An Intelligent Framework for Issue Ticketing System based on Machine Learning

Ruanda Qamili
Institute of Information Systems
HES-SO Valais-Wallis
Sierre, Switzerland
ruanda.qamili@gmail.com

Shaban Shabani
Department of Mathematics and Computer Science
University of Basel
Basel, Switzerland
shaban.shabani@unibas.ch

Johannes Schneider
Institute of Information Systems
University of Liechtenstein
Vaduz, Liechtenstein
johannes.schneider@uni.li

***Abstract*—Due to the rapid shift of companies towards superb customer experience and satisfaction, ticketing systems have come into a prominence and represent a strategic element in business competitiveness. Different software companies have developed very effective software tools for issue tracking, nevertheless, some sub-processes and tasks within the ticketing systems are still performed manually. These manually performed tasks represent bottlenecks, especially at large organizations they result in declined productivity and increased response time. Advancements in machine learning can be used in a novel way in which they are combined with the traditional issue tracking and ticketing systems on the market, to enable optimal operational efficiency in the Customer Service and Support(CSS) Department of large-scale businesses that deal with customer reports. This paper proposes an integrated approach to customer support by treating three seemingly different bottlenecks in the ticketing system: spam detection, ticket assignment and sentiment analysis. We use primary data to implement and apply the proposed machine learning approach. The evaluation shows promising results in terms of accuracy and efficiency of our approach.**

***Keywords*-ticketing system; spam detection; ticket assignment; sentiment analysis.**

I. Introduction

Due to the competitive environment, the Customer Support and Service (CSS) Department of any organization must meet and exceed customers expectations [1]. Consequently, the CSS department is undergoing significant changes to attain effective business processes. Hence, it is very important for companies to identify slow running processes, perform a root cause analysis and then improve their performance. However, in practice the issue resolution processes are ineffective and error prone. Very often as customers we come across the pathetic scenes where we must wait long times until getting a response or resolution for a reported faulty product. The main reason behind delays in support service is the manual work. Aiming to reduce the human efforts and process errors, enterprises focus on embedding *intelligent business processes* and *business process automation* [2]. Analytics and data mining techniques are integrated into business processes to make data-driven decisions and achieve optimal outcomes [3].

The main objective of this paper is to identify and improve bottlenecks in the *ticketing system*. This can be achieved through the application of simple artificial intelligent components that are very feasible to implement at a low cost.

Furthermore, this work aims to develop a system which combines the three main automation components that function on the basis of same implementation logic. The outcome of this would be an intelligent ticketing system which in the best scenario could be embedded by organizations as a whole. Nevertheless, each of these components can be individually and independently embedded by organizations to solve each of the problems separately.

Although, machine learning algorithms have been applied to a wide range of document classification tasks, its application to automatic spam detection scenario within a ticketing system remains an open battleground.

Our core contribution is the application of a novel approach, a conservative unanimity algorithm that as an aggregation strategy combines the output of several spam filtering classifiers. This strategy achieves the lowest false positive rates. As an overall system, tackling the spam filtering, ticket assignment and sentiment analysis as a package, represents a novelty as no other research paper has considered these three aspects together. Our contribution in terms of software, is that we propose and implement the three components as an application that could be easily implemented and integrated to the existing ticketing systems on market.

The remainder of the paper is structured as follows: Section 2 describes in details the three issues we address; Section 3 summarizes the related work and Section 4 demonstrates the proposed system design; Section 5 presents details on the steps and tools used for the implementation; Section 6 describes the results from the evaluation and Section 7 concludes and gives details about future work.

II. Problem Statement

We conducted interviews with employees from two different companies to identify the main causes that lead to delayed customer support. Both groups identified the long time interval between ticket creation and first action as the main factor to cause delayed issue resolution. In addition, we conduct a deep analysis on the reasons that cause the delayed triggering of the resolution process for these two companies. These causes are considered as the most important drawbacks with a high potential for improvement in issue tracking systems. The main concerns we have identified are: manual ticket assignment (ticket triage), spam (unsolicited or unwanted) emails and sentiment analysis. Each of these three concerncs is responsible for manual work causing delays as well as costs. We treat them in more detail in this work and resolve them through the practical applications of machine learning techniques to improve business outcome.

