Can Text Mining Assistants Help to Improve Requirements Specifications?

Bahar Sateli, Elian Angius, Srinivasan Sembakkam Rajivelu, and René Witte*
Semantic Software Lab
Department of Computer Science and Software Engineering
Concordia University, Montréal, QC, Canada
{b_satel, e_angiu, s_sembak, rwitte}@encs.concordia.ca

***Abstract*—Software requirements specifications are commonly written in natural language, making them prone to a number of defects, such as ambiguity, inconsistency, or lack of readability. Natural Language Processing (NLP) techniques have been proposed as a means to (semi-)automatically improve requirements specifications, but so far have not been widely adopted. We integrated a number of text mining assistants into a wiki-based requirements engineering platform to investigate two key questions: Can software engineers without prior training in NLP effectively leverage these techniques? And are text mining assistants actually helpful in improving the quality of a specification? Results obtained during two software engineering courses demonstrate that both are indeed the case.**

I. Introduction

The success or failure of a collaborative software project is highly dependent on its Requirements Engineering (RE) phase. Industry statistics point to poor Software Requirement Specification (SRS) documents as the root cause in about 50% of unsuccessful projects [12]. Requirements written in natural language account for nearly 90% of all specifications [7]. While easier to understand to all involved stakeholders (compared to semi-formal or formal methods), they are also prone to a number of defects, including ambiguity, inconsistency, omission, and redundancy [12].

While the structure of a specification can be enforced through the use of templates or RE tools, the unstructured nature of natural language prevents similar approaches for their content. Natural Language Processing (NLP) techniques have been proposed as possible ways of (semi-)automatically improving the quality of a specification [5], e.g., by pointing out possible ambiguities to the requirements engineer [4]. However, NLP tools have not yet been widely adopted in practice. This can be attributed to a number of challenges, including *(i)* technical integration (how can text mining tools be introduced into RE tools?), *(ii)* adoption (can software engineers, who are typically not trained in NLP methods, easily use these tools?), and *(iii)* effectiveness (can the NLP tools indeed help to improve the quality of a specification?).

The first challenge has been previously addressed by us, through the integration of NLP into the *ReqWiki* tool, an open source wiki-based requirements engineering platform [9]. In ReqWiki, text mining *assistants* work collaboratively together with the human users, within a single, cohesive interface, to help improve a specification. In order to understand how software engineers would interact with these new text mining assistants during the development of the specification, we performed a case study within two university courses in requirements engineering (one at the undergraduate level, one at the graduate level). In this study, we analysed the usability of NLP methods in the RE process and their impact on the quality of a specification.

In the next section, we first briefly introduce ReqWiki, followed by a description of the NLP services we deployed in Section III. The results of our case study are presented in Section IV, followed by conclusions and future work.

II. The ReqWiki System

Our ReqWiki system[1] [9] is a collaborative, wiki-based platform customized for RE that allows *(i)* capturing of SRS content into several artifact templates, *(ii)* formally representing and reasoning over the populated SRS knowledge in an embedded ontology, *(iii)* applying specialized NLP services to all or parts of artifacts, and *(iv)* generating query-based, revision and domain-specific traceability links.

Wikis are lightweight web applications that need no special client-side tools other than a web browser for project stakeholders to dynamically view and edit content. As an affordable and lightweight documentation and distributed collaboration platform, they have previously demonstrated their capabilities in distributed requirements elicitation [2] and documentation [11].

We also adopted a wiki engine, MediaWiki,[2] at the core of ReqWiki. However, unadorned wikis can not provide the semantic structure and NLP support that we need for SRS analysis. Semantic support is integrated into ReqWiki through Semantic MediaWiki (SMW).[3] This extension allows SRS content to be explicitly enriched with semantic metadata and fetched with in-line queries on pages. To simplify usability, ReqWiki also uses the Semantic Forms[4] extension, allowing users to enter and edit semantic wiki content using web forms, as opposed to working with the raw markup directly.

