OpenReq

OpenReq Feature List (July 2018)

Currently available features (stable, under evaluation, testing):

Analysis of Twitter feed of a specific product or company.
Customers interact with a company or a product via Twitter for several reasons. We are able to crawl the Twitter feed and automatically identify relevant tweet (reporting problems, asking for new features), as well as irrelevant for requirements engineering-related tasks (uninformative). Using this service, a company can deal with “social media overload” to quickly screen relevant tweets to understand their customers' needs. We currently support English and Italian languages.

Analysis of the reviews for an app in an app Store.
Users of a mobile app write reviews for several reasons. We are able to automatically filter relevant reviews and identify, within a single review, multiple requests for a feature and bug reports. We currently support English language.

Classify requirements.
Software specifications or requirements indicate not only how the system should behave (i.e., functional requirements) but also its quality attributes, such usability or performance (i.e., non-functional requirements). We are able to automatically identify functional and non-functional requirements in natural language text. We currently support English Language.

Automatic identification of users' stances in feedback.
When reviewing a product or service, for example on Amazon, users explain their decisions (e.g., why the decided to purchase a product rather than another). We automatically recognize user stances driving such decisions from the review text. We currently support English language.

Topic clustering of Tweet feed of a specific product or company.
Users interacting with a company or product via Twitter address a vast number of topics. We automatically identify such topics and link them together based on their semantics. A topic could, for instance, be a certain product feature. We currently support English and Italian languages.

Quality check for requirements text.
Requirements written in natural language might be ambiguous and misleading. We automatically identify such ambiguities and suggest text improvements. We currently support English language.

Informativeness of a requirement text.
Requirements written in natural language, especially for large system specifications, may contain additional irrelevant information. We automatically identify non-informative sentences in textual requirements by learning from previous assessments of the stakeholders.

Measuring the text similarity of two different requirements.
Especially for large systems, requirements written in natural language can present duplications or, at least, similarities. We automatically identify similar requirements, based on textual features, in a set of specifications for a product or service.

Prioritisation of requirements.
The prioritization of requirement specifications is usually a time-taking process in which stakeholder need to consider several factors. We automatically suggest priorities for requirements based on configurable metrics—e.g., cost, risks, and benefits.

Preference management of stakeholders for release planning.
When making decisions about a release plan, stakeholders have their personal preferences and bias which can be reflected in the process as well in the release itself. We manage stakeholders' preferences to de-bias their decisions during a release planning task and suggest stakeholder whom can better support the decision-making process.

e-Democracy for release planning.
Some stakeholders (such as end-users of a product or service) are often not included in the decision-making process, for example, regarding the features to be included in a future release. We provide a platform where a crowd interested in a product can vote for the inclusion of new features through a system of delegated voting.

Requirements dependencies, conflicts resolution and release alternatives
Requirements depend on each other. For example, a certain requirement might be a pre-requisite for the implementation of or in conflict to another. We identify several types of dependencies and, once conflicts are detected, suggest possible ways to resolve them. The conflicts can be derived from other partly defined dependencies (e.g., in issue trackers) or by comparing their text of different requirements.
We consider the different preferences of stakeholders as well as the dependencies between requirements to recommend multiple release and prioritization options.

Features under development or planned:

Analysis of context and usage data.
The interactions between users and mobile/web applications, together with the context in which such interaction takes place, offer the possibility to improve the user's experience as well as better understand their needs and derive new features. We will identify such implicit new requirements by sensing the usage patterns of similar groups of users, while giving special attention to privacy.

Interactive visualization of requirements-related data.

Data from several sources and different nature (e.g., social media posts, large unstructured requirements text) needs to be synthesized and visualized to give insights about requirements decisions. We will develop a dashboard which supports the identification of current issues with specific features of the system as well as opportunities for releasing new ones.

Negotiation approaches based on hidden profiles.
During the negotiation phase of a release planning, conflicts among stakeholders can emerge while consensus should be reached. We will detect hidden information relevant for taking a decision, leading to better outcomes of release planning meetings.

Stakeholders identification, scheduling and assignment.
Stakeholders with different expertise are needed during meetings focusing on a requirement engineering task. We will identify participants which are more likely to yield high-quality decisions based on their previous interactions and personal preferences, and schedule such tasks accordingly.