Empirical Software Engineering (2019) 24:3755–3789
https://doi.org/10.1007/s10664-019-09721-w

ESSMArT way to manage customer requests

Maleknaz Nayebi[1] · Liam Dicke[2] · Ron Ittyipe[3] · Chris Carlson[4] · Guenther Ruhe[3]

Published online: 21 May 2019

Abstract
Quality and market acceptance of software products is strongly influenced by responsiveness to customer requests. Once a customer request is received, a decision must be made whether to escalate it to the development team. Once escalated, the ticket must be formulated as a development task and assigned to a developer. To make the process more efficient and reduce the time between receiving and escalating the customer request, we aim to automate the complete customer request management process. We propose a holistic method called ESSMArT. The method performs text summarization, predicts ticket escalation, creates the ticket's title and content, and ultimately assigns the ticket to an available developer. We began evaluating the method through an internal assessment of 4114 customer tickets from Brightsquid's secure health care communication platform - Secure-Mail. Next, we conducted an external evaluation of the usefulness of the approach and concluded that: i) supervised learning based on context specific data performs best for extractive summarization; ii) Random Forest trained on a combination of conversation and extractive summarization works best for predicting escalation of tickets, with the highest precision (of 0.9) and recall (of 0.55). Through external evaluation, we furthermore found that ESSMArT provides suggestions that are 71% aligned with human ones. Applying the prototype implementation to 315 customer requests resulted in an average time reduction of 9.2 min per request. ESSMArT helps to make ticket management faster and with reduced effort for human experts. We conclude that ESSMArT not only expedites ticket management, but furthermore reduces human effort. ESSMArT can help Brightsquid to (i) minimize the impact of staff turnover and (ii) shorten the cycle from an issue being reported to a developer being assigned to fix it.

Keywords Automation · customer request management · Text summarization · Ticket escalation · Ticket assignment · Mining software repositories · Case study evaluation

Communicated by: Alexander Serebrenik

✉ Maleknaz Nayebi
mnayebi@polymtl.ca

Extended author information available on the last page of the article

1 Introduction

Software evolution is driven by customers' (also known as the end users) needs, typically in the form of feature or maintenance requests. Management of change requests is time consuming and requires significant training and domain experience. Data-driven automation of this process is proposed to increase responsiveness and improve the quality of this data-driven requirements, and change management process (Maalej et al. 2016).

Automated text summarization is the task of producing a concise and fluent summary while preserving key information, content, and overall meaning (Allahyari et al. 2017). A recent survey on the different concepts and techniques was given by Gambhir and Gupta (2017). While initially developed and used outside software engineering, text summarization becomes critical for handling the textual information that is now widely accessible in software development. Automated summarization of bug reports has been studied e.g. by Rastkar et al. (2014). However, summarization is just the first step in a more comprehensive process of leveraging textual responses for software product improvement. To increase its practical impact, we increase the scope of automation and propose ESSMArT as a method for automating text summarization, escalation of customer requests to the development team, as well as suggesting assignment and priority of the automated ticket report.

In this paper, we target the automated escalation, creation, prioritization, and job assignment of customer requests. To do so, we introduce a method called ESSMArT.[1]

ESSMArT combines summarization with information retrieval techniques for automated generation of escalated tickets based on customer requests. As a case study, we evaluated ESSMArT using data from the development of a Health Communication system offered by a company called Brightsquid. Analyzing customer change requests is an important part of their development process (Nayebi et al. 2018). Through analysis of 4,114 customer requests, we answered the following research questions:

RQ1: Automated Condensing of Customer Requests Among existing state-of- the art techniques for condensing customer requests by extractive summarization, which one works best in terms of F1 accuracy?

Why and How: Typically, once a change request arrives, a Customer Relationship Management (CRM) employee takes over the request and summarizes the request for the customer's confirmation. The summary generated is the base for the escalation decision and possibly for creating a development ticket. Automation of this step is intended to reduce the human workload and increase responsiveness using the state of the art ROUGE metric for evaluation.

RQ2: Predicting Escalation of Customer Requests Comparing three classification algorithms - Naive Bayes, Support Vector Machines, and Random Forest -, which one works best for predicting the escalation of customer requests?

Why and How: Support for Customer Relationship Management (CRM) staff in predicting escalation is expected to help in terms of (i) effort needed and (ii) quality of prediction. We evaluated quality of prediction by comparing algorithm results with those of three machine learners with proven success in similar contexts.

[1] ESSMArT: **ES**calation and **SuM**marization **A**u**T**omation

RQ3: Quality of Automatically Generated Ticket Content How well are the ES- SMArT generated ticket titles and contents aligned with those generated by human experts?

Why and How: Often, only the CRM manager creates and escalates development tickets. These tickets are more general than the summarized conversation, and are represented by a title and body that describes the problem or requested enhancement. ESSMArT uses abstractive summarization to create the ticket title, and a thesaurus to generate development tickets from the summary of tickets studied in **RQ1**.

RQ4: Quality of Operationalization In comparison to the results of a human expert, how correct are (i) the predicted priorities of the tickets and (ii) the assignments of the tickets generated by ESSMArT to developers?

Why and How: The priority of a ticket determines how urgently it should be handled. When correctly prioritized, tickets are handled in the correct order. The proper assignment of the ticket to an available developer familiar with the ticket's domain is critical to implementing the change request. This is currently a manual process constrained by developer expertise. We benchmarked with three states of the art classifiers to automate this process by searching for the analogy of the upcoming ticket with some former tickets.

RQ5: Usefulness of ESSMArT for Experts Utilizing a prototype implementation of ESSMArT, how much are the generated results aligned with the perception of the human experts and how useful are the results?

Why and How: External evaluation looks into the perceived value of ESSMArT from leveraging a prototype implementation. A set of 315 actual customer requests were executed by 21 CRM experts and 33 project managers. We measured execution time and surveyed the perceived usefulness of the tool.

The paper is subdivided into nine sections. In Section 2, we begin by providing more details on the context and motivation for this research. Related work is then analyzed in Section 3. This is followed by an outline of the ESSMArT method in Section 4. Results for different types of (internal) validation of the method are presented in Section 5. Results from external validation follow in Section 6. We provide a discussion on some of the assumptions of the paper and present them in Section 7, followed in Section 8 by a discussion of the limitations and threats to validity of the research. We conclude and give an outlook to future research in Section 9.

2 Context and Motivation: Customer Request Management at Brightsquid

Brightsquid Secure Communication Corp[2] is a global provider of HIPAA-compliant[3] communication solutions - providing compliant messaging and large file transfer for medical and dental professionals since 2009. Secure-Mail is Brightsquid's core communication and collaboration platform offering role-based API access to a catalog of services and automated workflow. It supports aggregating, generating, and sharing protected health

[2] https://Brightsquid.com/
[3] HIPPA: Health Insurance Portability and Accountability Act of 1996

information across communities of health care patients, practitioners, and organizations. Brightsquid has been working on a number of projects in this domain, and this study is focused on analyzing four of these systems. The company is facing the typical problem of software start-ups: the need to quickly enter a competitive market with innovative product ideas while generating short-term revenue by satisfying current customers and their expectations. At the same time, the company is facing the demand of growing their customer base (Kabeer et al. 2017).

The Brightsquid process of managing change requests is shown in Fig. 1. A comparison of the traditional (baseline) process with the one recommended by the proposed ESSMArT method is illustrated in Appendix 1. When a new customer request arrives, a member of CRM staff decides whether the request should be transitioned into a development ticket. If yes, the CRM manager escalates the customer request by summarizing the customer request, translating it into technical language, and opening a new ticket for the software development team. Then, the project manager defines the issue type and adds the ticket to the backlog of customer requests - a set of Jira tickets tagged as "CRM escalated". In each bi-weekly sprint at Brightsquid, the project manager scans through the escalated tickets, discusses the technical aspects of the ticket, and decides whether or not to assign the ticket to a member of the development team. If the ticket is not assigned, the request is maintained in the backlog of tasks to be solved at a later date.

Between *November,* 2014 and *June,* 2017, Brightsquid recorded 4,114 customer requests. 7.8% of these requests were escalated to the development team. These change requests constituted 10.7% of the whole backlog (including 3,026 tickets overall) over these 32 months. After mining the time stamp data of the ticket system, we identified ticket escalation by the CRM manager as the process bottleneck. The duration of time taken by the CRM manager to make an escalation decision (escalation time) is on average 26.6% of the total time from receiving to assigning the ticket. Moreover, the escalation process requires both CRM and project manager involvement (Nayebi et al. 2015).

3 Related Work

Motivated by the problem at Brightsquid, we propose ESSMArT for managing customer requests. The scope of this investigation starts at the time the customer request arrives, escalating the request as well as finally assigning it to a developer. To the best of our knowledge, this is the first study that analyses both the full process, as well as associated data repositories. Former studies have focused exclusively on predicting ticket escalation, or only on summarizing bug reports. Below, we provide an overview of the existing works.

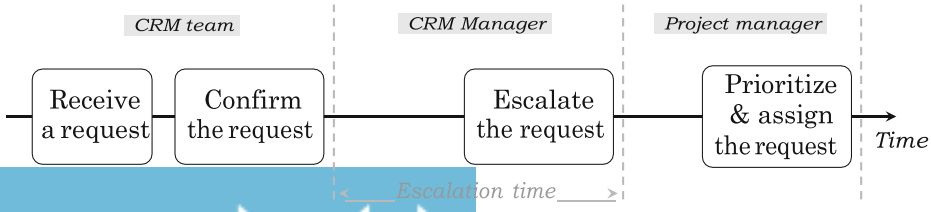

Fig. 1 Change request management process at Brightsquid

3.1 Escalating Customer Requests

Bruckhaus et al. (2004) performed one of the inaugural studies in ticket escalation, providing an early model to predict ticket escalation risk in Sun Microsystems. The study intended to reduce development process risk and cost by excluding previously- reported system defects. As the follow up of this study, the authors took a business-oriented perspective in (Ling et al. 2005) and created a system to predict escalations of known defects in enterprise software to maximize the return on investment (ROI). They limited their study to defect escalation with the intent of preventing risky and costly escalations. The study used a decision tree classifier to find the most cost-effective method, which had a significantly higher ROI than any other method. Also focused on the cost of escalating defects, Sheng et al. (2014) used a cost-sensitive decision tree algorithm to obtain the lowest possible cost in dealing with defects. This paper, similarly to the previous one includes negatives in the cost matrix to account for the benefits of correct classifications.

More recently, Montgomery and Damian (2017) performed a study at IBM Canada to analyze ticket escalation. Montgomery et al. (2017) suggested the tool ECrits to mitigate information overload when making customer request escalation decisions. These two IBM-focused studies define escalation as the process where customers request management escalation of their support ticket, which consequently triggers immediate involvement of a senior software engineer. This differs from our study, where escalation is triggered when CRM experts are unable to directly solve customers' reported problems. Montgomery and Damian (2017) focused on determining attributes that most accurately predict ticket escalation. Their approach included diverse data points, including detailed customer profiles to predict ticket escalation likelihood, as well as customers' response time expectations compared to analysts' average response time. Using a set of 2.5 million support tickets, they were able to achieve a recall of 0.79. In order to make accurate predictions at scale, their model focused on selective rather than comprehensive usage of available data.

