Semantic Analysis of Functional and Non-Functional Requirements in Software Requirements Specifications

Abderahman Rashwan

Department of Computer Science and Software Engineering
Concordia University, Montréal, Québec, Canada
a_rashwa@encs.concordia.ca

1 Introduction

Software Requirements Specifications (SRS) documents are important artifacts in the software industry. A SRS contains all the requirements specifications for a software system, either as functional requirements (FR) or non-functional requirements (NFR). FRs are the features of the system-to-be, whereas NFRs define its quality attributes. NFRs impact the system as a whole and interact both with each other and with the functional requirements. SRS documents are typically written in informal natural language [1], which impedes their automated analysis. The goal of this work is to support software engineers with semantic analysis methods that can automatically extract and analyze requirements written in natural language texts, in order to (i) make SRS documents machine-processable by transforming them into an ontological representation; (ii) apply quality assurance (QA) methods on the extracted requirements, in order to detect defects, like ambiguities or omissions; and (iii) attempt to build traceability links between NFRs and the FRs impacted by them, in order to aid effort estimation models.

2 Research Plan

The core of this work is based on methods from Natural Language Processing (NLP) and text mining, machine learning, as well as web ontologies (OWL) [2] and their population from text [3]. The work is divided into 3 phases:

(1) Extraction of Requirements and Ontology Population: the goal here is to detect requirements and classify them into FRs/NFRs by building text mining pipelines that find candidate sentences, classify them using a machine learning model, and populate the results into an ontology modeling the domain of SRS.
(2) Quality Assurance [4]: To support requirements engineers, a number of quality metrics will need to be defined and implemented to measure both the intrinsic quality of a single requirements statement (e.g., measurability) and the coverage of the complete specification. For FRs, this requires the addition of a domain ontology defining the concepts of the application domain.

(3) Traceability: The final step concerns the impact of the NFRs on other parts of a specified system, in particular the FRs. We plan to develop and evaluate different methods to create these links, based on existing project data. The output of this step will be provided as input to a connected project on effort estimation, which is based on the COSMIC method (Common Software Measurement International Consortium) [5].

All three steps will be evaluated on both publicly available and in-house datasets.

3 SRS Classifier

The first phase is the extraction and classification of requirements. So far, we built a machine learning-based FR/NFR classifier for SRS documents. The goal of this module is to classify input sentences into 4 major categories, with 8 classes: FR (Functional Requirements), Design Constraints, NR (Not a Requirement) and several types of NFRs (security, efficiency, reliability, functionality, usability and maintainability). Example sentences are: *"The ASPERA-3 data set shall be stored on a local SwRI archive"* (FR), *"The APAF ground data system shall have built-in error handling"* (NFR), and *"Section 4 contains general information to aid in the understanding of this specification"* (NR).

For processing the SRS documents, we use GATE [6], the General Architecture for Text Engineering, as a development tool. A custom text mining pipeline detects candidate sentences, classifies them using machine learning algorithms, including SVM, PAUM (Perceptron Algorithm with Uneven Margins), Naive Bayes, KNN and the C4.5 decision tree algorithm. Training is performed on data collected and annotated on an in-house data set (described below). The performance of the developed solution is measured using standard metrics, such as precision, recall, and F-measure.

3.1 Corpus

The gold standard corpus contains 3 manually annotated Software Requirements Specification documents, containing 2616 sentences. The total number of all types of NFR sentences in the corpus is 65, FR is 713, design constraint is 48 and NR is 1790. We separated these 3 SRS into 10 balanced documents to be able to perform a 10-fold cross validation.

The manual annotation task, as well as the consolidation into a gold standard, was previously carried out by several people within Concordia[1], using GATE Teamware, a collaborative, web-based corpus annotation tool.

3.2 Training

Training is performed by a GATE pipeline that first extracts features from the documents, which are then fed into a machine learning component. This pipeline

[1] Not yet published.

Table 1. Results for the SVM classifiers (left): Column # is the number of sentences in the corpus. Errors (right): Column K is the key (manual) and R the response (system).

Class	#	Prec.	Recall	F1
FR	713	94.22%	94.09%	94.15%
Constraint	48	97.91%	97.61%	97.83%
Security	38	98.08%	98.9%	98.04%
Functionality	14	99.32%	99.16%	99.24%
Efficiency	7	99.33%	99.17%	99.25%
Reliability	3	99.44%	99.48%	99.46%
Usability/Utility	2	99.69%	99.49%	99.41%
Maintainability	1	99.35%	99.49%	99.41%
Average		98.41%	98.42%	97.03%

