Expert Systems with Applications: X 1 (2019) 100001

Contents lists available at ScienceDirect

Expert Systems with Applications: X

journal homepage: www.elsevier.com/locate/eswax

Review

A review of machine learning algorithms for identification and classification of non-functional requirements

Manal Binkhonain, Liping Zhao*

[a] School of Computer Science, The University of Manchester, Kilburn Building, Oxford Road, Manchester M13 9PL, United Kingdom

ARTICLE INFO

Article history:
Received 17 August 2018
Revised 14 January 2019
Accepted 24 February 2019
Available online 12 March 2019

Keywords:
Requirements engineering
Non-functional requirements
Requirements documents
Requirements identification Requirements classification
Machine learning

ABSTRACT

Context: Recent developments in requirements engineering (RE) methods have seen a surge in using machine-learning (ML) algorithms to solve some difficult RE problems. One such problem is identification and classification of non-functional requirements (NFRs) in requirements documents. ML-based approaches to this problem have shown to produce promising results, better than those produced by traditional natural language processing (NLP) approaches. Yet, a systematic understanding of these ML approaches is still lacking.
Method: This article reports on a systematic review of 24 ML-based approaches for identifying and classifying NFRs. Directed by three research questions, this article aims to understand what ML algorithms are used in these approaches, how these algorithms work and how they are evaluated.
Results: (1) 16 different ML algorithms are found in these approaches; of which supervised learning algorithms are most popular. (2) All 24 approaches have followed a standard process in identifying and classifying NFRs. (3) Precision and recall are the most used matrices to measure the performance of these approaches.
Finding: The review finds that while ML-based approaches have the potential in the classification and identification of NFRs, they face some open challenges that will affect their performance and practical application.
Impact: The review calls for the close collaboration between RE and ML researchers, to address open challenges facing the development of real-world ML systems.
Significance: The use of ML in RE opens up exciting opportunities to develop novel expert and intelligent systems to support RE tasks and processes. This implies that RE is being transformed into an application of modern expert systems.

1. Introduction

Although non-functional requirements (NFRs) are regarded to be important and critical for the success of a software project, there is still no consensus in the requirements engineering (RE) community what NFRs are and how we should elicit, document, and validate them (Glinz, 2007). In particular, there is still no consensus where in the RE process NFRs should be considered: Should they be defined before the elicitation of functional requirements (FRs), after FRs, or at the same time with FRs? Questions like these remain unanswered (Chung, Nixon, Yu, & Mylopoulos, 2012).

In addition to these problems, NFRs are the system level requirements, which are orthogonal to FRs (Yang et al., 2012). As requirements documents are typically written and structured around FRs, NFRs become intertwined with FRs (Cleland-Huang, Settimi, Zou, & Solc, 2007). To complicate things further, during the design of software architecture, architects need to be able differentiate NFRs from FRs, so that they can transform different types of requirements into different types of architectural components (Nuseibeh, 2001). An important RE task is therefore to correctly extract NFRs from requirements documents and organize them in categories. This task is time consuming and error prone (Cleland-Huang et al., 2007; Rashwan, Ormandjieva, & Witte, 2013).

To support this task, RE researchers have been proposing automatic or semi-automatic approaches for identifying NFRs in requirements documents (Kayed, Hirzalla, Samhan, & Alfayoumi, 2009; Ko, Park, Seo, & Choi, 2007; Rago, Marcos, & Diaz-Pace, 2013;

* Corresponding author.
E-mail addresses: manal.binkhonain@postgrad.manchester.ac.uk (M. Binkhonain), liping.zhao@manchester.ac.uk (L. Zhao).

https://doi.org/10.1016/j.eswax.2019.100001

Sharma, Ramnani, & Sengupta, 2014; Vlas & Robinson, 2011). In recent years, machine learning (ML) algorithms have been integrated into these approaches with promising results being reported (Cleland-Huang et al., 2007; Knauss, Houmb, Schneider, Islam, & Jürjens, 2011; Mahmoud, 2015; Mahmoud & Williams, 2016). However, a systematic understanding of these emerging methods is currently inadequate in literature. This article aims to fill this gap.

Specifically, this article reports on a systematic review of 24 current ML-based approaches. The review intends to find out what ML algorithms have been used to classify NFRs, how they work and evaluated. These findings will enable us to identify what challenges need to be addressed in order to improve the state of ML-based approaches. Our review is driven by the following research questions (RQ):

- *RQ1: What machine learning algorithms have been applied in the selected studies? Which ones are the most popular?*
- *RQ2: What are the processes that the reported ML-based approaches follow to identify and classify NFRs in requirements documents?*
- *RQ3: What measures have been used to evaluate the performance of the ML algorithms applied in these approaches? What are the performance results of these algorithms?*

To continue our report, the remaining article is organized as follows: Section 2 discusses related work. Section 3 presents the review process whereas Section 4 analyses the review results. Section 5 discusses our key research findings and open challenges. Section 6 identifies the validity threats to this review. Finally, in Section 7, we conclude the review with a discussion of the implications of the review and some directions for future research.

2. Related reviews

Very few related reviews are available in literature. To the best of our knowledge, only one review, conducted by Meth and colleagues (Meth, Brhel, & Maedche, 2013), explored the tools used for eliciting automatic requirements from textual documents. That review included 36 studies that were published between January 1992 and March 2012. Those studies have been analysed by adopting two perspectives: design, which focuses on the technical concepts adopted in each tool, and evaluation, which describes methods for evaluating the effectiveness of those tools.

From the design perspective, the studies involved in that review were classified into four categories: identifying both FRs and NFRs, generating models, analysing quality requirements by defining defects and finding key abstractions. In addition, that review examined the degree of automation for each tool (full or semi-automation) as well as the approaches used to generate knowledge for the reuse of either requirements or knowledge related to requirements.

Conversely, the evaluation perspective contains evaluation approaches, concepts, and measures that are used to evaluate tool performance. Identifying requirements automatically is a small part of that review, and these requirements might be either NFRs or FRs, regardless of the techniques used. By contrast, this review focuses only on identifying NFRs via ML algorithms.

3. Review method and process

To answer our research questions, we have adopted the snowballing approach proposed by Wohlin (2014) to identify relevant studies. Snowballing is an alternative to the traditional systematic literature review (SLR) approach (Kitchenham & Charters, 2007). The idea is to use the reference list of a paper or the citations to the paper to systematically identify additional papers. Snowballing consists of two steps: backward snowballing (looking at the reference lists of a paper) and forward snowballing (looking at the citations in which the paper is actually cited). Fig. 1 depicts the process of snowballing (Wohlin, 2014).

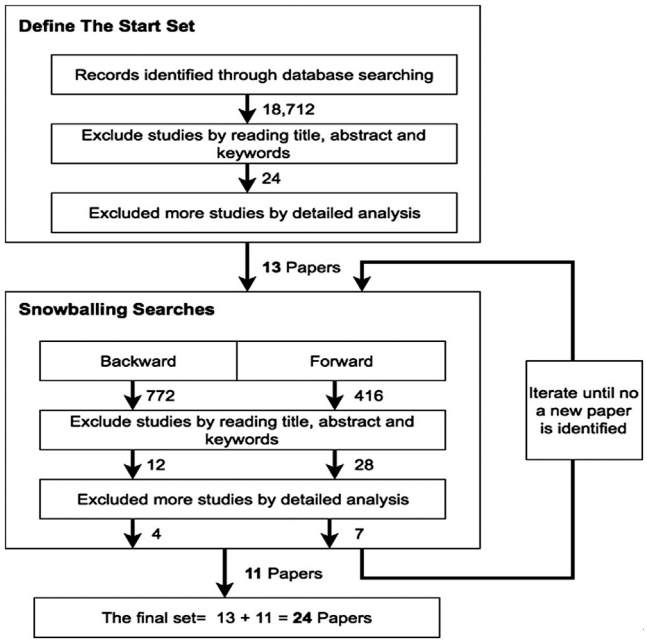

Fig. 1. The process of study selection based on Wohlin's snowballing method (Wohlin, 2014).

The starting point of the snowballing approach is the identification of a start set of relevant papers. Based on each paper in the start set, one can then conduct backward and forward snowballing. In the following subsections, we detail how we used the snowballing approach to identify the relevant papers for our review.

3.1. Identification of the start set

The start set can be identified through the traditional SLR search method, in which search strings are formulated to query relevant online databases. To avoid bias towards a specific publisher, Wohlin (2014) advises to use Google Scholar to search for the start set. However, we found that the search engine of Google Scholar is too general to be really useful. For example, the search string "("non-functional requirements" OR NFRs) AND classification AND "machine learning"" returned 26,700 results. We have therefore decided to conduct searches on online databases that are known to us. Our search string was defined by breaking down the research questions according to the PICOC criteria (population, intervention, comparison, outcome, and context), as recommended by Kitchenham & Charters, 2007). These terms were connected using Boolean operators; the operator OR was used for synonyms (alternative terms) and AND was used for linking the search terms, as illustrated in Table 1.