A. *Spam detection in tickets*

The ever growing problem of unwanted emails has turned into a significant problem for companies as well as for individuals. To investigate the problem of unsolicited emails in the context of ticketing systems a primary dataset from a software company located in Liechtenstein has been used and evaluated. The company has created their own ticketing system. They are suffering from spam emails. Spam emails arrive at the support email designated for receiving clients reports and concerns. The incoming emails at this address are automatically forwarded to the ticketing system in order to automatically generate tickets and incidents. Besides receiving legitimate emails, various spam emails are addressed as well, accordingly these spam emails generate *spam tickets* by default. The process of identifying, reviewing, manually labeling and canceling each of the spam incidents can be translated into wasted working hours of the support team of the company.

During the process of inspecting the dataset, useful information about another category of emails was revealed. The software developing company does not only face the problem of spam emails, but the problem of receiving unwanted emails has been recognized as well. Repeatedly, unimportant automated notification emails are sent from email addresses of different client servers, and these unwanted emails generate *unwanted tickets*. In terms of time consumption, the unwanted emails have the same effect as spam emails. Based on a study by Cranor [4] around 10% of the corporate emails are unwanted. Calculating the unwanted emails over the total number of emails, we obtained a significantly higher percentage of 21,97% of spam and unwanted emails was obtained. In what follows, we consider the unwanted emails as spam as well. Hence, the main outcome caused by the unwanted emails in this case declines employees' productivity, therefore an effective filtering model is a significant contribution to the effectiveness of the organization. To tackle this problem we apply supervised machine learning. Our focus is in finding a mechanism for filtering unwanted emails with high precision and maintaining low rates of false positives. Furthermore, a majority voting system approach has been applied that considers the decision of the majority of classifiers prior to deciding whether an email is a unwanted or not.

B. *Automatic ticket assignment*

Addressing and assigning a ticket to the appropriate support technician or to an expert is of critical importance in terms of speeding up the overall process of providing customer support. In many organizations, the analyzing, processing and assignment of support tickets is manually performed by service desk (support) teams [5]. Furthermore, the person who assigns the tickets must have very good knowledge of activities of each support technician, developer or expert in the project, in order to be able to correctly assign the tickets.
An automated ticket assignment would be the ideal scenario, more effective in terms of time and money. To assign tickets automatically, in this work a supervised machine learning approach has been applied. It predicts the assignee for new tickets using a classifier that was trained with past tickets.

C. *Sentiment Analysis*

Traditional ticketing systems embed surveys as built-in functionalities to gauge customers satisfaction. Once a ticket is resolved, the system asks customers to evaluate the service by completing a short customer satisfaction survey which consists of a few questions related to the service delivery. Although this method has been employed by many organizations, it is not widely accepted by the customers, as completing a survey is time-consuming. As a result, many customers avoid them.

When writing a ticket (reporting a fault, requiring information, asking for support, etc.) a vast amount of content is produced. An intelligent approach is needed to detect customers attitudes, opinions and emotions expressed in tickets. The number of satisfied customers can be translated into expected financial gains. The sentiments found in tickets' text represents the voice of customers and stands as an indicator of product acceptance. In this respect, the sentiment analysis task and its results, are especially important when taking strategic and data-driven decisions. Additionally, the sentiment analysis in tickets can be used as a trigger to alert when customer reports contain high levels of negative polarization.

Uncovering this data is of crucial importance. It can be achieved by employing sentiment analysis through the application of a supervised machine learning approach, where the output is one of the three classes: positive, negative or neutral.

III. Related Work

This section provides an insight into the three different areas of related work, namely: spam detection, automatic ticket assignment and sentiment analysis through the application of machine learning techniques.

