Canary: An Interactive and Query-Based Approach to Extract Requirements from Online Forums

Georgi M. Kanchev
Lancaster University
g.kanchev@lancaster.ac.uk

Pradeep K. Murukannaiah
Rochester Institute of Technology
pkmvse@rit.edu

Amit K. Chopra
Lancaster University
amit.chopra@lancaster.ac.uk

Pete Sawyer
Aston University
p.sawyer@aston.ac.uk

***Abstract*—Interactions among stakeholders and engineers is key to Requirements engineering (RE). Increasingly, such interactions take place online, producing large quantities of qualitative (natural language) and quantitative (e.g., votes) data. Although a rich source of requirements-related information, extracting such information from online forums can be nontrivial.**

We propose Canary, a tool-assisted approach, to facilitate systematic extraction of requirements-related information from online forums via high-level queries. Canary (1) adds structure to natural language content on online forums using an annotation schema combining requirements and argumentation ontologies, (2) stores the structured data in a relational database, and (3) compiles high-level queries in Canary syntax to SQL queries that can be run on the relational database.

We demonstrate key steps in Canary workflow, including (1) extracting raw data from online forums, (2) applying annotations to the raw data, and (3) compiling and running interesting Canary queries that leverage the social aspect of the data.

I. Introduction

Online interactions between stakeholders and engineers, e.g., on social media and product discussion forums, contain a variety of requirements-related information. Often, such information is unorganized and too noisy to be readily valuable for RE [3]. However, our observations [2] suggest that online discussions have a naturally emerging structure that can be leveraged to extract requirements-related information. A typical discussion starts with a problem description, which can be about a missing or a poorly-implemented feature of a target application. In response to the problem, other users may propose solutions, express support, or rebut the problem or solutions by pointing out unnecessary or unfeasible aspects. In addition to discussions in natural language, online forums include information such as votes and users' reputation.

We develop Canary, a tool-assisted approach for systematically extracting requirements-related information from online forums. Canary addresses two key challenges. It includes (1) an ontology for annotating online forums with entities of interest to RE, including requirements, solutions, and arguments (supports and rebuttals); and (2) a high-level language in which an engineer can query both technical (e.g., requirements) and social (e.g., most supported solution and least controversial requirement) information from online forums.

We demonstrate Canary's key tools, including those for acquiring discussions and annotations, a relational database for storing annotated discussions, and the Canary query compiler.

II. Design and Implementation

In this section, we describe the key steps in Canary and the technologies used in those steps.

1) Data Extraction: The first step in Canary is to extract information from an online forum. For this demo, we focus on Reddit. We used a custom-built java script using the JRAW Java API wrapper version 0.9.0 [1]. Our script extracts all relevant data and metadata from Reddit and stores them in a MySQL database, preserving the comment-reply tree structure.

2) Annotations: The second step is to annotate the raw data acquired from the online form. In our annotation scheme each comment in a discussion can have one of the four annotations of interest: *requirement*, *solution*, *support*, and *rebuttal*.

In practice annotations can be obtained efficiently and inexpensively via crowdsourcing. For this demo, we employed annotations from two experts (first two authors of this paper). The experts annotated each discussion in three rounds, communicating conflicts after each round, until complete agreement. We store the annotations, too, in a MySQL database.

3) Compiler: Given an annotated discussion, we can write queries in a high-level language to extract requirements-related information. Since the discussions and annotations are in a relational database, we need a compiler to translate queries in the high-level language to SQL.

We implemented a compiler for Canary syntax in Java. We used Eclipse XText (version 2.9) language definition and parsing library. Xtext is a framework that generates a recursive compiler for a given domain language grammar. We programmed the compiler to take queries written in Canary grammar and generate SQL queries. We selected MySQL as a dialect for the compiled queries.

4) Propagation: Forum discussions are naturally nested. Their comment-reply structure creates large trees of interaction. The challenge for Canary is to traverse the whole tree in order to make an assumption about an object of interest. Consider, for example, a requirement followed by two rebuttals; here, the second rebuttal is a rebuttal to a rebuttal and is thus a support to the requirement. Such nesting can be of arbitrary length. Canary's hierarchical queries systematically propagate support and rebuttals, and their corresponding votes.