By manually inspecting the search results, we selected 120 papers for further selection. This entailed (1) reading the title and abstract of each paper and (2) reading the full text of each remaining paper. The following inclusion and exclusion criteria were used for study selection:

Inclusion:

- All the papers that present the primary studies on using ML algorithms for identifying NFRs are included.

Table 1
PICOC criteria to define the search string for the start set.

	Population		Intervention	Comparison	Outcome	Context
	Requirements Engineering (RE)		Machine Learning (ML) approaches for RE	Not applicable	An understanding of the ML approaches for identifying and classifying non-functional requirements (NFRs) from natural language documents	Requirements Engineering (RE)
Main keywords	Non-Functional Requirement	Requirements Elicitation, Requirements Analysis, Requirements Specification	Machine Learning	Not applicable		
Alternatives	Non-functional requirements OR NFRs OR Quality Requirements OR Quality Attributes	Requirements Extraction OR Requirements Analysis OR Requirements Specification OR Requirements Categorization OR Requirements Classification	Machine Learning OR Supervised learning OR Unsupervised learning OR Semi-supervised learning			

Table 2
The initial start set of the relevant papers for snowballing.

Number	Author	Year	Title	Publication venue	Database	Citation counts
1	Casamayor et al.	2010	Identification of non-functional requirements in textual specifications: A semi-supervised learning approach	Information and Software Technology Journal	Science Direct	71
2	Slankas and Williams	2013	Automated extraction of non-functional requirements in available documentation	Natural Language Analysis in Software Engineering Conference	IEEExplore	39
3	Hussain et al.	2008	Using linguistic knowledge to classify non-functional requirements in SRS documents	International Conference on Application of Natural Language to Information Systems	Springer Link	28
4	Knauss et al.	2011	Supporting Requirements Engineers in Recognising Security Issues	Requirements Engineering: Foundation for Software Quality	Springer Link	16
5	Gokyer et al.	2008	Non-functional requirements to architectural concerns: ML and NLP at crossroads	International Conference on Software Engineering Advances	IEEExplore	8
6	Nguyen et al.	2015	Rule-based extraction of goal-use case models from text	European Software Engineering Conference and Symposium on the Foundations of Software Engineering	ACM	7
7	Abad et al.	2017	What Works Better? A Study of Classifying Requirements	RE Conference	IEEExplore	4
8	Jindal et al.	2016	Automated classification of security requirements	International Conference on Advances in Computing, Communications and Informatics	IEEExplore	3
9	Malhotra et al.	2016	Analyzing and evaluating security features in software requirements	Innovation and Challenges in Cyber Security	IEEExplore	3
10	Lu and Liang	2017	Automatic Classification of Non-Functional Requirements from Augmented App User Reviews	The Evaluation and Assessment in Software Engineering Conference	ACM	1
11	Kurtanović and Maalej	2017	Automatically Classifying Functional and Non-functional Requirements Using Supervised Machine Learning	RE Conference	IEEExplore	1
12	Knauss and Ott	2014	(Semi-) automatic Categorization of Natural Language Requirements	Requirements Engineering: Foundation for Software Quality	Springer Link	1
13	Deocadez et al.	2017	Automatically Classifying Requirements from App Stores: A Preliminary Study	RE Conference	IEEExplore	0

- If the same study is reported by more than one paper, the paper that provides the most detailed description of the study is included.

Exclusion:

- Grey literature and non-English written literature are excluded.
- Papers that present secondary studies, such as surveys and literature reviews, are excluded.

At the end of the selection process, we identified 13 relevant papers, which were then included in our start set (see Table 2).

3.2. Backward snowballing

For each paper in the start set, we conducted a backward snowballing search on the references of the paper. The process of backward snowballing, as shown in Fig. 1, is detailed as follows.

First, the title and reference context of each referenced paper were reviewed, and in some cases, additional sections of the referenced paper, such as the abstract and keywords, were evaluated. Reference context was based on the text surrounding the reference in the paper.

Second, the inclusion and exclusion criteria were applied based on a full-text reading. Papers identified as relevant were added to

the starting set, and this process was repeated until no new papers were identified. During four iterations of this process, 772 referenced papers were examined, 12 of them were identified as relevant but only 4 were added to the start set, as the remaining eight were overlapping with the papers in the start set. The start set now consists of 17 papers.

3.3. Forward snowballing

For each paper in the start set, we conducted a forward snowballing search on the citations of the paper. The citations of each paper in the start set were identified on Google Scholar. Each citation paper was reviewed as follows: first, the title, abstract, keywords and reference context (i.e., the context of text surrounding each citation) of each study were reviewed. Relevant papers were added to the starting set, and this process was repeated until no new papers were identified. During four iterations, 416 studies were examined, 28 studies were identified and 7 relevant studies were added to the start set based on the inclusion and exclusion criteria.

At the end of forward snowballing, 24 relevant papers were selected as the final set for our review. This set of papers is listed in Appendix A, numbered as S1, S2, etc., in chronological order.

3.4. Extracting and synthesizing the data

The required data were extracted from each of the 24 selected studies. The data extraction form (see Table 3) was used to record the data for each study. Two types of data were extracted: the data required for answering the research questions and the data required for displaying the bibliographic information of the study. The extracted data were stored in an Excel file.

The data synthesis method used in this review was based on the constant comparison method (CCM), a core element of grounded theory (Glaser, Strauss, & Strutzel, 1968) that has been widely used for qualitative analysis (Dixon-Woods, Agarwal, Jones, Young, & Sutton, 2005; Harding, 2013). CCM (Harding, 2013) focuses on finding the similarities and differences between the data extracted from the studies. The CCM method can be used independently (Harding, 2013) or combined with summarization. In our case, we applied both CCM and summarization to synthesize the extracted data.

The 24 studies were categorized according to their goals (contributions), listed in Table 4. Fig. 2 shows the distribution of the selected primary studies published between 2007 and 2017. The number of studies published each year fluctuated significantly, with the largest number of studies published in 2017 (6 studies) and 2016 (4 studies). Studies were published in different databases, including IEEEXplore (10 studies), Springer Link (8 studies), Science Direct (3 studies), ACM Digital Library (2 studies) and Semantic Scholar (one study). The ratio between conference papers and journal articles was 18:6. Most conference papers were published in the Requirements Engineering Conference and Requirements Engineering: Foundation for Software Quality Conference, whereas journal articles were published by Requirements Engineering Journals and Information and Software Technology.

Table 3
Data extraction form.

Data item	Description
Bibliographic information	Authors, title, publication venue, and publication year.
Study goal	The main purpose of the selected paper.
ML algorithm	Type of ML algorithm used to identify or classify the textual requirements
Requirements document	The source and size of the document
Types of NFRs	Types of NFRs identified
ML approach	The process and techniques used for identifying the NFRs.
Evolution methods and results	How the effectiveness of the algorithm was measured in the selected paper and what were the results.

Table 4
Goals of selected studies.

Study goal	Study ID	No. studies
To extract and classify security requirements.	S6, S11, S16, S17	4
To identify NFRs in the informal documents, including user comments, commit messages and installation manuals.	S7, S9, S10, S19, S20, S21, S22, S24	8
To extract NFRs from textual documents, to use the extracted requirements to improve the description of NFRs and support other RE tasks, such as generating architectural models automatically.	S3, S8, S13	3
To optimize NFR classifiers by using different text classification techniques such as features selection, weighting, representation or ML algorithms themselves.	S1, S2, S4, S5, S12, S14, S15, S18, S23	9

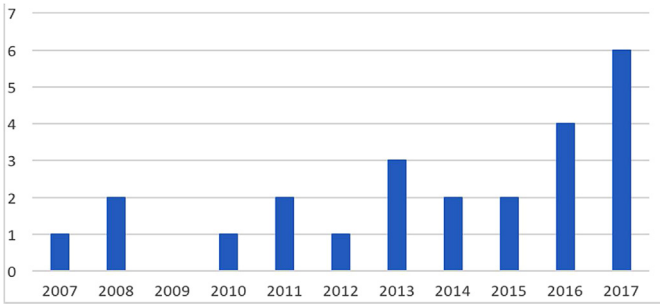

Fig. 2. Distribution of the 24 selected studies from the beginning of 2007 when the first study was reported to the end of 2017 when our search was completed.

4. Results

This section presents the review results and answers our research questions.

4.1. ML algorithms (RQ1)

RQ1: What machine learning algorithms have been applied in the selected studies? Which ones are the most popular?

A total of 16 different ML algorithms were found in the 24 selected studies. These algorithms fall into three types, comprising 7 supervised learning (SL), 4 unsupervised learning (USL) and 5 semi-supervised learning (SSL), as summarized in Table 5. Table 5 also shows that SL is the most popular type of ML, as it was used in 17 studies (71%), and SVM is overall the most popular ML algorithms, found in 11 studies (45.8%).

In the following subsections, we introduce the 16 ML algorithms found in our review.

