What Works Better? A Study of Classifying Requirements

Zahra Shakeri Hossein Abad*, Oliver Karras†, Parisa Ghazi‡, Martin Glinz‡, Guenther Ruhe*, Kurt Schneider†
*SEDS Lab, Department of Computer Science, University of Calgary, Calgary, Canada
{zshakeri, ruhe}@ucalgary.ca
†Software Engineering Group, Leibniz Universität Hannover, Hannover, Germany
{oliver.karras, kurt.schneider}@inf.uni-hannover.de
‡Department of Informatics, University of Zurich, Zurich, Switzerland
{ghazi, glinz}@ifi.uzh.ch

***Abstract*—Classifying requirements into functional requirements (FR) and non-functional ones (NFR) is an important task in requirements engineering. However, automated classification of requirements written in natural language is not straightforward, due to the variability of natural language and the absence of a controlled vocabulary. This paper investigates how automated classification of requirements into FR and NFR can be improved and how well several machine learning approaches work in this context. We contribute an approach for preprocessing requirements that standardizes and normalizes requirements before applying classification algorithms. Further, we report on how well several existing machine learning methods perform for automated classification of NFRs into sub-categories such as usability, availability, or performance. Our study is performed on 625 requirements provided by the OpenScience tera-PROMISE repository. We found that our preprocessing improved the performance of an existing classification method. We further found significant differences in the performance of approaches such as Latent Dirichlet Allocation, Biterm Topic Modeling, or Naïve Bayes for the sub-classification of NFRs.**

***Index Terms*—Functional and Non-Functional Requirements, Classification, Topic Modeling, Clustering, Naïve Bayes**

I. Introduction

In requirements engineering, classifying the requirements of a system by their kind into *functional requirements*, *quality requirements* and *constraints* (the latter two usually called *non-functional requirements*) [1] is a widely accepted standard practice today.

While the different kinds of requirements are known and well-described today [2], automated classification of requirements written in natural language into functional requirements (FRs) and the various sub-categories of non-functional requirements (NFRs) is still a challenge [3]. This is particularly due to the fact that stakeholders, as well as requirements engineers, use different terminologies and sentence structures to describe the same kind of requirements [4], [5]. The high level of inconsistency in documenting requirements makes automated classification more complicated and therefore error-prone.

In this paper, we investigate how automated classification algorithms for requirements can be improved and how well some of the frequently used machine learning approaches work in this context. We make two contributions. (1) We investigate if and to which extent an existing decision tree learning algorithm [6] for classifying requirements into FRs and NFRs can be improved by preprocessing the requirements with a set of rules for (automated) standardizing and normalizing the requirements found in a requirements specification. (2) We study how well several existing machine learning methods perform for automated classification of NFRs into sub-categories such as availability, security, or usability.

With this work, we address the *RE Data Challenge* posed by the 25th IEEE International Requirements Engineering Conference (RE'17).

II. Related Work

Software Requirements Specifications (SRSs) are written in natural language, with mixed statements of functional and non-functional requirements. There is a growing body of research studies that compare the effect of using manual and automatic approaches for classification of requirements [3], [7]. An efficient classification enables focused communication and prioritization of requirements [8]. Categorization of requirements allows filtering relevant requirements for a given important aspect. Our work is also closely related to the research on automatic classification of textual requirements.

Knauss and Ott [9] introduced a model of a socio-technical system for requirements classification. They evaluated their model in an industrial setting with a team of ten practitioners by comparing a manual, a semi-automatic, and a fully-automatic approach of requirements classification. They reported that a semi-automatic approach offers the best ratio of quality and effort as well as the best learning performance. Therefore, it is the most promising approach of the three.

Cleland-Huang et al. [10] investigated mining large requirements documents for non-functional requirements. Their results indicate that NFR-Classifier adequately distinguishes several types of NFRs. However, further work is needed to improve the results for some other NFR types such as 'look-and-feel'. Although their study is similar to ours as they trained a classifier to recognize a set of weighted indicator terms, indicative of each type of requirement, we used different classification algorithms and additionally assessed their precisions and recalls to compare their performance with each other.

Rahimi et al. [11] present a set of machine learning and data mining methods for automatically extracting quality concerns from requirements, feature requests, and online forums. Then, they generate a basic goal model from the requirements specification. Each concern is modeled as a softgoal. For attaching topics to softgoals they used an LDA approach to estimate the similarity between each requirement and the discovered topics. In addition, they used LDA to identify the best sub-goal placement for each of the unattached requirements. However, in this research, we used LDA as one of our approaches for classifying the non-functional requirements.

