Study of Various Classifiers for Identification and Classification of Non-Functional Requirements

László Tóth[1] and László Vidács[1,2]

[1] Department of Software Engineering, University of Szeged, Hungary
premissa@inf.u-szeged.hu

[2] MTA-SZTE Research Group of Artificial Intelligence, University of Szeged, Hungary
lac@inf.u-szeged.hu

Abstract. Identification of non-functional requirements in an early phase of software development process is crucial for creating a proper software design. These requirements are often neglected or given in too general forms. However, interviews and other sources of requirements often include important references also to non-functional requirements which are embedded in a bigger textual context. The non-functional requirements have to be extracted from these contexts and should be presented in a formulated and standardized way to support software design. The set of requirements extracted from their textual context have to be classified to formalize them. This task is to be accomplished manually but it can be very demanding and error-prone. Several attempts have been made to support identification and classification tasks using supervised and semi-supervised learning processes. These efforts have achieved remarkable results. Researchers were mainly focused on the performance of classification measured by precision and recall. However, creating a tool which can support business analysts with their requirements elicitation tasks, execution time is also an important factor which has to be taken into account. Knowing the performance and the results of benchmarks can help business analysts to choose a proper method for their classification tasks. Our study presented in this article focuses on both the comparison of performances of the classification processes and their execution time to support the choice among the methods.

Keywords: non-functional requirements, requirements elicitation, classification, natural language processing

1 Introduction

Non-functional requirements (NFRs) are crucial factors for software design [1]. The lack of a well structured set of non-functional requirements can lead to an inappropriate software design and the failure of the specific project. Security aspects are also a critical part of the software design process which is one of the emphasized concern of the software development these days. These aspects cannot be reviewed completely without security related non-functional requirements. In practice, many NFRs are out of the analysis and even those which are included in specifications are poorly engineered. Incomplete or ambiguous specifications can lead the system into unspecified state [2].

Requirements are originated from memos of interviews and other textual sources like regulations, laws or reports. Regulations and laws are well-structured documents but their structure reflects only the business viewpoints. Memos created during interviews, on the other hand, are very often unstructured or semi-structured. Requirements are embedded into these textual contexts and their extraction and classification are ones of the most important duty of business analysts. Identifying and classifying non-functional requirements from this collection of different documents can be demanding and error-prone. Several investigations have been accomplished with remarkable results to support the identification and classification process using natural language processing and machine learning methods [3, 4]. Some researchers investigated the use of ontologies which have been created on standards' basis [5, 6]. Some researchers like Lu Megmen et al. [7] or Abad et al. [8] utilized supervised learning methods, others utilized semi-supervised learning techniques such as Expectation Maximization strategy [9]. Abad et al. applied also clustering techniques to identify the best method for processing requirements sentences. These studies have shown that machine learning processes can be utilized successfully also for requirements engineering if their inputs are prepared for this purpose appropriately. A common representation of the texts containing requirements is the tf-idf model which formulates a sparse matrix containing the measure of the importance of the given word in a given text.

Semi-supervised methods can give better results if there are not enough labeled examples for learning as shown by Casamayor et al. [9].This is a common issue in requirements engineering that labeled examples can be obtainable with difficulty especially in case of some specific domain. Ontology-based methods can cope with also this issue.

We have compared the performance of classification methods of sklearn library [10] with each other and also with the results of former studies like [9, 7] as we will present it in this article later. Because of our experiments are based on sklearn the results of these experiments can show the difference of the performance of different implementation comparing them with the results of former researches. This difference suggests that experiments can be only reproduced if also the implementation of algorithms and methods are the same. Next to the comparison mentioned above, we have measured also the execution time which provides another important factor for business analysts to choose an appropriate classification method for their specific classification tasks.

The main contribution of our work is a broad comparison of processes implemented in sklearn regarding their performance and the execution time. This information together can support a better choice among the tools for requirement classification process. The outcomes of our experiments have also confirmed the results of former researches [9, 7, 8].