4.1.1. Supervised Learning algorithms

There are seven SL algorithms found in our review, which are:

- *Support Vector Machines (SVMs).* These are a group of supervised learning algorithms, which can be used for classification or regression. The SVM algorithm is based upon the statistical learning theory and the Vapnik–Chervonenkis (VC) dimensions (Vapnik & Chervonenkis, 2015). These algorithms look for

Table 5
ML algorithms used in the selected studies.

Type	ML algorithm	Purpose	Study ID	No. studies
SL	SVM	Create the optimal separating line to classify all of the input data into different classes.	S3, S5, S7, S8, S9, S10, S11, S12, S20, S21, S22	11
	NB	Compute the probability of each category for the given data based on the assumption of the independence between features.	S1, S4, S6, S7, S11, S19, S20	7
	DT	Model dataset into hierarchical structure where each leaf is a class.	S2, S16, S17, S19, S20	5
	KNN	Identify the nearest neighbours of the testing data for a given input and predict the category based on certain distance functions.	S9, S11, S14, S20, S21	5
	MNB	Similar to NB, but it also considers the frequency of words in each input.	S7, S10, S23	3
	RB	Model dataset as a collection of rules; the left side contains conditions, and the right side contains the classes.	S18	1
	DMNB	Compute a discriminative version of frequency estimate.	S7	1
USL	LDA	Identify topics that the documents contain based on certain probabilities.	S7, S23, S24	3
	K-means	Find K cluster centres that are representative of the dataset.	S15, S23	2
	HAC	Merge most similar clusters of a dataset into a large cluster until a specified number of clusters left.	S15, S23	2
	BTM	Identify the topics based on word co-occurrence patterns.	S23	1
SSL	EM	Maximize likelihood estimation in problems with missing data.	S4	1
	Self-training	Train supervised algorithm incrementally by adding most confidently labeled data to the training set in each iteration.	S20	1
	Active Learning	Choose the high-confidence label dataset in each iteration and manually label instances that have the least confidence.	S21	1
	RAS-CO	Select a random subspace from the featured spaces and trains the classifier of each subspace.	S20	1
	Rel-RASCO	Select a random subspace containing relevant feature subspaces	S20	1

all data points that are close to the opposing class. These data points, which are known as support vectors, are important in the classification task, while the rest of the training data points are ignorable. Then, the best separation line, known as the decision boundary, is defined. This line segregates classes using different functions such as the Liner Kernel function used in S5 where the authors of the study claimed that the linear function is superior to non-linear kernels for classifying textual contents (Bennett & Bredensteiner, 2000).

- *Naïve Bayes (NB).* This is a statistical method based on Bayes's theorem (Lewis, 1998), with a strong 'assumption of independence between features'. NB computes the probability of an input that is relevant to a specific pre-defined class, with the help of certain statistical functions (Ott, 2013). The output of this algorithm is the category with the highest probability.
- *Decision Tree (DT).* This is a logic-based algorithm where datasets are modeled in hierarchical structures using a series of comparisons of if/else statements (Alpaydin, 2014, Khan, Baharudin, Lee, & Khan, 2010). Each node in the tree consists of either decision nodes containing terms or 'leaves' (Alpaydin, 2014), which consist of class label predictions (Dobra, 2009). The branches are labeled by the weight of each term in the document.
- *K-Nearest Neighbours (K-NN).* This is a statistical method used to predict the new input by computing the similarity between the test data and the new instance through finding the nearest data point(s) (or data objects) in the training dataset based on certain distance functions. *K* represents the number of nearest data points (i.e., neighbours) (Alpaydin, 2014; Khan et al., 2010).
- *Multinomial Naive Bayes (MNB).* This is a special version of NB; however, it considers the frequency of words in each input (i.e. requirement) and not only their appearance (Ott, 2013)
- *Rule-Based (RB). This* is similar to DT where the dataset is modeled by collections of rules. However, RB classifiers allow overlaps in decision space, while DT uses a strictly hierarchical approach (Kotsiantis, Zaharakis, & Pintelas, 2006). The left side of these rules consists of conditions, while the right side contains the classes. These rules are derived from the dataset (Kotsiantis et al., 2006, Muhammad & Yan, 2015).
- *Discriminative Multinomial Naive Bayes (DMNV).* This is a discriminative version of MNB that adds a discriminative element to the frequency estimate in order to discriminatively compute the appropriate frequencies from the data (Su, Zhang, Ling, & Matwin, 2008).

4.1.2. Unsupervised Learning algorithms

The 4 USL algorithms identified in the selected studies are briefly described as follows:

- *Latent Dirichlet Allocation (LDA). This* is a generative probabilistic model used to automatically identify topics that documents contain based on certain probabilities (Zou, Xu, Yang, Zhang, & Yang, 2017; Blei, Ng, & Jordan, 2003).
- *K-means. This* is an iterative algorithm, which classifies documents randomly into certain numbers of clusters (*K*). *K*-means work iteratively to assign each data point in the space into the nearest single cluster of *K*-clusters (Abad et al., 2017; Mahmoud & Williams, 2016).
- *Hierarchical Agglomerative Clustering (HAC).* This is an iterative algorithm merging the similar elements of a dataset into a large cluster until the entire dataset forms a single cluster. In contrast to K-means, HAC does not need to determine the number of clusters up front. This algorithm includes three categories: complete linkage, single linkage, and average linkage. The difference between these categories is the method used to measure the distance between two clusters for each iteration. For example, single linkage uses the most similar pair of elements, while complete linkage chooses the most dissimilar pair of elements; average linkage defines the distance as the average between the pairs in the data element (Abad et al., 2017; Mahmoud & Williams, 2016).
- *Biterm Topic Modelling (BTM).* This approach identifies topics by modeling word to word co-concurrence patterns (Yan, Guo, Lan, & Cheng, 2013). Such patterns are called biterms, which are unordered pair of words that appear frequently together in a dataset (Yan et al., 2013).

4.1.3. Semi-supervised Learning algorithms

The five SSL algorithms identified in the selected studies are briefly described as follows:

- *Expectation Maximisation (EM).* This is an iterative approach using probabilistic functions for maximum likelihood estimation

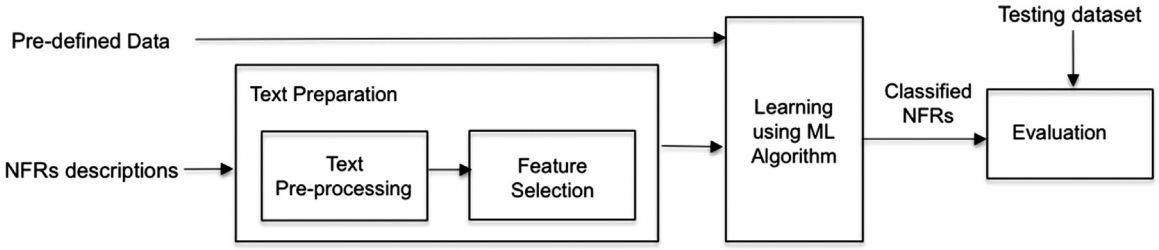

Fig. 3. A general process of applying ML algorithms to classify NFRs in textual requirements documents.

in problems with missing data (Nigam, McCallum, & Mitchell, 2006).

- *Self-training.* It is incremental semi-supervised training (Rosenberg, Hebert, & Schneiderman, 2005), where the labeled datasets are used to train a supervised classifier, which then is used to classify unlabelled data. The most confidently labeled data are added to the training set, and the classifier is re-trained (Zhu, 2005).
- *Active learning.* It is aimed to achieve high accuracy by choosing the labeled dataset carefully, which will derive high confidence/accuracy. The instance which has the least confidence will be manually labeled and added to the training set (Li, Huang, Ge, Luo, & Ng, 2017)
- *RAndom Subspace Method for Co-training (RAS-CO).* It is used both a random subspace method and a co-training method. Co-training consists of two classifiers that are trained with different parts of a labeled dataset (Zhu, 2005). RAS-CO basically selects a random subspace from the featured spaces and trains the classifier of each subspace. The idea behind this algorithm is that each classifier is sensitive to different features and can complement the other classifier (Wang, Luo, & Zeng, 2008).
- *Relevant Random Subspace Method for Co-training (Rel-RASCO).* This is similar to RAS-CO; however, the aim of this algorithm is to select a random subspace that contains relevant features' subspaces (Deocadez, Harrison, & Rodriguez, 2017).

4.2. Process of using ML algorithms to identify NFRs (RQ2)

RQ2: What are the processes that the reported ML-based approaches follow to identify and classify NFRs in requirements documents?

Our analysis of the 24 selected studies has revealed a general process pattern for applying a ML-based approach to identify and classify NFRs in a textual document. This process is divided into three major phases (Fig. 3): the text preparation phase, which involves pre-processing the text of the input requirements and selecting meaningful features from it; the learning phase, which involves applying ML algorithms; and the evaluation phase, which is concerned with assessing an ML algorithm's approach to classifying NFRs. In the following subsections, we describe this general process.

4.2.1. Text preparation phase

The training phase consists of two major steps: text pre-processing feature selection. These two steps are described as follows.

