IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 15, 2020, accepted October 13, 2020, date of publication October 21, 2020, date of current version November 3, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3032840

IT Ticket Classification: The Simpler, the Better

ALEKSANDRA REVINA[1,2], KRISZTIAN BUZA[3], AND VERA G. MEISTER[2]
[1]Chair of Information and Communication Management, Faculty of Economics and Management, Technical University of Berlin, 10623 Berlin, Germany
[2]Faculty of Economics, Brandenburg University of Applied Sciences, 14770 Brandenburg an der Havel, Germany
[3]Faculty of Informatics, Eötvös Loránd University, 1117 Budapest, Hungary

Corresponding author: Aleksandra Revina (revina@tu-berlin.de)

This work was supported in part by the Department of Data Science and Engineering, Faculty of Informatics, Eötvös Loránd University, and in part by the Thematic Excellence Programme, Industry and Digitization Subprogramme, National Research, Development, and Innovation (NRDI) Office, 2019 (Hungary) under Project no. ED_18-1-2019-0030.

ABSTRACT Recently, automatic classification of IT tickets has gained notable attention due to the increasing complexity of IT services deployed in enterprises. There are multiple discussions and no general opinion in the research and practitioners' community on the design of IT ticket classification tasks, specifically the choice of ticket text representation techniques and classification algorithms. Our study aims to investigate the core design elements of a typical IT ticket text classification pipeline. In particular, we compare the performance of TF-IDF and linguistic features-based text representations designed for ticket complexity prediction. We apply various classifiers, including kNN, its enhanced versions, decision trees, naïve Bayes, logistic regression, support vector machines, as well as semi-supervised techniques to predict the ticket class label of low, medium, or high complexity. Finally, we discuss the evaluation results and their practical implications. As our study shows, linguistic representation not only proves to be highly explainable but also demonstrates a substantial prediction quality increase over TF-IDF. Furthermore, our experiments evidence the importance of feature selection. We indicate that even simple algorithms can deliver high-quality prediction when using appropriate linguistic features.

INDEX TERMS IT tickets, linguistics, machine learning, text classification, TF-IDF, process complexity.

I. INTRODUCTION

With today's increased digitization, any enterprise maintains a broad application portfolio, which is often grown historically. Such a portfolio must be supported by large scale complex IT service environments [1]. These developments reveal a fundamental role of IT support systems in any organization's support operations. Two essential steps of any IT ticket processing, which are their correct prioritization and assignment, keep getting the attention of practitioners and the scientific community, especially in the context of the ever-increasing amount of tickets, errors, long lead times, and lack of human resources [2]–[7]. While small companies still tend to perform these steps manually, large organizations dedicate large budgets to the implementation of various commercial text classification solutions. Usually, these approaches are sophisticated monolithic software focused on accuracy at the cost of explainability and understandability. In our study, we are guided by this important issue in the choice of text representation and classification techniques.

Being one of the fastest-growing sectors of the service economy, recently, enterprise IT services have gained importance both in research and practice. One popular stream of this research is IT service management (ITSM) [8], [9], which focuses on the servitization of IT function, organization and management of IT service provision [10]. For a successful establishment of organizational infrastructure for IT services, IT practitioners implement IT service delivery with a reference process – the IT Infrastructure Library (ITIL) [11], [12]. Separate areas of ITIL, such as Incident, Problem or Change Management, deal with a large amount of unstructured text data, i.e., tickets issued in relation to the incidents, problems, or changes in the IT infrastructure products and services.

In the context of text classification, remarkable research has been done on the advantages and disadvantages of different text classification algorithms [13] and text representation techniques. In a specific context of IT tickets, a considerable amount of studies have been performed on the analysis of the ticket text data addressing such problems as correct

The associate editor coordinating the review of this manuscript and approving it for publication was Najah Abuali.

assignment and prioritization, identification of duplicates, and poorly described tickets [14], [15]. Prevailingly, TF-IDF (Term Frequency-Inverse Document Frequency) ticket text representation technique is used [16], [17]. Various ticket text classification [18], [19] and clustering approaches [20], [21] are studied.

In our previous research [22]–[24], we performed an in-depth linguistic analysis of the ITIL Change Management (CHM) ticket texts originating from the IT CHM ticket processing department of a big enterprise with more than 200,000 employees worldwide. Using the elaborated linguistic representations of the ticket texts, we implemented a rule-based approach to predict the process complexity [25]. The precision of this approach reached approximately 62%. Apart from such a precision quality, the process of rule set establishment demanded a lot of analysis and testing effort on our and process experts' side. Another substantial disadvantage of such an approach, particularly in complex scenarios, is difficulty in maintenance, inability to learn and scale, as adding new rules requires revising the existing ones.

In this study, our objective is to address the challenges above (choice of text representation technique, choice of a classifier, a growing amount of IT tickets, and the need for explainable text classification solutions to support the IT helpdesk workers). Furthermore, we aim to develop an understanding which factors are relevant for the prediction quality with focus on text representation techniques, feature selection, and text classification algorithms.

In the context of representation of ticket text data, we compare the performance of a commonly accepted and widely used TF-IDF [26] with the linguistic approach described in our previous work [22]–[24]. Despite massive research efforts on IT ticket classification, the representation based on the linguistic features considered in this study has not been systematically compared with TF-IDF representation in the ticket classification context yet.

In the classification methods, we focus on various classifiers widely used for text [13] and ticket [14] classification, including kNN, its enhanced versions, so-called hubness-aware classifiers [27], [28], decision trees [29], naïve Bayes [30, pp. 253–286], logistic regression [31], support vector machines [32] as well as semi-supervised techniques. The latter includes kNN with self-training [33], Semi-sUpervised ClassifiCation of TimE SerieS algorithm (SUCCESS) [34], and its improved version QuickSUCCESS [35]. Although state-of-the-art technology allows us to capture a considerable amount of instances in many applications (e.g., millions of images may be observed), in our case, it is costly and difficult to obtain class labels for a large number of tickets. On the one hand, to label the tickets, expert knowledge is required. On the other hand, as various aspects have to be taken into account, even experts need much time for this task. Therefore, due to the limited amount of labeled tickets, we also experimented with semi-supervised approaches for ticket classification.

The rest of the paper is organized as follows: we give an overview of related works in Section II. Section III presents our methods, followed by the experiments in Section IV. Subsequently, we discuss the implications of the findings and conclude with limitations and future research directions.

II. RELATED WORK

In this paper, as we focus on the problem of CHM ticket classification, the related work is structured as follows: (i) text representation, (ii) text and ticket classification.

A. TEXT REPRESENTATION

Text representation is one of the essential building blocks of approaches for text mining and information retrieval. It aims to transform a text document into a numeric feature vector [36], [37], to allow a computational analysis of textual data. We studied different techniques of text representation and feature extraction and structured them into three main categories: weighted word techniques, word embedding [13], and linguistic features.

1) WEIGHTED WORDS

Weighted words approaches are based on counting the words in the document and computing the document similarity directly from the word-count space [13]. Due to their simplicity, Bag-of-Words (BoW) [38] and TF-IDF [39] can be considered as the most common weighted words approaches. The employed techniques usually rely on the aforementioned text representations rather than in-depth linguistic analysis or parsing [40]. As these models represent every word in the corpus as a one-hot-encoded vector, they are incapable of capturing the word semantic similarity and can become very large and technically challenging [13]. To address these limitations, we use linguistic features in the proposed approach. As described in Section III, the number of our linguistic features is independent of the size of the corpus, and they are capable of capturing relevant aspects of semantics.

2) WORD EMBEDDING

With the progress of research, new methods, such as word embedding, have come up to deal with the limitations of weighted words approaches. Word embedding techniques learn from sequences of words by considering their occurrence and co-occurrence information. Each word or phrase from the corpus is mapped to a high dimensional vector of real numbers, and the mapping from words to vectors is learned based on the surrounding words in the corpus. Various methods have been proposed, such as Word2Vec [41], Doc2Vec [42], GloVe [43], FastText [44], [45], contextualized word representations [46], [47]. These methods consider the semantic similarity of the words. However, they need a large corpus of text datasets for training. To solve this issue, pre-trained word embedding models have been published [48]. Unfortunately, the models do not work for the words outside of the corpus. Another limitation of the word embedding models is related to the fact that they are

trained based on the words appearing in a pre-defined window. This is an inherent limitation for considering different meanings of the same word occurring in different contexts. While addressing this limitation, the contextualized word representations techniques based on the context of the word in a document [46], [47] were developed. Nonetheless, in real-world applications, new words may appear (in the description of a new ticket, the customer may use phrases and specific words that have not been used before). These new words are not included in the corpus at training time. Therefore, these word embedding techniques will fail to produce a correct mapping for these new words. While this problem may be alleviated by retraining the model, this requires a lot of data and computational resources (CPU, RAM). In contrast, as it is discussed next, it is straightforward to use our linguistic features text representation in case of new words as it is not based on word embedding.

3) LINGUISTIC FEATURES

Text representations based on linguistic features have been introduced to address the mentioned limitations of weighted words and word embeddings. Below, we list some examples.

Lexicon-based sentiment analysis is a well-established text classification technique [49]. Synonymy and hypernymy are known approaches to increase prediction quality [36]. Extensive research on the linguistic analysis of ticket texts has been performed in [50], [51]. The authors use parts-of-speech (PoS) count and specific terms extractions to define the reported problem's severity. Coussement and Van den Poel extract the following linguistic features: word count, question marks, unique words, the ratio of words longer than six letters, pronouns, negations, assents, articles, prepositions, numbers, time indication [52]. The study results indicated a profoundly beneficial impact of combining traditional features, like TF-IDF and singular value decomposition, with linguistic style into one text classification model. However, the authors declare a demand for more experiments and research. This demand is addressed in our work.