The paper is organized as follows. The next section introduces our approach and analysis model. In Section 3 we introduce the dataset of non-functional requirements and outline the addressed problem. Section 4 provides results of our analysis produced by various classifiers and compares closely related work. The related literature briefly presented in Section 5 and we conclude the paper in Section 6.

2 Background

The source of requirements are memos of different interviews and regulations, laws and standards. These documents contain texts written in natural languages. In order to these texts be capable of classification they have to be preprocessed. During the preprocessing, texts are transformed into another representation. The most common representation is the tf-idf model which is a measure of the importance of a given word in a given text. The measure of tf-idf is formulated as:

$$tfidf(t,d,D) = tf(t,d) * idf(t,D)$$

where t denotes terms (words in our case), d denotes the document (sentence in our case) and D denotes the collection of documents (the set of sentences in our case). The tf(t,d) is the term frequency in a given document. The idf (inverse document frequency) is formulated as:

$$idf(t,D) = log\frac{|D|}{1+|\{d \in D : t \in d\}|}$$

Statistical methods often use the bag of words model in which a text or a sentence is represented as the multiset of its words disregarding grammar or modality of the given sentence. In order to these model can be applied sentences have to be preprocessed appropriately. Punctuation characters, stop words are to be removed and words of the sentences are to be stemmed. The most commonly used method for this purpose is the Porter Stemmer. However, using these models some important information will vanish as mentioned above.

Classification of textual information can be performed using classifiers designed for multiclasses and binary classifiers can also be used following the one-versus-all strategy. In this case, classes are classified iteratively, only one class is selected in each step which gets label one, others get label zero. The cumulative result is that class, which produces the highest probability for the given example.

For evaluation, we have used precision and recall. The precision is formulated as:

$$Precision = \frac{tp}{tp+fp}$$

where tp is the true positive which is the number of correct positive classification, fp denotes the false positive which is the case when classifier accepts the example but it has to be rejected. This is called Type I error.

Recall can be formulated as:

$$Recall = \frac{tn}{tn + fn}$$

where tn denotes true negative which is a correct negative classification and fn denotes the false negative which is the case when classifier rejects the example but it has to be accepted. This is called Type II error.

We have used the average of these values to compare the classifiers each other.

3 Experiments

We used TERA Promise NFR dataset for our experiments [11]. This dataset was created by students of DePaul University and it was updated in 2010. This dataset contains requirements sentences of 15 projects which were classified by the students. The statistics about the classification and the related projects is shown in Table 1 and Figure 1.

Requirement type		Project No															Total
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Functional	(F)	20	11	47	25	36	27	15	20	16	38	0	0	0	0	0	255
Availability	(A)	1	2	2	0	2	1	0	5	1	1	2	1	1	1	1	21
Fault tolerance	(FT)	0	4	0	0	0	2	0	2	0	0	0	2	0	0	0	10
Legal	(L)	0	0	0	6	3	0	1	3	0	0	0	0	0	0	0	13
Look and feel	(LF)	1	4	0	2	3	2	0	6	0	7	2	2	4	3	2	38
Maintainability	(MN)	0	0	0	0	0	4	0	2	1	0	1	3	2	2	2	17
Operational	(O)	0	0	7	6	10	15	3	9	2	0	0	2	2	3	3	62
Performance	(PE)	2	6	2	2	4	1	2	17	4	4	3	5	0	1	1	54
Portability	(PO)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Scalability	(SC)	0	3	4	0	3	4	0	4	0	0	0	1	2	0	0	21
Security	(SE)	1	3	10	10	7	5	2	15	0	1	3	3	2	2	2	66
Usability	(US)	3	6	8	4	5	13	0	10	0	2	2	3	6	4	1	67
Total NFRs		8	29	33	30	37	47	8	73	8	15	13	22	19	16	12	370
Functional		20	11	47	25	36	27	15	20	16	38	0	0	0	0	0	255
Total Requirements		28	40	80	55	73	74	23	93	24	53	13	22	19	16	12	625

Table 1. Requirement labels in the 15 projects of the Promise NFR dataset

Regarding class of Portability, the dataset contains only one example so we had removed that example before our experiments were executed.

