Automated Extraction of Non-functional Requirements in Available Documentation

John Slankas and Laurie Williams
Department of Computer Science
North Carolina State University
Raleigh, North Carolina, USA
[john.slankas,laurie_williams]@ncsu.edu

Abstract—**While all systems have non-functional requirements (NFRs), they may not be explicitly stated in a formal requirements specification. Furthermore, NFRs may also be externally imposed via government regulations or industry standards. As some NFRs represent emergent system proprieties, those NFRs require appropriate analysis and design efforts to ensure they are met. When the specified NFRs are not met, projects incur costly re-work to correct the issues. *The goal of our research is to aid analysts in more effectively extracting relevant non-functional requirements in available unconstrained natural language documents through automated natural language processing.* Specifically, we examine which document types (data use agreements, install manuals, regulations, request for proposals, requirements specifications, and user manuals) contain NFRs categorized to 14 NFR categories (e.g. capacity, reliability, and security). We measure how effectively we can identify and classify NFR statements within these documents. In each of the documents evaluated, we found NFRs present. Using a word vector representation of the NFRs, a support vector machine algorithm performed twice as effectively compared to the same input to a multinomial naïve Bayes classifier. Our k-nearest neighbor classifier with a unique distance metric had an *F1* measure of 0.54, outperforming in our experiments the optimal naïve Bayes classifier which had a *F1* measure of 0.32. We also found that stop word lists beyond common determiners had no minimal performance effect.**

Keywords-non-functional requirements; natural language processing; machine learning; classification; documentation

I. Introduction

Just as with functional requirements, a system's success depends greatly upon adherence to non-functional requirements (NFRs). When NFRs are missed or ignored, significant, costly issues can arise. A recently deployed U.S. Army intelligence sharing application costing $2.7 billion has been labeled as useless due to capacity, performance, and usability issues [1]. Electronic health record (EHR) systems have been severely criticized for lack of usability, which has been one of the prime reasons why some EHRs adoption have failed [2].

Given the need to analyze and implement NFRs from a wide variety of available sources, system analysts need to quickly identify and categorize NFRs. Business analysts need to ensure completeness of their work. System architects need to understand constraints such as availability, performance, and recovery capabilities to appropriately design architectures to meet those constraints. System integrators, those who deploy large-scale software systems, need to be aware of NFRs in order to place systems into appropriate environments where NFRs such as capacity, security, and operational constraints are appropriately satisfied. Customers need usable applications they can immediately operate with minimal training and with appropriate user interface designs to minimize errors.

Software documentation continues to be a neglected best practice, despite persistent pleas from educators and practitioners alike [3]. Open-source projects often lack formal requirements with developers asserting new requirements themselves. For many open-source projects, archived mailing lists and message boards are the only available development documentation [4]. However, open-source projects typically contain administrative, install, and user manuals because the contained information is necessary for others to utilize the open-source system. Government and industry-based standards contain critical NFRs for projects. Sifting through all of the possible sources for NFRs, though, is a tedious, time-consuming effort.

The goal of our research is to aid analysts in more effectively extracting relevant non-functional requirements in available unconstrained natural language documents through automated natural language processing.

To meet this goal, we developed a tool-based approach, which we call NFR Locator, to classify and extract sentences in existing natural language texts into their appropriate NFR categories. The classification is necessary to determine what role a sentence has. From sentences marked as non-functional, we would then extract critical information specific to each NFR category. For example, access control NFRs would need subjects, resources, and actions extracted to create relevant access control policies. We categorized sentences into one of these 14 NFR categories: (1) access control, (2) audit, (3) availability, (4) capacity and performance, (5) legal, (6) look and feel, (7) maintainability, (8) operational, (9) privacy, (10) recoverability, (11) reliability, (12) security, (13) usability, and (14) other. With NFR Locator, individuals would utilize pre-defined classifiers to locate and extract NFRs from existing natural language documents. If necessary, they can use the tool in an interactive mode to train additional classifiers with new domain or document types.

To evaluate our approach, we developed the following research questions:

- RQ1: What document types contain NFRs in each of the 14 different categories?
- RQ2: What characteristics, such as keywords or entities (time period, percentages, etc.), do sentences assigned to each NFR category have in common?
- RQ3: What machine learning classification algorithm has the best performance to identify NFRs?
- RQ4: What sentence characteristics affect classifier performance?

Our research has the following contributions:

- Process and tool to identify NFRs by categories within available natural language documentation
- Distribution report containing the NFR categories and frequencies by document type
- Empirical performance results for machine learning classifiers on NFRs
- Sentence similarity algorithm to use with a k-nearest neighbor classifier or a k-medoids clustering algorithm.
- Publically-available labeled corpus of identified NFRs[1]

The rest of this paper is organized as follows. Section II reviews the background for this paper. Next, Section III describes related work. Section IV presents our proposed approach, NFR Locator. Then, Section V describes the research methodology. Section VI presents the results of our study. Section VII discusses limitations of the study. Section VIII presents future work. Finally, Section IX concludes the paper.

