Information and Software Technology 52 (2010) 436–445

Contents lists available at ScienceDirect

Information and Software Technology

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

Identification of non-functional requirements in textual specifications: A semi-supervised learning approach

Agustin Casamayor [*], Daniela Godoy, Marcelo Campo

ISISTAN Research Institute, UNICEN University, Campus Universitario, Paraje Arroyo Seco, B7001BBO, Tandil, Bs. As., Argentina
CONICET, National Council for Scientific and Technical Research, C1033AAJ, Bs. As., Argentina

A R T I C L E   I N F O

Article history:
Received 25 March 2009
Received in revised form 5 September 2009
Accepted 27 October 2009
Available online 10 November 2009

Keywords:
Non-functional requirements
Requirement classification
Semi-supervised text learning

A B S T R A C T

Context: Early detection of non-functional requirements (NFRs) is crucial in the evaluation of architectural alternatives starting from initial design decisions. The application of supervised text categorization strategies for requirements expressed in natural language has been proposed in several works as a method to help analysts in the detection and classification of NFRs concerning different aspects of software. However, a significant number of pre-categorized requirements are needed to train supervised text classifiers, which implies that analysts have to manually assign categories to numerous requirements before being able of accurately classifying the remaining ones.
Objective: We propose a semi-supervised text categorization approach for the automatic identification and classification of non-functional requirements. Therefore, a small number of requirements, possibly identified by the requirement team during the elicitation process, enable learning an initial classifier for NFRs, which could successively identify the type of further requirements in an iterative process. The goal of the approach is the integration into a recommender system to assist requirement analysts and software designers in the architectural design process.
Method: Detection and classification of NFRs is performed using semi-supervised learning techniques. Classification is based on a reduced number of categorized requirements by taking advantage of the knowledge provided by uncategorized ones, as well as certain properties of text. The learning method also exploits feedback from users to enhance classification performance.
Results: The semi-supervised approach resulted in accuracy rates above 70%, considerably higher than the results obtained with supervised methods using standard collections of documents.
Conclusion: Empirical evidence showed that semi-supervision requires less human effort in labeling requirements than fully supervised methods, and can be further improved based on feedback provided by analysts. Our approach outperforms previous supervised classification proposals and can be further enhanced by exploiting feedback provided by analysts.

1. Introduction

Non-functional requirements (NFRs) constrain the behavior and development of a software system as they specify overall qualities or attributes the resulting system should have. Examples of NFRs include security, performance, availability, extensibility and portability, among others. NFRs play a critical role in architectural design, so that the early detection of these expected software quality attributes is crucial in order to take them into consideration starting from initial design decisions. Despite this may seem to be an easy task, it can be really difficult and time consuming, especially considering that most NFRs are somehow hidden across requirements that mainly specify functionality, and could be easily ignored.

Methods for the elicitation of non-functional requirements often involve the use of questionnaires, checklists or templates for inquiring stakeholders concerning quality issues [12,22]. Further classification of the elicited requirements could be assisted by domain-specific knowledge (i.e. the Language Extended Lexicon or LEL, which uses the vocabulary of the domain together with a NFR knowledge base [10]) or by ontologies representing quality aspects that should be taken into consideration [2,20].

For textual requirements expressed in natural language, the detection and classification of NFRs have been approached using supervised learning techniques [7,8] and, in some cases, integrated into recommender systems [6,4]. From the supervised learning perspective, a set of previously categorized requirements is used

[*] Corresponding author. Address: ISISTAN Research Institute, UNICEN University, Campus Universitario, Paraje Arroyo Seco, B7001BBO, Tandil, Bs. As., Argentina. Tel.: +54 2293 439681/439682x42; fax: +54 2293 439681x52.
E-mail address: acasamay@exa.unicen.edu.ar (A. Casamayor).

0950-5849/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2009.10.010

to train a classifier to recognize the type of novel requirements (i.e. security, portability, etc.). The main shortcoming of supervised classification is the high number of examples or already categorized requirements needed to generalize a hypothesis with certain level of confidence. Even though requirements belonging to past software projects can be used for training, the variability in the vocabulary employed by different elicitation teams and system domains prevent the resulting text classifiers from obtaining accurate results.

In this paper we introduce a semi-supervised approach [5] for the automatic detection and classification of NFRs aiming at exploiting uncategorized requirements to reduce the number of examples needed for learning. For text classification, unlabeled data (requirements with no assigned class labels) used in conjunction with a small amount of labeled data (requirements with assigned class labels) can produce a considerable improvement in learning accuracy taking advantage of natural word co-occurrence in texts [32]. The resulting classifiers can be used for requirement analysis in software projects in which analysts are able to provide an initial set of categorized requirements detected, for example, during interviews or using an alternative method. In this scenario, analysts will receive suggestions about a possible classification of the remaining requirements and they can give feedback to refine classification in an iterative process.

In the experiments conducted and summarized in this paper, high performance rates obtained with the semi-supervised classifiers materialized through the Expectation Maximization strategy [32] suggest that this approach may be successfully used for requirement analysis in software development projects, remarkably reducing the effort of manual identification and classification of requirement documents written in natural language. In addition, the proposed approach outperforms several supervised algorithms it was compared to.

The remaining of this paper is organized as follows: Section 2 discusses related works in the use of linguistic techniques for analyzing textual requirements and detecting different types of non-functional requirements. Section 3 presents the proposed approach to semi-supervised categorization of textual requirements. Empirical evaluation of this approach is summarized in Section 4. Finally, concluding remarks are stated in Section 5.

2. Related works

Informal textual descriptions written in natural language are a common means for specifying requirements in early phases of software projects [30]. Numerous attempts have been made to construct automatic tools for assisting developers during the analysis of textual specifications of requirements. Both information retrieval (IR) and natural language processing (NLP) techniques have been applied in the development of tools supporting more efficient (semi-) automatic requirement analysis.

Park et al. [36] proposed a requirement support system that evaluates the resemblance of requirement sentences using similarity measures stemming from IR field to identify possible redundancies and inconsistencies as well as to detect potentially ambiguous requirements. *ReqSimile* [33] supports the manual linkage between customer wishes and product requirements by suggesting potential links based on standard query-retrieval techniques. In this system it is assumed that customer wishes and product requirements refer to the same functionality with the same terminology, so that potential links for an incoming requirement are obtained according to their similarity with preexisting requirements. Two-Tiered Clustering (TTC) algorithm [35] indexes and clusters requirement specifications by functionality. Reuse-Assisted Requirements Elicitation (RARE) [9] implements a scheme that combines natural language processing, in which texts are parsed for building a semantic network with the assistance of a domain-mapping thesaurus, with faceted classification for the analysis and refinement of requirements.

The problem of assessing the quality of textual requirements has also been approached using linguistic techniques. Experiences of using lightweight formal methods for partial validation of requirement documents show that not even simple errors but also more subtle ones, sometimes overlooked by a human inspection, can be detected [16]. Likewise, the use of logic for identifying and analyzing inconsistencies in requirements from multiple stakeholders has been found to be effective in a number of studies [17]. Fantechi et al. [14] discussed the application of linguistic techniques aiming to collect quality metrics and spot defects related to the inherent ambiguity of textual descriptions. NLP techniques have also been explored as a means for bridging the gap between informal and formal requirement specifications in several works [23,15].

