Artefact Detection Diagnostic artefact detection is an important component of information mining for automated incident remediation. Knowing the diagnosis-related utterances can help to understand what investigations were carried out to resolve an issue. In the conversations, we noted that most of the diagnostic utterances are questions or query type statements and they are either action type or symptom type utterances. For diagnostic artefact extraction, we identify query or question utterance from the action and symptom utterances. Query utterances, can take both explicit and implicit lexical forms (Shrestha and McKeown 2004; Forsythand and Martell 2007), e.g., explicit: *"Which services are affected ?"*, implicit: *"I was wondering what is the latest impact."*. Furthermore, the queries may also contain informal utterance construct as the conversations are informal in nature. To identify the queries, we adopted a semi-supervised approach augmented with lexical rules along with a simple and effective Naive Bayes classifier. The lexical rules are apt to capture queries containing question words (mainly constituting 5W1H question words, for e.g. who, when, where, what, why, how and presence of '?'), along with a set of other curated verb and adverb based question words (e.g., could, kindly, please). To detect implicit queries which capture the informal query utterances, we trained a Naive Bayes model on NPS Chat dataset[1]. When both lexical and Naive Bayes model yield a negative label for an utterance, then it is labelled as a negative query. Since the artefact detection modules are independent of each other, an utterance can be tagged to multiple artefact types. The Diagnostic artefact has the highest priority followed by Action and Symptom artefact classes.

Semi-Labelling

The unsupervised artefact detection from issue conversations described above doesn't require any label data, thus addressing the problem of cold start for artefact detection. We use the module to obtain conversation utterance pre-label. We ask SREs to verify and correct the pre-labels which can be used for the following supervised artefact detection model. One of the reasons to provide these pre-label is to minimize the human cognition effort and time as SREs are the domain expert but have serious time scarcity for complete label annotation.

For the label correction, about 8,400 utterances from 287 issue conversations were distributed among SREs. SREs being user of the system, we assumed SREs provide us gold labels. We further validated the labels of test data (about 400 utterances) for 7 issue conversations from SREs with domain expertise. Note, due to time scarcity of expert SREs, the number of annotations are not in abundance as in typical supervised classification approaches.

[1] http://faculty.nps.edu/cmartell/NPSChat.htm

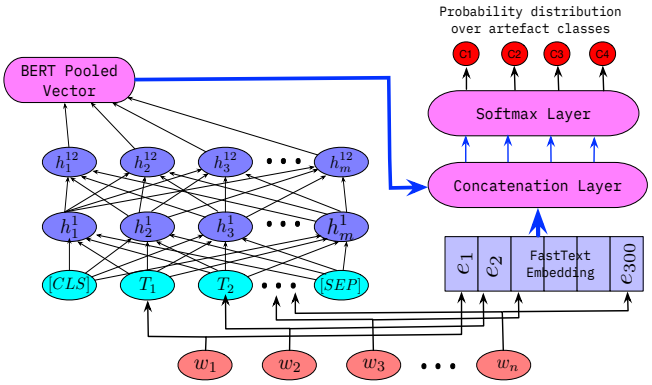

Figure 4: Architecture of FastText Fused BERT model.

FastText Fused BERT (FFB) based Artefact Prediction

This section explains the supervised approach for artefact prediction using the labelled data obtained from the previous step, i.e. semi-labelling method. Despite the fact that fine-tuning a pre-trained BERT model (Devlin et al. 2018) can produce attractive performance for classification task on natural language understanding tasks (Chen, Zhuo, and Wang 2019), such pre-trained BERT model fails to reproduce the performance on our artefact detection task due to limited training data and complexities associated with identifying the right representation of technical terms that are unique to our problem domain. Unfortunately, training a domain-dependent BERT model from scratch is infeasible for us as it is computationally expensive and requires a huge amount of relevant training data. Therefore, we employ a pre-trained BERT model to capture the understanding of natural language conversation and augment its capability with a light-weighted domain-dependent trained model. To obtain a light-weighted model that can represent the unique technical terms appropriately, we train a FastText model (Bojanowski et al. 2017) using our technical domain dataset. Thus, our proposed FastText fused BERT (FFB) model exploits the power of both BERT and FastText to improve the accuracy of artefact prediction – BERT excels in understanding the natural language conversations (e.g., chit-chats), and FastText encapsulates understanding of technical domain dependent terms.