III. Example Queries

In this section, we describe how users can run queries that follow the Canary grammar. In order to run the code generated

from the high level queries, we use a MySQL server running on MySQL workbench 6.4 CE.

Next, we show some interesting queries. Figure 1 shows an example discussion using real data from social media on which we run Canary queries. Figure 2 shows the query to select all requirements from the discussion.

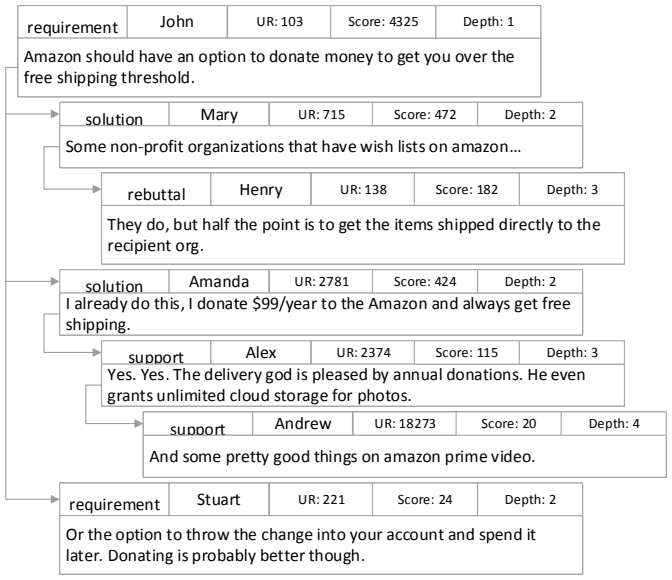

Fig. 1. Example discussion with real data from Reddit.com

```
requirement
```

text	annotation	user	UR	score	depth
donate money	requirement	John	103	4349	1
account credit	requirement	Stuart	221	24	2

Fig. 2. Canary requirement query

Requirements are the only objects of interest that stand alone. A solution must solve a requirement, and support and rebuttals must be toward another object of interest.

Figure 3 shows a query to select solutions for requirements that mention 'donate money'.

```
solution (
  requirement where text regexp 'donate money'
)
```

text	annotation	user	UR	score	depth
non-profit wish list	solution	Mary	715	290	2
Amazon Prime	solution	Amanda	2781	559	2

Fig. 3. Canary solution query and results

Figure 3 shows two interesting query elements.

(1) It shows *conditions*, which can be applied to objects of interest to leverage various metadata associated with them, such as score, natural language text (with support for regular expressions and fuzzy matching), user reputation or role, time of creation, and depth in discussion.
(2) It shows the value propagation brings to queries. In the original discussion '...wish list' has a bigger score than 'Amazon Prime', but with propagated values the score of 'Amazon Prime' increases drastically because of it's two (nested) supporting arguments.

Figure 4, queries for *popular* requirements; this is an example of what we refer to as *aggregator* queries. The idea of popular is to select requirements which caused a large amount of positive interaction. As positive interaction we take score, supporting arguments, derived objects. Canary is able to propagate support through nested positive interaction entities (support of support, support of derived, and so on).

```
popular ( solution ( requirement where regexp
    'save address' ))
```

text	annotation	user	UR	score	pop score
Amazon Prime	solution	Amanda	2781	559	>1

Fig. 4. Canary aggregator query and results

Popularity is measured using ratios. The calculation is done by summing quantitative data about positive interaction and dividing it by the sum of negative interaction.

$$\text{Popularity Score} = \frac{\sum_{i=0}^{n} \text{PV(i)} \times \text{n}}{\sum_{i=0}^{m} \text{NV(i)} \times \text{m}},$$

where n is the number of children with positive semantics and PV(i) the number of votes for a given child i, and m is the number of children with negative semantics and NV(i) the number of votes for a given child i.

IV. Conclusion and Future Work

Systematic understanding of requirements from the crowd can be crucial for development process of many software projects. In order to achieve that we propose Canary, a tool-assisted approach for systematic inclusion of requirements-related information from social media. It is designed to combine ontologies from two different fields of research (RE and argumentation) so that queries can fully leverage all aspects of social interaction found in online forums. In its' current implementation Canary is still heavy on manual work, an interesting future direction the application of machine learning algorithms for further automation.