NLP capabilities have been introduced into ReqWiki through

[1] ReqWiki, http://www.semanticsoftware.info/reqwiki
[2] MediaWiki, https://www.mediawiki.org
[3] Semantic MediaWiki, http://semantic-mediawiki.org/
[4] Semantic Forms Extension, http://www.mediawiki.org/wiki/Extension:Semantic_Forms

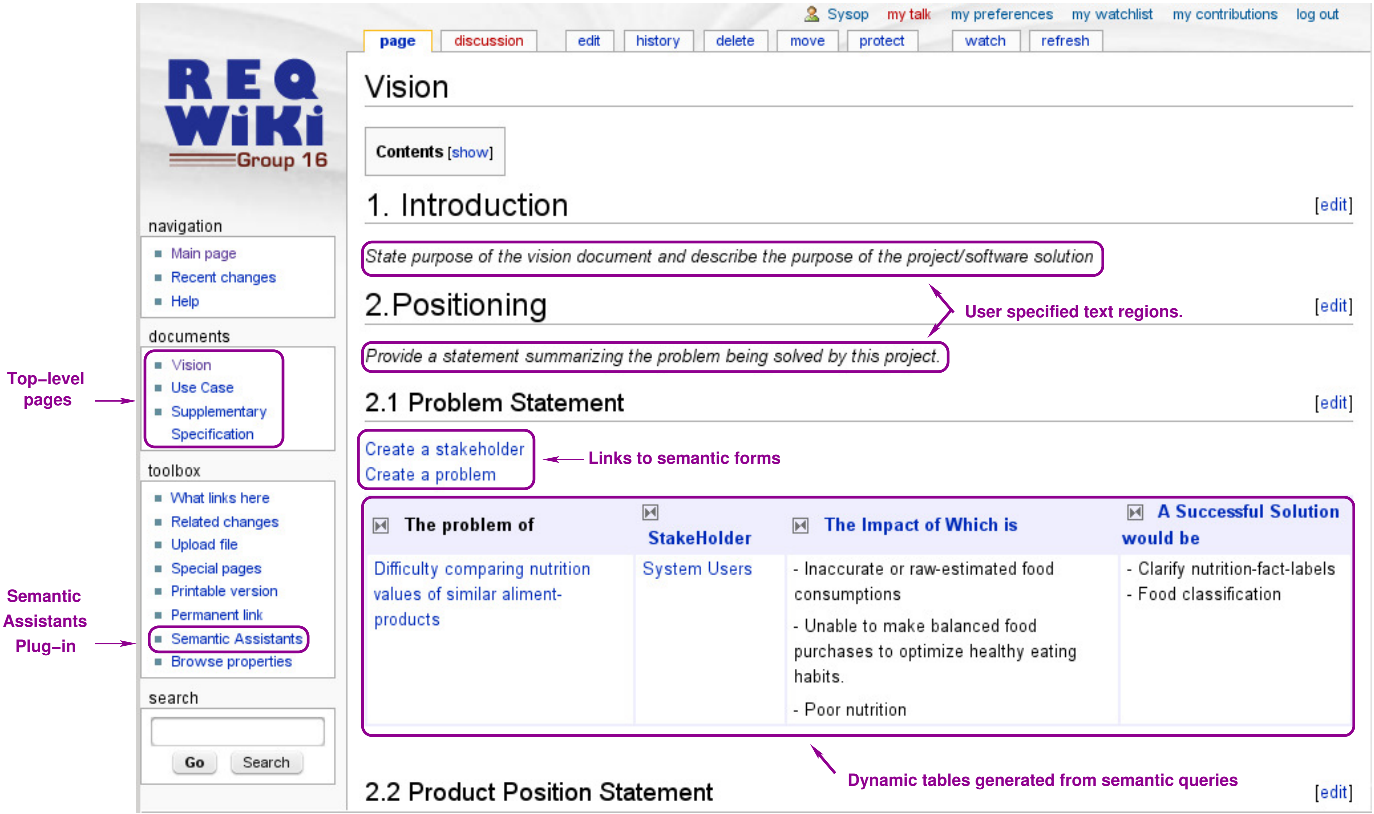


Fig. 1. ReqWiki's web interface with example content and a table automatically generated by an in-line query