Text categorization methods applied to requirement documents can also provide a good basis for efficient requirements analysis. Ko et al. [24,25] propose an automatic requirement classification method to be applied in a Web-based analysis-supporting system. To automatically classify the collected informal requirements into several views or topics, the system requires as input a set of words representing the viewpoint of each analyst. This initial set is afterwards expanded based on co-occurrence of words in the requirement texts to construct topic centroids allowing the classification of novel requirements. The main shortcoming of this method is its reliance on analysts for extracting topic words for the different views of the software to be developed. Ormandjieva et al. [34] applied text classification techniques to build a system for automatic detection of ambiguity in requirements based on several quality indicators defined in a pre-defined quality model.

More recently, the problem of detection and classification of NFR requirements was tackled from a supervised classification point of view [7,8]. In these works a NFR classifier uses a training set of pre-classified requirements to discover a set of weighted indicator terms for each NFR type, i.e. security, performance, etc. Hence, the likelihood of a new requirement to suit a certain NFR type is computed as a function of the occurrence of the corresponding indicator terms within the text of this requirement. The NFR classifier outperforms the results of other supervised classifiers such as naïve Bayes and standard decision trees, even considering different schemes for feature subset selection [19]. These ideas were taken even further by designing an automatic speech recognition technique for capturing non-functional requirements during meetings with stakeholders using a context-free grammar to recognize the indicator terms [43]. The principal drawback of applying supervised methods to NFR detection is related to the amount of pre-categorized requirements needed to reach good levels of precision in the classification process. The NFR classifier uses data from past projects to classify novel requirements in ongoing projects. However, the use of distinctive vocabulary, domain terminology and writing styles across different projects as well as requirement elicitation teams hinder the application of this method. Conversely, the approach proposed in this paper iteratively classifies requirements gathered for a single project starting from a few categorized requirements and exploiting statistical properties of texts.

3. Semi-supervised classification of NFRs

Non-functional requirements are one of the hardest problems analysts and designers have to deal with during software design. NFRs usually specify critical and highly important quality attributes the client asked for his software, and they not only have to be identified within a possible large set of requirements documents, but also classified and prioritized.

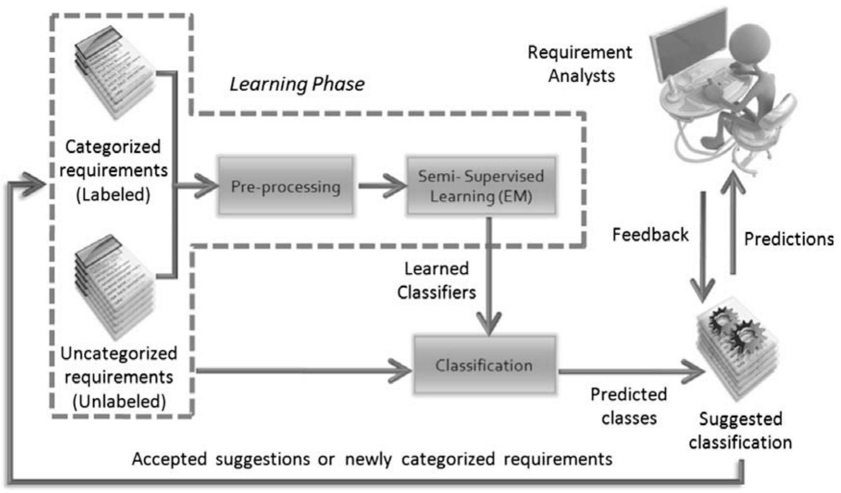
Learning Phase
Requirement Analysts
Categorized requirements (Labeled)
Pre-processing
Semi- Supervised Learning (EM)
Feedback
Predictions
Learned Classifiers
Uncategorized requirements (Unlabeled)
Classification
Predicted classes
Suggested classification
Accepted suggestions or newly categorized requirements

Fig. 1. Overview of the semi-supervised approach for NFRs classification.

In a real-world scenario, the analyst of a software development project needs to go through all the requirement documents gathered by a requirement elicitation team in order to decide whether they specify functional or non-functional requirements, as well as the categories or classes NFRs belong to (i.e. security, availability, scalability, etc.), with the ultimate goal of prioritizing and mapping them into architectural concerns.

This is an enormously time consuming task which requires a lot of effort from analysts since every requirement document must be read and manually classified. In order to deal with the problem of detecting and classifying non-functional requirements into a set of pre-defined categories or classes, we propose a semi-supervised text learning approach. In this approach, a classifier automatically recognizes different types of NFRs within a set of documents, each describing a requirement for the system written in natural language, and presents them to analysts for their inspection.

Fig. 1 depicts an overview of the proposed scheme. Initially, some categorized requirements are used in conjunction with non-categorized ones to learn a text classifier using a semi-supervised learning algorithm. The Expectation Maximization (EM) strategy was implemented with naïve Bayesian classifiers to accomplish this goal. An initial set of categorized requirements can be either detected by the requirement team as they perform interviews with users, or established with some alternative approach such as the simple method of using a pre-defined fixed set of keywords for classification proposed in [8]. Once a classifier is learned, it is used to categorize further unlabeled requirements. Optionally, the requirements classified with the highest confidence and/or those which received feedback from the analysts can be used as labeled requirements to repeat the process.

In the following subsections we firstly describe how unstructured documents describing requirements are transformed into suitable representations to be used as input for machine learning algorithms through the application of a number of well-known pre-processing steps. Then, an introduction to classic Bayesian classification is followed by an explanation of an existing semi-supervised learning method used in our approach for the identification and classification of NFRs. It worth noticing that these are well-known notions in the machine learning area that are used as part of the proposed requirement management approach.

3.1. Pre-processing

The most common method in the information retrieval (IR) field to obtain effective representations of documents is the vector space model (VSM) [40]. In this model, each document is identified by a feature vector in a space in which each dimension corresponds to a distinct term associated with a numerical value or weight indicating its importance.

Each document d from a document collection $\mathscr{D}$ is identified by a vector in the t-dimensional space, in which each vector component w_{ij} represents the weight of term t_i in the document d_j:

$$\vec{d_j} = (w_{1j}, w_{2j}, \ldots, w_{|T|j}) \tag{1}$$

Each term $t \in \mathscr{T}$, where $|\mathscr{T}|$ represents the total number of distinctive terms in the document collection or vocabulary, is weighted using the term frequency–inverse document frequency (TF–IDF) [39] term weighting function shown in the following equation:

$$tf\!-\!idf(t_i, d_j) = \#(t_i, d_j) \times \log\left(\frac{|\mathscr{D}|}{\#_{\mathscr{D}}(t_i)}\right) \tag{2}$$

where t_i denotes a term, d_j denotes a document, $\mathscr{D}$ is the complete set of documents in the collection, $\#(t_i, d_j)$ denotes the *term frequency*, that is, the number of times the term t_i occurs in d_j, $|\mathscr{D}|$ denotes the total number of documents in $\mathscr{D}$, and $\#_{\mathscr{D}}(t_i)$ denotes the *document frequency*, that is, the number of documents in $\mathscr{D}$ in which t_i occurs. This measure formalizes two empirical observations. First, the more times a term occurs in a document, the more relevant it would be to determine the document class. Second, the more times the term occurs along the collection of documents, the less powerful it is to discriminate among documents.