Managing customer requests for mobile applications was the content of a survey performed by Martin et al. (2017). However, the notion of escalation was not included as these studies focused instead on analyzing and prioritizing customer requests.

3.2 Text Summarization

Automated text summarization methods are usually discussed under two general categories of *extractive*, and *abstractive* text summarization. For a recent survey of these two categories, see the work of Das and Marins (2007).

3.2.1 Extractive Text Summarization

Extractive text summarization refers to a method of taking a pre-existing document and extracting the sentences that best make up the content of the document. These sentences are taken word for word from the original document. This process is guided by a variety of factors such as the frequency of the words in the sentence, or the similarity of the sentence to the title of the article (Gupta and Lehal 2010).

Extractive summarization has three major steps. First, an intermediate representation of the text is constructed. Second, the sentences are scored based on their calculated importance. In

the third step, the sentence score is used to rank sentences, and those with the highest rank are selected to be part of the extractive summary. Extractive summarization techniques use multiple features and different feature weights for selecting the most representative sentences for the summary. An overview of the extractive summarization process and methods are illustrated in Fig. 2. Extractive summarization methods are different in terms of constructing the intermediate representation. Two major representation techniques exist: *topic representation* and *indicator representation*.

Nazar et al. (2016) provide an overview of the literature of software artifacts. Extractive methods of summarization are most commonly applied in software engineering. Abstractive summarization, on the other hand, is most often applied to large documents, such as news reports (Batista et al. 2015), to facilitate concise reading of the entire document. Researchers have been benchmarking and adopting summarization techniques to improve the accuracy of summarizing software artifacts. In Table 1, we provide an overview of those papers most closely related to ours, including application domains, summarization techniques utilized, and size of datasets. While extractive methods are typically applied in software engineering studies (Table 2), a majority of them are either a subset of the general body of text summarization (e.g., Rastkar et al. (2014)) or are designed independently and for a specific task (e.g., code summarization).

For the design of ESSMArT, we adopted methods from the existing body of knowledge which were classified by systematic literature reviews (Allahyari et al. 2017; Batista et al. 2015) as belonging to one of the established categories of text summarization.

3.2.2 Abstractive Text Summarization

Abstractive text summarization refers to the summarization of text passages and documents, utilizing one of many corpus-backed NLP methods. The ultimate goal of this summarization is to synthesize sentences based on sentence generation, which is done after clustering, importance determination, and other information extraction techniques or ranking methods running on top of an underlying language model. Abstractive NLP summarization techniques most commonly utilize a large corpus and subsequently generated language model, which enables abstractive methods to perform information extraction and ranking (Singhal & Bhattacharya 2019).

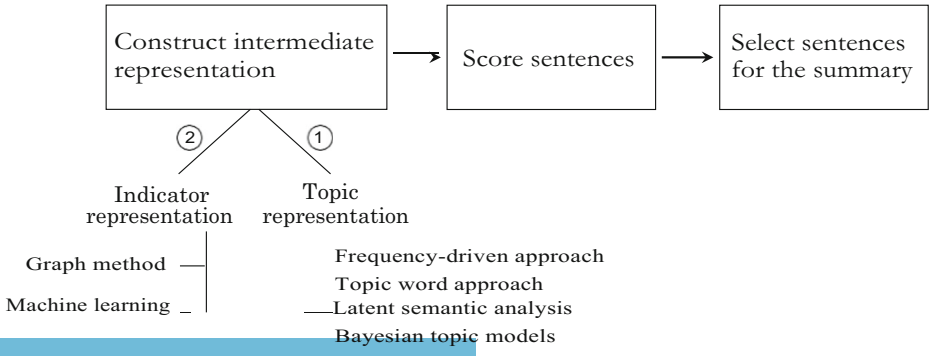

Fig. 2 Overview of extractive summarization methods inferred from (Allahyari et al. 2017). Summarization methods mainly differ in the way they construct the intermediate representation

Table 1 Related software engineering research that used extractive summarization

Application domain	Paper	Summarization technique	Dataset
Bug reports	Rastkar et al. (2010, 2014)	Machine learning	36 Mozilla Bug report
Bug reports	Mani et al. (2012)	Graph method	55 dB2 bug reports
App reviews	Di Sorbo et al. (2016)	Topic classification	17 mobile apps
Release notes	Moreno et al. (2014)	Pre-defined heuristic	1000 release notes

3.3 Ticket Prioritization and Assignment to Developers

Prioritization and assignment of tickets are well established problems in software engineering (Anvik et al. 2006; Kim and Ernst 2007). An assistant for creating bug report assignment recommendations was proposed by Anvik (2016) who demonstrated that sufficiently reliable bug recommendations can be offered even with limited project knowledge. A new framework

Table 2 Extractive summarization techniques analyzed for Step 1 of ESSMArT

ID	Class	Type	Description
SumBasic	Topic representation.	Frequency driven	Proposed by Vanderwende et al. (2007) used in (Jha and Mahmoud 2018; Williams and Mahmoud 2017). The method considers frequency of words in the cluster of input documents as summary sentence selection criteria.
Edmundson	Topic representation	Topic driven	The method uses word phrase frequency, position in a document, and key phrases as the summary sentence selection criteria.
Steinberger	Topic representation	Latent semantic	Uses word co-locations to determine the word's context and similarity of word meanings to cluster sentences and extract the- matic information.
LDA	Topic representation	Bayesian models	Uses measure of divergence between sentences (also known as KL measure) to rank and select sentences.
TextRank	Indicator representation	Graph method	Represent document as a graph where nodes represent sentences and edges demonstrate the degree of similarity between them. Sentences in the center of the graph are included in the summary.
Enron	Indicator representation	Machine learning	Trained a Naive Bayes classifier on the Enron email dataset. Enron contains data from 150 users and a total of about 0.5 MB messages. The data was made public by the Federal Energy Regulatory Commission during its investigation[7].
UDC	Indicator representation	Machine learning	Trained a Naive Bayes classifer on the Ubuntu-related conversations on the Freen- ode IRC network[8]. Internet Relay Chat (IRC) is a form of real-time Internet chat designed for group (many-to-many) communication in discussion fo- rums called channels. We used 5GB of chats be- tween 2004 and 2017.
BSC	Indicator representation	Machine learning	Trained a Naive Bayes classifer on the Brightsquid CRM conversation with customers logged between 2014 and 2017.

for bug triaging was proposed by Xia et al. (2017) using a specialized topic modeling algorithm named multi-feature topic model (MTM) which extends Latent Dirichlet Allocation (LDA) for bug triaging. Another recent method is provided by Bandera et al. (2018). Their patented approach is "based on identifying and scoring quantitative metrics, qualitative indicators, and customer tones contained in the content of respective problem tickets and determining an action step for each respective problem ticket".

Automated assignment of developers to tickets is a stand-alone research topic which has been studied by various authors. More recently, Jonsson et al. (2016) studied the usage of the ensemble-based machine learner called Stacked Generalization (SG) by adopting the methods introduced by Wolpert (1992) which combines several learner types (such as Naive Bayes, Support Vector Machine, KNN). The authors prove SG's applicability in large scale industrial contexts, and moreover, provided a comprehensive analysis of related work in automated bug assignment (Jonsson et al. 2016).

With ESSMArT, our intent was not to perform a comparative analysis to discover the most effective automated bug triaging assignment methods or techniques. Instead, we analyzed a dataset of prioritized and assigned bug tickets against three often-used machine learning techniques known to the decision makers. While investigating learner- impact on the complete triaging process, we demonstrated that applying individual learner components provides "good enough" real-world results.

4 ESSMArT for Automated Customer Request Management

The design of the method was inspired by real-world projects at Brightsquid. ESSMArT consists of five main steps as shown in Fig. 3. While some of these steps (e.g., summarization, escalation) were adapted from existing work, the main value of our method is that it provides a holistic approach covering the complete process starting from the arrival of a change request through to issuing a ticket and assigning it to a developer. As part of the method development, we studied and compared different variants of implementing these steps, described in the following subsections.

4.1 Condense Customer Requests

When a customer request is received, the CRM staff member summarizes the incoming request and then has the customer validate whether the summarization correctly reflects their request. Thus, Step 1 of ESSMArT starts with automated summarization of the customer request.

We used open source python libraries "Summy",[4] "Summa NLP",[5] and "Pyrouge"[6] to implement summarization techniques for ESSMArT.

By analyzing the 4,114 customer requests, we found that they were initially submitted to the CRM and included on average 9.3 (median = 8.8) sentences. Similarly, the summaries created by the CRM staff included on average 4.7 sentences (median = 4.1). Furthermore, we compared the length of tickets generated by eight different CRM staff members and could not find any significant length difference after running an ANOVA test ($p - value = 0.31$). This

[4] https://github.com/miso-belica/sumy
[5] https://github.com/summanlp
[6] https://github.com/summanlp

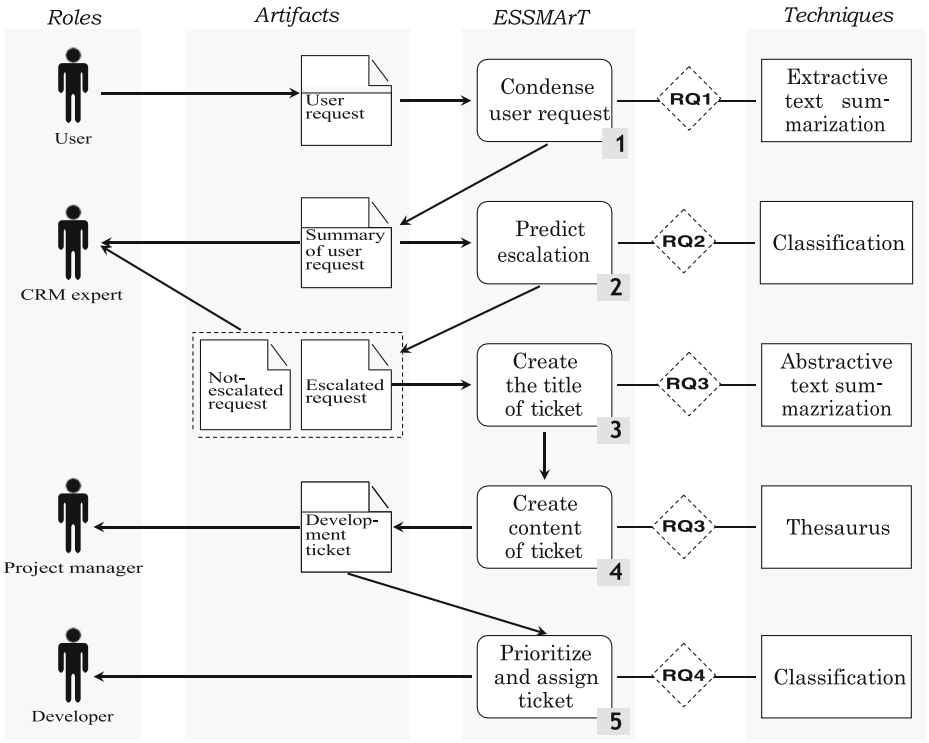

Fig. 3 Process of ESSMArT for automated generation of developer tickets from customer requests

means that the ticket length is about half the length of the original conversation, and is independent of the CRM team member who receives and responds to the customer request. To condense the conversation in the form of a customer ticket we used extractive summarization, limiting the number of sentences for our extractive summarization to five sentences. Summaries created by the CRM staff were used as the benchmark for measuring performance of our automated methods.