4.2.1.1. Text pre-processing. This step takes a textual requirements document as input and then applies different natural language processing (NLP) techniques to pre-process the input document. A total of 11 NLP techniques were found in the selected studies. Table 6 shows which study used which NLP technique. These 10 techniques are briefly summarised as follows.

Table 6
NLP techniques used in the selected studies for text processing.

NLP technique	No. studies	Studies ID
Stemming	11	S1, S2, S3, S4, S5, S8, S15, S17, S18, S19, S20
Stop Word Removing	10	S1, S4, S7, S9, S13, S15, S17, S19, S22, S24
POS	7	S2, S3, S5, S9, S18, S22, S23
Tokenization	7	S1, S2, S6, S8, S11, S16, S18
Lemmatization	6	S9, S13, S15, S19, S21, S22
N-gram	4	S5, S10, S12, S22
Normalization	2	S4, S24
Regular Expressions	2	S5, S23
Dependency Parsing	2	S9, S13
Entity Tagging	1	S23
Temporal Tagging	1	S23

- *Tokenization:* splitting a sentence into a series of words (Khan et al., 2010).
- *N-gram*: separating each given string into subsequent N items, where N can be words, letters or statements (Cavnar & Trenkle, 1994).
- *Stop-words removal*: removing auxiliary verbs (be, do and have), conjunctions (and, or) and articles (the, a, and an) in sentences (Khan et al., 2010).
- *Stemming*: reducing inflected (or sometimes derived) words to their word stem, base or root form. For example, the words 'goes', 'gone' and 'going' will map to 'go' (Jivani, 2011).
- *Lemmatization*: determining lemma (the infinitive form of verbs and the singular form of nouns and adjectives) of each word. For example, the words 'goes', 'gone', 'going' and 'went' will map to 'go' (Jivani, 2011).
- *Part of Speech (POS) Tagging*: assigning tags to noun, verb, and preposition in a sentence (Casamayor, Godoy, & Campo, 2010).
- *Regular Expressions*: matching a defined expression that describes string patterns. This expression contains specialized notations, such as the character (^) means 'not' (Abad et al., 2017).
- *Dependency Parsing:* identifying the grammatical relationships between words in a sentence to recognize the important parts and ignore unimportant parts in the sentence (De Marneffe & Manning, 2008).
- *Named Entity Recognition (NER)*: assigning a pre-defined category to named entities (Abad et al., 2017).
- *Temporal Tagging:* recognizing and normalizing temporal expression (Abad et al., 2017).

4.2.1.2. Feature selection. This step converts the pre-processed requirements document into a format that can be understood by ML algorithms (Zhang, Zhang, Dong, & Tan, 2005). In this step, a document is represented as a vector of terms, where the values of these terms are weighted via different techniques, including binary (i.e., Boolean) and non-binary methods (Khan et al., 2010; Zhang et al., 2005). In the binary method, the value is 1 if the document contains the words and 0 if it does not (Zhang et al., 2005). Two studies, S6 and S23, adopted this method. In the non-binary

method, the importance of the terms is calculated using different techniques, such as the following:

- *Bag of Words:* This method calculates the frequency of each word in a given document. (Khan et al., 2010) This method is adopted in six studies: S3, S7, S17, S19, S21, and S23.
- *Term Frequency-Inverse Document Frequency (TF-IDF)*: This method calculates the frequency that a word appears in the document inverse of the number of times the word appears in the corpus (Agarwal & Mittal, 2014; Ikonomakis, Kotsiantis, & Tampakas, 2005; Khan et al., 2010; Zhang et al., 2005). Seven studies used the TF-IDF method: S1, S4, S16, S19, S20, S21, and S22.

The next step is Feature Selection, which aims to reduce the number of features by removing irrelevant features and keeping those with the highest score (Agarwal & Mittal, 2014; Ikonomakis et al., 2005; Korde & Mahender, 2012). The methods that are widely used for feature selection in the reviewed studies include information gain (S16, S5, S11) and Chi-square (S19) (Agarwal & Mittal, 2014; Ikonomakis et al., 2005; Korde & Mahender, 2012).

4.2.2. Predefined data

As illustrated in Fig. 3, pre-defined data is a precondition for building NFRs classifiers using ML algorithms. SL algorithms require a labeled dataset, whereas USL algorithms require a set of predefined categories or keywords. SSL algorithms, similar to SL algorithms, also require a labeled dataset. Here we analyse the datasets used in the selected studies.

4.2.2.1. Supervised Learning. SL required a set of NFRs that had been correctly classified to their categories. The datasets used in the selected studies came from three sources: academic—where the requirements were written by academic students and researchers, industrial—where they were written to describe or simulate industrial products and informal—where they were written by end-users as comments, reviews, posts and requests in open source communities (sourceforge.net). The academic dataset was used the most in the selected studies (11/24, 46%), which was followed by the industrial dataset (5/24, 20%) and then the informal dataset (4/24, 16%).

The number of requirements used to train datasets varies greatly in the selected studies. For example, S16 used 58 requirements (in four categories) to train its data (Jindal, Malhotra, & Jain, 2016) and S11 use 10,963 (seven categories). The main categories of NFRs mentioned in the selected studies that adopted SL are represented in Table 7.

Datasets were labeled manually in all of the selected studies. Thus, two possible threats are posed: the subjectivity of the labelling process (i.e., annotations) and incorrectly labelling. There are two steps to avoid these threats: annotating the data by more than one person (i.e., annotators) and applying validation methods used to measure inter-rater agreement among the annotators. Only seven studies adopted these validation methods; however, there were only two methods used. The first one was Cohen's kappa (Carletta, 1996), which is a statistical measure used to calculate the agreement among human raters who each classify n items into x categories (S8, S9, and S10). The second method resolves the differences between two annotators with a third annotator (S7, S11, S19, and S20).

4.2.2.2. Unsupervised learning. The USL algorithms were simply about portioning text documents into different parts or categories. Thus, in general, the USL algorithms do not require any prerequisites where a given document is classified based on its content without any pre-defined categories. However, some pre-defined data were needed to classify the NFRs, such as the NFR categories. Using the data is not necessary to perform the USL algorithms; however, it improves the classification performance of the NFRs. For example, S23 used USL without pre-defined data and got the worst performance compared to other studies that used pre-defined data.

These studies, S7, S15 and S24, used pre-defined data, including domain independent wordlists. S7 identified three different sets of wordlists that were derived from different sources, such as the ontology for software quality measurement and ISO/IEC 9126-1 (2001). taxonomy. S24 used one of the wordlists proposed by S7 called "exp2" and included six high-level NFRs and related terms for each category that were extracted from ISO/IEC 9126-1 (2001). S15 created a list containing NFR categories and their representative words.

4.2.3. Learning phase

This phase involves the application of ML algorithms to the identification of NFRs from the pre-processed text. Different types of ML algorithm have a different application process, described as follows.

4.2.3.1. Applying SL algorithms. In this phase, SL algorithms use NFRs' labelled datasets, which have been manually classified by type, to learn certain parameters (features, patterns or functions) of each type of NFR. In other words, these algorithms find the relationships between NFR statements (input) and their labels (output). Then, they use this information for further NFR classification tasks.

Of the 17 studies that used SL algorithms, only 2 used more than one ML algorithm. S13 used two separate classifiers (one vertical, one horizontal) to reduce the size of the dataset required for hierarchical classification. S11 used a combined classifier consisting of three SL classifiers to improve the performance the KNN classifier.

4.2.3.2. Applying USL algorithms. These algorithms use input requirement documents to drive structure by looking at the relationship between the inputs (i.e. textual requirements) themselves. In the related primary studies, only four (S7, S15, S23, and S24) used this type of learning. S7, S15, and S24 used lists of keywords including pre-defined NFRs' categories to determine which cluster belonged to which NFR category. Although the authors of S7 claimed that they used semi-supervised learning because the use of the keywords, which were not derived from the dataset being analysed, we categorized their work as unsupervised as the method that they used was quite similar to what proposed in S15 and S24. In addition, the use of keywords was necessary to define the categories and improve performance. The three studies that used keywords showed a better performance than what was seen in S23, which did not use additional data.

4.2.3.3. Applying SSL algorithms. These algorithms iteratively apply an SL algorithm with both labeled and unlabelled data. The labeled data were used to learn the classifier to classify further unlabelled requirements. Two of the three primary studies used this algorithm to improve the classifier's accuracy; S4 optionally used requirements that received feedback from the analysts as labeled requirements, while S21, which used active learning, manually classified the least confident output in each iteration as a training set until an acceptable accuracy was achieved. However, S20 used this algorithm to classify NFRs in users reviews within the App Store.

4.2.4. Evaluation phase

In this phase, the evaluation of the NFR classifier is conducted using various techniques to determine the developing classifier's

Table 7
NFR categories and their identification in the selected studies.

