Text Categorization for Deriving the Application Quality in Enterprises using Ticketing Systems

Thomas Zinner[1], Florian Lemmerich[2], Susanna Schwarzmann[1], Matthias Hirth[1], Peter Karg[3], and Andreas Hotho[1,4]

[1] University of Würzburg, Institute of Computer Science, Würzburg, Germany
[2] GESIS - Leibniz Institute for the Social Sciences, Cologne, Germany
[3] kubus IT, München, Germany
[4] L3S Research Center, Hannover, Germany

Abstract. Today's enterprise services and business applications are often centralized in a small number of data centers. Employees located at branches and side offices access the computing infrastructure via the internet using thin client architectures. The task to provide a good application quality to the employers using a multitude of different applications and access networks has thus become complex. Enterprises have to be able to identify resource bottlenecks and applications with a poor performance quickly to take appropriate countermeasures and enable a good application quality for their employees. Ticketing systems within an enterprise use large databases for collecting complaints and problems of the users over a long period of time and thus are an interesting starting point to identify performance problems. However, manual categorization of tickets comes with a high workload.

In this paper, we analyze in a case study the applicability of supervised learning algorithms for the automatic identification of relevant tickets, i.e., tickets indicating problematic applications. In that regard, we evaluate different classification algorithms using 12,000 manually annotated tickets accumulated in July 2013 at the ticketing system of a nation-wide operating enterprise. In addition to traditional machine learning metrics, we also analyze the performance of the different classifiers on business-relevant metrics.

1 Introduction

Today's enterprise IT infrastructure is undergoing a significant change. Employees are working with an increasing number of different applications resulting in individual configurations of their personal computers. In order to reduce maintenance of the equipment, and to simplify the installation and management of diverse applications, companies move away from the traditional personal computer architecture, where each user has her own computer running all applications for daily work. Instead, companies try to manage many devices and applications centrally to keep end user devices as simple as possible. In that direction, thin client architectures provide an universal and basic terminal for user interaction while all applications requiring complex computations as well as data storage are carried out in the data center.

Although the overall maintenance of such an architecture is simplified compared to a purely decentralized architecture, some additional dependencies are introduced. Beside the thin client the user is interacting with, the transport network and the data center have a huge impact on the application performance and therewith on the productivity of the employees. Thus, it is of high importance for an enterprise to quickly identify slow-running applications and services, to perform a root-cause analysis, and then to improve the performance of these applications and services. A possible information source in an enterprise environment collecting complaints, problems and corresponding solutions are ticketing systems, i.e., central operational database systems used for the management of individual issues.

In this paper, we deduce the problematic applications and the application quality in an enterprise environment by using data from an internal ticketing system of a German company. Around 20.000 employees, working in $\approx$ 400 branch offices mostly located in small- to medium-sized cities, generate up to 1000 tickets per day. The proposed approach is challenging in the way, that the relevant tickets indicating performance problems have to be identified in a huge set of support tickets. A manual classification cannot be considered due to the huge amount of tickets submitted. For that reason, we present an automatic approach to identify relevant tickets based on machine learning. In that regard, we evaluate different classification algorithms using 12,000 manually annotated tickets accumulated in July 2013. From an enterprise perspective, the correct classification of a single ticket is not crucial as long as temporal trends and locally occurring performance problems are identified. Therefore, the classifier results are not only investigated by traditional machine learning metrics such as F_1-score, precision and recall, but also with respect to more business-relevant performance metrics. As an additional contribution of this paper, an anonymized version of the pre-processed dataset (as a tf-idf matrix) will be made publicly available with this paper[5].

The remainder of the paper is structured as follows: First, Section 2 reviews related work. Next, Section 3 introduces the utilized dataset and outlines the applied methodology, i.e., pre-processing procedures, classification algorithms and performance metrics. Afterwards, Section 4 discusses advantages and limitations of the applied approach. Finally, Section 5 concludes the paper with a summary and an outlook on future research directions.

2 Related Work

This section provides a brief overview of relevant related work. First, previous approaches to deducing business application quality are presented. After that, we shortly discuss text categorization with a special focus on ticket systems.