References

[1] JRAW: The Java Reddit API wrapper. https://github.com/thatJavaNerd/JRAW. Accessed: 2017-03-20.
[2] Georgi Kanchev and Amit Chopra. Social media through the requirements lens: A case study of google maps. *First International Workshop on Crowd-Based Requirements Engineering*, 2015.

[3] Walid Maalej, Maleknaz Nayebi, Timo Johann, and Guenther Ruhe. Toward data-driven requirements engineering. *IEEE Software*, 33(1):48–54, 2016.

APPENDIX

Our demo consists of two main parts: (1) a slideshow presentation and (2) a real time showcase of querying in Canary with viewer participation.

A. Slideshow Presentation

We intend to have the slideshow presentation running throughout the demo. The slideshow includes an overview of extracting raw data and applying annotations (the first two major conceptual steps) in the Canary methodology. These two steps are manually intensive and are not suitable for a real time demo. The slideshow also introduces the viewers to the Canary query language so that they can compose creative queries, later, in the interactive part of the demo.

1) Extracting Raw Data: Section II describes the extraction of raw data using a custom java algorithm. In the demo, we intend to (1) show a brief overview of the algorithm using pseudocode; (2) describe the subtleties of extracting the data such as preserving hierarchical structure and the relevant quantitative metadata; and (3) highlight the interesting aspects of the relational database schema used to store the information.

2) Applying Annotations: We showcase two strategies for applying annotations.

Expert annotation involves two or more participants with a good understanding of requirements engineering principles to process the raw data independently and highlight sections that contain one of the objects of interest (requirement, solution, support, and rebuttal). After completion the two experts synchronize their results and debate any remaining inconsistencies until complete agreement is reached.

Crowdsourcing annotation is more scalable than expert annotation. In a recent experiment, we evaluated the effectiveness of using crowdsourcing (namely Amazon Mechanical Turk workers) in annotating discussions found in social media. We present important results from that study.

3) Query Language Overview: In the last part of the slideshow, we introduce participants to Canary basics, including full syntax, keyword explanation, and examples. Being familiar with the syntax and the power of Canary keywords allows participants to understand the interesting examples we have prepared for the next part of the demo. The purpose of the examples is to promote creativity when participants are allowed to compose and run their own Canary queries by introducing them to the common uses of the language.

B. Interactive Querying with Real Data

Canary creates an abstract, requirements-oriented view over potentially large amounts of real data in natural language. Manually extracting pertinent information from such datasets is non-trivial and error-prone. The most powerful aspect of Canary is its ability to reduce such data using a powerful query language. For the main part of this demo, we will allow the participants to see Canary applied on a fairly large dataset and run their own queries on it in real time. The dataset is a result of careful expert annotation on five discussions extracted from social media. Table I provides a summary of these discussions.

Disc.	Comments	Source	Discussion title
1	141	Reddit	Feature Google Now should add: "Nearest x that is still open"
2	184	Reddit	There is no way to properly save an address in Google Maps
3	79	Reddit	Google Maps should use your average walking speed from Google Fit to calculate walking times
4	261	Reddit	Google maps should have an "I need gas" feature…
5	218	Google Forum	Make 'avoid tolls' option sticky otherwise directions are wrong

TABLE I: Online discussions available to query in the demo

The aim of this demo is to show the usefulness of Canary. We will show the highlights of the language features to the participants and allow them to exercise the features.

The demo can be interesting and engaging to the participants because they will be able to observe the application of the entire Canary methodology and get a first hand feeling of how the query language can leverage large amounts of data using powerful high-level abstractions from two different fields of research. Giving participants freedom of interaction with a novel querying approach will likely spark their creativity to come up with their own interesting queries. Each participant might emphasize on a different strength of Canary and therefore have a different output from running their queries. Given that Canary is a new concept, this demo may expose a limitation of the language as well as open up new avenues for future research.