Study ID	Security related	Performance related	Usability related
S1	Security	Performance, Maintainability, Availability, Scalability, Legal	Usability, Operationality, Look and Feel
S2	Not provided	Not provided	Not provided
S3	Integrity	Performance, Maintainability, Portability, Dependability, Deployability.	Usability
S4	Security	Performance, Maintainability, Availability, Scalability, Legal, Portability	Usability, Functionality, Operationality, Look and Feel
S5	Security	Performance, Availability, Scalability, Maintainability, Legal, Palm Operational	Usability, Operationality, Look and Feel, Fitness
S6	Security Requirements, security-relevant sentences, and security-related sentences.	N/A	N/A
S7	Integrity, security	Maintainability, Portability, Efficiency, Reliability, Interoperability, Testability, traceability, accuracy, modifiability, modularity, correctness, performance, verifiability	Usability, Functionality, understandability, flexibility
S8	Security	Efficiency, reliability.	Functionality, usability/utility
S9	Security, Access control, Audit, Privacy	Availability, Capacity and Performance, Legal, Maintainability, Recoverability, Reliability.	Usability, look and feel, Operationality.
S10	Not provided	Not provided	Not provided
S11	Confidentiality, Integrity, Identification & Authentication, Accountability, Privacy	Availability	N/A
S12	Not provided	Not provided	Not provided
S13	Not provided	Not provided	Not provided
S14	Security	Performance, Reliability scalability	Usability
S15	Security, Safety, Privacy	Performance, Accuracy, Portability, Legal, Reliability, Availability, Interoperability.	Accessibility
S16	Authentication-Authorization, Access control, Cryptography- Encryption, Data integrity.	N/A	N/A
S17	Authentication-Authorization, Access control, Cryptography- encryption, Data integrity	N/A	N/A
S18	N/A	Efficiency (Time behaviour, Resource utilization)	Functionality (Suitability, Accuracy, Security), Usability (Operability, Understandability, Attractiveness).
S19	N/A	Reliability, Portability, Performance	Usability
S20	N/A	Reliability, Performance, Supportability	Functionality, Usability,
S21	Security	Reliability, Performance, Lifecycle, Capability	Usability, System Interface
S22	Security	Performance	Usability, Operationality
S23	Security	Availability, Maintainability, Operability, Performance, Scalability, Fault Tolerance, and Portability, Legal & Licensing	Usability, Look & Feel.
S24	N/A	Reliability, Efficiency, Maintainability, Portability	Functionality, Usability

effectiveness. Evaluation techniques and results are discussed in Section 4.3. Different ML algorithms use different strategies to evaluate the NFR classifiers; they are described in the literature as follows:

4.2.4.1. Evaluating STL algorithms. The input dataset of these algorithms is divided into two main parts. The first part is used in the learning phase (discussed in Section 4.2.2) and the other part in the testing phase. The testing dataset is not seen during the training phase to ensure that the classifications are not biased by previously viewed data. Then, the predicated labels are compared with the originals to measure the NFR classifiers' performance.

Approximately three-quarters of the reviewed primary studies (17 out of 24) used the *K*-fold cross-validation strategy to enhance the accuracy of the NFR classifiers. In this strategy, the data are randomly divided into various *K*-folds. Each *K*-fold has the same data size; one is used for testing, while the others are used for training. The learning algorithm is run *K* times, and the average of *K* results is calculated to produce a single result.

4.2.4.2. Evaluating USL algorithms. The evaluation of USL algorithms is tricky compared to that of SL algorithms as there are not pre-defined labels to measure whether the NFRs are classified correctly. Thus, the primary studies using these algorithms (S7, S15 and S23) created testing datasets including pre-classified NFRs. The performance of these algorithms was evaluated by comparing the predicated labels with the original ones.

One study (S23) also referenced other approaches to measure the quality of each cluster of NFRs, including cohesion (members in each cluster are close to each other) and separation (members of a cluster are far away from other clusters) methods. These techniques are related to the internal quality of each cluster, not their final output (i.e., the predicated categories).

4.2.4.3. Evaluating SSL algorithms. SSL algorithms are similar to SL algorithms, as they also require a labelled dataset for training phase. Moreover, SSL algorithms require a labelled dataset for the testing phase. This dataset is used in a similar way to SL algorithms.

4.3. Performance of the reported ML algorithms (RQ3)

RQ3: What measures have been used to evaluate the performance of the ML algorithms applied in these approaches? What are the performance results of these algorithms?

Of the 24 selected studies, only 20 of them (83%) have reported their performance measures. Table 8 summarizes the seven

Table 8
Performance measures and results found in 20 selected studies.

ML algorithm	Study ID	Precision	Recall	F-Score	Accuracy	Transductive accuracy	Inductive accuracy	AUC
SVM	S8	84%	84%	84.96%				
	S9	72.8%	54.4%	62%				
	S10	>60%	>80%					
	S12	69%	90%	78%				
	S21	73.9–92.4%	54.6–81.9%					
	S22	>70%	>70%		78.4%			
NB	S1	12.4%	76.7%					
	S4				>70%			
	S6	>80%	>90%					
	S7			≈43				60–80%
DT	S2	97.8%	100%		98.56			
	S16		80%					75%
	S19	71.4%	72%	71.8%				
K-NN	S11	82%	79%	80%				
HAC	S15	53%	83%					
RB	S18	98%	96%	97%				
Self-Training	S20					74.6%	71.3	
MVB	S23	90%	91%					
LDA	S24	66.7%	66.7%					

performance measures identified from these studies and the corresponding results.

An interesting observation we made (as shown in Table 8) is that although SVM, NB, and DT have been reported in multiple studies, their performance varies from study to study. For example, SVM performs well in S8, but not so well in S9. In S9, SVM has a relatively better precision result than the recall result; on the other hand, it has better recall results in S10 and S12.

Among the seven different performance measures, precision and recall are most popular, followed by F-Score. In what follows, we explain how these seven measures are used to measure the performance of the reported ML algorithms.

4.3.1. Precision and recall

Of the 20 studies that report their performance measures, 15 of them (75%) have used these two matrices to measure their ML algorithms. Precision measures the total number of correctly classified NFRs in respect to the number of NFRs retrieved. It is defined as $P =$ true positive/ (true positive + false positive). Recall measures the percentage of NFRs that were correctly classified and is defined as $R =$ true positives/(true positives + false negatives) (Cleland-Huang et al., 2007). True positive represents the number of correctly classified requirements, false positive represents the number of incorrectly classified requirements, true negative represents the number of requirements correctly not classified, and false negative is the number of requirements incorrectly not classified (Casamayor et al., 2010).

Precision and recall are often used together (Zhang, Yang, Wang, & Shu, 2011) and there is a trade-off between them. Precision ensures that all retrieved requirements are truly relevant, while recall focuses on retrieving all relevant requirements. It is arguable which value is more important to measure the classification results. S1, S6, and S7 focused on achieving a high recall as it was believed that rejecting irrelevant requirements manually from a set of retrieved requirements was easier than reading an entire document looking for missing requirements. On the other hand, S5 argued that precision was more important if recall was acceptable for automatic classification to avoid large amounts of irrelevant NFRs from being misclassified as relevant (false positive), which might frustrate users.

It can be seen from the values of precision and recall represented in Table 8 that there are some studies that achieved higher recall and lower precision, such as S1 and S15, while there are other studies that achieved higher precision and lower recall such as S9. The number of keywords in S1 was between 10 and 20, while in S5, it was between 100 and 1000. The number of keywords used in S5 might be the reason why S5 achieved a lower recall and higher precision score in comparison with S1. On the other hand, there are studies that achieved high recall and high precision, which means that all the relevant requirements were retrieved correctly. The highest value of recall and precision was achieved by S2; however, this study used a ML approach to detect NFRs and could not classify them.

4.3.2. F-Score

This is the harmonic mean of precision and recall, and it is defined as F-Score $= 2 \times (P \times R)/(P + R)$ (Riaz, King, Slankas, & Williams, 2014). Seven of the primary studies applied the F-Score. However, the majority of them did not provide the reasons for applying this measure and only S9 and S20 provided reasons. S20 argued that this value is widely used for information retrieval tasks. S9 used this measure to combine both precision and recall as it is known that F-Score corresponds to the trade-off value of precision and recall (Mladeni, Brank, Grobelnik, & Milic-Frayling, 2004) and then used the value as an indicator for performance evaluation and comparison. S7 used this measure for readers who were not familiar with the other measure that they applied, which was a receiver operating characteristic (ROC) analysis.

4.3.2.1. Confusion matrix. This is used to analyse the classification results showing the number of correct classifications (true positive, true negative) and incorrect classifications (false positive, false negative). Only four studies provide the confusion matrix for their classifiers (S1, S2, S8, and S18). S1 and S8 mentioned that the reasons for computing this matrix were to provide useful classification results and identify room for more improvement.

4.3.2.2. Accuracy. This is the measure of the numbers of correct classifications divided by the total number of classifications. It is defined as accuracy = (true positives + true negatives)/(true positives + true negatives false positives + false negatives). This measure was applied by four studies (S2, S4, S13, and S22); however, none provided a reason for computing the accuracy of their work.