II. Background

A. Non-functional Requirements

While NFRs have existed since the early days of software engineering, consensus does not exist for the name or the definition of a NFR [5]. Many simply refer to them as the "ilities," the quality aspects of a system. Others have taken to labeling NFRs as systemic requirements [6]. The IEEE Recommended Practice for Software Requirements terms NFRs as constraints [7]. Our concern resides with how NFRs place different constraints on systems, how to quickly identify such constraints, and then to extract relevant information for the NFR. While the number of such constraint categories is rather large (Lawrence Chung et al. identified 156 NFR categories [8]), we choose to concentrate on 14 categories frequently appearing in literature and practical use.

B. Machine Learning and Classification

As our process seeks to identify NFRs from unconstrained natural language texts, we need flexible, yet effective classification methods to handle different documents and multiple ways of expressing similar concepts.. Machine learning provides such a foundation for our work. While techniques and algorithms vary widely in machine learning, they can be generally divided into two primary categories: supervised learning and unsupervised learning. In supervised learning, people train classifiers with labeled data. People and systems then use these classifiers to decide in which class a previously unseen instance belongs. In contrast, unsupervised learning algorithms search data for common patterns (clusters). The data is not directly labeled, but rather groups of common instances are created.

As part of this work, we utilized a k-nearest neighbor classifier (k-NN), which is a supervised algorithm. k-NN classifiers work by classifying a test item based upon which items previously classified are closest to the current test item. The classifier finds the k nearest "neighbors" and returns a majority vote of those neighbors to classify the test item. A distance metric determines the closeness between two items. Euclidean distance often serves as a metric for numerical attributes. For nominal values, the distance is binary - zero if the values are the same or one if they differ. k-NN classifiers may use custom distance functions specific to current problem. Advantages of k-NN classifiers include ability to incrementally learn as new items are classified, to classify multiple types of data, and to handle large number of item attributes. The primary drawback to k-NN classifiers is that if they have n items stored, classification takes $O(n)$ time.

We evaluated other machine learning algorithms including naïve Bayes and Support Vector Machine (SVM). A naïve Bayes classifier works by selecting a class with the highest probability from a set of trained data sets given a specific document. Fundamentally, it assumes that each feature of a class exists independently of other features. Despite such an oversimplification, the approach performs effectively in real-world problems. Naïve Bayes classifiers typically require fewer trained instances than other classifiers. SVM classifiers work by finding the optimal separator between two classes.

C. Classification Evalutation

To compare the results, we used recall, precision, and the F_1 measure. To compute these values, we first need to categorize the classifier's predictions into three categories for each classification value. True positives (TP) are correct predictions. False positives (FP) are predictions in which the sentence of another classification is classified as the one under evaluation. False negatives (FN) are predictions in which a sentence of the same classification under evaluation is placed into another classification. From these values, we define precision (P) as the proportion of corrected predicted classifications against all predictions against the classification under test: $P = TP/(TP + FP)$. We define recall as the proportion of classifications found for the current classification under test: R = TP/(TP+FN). The F_1measure is the harmonic mean of precision and recall, giving an equal weight to both elements: $F_1 = 2 \times \frac{P \times R}{P + R}$. From a NFR perspective, recall is more important than precision in that we want to extract all relevant NFRs from the available documents. However, precision cannot be ignored in that producing large amounts of false positives will frustrate users.

III. Related Work

While text classification, especially with regards to term frequency-inverse document frequency (TF-IDF), has been

[1] https://github.com/RealsearchGroup/NFRLocator

studied for a relatively long period of time [10], NFR classification first appeared in the literature in 2006 [11]. In their work, Cleland-Hung et al. applied TF-IDF with an additional parameter to specify the frequency of indicator terms for a NFR category as compared to the appearance of those terms in the requirement currently under test. To evaluate their scheme, they collected 15 sample requirements specifications from term projects in a Master's level class at DePaul University. They made this collection public through the PROMISE Data Repository [12]. Their work performed extremely well on the basis of a 0.8129 recall, meaning that they successfully found 81% of the possible NFRs in the dataset. However, their precision was a 0.1244 indicating a large number of false positives. While they intentionally choose to maximize recall, users would be frustrated with their process due to the large numbers of false positives to examine and discard.

In 2009, Casamayor et al. [13] repeated the experiment, but with using a multinomial naïve Bayes classifier coupled with an Expectation Maximization algorithm to boost the classifiers performance by labeling untrained data with guesses based upon the already trained data set. Unfortunately, we have not been able to duplicate their results which had an overall accuracy around 0.97. By training a naïve Bayes classifier with their documented approach with all 625 instances in the data set, we have only produced an accuracy result of 0.89 by then testing all of the sentences. While not an appropriate testing approach, the result approximates the maximum theoretical performance on the data set with a given algorithm.