our Wiki-NLP integration [10].[5] This integration is based on the Semantic Assistants framework [13], which in turn leverages the General Architecture for Text Engineering (GATE) [1] for deploying NLP pipelines. Using this architecture, sophisticated text mining services can be developed, not only to derive patterns within the unstructured SRS data, but also to enhance information management of a system by finding content based on meaning and context. The NLP services are seamlessly integrated into the wiki environment, where they can be executed on demand by a user or proactively based on events. An example ReqWiki page is shown in Fig. 1, highlighting some of the above mentioned features.

III. Text Mining Assistants for RE

The architecture described above is a service-oriented approach: rather than implementing a fixed number of NLP tasks, new NLP services can be added incrementally and are automatically discovered by the wiki front-end. For this initial evaluation we started with a number of basic NLP services to improve writing in general, as well as requirements specifications in particular. A few additional services were deployed to help users in analysing and structuring larger amounts of textual content. Once added to the architecture, ReqWiki users were able to execute any of these services on their specification.

[5]Semantic Assistants Wiki-NLP integration, http://www.semanticsoftware.info/semantic-assistants-wiki-nlp

Writing Quality Assessment is a service that integrates the *After The Deadline* [8] tool to perform contextual spell, style and advanced grammar checking. It provides explanations and suggestions for some of the detected errors. Of particular interest to RE are the detection of unwanted redundant phrases and passive voice defects. For example, the specification "*transactions must be validated before they are approved*" is written in passive voice and does not specify which parties (stakeholders, the system, or partner applications) are responsible for the action(s).

Readability Assessment measures the clarity of a given text based on standard metrics, like Flesch and Kincaid [3]. This service provides authors with an overall score of their written text, indicating how difficult it is for other stakeholders to read and comprehend. Therefore, specifications with poor scores are likely candidates for refactoring.

Information Extractor is a service performing named entity recognition on wiki pages, based on the ANNIE system [1]. This service can aid users in automatically extracting entities, such as persons, organizations or locations, which are especially useful for non-domain experts to quickly identify key domain concepts in (domain) documents.

Requirement Quality Assurance is a service based on the NASA requirements quality metrics [6]. It detects issues like *Incompletes*, *Options* and *Weak Phrases* within specifications. For instance, consider: "*the system may* *approve customer or supplier requests in a timely manner*," which contains two detectable defects: Here, the word "may" indicates an undesired option that provides the implementer latitude in satisfying the requirement. Namely, is request approval optional or should any one or both request types be supported? Also, the fragment "a timely manner" is an unwanted weak phrase. These are subject to uncertainty with multiple interpretations if the criteria for 'timely' is not quantified elsewhere. Invoking this service allows users to automatically find and correct these often overlooked defects, resulting in less ambiguous and more precise SRS documents.

Document Indexer is a service that creates a back-of-the-book style index of the wiki content as a new wiki page. This service builds on MuNPEx,[6] an open source tool that groups words into noun phrases, to generate an inverted index, with entries automatically linked to wiki pages. Users can compare the result of this service to the domain-specific glossary section of the SRS documents to check its completeness and consistency.

The above NLP services are invoked from within ReqWiki via the Semantic Assistants interface shown in Fig. 2 [10]. There, users can select wiki pages to be analyzed, as well as customize how results should be handled.