To select the best summarization techniques in Step 1 of ESSMArT, we compared various extractive summarization methods. The methods were selected based on their popularity in literature, in particular in software engineering research. In addition, we looked at the availability of the summarization methods as open source tools or libraries. For the frequency driven approach we used the SumBasic algorithm suggested by Vanderwende et al. (2007). On the topic word approach, we applied the Edmundson method (Edmundson 1969). For Latent Semantic Analysis (LSA) we applied the Steinberger and Jezek method (Steinberger & Jezek 2004). For Bayesian topic modeling, we used *Textrank* suggested by Mihalcea and Tarau (2004). Table 1 provides a summary and a pointer to related literature.

Also similar to Rastkar et al. (2014), we used machine learning to build an indicator representation. We built classifiers using Naive Bayes (Rish et al. 2001) as it proved to outperform other classifiers for summarization tasks (Allahyari et al. 2017). We trained the classifier only on the content of the tickets and based on the following three datasets: (i) The Enron dataset of emails, as it includes the conversation between human subjects (Carenini et al. 2007; Murray & Carenini 2008). One of the authors annotated the summary of this

dataset, (ii) the conversation between developers and users of Ubuntu on Fedora channel (denoted as UDC) with similar nature as the conversations at Brightsquid. One of the authors annotated the summary of this dataset; and (iii) the conversations between CRM team and Brightsquid customers (denoted as BSC). Two of the authors annotated these summaries.

To guide among the existing summarization techniques, we performed a pair-wise comparison of the extractive summarization methods. Second, we compared the results of automated extractive summarization with the summaries created by human experts. For this purpose, we used Recall-Oriented Understudy for Gisting Evaluation (ROUGE) (Lin 2004). ROUGE is the most widely used method for evaluation of summarization quality. While ROUGE has different variations, we followed (Allahyari et al. 2017) and used *ROUGE-n* and *ROUGE-SU*.

ROUGE-n is based on the comparison of n-grams. Within each comparison, one of the summaries is considered as the reference and the other summary, also known as the candidate, is compared against it. Within this comparison process, ROUGE-n elicits bi-grams and tri-grams:

$$ROUGE\_n = \frac{p}{q} \tag{1}$$

Where p represents the number of common n-grams between the two candidate summaries and q represents the number of n-grams that were extracted from reference summary only. *ROUGE-SU* elicits both bi-grams and uni-grams and allows the insertion of words in bi-grams. In other words, the bi-grams do not need to be consecutive sequences of words.

4.2 Predict Escalation

In Step 2 of ESSMArT, we predict whether a ticket should be escalated or not (See Fig. 3). To this end, we trained and compared the performance of three classifiers: Support Vector Machine (Hearst et al. 1998), Naive Bayes (Rish et al. 2001), and Random Forest (Liaw et al. 2002). Each of these techniques have been previously applied to solve software engineering problems, including fault prediction (Malhotra 2015) or predicting software outcomes (Cerpa et al. 2016). Our intent was not to rank the methods in general, but rather to find one that works best with the data set available. We found that Random Forest, one of the ensemble learning techniques which has been very successful in handling small-sized and imbalanced datasets (Galar et al. 2012), works well with a mixture of numerical and categorical attributes.

For training the classifiers, we used Scikit's package of Python. When applicable, we applied an exhaustive grid search over classifier parameters (such as *Kernel*, *Gamma*, and *C* values) in a way to maximize the score of the data omitted. GridSearchCV exhaustively considers all parameter combinations and optimizes parameters by cross- validated evaluation. We used the GridSearchCV function of Scikit and leveraged both the textual content of the requests as well as the non-textual content:

(i) *Using textual content of inquiries:* We trained and compared classifiers using the term frequency-inverse document frequency (TF-IDF) values of the words that make up the tickets. TF-IDF is a statistical measure frequently used in information retrieval and text mining to evaluate the importance of words in a collection of documents (Ramos et al. 2003). It consists of two components: Term Frequency (TF), which is a count of the number of times a word appears in a document, normalized by the total number of words in that document. The second component is the Inverse Document Frequency (IDF),

which is the logarithm of the number of documents in the corpus divided by the number of documents where the particular term appears.

We compared classifiers based on different text attributes. We compared the TF- IDF with (i) Bag-of-Words (BoW) (Wallach 2006) as a representation of the conversations, (ii) the extractive summaries, and (iii) a combination of the two. All three options were studied using both lemmatized and non-lemmatized tickets. Bag-of-Words is a simple approach for representing textual information. It is used to describe the occurrence of words within a document. All classifiers were run using the TF-IDF of the conversation and the extractive summaries.

(ii) *Using non-textual attributes for prediction:* We elaborated on the textual content of the customers' inquiries by using other attributes recorded alongside it in predicting the escalation and priority of tickets. To evaluate if using any of the recorded data, such as requester, organization, and the time stamp (see the full list in Table 3), could increase the accuracy of the classifiers, we used the Minimum Redundancy Maximum Relevance (mRMR) algorithm (Peng and Ding 2005). mRMR is an algorithm to select features for classifiers mRMR is an algorithm to select features for classifiers in a way that the selected features have strong correlation with the classification variable (maximum relevance), but being mutually far away from each other (minimum redundancy). This scheme has been found to be more powerful than the simple maximum relevance selection (Auffarth et al. 2010). We used an R software environment implementation of mRMR algorithm for this purpose. mRMR is superior to methods such as information gain analysis (Yu and Liu 2004) as mRMR[7] considers the relation between the attributes as well.

4.3 Create a Ticket Title and Content

Each development ticket consists of a title and a body that describes the problem. In Step 3 of ESSMArT, we used abstractive summarization to suggest ticket titles. The knowledge bases used for abstractive NLP summarization techniques most commonly utilize a large corpus and subsequently generated language model that allows for abstractive methods to perform information extraction and ranking.

We implemented the abstractive summarization using AbTextSumm (Banerjee et al. 2015). This method was designed and initially implemented by Banerjee et. al (2015). The abstractive summarization proposed by Banerjee et al. (2015) includes four main steps (i) identifying most important sentences (ii) clustering words (ii) generating k-shortest paths in each cluster using word graph, and (iv) optimized selection of words maximizing information content and readability of the summary. Abstractive text summarization is a growing field of research and its state of the art body of knowledge includes variety of techniques (Young et al. 2018). We used the method suggested by Banerjee et al. (2015) as it had superior performance in comparison to the other methods, in addition to an open source library (in Python) provided by the authors, which we used in developing ESSMArT.

[7] Comparing mRMR with Principle Component Analysis (PCA) (Wold et al. 1987) and Independent Component Analysis (ICA) (Hyvärinen and Oja 2004), mRMR does not need the mapping of features into the orthogonal and independent space.

Table 3 Attributes associated with customers' request

ID	Name	Description
Att_1	Conversation	Information provided in the ticket's conversation
Att_2	Requester	The individual who requested the ticket originally
Att_3	Ticket type	What type of request was made
Att_4	Tags	What tags are present in the ticket
Att_5	Via	What medium the ticket was introduced through
Att_6	Severity	The extent to which the ticket affects the product
Att_7	Assignee	Who was assigned to handle the ticket
Att_8	Time open	How long it has been since opening a ticket
Att_9	Time escalated	How long it has been since escalating the ticket
Att_{10}	Time to assign	How long the ticket takes to be assigned
Att_{11}	Subject	Content contained within the subject
Att_{12}	Brand name	Which product the ticket relates to
Att_{13}	Organization	Which organization is requesting the ticket

In Step 4 of ESSMArT, we built a thesaurus to map customer language into terms understandable by the developers. In a request, customers report their experience on using the software. However, a development ticket reflects the high level story in a way that is more understandable for the development team (for example to reproduce bugs). When a customer communicates with a CRM team, only the name and surname of the customer needs to be specified and the CRM team member will refer to their customer database to get related information such as the role of the customer, organization, and brand name.[8] When creating the development ticket the content should be self explanatory. To make this happen we take the summary of the conversation that we created in Step 1 of ESSMArT and:

(i) Specify brand name: The development ticket should explicitly mention which product the ticket relates to. As a result, each ticket starts with "in the XYZ systems" within which, XYZ is the name of the system. This system is the systems that have been granted access into once deploying the product.
(ii) Specify the role of the requester: The development ticket should specify the role or the requester to reflect the story. We found this by mining the satellite data around each customer. For example, if "Jane" is calling from "Crowfoot clinic" we find in the organization chart that she is the "administrator".
(iii) Abstract specific names to general entities: The specific names should be replaced and mapped to a known general entity (instead of "John Doe", it should be "patient"). In the case that we fail to map a specific name into a general entity, we eliminate it from the sentence.

Figure 4 shows this transition for a sample sentence. To make these mappings we built a thesaurus based on the Brightsquid data. To build this thesaurus we specified a set of documents that contain the related information about our entities. We used the *user stories* and *release notes* from Brightsquid as well as descriptions of the organizations using Brightsquid products. We used word co-occurrence and grammatical dependencies (Schütze 1998) using the Stanford NLP toolkit (Manning et al. 2014). Then we detected the specific names within the summaries

[8] We focused on the four products of Brightsquid among all the developed systems by them

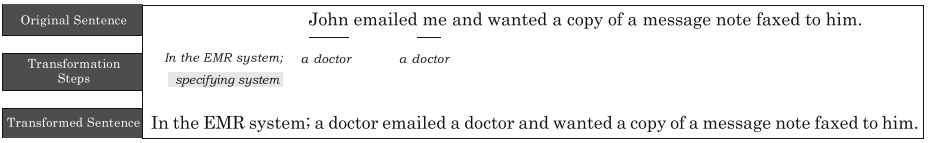

Fig. 4 An example of transforming a sentence from customer request to a development ticket using ESSMArT

using rule-based Named-Entity Recognition (NER) (Mohit 2014). Rule-based NER uses a set of named entity extraction rules for different types of named entity classes with an engine that applies the rules and the lexicons to the text (Mohit 2014).

4.4 Prioritize and Assign Tickets

Once the ticket is created, each is assigned a priority in the backlog of the project, the options being Blocker, Critical, Major, Minor, or Trivial. Then, a developer is assigned to the ticket to solve and close it. In Step 5 of ESSMArT, for both assignment and prioritization of the ticket, we reasoned by analogy. We built and compared the usage of three state of the art classifiers (Naive Bayes, SVM, and Random Forest) to predict the priority of the ticket and subsequent assignment to a developer. To find the analogy between the ticket and formerly assigned and prioritized tickets, we used the mRMR measure to select attributes among the list of Table 3 to train the classifiers. The details have been discussed in the method for Step 2 (See Section 4.2).

5 Internal Evaluation

The proposed method has been evaluated at the different steps and from two different perspectives (i.e., internal and external perspective). An overview of all evaluation done is given in Fig. 5. In this section, we discuss the different aspects of internal evaluation of ESSMArT. The analysis refers to RQ1, RQ2, RQ3, and RQ4. By its nature, this all is a retrospective analysis based on Brightsquid data.