Several pre-processing steps are followed to transform textual requirements into vectors according to the vector space model. When dealing with unstructured text documents, the most common pre-processing tasks are normalization (including changing cases of letters, digits, hyphens and punctuation marks), stop-word removal and stemming.

In a first step, normalization is performed on the documents describing requirements. This step includes removing numbers and terms that contain digits, breaking hyphens to get individual words, removing punctuation marks, and finally converting letters to lower case.

The next pre-processing task is stop-word removal. Stop-words are words that occur frequently in a given language, which help to construct sentences or phrases but have little or no inherent semantic content (common stop-words in English include *a*, *about*, *an*, *are*, *as*, etc.). Some of these words are articles, prepositions, conjunctions, pronouns and very common verbs. Usually, stop-words are removed using a standard list or negative dictionary composed of a set of words that, due to their frequency or semantics, do not possess sufficient discriminative power [26]. In this work, we used a standard English stop-word list[1] composed of 319 different words.

After stop-word removal, a stemming algorithm is applied in order to reduce the morphological variants of the remaining words to their common roots, considering plural nouns, verb tenses, gerunds, pronouns, etc. Stemming or conflation algorithms can be defined as processes of linguistic normalization in which morphological variants of words are reduced to their root form, also known as stem. Stemming allows to reduce the dimensionality of the vector space by mapping several morphologically similar words into their common word stem. Porter algorithm [37] was applied for stemming words in the documents describing requirements.

Finally, the last pre-processing step involves the assignment of weights to all the resulting terms (actually stems) based on the *tf–idf* weighting function defined in Eq. (2).

3.2. Bayesian classification

The problem of learning classifiers that can predict the class labels of new, previously unseen examples based on a set of labeled training examples is known as supervised learning. Supervised learning in text domains is usually referred to as text learning, text classification or text categorization. It involves assigning a set of documents to one or more pre-defined classes or categories $C = \{c_1, c_2, \ldots, c_{|C|}\}$, where each class is supposed to contain documents with certain common properties. In a learning phase, a supervised learning algorithm is applied to induce a classifier, model or hypothesis, which is in turn used to predict the class of new documents in a classification phase.

Naïve Bayesian classification is one of the most popular techniques for text classification and has been reported as performing extremely well in practice in many research studies [29]. The Bayesian approach for classification consists in finding the most probable class for a a new example within a finite set of classes given the attributes that describe this example.

In the Bayesian learning framework for text classification it is assumed that text data was generated by a parametric model and training data is used to calculate Bayes optimal estimates for the model parameters. Based on these estimates, new test documents are classified using Bayes rule to calculate the posterior probability that a class would have generated the test document in question. In consequence, classification consists simply in the selection of the most probable class.

From the several variants of naïve Bayes classifiers, it has been shown that the multinomial model is most often the best choice for text categorization applications since it captures word frequency information in documents [29,13]. In this model, documents are assumed to have been generated by a mixture model parameterized by Θ, being each document an ordered list of words drawn from the vocabulary $V = \langle w_1, w_2, \ldots, w_{|v|}\rangle$.

The estimate of Θ, denoted $\widehat{\Theta}$, is made based on the available training data. Given a set of training documents $\mathscr{D} = \{D_1, D_2, \ldots, D_{|C|}\}$, where D_j is the subset of data for class c_j and $|C|$ is the number of classes, the estimate probability of a word w_t given class c_j is the number of times that w_t occurs in the training data D_j divided by the total number of occurrences in the training data for that class. Considering Laplacian smoothing to handle zero counts for infrequent words this can be formulated as follows:

$$P(w_t|c_j;\widehat{\Theta}) = \frac{1+\sum_{i=1}^{|\mathscr{D}|} N_{ti} P(c_j|d_i)}{|V| + \sum_{s=1}^{|V|}\sum_{i=1}^{|\mathscr{D}|} N_{si} P(c_j|d_i)} \tag{3}$$

[1] www.dcs.gla.ac.uk/idom/ir_resources/linguistic_utils/stop_words.

where N_{ti} is the number of times that the word w_t occurs in the document d_i and $P(c_j|d_i) = 1$ for each document in D_j and $P(c_j|d_i) = 0$ for documents of other classes.

The class prior probabilities can be also calculated using training data as follows:

$$P(c_j|\widehat{\Theta}) = \frac{1+\sum_{i=1}^{|\mathscr{D}|} P(c_j|d_i)}{|C| + |\mathscr{D}|} \tag{4}$$

More informally, Eqs. (3) and (4) can be re-written as:

$$P(w_t|c_j;\widehat{\Theta}) = \frac{1 + \text{No. of occurrences of } w_t \text{ in class } j}{|V| + \text{No. of words in class } j} \tag{5}$$

$$P(c_j|\widehat{\Theta}) = \frac{1 + \text{No. of documents in class } j}{|C| + |\mathscr{D}|} \tag{6}$$

Given estimates of both parameters calculated from training documents, classification can be performed on test documents by calculating the posterior probability of each class given the evidence of the test document, and selecting the class with the highest probability. Using Bayes rule this can be formulated as follows:

$$P(c_j|d_i;\widehat{\Theta}) = \frac{P(c_j|\widehat{\Theta})P(d_i|c_j;\widehat{\Theta})}{P(d_i|\widehat{\Theta})} \tag{7}$$

$$= \frac{P(c_j|\widehat{\Theta})\prod_{k=1}^{|d_i|} P(w_{d_i,k}|c_j;\widehat{\Theta})}{\sum_{r=1}^{|C|} P(c_r|\widehat{\Theta})\prod_{k=1}^{|d_i|} P(\text{though } w_{d_i,k}|c_r;\widehat{\Theta})} \tag{8}$$

where $w_{d_i,k}$ is the word in position k of the document d_i. In order to reduce computation in evaluating $P(d_i|c_j;\widehat{\Theta}.)$, the class conditional independence assumption is made in naïve Bayesian classification. This presumes that words are conditionally independent of one another given the class label, accounting for the substitutions in Eq. (8).

The most probable class is called a *maximum a posteriori* (MAP) hypothesis, denoted c_{MAP}, and can be determined by using the Bayes theorem to calculate the posterior probability of each candidate class in C. For each possible class c_j, c_{MAP} is selected as follows:

$$c_{MAP} = \arg\max_{c_j \in C} P(c_j|d_i;\widehat{\Theta}) \tag{9}$$

3.3. Semi-supervised classification with EM

In supervised learning, the selected algorithm (for example, naïve Bayes, *k*-NN, etc.) uses some labeled training examples from every class to generate a classification function or hypothesis. The problem of this approach is the large number of labeled examples required to learn a classifier capable of accurately predicting the label of a novel example. Furthermore, labeling is a time and cost consuming as well as error prone task since it has to be performed manually by domain experts.