Zhang et al. [14] repeated the experiment again in 2011, but utilized a SVM with a linear kernel as their classifier. They reported significantly higher precision results, although lower recall results than Cleland-Huang et al. but did not provide details. Interestingly, they demonstrated that the performance of individual words was higher than models with multi-words.

Our work builds upon the prior work in that we utilized the same data set as part of our experiments, but we analyzed significantly more examples with multiple document types. We also examined the use of custom distances functions with a k-NN classifier. We studied the effect of different sentence characteristics on classification performance.

IV. NFR Locator

We now present our two-step process, NFR Locator, to extract NFR sentences within existing unconstrained natural language documentation.

For input, the process takes any natural language document related to a project. The process parses the natural language into an internal representation and then based upon relevant features, classifies sentences into specific NFR categories or returns "not applicable" if the sentence does not specify a NFR.

A. Step 1: Parse Natural Language

The process begins by entering the text into the system, parsing the text and converting the parsed representation into NFR Locator's sentence representation (SR). The SR represents each sentence as directed graph where the vertices are words and the edges are the relationships between words.

The tool parses text with the Stanford Natural Language Parser (NLP) and, for each sentence, outputs a graph in the Stanford Type Dependency Representation (STDR) [15]. We choose the STDR as it incorporates the sentence's structural information in a concise and usable format.

From the STDR generated by the parser, we create our SR. Fig. 1 shows the SR for the sentence "The system shall terminate a remote session after 30 minutes of inactivity." Although, in general the SR can be considered a tree, situations exist (primarily due to conjunctions) in which a vertex has multiple parents. Vertexes correspond to words in the sentence and contain the word, the word's lemma and collapsed part of speech. Edges correspond to the relationship between two words (unchanged from the STDR). For prepositions, edges also represent the corresponding word (e.g., "of" and "after" by "prep_of" and "prep_after"). Utilizing a pre-order traversal, the process creates the SR from the Stanford graph. As each vertex is created, we make two changes to the nodes. First, to avoid multiple versions of the same word, we use the lemma[2] of the original word. Second, to avoid differences in the part of speech, we collapse the parts of speeches for all nouns, verbs, and adjectives to their base category. For example, we treat all plural nouns and proper nouns as just nouns. Similarly, verbs with different tenses are treated collectively as a single group. We also utilize a very small stop word list to remove common determiners[3] from the SR as demonstrated in Figure 1 with the dashed lines.

B. Step 2: Classify Sentences

Once the tool completes the parsing and initial analysis of a sentence, a k-NN classification algorithm classifies each sentence into one or more NFR categories. Sentences classified besides "not applicable" appear on generated reports from the tool for use outside of the system.

A k-NN classifier predicts a classification by taking a majority vote of the existing classifications of the k nearest neighbors to the item under test. Thus, in our situation, to

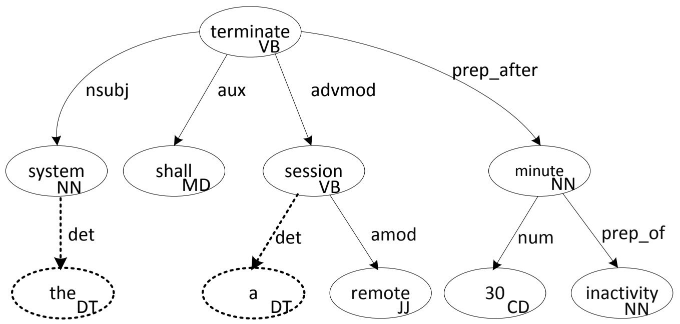

Figure 1. Sentence Representation

[2] A lemma is the base word form for a set of words. For instance, sang, sing, and sung all have the same lemma, "sing." Lemmas are more precise than stems as they take into account part of speech and other factors.

[3] a, an, the

classify a sentence into one of the 14 categories, the classifier needs to find which existing classified sentences are most similar to the current sentence under test. k-NN classifiers use a distance metric to find the closest neighbors. This metric is the sum of the differences among the attributes used to determine the classification. Typically, Euclidean distance serves as a metric for numerical attributes while for nominal values, the distance is generally considered to be zero if both attribute values are the same or one if they differ. Our situation is more complex as we have a variable number of attributes to consider for each sentence based upon the sentence length. Additionally, certain words may be more closely related to one another than other words. As such, we need to utilize a custom distance metric to compute a value representing the difference between two sentences.