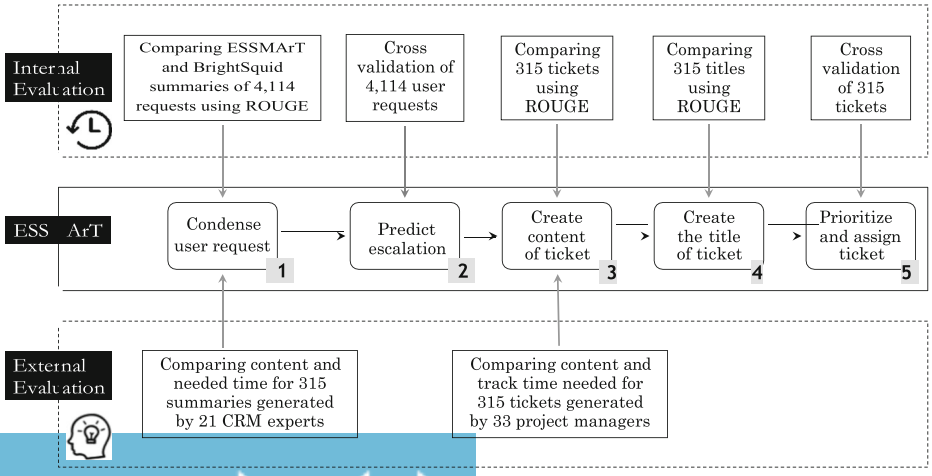

Fig. 5 Evaluation of ESSMArT

Customer requests were received by email, phone, or textual chatting and are stored in the Zendesk repository of Brightsquid. Looking over the 4,111 tickets in this study, each customer request has an average length of 152 words. This body of text has the vocabulary size of 3,276 words. 7.8% of all customer requests were escalated, summarized, and stored in a Jira repository. The development tickets have an average length of 87 words and a more focused vocabulary size of 2,021 words. For internal evaluation of ticket escalation, we applied 10-fold cross-validation and provided the average results of running it ten times.

5.1 RQ1: Summarization of Customer Requests

To condense the conversation in the form of a customer ticket we used extractive summarization. We applied and compared the results of eight summarization techniques, which covered different extractive summarization classes based on unsupervised (topic representation, indicator representation) and supervised learning (see Table 2). For supervised learning, we trained a Naive Bayes classifier on three different datasets: Email communications, Dialogues between developers and users of Ubuntu, and Brightsquid's conversation with the customers. We compared the eight techniques pairwise based on ROUGE and ROUGE-SU measures. Table 4 shows the result of pairwise comparison of the summarization techniques. Within this table, each row is compared with the technique in the column being considered as the baseline summary. As a general trend, the results of supervised learning methods (Enron, UDC, and BSC) are performing closely similar to each other. Among the unsupervised methods, TextRank almost always worked better than the others.

We further compared these methods by comparing these eight different summarization methods with summaries from CRM experts at Brightsquid. As illustrated in Fig. 6 we found that the supervised extractive summarization trained on Bright- squid data performs best, but it is only 4.2% better in terms of F1-score than the classifier trained with the Ubuntu dataset (UDC). TextRank as an unsupervised learning method performs as the third best method in our case study. Overall, Steinberger is the worst performing classifier. It is 20% less accurate than BSC in terms of the F1-score. Considering the effort needed to prepare training sets, unsupervised methods are proven to be an alternative to supervised methods.

Supervised learning based on context specific Brightsquid data performs best for summarizing customer request and outperforms the best unsupervised technique by 10%.

5.2 RQ2: Predicting Escalation

We compared three state-of-the art classifiers that exhibit good performance with a short text to predict the escalation of a user's requests, the priority of the tickets, and the developers assigned to it.

We used mRMR to select non-textual content for predicting escalation. However, we did not find any of the attributes Att_2 to Att_{13} significantly increased the accuracy of the predictive model (Step 2 of ESMMArT). This is aligned with the current practice at Brightsquid. In Brightsquid the content of the conversation, specifically the communicated concern by the customer, is the only decisive factor for the CRM team to escalate a ticket to the development team. As a result, we focused our effort on automating escalation only on the content of the tickets.

Table 4 F1-score of extractive techniques for summarizing customer requests

	ROUGE								*ROUGE-SU*							
Method	SumBasic	Edmondson	Steinberger	LDA	TextRank	Enron	UDC	BSC	SumBasic	Edmondson	Steinberger	LDA	TextRank	Enron	UDC	BSC
SumBasic	–	0.78	0.8	0.8	0.82	0.76	0.73	0.71	–	0.74	0.76	0.77	0.79	0.7	0.66	0.63
Edmondson	0.79	–	0.76	0.86	0.83	0.77	0.73	0.73	0.75	–	0.72	0.82	0.8	0.73	0.69	0.64
Steinberger	0.78	0.85	–	0.86	0.79	0.74	0.7	0.66	0.74	0.79	–	0.82	0.77	0.68	0.68	0.65
LDA	0.82	0.78	0.81	–	0.8	0.79	0.75	0.7	0.79	0.74	0.77	–	0.78	0.7	0.66	0.6
TextRank	0.79	0.73	0.78	0.8	–	0.74	0.69	0.67	0.77	0.7	0.75	0.78	–	0.69	0.65	0.6
Enron	0.65	0.66	0.7	0.7	0.73	–	0.81	0.79	0.69	0.7	0.71	0.7	0.71	–	0.83	0.81
UDC	0.64	0.63	0.68	0.65	0.63	0.94	–	0.91	0.61	0.61	0.63	0.6	0.61	0.85	–	0.84
BSC	0.6	0.61	0.6	0.59	0.61	0.95	0.96	–	0.55	0.55	0.52	0.56	0.59	0.88	0.88	–

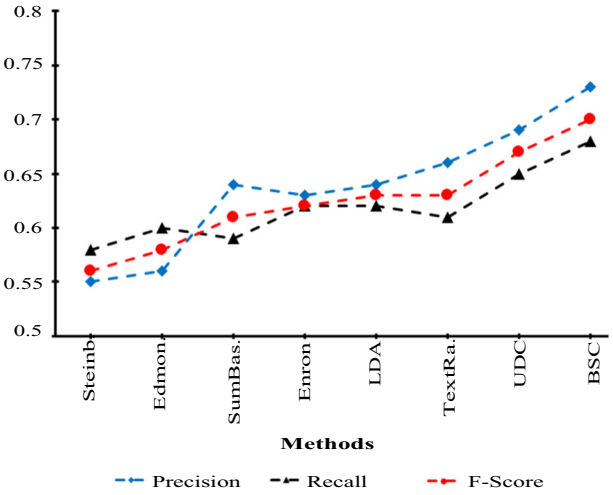

Fig. 6 Performance of extractive summarization techniques in comparison with human generated summaries using ROUGE-SU

To deal with the imbalanced number of escalated and non-escalated tickets, we down-sampled the non-escalated conversations. We benchmarked different techniques for increasing the accuracy of the textual similarity analysis such as customizing the list of stop words, lemmatization, stemming, BOW (bag-of-words), TF-IDF, and used extractive summaries for predicting escalation. Overall, we found that by using tf-idf the F1-score of the three classifiers is better by 8.7% on average. Also, we found that by using lemmatization instead of Porter's stemming, the F1-scores were improved by 4.3% on average. Our results also showed that eliminating a customized set of stop words from the conversation increases the accuracy of the classifier by 6.1% on average. Table 5 summarizes the results of the classification techniques using the 4,114 conversations to predict ticket escalation. Within the tables, the numbers in italic font represent the corresponding top values. The confusion matrix is presented in Appendix 2.

Table 5 Evaluation of classification algorithms for predicting tickets' escalation

	Precision	Recall	F1
Naive Bayes			
Conversation	0.83	0.45	0.58
Conversation + Lemmatization	0.85	0.49	0.62
Extractive summary + Lemmatization	0.82	0.43	0.57
Conversation + Extractive summary + Lemmatization	0.86	0.53	0.65
Support Vector Machine (SVM)			
Conversation	0.64	0.43	0.51
Conversation + Lemmatization	0.62	0.45	0.52
Extractive summary + Lemmatization	0.60	0.42	0.49
Conversation + Extractive summary + Lemmatization	0.62	0.49	0.54
RandomForest			
Conversation	0.88	0.49	0.62
Conversation + Lemmatization	0.89	0.53	0.66
Extractive summary + Lemmatization	0.89	0.42	0.56
Conversation + Extractive summary + Lemmatization	*0.90*	*0.55*	*0.68*

The classifiers used were Support Vector Machine, Naive Bayes, and Random Forest. For each of them, four alternatives were evaluated. Looking at the F1 measure as the one balancing precision and recall, we found that the combination of just looking at the conversation and the extractive summary in combination with lemmatization performed the most promising. When comparing the classification techniques, Random Forest performs best in terms of precision, and best in F1 on the best configuration. In contrast, SVM seems to be the lowest performing among the three techniques overall.

Random Forest classifier trained on a combination of conversation and extractive summarization outperforms the other models in terms of precision. Using the extractive summaries increases the F1-score of the prediction.

5.3 RQ3: Quality of Automatically Generated Ticket Content

Each development ticket consists of content and a title. ESSMArT suggests the ticket content by using a thesaurus for mapping and by generalizing entities from extractive summaries of **RQ1** as input. The ticket title is created from using abstractive summarization. We report the results of evaluating the content and title of these tickets.

Evaluating the Content of the Development Tickets To bridge between the customer request and the development ticket we built a thesaurus that maps the customer terminology and specific names into the developers' terminology or general entities.

For building the thesaurus we used 304 separate documents including customer stories, release notes, organization and brand descriptions, and team descriptions. As the result of this automated process, we built a thesaurus with 3,301 entries. We manually went through this thesaurus and in particular searched for the personnel names. We mapped each specific personnel name to an organizational role. For each distinct specific name (1,908 out of 2,467) we entered three separate entries for the first name, surname and the name as a whole. We ended up with a thesaurus with 7,117 entries in total. We then used ROUGE-SU for evaluating the alignment of the generated tickets with the tickets extracted by human experts.

Figure 7 shows the results of the comparison between automated and manually created tickets. In a majority of cases, the precision of the classifiers is better than their recall. Looking into the F1-Score, the combination of the supervised learning method and thesaurus performs best. Interestingly, unsupervised methods based on the graph model of represented indicators (TextRank (Mihalcea and Tarau 2004)) combined with the use of our thesaurus performed well.

Supervised learning trained on Brightsquid data combined with the use of a thesaurus performs best among all techniques. The accuracy is 5% better than the best unsupervised method in terms of F1-score.

Evaluating the Title of the Development Tickets We compared the titles created using abstractive summarization with the human-generated titles using ROUGE-SU. Within the process of creating the ticket titles, we limited the size of the abstractive summary to 11 words as it was the average length of the ticket titles in Brightsquid. We presented the results of this comparison in Table 6. Training the model based on the Brightsquid data performed the best among the other models having an F1-score of 0.65.

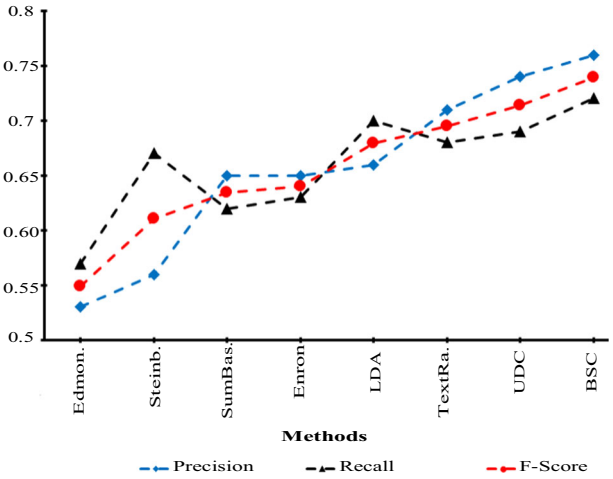

Fig. 7 Comparison of the content of the development tickets generated by human experts with the ones generated with different summarization techniques and the thesaurus in RQ3 using ROUGE-SU

5.4 RQ4: Prioritization and Assignment of Tickets

We compared three state of the art classifiers with good performance proven in other contexts. We used the ticket title to predict the priority of the escalated tickets and assign it to a developer.