Naïve Bayes classifier is used in several studies[12], [13] for automatic classification of requirements. Therefore, we included Naïve Bayes in our study to be comparable with other classifiers.

III. The Challenge and Research Questions

A. Context and Data Set

The challenge put forward by the Data Track of RE'17 consists of taking a given data set and performing an automated RE task on the data such as tracing, identifying/classifying requirements or extracting knowledge. For this paper, we chose the task of automated classification of requirements.

The data set given for this task comes from the OpenScience tera-PROMISE repository[1]. It consists of 625 labeled natural language requirements (255 FRs and 370 NFRs). The labels classify the requirements first into FR and NFR. Within the latter category, eleven sub-categories are defined: (a) ten *quality requirement categories*: Availability (A), Look & Feel (LF), Maintainability (MN), Operability (O), Performance (PE), Scalability (SC), Security (SE), Usability (US), Fault Tolerance (FT), and Portability (PO); (b) one *constraint category*: Legal & Licensing (L). These labels constitute the ground truth for our investigations.

B. Research Questions

We frame the goal of our study in two research questions: *RQ1. How do grammatical, temporal and sentimental characteristics of a sentence affect the accuracy of classifying requirements into functional and non-functional ones?*

With this research question, we investigate whether our preprocessing approach, which addresses the aforementioned characteristics, has a positive impact on the classification into FRs and NFRs in terms of precision and recall.

RQ2. To what extent is the performance of classifying NFRs into sub-categories influenced by the chosen machine learning classification method?

With this research question, we study the effects of the chosen machine learning method on the precision and recall achieved when classifying the NFRs in the given data set into the sub-categories defined in the data set.

IV. Preprocessing of Requirements Specifications

In this section, we describe the preprocessing we applied to reduce the inconsistency of requirements specifications by leveraging rich sentence features and latent co-occurrence relations.

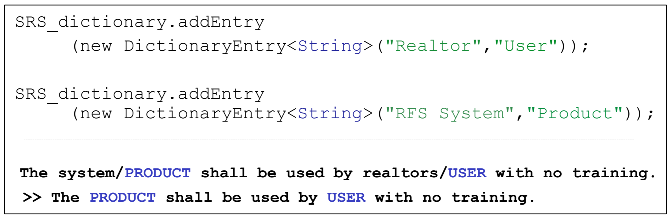
[1]https://terapromise.csc.ncsu.edu/!/#repo/view/head/requirements/nfr

A. Part Of Speech (POS) Tagging

We used the part-of-speech tagger of the Stanford Parser [14] to assign parts of speech, such as noun, verb, adjective, etc. to each word in each requirement. The POS tags[2] are necessary to perform the FR/NFR classification based on the approach of Hussain et al. [6].

B. Entity Tagging

To improve the generalization of input requirements, we used a "supervised training data" method in which all context-based products and users are blinded by assigning names as *PRODUCT* and *USER*, respectively. To this end, we used the LingPipe NLP toolkit[3] and created the $SRS\_dictionary$ by defining project specific users/customers and products (e.g., program administrators, nursing staff members, realtor, or card member marked as USER), such as below:



```
SRS_dictionary.addEntry
     (new DictionaryEntry<String>("Realtor","User"));

SRS_dictionary.addEntry
     (new DictionaryEntry<String>("RFS System","Product"));

The system/PRODUCT shall be used by realtors/USER with no training.
>> The PRODUCT shall be used by USER with no training.
```

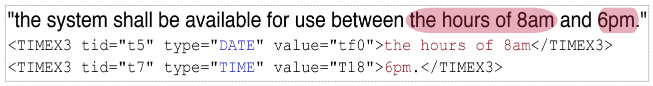
Then, each sentence is tokenized and POS tagged with the developed SRS dictionary. All of the tokens associated with *user* and *product* were discarded and we only kept these two keywords to represent these two entities. Finally, we used the POS tagger of the Stanford Parser [14] and replaced all Noun Phrases (NPs) including "USER" and "PRODUCT" with $USER$ and $PRODUCT$, respectively. For instance, registered USER in ⟨Only registered **USER** shall be able to access the **PRODUCT**⟩, is replaced with $USER$.

C. Temporal Tagging

Time is a key factor to characterize non-functional requirements, such as availability, fault tolerance, and performance.