In order to perform classification dataset had been preprocessed: punctuation characters and stop words were removed and the remaining words were stemmed using Porter Stemmer process. The preprocessed dataset was then transformed to tf-idf representation.

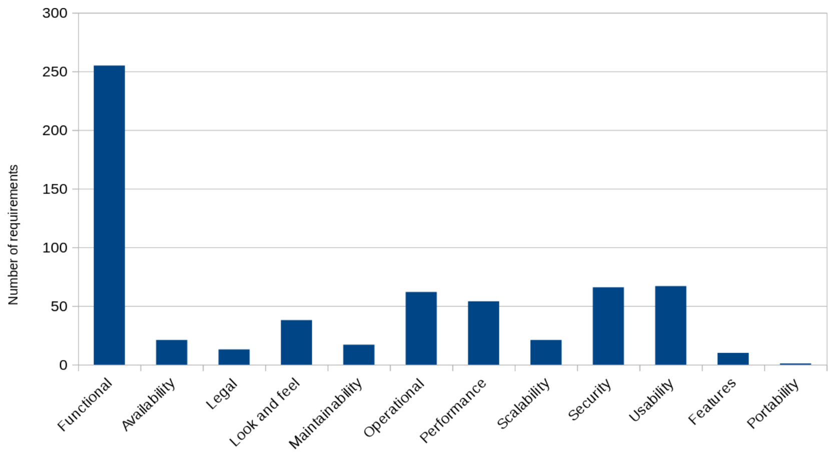

Fig. 1. Requirements classification process

We accomplished classification processes using algorithms implemented in sklearn [10] and compared the results each other and the results obtained by Cassamayor et al in their experiments [9]. The goal of our experiments was determining the best algorithm of classification implemented in sklearn for requirements classification tasks considering precision, recall and F-score, complemented by execution time. For our experiments, we selected those classifiers which had implemented one-vs-all strategy and/or were inherently capable of handling multiclasses. We selected Multinomial-, Gaussian- and Bernoulli Naive Bayes classifiers and also Support Vector Machine with linear kernel, Linear Logistic Regression, Label Propagation, Label Spreading, Decision Tree, Extra Tree, Extra Trees as an ensemble method, K-Nearest Neighbour and Multi Layer Perceptron methods were selected. The selection process was also influenced by classifiers' resource requirements.

The experiments were executed using repeated K-Fold cross-validation method. Both the number of groups and the repetition number were set to 10. Precision, recall and F-measure were calculated both for each test and each class by the corresponding sklearn function and the results were averaged for each classifier. Averaged variance was calculated as well and also the F-score value was computed for every classifier using the averaged precision and recall because the averaged F-score measure does not hold any useful information.

4 Results

In Table 2 the measured average Precision, Recall, F-score is represented such as their averaged variance. As mentioned in the previous chapter average F-score does not hold valuable information so F-score was computed using the following formula:

$$F - measure = 2 * \frac{Precision * Recal}{Precision + Recall}$$

The average of precision, recall and F-measure values are illustrated in Figure 3. The calculated F-measures are plotted in Figure 2. For comparison purpose, the results are illustrated using line diagrams. The averaged results and variance can be studied using Figure 4.