There is no unified opinion in the research community, whether the linguistic approaches are good enough to substitute or enhance the traditional text representations. They are more complex to implement and do not compensate for this complexity with the expected performance increase. Both proponents [53]–[58] and opponents [59] of the linguistic approach provide convincing experimental results. In our work, we amend these research discussions with case study-based findings while comparing the performance of linguistic features with TF-IDF. Word embedding techniques are not applicable in our study due to the following reasons: (i) pre-trained models would not perform well considering the domain-specific vocabulary which contains many new words compared to the training corpus, (ii) at the same time, a limited text corpus would not be enough for training a new model (or retraining an existing one), (iii) industrial applications prevailingly demand explainable models to be able to understand and correct classification mistakes.

TABLE 1. Text representation techniques.

Technique	Strengths	Weaknesses
Weighted words		
BoW, TF-IDF	• simple to implement • work well with new words • established approaches to extract the most descriptive terms in a document • do not need data to train the mapping	• do not consider syntax and semantics
Word embedding		
Word2Vec, Doc2Vec, GloVe, contextualized word representations	• consider syntax and semantics • contextualized word representations: consider polysemy	• need much data for training • consider only words that appear in the training data • computationally expensive to train (CPU, RAM)
Linguistic features		
word count, special characters, parts of speech, unique words, long words, context-specific taxonomies, lexicons, sentiment, etc.	• highly explainable and understandable • wide choice of features • do not necessarily depend on capturing semantics and context • depending on the selected features, can capture both syntax and semantics • do not need data to train the mapping	• expert knowledge is required to define an appropriate set of features

Table 1 summarizes the strengths and weaknesses of the discussed techniques.

B. TEXT AND TICKET CLASSIFICATION

Text classification, also referred to as text categorization or text tagging, is the task of assigning a text to a set of pre-defined categories [60]. Traditionally, this task has been done by human experts. Expert text classification, for example, remains widely used in qualitative research in the form of coding or indexing [61], [62]. Nevertheless, with the growing amount of text data, the attention of the research community and practitioners shifted to automatic methods. One can differentiate three main groups of automatic text classification approaches: rule-based, approaches based on machine learning (ML), and a combination of both in a hybrid system. Ticket classification in software development and maintenance has been studied to tackle challenges, such as correct ticket assignment and prioritization, predicting time and number of potential tickets, and avoiding duplicate tickets.

1) RULE-BASED APPROACHES

As the name suggests, this kind of text classification system is based on a set of rules determining classification into pre-defined categories. In general, rule-based classifiers are a

popular data mining method applicable to diverse data types. It became popular due to its transparency, explainability, and relative simplicity [63]. Various types of rule development can be distinguished: ML-based such as decision trees [64] or association rules [65], [66], handcrafted rules created by the experts, or hybrid approaches [67], [68]. It is essential to mention that rule-based systems work well with small rule sets. However, they become difficult to build, manage, and change with the growing number of rules.

In our previous work, we conceptualized a recommender system for IT ticket complexity prediction using rule-based classification, i.e., handcrafted rules and rules based on decision trees [25]. Nonetheless, due to the complexity of the domain, the process of rule development consumed much time and effort. The system was difficult to manage and change.

2) MACHINE LEARNING APPROACHES

In the context of ML, text classification can be defined using the following formalization approach. If there is a set of classification labels S and a set of training instances F, each labeled using class labels in S, the model must use F to learn to predict the class labels of unseen instances of the same type [69]. In text classification, F is a set of labeled documents from a corpus. The labels can be extracted topics, themes, writing styles, judgments of the documents' relevance [36]. Regarding the prediction itself, various techniques such as kNN, its enhanced versions (hubness-aware classifiers), decision trees, naïve Bayes, logistic regression, support vector machines, neural networks have been introduced [13]. ML approaches have been shown to be more accurate and easier to maintain compared to rule-based systems [70]. At the same time, it is challenging to select the best ML technique for a particular application [13]. Table 2 summarizes the strengths and weaknesses of the main approaches for text classification.

Most ML techniques, including all the approaches above, require a large amount of training data. However, as described previously, in our application, it is challenging to obtain labeled training data. In contrast, semi-supervised learning (SSL) allows inducing a model from a large amount of unlabeled data combined with a small set of labeled data. For an overview of major SSL methods, their advantages and disadvantages, we refer to [71]. For the above reasons, we experimented with semi-supervised ML approaches. In particular, we used kNN with self-training and SUCCESS [34]. Although SUCCESS showed promising results, it does not scale well to large datasets. We addressed this limitation by suggesting a scaling technique for SUCCESS, and we called the resulting approach QuickSUCCESS [35].

III. METHODS

In the choice of the methods, we put a special emphasis on the explainability and understandability of the classification results for the process worker.

TABLE 2. Text classification techniques.

Technique	Strengths	Weaknesses
Rule-based classification		
ML-based rule development with decision trees, association rules, handcrafted rules	• able to handle a variety of input data • explainable	• with the growing number of rules – difficult to build, manage, maintain, and change
ML-based classification		
kNN, hubness-aware classifiers	• non-parametric • adapts easily to various feature spaces	• finding an appropriate distance function for text data is challenging • prediction might become computationally expensive
SUCCESS/ QuickSUCCESS	• learns both from labeled and unlabeled instances	• finding an appropriate distance function for text data is challenging • computationally expensive training
decision trees	• fast in learning and prediction	• overfitting
	• explainable	• instability even to small variations in the data
naïve Bayes	• showed promising results on text data [72]	• a strong assumption about the data independence
logistic regression	• computationally inexpensive	• a strong assumption about the data independence • is not appropriate for non-linear problems
support vector machines	• robust against overfitting • able to solve non-linear problems	• lack of transparency in results • the problem of choosing an efficient kernel function
neural networks	• flexible with the design of features • can achieve rather accurate predictions	• requires a large amount of data for training • may be extremely computationally expensive • finding an efficient architecture and structure is difficult • not explainability

A. FEATURE EXTRACTION

To allow for automated analysis, texts are usually transformed into a vector space. Unnecessary characters and words (stop words) are removed. Afterward, diverse feature extraction techniques can be applied. In our study, we compare two types of features: TF-IDF and linguistic features.

Next, we describe linguistic features in detail. These features are specifically designed for the task of IT ticket complexity prediction. Three levels of text understanding are commonly distinguished: (1) objective (answering the

questions who, what, where, when) measured by semantic technologies, (2) subjective (an emotional component of a text) – by sentiment analysis, and (3) meta-knowledge (information about the author outside of the text) measured by stylometry or stylistic analysis [73]. Accordingly, we develop a set of features which are aggregated by the respective measures indicating the IT ticket complexity. We proposed these features in our initial works [22]–[24].[1] A detailed explanation of linguistic features is provided below using an anonymized IT ticket example (see also Table 3).

TABLE 3. Overview of linguistic features illustrated by a ticket example.

Ticket example: "Refresh service registry on the XYZ-ZZ YYY server. See attachment for details."		
Aspects	**Description**	**Linguistic feature**
objective knowledge aspect [24]	relative occurrence of words according to the taxonomy of routine, semi-cognitive and cognitive terms	routine = 0.8
		semi-cognitive = 0.2
		cognitive = 0
subjective knowledge aspect [23]	relative occurrence of words with positive, neutral, and negative sentiment	negative = 0
		neutral = 1
		positive = 0
meta-knowledge aspect [22]	word count	12
	occurrence of nouns in all words	0.5
	occurrence of unique nouns in all nouns	1
	occurrence of verbs in all words	0.17
	occurrence of unique verbs in all verbs	1
	occurrence of adjectives in all words	0.07
	occurrence of unique adjectives in all adjectives	1
	occurrence of adverbs in all words	0
	occurrence of unique adverbs in all adverbs	0
	wording style [22]	0 (no repeating words)

1) OBJECTIVE KNOWLEDGE ASPECT

Core research in Natural Language Processing (NLP) addresses the extraction of objective knowledge from text, i.e., which concepts, attributes, and relations between concepts can be extracted from text, including specific relations such as causal, spatial, and temporal ones [73]. Among diverse approaches, specifically, taxonomies and ontologies, are widely used in the business context [74], [75]. Thus, we suggest a specific approach of objective knowledge extraction using the Decision-Making Logic (DML) taxonomy [24] illustratively presented in Appendix I. Herewith, it is aimed to discover the decision-making nature of activities, called DML level. We use the following DML levels: *routine, semi-cognitive,* and *cognitive* (corresponding to the columns of the table in Appendix I). Using a Latent Dirichlet Allocation Algorithm (LDA) [76], we identify the most important keywords, see [24] for details. Each of the keywords is associated with a DML level. For example, the keywords *user*, *test, request, etc.* are associated with *routine*, whereas the keyword *management* belongs to *cognitive*. We detect these keywords in IT ticket texts. Based on the total number of detected keywords, we calculate the relative occurrence of the words of each category in the ticket text. In the example shown in Table 3, the total number of detected words equals five, out of which four words belong to *routine* (*server, attach, detail, see*), one to *semi-cognitive* (*service*), and no word to *cognitive*. Thus, the corresponding features are calculated as follows: *routine* $4/5 = 0.8$ and *semi-cognitive* $1/5 = 0.2$.

2) SUBJECTIVE KNOWLEDGE ASPECT

To extract a subjective knowledge aspect, we perform a sentiment analysis [77]. In [23], we suggest a specific business sentiment approach as an instrument for measuring the emotional component of an IT ticket. This latent information is extracted from the unstructured IT ticket texts with the help of a lexicon-based approach. As standard lexicons do not work well in our context of IT ticket classification, using the state-of-the-art VADER [78] and LDA, we developed a domain-specific lexicon, see [23] for details. Our lexicon can also be found in Appendix II.