Figure 4 delineates the architecture of our proposed FFB based artefact prediction method. We employ a 12-layer pre-trained BERT model[2], which is a multi-layer bidirectional transformer encoder built upon the transformer model proposed by (Vaswani et al. 2017). Given an input utterance, we begin by tokenizing the utterance and insert a special [CLS] token at the beginning and a special [SEP] token as the final token. Then, a concatenation of *WordPiece* embeddings, positional embeddings, and the segment embedding of the utterance tokens are fed to the BERT model as an input representation. Let the output of the BERT be $\boldsymbol{H} = (\boldsymbol{h_1}, \boldsymbol{h_2}, ..., \boldsymbol{h_N})$ for a given input $\boldsymbol{x} = (x_1, x_2, ..., x_N)$. The hidden representation of the [CLS] token, denoted by $\boldsymbol{h_1}$, is used for artefact prediction, as it provides a sentence level representation of the input text. In addition, we pass the input utterance to a domain-specific trained FastText model that produces a 300-dimensional embedding of the utterance, $\boldsymbol{e}$. Finally, we concatenate the [CLS] token output from BERT model and the FastText output embedding vector to obtain the final hidden layer output, $\boldsymbol{f} = (\boldsymbol{h_1} + \boldsymbol{e})$, and pass it through a *softmax* layer to get the probability distribution over the artefact classes which is expressed as $\boldsymbol{y}^i = \text{softmax}(\boldsymbol{W}^i \boldsymbol{f} + \boldsymbol{b}^i)$.

[2]https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/1

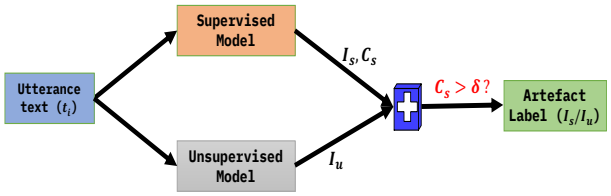

Figure 5: Ensemble method Architecture

Ensemble Method for Artefact Prediction

We experimentally observed that the performance of our supervised approach is not always superior than the unsupervised approach. This could be due to limited amount of labelled data and complexities associated with technical terms, the confidence score of the supervised artefact classification algorithm is relatively low in many cases. Therefore, we propose a simple and effective ensemble method to combine the power of both the supervised and the unsupervised artefact detection approaches. The key idea behind our ensemble approach is that we only rely upon the classification results of the supervised approach if the confidence score is high; otherwise we resort back to the decision of the unsupervised approach. Figure 5 illustrates the architecture of our proposed ensemble method, where the final artefact class label is taken from the decision of the supervised model, only if the confidence score is higher than a threshold parameter δ[3], else the class label from the unsupervised approach is considered as the final label.

Results and Analysis

This section evaluates our method on two dimensions: (i) Efficacy of conversation disentanglement module in terms of identifying segmented conversations; and (ii) Accuracy of the detected and extracted "artefacts" from the segmented conversations.

Conversation Disentanglement

To evaluate the conversation disentanglement module, we extracted threads and contextual messages from conversation data. We randomly selected 189 threads from our dataset which contains total 280 threads.Consecutive threads are separated by at least 24 hours to avoid overlap. For each thread, we extracted up-to 50 contextual messages before the first

[3]We set the value of δ to 0.9, which is obtained using grid search between 0.6 to 1 with 0.05 granularity on validation set.

# Conversations	# Messages	TP	FP	TN	FN
189	508	58	29	409	12

Table 1: Performance analysis of the Conversation Disentanglement method on our dataset.

# Utterances	Sym	Act	Diag	Chit-Chat
7975	18.4%	3.2%	20.1%	58.4%

Table 2: Data distribution over artefact types.

message and after the last message. These contextual messages are extracted within a two hours window. We extracted 508 contextual messages for our experiments. For evaluation, we manually annotated contextual messages to identify if messages are part of the thread or not. The precision and recall of the module are 67% and 83% respectively. Table 1 provides the key performance statistics of our disentanglement module. We observe that most of the false positive cases resulted because users who are participants in thread messages are also a participant of contextual messages. Most of these messages are of type *change request access* (e.g., *@bot Please provide access to* $\langle user \rangle$ *to cluster*).