2.1 Identification of Application Quality via Tickets

User satisfaction with application quality – also called Quality-of-Experience (QoE) – can be defined as *the degree of delight or annoyance of the user of an*

[5] https://github.com/lsinfo3/dataset_ticketing_system_Dawak_2015

application or service. It results from the fulfillment of his or her expectations with respect to the utility and / or enjoyment of the application or service in the light of the user's personality and current state [15]. Hence, this metric is of purely subjective nature.

Typically, user satisfaction is evaluated by adjusting the application behavior while conducting a user survey. This allows to link objectively measurable parameters like the applications response time with user satisfaction. Examples for such evaluations are Staehle et al. [19] or Casas et al. [3]. Both works aim at a better understanding of the influence of varying technical parameters on the QoE. In both cases, the tests were conducted in dedicated labs with students and not in a working environment with employees. Additionally, such a methodology does not scale well in a complex IT system with a huge number of different applications.

To gain a global view of the user satisfaction in a business environment, existing user feedback can be used, e.g., from support tickets. In this direction, Mockus et al. [17] examined the satisfaction of a user group within the first three months after installation of a new software release. They evaluated service interactions like software defect reports and requests for assistance. In their study, they examined the impact of occurring problems and their frequencies. On the basis of their results, they detected predictors for customer perceived user quality. Chulani et al.[5] analyzed the relationship between development and service metrics (number of defects discovered, time to resolve a defect, severity of defects, etc.) and customer satisfaction survey metrics (overall customer satisfaction, top attributes customers look for in a product, etc.). With this knowledge, they improved the user satisfaction by handling specific reports earlier, i.e., reports that concern problems related to an attribute, of which users thought is important, but simultaneously is not satisfying.

2.2 Text categorization of support tickets

Text categorization with classification algorithms is a well explored standard task for text mining and text analytics, one of its main applications being *text filtering*, cf. [20]. While machine learning techniques for text filtering have been successfully applied in various domains, e.g., spam filtering [10] and filtering of unsuitable content [4], the use of machine learning techniques in the context of support ticket systems for enterprises is only little explored. In that direction, Wei et al. applied conditional random fields to automatically extract individual units of information from ticket texts [21]. Kreyss et al. [14] applied text mining categorization technology to analyze data from the IBM Enterprise Information Portal V8.1 containing more than 15,000 product problem records. They used a proven software quality category set to categorize these problem records into different areas of interest. Medem et al. [16] used the free text of network trouble tickets, for analyzing general trends in network incidents and maintenance activities. Diao et al. used an hybrid approach of automatically learned and manually edited rules in order to improve an automated system for classifying support tickets in failure classes [7]. Recently, Altintas and Tantuk proposed an extension of an issue tracking system that aims at automatically assigning tickets to the relevant

person or department [2]. By contrast, this work focuses on filtering support tickets that are of high relevance to the target application in order to derive information about the application quality in a thin client environment.

In this paper, we do not aim at contributing algorithmic advances to the field of text categorization, but apply standard techniques to our application domain, that is, the automatic classification of support tickets for enterprises. Therefore, we only outline the applied algorithms in Section 3.2, and refer to overview articles for more detailed explanations, see e.g., [20, 12].

3 Automatic Classification of Ticket Data

3.1 Dataset

We propose to utilize user feedback from support tickets to estimate the overall QoE of employees in a large scale business. Ticketing systems like OTRS[6] provide a central point for collecting and maintaining service and help requests in companies. Tickets cover issues ranging from simple password reset requests to notifications about severe system failures. In our case, we are interested in finding tickets indicating annoying system behavior, e.g., long application response times. Automatic approaches are required for filtering these tickets, since several hundreds to thousand tickets may be created daily.

The investigated setting origins from a German company with about 400 branch offices. For this company about 10,000 tickets are submitted consistently each month. In this work, we focus on a subset of this data consisting of around 12000 tickets, that is, all tickets submitted in July 2013. These tickets were manually categorized with a binary label. 303 tickets were labeled as positive examples, i.e., tickets reporting application quality issues and thus indicating a reduced user-perceived application quality. In the following, we use this labeled dataset as gold standard for our evaluations.

The following information is available for each ticket: Each ticket has an unique *ticket-id* that can be used for identification and a *generation date* indicating when the ticket was submitted. It further contains *details* about the reported issue. These allow the user to describe his issues in free text form. Additionally, also information about the affected *site* of the company was included.