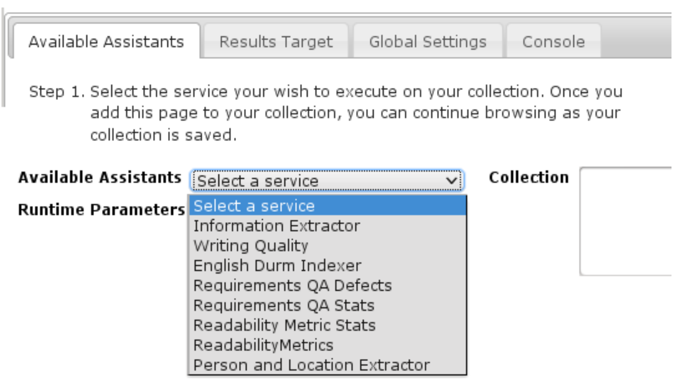


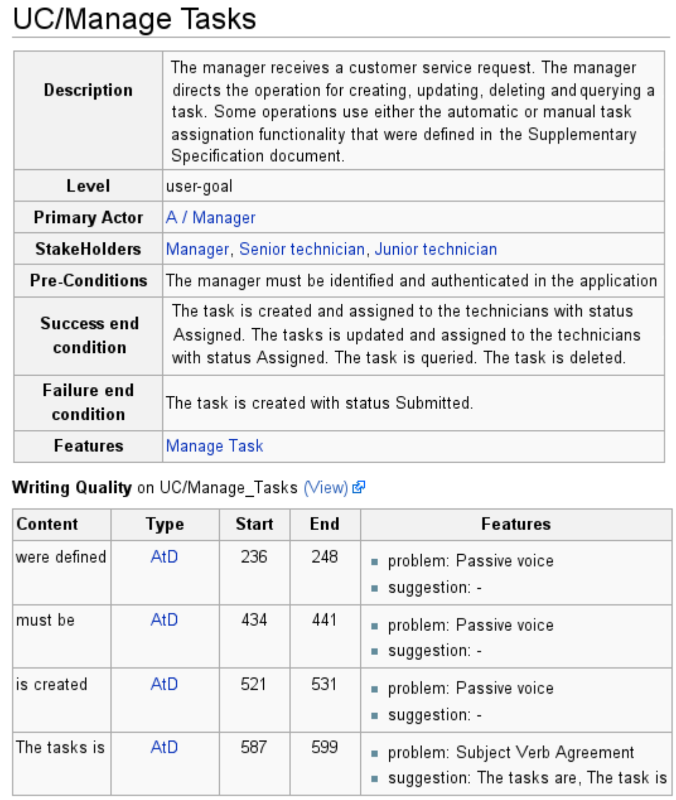
Fig. 2. Interface for executing NLP services in ReqWiki

Fig. 3 shows an example output from the 'Writing Quality' service, run on a wiki page that represents a particular use case, with the output of the NLP analysis automatically added to the wiki page in the form of a table. In ReqWiki, NLP services are currently executed only on demand (interactively), to avoid noise and false positives while the SRS is under development.

IV. Evaluation

We conducted a number of experiments to evaluate the NLP integration for our ReqWiki system along two dimensions, namely the *usability* of the text mining service for non-experts and their *effectiveness* for quality improvements. This study included a total of 22 students from one undergraduate and one graduate level software RE course. For the duration of the course, students were teamed up in pairs to collaboratively scope out an SRS for a medium-sized hypothetical (yet feasible) software project using ReqWiki.

[6]Multi-Lingual Noun Phrase Extractor (MuNPEx), http://www.semanticsoftware.info/munpex



UC/Manage Tasks

Description	The manager receives a customer service request. The manager directs the operation for creating, updating, deleting and querying a task. Some operations use either the automatic or manual task assignation functionality that were defined in the Supplementary Specification document.
Level	user-goal
Primary Actor	A / Manager
StakeHolders	Manager, Senior technician, Junior technician
Pre-Conditions	The manager must be identified and authenticated in the application
Success end condition	The task is created and assigned to the technicians with status Assigned. The tasks is updated and assigned to the technicians with status Assigned. The task is queried. The task is deleted.
Failure end condition	The task is created with status Submitted.
Features	Manage Task

Writing Quality on UC/Manage_Tasks (View)