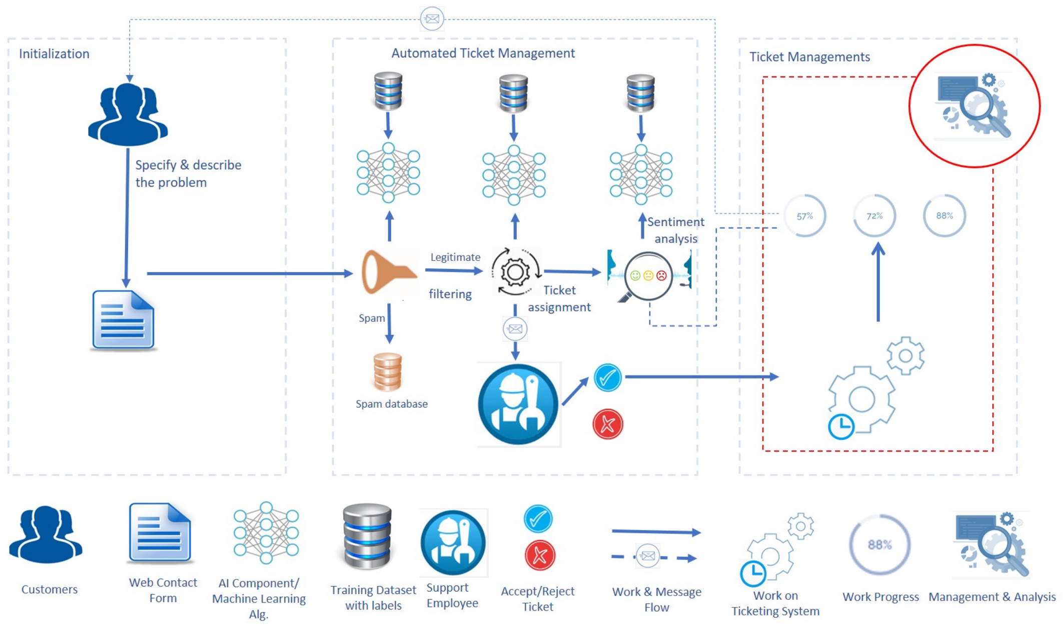

Figure 1: System architecture of the intelligent ticketing system

A. *Spam detection*

The earliest spam is an advertising message in 1978 which was sent by Gary Thuerk[1]. This phenomenon was later termed as ”spam”. The ever-growing volume of spam turned into a significant problem for companies and individuals. Thus, the trend was also reflected in research, as various research studies during the past years have proposed several solutions to overcome the spam problem [6], [7]. It is important to mention that in research works spam detection has been widely considered as a text classification task, and most of the techniques applied to avoid spam are content based. Our approach in this paper is content based as well. Among all the different ways of implementing Naïve Bayes classification, the approach by Graham[2] has become fairly famous, introducing a new formula for calculating token values and overall probabilities of an email being classified as spam. However, there is an unrealistic assumption in his formula which assumes the number of spam and ham documents are equal in the dataset for everyone. As a result, an improvement[3] correctly adjusts the formula later to make it fit for all datasets. Guzella et al. [6] give an overview on the different methods based on machine learning for filtering spam emails. SVM [8] as an important supervised learning algorithm for classification has its roots in statistical learning and has shown good empirical results in many practical applications, from handwritten digit recognition to text classification.

[1]http://www.templetons.com/brad/spamreact.html
[2]http://www.paulgraham.com/spam.html
[3]https://mail.python.org/pipermail/python-dev/2002-August/028216.html

B. *Automatic ticket assignment*

The automatic assignment of tickets has been treated in software development thus this literature review is based on automatic bug triage/assignment. During the past years, there have been many studies and attempts to automate the bug triage [9]. It is difficult to decide which of these methods are most accurately performing, due to the differences in the content and setup of datasets, and the quality of data.

Nevertheless, all studies agree that none of these models are 100% accurate, and they need to be continuously maintained. Cubranic [10] was the first to apply machine learning techniques in order to enable bug triage by using text classification approach. The description of the bug reports was used to train a Naïve Bayes classifier to correctly predict the developer to whom the bug ticket should be assigned. This method was applied to a dataset collected from the reports for Eclipse software, and attained an accuracy of 30%. Later on, Anvik et al. [11] studied the bug assignment through the application of text classification, expanding the work which was proposed by [10]. Their work focuses on having better prepared data and testing various algorithms to determine an algorithm which performs better. They applied the Support Vector Machine (SVM) algorithm in the bug assignment approach on datasets about Eclipse, Firefox and GCC. Results of this work report an accuracy of 64% for the Firefox (a dataset with 9,752 records) and 58% for the Eclipse (a dataset of 8,655 records). Ahsan et al. [12] applied the SVM classifier on a set of data that contained 1,983 resolved bug reports along with the developer activity data from the Mozilla open source project. The best obtained bug triage system was based on Latent Semantic Indexing

(LSI) and SVM achieving 44.4% classification accuracy. The average precision and recall values reported were 30% and 28%, respectively. The empirical study of [13] improved the already existing reports with additional historical bug re-assignment information scoring a precision of up to 68%, attained when suggesting three developers. Matter et al. [14] study the "tossed" bug reports of 37%-44% based on the Mozilla Data. They introduce a graph model based on Markov Chains, which captures bug tossing history, with the goal to reveal developer networks which can be used to discover team structures and to find suitable experts for a new task, and to better assign developers to bug reports. This experiment used data consisting of 445,000 bug reports and results report reduced tossing events by up to 72%. In addition, the model increased the prediction accuracy by up to 23 percentage points compared to traditional bug triaging approaches presented so far. Work done by Hu et al. [15] propose BugFixer as a method for computing bug report similarity by utilizing historical bug fix data and constructing network structures to model the relationships between developer, source code components and the bugs.