Each of the words is associated with positive, negative, or neutral sentiment. Words with valence scores greater than 0 are considered positive, whereas words with a valence score less than 0 are considered negative. All other words are considered to have a neutral sentiment. We determine the proportion of words with negative, neutral, and positive sentiment for each IT ticket text and use these values as features. In our example, there are no words with positive or negative sentiment. Therefore, the ticket is assigned to be entirely neutral.

3) META-KNOWLEDGE ASPECT

In our case, meta-knowledge is the knowledge about the author of an IT ticket. The quality of the written text will likely depend on such factors as the author's professionalism and expertise, level of stress, and different psychological and sociological properties [73]. To extract the meta knowledge aspect, we use the following features [22]: (1) IT ticket text length, (2) PoS features, (3) wording style [22] calculated with the Zipf's law of word frequencies [79].

By length, we mean the number of words in the IT ticket text. This feature is motivated by the following observation: in most cases, IT tickets texts containing a few words, such as *update firewalls,* refer to simple daily tasks. Therefore, short ticket texts may be an indication of the low complexity of the ticket. In the example shown in Table 3, the length of the ticket is 12 words.

As for PoS features, we consider the following PoS tags: nouns, verbs, adjectives, and adverbs. For each of them, we calculate their absolute occurrence, i.e., the *total* number of

[1] https://github.com/IT-Tickets-Text-Analytics

words having that PoS tag (for example, the total number of nouns). Subsequently, we calculate their relative occurrence, called *occurrence* for simplicity, i.e., the ratio of nouns, verbs, adjectives, and adverbs relative to the length of the text. We use these occurrences as features. In the example shown in Table 3, the occurrence of nouns (*registry, XYZ-ZZ, YYY, server, attachment, details*) in all words is $6/12 = 0.5$.

We also calculate the number of *unique* words having the considered PoS tags (for example, number of unique nouns). Then, we calculate the occurrence of *unique* nouns, verbs, adjectives, and adverbs relative to the number of *all* nouns, verbs, adjectives, and adverbs, respectively. We use these occurrences as features as well. In Table 3, the uniqueness of nouns is $6/6 = 1$ (no repeating words).

According to Zipf's word frequency law, the distribution of word occurrences is not uniform, i.e., some words occur very frequently. In contrast, others appear with a low frequency, such as only once. Our wording style feature describes how extreme this phenomenon is in the IT ticket text, i.e., whether the distribution of occurrences is close to being uniform or not. For details, we refer to [22].

B. SUCCESS

After the features have been extracted and the texts are represented in the form of numerical vectors, they can be fed into ML classifiers. To make sure that our paper is self-contained, below, we shortly review SUCCESS and its scaled version.

We define the semi-supervised classification problem as follows: given a set of labeled instances $L = \{(x_i, y_i)\}_{i=1}^{l}$ and a set of unlabeled instances $U = \{x_i\}_{i=l+1}^{n}$, the task is to train a classifier using both L and U. We use the phrase *set of labeled training instances* to refer to L, x_i is the i-th instance, y_i is its label, whereas we say that U is the *set of unlabeled training instances*. The labeled instances (elements of L) are called seeds. We wish to construct a classifier that can accurately classify any instance, i.e., not only elements of U. For this problem, we proposed the SUCCESS approach that has the following phases:

1. The labeled and unlabeled training instances (i.e., instances of U and L) are clustered with constrained single-linkage hierarchical agglomerative clustering [34]. While doing so, we include cannot-link constraints for each pair of labeled seeds, even if both seeds have the same class labels.
2. The resulting top-level clusters are labeled by their corresponding seeds.
3. The final classifier is 1-nearest neighbor trained on the labeled data resulting at the end of the 2nd phase. This classifier can be applied to unseen test data.

As we noticed the relatively slow training speed of semi-supervised classifiers, we implemented resampling [80], a technique similar to bagging, to accelerate the classification and possibly improve classification results [81].

In particular, we select a random subset of the data and train the model on the selected instances. This process is repeated several times. When making predictions for new instances, the predictions of all the models above are aggregated by majority voting. As the sample size is much smaller than the size of the original dataset and the training has a superlinear complexity, the training of several models on small samples is computationally cheaper than the training of the model on the entire dataset. While kNN with resampling has been established in the research community since a long time ago [82], to the best of our knowledge, this is the first attempt to speed up the SUCCESS algorithm using resampling. We call the resulting approach QuickSUCCESS [35]. Next, we explain QuickSUCCESS in detail.

Algorithm 1 Training QuickSUCCESS

Require: labeled training data L, unlabeled training data U, sample size r, number of classifiers m
Ensure: trained classifiers $\{C^{(i)}\}_{i=1}^{m}$, predicted labels for U
1: **for** i in $1 \ldots m$ **do**
2: $\quad U^{(i)} \leftarrow$ random_sample (U, r)
3: $\quad C^{(i)}, \hat{y^{(i)}} \leftarrow$ t rain SUCCESS $(L, U^{(i)})$
4: $\quad$ **for** x in $U^{(i)}$ **do**
5: $\quad\quad$ vote for the label $\hat{y^{(i)}}[x]$ for unlabeled instance x
6: $\quad$ **end for**
7: **end for**
8: return $\{C^{(i)}\}_{i=1}^{m}$, class labels predicted based on the majority vote

The core component of SUCCESS is constrained hierarchical agglomerative clustering. Although various implementations are possible, we may assume that it is necessary to calculate the distance between each pair of unlabeled instances as well as between each unlabeled instance and each labeled instance. This results in a theoretical complexity of $O(l(n-l) + (n-l)^2) = O(n^2 - l^2 - nl)$ at least,[2] where l denotes the number of labeled instances ($l = |L|$), while n indicates the number of all instances (labeled and unlabeled) that are available at training time ($n = |L| + |U|$). Under the assumption that l is a small constant, the complexity of distance computations is $O(n^2)$.

Considering only the aforementioned distance computations required for SUCCESS, the computational costs in the case of a dataset containing n/c instances is c^2-times lower than in the case of a dataset containing n instances. Therefore, even if the computations have to be performed on several "small" datasets, the overall computational costs may be an order of magnitude lower. In particular, computing SUCCESS m-times on a dataset containing r instances has a total computational cost of $O(m \times r^2)$. Under the assumption that $r = n/c$ and $m \approx O(c)$, the resulting complexity is $O(n^2/c)$. Based on the analysis, we propose to speed up SUCCESS by repeated sampling. In particular, we sample the set of unlabeled instances m-times. From now on, we will use $U^{(j)}$ to denote the j-th sample, $1 \leq j \leq m$.

[2]We use the $O(\ldots)$ notation of Complexity Theory

Each $U^{(j)}$ is a random subset of U containing r instances ($|U^{(j)}| = r$). For simplicity, each instance has the same probability of being selected. To train the j-th classifier, we use all the labeled instances in L together with $U^{(j)}$. When sampling the data, the size r of the sample should be chosen carefully so that the sampled data is representative in the sense that the structure of the classes can be learned. This is illustrated in Figure 1.

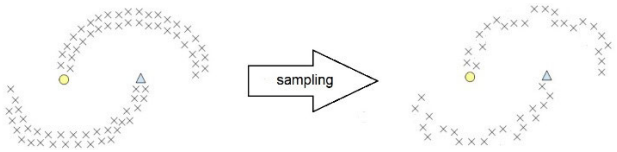


FIGURE 1. When sampling the data, the size r of the sample should be chosen carefully so that the sampled data is representative.

As the sampling is repeated m-times, we induce m classifiers denoted as $C^{(1)}, \ldots, C^{(m)}$. Each classifier $C^{(j)}$ predicts the label for the unlabeled instances in the corresponding sample of the data, i.e., for the instances of $U^{(j)}$. More importantly, each of these classifiers can be used to predict the label of new instances, i.e., instances that are neither in L nor in U. Labels for the instances in U are predicted as follows. For each $x_i \in U$, we consider all those sets $U^{(j)}$ for which $x_i \in U^j$ and the classifiers that were trained using these datasets. The majority vote of these classifiers is the predicted label of x_i. Our approach is summarized in Algorithm 1.

The label of a new instance $x \notin L \cup U$ is predicted as the majority vote of all the classifiers $C^{(1)}, \ldots, C^{(m)}$. We note that the computations related to the classifiers mentioned above $C^{(1)}, \ldots, C^{(m)}$ can be executed in parallel, which may result in additional speed-up in case of systems where multiple CPUs are available, such as high-performance computing (HPC) systems.

The same resampling technique can be applied with other classifiers as well. In particular, in our experiments, we used it with a semi-supervised variant of kNN, called *kNN with self-training and resampling*.

1) OTHER CLASSIFIERS

We experimented with various other classifiers such as kNN and its enhanced versions, kNN with self-training [83], and kNN with self-training and resampling. As the emergence of bad hubs was shown to characterize textual data [84], we included hubness aware variants of kNN in our study, in particular: kNN with **E**rror **C**orrection (ECkNN) [85], [86], **H**ubness-**W**eighted kNN (HWkNN) [87], **H**ubness-**F**uzzy kNN (HFNN) [88], **N**aive **H**ubness-**B**ayesian kNN (NHBNN) [89]. Additionally, we used decision trees, naïve Bayes, logistic regression, and support vector machines.

Self-learning is a semi-supervised technique known to improve the learning process in case of a large number of unlabeled and a small number of labeled instances [83]. Therefore, we also use kNN with self-training. Hereby, kNN is iteratively retrained with its own most confident predictions [33].

IV. EXPERIMENTAL EVALUATION

The goal of our experiments is to analyze the contribution of the proposed text representation, study the effect of feature selection and the performance of various classifiers.