Content	Type	Start	End	Features
were defined	AtD	236	248	problem: Passive voice suggestion: -
must be	AtD	434	441	problem: Passive voice suggestion: -
is created	AtD	521	531	problem: Passive voice suggestion: -
The tasks is	AtD	587	599	problem: Subject Verb Agreement suggestion: The tasks are, The task is

Fig. 3. Example for NLP analysis results from the 'Writing Quality' service

A. *Usability of NLP Services for Software Engineers*

The main goal of the usability evaluation was to see if users with fair to no background knowledge of NLP can leverage ReqWiki's built-in NLP capabilities. After students had developed the first iteration of their project artifacts, we briefed them on the system's NLP features and the deployed NLP services. The deliverables were: a vision document, followed by use cases and the supplementary specification, and finally requirements test cases.

At the end of the two courses, we provided students with a web-based, anonymous questionnaire containing 18 questions. Participants were explicitly asked, among other questions, about their background knowledge in the field of NLP, classifying them into groups with either *No Background*, *Academic Only* or *Professional* level knowledge in NLP. The questionnaire also asked about the ease-of-use of the NLP interface from a scale of *Very Easy* to *Very Difficult*.

Fig. 4 presents the results gathered from the students' feedback grouped by level of NLP expertise. As can be observed, 72% of students with no previous background knowledge, 40% with mere textbook knowledge, and 50% with professional experience in the field of NLP found the ReqWiki interface and its Semantic Assistants integration *Very*

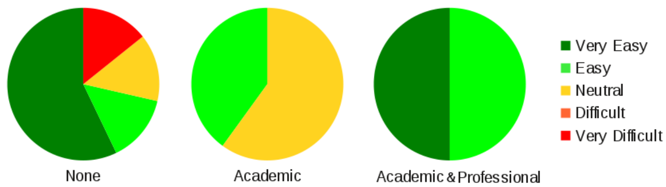


Fig. 4. Usability of ReqWiki based on participant's level of background knowledge in NLP

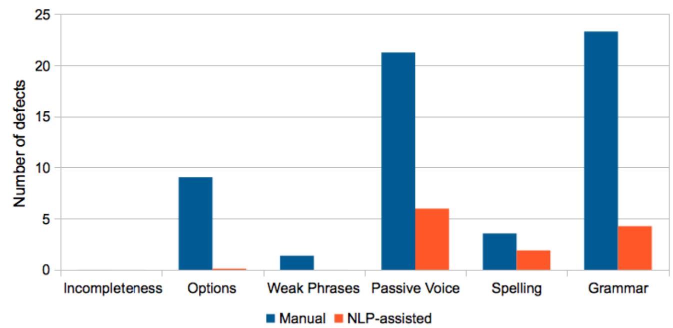


Fig. 5. Average number of outstanding defects remaining in SRS documents without vs. with NLP service support

Easy or *Easy* to use, followed by an overall average of 40% to be *Neutral*. This corroborates our hypothesis that users do not require background knowledge in NLP to make use of sophisticated semantic support.

B. Quality Improvements of SRS

To measure the effectiveness of the NLP services, we compared outstanding defects in revised SRS documents with and without NLP support: Once students completed and documented their SRS within the wiki, they were instructed to manually perform overall writing and requirements quality assessment of their work to correct any lingering flaws including incompleteness, weak phrases and passive voice. Upon submission, markers locally invoked the corresponding NLP services to record the number of defects missed by the students. These same services were then made available for the students to use. Once again, students were given the task of revising their SRS documents, but this time taking into account the automatically generated suggestions of the NLP services (described in Section III) provided by the ReqWiki interface.