4.3.2.3. Transductive and inductive accuracy. Transductive accuracy measures the accuracy of predicting unlabelled instances in training, while inductive accuracy measures the accuracy of predicting unseen test data (testing data) (Deocadez et al., 2017). Both of these values were used to measure the accuracy of the SSL algorithms in one study S20 as the authors of this studies believed

that there were two types of learning in SSL: one with testing data and the other with unlabelled data.

4.3.2.4. ROC value. Sometimes known as the "area under the curve" or AUC, this method visualizes the performance of classifiers as graphs; the *x*-axis represents '1-specificity' (false positive rate), while the *y*-axis represents sensitivity (true positive rate). (Hindle, Ernst, Godfrey, & Mylopoulos, 2013; Jindal et al., 2016) ROC represents the accuracy of the performance in relation to new instances. Only two studies adopted this method to compute the performance of their classifiers (S7 and S16). The authors of S 7 (Hindle et al., 2013) argued that ROC suffers from less bias compared to other measures, such as the F-Score, which skews toward the positive class, especially in case of an imbalanced class.

5. Key findings, limitations and open challenges

In this section, we derive the key research findings from our review results, and discuss the limitations of the surveyed approaches and open challenges.

5.1. Key findings

The key findings that we have identified from our review are:

- ML-based approaches have generally performed well, achieving an accuracy of more than 70% in detecting and classifying NFRs.
- Overall, SL algorithms perform better than USL algorithms. Among the surveyed SL algorithms, SVM and NB have the best performance, with SVM being the most frequently used algorithm.
- ML algorithms perform better when individual words are used as features, rather than phrases. Similarly, ML algorithms tend to produce the best results when the original word form is used, instead of stemming and lemmatization.

These findings show that, in spite of being still at the early state of research, ML-based approaches have already produced promising results for identification and classification of NFRs. This is a positive prospect.

5.2. Limitations

During our review, we have observed a number of limitations specific to reporting and evaluating ML approaches. These limitations are described as follows.

5.2.1. Lack of reporting standards

Both SL and USL algorithms require pre-defined data: SL requires labelled data, while USL needs pre-defined categories and associated terms to achieve high performance. However, before applying an ML (SL or USL) algorithm, ML approaches must take some steps to pre-process the input requirements document and identify relevant features for the ML algorithm. However, this process has not been clearly described in the reviewed approaches. The majority of the studies treat ML approaches as 'black boxes' and provide no details on how these approaches work. This makes our review quite difficult. To ensure review consistency, we used a general process (Fig. 3) as a common structure to assess individual studies. We also dug into the reported results to deduce how their approaches work.

5.2.2. Lack of evaluation standards

Although the majority of the studies provided evaluation results (20 out 24), 70% of them (14 out 20) did not explain how and why they used a certain evaluation method. For example, regarding precision and recall, these studies did not mention which was more important: high precision and low recall, or high recall and low precision. Furthermore, they did not say why they provided F-Score and not accuracy, or vice versa. From this, it can be concluded that the majority of the studies did not know why they used a specific method that was different from the others or how to explain their results.

5.3. Open challenges

Through our reflection on the review, we have identified three open challenges faced by researchers, elaborated as follows.

5.3.1. Lack of shared training datasets

One open challenge of the current research is the lack of the training data. A pre-classified dataset is required to apply SL algorithms; developing such a dataset is time-consuming and requires much effort. It requires the collection of real-life NFRs, the classification of these requirements, and the validation of the classifications.

Unfortunately, there are few shared datasets available, which were provided by academics and students. For example, Promise[1] and Concordia[2]. Promise was used most frequently by the primary studies (11/24 or 45%). However, Concordia, which was written in a specific format, was less used (2/24 or 8%). These datasets only contain a total of 326 NFRs, which are far few for ML. Building a shared requirement corpus can encourage the researchers to conduct more experiments and provide benchmarks for future performance.

5.3.2. Lack of standard definition, classification and representation of NFRs

As stated in the introduction, although NFRs are considered import and critical to successful software projects, there is still no consensus in the RE community what non-functional requirements are, and how we should classify and represent them (Glinz, 2007). Consequently, there exist a diverse range of terms (e.g., property, characteristic, attribute, quality, constraint, and performance) to define NFRs, leading to not only terminological, but also major conceptual discrepancies (Glinz, 2007).

The diversity of NFR definitions inevitably leads to divergent classification of NFRs. As our review results show (Table 7), each of the 24 studies uses a different set of NFR categories and there is little agreement between them. For example, portability was identified as part of the lifecycle category in S21, which define the project development requirements and constraints rather than the software itself. However, portability was identified as the main category in S19. It was defined as the ability to transform the system from one environment to another.

In addition to these definition and classification problems, the third problem is representation: There is no consensus how NFRs should be expressed and at what level of detail (Glinz, 2007). One consequence is that a NFR may become a FR, depending on how you express it. Consider the following example given by Glinz (2007): the requirement "The probability for successful access to the customer data by an unauthorized person shall be smaller than 10^{-5}" is a NFR. However, if we refine this requirement to "The database shall grant access to the customer data only to those users that have been authorized by their user name and password", it becomes a FR.

These problems make ML algorithms extremely challenging, as they can only be trained and used for specific NFRs with specific terms and categories. They cannot be generalizable or scalable. This

[1] https://terapromise.csc.ncsu.edu/!/#repo/view/head/requirements/nfr.
[2] http://www.semanticsoftware.info/non-testable-nfr-detector.

problem makes the automated classification of requirements more complex and error-prone. Furthermore, these problems also make it difficult to compare the performance of similar approaches and to set performance benchmarks.

5.3.3. Feature identification and selection

Extracting useful features from requirements documents is a difficult task. The techniques used by the primary studies do not always provide meaningful features. Although the proposed classifiers showed high performance, many of the features used by these classifiers were either insufficient or not appropriate. In the SL approaches, the features used were either meaningless or shared between different NFR categories. For example, in S1, 'take' and 'user' under the performance category were meaningless features, and the word 'product' was identified in operational, scalability and security categories. In addition, S10 included many trivial features, such as '10′, '0′ and 'increase' in performance, and 'word', 'let' and 'voice' in usability. On the other hand, in USL approaches, the features were either too abstract or meaningless. Examples of abstract features are those adapted by S15, such as 'security' and 'secure' for security. Furthermore, in S7, 'mouse' and 'human' can be considered as meaningless features for the usability category.

These meaningless features can significantly increase the number of false positives and affect the performance of the classifiers. In addition, this indicates that the proposed classifiers were limited in recognizing and retrieving NFRs from the domain for which they had been trained (i.e., overfitting). We also believe that these algorithms may not be able to be applied with a different writing style in the same domain.

In addition, the requirements sentences are typically shorter than sentences used in everyday language. This leads to low word co-occurrence and sparse features. This problem was addressed by extending short requirements statements with semantically similar words, as discussed in the study S19.

Therefore, we believe that identifying useful features for NFRs is an open challenging and needs further investigations.

6. Threats to the review validity

In this section, the threats to the results of this review are discussed. To organize this section the threats are classified according to the review process as described below:

6.1. Study identification

In this review, we used the snowballing approach rather than the traditional approach to identify the primary studies (database search). The main reason behind avoiding the traditional approach is due to the large number of false positive results returned, which required a lot of time to select and screen. The snowballing approach allows us to carefully select a seed set and then use snowballing to identify further relevant studies around this set. This may produce a more targeted set of papers. Nevertheless, there is still the possibility of misidentifying important studies. To overcome this threat, we formulated different search strings to identify the initial set in such a way as to return highly cited papers. We used different online database for ensuring diversity in studies identified and avoiding publisher bias.

6.2. Inclusion and exclusion

Some of the primary studies did not clearly describe their objectives, contribution, and research design. This made the inclusion/exclusion process difficult and increased the possibility of excluding relevant studies. In addition, some studies used difficult and complicated ways to describe their objectives, adopted approaches and results. This could have led to misunderstandings and an inaccurate selection of studies. In order to mitigate these threats, the decision regarding which studies to include in this review was discussed between the authors.

6.3. Data extraction

In some of the primary studies, important information was not clear for answering the research questions. Therefore, other resources were analysed to ensure that the correct data were extracted. These resources included related research that was mentioned in these studies or written by the same authors about similar topics.

7. Conclusions, implications of the review and directions for future research

In this paper, we have reported a systematic review of 24 carefully selected ML approaches used for identifying and classifying NFRs in requirements documents. We have asked a number of research questions related to what ML algorithms have been used in these approaches, how these approaches work and what performance measures have been used to evaluate these approaches. We have sought to answer these questions and provided a systematic understanding of the reported ML approaches.

The overall conclusion that stems from our review results is that using ML algorithms to identify NFRs is not an easy task, as it requires appropriate methods to extract features and to classify the text. For SL, a pre-classified dataset is also required. On the other hand, in comparison with the traditional manual classification technique, using ML algorithms can help improve the process of identifying NFRs, as it can reduce human effort and errors. This review found that ML algorithms, especially SL, hold great promise in RE, as they can achieve a high level of performance.