Partially-supervised classification implies that there is no need for full supervision, considerably reducing the labeling effort required from users or experts. One of the possible strategies for partial supervision, commonly known as semi-supervised learning, consists of learning from both labeled and unlabeled examples or documents in the case of text categorization. This strategy is also known as *LU* learning (*L* stands for *labeled* and *U* for *unlabeled*). *LU* learning algorithms are based on a small set of labeled examples belonging to each class and a considerably larger set of unlabeled examples that are used to improve learning [28]. Although small, every class must have a set of labeled examples in order to enable learning.

The Expectation Maximization (EM) algorithm [11] is a popular class of iterative algorithms for maximum likelihood estimation in problems with incomplete data. It consists of two steps, the *Expec-*

tation step or *E*-step and the *Maximization step* or *M*-step. Basically, the first step fills in the missing data based on the current estimation of the parameters and the second step re-estimates the parameters maximizing the likelihood [28]. Unlabeled documents can be regarded as having missing data because of their lack of class labels. The parameters found on the *M*-step are in turn used to begin another *E*-step, and the process is repeated until EM converges to a local minimum when the model parameters stabilize.

EM is not an algorithm, strictly speaking, but a strategy or general framework in which a base algorithm is ran iteratively. Nigam et al. [32] proposed the EM algorithm for *LU* learning with naïve Bayes classification, which is summarized in Algorithm 1. The parameters that EM estimates in this case are the probability of each word given a class and the class prior probabilities as were formulated in Eqs. (3) and (4) for Bayesian classification.

Algorithm 1. EM algorithm with naïve Bayesian classification

```
1: Learn an initial naïve Bayesian classifier $f$ from only the labeled set
   $L$ (using Eqs. (3) and (4));
2: repeat
//    E-Step
3:    for each example $d_i$ in $U$
4:       Using the current classifier $f$ to compute $P(c_j|d_i)$ (using Eq. (7))
5:    end for
//    M-Step
6:    Learn a new naïve Bayesian classifier $f$ from $L \cup U$ by computing
      $Pr(c_j)$ and $Pr(w_t|c_j)$ (using Eqs. (3) and (4))
7: until the classifier parameters stabilize
8: Return the classifier $f$ from the last iteration
```

Initially, the documents in the labeled set L have class labels, whereas the documents in the unlabeled set U have missing class labels. EM is used to estimate the missing class labels based on the current model, i.e. to assign probabilistic class labels to each document d_i belonging to U. Thus, in each iteration EM assign to every document in U a probability distribution on the classes that it may belong to, i.e. $P(c_j|d_i)$ which takes a value in the $[0,1]$ interval. Instead, documents in L belong to a single class c_k that is known beforehand, i.e. $P(c_k|d_i) = 1$ and $P(c_j|d_i) = 0$ for $j \neq k$. Using both the labeled set L and the unlabeled set U with the assignments of $P(c_j|d_i)$, a new naïve Bayes classifier is constructed. This gives place to the next iteration of the algorithm, continuing until the classifier parameters, i.e. $P(w_t|c_j)$ and $P(c_j)$, no longer change or exhibit minimum changes.

The basic EM approach is based on the assumption that there is one-to-one correspondence between mixture components and classes. For textual data, a violation of this assumption is equivalent to saying that a class may consist of several different sub-classes or sub-topics, each best characterized by a different word distribution. In principle, the assumption holds in the case of requirements since NFRs focuses in certain software characteristics instead of multiple topics.

In this work, the EM strategy was implemented as it is described in Algorithm 1 using *Java* with the naïve Bayes algorithm provided by *Classifier4J*,[2] a *Java* library designed for text classification. Our own implementation was used to carry out an experimental evaluation of the approach using a dataset of functional and non-functional requirements.

4. Empirical evaluation

In order to evaluate the proposed approach, we designed and carried out several experiments aiming at assessing the performance of semi-supervised classification of requirements as well as comparing the results with those of supervised approaches found in the literature. The experimental setting, results and comparison with other approaches are detailed in this section.

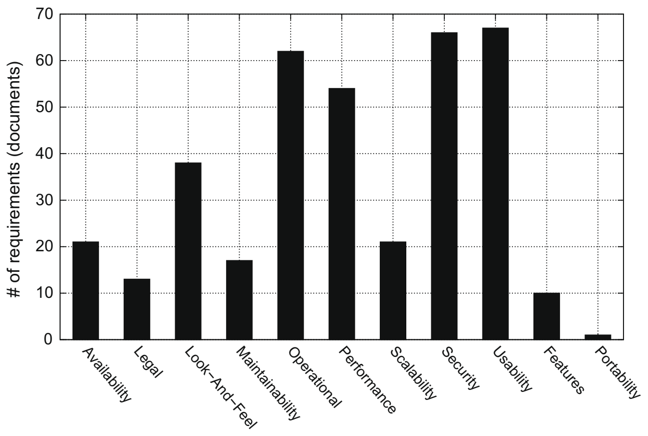

Fig. 2. Examples distribution in the collection of textual requirements.

4.1. Experimental setting

The approach proposed for NFRs identification is focused on semi-supervised text classification. Thus, the main aspect to evaluate is the performance of this kind of classifiers, particularly materialized through the Expectation Maximization strategy, in categorizing a set of textual requirements. For comparison purposes, we also tested some other common text classification approaches, including Rocchio algorithm*which is also* known as *TF–IDF* classifier because this is the most frequently used weighting scheme [21], *k-NN* and *naïve Bayes* [42].

NFRs are concerned with different software quality attributes a system must exhibit, such as accuracy, performance, security and modifiability. For the N different non-functional properties to be considered, N binary classifiers are learned, each one trained to distinguish the requirements belonging to a single class from those in all of the remaining classes. Hence, during the training process, a document that belongs to a class is added as a positive example to the corresponding classifier and as a negative example to the rest of the binary classifiers. At classification time, every document is tested against the N classifiers and the classifier with the highest output function or probability score assigns the class label. This is also referred to as *one-vs.-all* scheme in multi-class binary classification.

In order to perform the experiments, we used a collection of documents expressing a number of requirements available at PROMISE Software Engineering repository.[3] This dataset was built by MS students at DePaul University during a graduate course on Requirements Engineering. The collection consists of a total of 370 NFRs and 255 functional ones, corresponding to 15 different software development projects. Each document is composed by a description of the requirement, written in natural language, the ID of the project to which it belongs to and a label specifying the type of requirement. These labels indicate either that the requirement is a functional one or the type of non-functional requirement including the quality attributes availability, look-and-feel, legal, maintainability, operational, performance, scalability, security, usability, features and portability. NFRs are divided into the mentioned 11 different classes. However, the quality attribute portability had a single example in the collection and was excluded from

[2] http://classifier4j.sourceforge.net/.

[3] http://promisedata.org/?p=38.

Table 1
Distribution of functional and non-functional requirements in the collection.

