Exploring Techniques for Rationale Extraction from Existing Documents

Benjamin Rogers, James Gung, Yechen Qiao, Janet E. Burge
Department of Computer Science and Software Engineering
Miami University
Oxford, Ohio, USA
rogersbc@muohio.edu, gungjm@muohio.edu, qiaoy@muohio.edu, burgeje@muohio.edu

Abstract—The rationale for a software system captures the designers’ and developers’ intent behind the decisions made during its development. This information has many potential uses but is typically not captured explicitly. This paper describes an initial investigation into the use of text mining and parsing techniques for identifying rationale from existing documents. Initial results indicate that the use of linguistic features results in better precision but significantly lower recall than using text mining.

Keywords-rationale; text mining

I. INTRODUCTION

Designing and developing software consists of a series of decisions along the entire software development lifecycle. These decisions, alternatives considered, and reasons for the choices eventually made (or discarded) form the rationale for the software system. If this information was available to future developers and maintainers it could support many potential uses: reducing the risk of future changes, ensuring that new modifications are consistent with the goals of earlier ones, preventing adoption of previously rejected alternatives, and more. Unfortunately this information is not typically recorded explicitly during development and design. Instead, the rationale is often buried in the many forms of documentation, formal and informal, generated as byproducts of development.

Attempts to encourage explicit rationale capture have been met with resistance from designers and developers. Rationale capture is tedious, intrusive, and potentially expensive. An alternative approach is to see if this information can be mined from existing documentation and structured into a more accessible form. Even if only some of the rationale can be extracted, this will be an improvement over not having access to any rationale at all.

Several questions need to be answered. First, sources of rationale need to be identified, second, techniques need to be developed to extract the rationale from the source documents, and third, the rationale needs to be structured into a usable format. In this paper, we describe some initial steps towards the second of these questions—identifying which text is rationale.

This paper compares two techniques for rationale identification—text mining, using ontologies and the WEKA tool-kit [1], and annotating text for linguistic features, using the General Architecture for Text Engineering (GATE) [2] and the Stanford Parser [3]. For our initial data set, we used bug reports from the Chrome web browser.

Section II of this paper discusses related work, Sections III and IV describe rationale extraction using text mining and annotation using linguistic features, respectively. Section V presents our results for both techniques and Section VI gives our conclusions and plans for future work.

II. RELATED WORK

Rationale research has taken place in several disciplines including software engineering [4][5], human computer interaction [6], and engineering design [7]. Here, we will focus on selected related work in information extraction.

The WEKA tool-kit is very popular and has been applied to many different problems. WEKA was one of several tools used by Port et al. [8] in their investigation into using text mining on unstructured documents. The lessons learned by their investigation, as well as some of the techniques explored, were an influence on the process followed in the experiments described in this paper. Nikora and Balcom [9] compared using different WEKA classification algorithms to discover temporal logic patterns in requirements specifications. We used some of the same pre-processing steps (stopword removal and stemming) in our approach.

Wimalasuriya and Dou [10] designed a process to use multiple ontologies to extract information at the sentence level. They experimented with two kinds of combinations: using ontologies for different sub-domains and using ontologies that give different perspectives of the same problem. They had achieved higher recall on both kinds. With the use of multiple sub-domain ontologies, they achieved an increase in recall (47.97% multiple vs. 41.55% single). With the use of ontologies with multiple perspectives, they achieved an increase in recall using standard definitions (37.10% multiple vs. 29.44% single), and a larger increase in recall when they used definitions extended by WordNet [11][12] (46.77% multiple vs. 33.47% single). Using multiple sub-domain ontologies increased the precision (53.79% multiple vs. 50.86% single) but using multiple perspective ontologies decreased the precision, with and without the use of WordNet (32.39% multiple vs. 36.87% single for standard terms and 40.85 multiple vs. 41.92% single for the extended ontologies). Our approach also uses multiple ontologies to extract information at the sentence level but our ontologies provide domain-specific information and general information.
Zhou et al. studied the usefulness of various linguistic cues towards detecting deception in computer-mediated communication [13]. They clustered twenty-nine linguistic cues into linguistic constructs such as uncertainty (modifiers, modal verbs, uncertainty-denoting words) and nonimmediacy (passive voice, generalizing terms). Mochales and Moens [14] used part-of-speech tagging to identify features for the classification of arguments from unstructured texts. They used linguistic features such as the presence of modal auxiliaries, the presence of adverbs, and the tense of the main verbs in addition to various other non-syntactic features such as n-grams and keywords. They report a 73% F1-Score on a corpus consisting of an equal number of sentences containing arguments and sentences without arguments, with each sentence classified by text type (e.g. newspaper, magazine, parliamentary record). They report an 80% F1-Score on a corpus consisting of legal texts. We are looking for any type of rationale, not just arguments, which are a subset of rationale. Also, bug reports are written less formally than legal texts and thus have a less predictable structure.