Artefact Prediction

Baseline methods: We employ two state-of-the-art baseline methods for performance comparison against our proposed artefact detection approaches: (i) K-Means clustering (Popov et al. 2019) – an unsupervised approach in which conversation utterances are clustered over TF-IDF vector space and utterance labels are assigned based on majority voting for each cluster. We use it as a baseline against our unsupervised artefact extraction approach (DKG); and (ii) BiLSTM-CRF model (Kumar et al. 2018) – a supervised approach where utterance representations are learned using the BiLSTM model which are then used for predicting utterance labels modeled as a sequence labelling task. We use it as a baseline against our supervised FFB approach.

Dataset description: For artefact detection training purposes, we randomly obtained about 280 issue conversations that contained 7975 utterances from the collaborative channel involving hybrid cloud related operations and support work. All the supervised models (i.e., BiLSTM-CRF and FFB) are trained using these 7975 utterances from training data. Table 2 shows the artefact distribution statistics of the dataset.

For FFB approach, we used FastText embeddings trained on IT domain support data corpus consisting of documents (e.g., troubleshooting guides, manuals, etc) curated from Red Hat and other cloud services sites. Around 2 million sentences were extracted from these documents. We observe that the FastText model, which was trained from scratch, provides us a better word relations compared to BERT Language models (uncased). We trained FastText for 300 epochs to get a 300-dimensional vector representation for each word and considered sub-words from length 3 to 20. This enabled us to handle multiple word phrases like "Red Hat".

Experimental Evaluation: For evaluation of our proposed artifact detection algorithms, we created a test set consisting of seven issue conversations with about 400 utterances. We validated the annotation of the test set from the SREs with domain expertise. Considering expert SREs' time scarcity for annotation, we choose this relatively small sized test set. We used Precision, Recall and F1 measures to evaluate different artefact prediction methods. We compare the performance of DKG with KMeans, FFB model with BiLSTM-CRF and the Ensemble model against both DKG and FFB. The performance evaluation on our test set is presented in Table 3 and Table 4.

Method	**Average Precision**
K-Means(Popov et al. 2019)	36.7
Bi-LSTM-CRF(Kumar et al. 2018)	51.5
DKG	74.1
FFB	69.2
ENSEMBLE	**77.7**

Table 3: Performances of artefact prediction methods

As shown in Table 3, the unsupervised artefact detection approaches using K-Means clustering and the proposed DKG method have an average precision of 36.7 and 74.1, respectively. This extraordinary performance improvement can be attributed to the manually curated action dictionary, symptom dictionary and lexical rules used for action artefact detection, symptom artefact detection and diagnostic artefact detection, respectively. The supervised artefact detection methods using our FFB model improves the average precision from 51.5 to 69.2, an improvement of 25.5%, over the BiLSTM-CRF model. The proposed ensemble method, that leverages both unsupervised and supervised approaches, outperforms all the approaches by further improving the average precision to 77.7.

As shown in Table 4, with the availability of labelled data, the proposed FFB approach provides better recall as well as precision in all cases. The is because the FFB approach captures both the sequential structure of sentences through BERT and the domain-specific semantic through FastText embedding trained on support documents. However, it is interesting to note that the DKG performs better than FFB approach in some cases (e.g., P@CC and Re@Sym). This can be attributed to the lack of sufficiently labelled data for FFB, which is often a major challenge in industrial setting. Finally, the Ensemble approach leverages the power of both unsupervised and supervised approaches by combining DKG and FFB, yielding the best performance for all the artefact type detection.

We observed that existing approaches fails to perform well due to noisy nature of semi-formal conversation. For example, the utterance "That you are updating the case?" is labelled *diagnostic* where as it is a *Chit-Chat*. Our proposed approaches DKG and FFB can handle such noisy utterances, as DKG uses curated dictionary and FFB uses embeddings trained on IT domain dataset. However,in some cases our model also faces challenges, e.g. compound utterances when multiple artefact types are present. For example, "We are seeing significant improvement in service since 11:30 UTC and continue to work on restoring all operations" is labelled