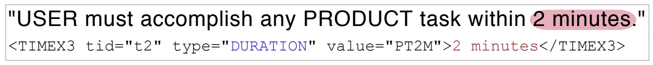
For this purpose, we used SUTime, a rule-based temporal tagger for recognizing and normalizing temporal expressions by TIMEX3 standard[4]. SUTIME detects the following basic types of temporal objects [15]:

1) *Time:* A particular instance on a time scale. SUTIME also handles absolute times, such as *Date*. As in:



```
"the system shall be available for use between the hours of 8am and 6pm."
<TIMEX3 tid="t5" type="DATE" value="tf0">the hours of 8am</TIMEX3>
<TIMEX3 tid="t7" type="TIME" value="T18">6pm.</TIMEX3>
```

2) *Duration and Intervals:* The amount of intervening time in a time interval. As in:



```
"USER must accomplish any PRODUCT task within 2 minutes."
<TIMEX3 tid="t2" type="DURATION" value="PT2M">2 minutes</TIMEX3>
```

Intervals can be described as a range of time defined by a start and end time points. SUTime represents this type in the form of other types.

[2]Check https://gist.github.com/nlothian/9240750 for a complete list of tags
[3]http://alias-i.com/lingpipe/
[4]See http://www.timeml.org for details on the TIMEX3 tag

3) *Set:* A set of temporals, representing times that occur with some frequency. As in:

"The product shall synchronize with the office system every hour."

```
<TIMEX3 tid="t1" type="SET" value="PT1H">every hour</TIMEX3>
```

After tagging a sample set of all the classified NFRs, we identified the following patterns and used them to normalize the entire data set. To do this, first, we replaced all expressions in this format "24[-//*]7" and "24 $\times$ 7 $\times$ 365" with "24 hours per day 365 days per year", and "everyday" with "every day". Likewise, we replaced all "sec(s)" and "min(s)" with "seconds" and "minutes", respectively. In the rest of this section, we present the rules we defined and applied to normalize the temporal expressions embedded in requirements' descriptions.

Temporal Rules

1) $\forall\ [\backslash exp]\langle DURATION\rangle, exp \leftarrow within$
where exp $\in${*no longer than, under, no more than, not be more than, no later, in, for less than, at a maximum*}

2) $\forall\ [\backslash DURATION\backslash TIME\backslash DATE]^{+} \leftarrow alltimes$

the product shall be available for use 24 hours per day 365 days per year.

```
<TIMEX3 tid="t1" type="DURATION" value="PT24H">24 hours</TIMEX3>
<TIMEX3 tid="t2" type="DURATION" value="P1D">day</TIMEX3>
<TIMEX3 tid="t3" type="DURATION" value="P365D">365 days</TIMEX3>
<TIMEX3 tid="t4" type="DURATION" value="P1Y">year</TIMEX3>
```

3) $within\ \langle DURATION\rangle \leftarrow fast$
$if\langle DURATION\rangle == [\backslash seconds\backslash minutes]$

the search results shall be returned no later 30 seconds after the user has entered the search criteria.

```
<TIMEX3 tid="t1" type="DURATION" value="PT30S">later 30
seconds</TIMEX3>
```

4) $\{timely, quick\}\ ||\ [\backslash \textit{positive adj}\ \backslash time] \leftarrow fast$

5) $[8\text{-}9][0\text{-}9][\backslash.?[0\text{-}9]?\%?][IN \mid DET]^{*} time \leftarrow alltimes$

D. Co-occurrence and Regular Expressions

Once the sentence features are utilized to reduce the complexity of the text, we used the co-occurrence and regular expressions to increase the weight of the influential words for each type of NFRs. To explore these rules we manually analyzed 6 to 10 requirements of each NFR and deployed different components of Stanford Parser such as part-of-speech, named entities, sentiment, and relations. Moreover, in this step, we recorded co-occurrence counts of each term within the provided NFR data set as the co-occurrence vector. We used this parameter as a supplement for exploring the SRS regular expressions. For instance, Table I represents the details of the rules we proposed for Security (SE) NFR. Please refer to the footnote[5] for the complete list of these rules containing regular expressions for all of the provided NFRs.

V. Analysis and Results

A. RQ1- Influence of grammatical, temporal and sentimental sentence characteristics on FR/NFR classification

For the classification of functional and non-functional requirements, we used the approach of Hussain et al. [6]. We applied this approach to the unprocessed data set of requirements as well as on the processed one resulting from our preprocessing.