Liang, Liu, Kwong, and Lee [15] used text-mining techniques on patent documents to discover design rationale. They model their rationale as three layers: issues, design solutions, and artifacts. Their proposed solution is to build a model in three steps: obtaining artifacts by running PageRank [16] on words appearing frequently, summarizing the issues using a manifold-ranking algorithm and semantic sentence graphs, and finally obtaining issue-solution pairs. Using their methods, they have achieved 51.95% ROUGE-1 F-value on issue summarization, 56.13% ROUGE-1 F-value on solution and issue discovery. Also, using their document profile model with mutual term relation, they had achieved 18.5% F-value for artifact extraction. We plan to eventually classify rationale into different argumentation components but start with investigating techniques to identify rationale in general.

III. RATIONALE DETECTION USING TEXT MINING

A. Training Data

The training data we have used are a collection of Chrome bug reports. We have obtained this collection from data provided for the Mining Software Repositories 2011 mining challenge (http://2011.msrconf.org/msr-challenge.html). We chose a random set of 100 bug reports to use as an initial training set. This allows us to explore more approaches in the exploratory phrase of the research (“Lesson 6: Start with modest objectives and pilot your TM techniques and tools on small data sets before jumping in to a full analysis” [Port]).

After the candidates were picked they were stripped of the HTML header information and converted into plain text files. These files were then manually annotated using GATE to indicate rationale. After the corpus was annotated, we used a sentence splitter to identify sentences. (We discard all the files that only contain special characters.) The XML file produced by GATE was then parsed to extract a separate file for each sentence where the file naming convention indicated if it contained rationale. Some of the bug reports provided data that appeared to be corrupted (duplicate information or information that defied sentence splitting) and was removed from consideration, resulting in a final set of 90. (“Lesson 2: TM requires input of relevant information and is deficient at ignoring irrelevant information” [8].) Our final data set contained 7737 sentences with 711 of them containing rationale.

B. Ontology Definition

The ontologies used were glossaries of security arguments, taken from an on-line glossary of internet security terms [17] and general software development arguments from the SEURAT Argument Ontology [18]. These terms were tokenized and set to lower case with some stopwords removed. After the ontologies were tokenized and combined, we manually marked each ontology word as belonging to either the general rationale ontology, the domain rationale ontology, or both.

The ontology was then expanded by using the synonyms and antonyms of all the senses of the words in both the domain and general ontology using WordNet.

We experimented using the domain only ontology, the general only ontology, and an ontology that was a combination of the two, which produced the best results. We then stemmed the combined ontology and then expanded the combined ontology with WordNet.

C. Data Pre-processing

We initially started with data where the only pre-processing done was to strip out the HTML. We were able to improve our results through the use of stemming and stopwords. The Porter stemmer [19] was run on both the ontologies and the training data. A 429-word stopword list was obtained from the Onix Text Retrieval Toolkit [20]. We stemmed the stopword list and applied the stemmed stopword list to the training data.

D. WEKA Machine Learning Software

WEKA is a machine learning software package that provides tools for both preprocessing and processing of data [1]. In our experiments, 47 different classifiers were used to discover the classifiers that consistently produced the best results. There were 8 Bayesian classifiers (B), 6 tree classifiers (T), 7 rules classifiers (R), 10 function classifiers (F), 2 lazy classifiers (L), 13 meta classifiers including boosting, bagging, and regression classifiers (M), and a Input Mapped Classifier. The three unstemmed ontologies were run on all 47 different classifiers.

The top 20% scoring classifiers were then used on the stemmed ontology. They were Bayes Net, IB1, IBk, JR48, JR48 graft, JRip, Logistic, NNge, PART, Random Committee, Random Forest, and Random Tree.

IV. RATIONALE DETECTION USING LINGUISTIC FEATURES

Intuitively, rationale-containing sentences share certain linguistic features. Identifying these features may provide a
useful aid in rationale detection for general domains. First, we compared a list of promising candidate features. We then compared the contributions of these features towards binary classification of sentences for detecting rationale.

A. Candidate Linguistic Features

We identified promising candidate features for detecting rationale on the basis of apparent relation to rationale and ease of identification through parsing. Modal auxiliaries impart functional information upon the main verb they modify, indicating degrees of possibility. Certain adverbial clauses also may be indicative of arguments or alternatives. Projective clauses set up the main clause as a belief or opinion, and are thus useful in indicating rationale. Table I shows the features used in the initial experiments described in this paper.