5.4.1 Prediction of the Priority of Escalated Tickets

Several attributes are recorded along with the customer requests as shown in Table 3. The mRMR analysis showed the importance of *organization* and *brand name* for predicting the priority of the tickets. We compared three classifiers, Naive Bayes, Support

Vector Machine (SVM), and Random Forest. We benchmarked the performance of these classifiers using the conversation between customers and Brightsquid, the extractive and abstractive summaries of the apps, as well as the organization and brand name attributes. Similar to the escalation prediction in the previous section, we evaluated the impact of different text pre-processing and processing methods. In all cases, lemmatization was applied. The results of our benchmark are shown in Table 7. The confusion matrix is presented in Appendix 2.

When using the textual content of the conversation only, Random Forest classifiers have a slightly better performance in comparison to Naive Bayes. Using extractive and abstractive summaries for predicting the tickets priority increased the F1-score of the classifiers up to 8.6% on average. Having abstractive summarization on top of that further increased the F1-

Table 6 Ticket titles using ESSMArT summarization in comparison with human experts using ROUGE-SU

Abstractive summarization	Precision	Recall	F1
Enron	0.61	0.57	0.59
UDC	0.56	0.53	0.54
BSC	0.68	0.63	0.65

Table 7 Evaluation of classification algorithms for predicting tickets' priority

	Precision	Recall	F1
Naive Bayes			
Conversation	0.64	0.61	0.62
Extractive summary	0.64	0.62	0.63
Conversation + Extractive summary	0.68	0.63	0.65
Abstractive summary + Extractive summary	0.68	0.66	0.67
Conversation + Abstractive summary + Extractive summary	0.68	0.66	0.67
Abstractive summary + Extractive summary + Organization+ Brand name	*0.74*	*0.72*	*0.73*
Support Vector Machine			
Conversation	0.52	0.51	0.51
Extractive summary	0.52	0.53	0.52
Conversation + Extractive summary	0.55	0.53	0.54
Abstractive summary + Extractive summary	0.56	0.53	0.54
Conversation + Abstractive summary + Extractive summary	0.58	0.55	0.56
Abstractive summary + Extractive summary + Organization + Brand name	0.70	0.67	0.68
Random Forest			
Conversation	0.64	0.62	0.63
Extractive summary	0.66	0.63	0.64
Conversation + Extractive summary	0.66	0.63	0.64
Abstractive summary + Extractive summary	0.69	0.66	0.67
Conversation + Abstractive summary + Extractive summary	0.70	0.68	0.69
Abstractive summary + Extractive summary + Organization+ Brand name	0.73	0.70	0.71

score. Moreover, using *organization* and *brand name* increased the F1-score by up to 12.3%. When comparing across all pre-processing and processing options, a Naive Bayes classifier using both textual and non-textual attributes performs best.

5.4.2 Assignment of Tickets to Developers

Similar to what we did for predicting prioritization, we compared the three state of the art classifiers with multiple textual attributes. The results of this benchmarking are shown in Table 8. The results showed that similar to the ticket prioritization, using the content of the conversation along with the abstractive and extractive summaries performs the best and Naive Bayes outperforms SVM and Random Forest in this task. The confusion matrix is presented in Appendix 2.

We used mRMR to select non-textual features (as listed in Table 3 and found *brand name* as an important factor for ticket assignment. On average, using the *brand name* on top of the textual features increased the F1-Score of the classifiers.

Using Naive Bayes perform the best for prioritizing (F1-score = 0.73) and assigning the tickets (F1-score = 0.86). The results showed that using extractive and abstractive summarization along with other features increases the accuracy of these predictions.

6 External Evaluation

So far, we built and compared the state-of-the-art techniques known for the different stages in the process of managing customer requests. As a form of retrospective analysis, we called that *internal evaluation*. However, the question of the perceived value of applying the method is still

Table 8 Evaluation of classification algorithms for assigning tickets to developers

Naive Bayes	Precision	Recall	F1
Conversation	0.8	0.83	0.81
Extractive summary	0.81	0.83	0.82
Conversation + Extractive summary	0.85	0.83	0.84
Abstractive summary + Extractive summary	0.85	0.84	0.84
Conversation + Abstractive summary + Extractive summary	0.87	0.84	0.85
Abstractive summary + Extractive summary + Brand name	*0.89*	*0.84*	*0.86*
Support Vector Machine			
Conversation	0.68	0.61	0.64
Extractive summary	0.68	0.62	0.65
Conversation + Extractive summary	0.68	0.63	0.65
Abstractive summary + Extractive summary	0.69	0.65	0.67
Conversation + Abstractive summary + Extractive summary	0.71	0.67	0.69
Abstractive summary + Extractive summary + Brand name	0.74	0.68	0.71
Random Forest			
Conversation	0.72	0.73	0.72
Extractive summary	0.76	0.73	0.74
Conversation + Extractive summary	0.76	0.75	0.75
Abstractive summary + Extractive summary	0.78	0.74	0.76
Conversation + Abstractive summary + Extractive summary	0.8	0.78	0.79
Abstractive summary + Extractive summary + Brand name	0.83	0.78	0.8

open. In this section, ESSMArT is evaluated by CRM experts and project managers. The section is closely related to RQ5, and is called *external evaluation*. The subjects are asked whether ESSMArT makes the process of escalation faster and better. As we did not have access to employees of Brightsquid, we recruited 21 CRM experts and 33 project managers from outside. We used convenient sampling for recruiting participants from social media to participate in this study. The whole external evaluation is described in the subsequent subsections.

6.1 Protocol for External Evaluation of ESSMArT

To evaluate the performance of ESSMArT, we asked CRM experts and project managers to go through the escalation of a sample set of customer requests, first without and then with using ESSMArT. Offering the task through social platforms, we attracted 21 CRM experts and 33 project managers to participate. The CRM experts participating in our study had 4.5 years of experience in a related job on average with a minimum of 18 months and a maximum of 13 years of experience. The participating project managers had an average of 6.2 years of experience in a related job, with a minimum of 4 years and a maximum of 16 years. We assigned 15 escalated customer requests to each CRM expert to perform the evaluation.

For the evaluation, we first performed a manual process for escalating tickets and then we used ESSMArT:

Manual Process We provided the complete conversation of 315 Brightsquid customer requests and asked to provide a summary. Each CRM expert evaluated 15 anonymized conversations and was asked to decide if each ticket should be escalated. Furthermore, we asked each to provide an extractive summary by selecting a subset of sentences of the conversation, without applying any rewording (Rastkar et al. 2014). We recorded the time needed per ticket called *Escalationtime*.

We submitted the summary of 315 customer requests to the 33 project managers participating in our experiment. Each project manager prepared a development ticket based on the summarized customer request. We recorded this time as *Decisiontime*. The screenshot of the survey for manual escalation is shown in Fig. 8.

ESSMArT Way We provided the prototype implementation of ESSMArT to the CRM experts and the project managers. We logged the time taken by them to perform each step. To make the results comparable with the manual process, we did not allow any CRM expert or project manager to work on any ticket they had already handled in the manual process. Figure 9 shows the screenshot of ESSMArT for a sample request. The left screen shows the ESSMArT UI for CRM experts used to log the *Escalationtime* while the right one is the UI shown to the project managers and logged as *Decisiontime*.

6.2 RQ5: Usefulness of ESSMArT for Experts

For the external evaluation, we surveyed experts to find the usefulness of the results and to figure out if the process would be faster for humans using our prototype tool.

6.2.1 Are the ESSMArT Results Aligned with Perception of the External Experts?

We compared the sentences selected by CRM experts with those selected by ESSMArT. Based on the results of internal evaluation in Section 5, we used Random Forest trained with former Brightsquid data for summarization. We asked survey participants in the role of CRM to select the most representative sentences. Comparing the selected sentences selected by ESSMArT and CRM experts resulted in 0.71 precision and 0.77 recall ($F\ 1 - score = 0.73$). Figure 10 - (a) shows the distribution of the number of sentences that were selected differently between ESSMArT and human experts. To evaluate the alignment of the tickets generates by ESSMArT with those generated by human experts, we tracked the number of words changed by project managers on the ESSMArT generated ticket. Figure 10 - (b) shows the distribution of the changed words for the 315 customer requests. In 25% of the tickets no word were

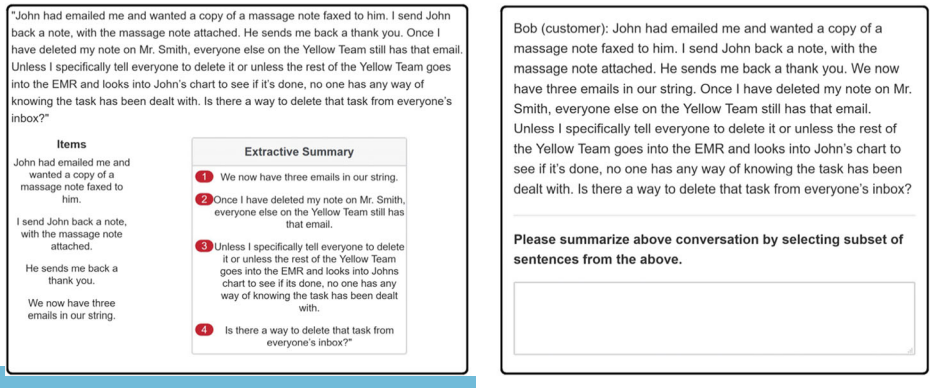

Fig. 8 Process of manual escalation for evaluation of ESSMArT. The left screen shot was shown to the CRM experts while the right one was shown to the project managers

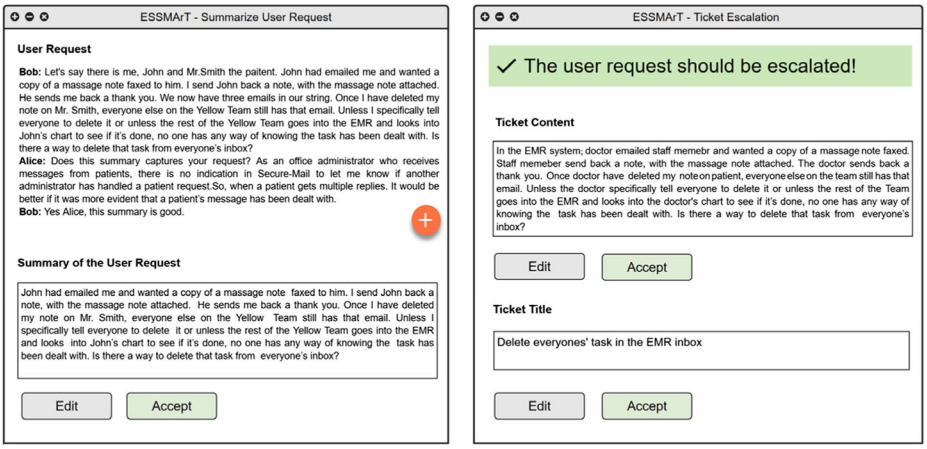

Fig. 9 Applying the prototype tool of ESSMArT, the left figure shows the summarization of a sample customer request. The right one suggests the content and title of the sample development ticket

changed and in 8.5% of the cases, more than 10 words were changed. Overall, across 315 customer requests, 3.7 words on average were changed by human experts.