Our experimental design is summarized in Figure 2 and includes the following steps: (1) collect the case study data; (2) pre-process the data; (3) label part of the data; (4) split the data into training and test sets; (5) extract two sets of features: TF-IDF and linguistic features; (6) apply the classifier; (7) evaluate the results with standard metrics. All the experiments were conducted using Python 3.6. In the case of naïve Bayes, logistic regression, decision trees, and support vector machines, we used the publicly available implementation from scikit-learn,[3] whereas in the case of hubness-aware classifiers and SUCCESS, we used PyHubs.[4] We implemented kNN with and without self-training as well as the proposed speed-up technique based on resampling on our own. We performed all the experiments on an Intel®Core™i7 16 GB RAM machine.

A. DATASETS

The datasets (see Table 4) in the form of IT ticket texts originate from an ITIL CHM department of a big enterprise. The datasets were received (step 1) according to their availability. They covered the whole period obtainable at the moment of this study. The first dataset (Data1) comprised 28,243 tickets created from 2015 to 2018.

TABLE 4. Datasets.

#	time period	# of ticket texts	# of acquired labels
Data1	2015 – 2018	28,243	30
Data2	January – May 2019	4,684	60

The data was pre-processed (step 2) by removing stop words, punctuation, turning to lowercase, stemming and converted into a CSV-formatted corpus of ticket texts. The tickets texts contained prevailingly English texts (more than 80% of English words, a small portion of German words was present). The second dataset (Data2) comprised 4,684 entries in prevailingly English language from the period January – May 2019. The length of IT ticket texts varies from 5-10 words to 120-150.

As labeling of the data is a time-consuming and tedious process and our data comes from an operational unit fully overbooked in the daily work, which is often the case in the industry, experts labeled 30 and 60 tickets of Data1 and Data2 (step 3). To provide a correct label, the case study workers needed to analyze all the factors influencing the IT

[3]https://scikit-learn.org/stable/
[4]http://www.biointelligence.hu/pyhubs/

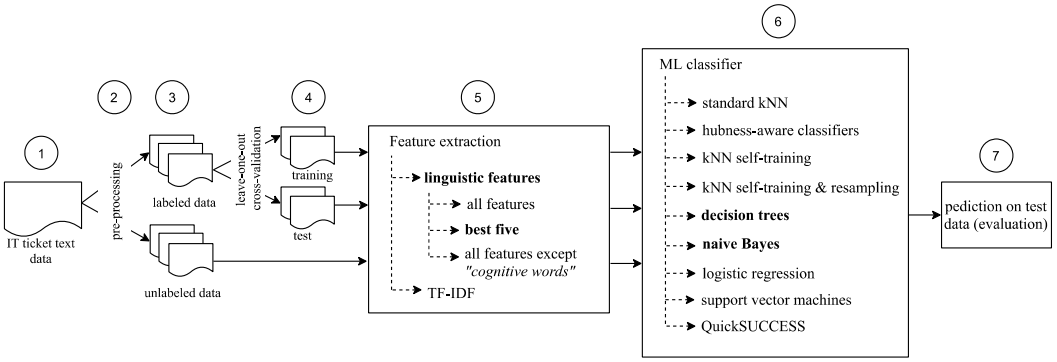


FIGURE 2. Overview of the experimental pipeline.

ticket complexity, such as the number of tasks; the number of Configuration Items (CIs), specifically applications; if the ticket had to be executed online or offline (planning of downtime and considering affected CIs); involvement of Change Advisory Board (CAB), etc. Therefore, labeling a single ticket could take up to 30 minutes. For complexity class labels, a qualitative scale of low, medium, and high has been selected to simplify the classification task and as a well-known scale of priority ratings [90]. Although two datasets are coming from the same case, a distinct period of time and a different number of labels allowed us to test our approaches in two independent settings.

B. EXPERIMENTAL SETTINGS

To evaluate classifiers, we use leave-one-out cross-validation [91]. We select one of the labeled instances as a test instance and use all the remaining instances as training data (step 4). The process of selecting a test instance and training the classifier using all the other instances is repeated as many times as the number of labeled instances. Hereby, in each round, a different instance is selected as the test instance. For example, in the case of Data2, we select one labeled instance out of the available 60 labeled instances as a test instance and use the remaining 59 as labeled training data L. The SSL classifiers also used the 4,684-60 = 4,624 unlabeled instances as unlabeled training data U. We repeat the process 60 times.

As evaluation metrics, we use accuracy, average precision, average recall, and F-score (step 7). To compare the differences in the classifiers' performance, we use the binomial test by Salzberg [92] at the significance threshold of 0.05.

When applying semi-supervised classifiers, we consider all the unlabeled instances available at the training time. Whenever the opposite is not stated, we use kNN with k=1 and Euclidian distance, which is also justified by theoretical and empirical studies [93], [94]. In the case of kNN with self-training, we set the number of self-training iterations to 100.

In respect to support vector machines (SVMs), we used internal leave-one-out cross-validation on the training data to identify appropriate settings of hyper-parameters. In particular, we considered polynomial and RBF kernels, and we used a grid search to determine the best values of the complexity constant C and the kernel coefficient *gamma* in the range 10^{-5}, 10^{-4}, …, 10^{4}, 10^{5}. As for the degree d of the polynomial kernel, we considered all integer values between 1 and 10.

Regarding linguistic features, we include feature selection tests to identify which features play an important role in prediction quality. Similarly to [95], [96], we use logistic regression to determine the most predictive features based on their weights. We also compare classifiers with a statistical method, i.e., an expert-defined decision rule based on the presence of cognitive words.

C. COMPARISON OF SUCCESS AND QUICKSUCCESS

As an initial experiment, we measured the execution time of SUCCESS and QuickSUCCESS. We used the linguistic features representation. In the case of QuickSUCCESS approach, we selected $r = 100$ instances and trained $m = 100$ models. Table 5 shows our results: the execution time of one round of cross-validation (in seconds) and the accuracy. As one can see, the proposed technique leads to several orders of magnitude speed-up, both in the case of Data1 and Data2, with negligible loss of prediction quality (the difference corresponds to the misclassification of one additional instance). Hence, in further experiments, we used the QuickSUCCESS algorithm.

TABLE 5. Execution time (in seconds) and accuracy of SUCCESS and QuickSUCCESS.

	time (seconds)	accuracy
Data1 SUCCESS	75,831	0.567
Data1 QuickSUCCESS	2	0.533
Data2 SUCCESS	452	0.767
Data2 QuickSUCCESS	2	0.750

D. RESULTS

Table 6 provides the obtained values of accuracy, an average of precision and recall calculated over the three classes of low, medium, and high complexity. F-score is calculated based on the average precision and recall.

We compare the classifiers' performance using linguistic features with that of classifiers using TF-IDF on Data1 and Data2. We point out the systematic improvement in the prediction quality of algorithms when using linguistic features.

TABLE 6. Evaluation results using linguistic features and TF-IDF.

Algorithm	Accuracy	Average precision	Average recall	F-score
Data1: linguistic features				
standard kNN	0.533	0.526	0.498	0.511
kNN self-training	0.533	0.526	0.498	0.511
kNN self-training & resampling	0.533	0.526	0.498	0.511
ECkNN	0.633	0.599	0.580	0.589
HFNN	0.600	0.411	0.490	0.447
HWkNN	0.533	0.526	0.498	0.511
NHBNN	0.433	0.144	0.333	0.202
decision trees	1.000	1.000	1.000	1.000
naïve Bayes	0.967	0.972	0.944	0.958
logistic regression	0.500	0.475	0.463	0.469
SVM	0.600	0.575	0.579	0.577
QuickSUCCESS	0.533	0.542	0.523	0.532
Data1: TF-IDF				
standard kNN	0.200	0.067	0.333	0.111
kNN self-training	0.200	0.067	0.333	0.111
kNN self-training & resampling	0.200	0.067	0.333	0.111
ECkNN	0.433	0.144	0.333	0.202
HFNN	0.433	0.144	0.333	0.202
HWkNN	0.200	0.067	0.333	0.111
NHBNN	0.433	0.144	0.333	0.202
decision trees	0.433	0.144	0.333	0.202
naïve Bayes	0.267	0.738	0.389	0.510
logistic regression	0.433	0.144	0.333	0.202
SVM	0.400	0.138	0.308	0.190
QuickSUCCESS	0.200	0.067	0.333	0.111
Data2: linguistic features				
standard kNN	0.750	0.593	0.576	0.584
kNN self-training	0.750	0.593	0.576	0.584
kNN self-training & resampling	0.750	0.593	0.576	0.584
ECkNN	0.733	0.586	0.568	0.577
HFNN	0.733	0.539	0.528	0.534
HWkNN	0.750	0.593	0.576	0.584
NHBNN	0.700	0.233	0.333	0.275
decision trees	1.000	1.000	1.000	1.000
naïve Bayes	1.000	1.000	1.000	1.000
logistic regression	0.800	0.552	0.582	0.567
SVM	0.767	0.531	0.544	0.537
QuickSUCCESS	0.750	0.593	0.576	0.584
Data2: TF-IDF				
standard kNN	0.700	0.233	0.333	0.275
kNN self-training	0.700	0.233	0.333	0.275
kNN self-training & resampling	0.700	0.233	0.333	0.275
ECkNN	0.700	0.233	0.333	0.275
HFNN	0.700	0.233	0.333	0.275
HWkNN	0.700	0.233	0.333	0.275
NHBNN	0.700	0.233	0.333	0.275
decision trees	0.700	0.233	0.333	0.275
naïve Bayes	0.467	0.419	0.468	0.442
logistic regression	0.700	0.233	0.333	0.275
SVM	0.700	0.233	0.333	0.275
QuickSUCCESS	0.700	0.233	0.333	0.275

Namely, we observed that the classifiers' performance with linguistic features was almost always higher than that one with TF-IDF under comparable conditions (the difference between the two experiments is only text representation technique). In the case of TF-IDF features, most classifiers predicted the dominant class for the vast majority of the instances, resulting in relatively low values for precision and recall, i.e., all classifiers had difficulty predicting the correct class labels.

