nlrpBENCH: A Benchmark for Natural Language Requirements Processing

Walter F. Tichy, Mathias Landhäußer, Sven J. Körner

IPD Tichy, Faculty of Computer Science
Karlsruhe Institute of Technology
Am Fasanengarten 5
76131 Karlsruhe
{walter.tichy,landhaeusser,sven.koerner}@kit.edu

Abstract: We present nlrpBENCH: a new platform and framework to improve software engineering research as well as teaching with focus on requirements engineering during the software engineering process. It is available on `http://nlrp.ipd.kit.edu`.

Recent advances in natural language processing have made it possible to process textual software requirements automatically, for example checking them for flaws or translating them into software artifacts. This development is particularly fortunate, as the majority of requirements is written in unrestricted natural language. However, many of the tools in in this young area of research have been evaluated only on limited sets of examples, because there is no accepted benchmark that could be used to assess and compare these tools. To improve comparability and thereby accelerate progress, we have begun to assemble nlrpBENCH, a collection of requirements specifications meant both as a challenge for tools and a yardstick for comparison.

We have gathered over 50 requirement texts of varying length and difficulty and organized them in benchmark sets. At present, there are two task types: model extraction (e.g., generating UML models) and text correction (e.g., eliminating ambiguities). Each text is accompanied by the expected result and metrics for scoring results. This paper describes the composition of the benchmark and the sources. Due to the brevity of this paper, we omit example tools comparisons which are also available.

1 Introduction

The majority (79%) of software requirements is written in unrestricted, natural language (NL) [MFI04]. Tools that analyze and transform requirements should therefore be capable of handling NL. Recent advances in natural language processing (NLP) indicate that this is an attainable goal. Among the most striking advances is IBM's Watson [1], which beat two former world champions in the game of Jeopardy! in Feb. 2011. Watson won with a commanding lead, not only because it can process text, but also because it can handle

[1] `http://www.pcworld.com/article/219900/IBM_Watson_Wins_Jeopardy_Humans_Rally_Back.html` and `http://www.forbes.com/sites/bruceupbin/2013/11/14/ibm-opens-up-watson-as-a-web-service/`, accessed 18/12/2014.

context. It has been made available to programmers as a web service in 2014. While Watson answers questions, Google Translate translates texts and web pages among over 60 languages. While not perfect, the results are usable and are improving with time. Jibbigo [2] translates both voice and text among 20+ languages and runs offline on smart phones. Given these feats, progress in processing NL requirements should be attainable.

A useful application of NLP is analyzing requirements for flaws such as ambiguity, imprecision, or incompleteness. Generation tasks, such as extracting models or test scripts from texts, are more demanding, with many open questions. Virtually all researchers, however, demonstrate their systems on their own and usually small examples. Without an accepted benchmark, results are difficult to reproduce and identifying superior approaches is nearly impossible. To improve this situation, we introduce nlrpBENCH, an evolving benchmark for comparing tools for NL requirements processing. Its primary goals are to provide challenges and to make requirements engineering (RE) tools comparable. A hoped-for, secondary effect is to accelerate progress: With the benchmark, it should be easier to determine superior techniques, which can then be adopted and improved by others much faster than presently. The examples in the benchmark can also be used for educational purposes, as they include realistic samples that could be used for study.

In their study on the effectiveness of benchmarks, Sim et al. [SEH03] note that in order to advance research it is important to create a culture of "collaboration, openness, and publicness", and that benchmarks significantly contribute to such a culture. According to Sim, "this kind of public evaluation contrasts sharply with the descriptions of tools and techniques that are currently found in software engineering conference or journal publications".

In 1998, Tichy observed that software researchers needed to conduct more experiments rather than work with small ad-hoc examples. Benchmarks provide a basis for repeatable experiments. Along with realistic inputs, nlrpBENCH also suggests methods for comparing and ranking solutions, making objective evaluation possible. It could become a basis for rigorous empirical research in NLP for RE.