3.2 Methodology

As common practice in text mining suggests, we utilize a *vector space approach*, cf. [18], with bag-of-words representations of the detailed ticket text. To transform the set of support tickets, we first applied standard tokenizing procedures as pre-processing. Additionally, stop words, i.e., very common words, were removed according to a dictionary. Furthermore, also tokens consisting of a single letter and tokens consisting of more than 25 letters were removed. In the vector representation, each document (ticket) $d \in D$ of our dataset D is represented as a feature

[6] http://www.otrs.com/

vector $x(d) = (x_{d,1}, \ldots x_{d,m})^T$, where each element of the vector corresponds to a specific term in the document corpus. For the experiments presented in this paper, vector elements were constructed as tf–idfs (term frequency – inverted document frequency) statistics, that is, $x(x_{d,t}) = \mathit{freq}(t,d) \cdot \log \frac{|D|}{|d \in D : t \in d|}$. Here, $\mathit{freq}(t,x)$ describes the frequency of the term t in the document d, $|D|$ the number of documents in the dataset, and $|d \in D : t \in d|$ the number of documents that contain the term t. In doing so, high values of $x_{d,t}$ indicate that the term t is specifically often used in the document d. In addition, each document in our data was manually provided a binary label $y(d)$, indicating the relevance of a document (ticket) to the user performance.

To categorize tickets with an unknown label, we employ several well established techniques for text classification, cf. also [12]. In particular, we evaluate the effects of the following classification methods:

- *Naive Bayes.* This probabilistic classifier uses the well-known Bayes formula under the assumption of independence of features (terms) [9]. Even though this assumption is of course not realistic, the Naive Bayes classifier has been shown to perform relatively well for text classification in practice [8, 13].
- *K-Nearest-Neighbor (KNN).* To classify a new instance, this method avoids building an explicit model, but retrieves the k most similar documents according to a distance metric instead and classifies the new document with the majority label in this neighborhood [6, 1].
- *Support vector machines (SVM).* Support vector machines in their most basic form classify new instances by computing a hyperplane in the vector space that separates positive and negative instances and maximizes the distances from the hyperplane to the closest positive and negative examples [13]. Since SVMs can handle large sets of features well, they are especially suited for text classification. To enhance performance of SVMs *kernel methods* have been proposed that transform the vector space into higher dimension. In that direction, we consider in particular the anova kernel, see [11], which was chosen after some initial experiments.

3.3 Metrics

To measure the classification performance of classifiers, we considered two kinds of evaluation metrics, that is, standard metrics from machine learning, and business relevant metrics that are more directly concerned with impact in practice.

Machine learning metrics Since the target label distribution in our corpus is heavily biased towards negative examples, i.e., documents that are not labeled as performance relevant, a high accuracy could be achieved by simply classifying all documents as negative. Therefore, we focus on the well known precision and recall framework. In particular, the *precision* is the fraction of correctly classified documents in the set of all documents that were classified as positive (performance relevant):

$$\mathit{precision} = \frac{|\mathit{positive\_examples} \cap \mathit{examples\_classified\_as\_positive}|}{|\mathit{examples\_classified\_as\_positive}|}.$$

By contrast, *recall* denotes the percentage of performance relevant tickets which are correctly identified as such:

$$recall = \frac{|positive\_examples \cap examples\_classified\_as\_positive|}{|positive\_examples|}.$$

Since there is usually a trade-off between these measures, the F_1 score, i.e., the harmonic mean between precision and recall, is commonly used as an overall performance metric:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}.$$

Business-relevant metrics Although text classifiers may yield a good performance with respect to scores like F_1, precision, and recall, they are not able to find all relevant tickets in general. From an enterprise view that might be fine, as long as the algorithms are capable of detecting major issues in the companies infrastructure. Of particular interest in our application context is to identify performance problems at specific branch offices (sites), and to preserve trends regarding the tickets per day. To measure classifier performances for these applications, we introduce two business-relevant metrics, namely:

- *relative amount of identified sites (rais)*: This metric depicts which share of relevant branch offices with face application performance degradations are identified by the classifier. It can be computed as:

  $$rais = \frac{\#\text{ correctly identified sites with performance relevant tickets}}{\#\text{ all sites issuing performance relevant tickets}}.$$

  This reflects the traditional recall measure on a site level. Performance problems reported for a branch office may be due to local incidents at the branch location like disturbances of the local network or the aggregation network.
- *trend preservation*: This metric highlights if the trend of identified tickets on a per-day basis independent of the specific office is preserved. This can be formalized using the well-known Pearson's correlation coefficient. Several tickets per day, possibly from different branch offices, are an indicator for large-scale incidents, probably in a data center or the wide-area network.