C. Sentiment Analysis

One of the main tasks in sentiment analysis is detection of the polarity of documents, reviews, emails and tickets [16]. Moreover, sentiment analysis is the computational detection of opinions, feelings and subjectivity of texts. Unlike spam detection and tickets triage, sentiment analysis is a relatively newer and ongoing research area. Before the year of 2000, very little research was done on sentiment analysis and opinion mining while the outbreak of sentiment analysis occurred after 2004 [17]. Since then, a considerable amount of research has been done, aiming to find novel ways to automate the process of assigning sentiments to the documents. The sentiment analysis problem has been applied to different topics[18], initially for analyzing online product reviews [19] and shifting the focus mainly towards analyzing text on social media, covering topics such as: election analysis [20], stock market prediction [21] and medicine [22]. In addition, there have been different companies that provide online opinion mining services.

IV. System Design

In this section, we briefly present the implementation of the three different components of the proposed solution and the content that has been integrated in our prototype. Figure 1 illustrates the overall system architecture. It represents the conceptual model which defines the structure and behavior of the Intelligent Ticketing System environment. It essentially defines a set of functional requirements and how the proposed system should be built. The system is divided into five main phases:

- Initialization - customers filling the online web-form.
- Automated filtering of emails as legitimate or spam
- Automated assignment - trained model that classifies the ticket through supervised text classifier
- Ticket management using existing ticketing tool
- Automatic detection of the polarity of tickets

Figure 1 illustrates the system architecture and the overall process. The process starts with a customer who has a faulty product and aims to report it by filling the online web-form or sending a report email. Once the form or email is stored in the CSS database, supervised machine learning models analyze the text of the email and classify it as legitimate or spam. If the email is classified as spam, it is stored in the database of spam emails and the process finishes here. If not, the second step is to classify the ticket to which department or technician it should be assigned.

Once the ticket has been assigned, the ticket can be created using the API of any support ticketing tool available on market. In this work, we suggest the usage of Jira[4] or Zendesk[5], two issue tracking systems which have quite good community support, rich and well documented API. The support technicians will receive notifications that a new ticket has been assigned to them.

Furthermore, based on the descriptive text of the ticket, the system automatically evaluates the opinions, emotions and satisfaction of customers, without asking the customer to fill a satisfaction survey.

V. Implementation

In this section, we present the implementation part, describing the dataset, tools and machine learning methods that are used in our system.

A. Dataset

We use primary-data to evaluate the performance of the algorithms. The dataset is extracted from the ticketing system of a software developing company. It consists of 18,917 records which are emails generated from customer reports from which the tickets are automatically generated. We initially use language detection tool *langdetect*[6] to analyze the language used in the tickets. This tool detected German as mostly used language on about 72% of the tickets, and English on 28% of the tickets. The corpus of data we use in our experiment covers the time span of 09.03.2009 - 28.03.2017. The dataset comprises 14 columns, among which we identified 3 columns as relevant for the classification problems:

- *created_date* - date when the ticket was created
- *subject* - the header of the email
- *description* - the content of the email

Table I shows basic statistics about the dataset and the text available. We can observe that there is sufficient text data in order to run the algorithms for document classification.

[4]https://www.atlassian.com/software/jira
[5]https://www.zendesk.com/
[6]https://pypi.python.org/pypi/langdetect

	#
Number of columns	14
Total tickets	18,917
Total words in *subject*	95,015
Total words in *description*	1,753,667
Average words per *subject*	5,16
Average words per *description*	95,3

Table I: Dataset overall statistics

B. Tools

For our supervised learning classification problem, we make use of *Python scikit-learn* [23], a simple and efficient data mining and analysis tool. It implements all the necessary tools and machine learning techniques for feature extraction, classification and results analysis. Furthermore, we use Python's Pandas and Numpy libraries [24] for data preparation and cleaning.