Due to the higher number of labeled tickets, Data2 revealed an expected systematic classification quality increase compared to Data1, independently of applied algorithm or text representation technique.

When enhancing kNN with a self-training, we expected a noticeable increase in performance quality. Nonetheless, the evaluation results evidenced no improvement.

As stated in the Experimental settings subsection, using logistic regression, we identified the most predictive features for Data1 and Data2. According to the weights of logistic regression, in the case of Data 1, the five most important features are 1) relative occurrence of cognitive words; 2) occurrence of unique adjectives in all adjectives, 3) wording style, 4) occurrence of unique verbs in all verbs; 5) relative occurrence of words with positive sentiment. In the case of Data2, the five most important features are 1) relative occurrence of cognitive words; 2) occurrence of unique adjectives in all adjectives, 3) occurrence of unique verbs in all verbs; 4) relative occurrence of words with negative and 5) positive sentiments. As can be concluded, the most essential feature appeared to be a relative occurrence of cognitive words, further referred to as the *"cognitive words" feature*. After that, we trained the classifiers with the selected linguistic features. The results are summarized in Table 7.

As can be seen in Tables 6 and 7, the usage of our linguistic features delivers excellent performance with simple algorithms, such as decision trees and naïve Bayes. Both of them are statistically significantly better than other classifiers. Additionally, we have shown that selecting the best set of features improves the performance of classifiers consistently. Hence, using a smaller set of features also simplifies the process of extraction and reduces computational costs.

As mentioned above and according to the weights of logistic regression, the most important feature in the ticket text appears to be the *"cognitive words" feature*. To confirm this observation, we tested the algorithms' performance without this feature (see Table 8). In the case of exclusion of the most predictive feature, we see a statistically significant decrease in terms of prediction quality both for decision trees and naïve Bayes. In the case of classifiers that treat all features equally important (e.g., kNN and its variants), the accuracy was generally low, and we did not observe differences when excluding the *"cognitive words" feature*.

Furthermore, we experimented with a statistical method, i.e., the expert decision rule based on the occurrence of the most predictive "*cognitive words feature*" in a ticket text denoted as *COG*. Our prediction ($\hat{y}$) is based on our already mentioned previous research [25], and it is determined as

TABLE 7. Evaluation results using the five best performing linguistic features.

Algorithm	Accuracy	Average precision	Average recall	F-score
Data1: five best linguistic features				
standard kNN	0.667	0.589	0.585	0.587
kNN self-training	0.667	0.589	0.585	0.587
kNN self-training & resampling	0.667	0.589	0.585	0.587
ECkNN	0.667	0.578	0.580	0.579
HFNN	0.633	0.429	0.515	0.468
HWkNN	0.667	0.589	0.585	0.587
NHBNN	0.433	0.144	0.333	0.202
decision trees	1.000	1.000	1.000	1.000
naïve Bayes	1.000	1.000	1.000	1.000
logistic regression	0.633	0.611	0.580	0.595
SVM	0.967	0.976	0.970	0.973
QuickSUCCESS	0.733	0.814	0.636	0.714
Data2: five best linguistic features				
standard kNN	0.817	0.711	0.670	0.690
kNN self-training	0.817	0.711	0.670	0.690
kNN self-training & resampling	0.700	0.233	0.333	0.275
ECkNN	0.750	0.523	0.558	0.539
HFNN	0.800	0.526	0.582	0.553
HWkNN	0.817	0.711	0.670	0.690
NHBNN	0.700	0.233	0.333	0.275
decision trees	1.000	1.000	1.000	1.000
naïve Bayes	1.000	1.000	1.000	1.000
logistic regression	0.850	0.554	0.606	0.579
SVM	0.983	0.958	0.970	0.962
QuickSUCCESS	0.867	0.791	0.693	0.739

follows:

$$\hat{y} = \begin{cases} low, & \text{if } COG = 0 \\ medium, & \text{if } 0 < COG < 0.3 \\ high, & \text{otherwise} \end{cases}$$

This expert rule is competitive with some of the examined classifiers. However, its performance is significantly worse than that of decision trees and naïve Bayes (see Table 9).

In the case of kNN, we also tested other k values, in particular, all odd numbers between 1 and 10 and different distance functions, such as Euclidian, Manhattan, and cosine. As a result, we did not observe any substantial differences (see Figure 3).

V. DISCUSSION

In this study, our focus was to gain a better understanding of the factors influencing the quality of prediction in text classification tasks. At the same time, we addressed (i) the need for further experiments in industrial settings, especially with recent classifiers that have not been used for ticket classification before, and (ii) limitations of our rule-based approach.

TABLE 8. Evaluation results using linguistic features without *"cognitive words"* feature.

Algorithm	Accuracy	Average precision	Average recall	F-score
Data1: linguistic features except for the "cognitive words" feature				
standard kNN	0.533	0.526	0.498	0.511
kNN self-training	0.533	0.526	0.498	0.511
kNN self-training & resampling	0.533	0.526	0.498	0.511
ECkNN	0.633	0.599	0.580	0.589
HFNN	0.600	0.411	0.490	0.447
HWkNN	0.533	0.526	0.498	0.511
NHBNN	0.433	0.144	0.333	0.202
decision trees	0.233	0.188	0.189	0.188
naïve Bayes	0.467	0.409	0.483	0.443
logistic regression	0.367	0.274	0.296	0.284
SVM	0.633	0.610	0.610	0.610
QuickSUCCESS	0.523	0.523	0.523	0.523
Data2: linguistic features except for the "cognitive words" feature				
standard kNN	0.750	0.593	0.576	0.584
kNN self-training	0.750	0.593	0.576	0.584
kNN self-training & resampling	0.750	0.593	0.576	0.584
ECkNN	0.733	0.586	0.568	0.577
HFNN	0.733	0.539	0.528	0.534
HWkNN	0.750	0.593	0.576	0.584
NHBNN	0.700	0.233	0.333	0.275
decision trees	0.717	0.538	0.560	0.548
naïve Bayes	0.400	0.379	0.409	0.393
logistic regression	0.767	0.504	0.544	0.523
SVM	0.750	0.529	0.536	0.533
QuickSUCCESS	0.767	0.603	0.584	0.593

TABLE 9. Evaluation results using expert decision rule.

Data	Accuracy	Average precision	Average recall	F-score
Data1	0.633	0.451	0.667	0.538
Data2	0.883	0.537	0.667	0.595

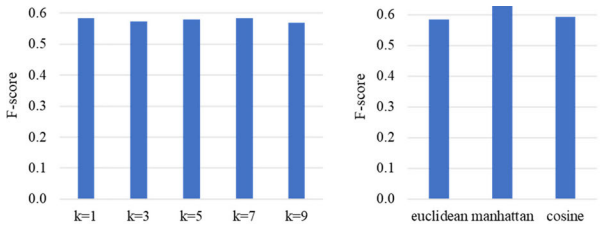


FIGURE 3. kNN performance with various k values (in the left, using Euclidean distance) and distance functions (in the right, k=1) on Data2.

As stated at the beginning of the study, since Artificial Intelligence (AI) becomes an increasing part of our business and daily lives, we recognize the paramount importance of the explainability of AI-based solutions. The need to understand the decisions made by an AI system is crucial in the context of trust and primarily to efficiently address the errors made by such a system. In our work, we closely study the ITIL Change Management based IT ticketing process of a big telecommunication company. Implementing the changes in the IT infrastructure means intervening in the organization's IT environment, its functions, and experience every time the change is performed. Below, we review the study findings in light of the experimental results.

Our study findings show that feature selection is an important component of ticket classification pipelines, not only to reduce the dimensionality of the data and computational costs but also to increase the performance of classifiers. We demonstrate that five linguistic features we identified based on the weights of logistic regression are enough to deliver accurate predictions. While comparing the algorithms' performance with and without the most important feature *relative occurrence of cognitive words*, we evidence that this feature undoubtedly influences the prediction results. In the case of kNN classifier, when experimenting with different values of k and distance functions, we also show that their choice does not significantly impact the prediction, especially when comparing with the choice of features (TF-IDF vs. linguistic features).

As there is a general discussion on the advantages and disadvantages of various text representation techniques and classification algorithms, in our paper, we offer a new analysis of both design decisions (text representation and classifier) and their interaction. First, we systematically compared the efficiency of linguistic features and TF-IDF representations in the context of IT ticket complexity prediction. To the best of our knowledge, this is the first attempt of such a research setting. Second, using different datasets and text representation techniques, we consistently tested twelve ML classifiers and showed that simple algorithms, such as decision trees and naïve Bayes, achieved accurate prediction with the linguistic features representation. Hereby, the five best performing features delivered results of the same quality. Hence, building explainable text classification pipelines, i.e., using linguistic features with simple algorithms, can have great potential when applied in real-world tasks.

Our study offers valuable insights for managers and ML experts by enabling them to understand the interdependencies or their absence between selected text representation techniques, classification algorithms, and prediction quality.

The most remarkable practical implication of this study is improving our rule-based approach. We used the described linguistic features approach and both handcrafted and decision trees-based rules to predict the IT ticket complexity value of low, medium, or high [25]. Using the ML classification pipeline discussed in the paper, we managed to improve prediction quality significantly without the need to define and constantly update the rules with the experts, which is a big hurdle in the management, maintenance, and scalability of such systems.