4 Evaluation

The next section presents experimental results for different classifiers in our application. We first present results based on classical machine learning metrics, then we discuss classifier performance with respect to the proposed business relevant metrics. All experiments were conducted in the RapidMiner[7] environment using additional scripts for pre- and postprocessing. Unless stated otherwise,

[7] https://rapidminer.com

we used the default parameters provided by RapidMiner. As an exception, we determined $C = 10$ as the best parameter setting for the SVMs after initial experiments. Similarly, we focused on the Euclidean distance as the distance metric for the KNN variations after initial experiments with other distance measures, e.g., the cosinus distance[8].

4.1 Performance of Machine Learning Algorithms

In this subsection, we investigate the performance of several classifiers with respect to the performance metrics *precision*, *recall* and F_1*-score*. For that purpose, we defined a training set consisting of the first 80% of the data set (in chronological order) and a fixed-size test set consisting of the last 20% of the data. The split was done with the rational, that available performance tickets are gathered and annotated to train a classifier. The trained classifier is then applied to newly generated tickets. Since the acquisition of correctly labeled training data is costly (in working hours) as it requires manual annotation of tickets, we are also specifically interested in how many labeled tickets are required by a classifier to achieve a certain performance. Therefore, we analyzed the impact of using only a randomly sampled subset of the training data. Results for the mean F_1 score as well as 80%-confidence intervals resulting from the analysis of at least 10 different samples are shown in Figure 1 and Table 1. It can be observed that support vector machines overall outperform Naive Bayes as well as K-nearest-neighbor (KNN) approaches. While Naive Bayes is clearly outmatched, KNN shows competitive performance if enough training data is available and if it is well parametrized. The best performance of KNN was achieved by $k = 25$, higher as well lower parameters for k lead to worse results. However, using less training

[8] Due to space constraints we do not report on initial experiments performed for parameter optimization

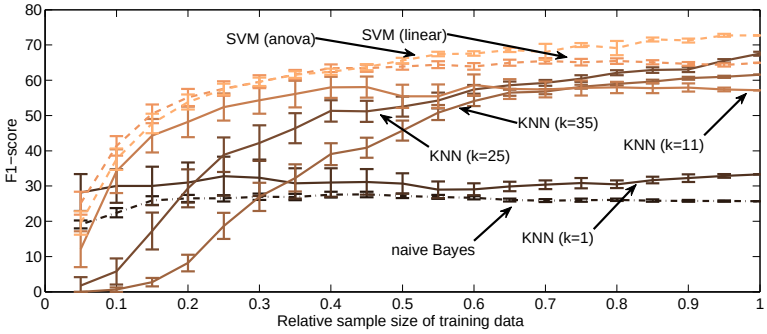


Fig. 1: Impact of differently sized random samples of the training set. In addition to the mean F_1-score (as shown by different line types and colors), confidence intervals from runs with multiple random samples are shown for different classifiers.

Table 1: F_1-Score, Precision, and Recall for different sample sizes

	F1		Precision		Recall	
Sample Size	40%	100%	40%	100%	40%	100%
Naive Bayes	27.5%	25.6%	19.9%	17.4%	44.9%	48.7%
KNN (N=1)	30.9%	33.3%	43.6%	47.6%	30.1%	25.6%
KNN (N=11)	57.9%	57.1%	83.2%	94.1%	44.9%	41.0%
KNN (N=25)	50.6%	68.8%	81.6%	95.4%	37.6%	53.8%
KNN (N=35)	39.0%	61.5%	87.5%	76.9%	26.2%	51.2%
SVM (linear)	63.4%	64.9%	73.6%	65.7%	56.0%	64.1%
SVM (anova)	62.4%	72.7%	83.9%	88.8%	49.9%	61.5%

data, SVMs clearly show the best classification results in linear as well as anova kernel variations. While these variations show similar F_1 scores for smaller sets of training data, the anova kernel version leads to better results for the full training data set. This difference seems to be specifically caused by a better precision of the classification, see Table 1. The overall superiority of support vector machines in our application is in line with previous findings for classifying text data, see [8, 13]. Based on these results, we focused on the support vector machines and KNN with $k = 25$ for further experiments.