Classifier	Average			Comp	Variance		
	P	R	F	F	P	R	F
BernoulliNB	0.43	0.22	0.25	0.29	0.18	0.09	0.06
DT	0.66	0.64	0.62	0.65	0.01	0.03	0.01
ET	0.63	0.62	0.59	0.62	0.01	0.04	0.02
ETs	0.63	0.63	0.59	0.63	0.01	0.04	0.02
GNB	0.72	0.69	0.67	0.70	0.02	0.02	0.02
KNeighbours	0.71	0.52	0.55	0.60	0.08	0.07	0.05
LabelPropagation	0.70	0.68	0.65	0.69	0.02	0.03	0.02
LabelSpread	0.70	0.67	0.65	0.68	0.02	0.03	0.02
Logistic	0.87	0.67	0.72	0.76	0.01	0.05	0.03
MLP	0.38	0.66	0.36	0.48	0.01	0.03	0.01
MultinomialNB	0.84	0.68	0.72	0.75	0.02	0.03	0.02
SVM	0.89	0.65	0.71	0.75	0.01	0.05	0.02

Table 2. Precision, recall and F-measure of classifiers

As the results of our experiments present, regarding precision the Multinomial Naive Bayes Classifier, Support Vector Machine with linear kernel and Linear Logistic Regression has produced the best values. Naive Bayes Classifier was found also in former researches as the best classifier for classification of requirement sentences comparing it other classifiers such as tf-idf classifier[9], k-Nearest Neighbour[9, 8], Bittern Topic Model (BTM) or Latent Dirichlet Allocation (LDA)[8]. Lu Meng et al. have found SVM classifier has performed best in their research [7].

Analysing the results can be seen that the average precision of other methods is lower and the worst result was produced by Multilayer Perceptron model. This outcome could be misleading because both the early stopping was enabled for that process and the number of layers was reduced for performance purpose. Therefore the usability of that model for requirements classification needs to be checked thoroughly. Another classifier which produced weak precision results is the Bernoulli Naive Bayes. This classifier binarizes the feature vector before performs the classification process because its algorithm works on feature vector containing boolean values. Using tf-idf representation this approach has been proved wrong for classification requirement sentences.

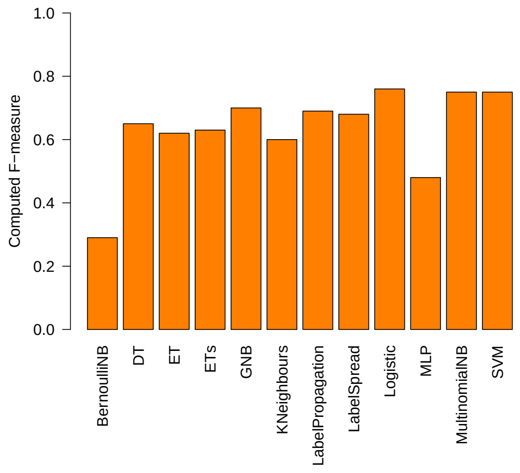

Fig. 2. Computed F-measures of the classifiers

Recall is well balanced among the classifiers as shown in Figure 3.The two exceptions are Bernoulli Naive Bayes and MLP classifiers which have produced also weak precision values.

The average variance of precision is low (under 5 %) in case of the most classifiers except the two mentioned cases. The variance in case of K-Nearest Neighbours is a bit higher (7 %) but its performance is also lower than others'.

The size of the training dataset is small, only 625 examples are presented in NFR database. These data are labeled using 12 classes. We have removed the only portability example but the remaining dataset has contained also small classes as shown in Table 1. Therefore the results presented in this paper are to be validated using a bigger labeled dataset.