Researcher and practitioners are encouraged to use and extend nlrpBENCH and provide additional metrics for comparison purposes. It might have an accelerating effect on RE, just as benchmarks accelerated research in other fields.

2 nlrpBENCH

2.1 The Structure of nlrpBENCH

nlrpBENCH is a set of tasks, grouped into benchmarks. A task is a NL requirements document and possible solutions. A task is associated with a task type. There are two task types: model extraction and text correction. Additional task types will be added (e.g. test code generation), as the capability of tools expands. Every task has a NL document and

[2] `http://www.jibbigo.com`, accessed 12/18/2014.

an expected result; for every task type there are metrics for determining the quality of a solution (e.g. recall, precision, and F-measure).

2.2 Sources and Approach

The current collection holds over 50 tasks. The expected solutions were constructed by hand and reviewed. The tasks are broken down by categories (e.g. teaching example, industrial specification, standard), by language, and by the availability of solutions. Unfortunately, not all of the tasks have unique solutions.

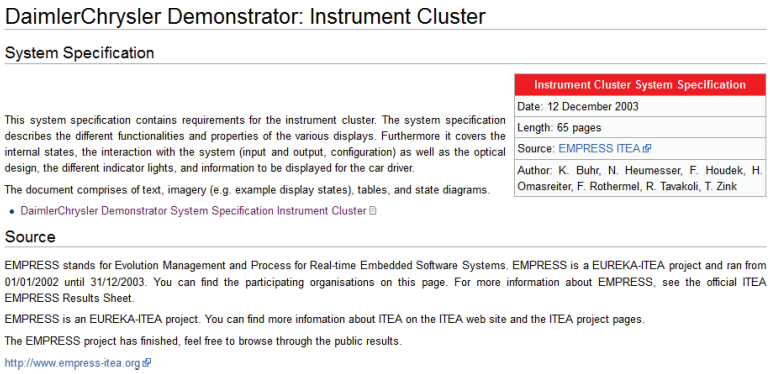
For overall progress, one needs real requirements. At conferences and in personal discussions, researchers often criticize the lack of real-world requirement examples: textbooks contain few examples and they seem to be written by the authors or copied from other textbooks. Many examples about NLP requirements processing use an artificial, strongly restricted language. Also, companies often hesitate to provide samples due to fear of exposing intellectual property or because they think their requirements to be poorly written or inferior in some other way.

As a starting point, we collected previously published examples (and their solutions). Berry et al. published specification texts [KZMB08, BKK03, BBGT08] in order to study flaws. Kof published a solution to Abrial's well-known steam boiler example [Abr96, Kof04a] and to the Daimler Chrysler Demonstrator [BHH+03, Kof04b]. Industry/research cross-breeds like Accenture's RAT [VK08] provide cleaned-up real-world samples. Other tasks have been provided by the research community (Universidad Politécnica de Madrid, Gordon College, and others), companies (Accenture USA, Agilent, BOSCH), or have been taken from textbooks and teaching materials. Text for which we have permission are provided on our website; for others, we provide links.

When we designed the benchmark, we kept Sim et al's [SEH03] desiderata in mind: Accessibility, affordability, clarity, relevance, solvability, portability, and scalability. As our benchmark is fully open and the entire material can be downloaded free of charge, accessibility, and affordability are given. Every task is accompanied with clear instructions and evaluation criteria. The difficulty of the texts varies greatly, so there should be enough material suitable for testing research prototypes as well as industrial-strength tools. The realism of the texts also varies: We included simple textbook examples as well as industrial examples.

2.3 The Tasks

The nlrpBENCH website lists the available tasks in alphabetical order. The website also allows search and browsing through the task categories. Where there are published solutions, these are listed; if there is no gold standard for a given task, the available solutions form a baseline to improve upon. If there are multiple solutions for a task, we provide all of them to provoke discussions of pros and cons.