Classification Process: Firstly, we clean up the respective data set by iteratively removing encoding and formatting errors to ensure the further processing. Subsequently, we apply the part-of-speech tagger of the Stanford Parser [14] to assign parts of speech to each word in each requirement.

Based on the tagging of all requirements, we extract the five syntactic features *number of adjectives*, *number of adverbs*, *number of adverbs that modify verbs*, *number of cardinals*, and *number of degree adjective/adverbs*. For each feature, we determine its rank based on the feature's probability of occurrence in the requirements of the data set. According to Hussain et al. [6], we selected a cutoff threshold of > 0.8. Therefore, we determined *number of cardinals* and *number of degree of adjectives/adverbs* as valid features among all five ones for the unprocessed data set. For the processed data set, we identified *number of cardinals* and *number of adverbs* as valid features.

Afterwards, we extract the required keyword features for the nine defined part-of-speech keyword groups *adjective*, *adverb*, *modal*, *determiner*, *verb*, *preposition*, *singular noun*, and *plural noun*. For each keyword group, we calculate the smoothed probability measure and selected the respective cutoff threshold manually to determine the most discriminating keywords for each data set, corresponding to Hussain et al. [6].

Our final feature list for the unprocessed data set consisted of the ten features *number of cardinals*, *number of degree of adjectives/adverbs*, *adjective*, *adverb*, *modal*, *determiner*, *verb*, *preposition*, *singular noun*, and *plural noun*.

Our final feature list for the processed data set consisted of the ten features *number of cardinals*, *number of adverbs*, *adjective*, *adverb*, *modal*, *determiner*, *verb*, *preposition*, *singular noun*, and *plural noun*.

To classify each requirement of the respective data set, we implemented a Java-based feature extraction prototype that parses all requirements from the data set and extracts the values for all ten features mentioned above. Subsequently, we used Weka [16] to train a C4.5 decision tree algorithm [17] which comes with Weka as J48 implementation. According to Hussain et al. [6], we set the parameters for the minimum number of instances in a leaf to 6 to counter possible chances of over-fitting.

Since the data set was not very large with 625 requirements, we performed a 10-fold-cross validation. In the following, we report our classification results for each data set.

Results: The classification of the *unprocessed data set* results in 89.92% correctly classified requirements with a weighted average precision and recall of 0.90. The classification of the *processed data set* results in 94.40% correctly classified requirements with a weighted average precision of 0.95 and recall of 0.94. TABLE II and TABLE III show the details. By applying our approach, we could achieve an improvement of 4.48% correctly classified requirements. In total, we could correctly classify 28 additional requirements, which consist of 9 functional and 19 non-functional ones. When classifying

[5] http://wcm.ucalgary.ca/zshakeri/projects

TABLE I: Proposed co-occurrence and regular expressions for preprocessing SRSs [$(CO(w))$: the set of terms co-occur with word w]

NFR	Keywords	Part of Speech (POS) and Regular Expressions	Replacements
Security [SE]	protect, encrypt, policy, authenticate, prevent, malicious, login, logon, password, authorize, secure, ensure, access	(only /.../nsubj /.../root /...) || (only /.../root /.../nmod: agent /...) $\land$(root is a VBP)	$\Rightarrow$ nsubj agent $\leftarrow$ *authorized user*; root $\leftarrow access$
		$\forall\omega \in \{$*username & password, login, logon security, privacy, right, integrity, polict*$\}$ $\land CO(\omega) \cap CO(NFR_{se}) \neq \emptyset$	$\Rightarrow \omega \leftarrow authorization$
		$\forall\omega \in \{$*reach, enter, protect, input, interface* $\land$product is obj $\land CO(\omega) \cap CO(NFR_{se}) \neq \emptyset$	$\Rightarrow \omega \leftarrow access$

NFRs into sub-categories, the influence of our preprocessing is much stronger. The last two columns of TABLE IV show the overall precision and recall of six different machine learning algorithms for sub-classifying NFRs into the categories listed in columns 1–10 of the table. For all algorithms, results are dramatically better when using the preprocessed data (column Total P) compared to using the raw data (column Total UP).