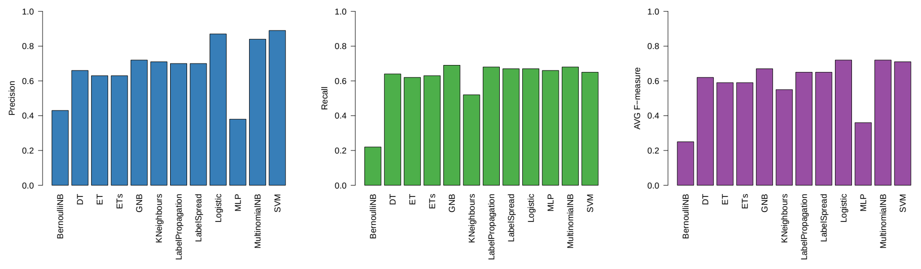

Fig. 3. Average precision, recall and F-measure

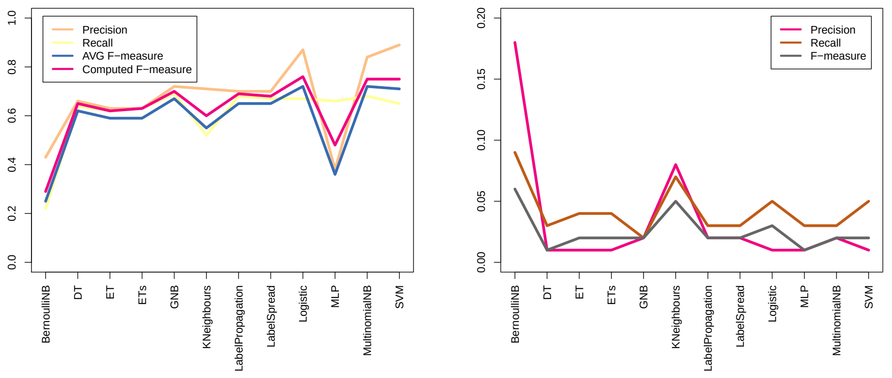

Fig. 4. Classification precision, recall and f-measures (left hand side) and variance statistics (right hand side)

As mentioned above the best classifiers identified by our experiences are similar to those that other research has already demonstrated. To choose the best classifier in practice the execution time is to be considered as well. Table 4 and Figure 5 represents the execution time of classifiers used in our experiments. These values represent the whole execution time of the measurement using the same environment.

According to our experiments, the process based on Multinomial Naive Bayes has produced the best execution time. Based on the precision, recall and the execution time the Multinomial Naive Bayes classifier is the best choice in practice for classification of requirement sentences. As mentioned above this classifier was identified as the best based on its performance by some former research [8, 9] comparing with some other methods. Our research has complemented the comparison with other popular classifiers implemented by sklearn library. However, classifiers like Logistic Regression or Support Vector Machine has performed a little bit better regarding to their precision and recall but considering the execution time, Multinomial Naive Bayes classifier can be denoted as the best choice for practice.

5 Related work

The problem of processing requirements documents using natural language processing and machine learning methods has been a research topic for decades [3]. Although non-functional requirements are less dependent on the application domain, it is not a trivial problem to set up a general list of NFR types. The types identified in the literature are widespread, for example Lawrence Chung et al. [12] identified 156 NFR categories, while Mairiza et al. [13] separated 114

Classifier	Execution time (s)
BernoulliNB	40.12
DT	455.16
ET	30.80
ETs	284.51
GNB	52.37
KNeighbours	274.66
LabelPropagation	120.67
LabelSpread	150.41
Logistic	353.51
MLP	455.82
MultinomialNB	21.61
SVM	343.68

Table 3. Execution time of classifiers

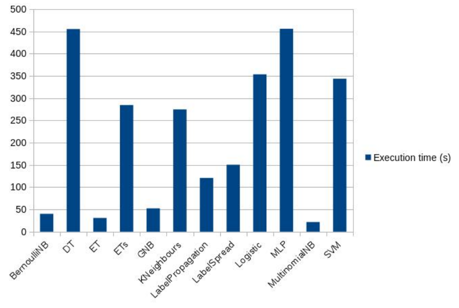

Fig. 5. Execution time of classifiers

different NFR classes in their work, on the contrary to the 6 high-level categories defined by the ISO standard.