Our review, being of an exploratory and interpretive nature, raises a number of opportunities for future research, both in terms of addressing the open challenges identified, and developing new and better learning approaches. In what follows, we briefly discuss the implications derived from our findings and some of the directions for future research.

First, our review has identified the lack of labelled datasets as the first open challenge. Datasets are an important factor in determining the classification accuracy, as ML models are trained on the data. The quality, quantity and variety of training datasets can improve the performance of NFRs classifier as well as the quality of the classification results. Shared datasets need to be built to take full advantage of supervised ML algorithms and to encourage future research and proper comparison. We believe that sharing high quality datasets and building supportive tools are fundamental for further growth and development in the RE community.

Second, in terms of the open challenge concerning the lack of standard definition, classification and representation of NFRs, the RE community must work together to provide a solution. This may entail adopting an existing quality model such as ISO/IEC 9126-1 (2001). Defining such a standard is the precondition for the successful application of ML approaches.

Third, our review has shown that the majority of studies build ML classifiers on obvious keywords. However, using keywords are not sufficient, as they are primitive and could lead to high overfitting, making classifiers not adaptable to different domains or even different requirements documents. Future research should consider other semantic features in classifying NFRs, such as domain concepts and semantic roles of terms on sentences.

Fourth, our review has uncovered the limitations of the selected studies in terms of their reporting and evaluation methods. We

suggest that standard reporting and evaluation procedures should be developed to help researchers to report and evaluate their studies. We believe that sound methodologies can facilitate the understanding and the reproducibility of the studies.

In conclusion, we believe that our review is timely, as it reports on the results of recent advances in RE research. More significantly, using ML for RE provides exciting opportunities to develop novel expert and intelligent systems to support RE tasks and processes. We conclude this review by calling for the close collaboration between RE and ML communities, to address the open challenges facing the development of real-world applications of ML to RE.

Conflict of interest

The authors Manal Binkhonain and Liping Zhao certify that they have NO affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements), or non-financial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.

CRediT authorship contribution statement

Manal Binkhonain: Data curation, Formal analysis, Investigation, Validation, Visualization, Writing - original draft, Writing - review & editing. **Liping Zhao:** Conceptualization, Formal analysis, Methodology, Supervision, Validation, Visualization, Writing - review & editing.

Acknowledgments

We are really grateful for the expert comments and excellent advice we have received from our reviewers. Their thoughtful guidance has really helped in strengthening the manuscript. In addition, Manal would like to thank King Saud University for giving her the chance to study Ph.D. at the University of Manchester.

Appendix A: The 24 selected studies

S1 Cleland-Huang, J., Settimi, R., Zou, X., & Solc, P. (2007). Automated classification of non-functional requirements. Requirements Engineering, 12(2), 103–120.

S2 Hussain, I., Kosseim, L., & Ormandjieva, O. (2008, June). Using linguistic knowledge to classify non-functional requirements in SRS documents. Paper presented at the International Conference on Application of Natural Language to Information Systems, London, UK.

S3 G. Gokyer, S. Cetin, C. Sener, and M. T. Yondem. (2008, October). Non-functional requirements to architectural concerns: ML and NLP at crossroads. In Proceedings of the 2008 the 3rd international conference on software engineering advances, Sliema, Malta.

S4 Casamayor, A., Godoy, D., & Campo, M. (2010). Identification of non-functional requirements in textual specifications: A semi-supervised learning approach. Information and Software Technology, 52(4), 436–445.

S5 Zhang, W., Yang, Y., Wang, Q., & Shu, F. (2011, July). An empirical study on classification of non-functional requirements. Paper presented at the Twenty-Third International Conference on Software Engineering and Knowledge Engineering, Miami Beach, USA.

S6 Knauss, E., Houmb, S., Schneider, K., Islam, S., & Jürjens, J. (2011, March). Supporting requirements engineers in recognising security issues. Paper presented at the International Working Conference on Requirements Engineering: Foundation for Software Quality, Essen, Germany.

S7 Hindle, A., Ernst, N. A., Godfrey, M. W., & Mylopoulos, J. (2013). Automated topic naming. Empirical Software Engineering, 18(6), 1125–1155.

S8 Rashwan, A., Ormandjieva, O., & Witte, R. (2013, July). Ontology-based classification of non-functional requirements in software specifications: a new corpus and svm-based classifier. Paper presented at IEEE 37th Annual Computer Software and Applications Conference, Kyoto, Japan.

S9 Slankas, J., & Williams, L. (2013, May). Automated extraction of non-functional requirements in available documentation. Paper presented at the 2013 1st International Workshop on Natural Language Analysis in Software Engineering, San Francisco, USA.

S10 Ott, D. (2013, April). Automatic requirement categorization of large natural language specifications at Mercedes-Benz for review improvements. Paper presented at the International Working Conference on Requirements Engineering: Foundation for Software Quality, Essen, Germany.

S11 Riaz, M., King, J., Slankas, J., & Williams, L. (2014, August). Hidden in plain sight: Automatically identifying security requirements from natural language artifacts. In Proceedings of IEEE International Conference on Requirements Engineering Conference, Karlskrona, Sweden.

S12 Knauss, E., & Ott, D. (2014, April). (Semi-)automatic Categorization of Natural Language Requirements. Paper presented at the International Working Conference on Requirements Engineering: Foundation for Software Quality, Essen, Germany.

S13 Nguyen, T. H., Grundy, J., & Almorsy, M. (2015, August). Rule-based extraction of goal-use case models from text. Paper presented at the Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, Bergamo, Italy

S14 Maiti, R. R., & Mitropoulos, F. J. (2015, June). Capturing, eliciting, predicting and prioritizing (CEPP) non-functional requirements metadata during the early stages of agile software development. Paper presented at the SoutheastCon 2015, Fort Lauderdale, USA.

S15 Mahmoud, A., & Williams, G. (2016). Detecting, classifying, and tracing non-functional software requirements. Requirements Engineering, 21(3), 357–381.

S16 Jindal, R., Malhotra, R., & Jain, A. (2016, November). Automated classification of security requirements. Paper presented at the Advances in Computing, Communications and Informatics, Jaipur, India.

S17 Malhotra, R., Chug, A., Hayrapetian, A., & Raje, R. (2016, February). Analyzing and evaluating security features in software requirements. Paper presented at the Innovation and Challenges in Cyber Security, Noida, India.

S18 Singh, P., Singh, D., & Sharma, A. (2016, September). Rule-based system for automated classification of non-functional requirements from requirement specifications. Paper presented at the Advances in Computing, Communications and Informatics, Jaipur, India.

S19 Lu, M., & Liang, P. (2017, June). Automatic Classification of Non-Functional Requirements from Augmented App User Reviews. Proc. 21st International Conference on Evaluation and Assessment in Software Engineering, Karlskrona, Sweden.

S20 Deocadez, R., Harrison, R., & Rodriguez, D. (2017, September). Automatically Classifying Requirements from App Stores: A Preliminary Study. Paper presented at IEEE 25th International Requirements Engineering Conference Workshops, Lisbon, Portugal.

S21 Li, C., Huang, L., Ge, J., Luo, B., & Ng, V. (2017). Automatically Classifying User Requests in Crowdsourcing Requirements Engineering. Journal of Systems and Software. 138, 108–123.

S22 Kurtanović, Z., & Maalej, W. (2017, September). Automatically Classifying Functional and Non-functional Requirements Using Supervised Machine Learning. Paper presented at IEEE 25th International Requirements Engineering Conference Workshops, Lisbon, Portugal.

S23 Abad, Z. S. H., Karras, O., Ghazi, P., Glinz, M., Ruhe, G., & Schneider, K. (2017, September). What Works Better? A Study of Classifying Requirements. Paper presented at IEEE 25th International Requirements Engineering Conference Workshops, Lisbon, Portugal.

S24 Zou, J., Xu, L., Yang, M., Zhang, X., & Yang, D. (2017). Towards comprehending the non-functional requirements through Developers' eyes: An exploration of Stack Overflow using topic analysis. Information and Software Technology, 84, 19–32.

References

Abad, Z. S. H., Karras, O., Ghazi, P., Glinz, M., Ruhe, G., & Schneider, K. (2017). What works better? A study of classifying requirements. In *Proceedings of the IEEE twenty-fifth international requirements engineering conference (RE)* September.

Agarwal, B., & Mittal, N. (2014). Text classification using machine learning methods – A survey. In B. V. Babu, A. Nagar, K. Deep, M. Pant, J. C. Bansal, K. Ray, & U. Gupta (Eds.), *Proceedings of the second international conference on soft computing for problem solving* December.

Alpaydin, E. (2014). *Introduction to machine learning*. Cambridge, Massachusetts: MIT press.

Kitchenham B., Charters S. (2007). Guidelines for performing systematic literature reviews in software engineering. Technical report, Ver. 2.3, EBSE.

Bennett, K. P., & Bredensteiner, E. J. (2000). Duality and geometry in SVM classifiers. In *Proceedings of the seventeenth international conference on machine learning* July.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, 993–1022.