Figure 11 shows the results of the questions asked to the 54 survey participants. The survey questions are presented as Appendix 3 to the paper. We asked the participants how much they agree that the ESSMArT results were understandable. Only 1.9% (one) of the participants disagreed with this statement. 51.9% of the participants stated they likely or very likely would use ESSMArT in practice. Trusting decision support tools is a common problem in their usability (Du and Ruhe 2009). 68.6% of participants stated that they trust the ESSMArT results while 7.5% of them stated it is unlikely that they would trust the results.

ESSMArT provides suggestions that are 71% aligned to the selection of human experts. The average change of 7.5% words per ticket also demonstrates the usefulness of its results for project managers.

6.3 Does ESSMArT Make the Escalation Process Faster?

We logged the *Escalationtime* and *Decisiontime* for 315 customer requests when done completely manual and by using ESSMArT. Figure 12 shows the the time taken for each of

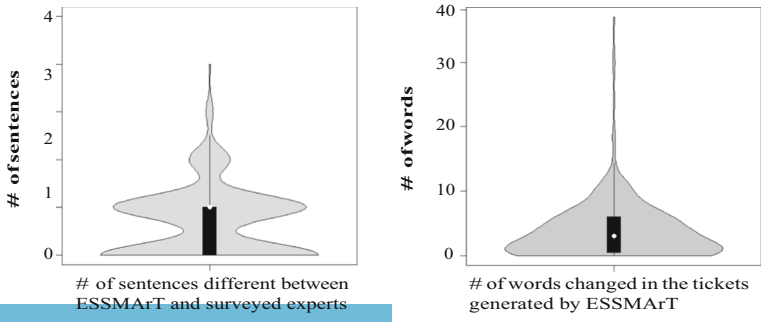

Fig. 10 Analysis of the response to the survey questions from 21 CRM experts and 33 project managers in terms of the number of different sentences (left) and the number of words changed (right)

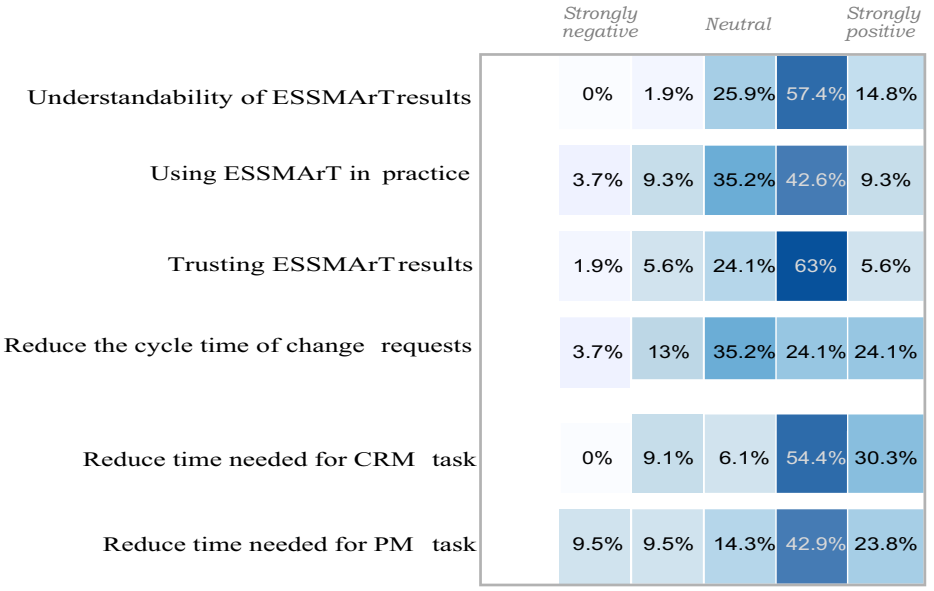

Fig. 11 Analysis of responses from 21 CRM experts and 33 project managers having participated in the survey

these tasks. Figure 12 - (a) shows the *Escalationtime* for CRM experts with and without ESSMarT. Using ESSMArT reduces *Escalationtime* by 3.2 minutes on average, per ticket:

(i) *ESSMArT* (*Escalationtime*) < *Manual*(*Escalationtime*): For 297 (94.2%) of the requests, ESSMArT allowed CRM experts do the task faster.

(ii) *ESSMArT* (*Escalationtime*) ≥ *Manual*(*Escalationtime*): For 18 (5.8%) of the customer requests ESSMArT appeared not helpful in making the process faster. In these cases, the manual process took on average 0.35 s less time for escalation. Considering the small number of cases, small time difference between cases, and the same ticket having been escalated by different participants for manual and ESSMArT enabled process, we attribute variances to differences in participant cognitive abilitities as well as possible participant distractions.

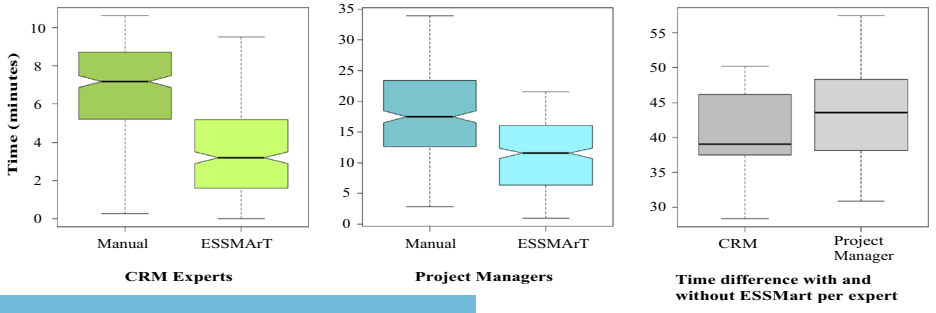

Fig. 12 The logged time when the escalation is done manually versus escalation using ES- SMArT. This is the sum of the time taken by CRM experts and by project managers in categorizing 315 change requests

Similarly, we logged and compared the *Decisiontime* for the participated project managers. In this case, the *Decisiontime* has been improved in all cases. Using ESS- MArT allowed project managers to decide on average 6.3 minutes faster in comparison to the manual process. Figure 12-(b) shows the boxplot for the time taken by project managers to decide on a ticket with and without using ESSMArT.

Independently, we asked the survey participants how likely ESSMArT usage would reduce the cycle time of change requests. 48.2% of the participants agreed or strongly agreed that ESSMArT reduces the time needed for escalating a customer request. Of the CRM experts, 54.4% agreed and 30.3% strongly agreed that ESSMArT make their job faster and 42.9% of the project managers agreed with 23.8% strongly agreeing about the same for their escalation tasks. Figure 12-(c) shows the total time saved for each expert across all the tickets.

ESSMArT reduces the escalation time of a change request by 9.2 min on average. 84.7% of the CRM experts and 66.7% of project managers agreed or strongly agreed that ESSMArT helps them to perform the task faster.

7 Discussion

7.1 Scope of ESSMArT

We focused on a system to support the decision of CRM experts and the project managers with the intent to increase the satisfaction of the end users (customers). Software engineering literature is rich in providing decision support and recommendations for software developers for different tasks (Robillard et al. 2014), such as finding the resolution of a reported bug or assigning developers to bugs. Nowadays, there is often some level of automation for development decisions in software companies (like Brightsquid). In this study, we have not studied the overhead for integrating ESSMArT with such existing systems. On the other hand, we do not have any evidence or indication if the full automation of the whole pipeline is desirable as that would involve more stakeholders in a single decision support system and increase the cost of maintenance. Hence, one might gather evidence on the pros and cons of extending the scope of ESSMArT to support development decisions and possibly extend ESSMArT.

7.2 Criteria for Selecting Machine Learning Models

Understandability of the Results With the recent advances in machine learning state of the art and practice there is an increasing temptation to use the methods that result in the highest precision and accuracy in the results. Understanding the logic and process that leads into a particular machine learning result often needs solid scientific understanding of the underlying model and techniques. To this end, the decision maker should understand the reasoning behind automated models to trust and use the results. In the case of this study with Brightsquid, we assisted three types of decision makers: CRM staff, CRM chief, and project manager. They all confirmed the need for understandability of the automation results for adopting and integrating ESSMArT.

Building on Top of the Existing Practices We emphasized the importance of reusing the state of the art practices. We selected and used techniques and data sets that were

replicable and preferably used the open source implementation of the techniques. While this was not possible for all the benchmarked techniques, most of the ESSMArT modules are based on open libraries.

7.3 Training machine learning models

We studied the usefulness of features (textual and non-textual) in increasing the accuracy of our machine learning models. However;.

- In predicting escalation (Section 5.2), none of the Att_2 to Att_{13} significantly increase the accuracy of our predictive model (Step 2) hence we relied on content of the conversations only. This was also aligned with the current practice at Bright- squid.
- The mRMR analysis showed the importance of Organization (Att_{13}) and brand name (Att_{12}) in predicting the priority of the tickets (Section 5.4.1).
- Using the same method, we found that only brand name (Att_{12}) is an important feature for predicting the assignment of the tickets (Section 5.4.2).

In a nutshell, our study showed that using more features does not necessarily imply better performance of machine learning techniques. We follow the argumentation of Lemberger et al. (Lemberger and Morel 2013) that simpler models make it easier for users to understand and accept the suggestions made by them.

8 Limitations and Threats to Validity

The ESSMArT process described in Fig. 2 is general enough to be applicable beyond Brightsquid. One key prerequisite for the applicability is the existence of both a customer ticket as well as a developer ticket system. In any case, the results of our study should be treated with caution due to the existing threats to validity discussed below.

Construct Validity To measure the quality of summarization techniques for condensing customer request we used the ROUGE metrics as it was shown valid results in similar contexts before. In most of the comparisons, we compared the result of summarization with the content generated by human experts at Brightsquid, retrospectively. For training machine learning extractive summarization, we used a human annotator which her annotation might have been biased and impose threats to the validity of the results Ideally, the threat would be mitigated by involving more than one person in the annotation process (e.g as done in (Nayebi et al. 2017)). In former studies, with the intent to find the best summarization method, researchers asked external developers to take the bug reports and select a subset of the sentences. We extended the scope of the evaluation and intended to provide evidence that the results of our automation could be used to assist human experts.

We trained and compared classifiers using the term frequency-inverse document frequency (TF-IDF) values of the words that make up the tickets. We did not experiment with more advanced, sematic-exploring methods word2vec and doc2vec. First, they are requiring higher effort. Second, we were following the “Principles of Industrial Data Mining” arguments of Menzies et al. (2011) emphasizing the importance of users versus algorithms. The main

argument was that the "return-on-investment" of ESSMArT was evaluated very positive from an industrial perspective (see Section 6), so we did not further invest substantially into refinements of algorithms. We needed to explain the methods clearly and the achieved results to make the industry admittance possible.