A fundamental study of NFR classification is published relatively lately in 2006 by Cleland et al. [14]. They used 14 NFR categories separated from functional requirements. More than 600 requirements from 15 projects were collected and manually categorized to train and test their categorization methods. Although this is still a valuable dataset today, the requirements were originated from university student projects, not from requirements documents of a production system. Cleland et al. achieved high recall with the tradeoff of really low precision. This experiment was reproduced by several researchers in the past [15, 16]. Casamayor et al. [16] employed multinomial naive Bayes classifier coupled with an Expectation Maximization algorithm. Although they improved the precision, the replication of the original study was partial only.

Requirements traceability is a related field, where NLP and information retrieval techniques are frequently applied [17–19]. Hindle et al. [20] used topic modeling to link NFRs to topics found in commit messages. Falessi et al. [21] conducted a large-scale experiment with various NLP techniques including different algebraic models, term weightings and similarity metrics in order to detect identical non-functional requirements.

Sharma et al. [22] addressed the NFR extraction problem with a rule based approach. They implemented a framework for NFR analysis including a DSL language. Sawyer et al. have focused on document archaeology and created an NLP based tool called REVERE to support business analysts in the investigation of different documents containing requirements [23]. This tool has utilized some standard NLP techniques like part-of-speech tagging or semantic tagging and determination of modality.

Denger et al. examined the ambiguity of requirements sentences and investigated the use of language patterns for rewriting these requirements into less ambiguous sentences [24]. The ambiguity of requirements is one of the most prominent issues in requirements engineering which has to be resolved as pointed by Firesmith in his paper [2]. From the perspective of mining software repositories, Paixao et al. [25] investigated the relationship between build results obtained from continuous integration tools with non-functional requirements.

6 Conclusions

Software design and implementation need collected and classified requirements. One of the most important tasks of business analysts is to collect and classify these requirements during elicitation process. The classification process can be a demanding and error-prone task in case of a vast amount of sources. Experiments were conducted to identify appropriate machine learning methods can be used for requirement classification task to support business analysts in their elicitation process. We have compared methods implemented in sklearn library regarding their precision and recall value and also completed this comparison with execution time. As the results of our experiments show the best choice for practice is the Multinomial Naive Bayes classifier. This method has produced very good precision and recall values and the execution time is far the best compared with other processes. This outcome also supports the results of the former researches. We have used the PROMISE dataset which is a small database containing labeled examples. Because of the classification processes depend on strongly the number of learning examples validation have to be performed using a bigger database. Since labeled examples cannot be obtained freely semi-supervised learning methods and also ontology-based methods can be considered. Using topics of Internet forums sentences can be extracted and classified by their topics. This method can produce examples which can be utilized for the learning process of classifiers.

The goal of our future work is to augment the learning examples using sentences extracted from professional topics of the Internet and to find other methods for improving the performance of classifiers.

Acknowledgements

The project has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002).

References

1. Glinz, M.: On Non-Functional Requirements. In: 15th IEEE International Requirements Engineering Conference (RE 2007), IEEE (oct 2007) 21–26
2. Firesmith, D.: Common requirements problems, their negative consequences, and the industry best practices to help solve them. Journal of Object Technology **6**(1) (2007) 17–33
3. Ambriola, V., Gervasi, V.: Processing natural language requirements. In: Proceedings 12th IEEE International Conference Automated Software Engineering, IEEE Comput. Soc (1997) 36–45