DaimlerChrysler Demonstrator: Instrument Cluster

System Specification

This system specification contains requirements for the instrument cluster. The system specification describes the different functionalities and properties of the various displays. Furthermore it covers the internal states, the interaction with the system (input and output, configuration) as well as the optical design, the different indicator lights, and information to be displayed for the car driver.

The document comprises of text, imagery (e.g. example display states), tables, and state diagrams.

- DaimlerChrysler Demonstrator System Specification Instrument Cluster

Instrument Cluster System Specification
Date: 12 December 2003
Length: 65 pages
Source: EMPRESS ITEA
Author: K. Buhr, N. Heumesser, F. Houdek, H. Omasreiter, F. Rothermel, R. Tavakoli, T. Zink

Source

EMPRESS stands for Evolution Management and Process for Real-time Embedded Software Systems. EMPRESS is a EUREKA-ITEA project and ran from 01/01/2002 until 31/12/2003. You can find the participating organisations on this page. For more information about EMPRESS, see the official ITEA EMPRESS Results Sheet.

EMPRESS is an EUREKA-ITEA project. You can find more infomation about ITEA on the ITEA web site and the ITEA project pages.

The EMPRESS project has finished, feel free to browse through the public results.

http://www.empress-itea.org

Figure 1: The DaimlerChrysler Demonstrator Specification. A short summary and downloads for detailed information are provided at a glance.

For every task there is a summary page listing the task's properties (such as length, difficulty, and source). Figure 1 shows the DailmerChrysler Demonstrator [BHH+03]. It consists of two documents: the system requirements and the system specification. For both texts there is a short summary and a link to the full document. The source is acknowledged.

2.4 Current Benchmarks

Text to UML Benchmark The goal of the Text to UML benchmark is the extraction of UML class models from text. It is comprised of five short and simple English specifications (two of which are provided in German as well). The lengths of the texts range from 49 to 219 words. The first specification, a public library, has been published several times in SE papers (e.g. in reference [HG03]). The second one is the WHOIS server protocol as described by the IETF RFC 3912. The three remaining texts stem from SE exams and should therefore be consistent, written in a clear language, and straight-forward to model.

Evaluation criteria for UML class diagrams have been proposed in 2003 [HG03]. Harmain and Gaizauskas show how to determine recall, precision, and over-specification of an UML class diagram by mapping the solution to the expected result. Over-specification "measures how much extra correct information in the system response is not found in the" expected solution. The evaluation method is manual at the moment but could be partly automated using model comparison features of the Eclipse Modeling Framework and others.

Text Correction Benchmark The goal of the Text Correction benchmark is the automatic detection and correction of linguistic flaws. The benchmark texts are interspersed with known flaws such as ambiguities, nominalization, and incompleteness. The texts were published in references [KZMB08, BKK03, BBGT08] and were accompanied with

comprehensive lists of flaws. Recall and precision are suitable evaulation criteria. With a common benchmark, different approaches, the benefits, and drawbacks could be easily assessed. It can also be used in case studies and controlled experiments.

3 Related Work

Benchmarks have been used in a variety of areas. The Transaction Processing Performance Council (TPC) [Gra92] published benchmarks for comparing databases. The Standard Performance Evaluation Corporation (SPEC) benchmark evaluates performance of CPUs [Hen00], web servers, mail servers, application servers, etc.

The DARPA Grand Challenge for driverless vehicles (e.g. [DAR04]) can also be seen as a benchmark. The task was for autonomous vehicles to navigate across a stretch of desert. In 2007, this benchmark was extended to driving in urban settings in the DARPA Urban Challenge. This is a good example how benchmarks and competition can speed up progress: In the span of about ten years, this benchmark helped develop autonomous vehicles for real traffic.

About a handful of examples have been used in the RE literature to compare tools; these include a meeting scheduler [FFFL97], an elevator controller [PR99], a steam boiler controller [Abr96, Kof04a], and a public library [Cal94, HG03]. These are popular examples and they are included nlrpBENCH.