Type	Project number															Total
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Functional	20	11	47	25	36	27	15	20	16	38	0	0	0	0	0	255
Availability	1	2	2	0	2	1	0	5	1	1	2	1	1	1	1	21
Legal	0	0	0	6	3	0	1	3	0	0	0	0	0	0	0	13
Look-and-feel	1	4	0	2	3	2	0	6	0	7	2	2	4	3	2	38
Maintainability	0	0	0	0	0	4	0	2	1	0	1	3	2	2	2	17
Operational	0	0	7	6	10	15	3	9	2	0	0	2	2	3	3	62
Performance	2	6	2	2	4	1	2	17	4	4	3	5	0	1	1	54
Scalability	0	3	4	0	3	4	0	4	0	0	0	1	2	0	0	21
Security	1	3	10	10	7	5	2	15	0	1	3	3	2	2	2	66
Usability	3	6	8	4	5	13	0	10	0	2	2	3	6	4	1	67
Features	0	4	0	0	0	2	0	2	0	0	0	2	0	0	0	10
Portability	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Total	28	40	80	55	73	74	23	93	24	53	13	22	19	16	12	625

the experiments due to its low incidence. The distribution of NFRs by class is depicted in Fig. 2, whereas Table 1 shows a summary of the complete requirement collection.

4.2. Evaluation metrics

The purpose of the requirement classification approach is to identify the type of each textual requirement in the classes described above. The results of this classification process were evaluated using the standard definitions of accuracy, precision, recall, and F-measure metrics [44]. Given a test set of N documents expressing system requirements, a contingency table is constructed for each binary classification problem containing the count of true positives (TP) or number of correctly classified requirements, false positives (FP) or number of requirements incorrectly classified in the category in question, true negatives (TN) or number of requirements correctly not classified in the category in question and false negatives (FN) or number of requirements incorrectly not classified in the category in question. Using these values, the metrics for binary-decisions are defined as follows:

$$\text{Accuracy} = \frac{TP + TN}{N} \tag{10}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{11}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{12}$$

$$\text{F-measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{13}$$

In multi-label classification, the simplest method for computing an aggregate score across categories is to average the scores of all binary tasks. The resulting scores are called macro-averaged accuracy, recall, precision and F-measure, respectively.

For each experiment, we randomly split the collection into a training set, which is used to learn binary classifiers for requirements, and a testing set, which is used to evaluate their joint performance in classifying previously unseen requirements. Every experiment was ran 10 times using stratified 10-fold cross-validation in order to obtain average scores for the metrics mentioned above. Since the collection used for experiments has an unbalanced distribution of examples, stratification is used to ensure that each fold contains roughly the same proportion of examples in each class as in the original collection. It is important to remark that every training set needs to have at least one document of each class; otherwise, neither supervised nor semi-supervised classifiers can learn to distinguish examples in that class.

4.3. Experimental results

The Expectation Maximization strategy with naïve Bayesian classifiers was implemented according to the algorithm detailed in Algorithm 1. To evaluate the effectiveness of the classifiers learned with this algorithm, we calculated the accuracy of categorizing the collection of requirements using different sizes of the training set and, consequently, different proportions of labeled and unlabeled examples for learning classifiers. We split the collection into 468 requirements (approximately 75% of the collection) for training, preserving the remaining 156 (approximately 25%) for testing.

Fig. 3 depicts the results of using increasing percentages of the training set as labeled examples and the remaining requirements also in the training set as unlabeled ones. The 100% of labeled examples corresponds to the total of the 468 requirements in the training set, whereas the size of the test set is maintained along all the experiments (156 requirements). It can be observed in the figure that the semi-supervised approach proposed in this paper takes advantage of unlabeled examples to improve classification accuracy in comparison with a supervised approach using naïve Bayes, which is based exclusively on labeled examples.

Fig. 4 compares the experimental results obtained with EM strategy and other classical algorithms for supervised text categorization as naïve Bayes, k-NN and vectorial TF–IDF using the same split of requirements. EM outperforms these algorithms in terms of

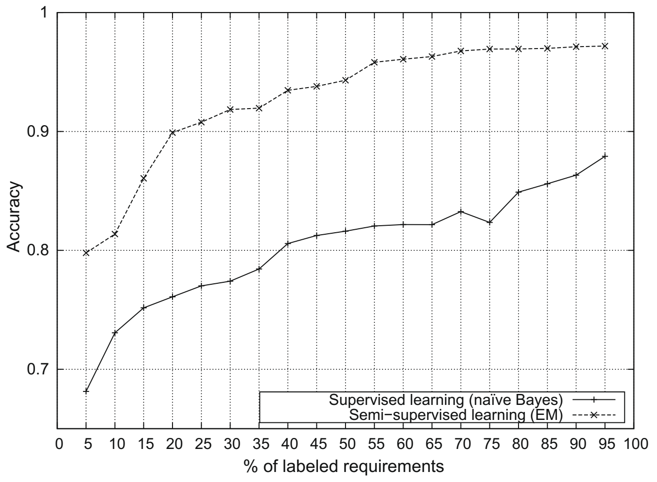

Fig. 3. Classification accuracy of semi-supervised and supervised learning approaches.

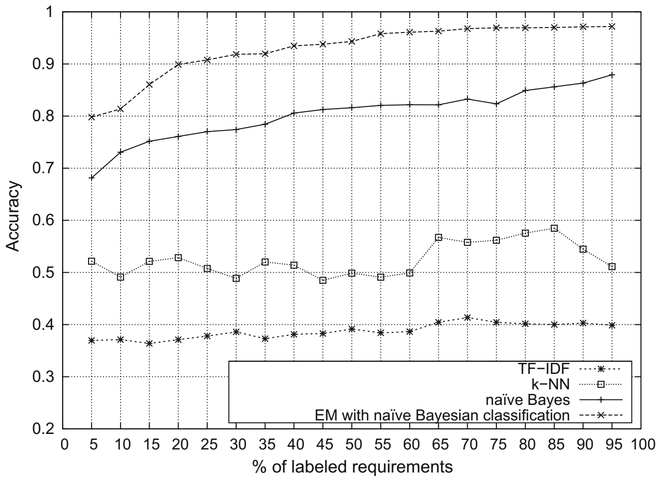

Fig. 4. Comparison of several classification algorithms versus semi-supervised EM strategy.

accuracy since unlabeled requirements provide some insights about the different types of requirements that are exploited during learning, for instance words that tend to appear together in positive examples of a certain type of requirements or belong to negative examples of such type.

In the results described before, accuracy was calculated considering both functional and non-functional requirements of different types (availability, look-and-feel, maintainability, etc.). However, the goal of the classification approach introduced in this paper is to identify NFRs in their corresponding categories. Fig. 5 shows the precision of classification for functional requirements and the average for the different categories of NFRs. Naturally, precision is slightly better for functional requirements as they outnumber NFRs in the different categories (255 requirements are functional, whereas NFRs range from 10 to 67 requirements per category).

NFRs are not only distinguished from functional requirements, but also classified into different classes according to their type with a high level of precision. Even using a reduced number of labeled examples for training the classifiers, semi-supervised classification reaches good levels of performance. Fig. 5, for instance, shows that EM overcomes the 75% of precision for NFRs with less of 25% out of the total number of training requirements being labeled.