Fig. 5 shows the average defect count observed after manual revisions, compared to the number after running the provided NLP services for each of the defect categories (the defect 'Incompleteness' did not appear in the submitted documents). As can be seen, using NLP services for SRS quality assessment purposes significantly reduced the number of remaining issues throughout all defect categories. This demonstrates that the NLP analysis results have significant potential in improving SRS quality, even though some of their suggestions can be false positives. Among these are compound words (i.e., "AdministratorUser") in glossary terms or product names, which may be errors from a linguistic perspective but perfectly acceptable for a SRS document. This explains the existence of some unresolved defects, highlighting the necessity for a collaborative approach between NLP assistants and human engineers.

V. Conclusions and Future Work

Text analysis has long been suggested as an ideal technique to improve the development of SRS documents. To the best of our knowledge, we performed the first evaluation that investigates how software engineers without prior training in NLP can benefit from these techniques. The results are encouraging: provided with a intuitive interface and a suitable human-computer interaction metaphor, namely that of an 'assistant' helping with the specification, the provided NLP services were readily adopted. Moreover, even the basic NLP services we provided in this first iteration had measurable impact on the quality of the developed SRS documents, compared to the version without NLP support. Hence, we now plan to develop additional NLP services that provide semantically richer guidance for SRS documents and also investigate the integration of ReqWiki with other SE tools.

Acknowledgements

We would like to thank the Concordia software engineering students that participated in the evaluation of ReqWiki.

References

[1] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, N. Aswani, I. Roberts, G. Gorrell, A. Funk, A. Roberts, D. Damljanovic, T. Heitz, M. A. Greenwood, H. Saggion, J. Petrak, Y. Li, and W. Peters. *Text Processing with GATE (Version 6)*. University of Sheffield, Department of Computer Science, 2011.

[2] B. Decker, E. Ras, J. Rech, P. Jaubert, and M. Rieth. Wiki-Based stakeholder participation in requirements engineering. *IEEE Software*, 24(2):28–35, 2007.

[3] W. H. DuBay. *The Principles of Readability*. Impact Information, 2004.

[4] I. Hussain, O. Ormandjieva, and L. Kosseim. Automatic Quality Assessment of SRS Text by Means of a Decision-Tree-Based Text Classifier. In *7th International Conference on Quality Software (QSIC 2007)*, pages 209–218. IEEE Computer Society, 2007.

[5] L. Kof. *Text Analysis for Requirements Engineering: Application of Computational Linguistics*. VDM Verlag Dr. Mueller, Saarbruecken, Germany, 2007. ISBN 978-3-8364-4525-2.

[6] P. A. Laplante. *Requirements Engineering for Software and Systems*. Auerbach Publications, 1st edition, 2009.

[7] M. Luisa, F. Mariangela, and N. Pierluigi. Market research for requirements analysis using linguistic tools. *Requirements Engineering*, 9:40–56, 2004. 10.1007/s00766-003-0179-8.

[8] R. Mudge. The Design of a Proofreading Software Service. In *Workshop on Computational Linguistics and Writing: Writing Processes and Authoring Aids, CL& W 2010*, 2010.

[9] B. Sateli, E. Angius, S. S. Rajivelu, and R. Witte. ReqWiki: A Semantic System for Collaborative Software Requirements Engineering. In *Proceedings of the 2012 International Symposium on Wikis (WikiSym 2012)*. ACM, 2012.

[10] B. Sateli and R. Witte. Natural Language Processing for MediaWiki: The Semantic Assistants Approach. In *Proceedings of the 8th International Symposium on Wikis and Open Collaboration (WikiSym '12)*, 2012.

[11] C. Silveira, J. P. Faria, A. Aguiar, and R. Vidal. Wiki-Based Requirements Documentation of Generic Software Products. In *Proc. of the 10th Australian Workshop on Requirements Engineering (AWRE05)*, 2005.

[12] A. van Lamsweerde. *Requirements Engineering*. Wiley, 2009.

[13] R. Witte and T. Gitzinger. Semantic Assistants – User-Centric Natural Language Processing Services for Desktop Clients. In *3rd Asian Semantic Web Conference (ASWC 2008)*, volume 5367 of *LNCS*, pages 360–374. Springer, 2008.