4 Conclusion

We present a publicly available collection of requirements specifications. This collection is intended to make tools that process NL requirements comparable. We assembled two benchmarks, one for model extraction and one for text correction. The specifications can be used for evaluations and educational purposes. We invite both professionals and researchers to use, expand, and improve nlrpBENCH. If accepted by the RE community, the benchmarks might lead to public competitions, awards, and prizes.

References

[Abr96] J.-R. Abrial. Steam-boiler control specification problem. In J.-R. Abrial, E. Börger, and H. Langmaack, editors, *Formal Methods for Industrial Applications*, volume 1165 of *LNCS*, pages 500–509. Springer, 1996.

[BBGT08] D. M. Berry, A. Bucchiarone, S. Gnesi, and G. Trentanni. A New Quality Model for Natural Language Requirements Specifications. 2008.

[BHH+03] K. Buhr, N. Heumesser, F. Houdek, H. Omasreiter, F. Rothermel, R. Tavakoli, and T. Zink. DaimlerChrysler Demonstrator: System Requirements Instrument Cluster,

2003. http://www.empress-itea.org/ accessed: 12/18/2014.

[BKK03] D. M. Berry, E. Kamsties, and M. M. Krieger. *From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity - A Handbook*. 2003.

[Cal94] R. E. Callan. *Building object-oriented systems : an introduction from concepts to implementation in C++*. Computational Mechanics Publications, 1994.

[DAR04] DARPA. Grand Challenge '04, 2004. http://archive.darpa.mil/grandchallenge04/ accessed: 06/02/2014.

[FFFL97] M. S. Feather, S. Fickas, A. Finkelstein, and A. van Lamsweerde. Requirements and Specification Exemplars. *Automated Software Eng.*, 4(4):419–438, 1997.

[Gra92] J. Gray. *Benchmark Handbook: For Database and Transaction Processing Systems*. Morgan Kaufmann Publishers Inc., 1992.

[Hen00] J. L. Henning. SPEC CPU2000: measuring CPU performance in the New Millennium. *Computer*, 33(7):28 –35, 2000.

[HG03] H. M. Harmain and R. Gaizauskas. CM-Builder: A Natural Language-Based CASE Tool for Object-Oriented Analysis. *Automated Software Eng.*, 10:157–181, 2003.

[Kof04a] L. Kof. An Application of Natural Language Processing to Requirements Eng. – A Steam Boiler Case Study. Contribution to SEFM 2004, 2004.

[Kof04b] L. Kof. Natural Language Processing for Requirements Eng.: Applicability to Large Requirements Documents. In *Automated Software Eng., Proc. of the Workshops*, 2004.

[KZMB08] N. Kiyavitskaya, N. Zeni, L. Mich, and D. M. Berry. Requirements for tools for ambiguity identification and measurement in natural language requirements specifications. *Requir. Eng.*, 13(3):207–239, 2008.

[MFI04] L. Mich, M. Franch, and P. N. Inverardi. Market research for requirements analysis using linguistic tools. *Requirements Eng.*, 9(1):40–56, 2004.

[PR99] M. Plath and M. Ryan. SFI: a Feature Integration Tool. 1999.

[SEH03] S. E. Sim, S. Easterbrook, and R. C. Holt. Using benchmarking to advance research: a challenge to software engineering. In *Proc. of the 25th Int. Conf. on Software Eng.*, ICSE '03, pages 74–83. IEEE Computer Society, 2003.

[VK08] K. Verma and A. Kass. Requirements Analysis Tool: A Tool for Automatically Analyzing Software Requirements Documents. In A. P. Sheth, S. Staab, M. D., M. Paolucci, D. Maynard, T.W. Finin, and K. Thirunarayan, editors, *Int. Semantic Web Conf.*, volume 5318 of *LNCS*, pages 751–763. Springer, 2008.