TABLE II: Classification results of the unprocessed data set

	Correctly Classified	Incorrectly Classified	Precision	Recall	F-Measure	Kappa
NFR	325 (87.84%)	45 (12.16%)	0.95	0.88	0.91	
FR	237 (92.94%)	18 (7.06%)	0.84	0.93	0.88	0.79
Total	562 (89.92%)	63 (10.08%)	0.90	0.90	0.90	

TABLE III: Classification results of the processed data set

	Correctly Classified	Incorrectly Classified	Precision	Recall	F-Measure	Kappa
NFR	344 (92.97%)	26 (7.03%)	0.98	0.93	0.95	
FR	246 (96.47%)	9 (3.53%)	0.90	0.97	0.93	0.89
Total	590 (94.40%)	35 (5.60%)	0.95	0.94	0.94	

B. RQ2- Classifying Non-functional Requirements

In this section, we describe the machine learning algorithms we used to classify NFRs. The performance of each method is assessed in terms of its recall and precision.

1) Topic Modeling: Topic modeling is an unsupervised text analysis technique that groups a small number of highly correlated words, over a large volume of unlabelled text [18], into *topics*.

Algorithms: The *Latent Dirichlet Allocation (LDA)* algorithm classify documents based on the frequency of word co-occurrences. Unlike the LDA approach, the *Biterm Topic Model* (BTM) method models topics based on the word co-occurrence patterns and learns topics by exploring word-word (i.e., biterm) patterns. Some recent studies on the application of topic modeling in classifying short text documents stated that the BTM approach has a better ability in modeling short and sparse text, as the ones typical for requirements specifications.

Results and Evaluation: The modeled topics for both LDA and BTM, including the top frequent words and the NFR assigned to each topic are provided in our source code package[6]. We determined each topic by the most probable words that are assigned to it. For instance, LDA yields the word set {user, access, allow, prior, and detail} for the topic describing the Fault Tolerance sub-category, while BTM yields the set {failure, tolerance, case, use and data}.

Generally, the word lists generated by BTM for each topic are more intuitive than those produced by LDA. This confirms previous research that BTM performs better than LDA in terms of modeling and generating the topics/themes of a corpus consisting of short texts. However, surprisingly, BTM performed much worse than LDA for sub-classifying NFRs as shown in Table IV. This might be because BTM performs its modeling directly at the corpus level and biterms are generated independently from topics.

2) Clustering: Clustering is an unsupervised classification technique which categorizes documents into groups based on likeness [20]. This likeness can be defined as the numerical distance between two documents D_i and D_j which is measured as:

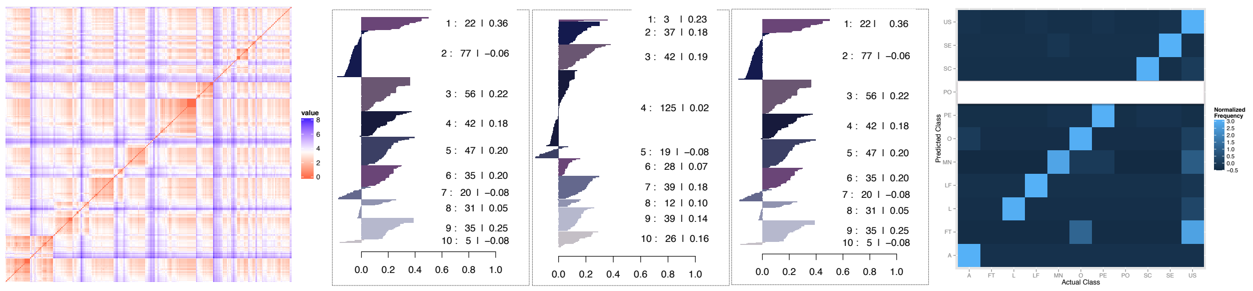
$$d(D_i, D_j) = \sqrt{(d_{i1} - d_{j1})^2 + (d_{i2} - d_{j2})^2 + ... + (d_{in} - d_{jn})^2}$$

Where $(d_{i1}, d_{i2}, ..., d_{in})$ and $(d_{j1}, d_{j2}, ..., d_{jn})$ represent the coordinates (i.e., word frequencies) of the two documents.

Algorithms: The *Hierarchical* (i.e., Agglomerative) algorithm, first, assigns each document to its own cluster and iteratively merges clusters that are closest to each other until the entire corpus forms a single cluster. Despite the hierarchical approach in which we do not need to specify the number of clusters upfront, the *K-means* algorithm assigns documents randomly to *k* bins. This approach computes the location of the centroid of each bin and computes the distance between each document and each centroid. We defined the *k=10* to run this algorithm. However, the k-means approach is highly sensitive to the initial random selection of cluster centroid (i.e., mean), which might lead to different results each time we run this algorithm. Thus, we used a *Hybrid* algorithm, which combined hierarchical and K-means algorithms. This algorithm, first, computes the center (i.e., mean) of each cluster by applying the hierarchical approach. Then computes the K-means approach by using the set of defined clusters' centers.