In real-life scenarios, IT ticket processing teams continuously undergo changes and fluctuations, causing the resolver groups to be split, merged, or reorganized. Due to the fast technological developments, the problems they have to solve also change. Hence, the trained machine learning model can become outdated very fast and will demand retraining [97]. This again justifies our message – the simpler, the better, i.e., we recommend using those classifiers which do not demand complex parameter search and are simple and cheap to implement if their prediction quality is acceptable.

To sum up, systematic testing of the discussed representation techniques and classification algorithms and their application for the IT ticket classification task of complexity prediction can be of interest for text data scientists and managers who seek to understand the role of factors influencing the prediction.

VI. CONCLUSION

Our work aimed to provide a comparative analysis of text representation techniques and classifiers while developing an IT ticket classification pipeline. Our observations can be useful for the design of decision support systems in enterprise applications, specifically in the IT ticket area.

The contributions of our work can be summarized as follows: (i) our comprehensive comparative analysis of linguistic features with TF-IDF and various ML algorithms confirms the positive influence of linguistic style predictors [52] on the prediction quality; (ii) our observation that simple algorithms work well if using appropriate linguistic features contributes to the general discussion on the advantages and disadvantages of various text classification algorithms [13], [14]; (iii) we showed that ML-based IT ticket classification outperforms our rule-based approach.

As a part of future work, one can: (i) consider further information regarding IT ticket complexity prediction, such as the number of tasks and configuration items per ticket; (ii) test other application cases in the IT ticket area and beyond, i.e., further explore the potential of linguistic features; (iii) as we showed that selecting an appropriate subset of linguistic features can considerably improve the performance of classifiers, one may conduct further experiments with more advanced feature selection techniques [98].

APPENDIX

List of attached Appendices:

Appendix I. Taxonomy of Decision-Making Logic Levels

Appendix II. Business Sentiment Lexicon with assigned valences

APPENDIX I. Taxonomy of decision-making logic levels. following [24], we consider diverse semantic concepts: *Resources, Techniques, Capacities*, and *Choices*, elements of RTCC framework. We designed contextual variables [99], based on which experts categorized words into one of the three DML levels and one of the four semantic concepts.

CONTEXTUAL VARIABLES	DECISION-MAKING LOGIC LEVELS		
	routine	*semi-cognitive*	*cognitive*
	CONCEPTUAL ASPECTS		
	RESOURCES		
Problem Processing Level	user, user request, task, test, check, target, release, contact role, access, interface, cluster, tool, client, file system, partner, node	team, leader, project, colleague, property	management, system, CAB, measure, approval
Accuracy	time, application, product, configuration item, CI, right, instance, machine, minute, hour, day, week, detail, description	description, environment, requirement, validity, reason, solution, method, problem, rule, modification	
Situational Awareness	name, password, group, directory, number, email, package, phone, ID, IP, attachment	request for change, RfC, customer, rollout, backout	server farm
Information	server, file, location, dataset, network, data, patch, port, information, type, root, certificate, account, device, cable, parameter, agent, folder, disk, fallback, database, db, backup, version, tool, firewall, system, hotfix, supervisor, reference, instruction, format	Requestor, software, downtime, production, power-supply, outage, service, case	risk, freeze, impact
	TECHNIQUES		
Experience	need, see, deploy, document, monitor, use, follow, note, provide, test, contain, accompany, inform, consist, describe	implement, create, support, require, classify	approve, delegate, propose
Action Choice	start, finish, monitor, import, export, run, stop, step, end, put, send, switch, install, reject, update, upgrade, include, replace, remove, move, begin, make, get, migrate, open, initialize, revoke	deploy, migrate, process, modify, forget, increase, miss	freeze
Effort	cancel, rundown, decommission, restart, delete, set, add, activate, reboot, specify, agree upgrade, mount, execute, transfer, write, find	perform, modify, assign, check, need, expect, verify	define
	CAPACITIES		
Specificity	additional, preapproved, affected, initial, attached, internal, external, reachable, regular, active, scheduled, next, whole, formal, virtual, wrong, individual, administrative, local	secure, separate, specific, technical, urgent, separate, corrected, minor, normal	related, multiple, multi-solution, major, high, small, big
Decisions Formulation	new, old, preinstalled, fixed, ready, following, current, valid, primary, necessary	available, necessary, important, significant, successful, appropriate, relevant, main, further, responsible	possible, many, desired, different, various
Predictability	actual, full, online, standard, responsible, administrative, existing, minimum, same, visible	strong, temporary, offline, previous, last, other, more, much, similar, standard	random, strong randomized, encrypted, expected
	CHOICES		
Precision	automatically, instead, manually, there, where, here, separately, additionally, internally	normally, newly, shortly, urgently, temporarily	maybe, randomly, likely
Scale	permanently, currently, still, now, often, never, already, just, always, yet, anymore, firstly, before, together, daily, meanwhile, really, furthermore, afterwards, therefore	again, later, however, usually, previously, recently	soon
Ambiguity	correctly, therefore, accordingly, actually, consequently, completely, simultaneously, anyway, necessarily	well, enough, immediately, easily, simply	approximately, properly

APPENDIX II. Business sentiment lexicon with assigned valences.

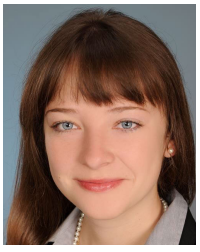
Tickets	ITIL	Valence
Expressions		
no risk, no outage		+2
be so kind, would be nice		0.5
disaster recovery, set alarms warnings, poison attack vulnerability, critical security leaks, fan, outstanding windows updates, thank you, kind regards, would like, best regards	request for change, RfC	0
big measure	projected service outage, change advisory board, high impact, major change	-0.5
Single keywords		
kind, success, correct, like, nice	well, successful, happy	0.5
disaster, recovery, affected, stop, disable, dump, alarm, warning, poison, attack, vulnerability, error, prevent, drop, cancel, delete, exclude, problem, problems, faulty, failed, destroy, defective, obsolete, lack, security, leak, crash, please, support, optimize, grant, privilege, create, dear, acceptance, clarity, restore, increase, danger, balance, right, deny, wrong, retire, missing, weak, invalid, see, follow, yes, allow, approve, approval, confirm	problem, failed, information, operational, identify, order, include, adequately, procedure, necessary, assess, criteria, clear, provide, potentially, identification, adequate, initiate, value, KPI, standard, schedule, align, properly, release, accurate, report, organization, continuous, ensure, service, beneficial, stakeholder, requirement, correct, record, essential, clearly, RfC, support, tool, relevant, attempt, subsequently, configuration, different, follow, directly, CI, potential, request, individual, plan, work, evaluate, author, organizational, manage, number, financial, status, low, chronological, recommend, responsible, model, accountable, handle, timescale, business, normal, submit, update, create, manual, consider, backout, accept, item, project, deliver, formal, data, iterative, produce, local, describe, test, improve, result, deployment, deploy, technical, management, repeatable, determine, minimum, develop, appropriate, activate, implement, require, process, evaluation, customer, contractual, authorize, share, acceptable	0
blocked, critical	cost, PSO, CAB, important, unauthorized, major, significant, undesirable, incomplete, delegate, avoid, coordinate, immediately, significantly	-0.5
offline, risk, outage, emergency, downtime	impact, risk, emergency, incident, outage, downtime	-1
rejected	unacceptable	-2

REFERENCES

[1] Y. Diao and K. Bhattacharya, "Estimating business value of IT services through process complexity analysis," in *Proc. IEEE Netw. Oper. Manage. Symp. (NOMS)*. Salvador, Brazil, 2008.
[2] S. P. Paramesh and K. S. Shreedhara, *Automated IT Service Desk Systems Using Machine Learning Techniques* (Lecture Notes in Networks and Systems), vol. 43. Singapore: Springer, 2019, pp. 331–346.
[3] S. P. Paramesh, C. Ramya, and K. S. Shreedhara, "Classifying the unstructured IT service Des. tickets using ensemble of classifiers," in *Proc. 3rd Int. Conf. Comput. Syst. Inf. Technol. Sustain. Solutions (CSITSS)*, Dec. 2018, pp. 221–227.
[4] S. Roy, D. P. Muni, J. J. Y. T. Yan, N. Budhiraja, and F. Ceiler, *Clustering and Labeling IT Maintenance Tickets*. Berlin, Germany: Springer, 2016.
[5] G. B. Dasgupta, T. K. Nayak, A. R. Akula, S. Agarwal, and S. J. Nadgowda, *Towards Auto-Remediation in Services Delivery: Context-Based Classification of Noisy and Unstructured Tickets*. Berlin, Germany: Springer, 2014.
[6] S. Agarwal, R. Sindhgatta, and B. Sengupta, "SmartDispatch: Enabling efficient ticket dispatch in an IT service environment," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, vol. 2012, pp. 1393–1401.
[7] S. Agarwal, V. Aggarwal, A. R. Akula, G. B. Dasgupta, and G. Sridhara, "Automatic problem extraction and analysis from unstructured text in IT tickets," *IBM J. Res. Develop.*, vol. 61, no. 1, pp. 41–52, Jan. 2017.
[8] E. Orta, M. Ruiz, N. Hurtado, and D. Gawn, "Decision-making in IT service management: A simulation based approach," *Decis. Support Syst.*, vol. 66, pp. 36–51, Oct. 2014.
[9] M. Ruiz, J. Moreno, B. Dorronsoro, and D. Rodriguez, "Using simulation-based optimization in the context of IT service management change process," *Decis. Support Syst.*, vol. 112, pp. 35–47, Aug. 2018.
[10] E. Fielt, T. Böhmann, A. Korthaus, S. Conger, and G. Gable, "Service management and engineering in information systems research," *J. Strategic Inf. Syst.*, vol. 22, no. 1, pp. 46–50, Mar. 2013.