Fig. 7 details the classification results for each class of NFRs in terms of F-measure, whereas precision and recall are shown in

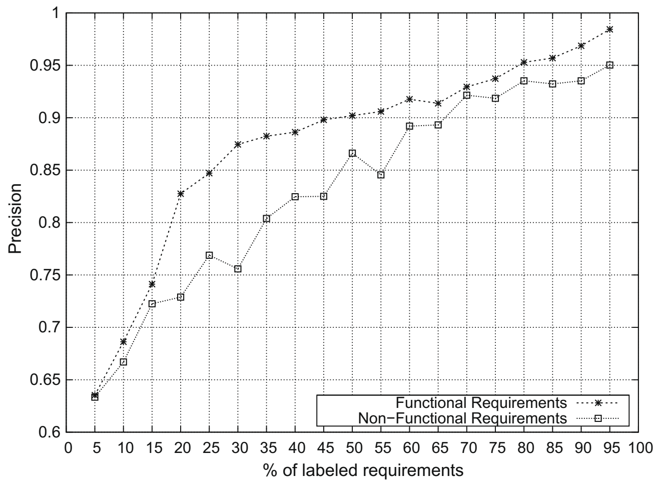

Fig. 5. Classification performance for functional and non-functional requirements.

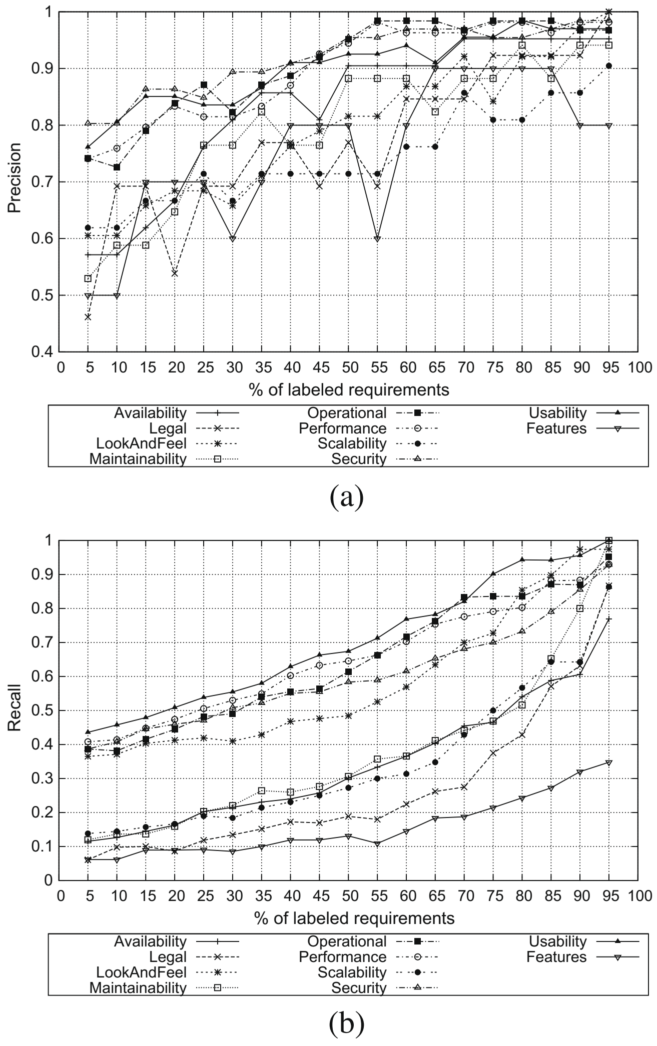

Fig. 6. Classification performance for NFRs in term of precision and recall.

Fig. 6a and b respectively. Among the categories with the poorest performance are *Features* and *Legal* requirements which are also the ones with the smaller number of examples. F-measure scores are affected by an initially low recall caused by the variety of NFRs the requirements have to be classified into and the existence of some categories with only a few examples. The effectiveness of classifiers improves as more labeled requirements become available and classifiers are able to better distinguish requirements in each class. In a real-world scenario, the increase in the amount of labeled examples will be given by processing feedback from analysts as it will be explained in the following subsection. It is also important to notice that the present collection covers requirements in 15 different projects, so that it may be some variability in the vocabulary employed to describe them. An improvement in classification performance can be expected if documents belonging to a single software project are considered during learning.

4.4. User feedback learning

In previous sections, the learning and categorization scenario in which classifiers were trained was based on a set of pre-labeled examples, in this case, requirement documents with an assigned category or type. In order to evaluate the proposed semi-supervised approach, a collection of documents labeled beforehand was used to train classifiers with incremental number of labeled

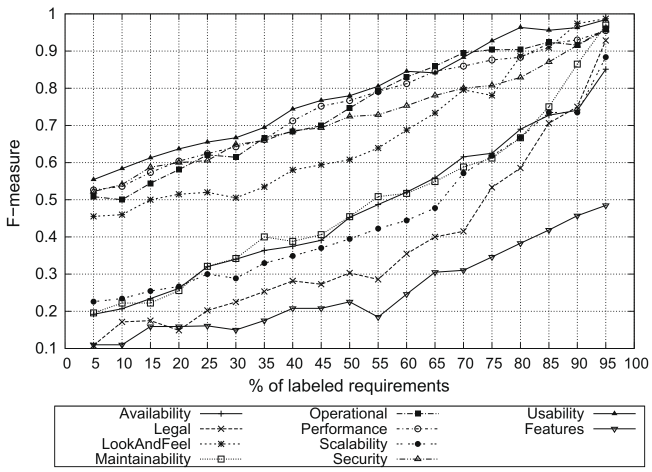

Fig. 7. Classification performance for the different types of NFRs.

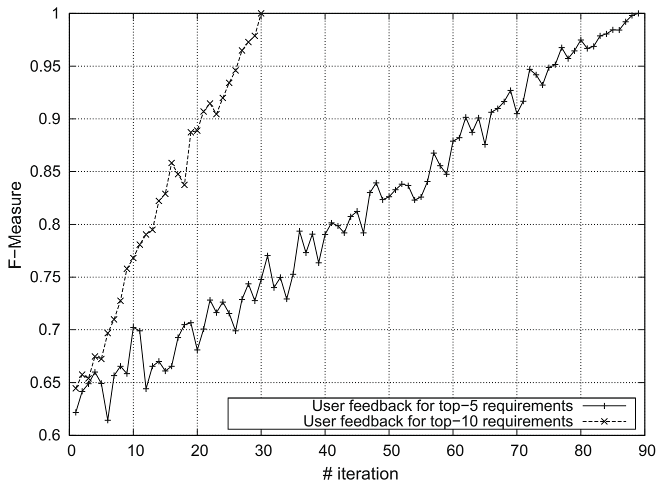

Fig. 8. Semi-supervised classification of requirements considering feedback from analysts.

requirements. Experiments showed not only an increase in classification performance as more labeled requirements were available, but also the capacity of semi-supervised learning for taking advantage of unlabeled texts thereby reducing the number of labeled requirements needed for achieving accurate classification results.