Our distance metric is a modified version of Levenshtein distance [16]. Rather than using the resulting number of edits to transform one string into another as the value as the Levenshtein distance does, our metric computes the number of word transformations to change one sentence into another. Rather than strictly using just zero or one as the difference between words, the metric uses the function defined in Fig. 2. The function first checks the structure of graph around each vertex to ensure it corresponds to other vertex. Next, the functions checks to see if the two vertices are the same (lemmas are equal). In line 7, we check if both words are numbers. Next line 8 checks to see if both words are the same type of named entity such as a person or an organization. Then in line 9, the process checks to see if the two words are related through sets of cognitive synonyms (synsets) within WordNet[4] via semantic relationships (hypernym or hyponym). If a relationship value is found, then a value between 0.1 and 0.4 is returned based upon the number of relationships traversed. Finally, a default value of 1 is returned if none of the other conditions are met.

Once the classification is complete, the user may review the predicted classifications and related sentences. If necessary, they can correct the classifications within the tool. As the user completes each classification, those classifications would be used within the k-NN classifier as additional sentences are processed.

```
computeVertexDistance(Vertex a, Vertex b)
 1: if a = NULL or b = NULL return 1
 2: if a.partOfSpeech <> b.partOfSpeech return 1
 3: if a.parentCount <> b.parentCount return 1
 4: for each parent in a.parents
 5:     if not b.parents.contains(parent) return 1
 6: if a.lemma = b.lemma return 0
 7: if a and b are numbers, return 0
 8: if ner classes match, return 0
 9: wnValue = wordNetSynonyms(a.lemma,b.lemma)
10: if wnValue > 0 return wnValue
11: return 1
```

Figure 2. Compute Vertex Distance Logic

[4] http://wordnet.princeton.edu/

V. Research Method

This section describes the context regarding our problem and NFR categories. We also discussed how we collected relevant documents and prepared those documents for use within our evaluations.

A. Context

We chose the EHR domain to initially evaluate our research questions. A wide variety of open and closed source EHRs, various industry standards (HL7 [5], CCHIT [6]), governmental regulations, and other document sources exist to elicit documentation. While not directly to healthcare, we included the PROMISE NFR Data Set [12] in our evaluations to provide comparisons to prior research.

The nine initial NFR categories were chosen based upon existing NFR classification work [11,13,14] with the PROMISE NFR Data Set. From industry experience, we added reliability and recoverability. We also combined performance and scalability into a single category as the two concepts are considered together in system architecture and design activities. To meet future research needs, we separated access control and audit from security. As there were specific NFRs that did not readily fall into any other existing categories, we added "other" to categorize these NFRs. By using the "other" category, we avoid placing NFRs into categories in which they do not belong. If the "other" NFRs were placed into existing categories, machine learning algorithms would perform less effectively due to irrelevant terms and sentences within a specific category.

B. Study Procedure

To perform this study, we first collected a series of 11 documents related to EHRs. The specific source locations are listed within our available labeled corpus[7]. For requirement specifications, we utilized the CCHIT Ambulatory Requirements, iTrust [17], and the PROMISE NFR Data Set. To represent documents from an open-source project without a formal requirements document, we utilized the OpenEMR[8] Install Manual and User Manual. We analyzed two data use agreements (DUAs) from the Centers for Medicare & Medicaid Services[9] and the North Carolina Department of Public Health. DUAs are legal contracts among two or more parties that specify what data is shared, who can access the data (authorizations), and for what purpose the data may be used (purposes) [18]. We also used two request for proposals (RFPs) from organizations within the state of California for EHR systems. Many organizations use RFPs to solicit vendor bids for software systems. The RFPs often contain detailed lists of requirements for vendors to provide a response as to their capabilities in meeting a particular requirement. To

[5] http://hl7.org

[6] http://cchit.org

[7] https://github.com/RealsearchGroup/NFRLocator

[8] http://www.open-emr.org/

[9] http://www.cms.gov/

represent governmental regulations, we utilized three sections of the United States Code of Federal Regulations (CFRs) related to healthcare.

We then converted those 11 documents into a text-only format using the "save as" feature in the relevant document application. The only changes made to the resulting text files were to account for misplaced line breaks and to remove table-based information that the natural language processor cannot process. Next, we imported each document into NFR Locator. First, we performed cluster analysis on the imported document to look for common sentences and detect patterns among those sentences. Each sentence (or line) in the file was then manually classified into (1) one or more NFR categories; (2) functional (FT); or (3) not-applicable (NA). We allowed a sentence to be placed into multiple categories with the exception of "not-applicable." For example, we categorize "The system has capability to electronically provide patient reports on demand following and allow for hiding private information to comply with HIPAA Privacy and Security requirements" into functional and into two NFR categories (legal and privacy) due to the different sentence elements.