In a second series of experiments, we assume a scenario where tickets are labeled in chronological order, e.g., on a per day basis. This follows the intuition that manpower for manual annotation of tickets may be available in some time intervals, but not continuously. Varying training set sizes are again evaluated on a fixed-size test set consisting of the (chronologically) last 20% of the data. Results for the F_1 score are shown in Figure 2 and Table 2. It can be seen that the SVMs achieve a good F_1 score already for $\approx 40\%$ of the training set. For a similar F_1 score, a training set size of 80% is required when using the KNN approach. Details on the F_1 score, precision, and recall are provided in Table 2. Hence, we can conclude a training set that contains $\approx 1/3$ of the whole data set

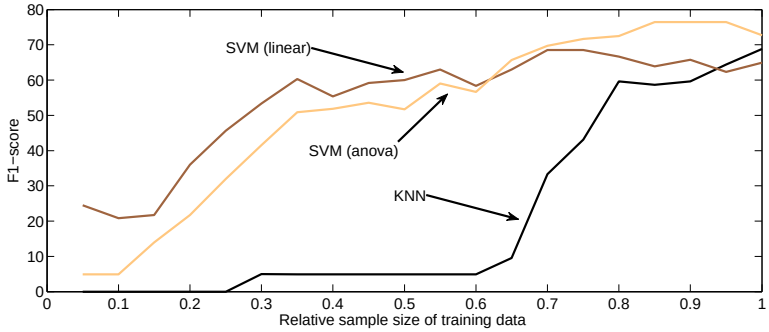


Fig. 2: Impact of differently sized subset of the training set in chronological order on the F_1-score. The investigated classifiers are highlighted with different colors.

Table 2: F_1-Score, Precision, and Recall for different subset sizes (40% and 80% of the training set).

	F1		Precision		Recall	
Sample Size	40%	80%	40%	80%	40%	80%
KNN (N=25)	4.8%	59.6%	50.0%	94.4%	2.5%	43.5%
SVM (linear)	55.3%	66.6%	69.2%	72.7%	46.1%	61.5%
SVM (anova)	51.8%	72.4%	93.3%	83.3%	35.8%	64.1%

Table 3: Precision for all sites and precision for sites with exactly one relevant ticket within the test data set.

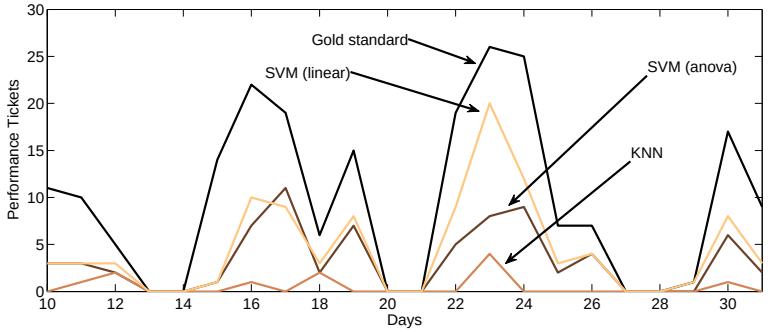
	Training set size: 1/3 of data		Training set size: 2/3 of data	
	$rais_{all}$	$rais_{oneticket}$	$rais_{all}$	$rais_{oneticket}$
KNN (N=25)	11.1%	2.9 %	50.0 %	31.3%
SVM (linear)	56.8%	25.7%	64.2%	43.8%
SVM (anova)	48.1%	22.9%	66.1%	43.8%

is sufficient to train a SVM model that performs well in tests on the last 20% of the data. In the following, we evaluate the impact of all classifiers on the business-relevant metrics for two training set sizes, namely $\approx 1/3$ and $\approx 2/3$ of the available data.