During a software development project in a real-world scenario, semi-supervised classification can be used to interactively classify a large number of requirements the system should fulfill and, at the same time, identifying different types of NFRs in early stages of software development. Integrated in a decision support system, this approach aims at predicting the categories of requirements to suggest classifications to analysts and receive feedback from them in order to improve learning and classification in an iterative process.

Recommender systems have become an important research area, mainly due to the abundance of practical applications that help users to deal with information overload by providing personalized recommendations [1]. These systems are characterized by the ability of making recommendations of potentially useful information in many application domains, such as web pages [27], news [38], e-commerce [41] and movies [31], among others. Recommender systems made significant progress over the last decade when numerous content-based, collaborative, and hybrid methods were proposed and subsequently developed for different software engineering activities [18].

The evaluated approach of semi-supervised detection and classification of NFRs was integrated into a recommender system that provides assistance to human analysts in early software development stages [3]. Recommendations made by the system can be used for requirement analysis in software development projects, reducing the effort of manual identification and classification of requirement documents. In this scenario, requirement analysts should provide a small initial set of categorized documents, obtaining recommendations about a possible classification of the remaining documents. Additionally, analysts can provide feedback to refine classification in an iterative process.

Initially, documents describing the requirements gathered by an elicitation team are typically unlabeled. The goal of the classification process is reducing the effort of the elicitation team in labeling all the requirements by interacting with the support system. A small set of labeled requirements is required to start the learning process and posterior classification. It is likely that the elicitation team would be able to recognize requirements of different types during the elicitation phase to serve this purpose. Alternatively, a straightforward method such as using a pre-defined set of keywords to identify a small set of better matching requirements for the different classes can be used to bootstrap the semi-supervised classification process.

Given the initial training set of requirement documents manually labeled by analysts, the system trains a set of binary classifiers using the EM algorithm and classifies the remaining requirements using these classifiers, using uncategorized requirements as an additional knowledge source. Afterwards, documents with categories automatically assigned by the recently trained classifiers are presented to the analysts for validation. That is, analysts can verify the correctness of the suggested categories for requirements and provide some feedback confirming or rejecting candidate NFRs. A subsequent run of EM algorithm using the newly labeled requirements, i.e. those which received feedback from analysts, should lead to an improvement in the previously registered classification accuracy. This iterative process can be repeated several times, until analysts are satisfied with the current classification of requirements or all requirements have been assigned to a category.

Fig. 8 shows a simulation of this iterative learning and classification process aiming at supporting analysis in processing of textual requirements. In each iteration, unlabeled requirements (belonging to the previously described collection) are classified using the lastly trained classifiers and ranked according to their probability of belonging to the predicted class. We simulated a user providing feedback over the top-5 and top-10 requirements in the list by using the real classes of the requirements in the collection. For the next interaction, the training set is augmented with the requirements which received feedback, classifiers are updated and the process is repeated. The simulation of feedback learning was carried out considering that due to previous experience, the human analyst interacting with the recommender system is able to read and understand each requirement document, making no mistakes when validating and correcting the provided recommendations.

Feedback about the classes assigned to the top-10 requirements in the ranked list allows to converge to the real classification in around 30 iterations, having asked the user for labeling roughly half of the total number of requirements. Likewise, providing feedback about the top-5 requirements takes an additional number of labeled requirements and iterations. It is important to remark that even though the analysts have to read each requirement placed in the top-5 or top-10 position in the list to express the agreement or disagreement with the predicted class, the accuracy of the suggested classification for such requirements in the experiments was 90.11% considering the top-5 requirements and 93.66% considering the top-10. Thus, the cognitive load of analysts is reduced since they will have to change the suggested class only for a few requirements. It is also worth highlighting that even though 30 iterations are required during user feedback learning to reach a 100% of accuracy considering manual inspection of the top-10 classified documents, good results are achieved only with 10 iterations (about 75% of accuracy), a reasonable number for a requirement analyst to deal with.

5. Conclusions

Identification of non-functional requirements is critical for making correct software design decisions starting from early stages of software development projects. Natural language descriptions transformed into textual specifications is a common means for capturing requirements in these stages. Methods of supervised text learning have been proposed in the literature to address the problem of identifying and classifying NFRs [8]. However, supervision implies the need of an important number of already categorized requirements to induce an accurate classifier capable of identifying the type of novel requirements.

In this paper, we have presented and evaluated a semi-supervised learning approach for classification of NFRs aiming at reducing the number of categorized requirements needed for learning by taking advantage of the underlying characteristics of texts, such as co-occurrence of words in documents belonging to the same classes. Integrated in a support system for requirement analysis, this approach can help to mitigate the labeling effort required from analysts, involving the manual revision and classification of available textual requirements. In future works we are planning to introduce active learning in this iterative classification process striving to reduce even more the required labeling effort while retaining the accuracy by selecting the examples to be labeled by analysts in an intelligent way (i.e. analysts should be asked to label the more informative examples instead of the top ranked ones).

Experimental results demonstrate the feasibility of using semi-supervised learning as a method for detecting NFRs. Empirical evidence showed that semi-supervision requires less human effort in labeling requirements than fully supervised methods and can be further improved based on feedback from analysts once integrated within a decision support system for managing requirements [3]. In addition, the proposed approach does not rely on the existence of previously categorized requirements (e.g. belonging to previous projects) as this can introduce noise in texts stemming from the variations in the vocabulary employed by elicitation teams in heterogeneous domains. In contrast, semi-supervised classification needs a small amount of labeled requirements belonging to the current software project to initiate learning.