We validated the classifications through several approaches. First, we computed clusters for all of the different sentences within a document and then compared the assigned labels and sentence content. We also generated listings by classification of the sentences to check for mislabeled sentences. As we labeled documents, we used already classified sentences in the k-NN classifier to examine similarities among the current sentence being labeled and those already classified. Any discrepancies in the predicted categories could easily be traced back to the source sentences and appropriate changes made. An internal validity threat may exist as the first author performed all of the initial sentence classifications. To check if this was a significant issue or not, we had six software developers classify a representative sample of 30 sentences. Utilizing Randolph's Online Kappa Calculator [19], we had a free-marginal kappa of 0.714, indicating adequate inter-rater agreement, by comparing the first authors selections against the majority vote of the other raters.

Once documents are completely labeled, we performed various experiments on the labeled data through the use of different classification algorithms and sentence features. To evaluate classifiers, we tested each classifier with a stratified n-fold cross-validation and computed the precision, recall, and F_1 measure. With the n-fold cross-validation, data is randomly partitioned into n folds based upon each fold of approximately equal size and equal response classification. For each fold, the classifiers are trained on the remaining folds and then the contents of the fold are used to test the classifier. The n results are then averaged to produce a single result. We follow Han et al.'s recommendation [9] and use 10 as the value for n as this produces relatively low bias and variance. The cross-validation ensures that all sentences are used for training and each sentence is tested just once. For our evaluations, we utilized standard machine learning classifiers within Weka [20] suite in addition to the k-NN classifier and distance function presented in Section 4. To generate input data for the Weka classifiers, we exported the original sentences as text strings along with Boolean flags for each of the NFR categories. We converted all strings to word vectors through a Weka filter. As the Weka classifiers do not natively support multi-label classifications, we train a Weka classifier for each NFR category. Each category is independently test for membership.

VI. Evaluation

This section presents our evaluation of the research questions.

RQ1: What document types contain NFRs in each of the 14 different categories?

With RQ1, we look to see which document types are better sources for each of the 14 NFRs categories. Table 1 presents the documents with their classification breakdown. (Column abbreviations are listed in Table 2.) Our breakdown for the PROMISE NFR Data Set [12] differs due to category changes and for classifying each sentence individually.

All evaluated document contained NFRs. RFPs had a wide variety of NFRs with the exception of look and feel NFRs. Due to their directed purpose, DUAs contained high frequencies of legal and privacy NFRs. Access control and/or security NFRs appeared in all of the documents. The low frequency of functional and NFRs with CFRs exemplifies why tool support is critical to efficiently extract requirements from those documents.

RQ2: What characteristics, such as keywords or entities do sentences assigned to each NFR category have in common?

To evaluate RQ2, we preformed two tasks. First, we extracted the top 20 keywords for each category based upon their ability to predict whether a sentence belongs to a specific category. Using Equation 1, we ranked the probability of every keyword for each NFR category. The first and second parts of the equation are the term frequency-inverse document frequency (TF-IDF) commonly used for information retrieval. The third term applies a selectivity factor to the probability to improve results where the term resides primarily in one classification versus multiple categories. $N_{K,C}$ represents the number of sentences containing the keyword for a particular category. N_C is the total number of sentences in the category. N denotes the total number of sentences. N_K is the number of sentences containing the keyword. $tf-idf_C$ equals the TF-IDF for a particular category and term. We eliminated any results in which there were not at least three occurrences of a keyword within a class.

$$P_k = \frac{N_{K,C}}{N_C} \times \log(\frac{N}{N_K}) \times \frac{tf-idf_C}{\sum_{i \in C} tf-idf_i} \quad (1)$$

Table 2 presents the top 20 keywords found for each category. The terms extracted for the maintenance category primarily represent specific standards healthcare system must implement. In future work, we will break these requirements into their own category, "data standards." Also note that document specific words do appear in Table 2. This issue

TABLE 1. CLASSIFIED DOCUMENTS BY REQUIREMENTS CATEGORY

Document	Document Type	Size	AC	AU	AV	LG	LF	MT	OP	PR	PS	RC	RL	SC	US	OT	FN	NA
CCHIT Ambulatory Requirements	Requirement	306	12	27	1	2	0	10	0	0	1	5	2	28	4	8	228	6
iTrust	Requirement, Use Case	1165	439	44	0	2	2	18	2	9	0	9	9	55	2	0	734	376
PromiseData	Requirement	792	164	20	36	10	50	26	89	7	75	4	12	71	101	19	340	0
Open EMR Install Manual	Installation Manual	225	3	0	0	0	0	0	5	1	0	6	1	25	0	0	2	184
Open EMR User Manual	User Manual	473	169	0	0	0	14	0	0	0	0	0	0	8	4	0	286	95
NC Public Health DUA	DUA	62	1	0	0	20	0	0	0	4	0	0	0	1	0	0	0	41
US Medicare/Medicaid DUA	DUA	140	1	0	0	26	0	0	0	17	0	0	0	0	0	5	2	108
California Correctional Health Care	RFP	1893	94	120	9	85	0	133	94	52	13	16	13	193	14	38	987	409
Los Angeles County EHR	RFP	1268	58	37	8	3	2	28	19	3	11	8	13	108	21	10	639	380
HIPAA Combined Rule	CFR	2642	28	8	3	0	0	78	0	213	0	9	0	41	1	0	317	2018
Meaningful Use Criteria	CFR	1435	0	0	0	0	0	0	0	0	0	0	0	8	0	0	116	1311
Health IT Standards	CFR	1475	10	20	0	0	0	119	0	1	0	2	2	71	1	2	164	1146
Total		**11876**	**979**	**276**	**57**	**152**	**68**	**413**	**207**	**300**	**100**	**50**	**43**	**563**	**148**	**82**	**3568**	**6076**