Algo	DKG	Bi-LSTM CRF	FFB	Ensemble
P@Sym	0.77	0.48	0.85	**0.81**
P@Act	0.60	0.2	0.48	**0.51**
P@Diag	0.48	0.64	0.56	**0.48**
P@CC	0.78	0.55	0.69	**0.84**
Re@Sym	0.72	0.48	0.53	**0.75**
Re@Act	0.52	0.58	0.68	**0.64**
Re@Diag	0.28	0.12	0.32	**0.38**
Re@CC	0.88	0.85	0.86	**0.87**
F1@Sym	0.75	0.48	0.65	**0.78**
F1@Act	0.56	0.29	0.57	**0.57**
F1@Diag	0.35	0.20	0.40	**0.38**
F1@CC	0.82	0.66	0.77	**0.85**

Table 4: Performance analysis for Artefact detection methods with Precision (P), Recall (Re) and F1 score (F1) for each artefact .

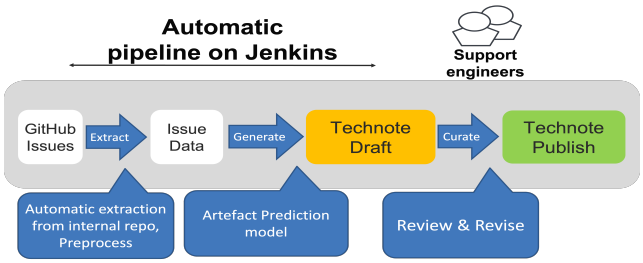

Figure 6: Technotes Publish Pipeline: Automated pipeline on Jenkins for Technotes drafts generation and publishing.

as *action* while the ground truth is *symptom*.

Deployment: Technotes Draft Publishing

In this section, we describe the deployment case study of Technotes creation from GitHub Issue conversations using artefact detection approach.

Due to the current shift from traditional software products to cloud based software, the support process has changed drastically and information from bug reports are overwhelming. Many a time similar issues or bugs keep coming and SREs resolve them afresh without utilizing historical bug reports. There is need of knowledge base with troubleshooting articles summarizing one or multiple bug reports. This knowledge base can be used by support team to resolve incoming issues more efficiently, thus, reducing overall manual effort and cost. Depending on cloud service complexity and usage there could be small to large number of issues(tickets) per day. If for each issue one creates customized troubleshooting document, it will require untenable manual effort by support engineers to maintain knowledge base. To overcome this, we create troubleshooting article drafts, called Technotes, automatically and proactively using our framework. These drafts are curated and published by support engineers and developers.

Pipeline and Implementation Detail

Figure 6 depicts the key components of the Technotes publishing pipeline. Github issues consist of bug report descriptions and comments. These issues are similar to SRE conversations in structure. New Github issues are retrieved periodically using API calls. These issues are processed by a data processing pipeline which is deployed on Jenkin automation server. The pipeline consists of three steps: 1) Pre-processing of raw data obtained from Gihub 2) Creating technote drafts 3) Curation and publishing of technote drafts.

The pre-processing involves stop word removal, stemming and removing unnecessary text such as code block, image tags, URLs and special characters. The pre-processed data is, then, sent to the Artefact prediction model for filtering out *not useful* comments by classifying each comment into the following classes: “investigation”, “symptom”, “action” - which constitute *useful* comments - or “chit-chat” (*not useful* comments). The symptom and action dictionary of Artefact prediction module also includes Github specific additional keyterms such as “fix”, “patch”, “solution”, “workaround”. The set of useful comments, so obtained, constitutes a Technotes draft. The Technotes draft also contains the issue description along with formatted GitHub metadata. Finally, each Technotes draft is reviewed by a SRE for its relevancy and correctness. SRE, based on their domain knowledge, approves the Technotes draft which is then published and indexed in ElasticSearch.

Evaluation

In Figure 7, we show an example Technotes draft and its corresponding publish excerpt from our Technotes publishing pipeline. For the evaluation, we selected a sample of 409 comments from GitHub issues. These comments were annotated by domain experts, and consist of 188 *useful* and 221 *not useful* comments. These annotations form the ground truth for our evaluation. The automated Technotes publishing pipeline identified 164, 80, 140, 24 as True Positives, False Positives, True Negatives and False Negatives respectively. That is, 164 comments out of 188 were correctly identified as useful, and 140 comments out of 221 were correctly identified as not useful by the pipeline. This resulted in an average precision of 66%. Note that, the evaluation results presented in section 3.2 showed the average precision of 77%. There is a 10% drop in average precision which is due to the domain shift; the earlier evaluation was done on the conversation dataset curated from collaborative channel involving hybrid cloud related operations and support work, whereas this experiment is performed on the Github issue conversations for a middleware application server deployment.