References

[1] G. Adomavicius, E. Tuzhilin, Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions, IEEE Transactions on Knowledge and Data Engineering 17 (2005) 734–749.
[2] T.H. Al Balushi, P.R. Falcone Sampaio, D. Dabhi, P. Loucopoulos, ElicitO: a quality ontology-guided NFR elicitation tool, in: Requirements Engineering: Foundation for Software Quality, LNCS, vol. 4542, Springer, 2007, pp. 306–319.
[3] A. Casamayor, D. Godoy, M. Campo, A recommender system for classification of non-functional requirements, in: Proceedings of the 10th Argentine Symposium on Artificial Intelligence (ASAI'2009) in the 38th Argentine Meetings on Informatics and Operations Research (JAIIO'2009), Mar del Plata, Argentina, SADIO, August 2009.
[4] C. Castro-Herrera, C. Duan, J. Cleland-Huang, B. Mobasher, A recommender system for requirements elicitation in large-scale software projects, in: ACM Symposium on Applied Computing, 2009.
[5] O. Chapelle, B. Schölkopf, A. Zien, Semi-Supervised Learning, MIT Press, 2006.
[6] J. Cleland-Huang, B. Mobasher, Using data mining and recommender systems to scale up the requirements process, in: Proceedings of the 2nd International Workshop on Ultra-Large-Scale Software-Intensive Systems (ULSSIS'2008), 2008, pp. 3–6.
[7] J. Cleland-Huang, R. Settimi, X. Zou, P. Solc, The detection and classification of non-functional requirements with application to early aspects, in: Proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06), 2006, pp. 36–45.
[8] J. Cleland-Huang, R. Settimi, X. Zou, P. Solc, Automated classification of non-functional requirements, Requirements Engineering 12 (2) (2007) 103–120.
[9] J.L. Cybulski, K. Reed, Computer-assisted analysis and refinement of informal software requirements documents, in: Proceedings of the 5th Asia Pacific Software Engineering Conference (APSEC'98), 1998, p. 128.
[10] L.M. Cysneiros, J.C.S. do Prado Leite, Nonfunctional requirements: from elicitation to conceptual models, IEEE Transactions on Software Engineering 30 (5) (2004) 328–350.
[11] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, Journal of the Royal Statistical Society, Series B 39 (1) (1977) 1–38.
[12] J. Dörr, D. Kerkow, A. Von Knethen, B. Paech, Eliciting efficiency requirements with use cases, in: Proceedings of the International Workshop on Requirements Engineering: Foundations of Software Quality (REFSQ'2003), 2003, pp. 23–32.
[13] S. Eyheramendy, D.D. Lewis, D. Madigan, On the naive bayes model for text categorization, in: Proceedings of the 9th International Workshop on Artificial Intelligence and Statistics, 2002.
[14] A. Fantechi, S. Gnesi, G. Lami, A. Maccari, Application of linguistic techniques for use case analysis, in: Proceedings of the 10th Anniversary IEEE Joint International Conference on Requirements Engineering (RE'02), 2002, pp. 157–164.
[15] R. Fernandes, A. Cowie, Capturing informal requirements as formal models, in: Proceedings of the 9th Australian Workshop on Requirements Engineering (AWRE'04), 2004, pp. 1–8.
[16] V. Gervasi, B. Nuseibeh, Lightweight validation of natural language requirements, Software – Practice & Experience 32 (2) (2002) 113–133.
[17] V. Gervasi, D. Zowghi, Reasoning about inconsistencies in natural language requirements, ACM Transactions on Software Engineering and Methodology (TOSEM) 14 (3) (2005) 277–330.
[18] H. Happel, W. Maalej, Potentials and challenges of recommendation systems for software development, in: RSSE '08: Proceedings of the 2008 International Workshop on Recommendation Systems for Software Engineering, ACM, New York, NY, USA, 2008, pp. 11–15.
[19] A. Jalaji, R. Goff, M. Jackson, N. Jones, T. Menzies, Making sense of text: identifying nonfunctional requirements early. Technical Report, West Virginia University CSEE, 2006.
[20] T. Jingbai, H. Keqing, W. Chong, L. Wei, A context awareness non-functional requirements metamodel based on domain ontology, in: Proceedings of the IEEE International Workshop on Semantic Computing and Systems (WSCS'08), 2008, pp. 1–7.
[21] T. Joachims, A probabilistic analysis of the rocchio algorithm with tfidf for text categorization, in: 12th International Conference on String Processing and Information Retrieval (SPIRE), 1997, pp. 143–151.
[22] H. Kaiya, A. Osada, K. Kaijiri, Identifying stakeholders and their preferences about NFR by comparing use case diagrams of several existing systems, in: Proceedings of the 12th IEEE International Requirements Engineering Conference (RE'04), Washington, DC, USA, 2004, pp. 112–121.
[23] H. Kitapci, B.W. Boehm, Using a hybrid method for formalizing informal stakeholder requirements inputs, in: Proceedings of the 4th International Workshop on Comparative Evaluation in Requirements Engineering (CERE'06), 2006, pp. 48–59.
[24] Y. Ko, S. Park, J. Seo, Web-based requirements elicitation supporting system using requirements categorization, in: Proceedings of 12th International Conference on Software Engineering and Knowledge Engineering (SEKE'2000), 2000, pp. 334–451.
[25] Y. Ko, S. Park, J. Seo, S. Choi, Using classification techniques for informal requirements in the requirements analysis-supporting system, Information and Software Technology 49 (11–12) (2007) 1128–1140.
[26] G. Kowalski, M.T. Maybury, Information Storage and Retrieval Systems: Theory and Implementation, Kluwer Academic Publishers, Norwell, MA, USA, 2000.
[27] H. Lieberman, C. Fry, L. Weitzman, Exploring the web with reconnaissance agents, Communication of the ACM Journal 44 (8) (2001) 69–75.
[28] B. Liu, Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data (Data-Centric Systems and Applications), Springer-Verlag, 2006.
[29] A. Mccallum, K. Nigam, A comparison of event models for Na Bayes text classification, in: AAAI-98 Workshop on Learning for Text Categorization, vol. 752, 1998, pp. 41–48.
[30] L. Mich, M. Franch, P. Inverardi, Market research for requirements analysis using linguistic tools, Requirements Engineering 9 (1) (2004) 40–56.
[31] B.N. Miller, I. Albert, S.K. Lam, J.A. Konstan, J. Riedl, Movielens unplugged: experiences with an occasionally connected recommender system, in: IUI '03: Proceedings of the 8th International Conference on Intelligent User Interfaces, ACM, New York, NY, USA, 2003, pp. 263–266.
[32] K. Nigam, A.K. McCallum, S. Thrun, T. Mitchell, Text classification from labeled and unlabeled documents using EM, Machine Learning 39 (2000) 103–134.
[33] J. Natt och Dag, V. Gervasi, S. Brinkkemper, B. Regnell, A linguistic-engineering approach to large-scale requirements management, IEEE Software 22 (1) (2005) 32–39.

[34] O. Ormandjieva, I. Hussain, L. Kosseim, Toward a text classification system for the quality assessment of software requirements written in natural language, in: Proceedings of the 4th International Workshop on Software Quality Assurance (SOQUA'07), 2007, pp. 39–45.

[35] J. Palmer, Y. Liang, Indexing and clustering of software requirements specifications, Information and Decision Technologies 18 (4) (1992) 283–299.

[36] S. Park, H. Kim, Y. Ko, J. Seo, Implementation of an efficient requirements-analysis supporting system using similarity measure techniques, Information and Software Technology 42 (6) (2000) 429–438.

[37] M. Porter, An algorithm for suffix stripping, Program 14 (3) (1980) 130–137.

[38] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl, Grouplens: an open architecture for collaborative filtering of netnews, in: CSCW '94: Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work, ACM, New York, NY, USA, 1994, pp. 175–186.

[39] G. Salton, C. Buckley, Term weighting approaches in automatic text retrieval, Information Processing and Management 24 (5) (1988) 513–523.

[40] G. Salton, A. Wong, C.S. Yang, A vector space model for automatic indexing, Communications of the ACM 18 (11) (1975) 613–620.

[41] J.B. Schafer, J.A. Konstan, J. Riedl, E-commerce recommendation applications, Data Mining and Knowledge Discovery 5 (1–2) (2001) 115–153.

[42] F. Sebastiani, Machine learning in automated text categorization, ACM Computing Surveys 34 (1) (2002) 1–47.

[43] A. Steele, J. Arnold, J. Cleland-Huang, Speech detection of stakeholders' non-functional requirements, in: Proceedings of the 1st International Workshop on Multimedia Requirements Engineering (MERE'06), 2006, p. 3.

[44] Y. Yang, An evaluation of statistical approaches to text categorization, Information Retrieval 1 (1–2) (1999) 69–90.