Results and Evaluation: Before applying clustering algorithms we used Hopkins (H) statistic to test the spatial randomness and assess the *clustering tendency* (i.e., clusterability) of our data set. To this end, we raised the following null hypothesis: $\big($$H_0$*: the NFR data set is uniformly distributed and has no meaningful clusters*$\big)$. As presented in Figure 1 (a), the *H-value* of this test is 0.1 (close to zero), which rejects this hypothesis and concludes that our data set is significantly clusterable. However, as presented in Table IV, the clustering algorithms had poor performance at classifying NFRs. This may imply that the data set under study is quite unstructured and sub-categories of NFRs are not well separated. Thus, an unsupervised algorithm (e.g. Hierarchical or K-means) cannot accurately achieve segmentation.

[6]http://wcm.ucalgary.ca/zshakeri/projects



(a) Hopkins statistic to assess the cluster-ability of the data set (hopkins-stat = 0.1) (b) Hierarchical= 0.13 (c) K-means= 0.1 (d) Hybrid= 0.13 (e) A visual representation of the confusion matrix (BNB algorithm)

Fig. 1: Detailed visual representation of classifying NFRs

TABLE IV: Comparison between classification algorithms for classifying non-functional requirements [(U)P= (Un)Processed]

Algorithm	A		US		SE		SC		LF		L		MN		FT		O		PE		PO		Total [P]		Total [UP]	
	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P
LDA	95	60	61	76	87	87	81	57	60	85	47	20	70	52	10	2	35	70	70	95	-	-	62	62	31	31
BTM	0	0	6	12	13	18	9	8	5	7	0	0	0	0	40	17	0	0	18	43	-	-	8	8	3	3
Hierarchical	13	14	25	20	24	17	16	29	5	3	6	15	19	35	18	40	32	29	26	22	-	-	21	21	16	16
K-means	10	23	19	14	29	18	14	14	21	21	8	15	22	47	18	40	26	30	31	11	-	-	20	20	15	15
Hybrid	15	14	27	22	29	18	20	4	26	24	6	15	17	35	18	40	22	27	26	22	-	-	22	22	19	17
Naïve Bayes	90	90	97	77	97	100	83	83	94	94	75	100	90	82	97	90	78	91	90	100	-	-	91	90	45	45

Moreover, we used Silhouette (s) analysis to assess the cohesion of resulted clusters. We used the function *silhouette()* of *cluster package* to compute the silhouette coefficient. Small *s-value* (i.e., around 0) means that the observation lies between two clusters and has a low cohesion. The results of this test and the details of each cluster, including a number of requirements assigned to it, and its *s-value* are illustrated in Figure 1(b-d).

---

3) Naïve Bayes Classification: This approach is a supervised learning method which predicts unseen data based on the *bayes' theorem* [21] used to calculate conditional probability:

$$P(C = c_k \mid F = f) = \frac{P(F = f \mid C = c_k)P(C = c_k)}{P(f)}$$

Where $C = (c_1, c_2, ..., c_k)$ represents classes and $F = (f_1, f_2, ..., f_d)$ is a vector random variable, which includes one vector for each document.

Algorithm: We use a variation of the multinomial Naïve Bayes (BNB) algorithm known as *Binarized Naïve Bayes*. In this method, the term frequencies are replaced by Boolean presence/absence features. The logic behind this is the higher importance of word occurrence than word frequency to sentiment classification.

Results and Evaluation: To apply this algorithm we employed a 5-fold-cross validation. To reduce the data splitting bias, we run five runs of the 5-fold-cross validation. Overall accuracy is calculated at just over 90% with a *p-value* of 2.2e-16. As illustrated in Table IV, results obtained using the BNB algorithm were generally more accurate. All of the NFRs (except for PO) were recalled at relatively high values ranging from 75 (i.e., Legal requirements) to 97% (i.e., security and performance requirements). To represent more details about the performance of our classifier for each NFR, we visualized the confusion matrix resulted from applying the BNB algorithm (Figure 1 (e)). Each column and row of this matrix represent the actual (i.e., reference) and the prediction data, respectively. The blocks are colored based on the frequency of the intersection between actual and predicted classes (e.g., the diagonal represents the correct predictions for the actual class). Since some of the NFRs in our data set occur more frequently, we normalized our data set before visualizing the confusion matrix. As illustrated in Figure 1 (e), requirements in classes FT, L, MN, O, and SC were often assigned to class US. We can imply that *the terminology we use for representing usability requirements is very general, which covers other NFRs that are indirectly related to usability.* This shows a clear need for additional (or better) sentimental patterns, which differentiate this category of NFR from other similar categories. to differentiate usability requirements from other types of NFRs.