C. Data Preprocessing, Feature Extraction and Training

Since the only available information from the ticket is text, we initially apply text cleaning methods, e.g. to remove punctuation and for *stop_word* removal (both "english" and "german"). We employ a bag-of-words approach meaning that individual words correspond to features, which are obtained using a *word_tokenizer*. For weighing of features, i.e. words, we use the term-frequency, inverse document frequency *tf-idf* statistic. Using the extracted features for each ticket we train a set of classifiers for our tasks.

VI. Evaluation

We use the pre-processed dataset described above for training and evaluating the performance of a variety of classifiers based on machine learning, i.e. we split the dataset into training and test sets. We follow the state-of-the-art recommendations of splitting the dataset into 90% of the data as training set, and 10% of the data for the testing phase.

In the following we will present the results from the three use cases: spam detection, ticket assignment and sentiment analysis. Due to the structure of the data and considering the labels available, for both scenarios, we explain the assumptions we have made and the process of the evaluation.

A. Spam Detection

Considering the spam detection problem, the dataset is not balanced in terms of the two classes: spam and ham. Only 3,880 from the tickets are labeled as spam, hence in our scenario we use a balanced dataset adding 3,880 ham tickets, summing up to 7,760 tickets. Next, we split the data into 90% training set and 10% testing set consisting of 6,984 and 776 tickets respectively. In the training phase, we run 10-fold cross-validation, where the accuracy is computed by taking the mean of the 10 runs, and additionally we calculate the standard deviation. We chose several algorithms provided by scikit-learn that are recommended when dealing with supervised learning and document classification problems:

- Support Vector Machines (SVM)
- Random Forest (RF)
- Decision Trees (DT)
- Naïve Bayes (NB)
- Stochastic Gradient Descent (SGD)
- Logistic Regression (Log)

The results from the cross-validation phase are listed in Table II.

	Training set (3492 spam + 3492 ham)	
	mean accuracy	standard deviation
Logistic Regression	97.88	2.02
SVM	99.30	0.80
Naïve Bayes	95.65	4.04
Decision Trees	93.62	19.14
SGD	99.40	0.60
Random Forest	99.85	0.20

Table II: Performance of the algorithms in the cross-validation phase for the spam detection scenario

A box plot visualizing the performance of the algorithms is shown in Figure 2.

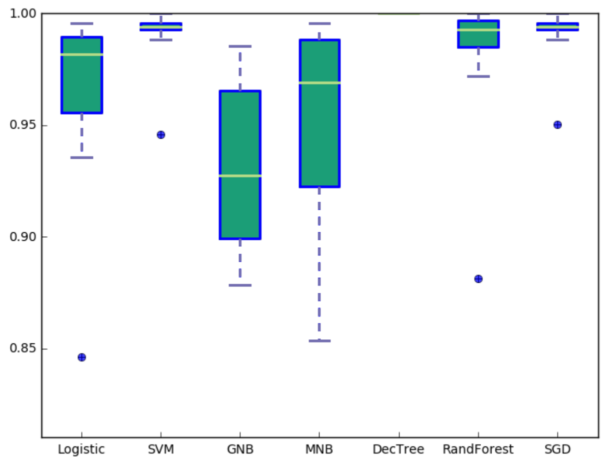

Figure 2: Boxplot representing the performance of the classifiers: mean accuracy and standard deviation during the cross-validation phase of the spam detection

Our results show that the SVM algorithm reaches a significantly higher classification accuracy than the Naïve Bayes approach. The best algorithms are those having higher mean accuracy and lower standard deviation of the accuracy. All algorithms perform well except Decision Trees, which have low mean accuracy. The top performers are Random Forest, SVM, and SGD.

On the evaluation or, equivlaently, testing phase, 10% of the tickets have been used, i.e. 776 tickets (388 ham + 388 spam). For each of the 776 tickets, we extract features from the text available on the subject and the description of the ticket and get the predicted class from each algorithm. For measuring the performance of the trained classification algorithms on the testing phase, we use well-known reporting scores:

- Precision - is the fraction of the relevant documents among the retrieved documents

$$precision = \frac{tp}{tp + fp} \tag{1}$$

- Recall - is the fraction of relevant documents retrieved over the total relevant documents

$$recall = \frac{tp}{tp + fn} \tag{2}$$

- F1-score - is the harmonic mean of precision and recall

$$F = 2 * \frac{precision * recall}{precision + recall} \tag{3}$$

Initially, we run the testing over the 5 chosen algorithms that were used for training. The Naïve Bayes model achieves the highest precision of 0.97, followed by Logistic Regression. The results of the 5 algorithms are shown in Table III.