B. Rationale Identification Using GATE

GATE (General Architecture for Text Engineering) is a Java-based framework for carrying out a variety of NLP-based tasks [2]. Using a combination of the POS tags and dependency relations identified by the Stanford Parser [3] and GATE’s JAPE (Java Annotations Patterns Engine) grammar functionality, we automatically annotated each of the linguistic features in the corpus.

V. RESULTS

A. Rationale Extraction Using Text Mining

After running our data through the classifiers using the domain ontology, the general ontology, and the combined domain and general ontology, we ranked the ontologies for each classifier after 10-fold cross validation. There were 8 classifiers where all three ontologies tied, as they all had an F1-value of 0, and were omitted from the results. The classifiers using both the general and domain ontologies performed markedly better than when using either ontology independently, while the general ontology performed better than the domain ontology by itself.

We then took the top 20% scoring classifiers and used them to classify our stemmed data using a stemmed ontology consisting of both the general and domain ontologies. The results were then compared to the results of the unstemmed runs with both general and domain ontologies. We compared precision (p), the number of returned sentences containing rationale over the number of sentences containing rationale in the data set, and F1-value (f), the harmonic mean of the precision and recall, and ranked them (R) on stemmed F1-value. Subscripts u and s denote unstemmed and stemmed respectively. The gaps in rank are due to classifiers excluded because of their similarity to others: Random Committee (3), due to its similarity to Random Forest and J48 (8) due to its similarity to J48. Table II shows the results.

<table>
<thead>
<tr>
<th>R</th>
<th>Name</th>
<th>p_u</th>
<th>r_u</th>
<th>f_u</th>
<th>p_s</th>
<th>r_s</th>
<th>f_s</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Part (R)</td>
<td>60.1</td>
<td>80.1</td>
<td>74.4</td>
<td>64.5</td>
<td>55.6</td>
<td>59.7</td>
</tr>
<tr>
<td>2</td>
<td>RandomForest (T)</td>
<td>63.8</td>
<td>62.6</td>
<td>61.8</td>
<td>58.5</td>
<td>50.2</td>
<td>59.2</td>
</tr>
<tr>
<td>3</td>
<td>BayesNet (B)</td>
<td>52.7</td>
<td>66.5</td>
<td>58.8</td>
<td>50.9</td>
<td>66.8</td>
<td>57.8</td>
</tr>
<tr>
<td>4</td>
<td>J48 (T)</td>
<td>66.6</td>
<td>63.2</td>
<td>62.6</td>
<td>64.4</td>
<td>50.8</td>
<td>57.3</td>
</tr>
<tr>
<td>5</td>
<td>SGD (F)</td>
<td>62.5</td>
<td>89.4</td>
<td>72.1</td>
<td>63.7</td>
<td>51.6</td>
<td>57.0</td>
</tr>
<tr>
<td>6</td>
<td>NNge (R)</td>
<td>58.3</td>
<td>55.1</td>
<td>60.2</td>
<td>56.5</td>
<td>57.0</td>
<td>56.9</td>
</tr>
<tr>
<td>7</td>
<td>RandomTree (T)</td>
<td>57.4</td>
<td>49.4</td>
<td>53.1</td>
<td>60.2</td>
<td>50.2</td>
<td>54.8</td>
</tr>
<tr>
<td>8</td>
<td>MultiClassClassifier (M)</td>
<td>61.5</td>
<td>43.2</td>
<td>50.7</td>
<td>66.6</td>
<td>43.5</td>
<td>52.6</td>
</tr>
<tr>
<td>9</td>
<td>J48 (R)</td>
<td>62.6</td>
<td>38.7</td>
<td>47.8</td>
<td>63.9</td>
<td>38.1</td>
<td>47.8</td>
</tr>
</tbody>
</table>

Precision improved in most cases when stemming was used while recall fell.

We then took the unstemmed combination general and domain ontology used in Table II and created an expanded ontology including synonyms and antonyms generated by WordNet. We then stemmed the resulting ontology and ran it on the nine machine learning models listed in the previous table. The following table compares the original stemmed ontology results to the results of the expanded stemmed ontology. We compared precision (p), recall (r), and F1-value (f) and ranked them (R) on the expanded ontology result F1-value. Subscripts o and e denote original and expanded respectively. Table III shows the results.