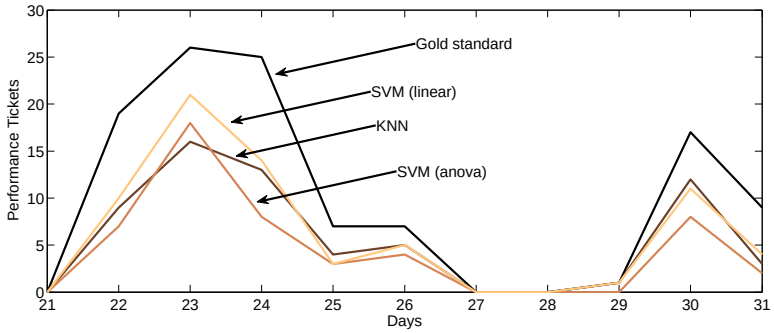
4.2 Impact of different Algorithms on Business-Relevant Metrics

This subsection highlights the performance of the presented approach with respect to the business-relevant metrics *identified sites* and *trend preservation*. The classifiers are trained with data sets of either the first 1/3 or the first 2/3 of the data set. The relative number of identified sites reporting performance problems ($rais_{all}$) are summarized in Table 3. This table additionally shows this measure restricted to those sites that reported exactly one performance problem, noted as $rais_{oneticket}$. For all cases, a training set of 2/3 of the data results in better identification ratio. Also, SVM classifiers outperform the KNN classifier in both settings. For the smaller training set, *rais* decreases considerably when using KNN, resulting in a poor detection of sites with application problems. Although the precision of the SVMs is reduced compared to a larger training data set, the overall detection rate remains good. However, a reliable detection of issues at specific sites cannot be guaranteed.

The results for the identification of relevant tickets on a per day basis are illustrated in Figure 3. Figure 3a depicts the quality of the different approaches for 2/3 as testing data, respectively day 10-31. Figure 3a depicts the quality of the different approaches for 1/3 as testing data, respectively day 19-31. None of the analyzed approaches is capable of identifying all relevant performance tickets per day, however, global trends are mostly preserved in most scenarios. For a training set size of 1/3, depicted in Figure 3a, the KNN classifier performs poorly. Several days with reported performance issues are not identified. Both



(a) Training set size: 1/3 of data



(b) Training set size: 2/3 of data

Fig. 3: Identification of relevant tickets on a per-day basis for the different classifiers (illustrated in different colors) and comparison with the gold standard.

SVM classifiers perform better, whereas the SVM with linear kernel performs best. Further, the SVM with linear kernel is the only one capable to identify the single performance ticket occurring on day 29. In case of a training set size of 2/3, illustrated in Figure 3b all approaches perform better, with the SVM with linear kernel performing best, and the SVM with anova kernel performing worst. The SVM with anova kernel is not capable to detect the single performance ticket on day 29, while both other approaches detect this ticket. The trend preservation is quantified using Pearson's correlation coefficient, as shown in Table 4. It can be observed, that the SVMs perform very well for our data set with respect to the business-relevant metrics. Although the KNN approach achieves a similar result quality in case of a large training data set, it clearly underestimates the performance problems for small training set sizes. Overall, the applied classifiers preserve the temporal trend of relevant tickets very well.

Table 4: Quantification of the trend preservation with Pearson's correlation coefficient.

	KNN (N=25)	SVM (linear)	SVM (anova)
Training set size: 1/3 of data	36.7%	90.1 %	89.8 %
Training set size: 2/3 of data	90.3%	96.5 %	97.4 %

5 Conclusions

This paper featured a case study that investigated the performance of machine learning techniques for the task of identifying relevant support tickets, i.e., tickets that indicate problematic applications in a thin client architecture environment. To this end, we evaluated the performance of several classification algorithms on over 12,000 manually annotated tickets in different scenarios and differently sized training sets. Besides traditional machine learning metrics such as F_1-score, precision and recall, also business relevant metrics derived from our target application were analyzed. As a result, supervised learning methods are overall well suited to identify application problems based on ticket data. Even though classifiers cannot categorize each individual ticket accurately, major issues at specific sites and general trends can be detected. Comparing different classifiers, support vector machines outperformed other evaluated techniques in our data set. For reproducibility and further studies, the pre-processed data sets will be made publicly available in an anonymized form.