4. Li, Y., Guzman, E., Tsiamoura, K., Schneider, F., Bruegge, B.: Automated Requirements Extraction for Scientific Software. Procedia Computer Science **51** (jan 2015) 582–591
5. Rashwan, A., Ormandjieva, O., Witte, R.: Ontology-Based Classification of Non-functional Requirements in Software Specifications: A New Corpus and SVM-Based Classifier. In: 2013 IEEE 37th Annual Computer Software and Applications Conference, IEEE (jul 2013) 381–386
6. Al Balushi, T.H., Sampaio, P.R.F., Dabhi, D., Loucopoulos, P.: ElicitO: A Quality Ontology-Guided NFR Elicitation Tool. In: Requirements Engineering: Foundation for Software Quality. Springer Berlin Heidelberg (2007) 306–319
7. Lu, M., Liang, P.: Automatic Classification of Non-Functional Requirements from Augmented App User Reviews. In: Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering - EASE'17, New York, New York, USA, ACM Press (2017) 344–353
8. Abad, Z.S.H., Karras, O., Ghazi, P., Glinz, M., Ruhe, G., Schneider, K.: What Works Better? A Study of Classifying Requirements. In: Proceedings - 2017 IEEE 25th International Requirements Engineering Conference, RE 2017. (jul 2017) 496–501
9. Casamayor A, Godoy D, C.M.: Identification of non-functional requirements in textual specifications: A semi-supervised learning approach. Information and Software Technology **52**(4) (apr 2010) 436–445
10. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12** (2011) 2825–2830
11. Menzies, T., Caglayan, B., Kocaguneli, E., Krall, J., Peters, F., Turhan, B.: The promise repository of empirical software engineering data (2012)
12. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers, Boston/Dordrecht/London (2000)
13. Mairiza, D., Zowghi, D., Nurmuliani, N.: An investigation into the notion of non-functional requirements. In: Proceedings of the 2010 ACM Symposium on Applied Computing - SAC '10, ACM Press (2010) 311
14. Cleland-Huang, J., Settimi, R., Zou, X., Solc, P.: Automated classification of non-functional requirements. Requirements Engineering **12**(2) (may 2007) 103–120
15. Slankas, J., Williams, L.: Automated extraction of non-functional requirements in available documentation. In: 2013 1st International Workshop on Natural Language Analysis in Software Engineering (NaturaLiSE), IEEE (may 2013) 9–16
16. Casamayor, A., Godoy, D., Campo, M.: Functional grouping of natural language requirements for assistance in architectural software design. Knowledge-Based Systems **30** (jun 2012) 78–86
17. Zou, X., Settimi, R., Cleland-Huang, J.: Improving automated requirements trace retrieval: a study of term-based enhancement methods. Empirical Software Engineering **15**(2) (apr 2010) 119–146
18. Mahmoud, A.: An information theoretic approach for extracting and tracing non-functional requirements. In: 2015 IEEE 23rd International Requirements Engineering Conference (RE), IEEE (aug 2015) 36–45
19. Mahmoud, A., Williams, G.: Detecting, classifying, and tracing non-functional software requirements. Requirements Engineering **21**(3) (sep 2016) 357–381
20. Hindle, A., Ernst, N.A., Godfrey, M.W., Mylopoulos, J.: Automated topic naming. Empirical Software Engineering **18**(6) (dec 2013) 1125–1155

21. Falessi, D., Cantone, G., Canfora, G.: A comprehensive characterization of NLP techniques for identifying equivalent requirements. In: Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '10, ACM Press (2010) 1
22. Sharma, V.S., Ramnani, R.R., Sengupta, S.: A Framework for Identifying and Analyzing Non-functional Requirements from Text Categories and Subject Descriptors. In: Proceedings of the 4th International Workshop on Twin Peaks of Requirements and Architecture, New York, New York, USA, ACM Press (2014) 1–8
23. Sawyer, P., Rayson, P., Garside, R.: REVERE: Support for Requirements Synthesis from Documents. Information Systems Frontiers **4**(3) (2002) 343–353
24. Denger, C., Berry, D.M., Kamsties, E.: Higher quality requirements specifications through natural language patterns. In: Proceedings - IEEE International Conference on Software- Science, Technology and Engineering, SwSTE 2003, IEEE Comput. Soc (2003) 80–90
25. Paixao, K.V., Felicio, C.Z., Delfim, F.M., Maia, M.D.A.: On the interplay between non-functional requirements and builds on continuous integration. In: IEEE International Working Conference on Mining Software Repositories, IEEE (may 2017) 479–482