External Validity Selection of techniques for ESSMArT and validation of the different steps were closely connected with the real-world data set of 4,114 tickets. At each step of ESSMArT we compared and selected among state of the art methods. However, our study and the results are limited to the context of Brightsquid. The challenge for testing ESSMArT in other contexts is access to the holistic dataset of the whole process. This limits our ability to provide evidence on the generalizability of the method. The process of managing customer requests is not unique to BrightSquid which makes ESSMArT useful for other software companies.

Also, we used convenient sampling which imposes the risk of a selection bias and thus causing a lack of credibility in general. However, it was considered acceptable as it just served as an initial evaluation for exploratory purposes (Kitchenham and Pfleeger 2008).

Internal Validity To provide evidence that ESSMArT makes the escalation process faster, we used experts from outside Brightsquid. We mitigated this risk by careful screening of the participants and their background.

Conclusion Validity We analyzed the tickets of Brightsquid within a limited time frame available. The methods of the company might change over the lifetime of the projects, and the same is true for their customers. While we compared the length of the tickets for different CRM employees, the CRM manager, and the project manager were always the same person. A change in the staff might slightly change the results presented in this paper caused by the difference in their interaction with customers, text, and tickets. Also, for a few customer requests (18 cases), ESSMArT appeared to be not helpful in making the process faster. Considering the small number of cases and the small amount of slowdown, we believe that these rare cases happen because of the difference between the cognitive ability of the participants and possible distractions.

9 Conclusions and Future Work

ESSMArt addresses customer request management from a holistic perspective and supports decision-makers by providing them with intelligent suggestions within the process. CRM is a key factor for keeping customers satisfied. The main intention of ESSMArT was NOT to fully automate and completely exclude humans from the process. However, Brightsquid was highly interested in reducing the resource bottleneck, while keeping the humans in the loop.

We looked into the whole process from the time of receiving a request or question from a customer to the time that the problem is resolved by the CRM or a task is assigned to a developer. This is unique as the existing research has been focused on the steps of this process in isolation only. The method development was inspired by the industrial collaboration project with Brightsquid. However, its underlying process is following the general steps of customer request management and thus is applicable more broadly. We believe that automating the management of customer requests and their escalation would increase the chance of innovation within organizations (Nayebi and Ruhe 2014).

We consider the results as necessary, but not sufficient for claiming external validity. More empirical evaluation of the individual steps of the method as well as on the impact of the whole method is required. In particular, as the data is coming from one company only, the evaluation needs to be extended to other environments. Also, the existing prototype tool was intended to perform an initial evaluation and needs to be further enhanced and more comprehensively tested.

Acknowledgments This research was partially supported by the Natural Sciences and Engineering Research Council of Canada, NSERC Discovery Grant RGPIN-2017-03948 and the NSERC Collaborative Research and development project NSERC-CRD-486636-2015. We appreciate the discussion with and suggestions made by *Lloyd Montgomery* and acknowledge all the comments and suggestions made by the anonymous reviewers.

Appendix 1 - Illustrative Example for Comparison of Processes

In this appendix we provide an anonymized Brightsquid customer support example to compare the traditional process of managing customer requests with the ESSMArT process.

Traditional Process of Managing Customer Requests at Brightsquid

Customers report issues to Brightsquid via telephone, chat or email, and issues are recorded in Zendesk. In this example, illustrated in Figure 13 below, the customer (Bob) contacts the CRM team, concerned about the management of secure messages in a team environment. The CRM agent (Alice) elaborates on the problem and provides a summary for the customer's approval. Once Bob approves the description, Alice decides whether to escalate the ticket or resolve it on her own. If unable to determine a solution, Alice escalates the issue and informs Bob. The CRM manager (Carol), who is responsible for resolving or further escalating issues to the development project manager, reads through the ticket, expands upon the description as necessary, and escalates the issue to the project manager. The project manager (Erin) defines the ticket's priority and assigns it to the most appropriate developer for resolution.

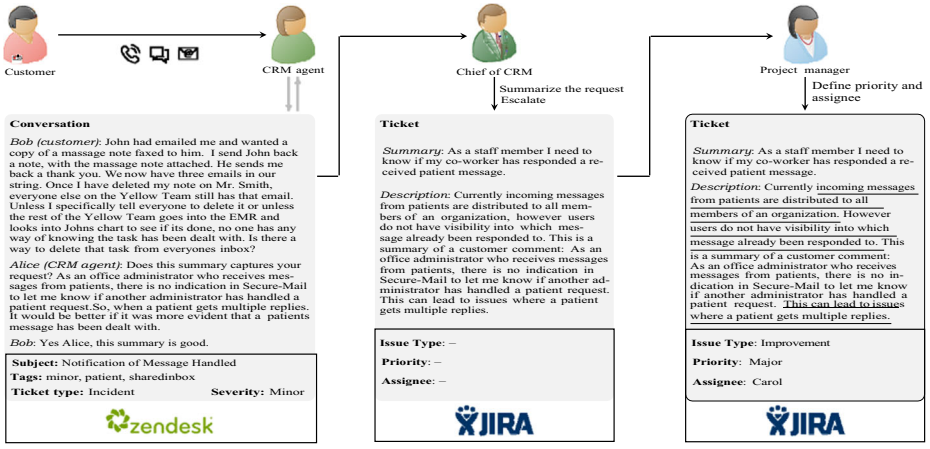

Fig. 13 Process in BS showing the conversational structure of a customer request at Bright- Squid. The underlined sentences have been chosen by our annotators to train machine learning based summarization techniques

Managing Customer Requests with ESSMArT

When Bob's request is received, Alice elaborates the description further with the customer, as above. ESSMArT consequently summarizes the conversation as below:

Zendesk ticket (summary of the customer request)
John emailed me and wanted a copy of the a message note faxed to him. I sent John back a note, with the message note attached. He sends me back a thank you. Once I have deleted my note on Mr. Smith, everyone else on the Yellow Team still has that email. Unless I specifically tell everyone to delete it or unless the rest of the Yellow Team goes into the EMR and looks into John's chart if it's done, no one has any way of knowing the task has been dealt with. Is there a way to delete that task from everyone's inbox?

ESSMArT then escalates the ticket to Carol, who receives a notification comprising the ESSMArT summary and escalation recommendation. If Carol agrees, ESSMArT creates a Jira development ticket by assigning a title using abstractive summarization (Check Figure 4 for a detailed example), assigning ticket priority, assigning a specific developer, and notifying Erin.

Jira ticket (development tickets)
Title: Delete everyone task in the EMR inbox
Content: In the EMR system; a doctor had emailed a doctor and want a copy of a message note faxed to him. Staff member send back a note, with the message attached. The doctor sends back a thank you. Once doctor delete his note on patient, everyone else on the team still has that email. Unless the doctor specifically tell everyone to delete it or unless the rest of the team goes into the EMR and looks into the doctor's chart to see if it's done, no one has any way of knowing the task has been dealt with. is there a way to delete that task from everyone's inbox?
Priority: Major
Assignee: Jane Doe

Once Erin agrees, the ESSMArT ticket is added to the Jira backlog. Screenshots of this example in ESSMArT were previously presented in Figure 9.

Appendix 2 - Confusion Matrices

In Tables 5, 7, and 8, we presented precision, recall, and F1-score of three state-of-the art classifiers (Naive Bayes, SVM, and Random Forest) to predict ticket escalation, prioritize the escalated tickets, respectively assign them to a developer. These are the results of ten times of running cross validation and we provided the averages. In addition to that, we provide the aggregated confusion matrix for the classifiers with the best performance and for all the three types of prediction. These confusion matrices are presented below.

Confusion matrix for ticket escalation

		Predicted	
		Yes	No
Actual	Yes	218	24
	No	103	297

The False-Positives demonstrate that the classifier mistakenly considered a ticket is escalated while in fact it has not been escalated. The False-Negatives indicate the tickets that should have been escalated but were not detected by the classifier. The False-positives may add additional effort to the development team while the False- Negatives may result in customer's dissatisfaction by not properly addressing their problem.

Confusion matrix for ticket assignment
**Aggregated across eight different classes*

		Predicted	
		Yes	No
Actual	Yes	218	40
	No	26	41

The False-Positives may result in more work for developers as the ticket may be assigned to a developer with not enough expertise. False-Negatives may result in more time and effort to fix the ticket as the developer with less expertise would handle the ticket.

Confusion matrix for ticket prioritization
**Aggregated across five different classes*

		Predicted	
		Yes	No
Actual	Yes	133	49
	No	44	101

The False-Positives and Negatives would delay fixing some of the important customers' concerns prioritize lower or stuck behind lower priority tickets.

Appendix 3 - Survey Questions

The user study to evaluate ESSMArT was conducted in two main parts: First, using the prototype tool of ESSMArT using the data from Brightsquid for evaluation purpose, and second, asking questions to understand the perception of participants about the usability of ESSMArT in practice. In this appendix, we present the five questions raised to understand the perception of users about ESSMArT.

Figure 11 shows the results of this survey and the perception of CRM experts and project managers. The sample of questions and the screenshots of the used prototype were presented in Fig. 8 respectively Fig. 9.

1. How understandable did you find the results of ESSMArT?
   5- Very understandable 4- understandable 3- Somewhat understandable but slightly ambiguous 2- Somewhat understandable but mostly ambiguous 1- Not understandable

2. How likely you would use ESSMArT in practice?
   5- Definitely 4- Very likely 3- Maybe 2- not likely 1- Definitely not
3. To what extent you trust and rely on the ESSMArT results?
   5- Totally trust it 4- Trust it 3- Neutral 2- Not trust it 1- Not trust it at all
4. To what extent are you agree that ESSMArT reduces the time for deciding on a change request?
   5- Strongly agree 4- Agree 3- Neutral 2- Disagree 1- Strongly disagree
5. To what extent are you agree that ESSMArT reduces the time needed for CRM/PM tasks?
   5- Strongly agree 4- Agree 3- Neutral 2- Disagree 1- Strongly disagree

References

Allahyari M, Pouriyeh S, Assefi M, Safaei S, Trippe ED, Gutierrez JB, Kochut K (2017) Text summarization techniques: a brief survey. *arXiv preprint arXiv:1707.02268*

Anvik J (2016) Evaluating an assistant for creating bug report assignment recommenders, vol 1705, pp 26–39

Anvik J, Hiew L, Murphy GC (2006) Who should fix this bug? In: Proceedings of the 28th international conference on Software engineering. ACM, pp 361–370

Auffarth B, López M, Cerquides J (2010) Comparison of redundancy and relevance measures for feature selection in tissue classification of ct images. In: Industrial conference on data mining. Springer, pp 248–262

Bandera H, Bell DA, Little AD, York BB (2018) Increasing efficiency and effectiveness of support engineers in resolving problem tickets, Apr. 19 2018. US Patent App. 15/293,988

Banerjee S, Mitra P, Sugiyama K (2015) Multi-document abstractive summarization using ilp based multi-sentence compression. In: IJCAI, pp 1208–1214

Batista J, Ferreira R, Tomaz H, Ferreira R, Dueire Lins R, Simske S, Silva G, Riss M (2015) A quantitative and qualitative assessment of automatic text summarization systems. In: Proceedings of the 2015 ACM Symposium on Document Engineering, DocEng '15. ACM, New York, pp 65–68

Bruckhaus T, Ling CX, Madhavji NH, Sheng S (2004) Software escalation prediction with data mining. In: Workshop on predictive software models (PSM 2004), A STEP Software Technology & Engineering Practice