Class	K	R	K	R	K	R	K	R	K	R	K	R
FR	F	T	T	F	T	F	F	F	T	F	F	F
Con.	F	F	F	F	F	F	F	F	F	F	F	T
Sec.	F	F	F	F	T	F	F	T	T	T	F	F
Fun.	F	F	F	F	F	F	F	F	F	F	F	F
Eff.	F	F	F	F	F	F	F	F	F	F	F	F
Rel.	F	F	F	F	F	F	F	F	F	F	F	F
Usa.	F	F	F	F	F	F	F	F	F	F	F	F
Main.	F	F	F	F	F	F	F	F	F	F	F	F
# Err.	42		12		9		8		1		1	

contains Processing Resources (PRs), in particular the ANNIE components [6] and a machine learning PR. To obtain the features for machine learning, documents are pre-processed by using the ANNIE English Tokeniser PR, ANNIE Sentence Splitter, ANNIE Part-of-Speech (POS) tagger, and finally the GATE Morphological Analyzer.

In our evaluation, Support Vector Machines (SVMs) with third order polynomial kernel provided the best performance. The features used for training are the unigram of the sentences' tokens using its POS tagging and its root. Instead of a multi-classification, we perform a binary classification for each type of FR/NFR, as some sentences contain two or more types of requirements; e.g., *"Web-based displays of the most current ASPERA-3 data shall be provided for public view"* is annotated as both a functional requirement (FR) and design constraint.

3.3 Evaluation

The ML classifier is evaluated with the metrics precision, recall and F-measure:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \text{F-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Here, TP (true positive) is the number of correctly classified requirements, FP (false positive) the number of requirements incorrectly classified and FN (false negative) the number of requirement incorrectly not classified.

The results are summarized in Table 1 (left): the overall average F1 measure is 97.03%. The results are copacetic for a number of reasons, one of them that SVM gives generally good results for text classification. For each NFR, we only have a limited number of training sentences, and finally the binary classification for each class gives better results than one multi-classification. Table 1 (right) contains an analysis of all error types, i.e., 42 of the sentences annotated as a NR are mis-classified as a FR, and another 8 are mis-classified as 'security'.

3.4 Related Work

Several attempts have been made to develop automated tools for detecting and classifying requirements from SRS. Cleland-Huang et al. [7] used supervised learning with weighted indicators; Hussain et al. [8] also used a supervised learning approach, but with decision trees; and Casamayor used a semi supervised learning with TF-IDF technique [9]. My work presented here used a new corpus created at Concordia, which contains some additional classes, such as NR and constraints.

4 Conclusions

This paper presents our preliminary results on the semantic analysis of SRS documents. Future work includes increasing the size of the training and testing data, developing additional syntactic and semantic features for the classifiers, and populate the results into an ontology, using the OwlExporter [3], for further reasoning on them.

Surveys [4] show that incomplete requirements are a primary source of software projects failures. Moreover, there is a lack of available methods and tools that aid software engineers in managing requirements. The results of this work will be of interest to researchers as well as practitioners from industry, who are interested in estimating the effort for building requirements in general and improving software quality in particular, and use measurement data in requirements engineering.

References

1. Luisa, M., Mariangela, F., Pierluigi, I.: Market research for requirements analysis using linguistic tools. Requirements Engineering 9(1), 40–56 (2004)
2. Baader, F., Calvanese, D., MacGuinness, D., Nardi, D., Patel-Schneider, P.: The description logic handbook: theory, implementation and applications, 2nd edn. Cambridge University Press (2007)
3. Witte, R., Khamis, N., Rilling, J.: Flexible Ontology Population from Text: The OwlExporter. In: The Seventh International Conference on Language Resources and Evaluation (LREC 2010), Valletta, Malta, ELRA, May 19–21, pp. 3845–3850 (2010)
4. Lamsweerde, A.V.: Requirements Engineering: From System Goals to UML Models to Software Specifications, 1st edn. Wiley (2009)
5. Abran, A., Descharnais, J.M., Oligny, S., Pierre, D.S., Symons, C.: The COSMIC implementation guide for ISO/IEC 19761: 2003 (2009)
6. Cunningham, H., et al.: Text Processing with GATE (Version 6). University of Sheffield, Department of Computer Science (2011)
7. Cleland-Huang, J., Settimi, R., Zou, X., Solc, P.: The detection and classification of non-functional requirements with application to early aspects. In: Proceedings of 14th IEEE International Conference on Requirements Engineering, Minneapolis/St. Paul, MN, pp. 39–48 (2006)
8. Hussain, I., Kosseim, L., Ormandjieva, O.: Using Linguistic Knowledge to Classify Non-functional Requirements in SRS documents. In: Kapetanios, E., Sugumaran, V., Spiliopoulou, M. (eds.) NLDB 2008. LNCS, vol. 5039, pp. 287–298. Springer, Heidelberg (2008)
9. Casamayor, A., Godoy, D., Campo, M.: Semi-supervised classification of non-functional requirements:an empirical analysis 13, 35–45 (2009)