could be resolved by adding an additional term to Equation 1 to account for document specificity as Cleland-Huang et al. performed [11] or by adding more documents to the corpus.

In the second task for RQ2, we looked at selected NFR categories to find commonality. Within availability NFRs, we found that 30 sentences contained some form of the word "available", 20 sentences contained a time period, nine expressed a percentage, six stated some form of a maintenance window, and five mentioned replication. Only four of the 57 sentences did not contain one of these features. (Note: a sentence could express more than one of the features.) Depending upon the presence of those features in other requirements, they may be suitable candidates for use in classifiers for their ability to discern availability NFRs. Alternatively, a set of hand coded rules could be used to process availability NFRs as there appears to be a relatively small number of patterns for this NFR category.

TABLE 2. TOP 20 KEYWORDS BY CATEGORY

NFR Category	Keywords
Access Control (AC)	choose, lhcp, hcp, visit, privilege, read, office, add, representative, sort, name, administrator, personal, dlhcp, view, status, accessor, edit, role, list
Audit (AU)	authorship, trail, arise, worksheet, auditable, exclusion, reduction, deletion, examine, editing, stamp, non-repudiation, inclusion, id, alteration, finalize, disable, summarize, attestation, log
Availability (AV)	achieve, 24, availability, 98, addition, available, 99, hour, day, online, schedule, confidentiality, resource, technical, year, transmit, integrity, maintenance, %, period
Legal (LG)	infeasible, custodian, hipaa, breach, dua, discovery, iihus, publication, iihi, recipient, delay, secretary, definition, harm, scope, jurisdictional, affect, derive, vocabulary, reuse
Look & Feel (LF)	appearance, scheme, tree, radio, appeal, color, look, navigation, sound, feel, ship, left, shot, menu, ccr, button, corporate, page, openemr, employer
Maintenance (MT)	4010, washington, ibr, x12n, asc, 2002, addenda, 837, september, 1999, 1.1, tele-communication, 5.1, astm, draft, february, january, 2010, context-aware, infobutton
Operational (OP)	mysql, microsoft, euhr, soms, letter, infrastructure, interoperability, connect, cchcs, machine, browser, platform, cardmember, central, cdcd, extraction, cchc, model, registry, interchange
Privacy (PR)	health, protected, entity, disclose, covered, use, disclosure, individual, such, purpose, law, permit, other, section, require, plan, person, paragraph, care, request
Recoverability (RC)	restore, credentials, backup, back, recovery, disaster, previous, emergency, establish, copy, state, need, implement, loss, plan, event, failure, organization, business, hour
Performance & Scalability (PS)	fast, simultaneous, 0, second, scale, capable, increase, peark, longer, average, acceptable, lead, handle, flow, response, capacity, 10, maximum, cycle, distribution
Reliability (RL)	reliable, dependent, validate, validation, input, query, accept, loss, failure, operate, alert, laboratory, prevent, database, product, appropriate, event, application, capability, ability
Security (SC)	cookie, encrypted, ephi, http, predetermined, strong, vulnerability, username, inactivity, portal, ssl, deficiency, uc3, authenticate, certificate, session, path, string, password, incentive
Usability (US)	easy, enterer, wrong, learn, word, community, drop, realtor, help, symbol, voice, collision, training, conference, easily, successfully, let, map, estimator, intuitive

RQ3: What machine learning classification algorithm has the best performance to identify NFRs?

In addition to the k-NN classifier, we used the multinomial naïve Bayes and the sequential minimal optimization (SMO)[10] classifiers in Weka [20]. We also used two random methods, weighted and 50%. The weighted random model assumes that the classifier knows the population proportions within the training set and randomly returns answers proportionally to the existing classifications. The 50% model has no knowledge of the training set and randomly returns answers with equal likelihood between the two classifications values. As we allow for each sentence to have one or more classifications, we test each category individually for each classifier and produce the precision, recall, and F_1 values. We then average the results of all of the categories for each classifier to produce the results displayed in Table 3. This "macro-averaging" allows for each category to be equally represented in the result. If we used "micro-averaging" and allowed the categories to be weighted in terms of their size, the "functional" category would have dominated the results[11]. For each classifier, we repeat this process five times. The average results for precision, recall, and the F_1 measure are displayed in Table 3. The standard deviation (SD) for the F_1 measure is presented as well. For the k-NN classifier, we report $k = 1$ as it had the best performance for k in our experirment.