Carletta, J. (1996). Assessing agreement on classification tasks: The kappa statistic. *Computational Linguistics, 22*(2), 249–254.

Casamayor, A., Godoy, D., & Campo, M. (2010). Identification of non-functional requirements in textual specifications: A semi-supervised learning approach. *Information and Software Technology, 52*(4), 436–445.

Cavnar, W. B., & Trenkle, J. M. (1994). N-gram-based text categorization. *Ann Arbor Mi, 48113*(2), 161–175.

Chung, L., Nixon, B. A., Yu, E., & Mylopoulos, J. (2012). Non-functional requirements in software engineering, onceptual modeling: Foundations and applications. In A. T. Borgida, V. K. Chaudhri, P. Giorgini, & E. S. Yu (Eds.). *Conceptual modeling: Foundations and applications. Lecture notes in computer science*: 5. Berlin, Heidelberg: Springer.

Cleland-Huang, J., Settimi, R., Zou, X., & Solc, P. (2007). Automated classification of non-functional requirements. *Requirements Engineering, 12*(2), 103–120.

De Marneffe, M.-C., & Manning, C. D. (2008). The stanford typed dependencies representation. In *Proceedings of the workshop on cross-framework and cross-domain parser evaluation (Coling 2008)* August.

Deocadez, R., Harrison, R., & Rodriguez, D. (2017). Automatically classifying requirements from app stores: a preliminary study. In *Proceedings of the IEEE twenty–fifth international requirements engineering conference workshops* September.

Dixon-Woods, M., Agarwal, S., Jones, D., Young, B., & Sutton, A. (2005). Synthesising qualitative and quantitative evidence: A review of possible methods. *Journal of Health Services Research & Policy, 10*(1), 45–53.

Dobra, A. (2009). Decision tree classification. In L. Liu, & M. T. ÖZsu (Eds.), *Encyclopedia of database systems* (pp. 765–769). Boston, MA: Springer US.

Glaser, B. G., Strauss, A. L., & Strutzel, E. (1968). The discovery of grounded theory; strategies for qualitative research. *Nursing Research, 17*(4), 364.

Glinz, M. (2007). On non-functional requirements. In *Proceedings of the fifteenth IEEE international requirements engineering conference* October.

Harding, J. (2013). *Qualitative data analysis from start to finish*. London: Sage.

Hindle, A., Ernst, N. A., Godfrey, M. W., & Mylopoulos, J. (2013). Automated topic naming. *Empirical Software Engineering, 18*(6), 1125–1155.

Ikonomakis, M., Kotsiantis, S., & Tampakas, V. (2005). Text classification using machine learning techniques. *WSEAS Transactions on Computers, 4*(8), 966–974.

Jindal, R., Malhotra, R., & Jain, A. (2016). Automated classification of security requirements. In *Proceedings of the 2016 international conference on advances in computing, communications and informatics* September.

Jivani, A. G. (2011). A comparative study of stemming algorithms. *International Journal of Computer Applications in Technology, 2*(6), 1930–1938.

Kayed, A., Hirzalla, N., Samhan, A. A., & Alfayoumi, M. (2009). Towards an ontology for software product quality attributes. In *Proceedings of the Internet and web applications and services* May.

Khan, A., Baharudin, B., Lee, L. H., & Khan, K. (2010). A review of machine learning algorithms for text-documents classification. *Journal of Advances in Information Technology, 1*(1), 4–20.

Knauss, E., Houmb, S., Schneider, K., Islam, S., & Jürjens, J. (2011). Supporting requirements engineers in recognising security issues. In *Proceedings of the international working conference on requirements engineering: foundation for software quality* May.

Ko, Y., Park, S., Seo, J., & Choi, S. (2007). Using classification techniques for informal requirements in the requirements analysis-supporting system. *Information and Software Technology, 49*(11–12), 1128–1140.

Korde, V., & Mahender, C. N. (2012). Text classification and classifiers: A survey. *International Journal of Artificial Intelligence & Applications, 3*(2), 85.

Kotsiantis, S. B., Zaharakis, I. D., & Pintelas, P. E. (2006). Machine learning: A review of classification and combining techniques. *Artificial Intelligence Review, 26*(3), 159–190.

Lewis, D. D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. In *Proceedings of the tenth European conference on machine learning* April.

Li, C., Huang, L., Ge, J., Luo, B., & Ng, V. (2017). Automatically classifying user requests in crowdsourcing requirements engineering. *Journal of Systems and Software, 138*, 108–123.

Mahmoud, A. (2015). An information theoretic approach for extracting and tracing non-functional requirements. In *Proceedings of the requirements engineering conference* August.

Mahmoud, A., & Williams, G. (2016). Detecting, classifying, and tracing non-functional software requirements. *Requirements Engineering, 21*(3), 357–381.

Meth, H., Brhel, M., & Maedche, A. (2013). The state of the art in automated requirements elicitation. *Information and Software Technology, 55*(10), 1695–1709.

Mladeni, D., Brank, J., Grobelnik, M., & Milic-Frayling, N. (2004). Feature selection using linear classifier weights: interaction with classification models. In *Proceedings of the twenty-seventh annual international ACM SIGIR conference on research and development in information retrieval* July.

Muhammad, I., & Yan, Z. (2015). Supervised machine learning approaches: A survey. *ICTACT Journal on Soft Computing, 5*(3), 946–952.

Nigam, K., McCallum, A., & Mitchell, T. (2006). Semi-supervised text classification using EM. In O. Chapelle, B. Scholkopf, & A. Zien (Eds.), *Semi-Supervised Learning* (pp. 33–56). MIT Press Scholarship Online: August 2013. doi:10.7551/mitpress/9780262033589.001.0001.

Nuseibeh, B. (2001). Weaving together requirements and architectures. *Computer, 34*(3), 115–119.

Ott, D. (2013). Automatic requirement categorization of large natural language specifications at mercedes-benz for review improvements. In *Proceedings of the international working conference on requirements engineering: foundation for software quality* April.

Rago, A., Marcos, C., & Diaz-Pace, J. A. (2013). Uncovering quality-attribute concerns in use case specifications via early aspect mining. *Requirements Engineering, 18*(1), 67–84.

Rashwan, A., Ormandjieva, O., & Witte, R. (2013). Ontology-based classification of non-functional requirements in software specifications: A new corpus and svm-based classifier. In *Proceedings of the IEEE thirty-seventh annual computer software and applications conference* July.

Riaz, M., King, J., Slankas, J., & Williams, L. (2014). Hidden in plain sight: Automatically identifying security requirements from natural language artifacts. In *Proceedings of the twenty-second international requirements engineering conference (RE)* August.

Rosenberg, C., Hebert, M., & Schneiderman, H. (2005). Semi-supervised self-training of object detection models. In *Proceedings of the IEEE workshops on applications of computer vision* January.

Sharma, V. S., Ramnani, R. R., & Sengupta, S. (2014). A framework for identifying and analyzing non-functional requirements from text. In *Proceedings of the forth international workshop on twin peaks of requirements and architecture* June.

Su, J., Zhang, H., Ling, C. X., & Matwin, S. (2008). Discriminative parameter learning for bayesian networks. In *Proceedings of the twenty-fifth international conference on machine learning* July.

Vapnik, V. N., & Chervonenkis, A. Y. (2015). On the uniform convergence of relative frequencies of events to their probabilities. *Measures of Complexity*, 11–30.

Vlas, R., & Robinson, W. N. (2011). A rule-based natural language technique for requirements discovery and classification in open-source software development projects. In *Proceedings of the forty-forth Hawaii international conference system sciences (HICSS)* January.

Wang, J., Luo, S.-W., & Zeng, X.-H. (2008). A random subspace method for co-training. In *Proceedings of the neural networks, 2008 (IEEE world congress on computational intelligence)* June.

Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the eighteenth international conference on evaluation and assessment in software engineering* May.

Yan, X., Guo, J., Lan, Y., & Cheng, X. (2013). A biterm topic model for short texts. In *Proceedings of the twenty-second international conference on world wide web* May.

Yang, X., Zhao, L., Wang, X., Wang, Y., Sun, J., & Cristoforo, A. J. (2012). Satisfying quality requirements in the design of a partition-based, distributed stock trading system. *Software: Practice and Experience, 42*(2), 131–157.

Zhang, Q., Zhang, L., Dong, S.-B., & Tan, J.-H (2005). Document indexing in text categorization. In *Proceedings of the machine learning and cybernetics* August.

Zhang, W., Yang, Y., Wang, Q., & Shu, F. (2011). An empirical study on classification of non-functional requirements. In *Proceedings of the twenty-third international conference on software engineering and knowledge engineering* July.

Zhu, X. (2005). *Semi-supervised learning literature survey. technical report 1530, computer sciences*. University of Wisconsin-Madison.

Zou, J., Xu, L., Yang, M., Zhang, X., & Yang, D. (2017). Towards comprehending the non-functional requirements through developers' eyes: An exploration of stack overflow using topic analysis. *Information and Software Technology, 84*, 19–32.