[11] T. R. Eikebrokk and J. Iden, "Strategising IT service management through ITIL implementation: Model and empirical test," *Total Qual. Manage. Bus. Excellence*, vol. 28, nos. 3–4, pp. 238–265, Feb. 2017.
[12] R. D. Galliers, "Towards a flexible information architecture: Integrating business strategies, information systems strategies and business process redesign," *Inf. Syst. J.*, vol. 3, no. 3, pp. 199–213, Jul. 1993.
[13] Kowsari, J. Meimandi, Heidarysafa, Mendu, Barnes, and Brown, "Text classification algorithms: A survey," *Information*, vol. 10, no. 4, p. 150, Apr. 2019.
[14] Y. C. Cavalcanti, P. A. da Mota Silveira Neto, I. D. C. Machado, T. F. Vale, E. S. de Almeida, and S. R. D. L. Meira, "Challenges and opportunities for software change request repositories: A systematic mapping study," *J. Softw., Evol. Process*, vol. 26, no. 7, pp. 620–653, Jul. 2014.
[15] A. Lazarov and P. Shoval, "A rule-based system for automatic assignment of technicians to service faults," *Decis. Support Syst.*, vol. 32, no. 4, pp. 343–360, Mar. 2002.
[16] J. Kanwal and O. Maqbool, "Bug prioritization to facilitate bug report triage," *J. Comput. Sci. Technol.*, vol. 27, no. 2, pp. 397–412, Mar. 2012.
[17] T. Menzies and A. Marcus, "Automated severity assessment of software defect reports," in *Proc. IEEE Int. Conf. Softw. Maintenance*, Sep. 2008, pp. 346–355.
[18] S. N. Ahsan and F. Wotawa, "Impact analysis of SCRs using single and multi-label machine learning classification," in *Proc. ACM-IEEE Int. Symp. Empirical Softw. Eng. Meas. (ESEM)*, 2010, pp. 1–4.
[19] S. N. Ahsan, J. Ferzund, and F. Wotawa, "Automatic classification of software change request using multi-label machine learning methods," in *Proc. 33rd Annu. IEEE Softw. Eng. Workshop*, Oct. 2009, pp. 79–86.
[20] V. Rus, X. Nan, S. Shiva, and Y. Chen, "Clustering of defect reports using graph partitioning algorithms," in *Proc. 21st Int. Conf. Softw. Eng. Knowl. Eng. (SEKE)*, 2009, pp. 442–445.
[21] A. Santana, J. Silva, P. Muniz, F. Araújo, and R. M. C. R. de Souza, *Comparative Analysis of Clustering Algorithms Applied to the Classification of Bugs*. Berlin, Germany: Springer, 2012.
[22] N. Rizun, A. Revina, and V. Meister, "Discovery of stylistic patterns in business process textual descriptions: IT ticket case," in *Proc. Conf., 33rd Int. Bus. Inf. Manage. Assoc. Conf. (IBIMA)*, 2019, pp. 2103–2113.
[23] N. Rizun and A. Revina, "Business sentiment analysis. Concept and method for perceived anticipated effort identification," in *Proc. Inf. Syst. Develop., Inf. Syst. Beyond (ISD)*, A. Siarheyeva, C. Barry, M. Lang, H. Linger, and C. Schneider, Eds. Toulon, France: ISEN Yncréa Méditerranée, 2019. [Online]. Available: https://aisel.aisnet.org/isd2014/proceedings2019/ManagingISD/3/
[24] N. Rizun, A. Revina, and V. Meister, "Method of decision-making logic discovery in the business process textual data," in *Proc. Int. Conf. Bus. Inf. Syst. (BIS)*, Seville, Spain. Cham, Switzerland: Springer, 2019, pp. 70–84.
[25] A. Revina and N. Rizun, "Multi-criteria knowledge-based recommender system for decision support in complex business processes," in *Proc. Workshop Workshop Recommendation Complex Scenarios Co-Located 13th ACM Conf. Recommender Syst. (RecSys)*, 2019, pp. 16–22.
[26] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Inf. Process. Manage.*, vol. 24, no. 5, pp. 513–523, Jan. 1988.
[27] N. Tomašev and K. Buza, "Hubness-aware kNN classification of high-dimensional data in presence of label noise," *Neurocomputing*, vol. 160, pp. 157–172, Jul. 2015.
[28] N. Tomašev, K. Buza, K. Marussy, and P. B. Kis, "Hubness-aware classification, instance selection and feature construction: Survey and extensions to time-series," in *Feature Selection for Data and Pattern Recognition*, U. Stańczyk and L. C. Jain, Eds. Berlin, Germany: Springer, 2015, pp. 231–262.
[29] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, Mar. 1986.
[30] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge, U.K.: Cambridge Univ. Press, 2009, pp. 253–286.
[31] D. W. Hosmer, S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*, vol. 398. Hoboken, NJ, USA: Wiley, 2013.
[32] T. Joachims, *Text Categorization With Support Vector Machines: Learning With Many Relevant Features*. Berlin, Germany: Springer, 1998, pp. 137–142.
[33] I. Triguero, S. García, and F. Herrera, "Self-labeled techniques for semi-supervised learning: Taxonomy, software and empirical study," *Knowl. Inf. Syst.*, vol. 42, no. 2, pp. 245–284, Feb. 2015.
[34] K. Marussy and K. Buza, "SUCCESS: A new approach for semi-supervised classification of time-series," in *Proc. Int. Conf. Artif. Intell. Soft Comput.*, 2013, pp. 437–447.
[35] K. Buza and A. Revina, "Speeding up the SUCCESS approach for massive industrial datasets," in *Proc. Int. Conf. Innov. Intell. Syst. Appl. (INISTA)*, Aug. 2020, pp. 1–6.
[36] S. Scott and S. Matwin, "Text classification using WordNet Hypernyms," in *Proc. Usage WordNet Natural Language Process. Syst.*, 1998, pp. 45–51.
[37] J. Yan, "Text representation," in *Encyclopedia of Database Systems*. Boston, MA, USA: Springer, 2009, pp. 3069–3072.
[38] Y. Zhang, R. Jin, and Z.-H. Zhou, "Understanding bag-of-words model: A statistical framework," *Int. J. Mach. Learn. Cybern.*, vol. 1, nos. 1–4, pp. 43–52, Dec. 2010.
[39] K. J. Sparck, "A statistical interpretation of term specificity and its application in retrieval," *J. Document.*, vol. 28, no. 1, pp. 11–21, Jan. 1972.
[40] M. Radovanović and M. Ivanović, "Text mining: Approaches and applications," *Novi Sad J. Math*, vol. 38, no. 3, pp. 227–234, 2008.
[41] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*. [Online]. Available: http://arxiv.org/abs/1301.3781
[42] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," 2014, *arXiv:1405.4053*. [Online]. Available: http://arxiv.org/abs/1405.4053
[43] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1532–1543.
[44] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, "FastText.Zip: Compressing text classification models," 2016, *arXiv:1612.03651*. [Online]. Available: http://arxiv.org/abs/1612.03651
[45] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," 2016, *arXiv:1607.04606*. [Online]. Available: http://arxiv.org/abs/1607.04606
[46] O. Melamud, J. Goldberger, and I. Dagan, "Context2vec: Learning generic context embedding with bidirectional LSTM," in *Proc. 20th SIGNLL Conf. Comput. Natural Lang. Learn.*, 2016, pp. 51–61.
[47] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," 2018, *arXiv:1802.05365*. [Online]. Available: http://arxiv.org/abs/1802.05365
[48] S. Mahdi Rezaeinia, A. Ghodsi, and R. Rahmani, "Improving the accuracy of pre-trained word embeddings for sentiment analysis," 2017, *arXiv:1711.08609*. [Online]. Available: http://arxiv.org/abs/1711.08609
[49] F. Yin, Y. Wang, J. Liu, and L. Lin, "The construction of sentiment lexicon based on context-dependent part-of-speech chunks for semantic disambiguation," *IEEE Access*, vol. 8, pp. 63359–63367, 2020.
[50] A. Sureka and K. V. Indukuri, "Linguistic analysis of bug report titles with respect to the dimension of bug importance," in *Proc. 3rd Annu. ACM Bengaluru Conf. (COMPUTE)*, 2010, pp. 1–6.
[51] A. J. Ko, B. A. Myers, and D. Horng Chau, "A linguistic analysis of how people describe software problems," in *Proc. Vis. Lang. Human-Centric Comput. (VL/HCC)*, 2006, pp. 127–134.
[52] K. Coussement and D. Van den Poel, "Improving customer complaint management by automatic email classification using linguistic style features as predictors," *Decis. Support Syst.*, vol. 44, no. 4, pp. 870–882, Mar. 2008.
[53] J. Fürnkranz, T. Mitchell, and E. Riloff, "Case study in using linguistic phrases for text categorization on the WWW," Assoc. Advancement Artif. Intell., Palo Alto, CA, USA, Workshop Ser. Tech. Rep. WS-98-05, 1998.
[54] D. Mladenic and M. Grobelnik, "Word sequences as features in text-learning," in *Proc. 17th Electrotech. Comput. Sci. Conf. (ERK)*, 1998, pp. 145–148.
[55] B. Raskutti, H. Ferrá, and A. Kowalczyk, "Second order features for maximising text classification performance," in *Proc. Eur. Conf. Mach. Learn.*, 2001, pp. 419–430.
[56] R. Bekkerman, R. El-Yaniv, N. Tishby, and Y. Winter, "On feature distributional clustering for text categorization," in *Proc. 24th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, 2001, pp. 146–153.
[57] C.-M. Tan, Y.-F. Wang, and C.-D. Lee, "The use of bigrams to enhance text categorization," *Inf. Process. Manage.*, vol. 38, no. 4, pp. 529–546, Jul. 2002.
[58] S. Scott and S. Matwin, "Feature engineering for text classification," in *Proc. ICML*, 1999, pp. 379–388.