Findings

Finding 1: Our preprocessing approach positively impacted the performance of the applied classification of functional and non-functional requirements. We could improve the accuracy from 89.92% to 95.04%.

Finding 2: Our preprocessing approach strongly impacted the performance of all applied sub-classification methods. For LDA and BNB, both precision and recall doubled.

Finding 3: Among the machine learning algorithms LDA, BTM, Hierarchical, K-means, Hybrid and Binarized Naïve Bayes (BNB), *BNB* had the highest performance for sub-classifying NFRs.

Finding 4: While BTM generally works better than LDA for exploring the general themes and topics of a short-texts corpus, it did not perform well for sub-classifying NFRs.

Finding 5: There is a clear need for additional sentimental patterns/sentence structures to differentiate usability requirements from other types of NFRs.

VI. Limitations and Threats to Validity

In this section, we discuss the potential threats to the validity of our findings in two main threads:

(1) Data Analysis Limitations: The biggest threat to the validity of this work is the fact that our preprocessing model was developed on the basis of the data set given for the RE Data Challenge and that we had to use the same data set for evaluating our approach. We mitigate this threat by using sentence structure features such as temporal, entity, and functional features which are applicable to sentences with different structures from different contexts. We also created a set of regular expressions which are less context-dependent and have been formed mainly based on the semantics of NFRs.

Another limiting factor is that our work depends on the number and choice of the NFR sub-categories used by the creators of our data set. However, our preprocessing can be adapted to a different set of NFR sub-categories. In terms of the co-occurrence rules and regular expressions presented in Table I, we aim to expand these rules by adding more NFR sub-categories in future work, as we gain additional insights from processing real world requirements specifications.

(2) Dataset Limitations: Due to the nature of the RE'17 Data Challenge, we used the data set as is, although it has major data quality issues: (1) Some requirements are incorrectly labeled. For example, R2.18 "The product shall allow the user to view previously downloaded search results, CMA reports and appointments" is labeled as NFR. Obviously, however, this is a functional requirement. (2) The important distinction between quality requirements and constraints is not properly reflected in the labeling. (3) The selection of the requirements has some bias. For example, the data set does not contain any compatibility, compliance or safety requirements. Neither does it contain any cultural, environmental or physical constraints. (4) Only one single requirement is classified as PO which makes this sub-category useless for our study. The repetition of our study on a data set of higher data quality is subject to future work.

Furthermore, the *unbalanced data set* we used for classifying the NFRs may affect the findings of this study. However, a study by Xue and Titterington [22] revealed that there is no reliable empirical evidence to support the claim that an unbalanced data set negatively impacts the performance of the LDA/BTM approaches. Further, a recent study by López et al. [23] shows that the unbalanced ratio by itself does not have the most significant effect on the classifiers performance, but there are other issues such as (a) the presence of small disjuncts, (b) the lack of density, (c) the class overlapping, (d) the noisy data, (e) the management of borderline examples, and (f) the dataset shift that must be taken into account. The pre-processing step we conducted before applying the classification algorithms helps discriminate the NFR sub-classes more precisely and mitigates the negative impact of the noisy data and borderline problems. Moreover, we employed the n-fold cross validation technique which helps generate enough positive class instances in different folds and reduces additional problems in the data distribution especially for highly unbalanced datasets. This technique, to a great extent, mitigates the negative impact of class overlapping, the dataset shift, and the presence of small disjuncts issues on the performance of the classification algorithms we applied in this study.

VII. Conclusion and Implications

Our findings are summarized in the box at the end of Section V. In particular, we conclude that using our preprocessing approach improves the performance of both classifying FR/NFR and sub-classifying NFR into sub-categories. Further, we found that, among popular machine learning algorithms, Binarized Naïve Bayes (BNB) performed best for the task of classifying NFR into sub-categories. Our results further show that, although BTM generally works better than LDA for extracting the topics of short-texts, BTM does not perform well for classifying NFRs into sub-categories. Finally, additional (or better) sentimental patterns and sentence structures are needed for differentiating usability requirements from other types of NFRs.