algorithm	precision	recall	f1-score
Logistic	0.93	0.92	0.92
SVM	0.89	0.86	0.86
Naïve Bayes	**0.97**	**0.97**	**0.97**
SGD	0.89	0.86	0.86
Random Forest	0.81	0.70	0.67

Table III: Performance of the algorithms individually in the testing phase for the spam detection

Aggregation strategies

Due to the sensitivity to errors in spam detection in real systems, e.g. deciding that a ticket is spam but it is not, we carefully consider the *false positives* fraction. This fraction states the percentage of legitimate tickets that are wrongly classified as spam. Ignoring or delaying a ticket of a customer that, e.g. suffer from system outages, is a major mistake. It could be translated to unsatisfied customers and financial damages for the organization. Therefore, we inspect the precision of the individual algorithms on both classes unwanted and legitimate. Analyzing the precision of algorithms when classifying a ticket as *legitimate*, we could observe that Naïve Bayes has a precision of 95%.

To address the false positives issue, we consider an ensemble approach of combining multiple models' decisions as the final output. We employ the majority voting (MV) strategy, which selects the alternative that obtains most votes. An example of the voting strategy is shown in Table IV.

RF Classifier	Log	SVM	NB	SGD
spam	spam	ham	spam	ham

Table IV: Example of majority voting in the spam detection

For the MV strategy, we test several combinations and the best performance is achieved when considering the majority vote of the top 3 performers: SVM + Naïve Bayes + Logistic Regression. Furthermore, we consider weighted majority voting strategy (WMV), which applies weights for each algorithm based on their performance on the cross-validation phases. Figure 3 illustrates the idea behind this strategy, and Table V presents the classification results. However, both strategies do not improve the results compared to the best performer algorithm alone.

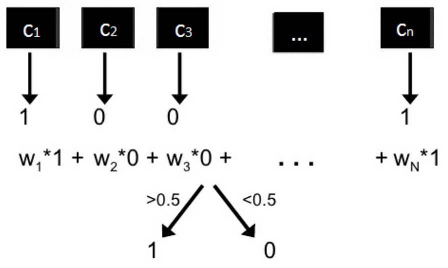

Figure 3: A visual representation of Weighted Majority Voting (WMV) strategy

algorithm	precision	recall	f1-score
MV (top 5)	0.91	0.89	0.88
MV (SVM + NB + Log)	0.95	0.94	0.94
WMV (SVM + NB + Log)	**0.95**	**0.94**	**0.94**

Table V: Results of voting strategies on spam detection

To further lower the percentage of false positives, as a next strategy we apply *unanimity with confidence threshold*, which considers the probability of prediction of each algorithm [25]. The scikit-learn toolkit provides the confidence for every classified task. Using this parameter, we apply a threshold on the probability estimation on the confidence of the algorithms in the unanimity strategy. We apply this only in the case when the ticket is classified as spam. The threshold is applied on the mean probabilities predicted by the 5 algorithms. For instance, if the mean probability of all algorithms classifying a ticket as spam is higher than 0.90 then we consider the ticket spam, otherwise it is classified as a ham ticket. This is a very strict strategy that decreases the percentage of false positives to less than 1%, but on the other hand it leads to many spam tickets being classified as ham, i.e. about 18% of tickets are being classified as ham. However, comparing the risk aspect, the later case would lead to more time consumption due to filtering manually these tickets by a support team member. Increasing the confidence threshold to 0.95 achieves precision of 100%, meaning 0 false positives (no ham ticket could be classified as spam). However, that allows almost 33% of the spam tickets to be classified as ham, which means more spam tickets are received in the ticketing system.

B. *Ticket assignment*

In our scenario, the automatic ticket assignment is a 3-class classification problem where the 3 classes represent support teams in the company that manage and solve the issue tickets. We follow the same process as in the spam detection use case in terms of data pre-processing, feature extraction and 10-fold cross-validation. Table VI presents the results on the cross-validation set.We can observe that Random Forest and SVM achieve the highest accuracy of 86%, followed by Logistic Regression.