Carenini G, Ng RT, Zhou X (2007) Summarizing email conversations with clue words. In: Proceedings of the 16th international conference on world wide web. ACM, pp 91–100

Cerpa N, Bardeen M, Astudillo CA, Verner J (2016) Evaluating different families of prediction methods for estimating software project outcomes. J Syst Softw 112:48–64

Das, Martins AF (2007) A survey on automatic text summarization. Literature Survey for the Language and Statistics II course at CMU 4:192–195

Du, Ruhe G (2009) Does explanation improve the acceptance of decision support for product release planning? In: Empirical Software Engineering and Measurement, 2009. ESEM 2009. 3rd International Symposium on, pages 56–68. IEEE

Edmundson P (1969) New methods in automatic extracting. Journal of the ACM (JACM) 16(2):264–285

Galar M, Fernandez A, Barrenechea E, Bustince H, Herrera F (2012) A review on ensem- bles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Re- views) 42(4):463–484

Gambhir M, Gupta V (Jan 2017) Recent automatic text summarization techniques: a survey. Artif Intell Rev 47(1):1–66

Gupta V, Lehal GS (2010) A survey of text summarization extractive techniques. Journal of emerging technologies in web intelligence 2(3):258–268

Hearst MA, Dumais ST, Osuna E, Platt J, Scholkopf B (1998) Support vector machines. IEEE Intelligent Systems and their applications 13(4):18–28

Hyvärinen JK, Oja E (2004) Independent component analysis, vol 46. Wiley

Jha N, Mahmoud A (2018) Using frame semantics for classifying and summarizing application store reviews. Empir Softw Eng:1–34
Jonsson L, Borg M, Broman D, Sandahl K, Eldh S, Runeson P (2016) Automated bug assignment: ensemble-based machine learning in large scale industrial contexts. Empir Softw Eng 21(4):1533–1578
Kabeer SJ, Nayebi M, Ruhe G, Carlson C, Chew F (2017) Predicting the vector impact of change-an industrial case study at brightsquid. In: Empirical software engineering and measurement (ESEM), 2017 ACM/IEEE international symposium on. IEEE, pp 131–140
Kim S, Ernst MD (2007) Which warnings should i fix first? In: Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering. ACM, pp 45–54
Kitchenham BA, Pfleeger SL (2008) Personal opinion surveys. In: Guide to advanced empirical software engineering. Springer, pp 63–92
Lemberger PP, Morel M (2013) Managing complexity of information systems: the value of simplicity. Wiley
Liaw MW et al (2002) Classification and regression by randomforest. R news 2(3):18–22
Lin C-Y (2004) Rouge: a package for automatic evaluation of summaries. Text Summarization Branches Out
Ling CX, Sheng S, Bruckhaus T, Madhavji NH (2005) Predicting software escalations with maximum roi. In: Data Mining, Fifth IEEE International Conference on. IEEE, p 4
Maalej W, Nayebi M, Johann T, Ruhe G (2016) Toward data-driven requirements engi- neering. Software, IEEE 33(1):48–54
Malhotra R (2015) A systematic review of machine learning techniques for software fault pre- diction. Appl Soft Comput 27:504–518
Mani S, Catherine R, Sinha VS, Dubey A (2012) Ausum: approach for unsupervised bug report summarization. In: Proceedings of the ACM SIGSOFT 20th international symposium on the foundations of software engineering. ACM, p 11
Manning C, Surdeanu M, Bauer J, Finkel J, Bethard S, McClosky D (2014) The Stanford corenlp natural language processing toolkit. In: Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations, pp 55–60
Martin W, Sarro F, Jia Y, Zhang Y, Harman M (2017) A survey of app store analysis for software engineering. IEEE Trans Softw Eng 43(9):817–847
Menzies T, Bird C, Zimmermann T, Schulte W, Kocaganeli E (2011) The inductive software engineering manifesto: principles for industrial data mining. In: Proceedings of the international workshop on machine learning Technologies in Software Engineering. ACM, pp 19–26
Mihalcea R, Tarau P (2004) Textrank: bringing order into text. In: Proceedings of the 2004 conference on empirical methods in natural language processing
Mohit B (2014) Named entity recognition. In: Natural language processing of semitic languages. Springer, pp 221–245
Montgomery L, Damian D (2017) What do support analysts know about their customers? On the study and prediction of support ticket escalations in large software organizations. In: Requirements engineering conference (RE), 2017 IEEE 25th international. IEEE, pp 362–371
Montgomery L, Reading E, Damian D (2017) Ecrits—visualizing support ticket escalation risk. In: Requirements engineering conference (RE), 2017 IEEE 25th international. IEEE, pp 452–455
Moreno L, Bavota G, Di Penta M, Oliveto R, Marcus A, Canfora G (2014) Automatic generation of release notes. In: Proceedings of the 22nd ACM SIGSOFT international symposium on foundations of software engineering. ACM, pp 484–495
Murray, Carenini G (2008) Summarizing spoken and written conversations. In: Proceedings of the conference on empirical methods in natural language processing. Association for Computational Linguistics, pp 773–782
Nayebi M, Ruhe G (2014) Analytical open innovation for value-optimized service portfolio planning. In: International conference of software business. Springer, pp 273–288
Nayebi M, Ruhe G, Mota RC, Mufti M (2015) Analytics for software project management–where are we and where do we go? In: Automated Software Engineering Workshop (ASEW), 2015 30th IEEE/ACM International Conference on. IEEE, pp 18–21
Nayebi M, Marbouti M, Quapp R, Maurer F, Ruhe G (2017) Crowdsourced exploration of mobile app features: a case study of the fort mcmurray wildfire. In: Proceedings of the 39th international conference on software engineering: software engineering in society track. IEEE Press, pp 57–66

Nayebi M, Kabeer S, Ruhe G, Carlson C, Chew F (2018) Hybrid labels are the new measure!. IEEE Software, 35(1):54–57.
Nazar N, Hu Y, Jiang H (2016) Summarizing software artifacts: a literature review. J Comput Sci Technol 31(5):883–909
Peng FL, Ding C (2005) Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. IEEE Trans Pattern Anal Mach Intell 27(8):1226–1238
Ramos et al (2003) Using tf-idf to determine word relevance in document queries. In: Proceed- ings of the first instructional conference on machine learning, vol 242, pp 133–142
Rastkar S, Murphy GC, Murray G (2010) Summarizing software artifacts: a case study of bug reports. In: Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering - Volume 1, ICSE '10. ACM, New York, pp 505–514
Rastkar S, Murphy GC, Murray G (2014) Automatic summarization of bug reports. IEEE Trans Softw Eng 40(4):366–380
Rish et al (2001) An empirical study of the naive bayes classifier. In: IJCAI 2001 workshop on empirical methods in artificial intelligence, vol 3. IBM, New York, pp 41–46
Robillard MP, Maalej W, Walker RJ, Zimmermann T (2014) Recommendation systems in software engineering. Springer Science & Business
Schütze H (1998) Automatic word sense discrimination. Computational linguistics 24(1):97–123
Sheng VS, Gu B, Fang W, Wu J (2014) Cost-sensitive learning for defect escalation. Knowl-Based Syst 66:146–155
Singhal S, Bhattacharya A (2019) Abstractive text summarization. home.iitk.ac.in/~soumye/cs498a/report.pdf
Sorbo D, Panichella S, Alexandru CV, Shimagaki J, Visaggio CA, Canfora G, Gall HC (2016) What would users change in my app? Summarizing app reviews for recom- mending software changes. In: Proceedings of the 2016 24th ACM SIGSOFT international symposium on foundations of software engineering. ACM, pp 499–510
Steinberger, Jezek K (2004) Using latent semantic analysis in text summarization and summary evaluation. In: Proc. ISIM, vol 4, pp 93–100
Vanderwende HS, Brockett C, Nenkova A (2007) Beyond sumbasic: task-focused summarization with sentence simplification and lexical expansion. Inf Process Manag 43(6):1606–1618
Wallach HM (2006) Topic modeling: beyond bag-of-words. In: Proceedings of the 23rd inter- national conference on machine learning. ACM, pp 977–984
Williams G, Mahmoud A (2017) Mining twitter feeds for software user requirements. In: Requirements engineering conference (RE), 2017 IEEE 25th International. IEEE, pp 1–10
Wold S, Esbensen K, Geladi P (1987) Principal component analysis. Chemom Intell Lab Syst 2(1–3):37–52
Wolpert DH (1992) Stacked generalization. Neural Netw 5(2):241–259
Xia X, Lo D, Ding Y, Al-Kofahi JM, Nguyen TN, Wang X (2017) Improving automated bug triaging with specialized topic model. IEEE Trans Softw Eng 43(3):272–297
Young T, Hazarika D, Poria S, Cambria E (2018) Recent trends in deep learning based natural language processing. IEEE Comput Intell Mag 13(3):55–75
Yu, Liu H (2004) Efficient feature selection via analysis of relevance and redundancy. J Mach Learn Res 5(Oct): 1205–1224

Maleknaz Nayebi , is an Assistant Professor at Ecole Polyetchnique of Montreal. She received her PhD degree from at the Software Engineering Decision Support lab from The University of Calgary in Canada. The PhD was on Analytical Release Management for Mobile Apps. She has six years of professional software engineering experience. Her main research interests are in mining software repositories, release engineering, open innovation and empirical software engineering. Maleknaz co-chaired RE data track 2018, IWSPM 2018, IASESE 2018 advanced school, and OISE 2015. Maleknaz is a member of the IEEE and ACM. Her homepage is http://maleknazn.com.

Liam Dicke , is a 4th year undergraduate student at the university of Alberta studying Engineering Physics. He did an internship in SEDS lab at the university of Calgary. He is currently performing research in geostatistics under Dr. Jeffery Boisvert, and is a member of the University of Alberta Aerial Robotics Group.

Ron Ittyipe , is a 4th year Electrical and Computer Engineering Student at the University of Calgary. He worked at the Software Engineering Decision Support (SEDS) laboratory at the University of Calgary in 2017 and is currently a software engineering intern at Garmin Canada.

Chris Carlson , brings over 20 years of project and product management to his role developing and implementing strategy. Serving in many senior roles, Chris has proven expertise in business model development and operational optimization. From 2009 to 2014, Chris managed patient data migration and clinical change management while implementing cloud-based Electronic Medical Record solutions across Alberta.

Guenther Ruhe , is the Industrial Research Chair in Software Engineering at the University of Calgary. His research focuses on product release planning, software project management, decision support, data analytics, empirical software engineering, and search-based software engineering. He is the editor in chief of Information and Software Technology and was the General Chair of the Requirements Engineering conference RE'18. He is a senior member of IEEE and a member of ACM.

Affiliations

Maleknaz Nayebi[1] · Liam Dicke[2] · Ron Ittyipe[3] · Chris Carlson[4] · Guenther Ruhe[3]

Liam Dicke
dicke@ualberta.ca

Ron Ittyipe
ron.ittyipe@ucalgary.ca

Chris Carlson
chris.carlson@Brightsquid.com

Guenther Ruhe
ruhe@ucalgary.ca

[1] Ecole Polytechnique of Montreal, Montreal, Canada

[2] University of Alberta, Edmonton, Canada

[3] University of Calgary - SEDS lab, Calgary, Canada

[4] Brightsquid Inc., Calgary, Canada