In the future, we plan to explore the combination of standard machine learning classifiers with manually created keyword lists in order to further increase the classification accuracy. In addition, the full integration of automatic classification within business processes will allow for a continuous annotation of support tickets. Finally, the binary classification of tickets could be extended into a ticket ranking system that will allow to analyze the most important tickets first.

Acknowledgement

This work is supported by the Deutsche Forschungsgemeinschaft (DFG) under Grants HO TR 257/41-1. The authors alone are responsible for the content.

References

1. David Aha. *Lazy learning*. Springer Science & Business Media, 1997.
2. Mucahit Altintas and A. Cuneyd Tantug. Machine learning based ticket classification in issue tracking systems. In *Proc. of International Conference on Artificial Intelligence and Computer Science (AICS 2014)*, 2014.
3. Pedro Casas, Michael Seufert, Sebastian Egger, and Raimund Schatz. Quality of experience in remote virtual desktop services. In *Proc.of the Workshop on Quality of Experience Centric Management (QCMAN)*, Ghent, Belgium, May 2013.

4. K.V. Chandrinos, Ion Androutsopoulos, G. Paliouras, and C.D. Spyropoulos. Automatic web rating: Filtering obscene content on the web. In *Research and Advanced Technology for Digital Libraries*, volume 1923 of *Lecture Notes in Computer Science*, pages 403–406. Springer Berlin Heidelberg, 2000.
5. Sunita Chulani, P. Santhanam, Darrell Moore, Bob Leszkowicz, and Gary Davidson. Deriving a software quality view from customer satisfaction and service data. In *European Conference on Metrics and Measurement*, 2001.
6. Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.
7. Yixin Diao, Hani Jamjoom, and David Loewenstern. Rule-based problem classification in it service management. In *Proc. of IEEE International Conference on Cloud Computing (CLOUD'09)*, pages 221–228. IEEE, 2009.
8. Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In *International Conference on Information and Knowledge Management*, pages 148–155. ACM, 1998.
9. Irving John Good, Ian Hacking, R. C. Jeffrey, and Håkan Törnebohm. The estimation of probabilities: An essay on modern bayesian methods. *Synthese*, 16(2):234–244, 1966.
10. Thiago S. Guzella and Walmir M. Caminhas. A review of machine learning approaches to spam filtering. *Expert Systems with Applications*, 36(7):10206–10222, 2009.
11. Thomas Hofmann, Bernhard Schölkopf, and Alexander J. Smola. Kernel methods in machine learning. *The Annals of Statistics*, 36(3):1171–1220, 06 2008.
12. Andreas Hotho, Andreas Nürnberger, and Gerhard Paaß. A brief survey of text mining. In *Ldv Forum*, volume 20, pages 19–62, 2005.
13. Thorsten Joachims. *Text categorization with support vector machines: Learning with many relevant features*. Springer, 1998.
14. Jutta Kreyss, Steve Selvaggio, Michael White, and Zach Zakharian. Text mining for a clear picture of defect reports: A praxis report. In *Proc. of International Conference on Data Mining*, Melbourne, USA, November 2003.
15. Patrick Le Callet, Sebastian Möller, Andrew Perkis, et al. Qualinet white paper on definitions of quality of experience. *European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003)*, 2012.
16. Amelie Medem, Marc-Ismael Akodjenou, and Renata Teixeira. Troubleminer: Mining network trouble tickets. In *Proc. of Symposium on Integrated Network Management*, Long Island, USA, June 2009.
17. Audris Mockus, Ping Zhang, and Paul Luo Li. Predictors of customer perceived software quality. In *Proc. of International Conference on Software Engineering*, St. Louis, USA, May 2005.
18. Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.
19. Daniel Schlosser, Barbara Staehle, Andreas Binzenhöfer, and Björn Boder. Improving the qoe of citrix thin client users. In *Proc. of International Conference on Communications*, Cape Town, South Africa, May 2010.
20. F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.
21. Xing Wei, Anca Sailer, Ruchi Mahindru, and Gautam Kar. Automatic structuring of it problem ticket data for enhanced problem resolution. In *Integrated Network Management, 2007. IM'07. 10th IFIP/IEEE International Symposium on*, pages 852–855. IEEE, 2007.