[59] A. Moschitti and R. Basili, "Complex linguistic features for text classification: A comprehensive study," in *Proc. Eur. Conf. Inf. Retr.*, 2004, pp. 181–196.
[60] M. Sasaki and K. Kita, "Rule-based text categorization using hierarchical categories," in *Proc. Conf. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 1998, pp. 2827–2830.
[61] J. Mason, Qualitative researching. Sage Publications Ltd, 2002.
[62] J. Seidel and U. Kelle, *Computer-Aided Qualitative Data Analysis: Theory, Methods and Practice*. London, U.K.: SAGE, 1995.
[63] S. Chua, F. Coenen, and G. Malcolm, "Classification inductive rule learning with negated features," in *Proc. 6th Int. Conf. Adv. Data Mining Appl. (ADMA)*, 2010, pp. 125–136.
[64] L. Rokach, "Decision forest: Twenty years of research," *Inf. Fusion*, vol. 27, pp. 111–125, Jan. 2016.
[65] E. García, C. Romero, S. Ventura, and T. Calders, "Drawbacks and solutions of applying association rule mining in learning management systems" in *Proc. Int. Workshop Applying Data Mining e-Learning*, 2007, pp. 13–22.
[66] G. Kaur, "Association rule mining: A survey," *Int. J. Comput. Sci. Inf. Technol.*, vol. 5, no. 2, pp. 2320–2324, 2014.
[67] H. Hu and J. Li, "Using association rules to make rule-based classifiers robust," in *Proc. 16th Austral. Database Conf.*, 2005, pp. 47–54.
[68] V. Chakravarthy, S. Joshi, G. Ramakrishnan, S. Godbole, and S. Balakrishnan, "Learning decision lists with known rules for text mining," in *Proc. 3rd Int. Joint Conf. Natural Lang. Process.*, 2008, pp. 835–840.
[69] T. M. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill, 1997.
[70] O. Uzuner, X. Zhang, and T. Sibanda, "Machine learning and rule-based approaches to assertion classification," *J. Amer. Med. Inform. Assoc.*, vol. 16, no. 1, pp. 109–115, Jan. 2009.
[71] X. Zhu, *Semi-Supervised Learning Literature Survey*. New York, NY, USA: Wisconsin, Jul. 2008.
[72] E. Frank and R. R. Bouckaert, "Naive Bayes for text classification with unbalanced classes," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (LNAI), vol. 4213. 2006, pp. 503–510.
[73] W. Daelemans, "Explanation in computational stylometry," in *Proc. Int. Conf. Intell. Text Process. Comput. Linguistics*, 2013, pp. 451–462.
[74] *Taxonomy Strategies | Bibliography–Taxonomy Strategies*. Accessed: Sep. 2, 2019. [Online]. Available: https://taxonomystrategies.com/library/bibliography/
[75] A. Blumauer. *Taxonomies and Ontologies | LinkedIn Blog Article*. Accessed: Sep. 2, 2019. [Online]. Available: https://www.linkedin.com/pulse/taxonomies-ontologies-andreas-blumauer/
[76] D. M. Blei, "Probabilistic topic models," *Commun. ACM*, vol. 55, no. 4, pp. 77–84, Apr. 2012.
[77] B. Liu, "Sentiment analysis and opinion mining," *Synth. Lectures Hum. Lang. Technol.*, vol. 5, no. 1, pp. 1–184, 2012.
[78] C. J. Hutto and E. Gilbert, "VADER: A parsimonious rule-based model for sentiment analysis of social media text," in *Proc. 8th Int. Conf. Weblogs Social Media (ICWSM)*, 2014, pp. 1–10.
[79] G. K. Zipf, *Selected Studies of the Principle of Relative Frequency in Language*. Cambridge, MA, USA: Harvard Univ. Press, 1932.
[80] J. Shao and D. Tu, *The Jackknife and Bootstrap*. New York, NY, USA: Springer, 1995.
[81] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants," *Mach. Learn.*, vol. 36, nos. 1–2, pp. 105–139, 1999.
[82] L. Firte, C. Lemnaru, and R. Potolea, "Spam detection filter using KNN algorithm and resampling," in *Proc. IEEE 6th Int. Conf. Intell. Comput. Commun. Process.*, Aug. 2010, pp. 27–33.
[83] V. Ng and C. Cardie, "Weakly supervised natural language learning without redundant views," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics Human Lang. Technol. (NAACL)*, 2003, pp. 173–180.
[84] K. Hara, I. Suzuki, M. Shimbo, K. Kobayashi, K. Fukumizu, and M. Radovanovic, "Localized centering: Reducing hubness in large-sample data," in *Proc. 29th AAAI Conf. Artif. Intell.*, B. Bonet and S. Koenig, Eds. Austin, TX, USA: AAAI Press, Jan. 2015, pp. 2645–2651. [Online]. Available: http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9898
[85] K. Buza and D. Neubrandt, "A new proposal for person identification based on the dynamics of typing: Preliminary results," *Theor. Appl. Informat.*, vol. 28, nos. 1–2, pp. 1–12, Feb. 2017.
[86] K. Buza, A. Nanopoulos, and G. Nagy, "Nearest neighbor regression in the presence of bad hubs," *Knowl.-Based Syst.*, vol. 86, pp. 250–260, Sep. 2015.
[87] M. Radovanović, A. Nanopoulos, and M. Ivanović, "Nearest neighbors in high-dimensional data," in *Proc. 26th Annu. Int. Conf. Mach. Learn. (ICML)*, 2009, pp. 1–8.
[88] N. Tomašev, M. Radovanović, D. Mladenić, and M. Ivanović, "Hubness-based fuzzy measures for high-dimensional k-nearest neighbor classification," *Int. J. Mach. Learn. Cybern.*, vol. 5, no. 3, pp. 16–30, 2011.
[89] N. Tomasev, M. Radovanović, D. Mladenić, and M. Ivanović, "A probabilistic approach to nearest-neighbor classification: Naive Hubness Bayesian kNN," in *Proc. 20th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2011, pp. 2173–2176.
[90] T. L. Saaty, "The analytic hierarchy and analytic network processes for the measurement of intangible criteria and for decision-making," in *Multiple Criteria Decision Analysis: State of the Art Surveys*. New York, NY, USA: Springer, 2005, pp. 345–405.
[91] G. I. Webb, "Leave-one-out cross-validation," in *Encyclopedia of Machine Learning*. Boston, MA, USA: Springer, 2011, pp. 600–601.
[92] S. L. Salzberg, "On comparing classifiers: Pitfalls to avoid and a recommended approach," *Data Mining Knowl. Discovery*, vol. 1, no. 3, pp. 317–328, 1997.
[93] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana, "Fast time series classification using numerosity reduction," in *Proc. 23rd Int. Conf. Mach. Learn. (ICML)*, 2006, pp. 1033–1040.
[94] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967.
[95] A. Y. Ng, "Feature selection, l1 vs. l2 regularization, and rotational invariance," in *Proc. 21st Int. Conf. Mach. Learn. (ICML)*, 2004, pp. 615–622.
[96] Q. Cheng, P. K. Varshney, and M. K. Arora, "Logistic regression for feature selection and soft classification of remote sensing data," *IEEE Geosci. Remote Sens. Lett.*, vol. 3, no. 4, pp. 491–494, Oct. 2006.
[97] A. Mandal, N. Malhotra, S. Agarwal, A. Ray, and G. Sridhara, "Automated Dispatch of Helpdesk Email Tickets: Pushing the Limits with AI," *Proc. AAAI Conf. Artif. Intell.*, vol. 33, no. 01, pp. 9381–9388, Jul. 2019.
[98] A. Szenkovits, R. Meszlényi, K. Buza, N. Gaskó, R. Ioana Lung, and M. Suciu, "Feature selection with a genetic algorithm for classification of brain imaging data," in *Advances in Feature Selection for Data and Pattern Recognition*, vol. 138, U. Stanczyk, B. Zielosko, and L. C. Jain, Eds. Cham, Switzerland: Springer, 2018, pp. 185–202.
[99] N. Rizun and Y. Taranenko, "Simulation models of human decision-making processes," *Manage. Dyn. Knowl. Economy*, vol. 2, no. 2, pp. 241–264, 2014.



ALEKSANDRA REVINA received the M.Sc. degree in business administration with a focus on business process and knowledge management and engineering from the Brandenburg University of Applied Sciences, in 2015. Afterward, she worked as a Research Scientist and a Project Manager with the Deutsche Telekom Innovation Laboratories, Berlin. In early 2018, she started her industrial Ph.D. at the Technical University of Berlin and the Brandenburg University of Sciences (cooperative procedure). Her research interests include diverse methods and tools for business process analysis and automation with the goal of developing efficient decision support systems for process workers.

KRISZTIAN BUZA received the diploma degree in computer science from the Budapest University of Technology and Economics, in 2007, and the Ph.D. degree from the University of Hildesheim, in 2011. He is the coauthor of more than 50 publications, including the best paper of the IEEE Conference on Computational Science and Engineering in 2010. He regularly serves as a Reviewer for renowned journals, such as *Neurocomputing*, and *Knowledge-Based Systems* or *Knowledge and Information Systems*. He is also a member of the program committee of international conferences, such as the Pacific-Asia Conference on Knowledge Discovery and Data Mining and the International Conference on Computational Collective Intelligence. His research interests include applied machine learning and data mining.

VERA G. MEISTER received the Ph.D. degree in mathematics from St. Petersburg State University, in 1988. She worked as a Researcher, a Lecturer, and a Project Manager in vocational training for more than 20 years. She joined the Brandenburg University of Applied Sciences in 2012 as a Full Professor for business information systems in the Department of Economics. Her research interests include knowledge engineering and information retrieval.