References

[1] M. Glinz, *A Glossary of Requirements Engineering Terminology, Version 1.6*, International Requirements Engineering Board (IREB). Available at https://www.ireb.org/en/downloads/cpre-glossary, 2014.

[2] ——, "On non-functional requirements," in *15th IEEE International Requirements Engineering Conference (RE'07)*, 2007, pp. 21–26.

[3] N. A. Ernst and J. Mylopoulos, "On the perception of software quality requirements during the project lifecycle," in *Requirements Engineering: Foundation for Software Quality: 16th International Working Conference, REFSQ 2010, Essen, Germany, June 30–July 2, 2010. Proceedings*, R. Wieringa and A. Persson, Eds. Springer Berlin Heidelberg, 2010, pp. 143–157.

[4] Z. S. H. Abad and G. Ruhe, "Using real options to manage technical debt in requirements engineering," in *23rd IEEE International Requirements Engineering Conference (RE)*, 2015, pp. 230–235.

[5] Z. S. H. Abad, A. Shymka, S. Pant, A. Currie, and G. Ruhe, "What are practitioners asking about requirements engineering? an exploratory analysis of social q a sites," in *24th IEEE International Requirements Engineering Conference Workshops (REW)*, 2016, pp. 334–343.

[6] I. Hussain, L. Kosseim, and O. Ormandjieva, "Using linguistic knowledge to classify non-functional requirements in SRS documents," in *International Conference on Application of Natural Language to Information Systems*, Springer, 2008, pp. 287–298.

[7] N. Niu and S. Easterbrook, "Extracting and modeling product line functional requirements," in *2008 16th IEEE International Requirements Engineering Conference*, 2008, pp. 155–164.

[8] C. Duan, P. Laurent, J. Cleland-Huang, and C. Kwiatkowski, "Towards automated requirements prioritization and triage," *Requir. Eng.*, vol. 14, no. 2, pp. 73–89, 2009.

[9] E. Knauss and D. Ott, "Automatic requirement categorization of large natural language specifications at mercedes-benz for review improvements," in *Proceedings of the 19th International Conference on Requirements Engineering: Foundation for Software Quality*, ser. REFSQ'13, Essen, Germany: Springer-Verlag, 2013, pp. 50–64.

[10] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc, "The detection and classification of non-functional requirements with application to early aspects," in *14th IEEE International Requirements Engineering Conferenc*, 2006, pp. 39–48.

[11] M. Rahimi, M. Mirakhorli, and J. Cleland-Huang, "Automated extraction and visualization of quality concerns from requirements specifications," in *2014 IEEE 22nd International Requirements Engineering Conference*, 2014, pp. 253–262.

[12] E. Knauss, D. Damian, G. Poo-Caamano, and J. Cleland-Huang, "Detecting and classifying patterns of requirements clarifications," IEEE Computer Society, 2012, pp. 251–260.

[13] Y. Ko, S. Park, J. Seo, and S. Choi, "Using classification techniques for informal requirements in the requirements analysis-supporting system," *Inf. Softw. Technol.*, vol. 49, no. 11-12, pp. 1128–1140, Nov. 2007.

[14] D. Klein and C. D. Manning, "Accurate unlexicalized parsing," in *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ser. ACL '03, Association for Computational Linguistics, 2003, pp. 423–430.

[15] A. X. Chang and C. D. Manning, "Sutime: A library for recognizing and normalizing time expressions.," in *LREC*, 2012, pp. 3735–3740.

[16] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[17] J. R. Quinlan, *C4. 5: Programs for machine learning*. Elsevier, 2014.

[18] H. M. Wallach, "Topic Modeling: Beyond Bag-of-words," in *Proceedings of the 23rd International Conference on Machine Learning*, ACM, 2006, pp. 977–984.

[20] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, 1999.

[21] D. D. Lewis, "Naive (bayes) at forty: The independence assumption in information retrieval," in *Machine Learning: ECML-98: 10th European Conference on Machine Learning Chemnitz, Germany, April 21–23, 1998 Proceedings*, C. Nédellec and C. Rouveirol, Eds. Springer Berlin Heidelberg, 1998, pp. 4–15.

[22] J. H. Xue and D. M. Titterington, "Do unbalanced data have a negative effect on lda?" *Pattern Recognition*, vol. 41, no. 5, pp. 1558 –1571, 2008.

[23] V. López, A. Fernández, S. Garía, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," *Information Sciences*, vol. 250, pp. 113 –141, 2013.