TABLE 3. STRATRIFIED 10-FOLD CROSS VALIDATION

Classifier	Precision	Recall	F_1	F_1 SD
Weighted Random	.047	.060	.053	.0042
50% Random	.044	.502	.081	.0016
Naïve Bayes	.227	.347	.274	.0043
SMO	.728	.544	.623	.0132
NFR Locator k-NN	.691	.456	.549	.0047

As the NFR Locator k-NN classifier did not perform as well as the SMO classifier, future work will utilize SMO as the primary classifier. We will continue to use the k-NN classifier to display closely related sentences. Comparing our classifier to the initial work performed by Cleland-Huang et al. for just the 15 documents in the PROMISE NFR Data Set, our F_1 measure was .382 and theirs was .239.

RQ4: What sentence characteristics affect classifier performance?

Classifiers use any number of sentence characteristics (features) to make decisions. Sentence features include number of words, the words themselves, words represented as distance vectors, stems, lemmas, parts of speech, and known semantic entities. From a classification perspective, we need to utilize the features most useful and ignore other features providing little or no value. In Table 4, we present the results of using different word forms and stop words with the naïve Bayes and SMO classifiers. The "original" word form represents the word as it appeared in the sentence. "Lemma" is the lemma of the original word as produced by the Stanford NLP. "Porter" is stem of the word produced by the Porter stemming algorithm [21]. "Casamayor" signifies the pre-processing steps Casamayor et al. performed [13]. Stop words are lists of words to exclude from use in the classifiers due to the commonality of the words. "Determiners" are "a," "an," and "the." "Frakes" [22] contains 400 words and "Glasgow" contains 319 words[12]. By just eliminating the determiners, Naïve Bayes' F1 measure increased by 0.027 than with just the original sentence. Using additional stop words, lemmas, or stems had no more than 0.01 change in effectiveness. With SMO, effectiveness decreased by 0.019 using lemmas. Longer stop word lists appeared to have no effect for SMO. Each test was repeated five times with the average F_1 measure reported along with the standard deviation.

TABLE 4. WORD TYPE AND STOP WORDS

Model	Word Form	Stop Words	F_1	F_1 SD
Naïve Bayes	Original	Determiners	.291	.0022
Naïve Bayes	Porter	Determiners	.287	.0021
Naïve Bayes	Lemma	Determiners	.292	.0032
Naïve Bayes	Lemma	Frakes	.297	.0021
Naïve Bayes	Casamayor	Glasgow	.327	.0018
SMO	Original	Determiners	.603	.0044
SMO	Lemma	Determiners	.584	.0039
SMO	Lemma	Frakes	.586	.0042

We examined the use of named entities (dates, durations, numbers, person, time, etc.) as sentence features. By representing the presence of a named entities as binary flags, we slightly increased the F_1 measure of the SMO classifier for the availability and reliability categories by 3.5%.

VII. LIMITATIONS

Several limitations exist for our work. As NFR Locator works on textual sentences, the process cannot extract information contained in images and tables. Within text documents, the process loses document structural information contained within the original Microsoft Word and Adobe PDF files. In several cases, there were lists of items that when examined as a whole would have been classified into a particular NFR category. To mitigate this limitation, we can directly parse document files in their native formats or utilize document structural analysis to discover titles, sections, lists and other document elements. Another limitation exists in that we do not know how our process and tool will generalize to other systems and domains.

VIII. FUTURE WORK

Two primary directions exist to extend this work. The first direction is to expand the corpus both within the healthcare domain, but also to expand it into other domains. By using other domains, we can see how those domains compare to healthcare in terms of the distribution of the NFRs across

[10] SMO is a Support Vector Machine classifier

[11] Excluding the random classifiers, all classifiers had F_1 values above .8 for the "functional" category.

[12] http://ir.dcs.gla.ac.uk/resources/linguistic_utils/stop_words

different document types. We can also evaluate how classifiers trained within one domain perform in other domains. We surmise that for certain NFR categories in which little domain specific text exists, the performance should be relatively the same as staying within the same domain. However, for other NFR categories such as access control with more domain or problem specific texts, the classifier may not perform as well.