	Cross-validation set	
	mean accuracy	standard deviation
Logistic	85.30	0.71
SVM	86.00	0.93
Naïve Bayes	83.46	1.05
SGD	78.83	0.97
Random Forest	86.1	0.89

Table VI: Ticket assignment - results on the cross-validation

On the testing phase, similar as in the spam detection part, we extract the text and description columns of each ticket and run the prediction process. We analyze the precision, recall and f1-score performance measurement parameters. Table VII lists the results of the testing phase. SVM is the top performer with f1-score of 0.75.

algorithm	precision	recall	f1-score
Logistic	0.75	0.72	0.73
SVM	**0.77**	**0.76**	**0.75**
Naïve Bayes	0.74	0.72	0.72
SGD	0.75	0.73	0.73
Random Forest	0.72	0.68	0.66

Table VII: Results on testing phase for the ticket assignment

C. *Sentiment analysis*

Sentiment analysis allows to automatically analyze the customer satisfaction expressed in the tickets. Commonly, three levels of sentimetns are distinguished: positive, neutral and negative. In our case, because the dataset was not manually annotated, we could not run the same machine learning approach as in the spam detection and ticket assignment. Obtaining labels for such a big dataset is a costly and time consuming process.

We use existing tools to analyze the sentiment considering solely the text available in the title and the detailed description of the tickets. We use sentiment analysis packages (*pattern.en* [26]) that focus on the adjectives used in the text to provide a score for the sentiment. Results on this analysis are depicted in Figure 4. This tool classifies 43.6% of the German written tickets as positive, 36.9% as neutral and 19.5% as negative. On the tickets which are written in English language, the ratio of neutral tickets is the highest with 44.1%, whereas the percentage of positive and negative tickets is 31.2% and 24.7% respectively.

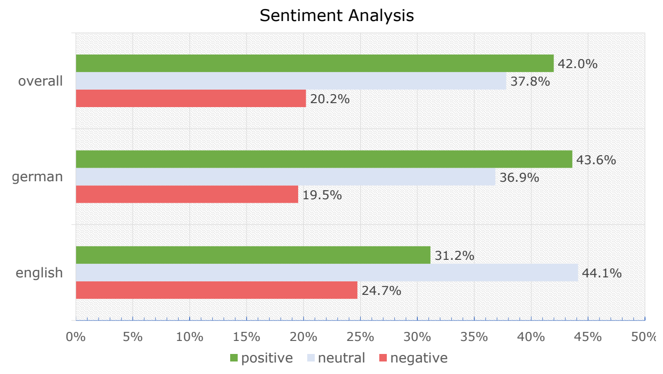

Figure 4: Sentiment analysis on the "subject + description"

VII. Conclusion and Future Work

The overall objective of this study was to design an efficient system toward providing a fully automated ticketing system addressing issues in 3 different dimensions: spam filtering, automatic ticket assignment and sentiment analysis. We implemented a prototype application for a real scenario with ticketing system data provided from a software developing company, with main focus on detecting spam tickets, and as second step, automatic assignment of genuine tickets to the corresponding department within the company.

On the automatic spam detection, we apply the hybrid approach of combining multiple models to address the false positives issue. Our unanimity with confidence threshold hybrid model strictly filtered the false positives but in exchange of allowing more spam tickets to be classified as legitimate tickets.

Due to missing ground truth annotation of the dataset for the sentiment analysis part, we were not able to train our own classifier, tailored to our dataset, hence this part remains the first major future work, especially due to its importance for the company, as it will enable to automatically find and prioritize tickets with unsatisfactory content (negative sentiment). As a final step, we would like to test the entire cycle with all the three components. Our long term goal is to integrate our system with the existing ticketing system to manage tickets in an automated manner. We might also try to conduct further work using unsupervised machine learning. For instance, we intend to leverage clustering techniques such as centroid or density-based methods [27] as well as topic models such as LDA or TKM [28] to further refine ticket categorizations over time and to perform a more fine granular sentiment analysis stating sentiments per topic or ticket (sub)category.

References

[1] K. Blackwell, *Achieving high customer satisfaction*. "Course Technology", 2015.

[2] A.-W. Scheer, F. Abolhassan, W. Jost, and M. Kirchmer, *Business process automation*. Springer, 2004.

[3] C. Rygielski, J.-C. Wang, and D. C. Yen, "Data mining techniques for customer relationship management," *Technology in society*, vol. 24, no. 4, pp. 483–502, 2002.

[4] L. F. Cranor and B. A. LaMacchia, "Spam!" *Communications of the ACM*, vol. 41, no. 8, pp. 74–83, Aug. 1998. [Online]. Available: http://doi.acm.org/10.1145/280324.280336

[5] A. Fiaz, N. Devi, and S. Aarthi, "Bug tracking and reporting system," *arXiv preprint arXiv:1309.1232*, 2013.