<table>
<thead>
<tr>
<th>R</th>
<th>Name</th>
<th>p_o</th>
<th>r_o</th>
<th>f_o</th>
<th>p_e</th>
<th>r_e</th>
<th>f_e</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Part (R)</td>
<td>60.1</td>
<td>80.1</td>
<td>74.4</td>
<td>64.5</td>
<td>55.6</td>
<td>59.7</td>
</tr>
<tr>
<td>2</td>
<td>RandomForest (T)</td>
<td>63.8</td>
<td>62.6</td>
<td>61.8</td>
<td>58.5</td>
<td>50.2</td>
<td>59.2</td>
</tr>
<tr>
<td>3</td>
<td>BayesNet (B)</td>
<td>52.7</td>
<td>66.5</td>
<td>58.8</td>
<td>50.9</td>
<td>66.8</td>
<td>57.8</td>
</tr>
<tr>
<td>4</td>
<td>J48 (T)</td>
<td>66.6</td>
<td>63.2</td>
<td>62.6</td>
<td>64.4</td>
<td>50.8</td>
<td>57.3</td>
</tr>
<tr>
<td>5</td>
<td>SGD (F)</td>
<td>62.5</td>
<td>89.4</td>
<td>72.1</td>
<td>63.7</td>
<td>51.6</td>
<td>57.0</td>
</tr>
<tr>
<td>6</td>
<td>NNge (R)</td>
<td>58.3</td>
<td>55.1</td>
<td>60.2</td>
<td>56.5</td>
<td>57.0</td>
<td>56.9</td>
</tr>
<tr>
<td>7</td>
<td>RandomTree (T)</td>
<td>57.4</td>
<td>49.4</td>
<td>53.1</td>
<td>60.2</td>
<td>50.2</td>
<td>54.8</td>
</tr>
<tr>
<td>8</td>
<td>MultiClassClassifier (M)</td>
<td>61.5</td>
<td>43.2</td>
<td>50.7</td>
<td>66.6</td>
<td>43.5</td>
<td>52.6</td>
</tr>
<tr>
<td>9</td>
<td>J48 (R)</td>
<td>62.6</td>
<td>38.7</td>
<td>47.8</td>
<td>63.9</td>
<td>38.1</td>
<td>47.8</td>
</tr>
</tbody>
</table>

Adding synonyms and antonyms gave good results, indicated by a general increase of precision, recall and F1-value.

B. Rationale Extraction Using Linguistic Features

To evaluate the contributions of the linguistic features, we applied GATE’s built in machine learning tool, the Batch Learning Processing Resource (PR) [21]. We used LibSVM, which is integrated with the Batch Learning PR for increased efficiency [22]. We use an uneven tau value (0.4) because the corpus contains many more negative examples (sentences not containing rationale) than positive examples (sentences
TABLE IV. RESULTS USING LINGUISTIC FEATURES

<table>
<thead>
<tr>
<th>Feature Set</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modal Auxiliaries</td>
<td>61.9</td>
<td>20.8</td>
<td>30.3</td>
</tr>
<tr>
<td>Adverbiaal Clauses</td>
<td>67.3</td>
<td>10.6</td>
<td>17.8</td>
</tr>
<tr>
<td>Projective Clauses</td>
<td>10.0</td>
<td>0.07</td>
<td>0.1</td>
</tr>
<tr>
<td>Combined</td>
<td>60.5</td>
<td>24.0</td>
<td>33.6</td>
</tr>
</tbody>
</table>

Using only a handful of linguistic features, we have achieved precision comparable to our ontology-based approaches. Projective clauses were of negligible value for identifying rationale in our corpus. The presence of modal auxiliaries is the most useful/prevalent linguistic feature for identifying rationale followed closely by adverbiaal clauses.

VI. CONCLUSIONS AND FUTURE WORK

While we were able to improve our classification performance by pre-processing the data, we still need to improve our accuracy. The linguistic feature results are promising but while the precision is better than the text mining results, the recall is significantly lower. We have only experimented with three linguistic features and have six additional ones we will include in future experiments. Our results appear to indicate that recall increases with the inclusion of additional features.

We also plan to develop a larger training and test set now that we have refined our techniques on the smaller set. This will enable us to progress from identifying if a sentence contains rationale to identifying what type of rationale (decision problem vs. alternative vs. argument) is present. We also will be expanding our ontologies to incorporate a wider selection of domain terminology.

One of the challenges in our work is dealing with data with little or no structure. We have not yet been taking advantage of the meta-data associated with the bug reports (violating “Lesson 4: Metadata is vital but often neglected and hard to get” [8]). Further research needs to be done to determine if it is possible to create generic rationale extraction tools or if customization to specific data sets is required. We suspect there will be tradeoffs between a broader applicability of the techniques and their accuracy.

ACKNOWLEDGMENT

This work was supported by NSF CAREER Award CCF-0844638 (Burge). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation (NSF). We would also like to acknowledge the contributions of Kendra Schroeder in preparing the training data and Lin Li in compiling the ontologies.

REFERENCES