The other direction of work we plan to pursue is to use information extraction techniques to derive specific data elements from the classified NFR sentences. As is the case in this current work, we will incorporate machine learning algorithms so the process can learn and apply new patterns as more and more natural language text has been evaluated

IX. Conclusion

In this paper, we presented a new process, NFR Locator, and a tool to assist analysts in effectively extracting relevant NFRs from a wide variety of documents. To evaluate the tool, we collected and made available a series of system-related documents in the healthcare domain to identify and extract NFRs. We demonstrated that NFRs exist in a wide variety of documents. In analyzing specific NFR categories, we found specific features unique to those categories that could be used by programs to identify other NFRs in the same category. We evaluated multiple classifiers to identify NFRs in specific categories and found SMO had the highest effectiveness. Our k-nearest neighbor classifier with a unique distance metric had a F_1 Measure of 0.54, outperforming in our experiments the optimal naïve Bayes classifier which had a F_1 Measure of 0.32. We also found that stop word lists beyond common determiners had no little performance effect.

Acknowledgment

This work was supported by the U.S. Army Research Office (ARO) under grant W911NF-08-1-0105 managed by NCSU Secure Open Systems Initiative (SOSI). We would like to thank the North Carolina State University Realsearch group for their helpful comments on the paper.

References

[1] C. Hoskinson, “Army’s Faulty Computer System Hurts Operations,” *Politico*, 2011. [Online]. Available: http://www.politico.com/news/stories/0611/58051.html.

[2] J. Bertman and N. Skolnik, “EHRs Get a Failing Grade on Usability,” *Internal Medicine News*, vol. 43, no. 11, p. 50, Jun. 2010.

[3] S. C. B. de Souza, N. Anquetil, and K. M. de Oliveira, “A Study of Documentation Essential to Software Maintenance,” in *Proceedings of the 23rd annual international conference on Design of communication documenting & designing for pervasive information - SIGDOC ’05*, 2005, pp. 68–75.

[4] J. Noll and W.-M. Liu, “Requirements elicitation in open source software development,” in *Proceedings of the 3rd International Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development - FLOSS ’10*, 2010, pp. 35–40.

[5] M. Glinz, “On Non-Functional Requirements,” in *15th IEEE International Requirements Engineering Conference (RE 2007)*, 2007, pp. 21–26.

[6] J. Browne, “Systemic Requirements,” 2008. [Online]. Available: http://www.julianbrowne.com/article/viewer/systemic-requirements. [Accessed: 01-Oct-2013].

[7] *IEEE Recommended Practice for Software Requirements Specifications - IEEE Std 830-1998*, vol. 1998, no. October. 1998.

[8] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-functional Requirements in Software Engineering*. Norwell, MA, USA: Kluwer Academic Publishers Group, 2000, p. 439.

[9] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. Morgan Kaufmann, 2011, p. 744.

[10] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. New York, NY, USA: McGraw-Hill, Inc., 1986.

[11] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc, “Automated Classification of Non-functional Requirements,” *Requirements Engineering*, vol. 12, no. 2, pp. 103–120, Mar. 2007.

[12] T. Menzies, B. Caglayan, E. Kocaguneli, J. Krall, F. Peters, and B. Turhan, “The PROMISE Repository of Empirical Software Engineering Data,” *West Virginia University, Department of Computer Science*, 2012. [Online]. Available: http://promisedata.googlecode.com/.

[13] A. Casamayor, D. Godoy, and M. Campo, “Identification of non-functional requirements in textual specifications: A semi-supervised learning approach,” *Information and Software Technology*, vol. 52, no. 4, pp. 436–445, Apr. 2010.

[14] W. Zhang, Y. Yang, Q. Wang, and F. Shu, “An Empirical Study on Classification of Non-Functional Requirements,” in *The Twenty-Third International Conference on Software Engineering and Knowledge Engineering (SEKE 2011)*, 2011, pp. 190–195.

[15] M.-C. de Marneffe, B. MacCartney, and C. Manning, “Generating Typed Dependency Parses from Phrase Structure Parses,” *Proceedings of Language Resources and Evaluation*, pp. 449–454, 2006.

[16] V. I. Levenshtein, “Binary Codes Capable of Correcting Deletions, Insertions, and Reversals,” *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707–710, 1966.

[17] A. Meneely, B. Smith, and L. Williams, “iTrust Electronic Health Care System: A Case Study,” in *Software System Traceability*, 2011.

[18] J. Y. Schmidt, A. I. Antón, L. Williams, and P. N. Otto, “The Role of Data Use Agreements in Specifying Legally Compliant Software Requirements,” in *Fourth International Workshop on Requirements Engineering and Law*, 2011, no. Relaw, pp. 15–18.

[19] J. J. Randolph, “Online Kappa Calculator,” 2008. [Online]. Available: http://justusrandolph.net/kappa/. [Accessed: 02-Apr-2013].

[20] M. Hall, H. National, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The WEKA Data Mining Software : An Update,” *SIGKDD Explorations*, vol. 11, no. 1, pp. 10–18, 2009.

[21] M. Porter, “An algorithm for suffix stripping,” *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[22] W. B. Frakes, “Information retrieval,” W. B. Frakes and R. Baeza-Yates, Eds. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1992, pp. 1–12.