The automated Technotes draft creation can help in minimizing bias that can be induced by human annotation. There are some obvious comments which are *not useful*, e.g., “FYI” and notifying another SRE by name. Indisputably *useful* comments are, e.g., step by step instructions to resolve the issue. For other comments, though, different SREs might disagree about the usefulness of the content. In the future, more experts should be involved into annotation. This would enable majority voting on usefulness and also help to improve the ground truth for the artefact detection module. The described proceeding has now been used since two months. Feedback of SREs so far is positive.

Figure 7: Example Technote draft: The upper part depicts useful vs. not useful comments extracted from a GitHub issue and lower part is an excerpt from Technote authoring tool and published document. Sensitive information are blurred.

Concluding Remarks

This paper proposes a novel approach for artefact prediction in IT troubleshooting conversations that uses an ensemble of unsupervised and supervised model. The unsupervised model leverages domain knowledge for artefact extraction without using any label data. The supervised model leverages FastText model trained on domain specific data which is fused with BERT model for improved performance with minimal label data. Through experimental study, we show that both the unsupervised and supervised model outperform existing state-of-the-art models on our issue conversation dataset. Further, the proposed ensemble of unsupervised and supervised model achieves superior performance than using either one of them individually. We also presented a deployment case study to show the value of Artefact prediction.

References

Ayachitula, A.; and Rohit Khandekar, R. 2020. AI for IT: Topic Clustering of Unstructured Texts in IT Services. https://www.linkedin.com/pulse/ai-topic-clustering-unstructured-texts-services-ayachitula/. Accessed: 2021-09-06.

Bojanowski, P.; Grave, E.; Joulin, A.; and Mikolov, T. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5: 135–146.

Chen, Q.; Zhuo, Z.; and Wang, W. 2019. BERT for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Forsythand, E. N.; and Martell, C. H. 2007. Lexical and Discourse Analysis of Online Chat Dialog. In *International Conference on Semantic Computing (ICSC 2007)*, 19–26.

Gildea, D.; and Jurafsky, D. 2002. Automatic Labeling of Semantic Roles. *Comput. Linguist.*, 28(3): 245–288.

Kumar, H.; Agarwal, A.; Dasgupta, R.; and Joshi, S. 2018. Dialogue act sequence labeling using hierarchical encoder with CRF. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Lowe, R.; Pow, N.; Serban, I.; and Pineau, J. 2015. The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems. In *Proceedings of the SIGDIAL 2015 Conference, The 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 2-4 September 2015, Prague, Czech Republic*, 285–294. The Association for Computer Linguistics.

Lowe, R. T.; Pow, N.; Serban, I. V.; Charlin, L.; Liu, C.; and Pineau, J. 2017. Training End-to-End Dialogue Systems with the Ubuntu Dialogue Corpus. *Dialogue Discourse*, 8(1): 31–65.

Mohapatra, P.; Deng, Y.; Gupta, A.; Dasgupta, G.; Paradkar, A.; Mahindru, R.; Rosu, D.; Tao, S.; and Aggarwal, P. 2018. Domain Knowledge Driven Key Term Extraction for IT Services. In *International Conference on Service-Oriented Computing*, 489–504. Springer.

Popov, A.; Bulatov, V.; Polyudova, D.; and Veselova, E. 2019. Unsupervised dialogue intent detection via hierarchical topic model. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, 932–938.

Pradhan, S. S.; Ward, W. H.; Hacioglu, K.; Martin, J. H.; and Jurafsky, D. 2004. Shallow Semantic Parsing using Support Vector Machines. In *HLT-NAACL*, 233–240.

Roth, M.; and Woodsend, K. 2014. Composition of word representations improves semantic role labelling. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 407–413.

Shrestha, L.; and McKeown, K. 2004. Detection of Question-Answer Pairs in Email Conversations. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, 889. USA: Association for Computational Linguistics.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.