In Figure 5 we show what we might include in the presentation when explaining how the Canary methodology works. On the top part we have a simple Canary query, shown as is in Xtext. When the compiler is run it generates SQL code that can be executed on our relational database schema. The generated SQL from that example is shown in the same figure below, separated by a blue line. The output code is slightly modified for the purposes of this example. The SQL is in the MySQL dialect and the example is shown being executed in MySQL Workbench. The database used in this example is loaded with the same real data as the one we will be using for actual the demo. The output from this query is also shown. We modified the output query so we can showcase two things. First, The hierarchical structure of the data (each nested reply has added indentation to show it's parents). Second, Canary would usually apply propagation and merge all these rows into one, using the metrics to calculate aggregated popularity. Special attention will be given to the stored procedure *connect_new()*, where all the logic behind the non-trivial task of hierarchical querying in MySQL is. All

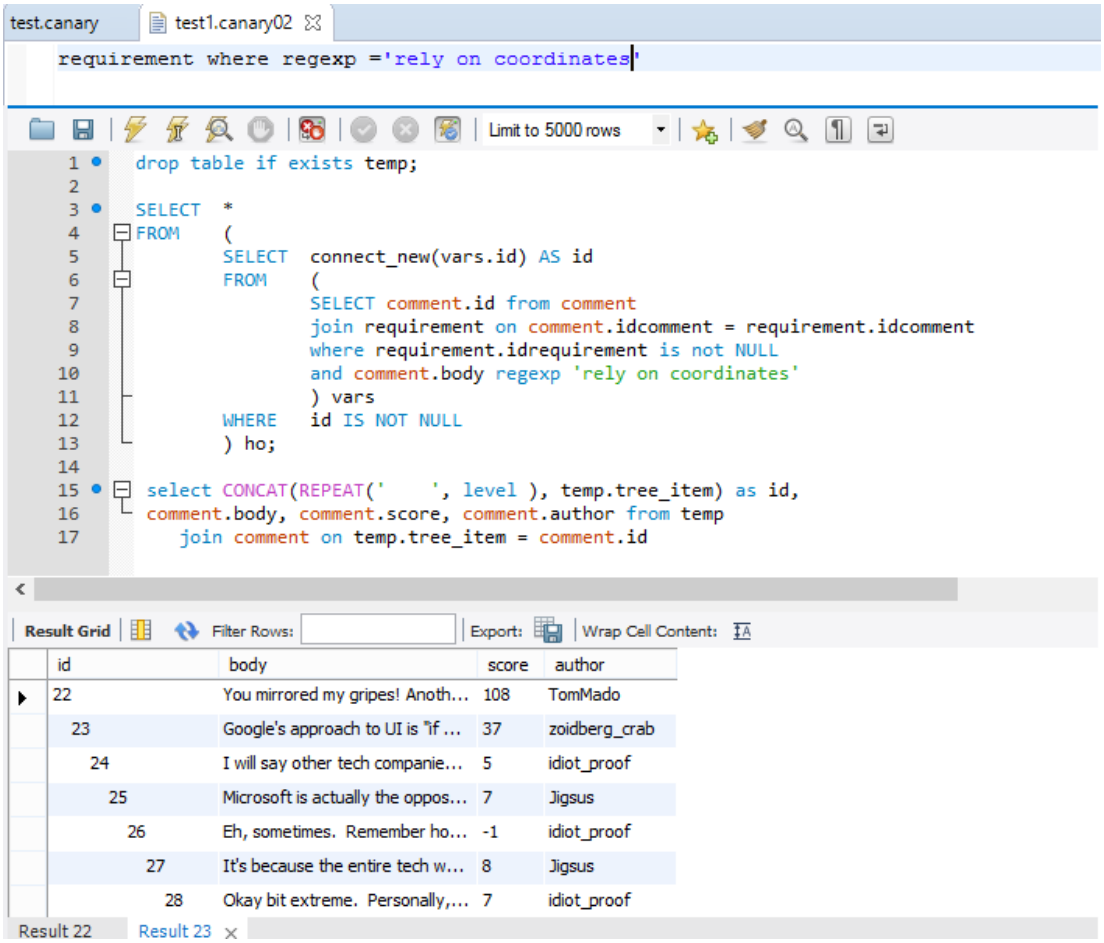

Fig. 5. Screenshot showing separated by a blue line (1) Canary query in Xtext and (2) generated SQL code and output

these things will be explained and shown more clearly and in more detail in the presentation during the demo.