[6] T. S. Guzella and W. M. Caminhas, "A review of machine learning approaches to spam filtering," *Expert Systems with Applications*, pp. 10 206–10 222, 2009.

[7] S. Nazirova, "Survey on spam filtering techniques," *Communications and Network*, vol. 20, no. 3, 2011. [Online]. Available: http://dx.doi.org/10.4236/cn.2011.33019

[8] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[9] V. Dedik and B. Rossi, "Automated bug triaging in an industrial context," in *2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Aug 2016, pp. 363–367.

[10] D. Cubranic, "Automatic bug triage using text categorization," in *In Proceedings of the Sixteenth International Conference on Software Engineering and Knowledge Engineering*, 2004, pp. 92–97.

[11] J. Anvik, L. Hiew, and G. C. Murphy, "Who should fix this bug?" in *Proceedings of the 28th International Conference on Software Engineering*, ser. ICSE '06, 2006, pp. 361–370.

[12] S. N. Ahsan, J. Ferzund, and F. Wotawa, "Automatic software bug triage system (bts) based on latent semantic indexing and support vector machine," in *Proceedings of the 2009 Fourth International Conference on Software Engineering Advances*, ser. ICSEA '09. IEEE Computer Society, 2009, pp. 216–221.

[13] G. Jeong, S. Kim, and T. Zimmermann, "Improving bug triage with bug tossing graphs," in *Proceedings of the the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*, ser. ESEC/FSE '09. ACM, 2009, pp. 111–120.

[14] D. Matter, A. Kuhn, and O. Nierstrasz, "Assigning bug reports using a vocabulary-based expertise model of developers," in *Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories*, ser. MSR '09, 2009, pp. 131–140.

[15] H. Hu, H. Zhang, J. Xuan, and W. Sun, "Effective bug triage based on historical bug-fix information," in *2014 IEEE 25th International Symposium on Software Reliability Engineering*, 2014, pp. 122–132.

[16] B. Pang and L. Lee, "Opinion mining and sentiment analysis," *Foundations and Trends in Information Retrieval*, vol. 2, no. 12, pp. 1–135, 2008. [Online]. Available: http://dx.doi.org/10.1561/1500000011

[17] M. V. Mäntylä, D. Graziotin, and M. Kuutila, "The evolution of sentiment analysisa review of research topics, venues, and top cited papers," *Computer Science Review*, pp. 16–32, 2018.

[18] K. Ravi and V. Ravi, "A survey on opinion mining and sentiment analysis: tasks, approaches and applications," *Knowledge-Based Systems*, vol. 89, pp. 14–46, 2015.

[19] J. Liu, Y. Cao, C. Y. Lin, Y. Huang, and M. Zhou, "Low-quality product review detection in opinion summarization," in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007, pp. 334–342.

[20] H. Wang, D. Can, A. Kazemzadeh, F. Bar, and S. Narayanan, "A system for real-time twitter sentiment analysis of 2012 us presidential election cycle," in *Proceedings of the ACL 2012 System Demonstrations*. Association for Computational Linguistics, 2012, pp. 115–120.

[21] J. Bollen, H. Mao, and X. Zeng, "Twitter mood predicts the stock market," *Journal of computational science*, pp. 1–8, 2011.

[22] K. Denecke and Y. Deng, "Sentiment analysis in medical settings: New opportunities and challenges," *Artificial intelligence in medicine*, pp. 17–27, 2015.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, pp. 2825–2830, 2011.

[24] W. McKinney, *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. " O'Reilly Media, Inc.", 2012.

[25] V. Vovk, "Universal well-calibrated algorithm for on-line classification," in *Learning Theory and Kernel Machines*, B. Schölkopf and M. K. Warmuth, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 358–372.

[26] T. D. Smedt and W. Daelemans, "Pattern for python," *Journal of Machine Learning Research*, vol. 13, no. Jun, pp. 2063–2067, 2012.

[27] J. Schneider and M. Vlachos, "Scalable density-based clustering with quality guarantees using random projections," *Data Mining and Knowledge Discovery*, vol. 31, no. 4, pp. 972–1005, 2017.

[28] J. Schneider and M. Vlachos, "Topic modeling based on keywords and context," in *Proceedings of the 2018 SIAM International Conference on Data Mining*. SIAM, 2018, pp. 369–377.