Content-Based Recommendation Techniques for Requirements Engineering

Gerald Ninaus
Institute for Software Technology
Graz University of Technology
Inffeldgasse 16b/2
A-8010 Graz, Austria
gninaus@ist.tugraz.at

Florian Reinfrank
Institute for Software Technology
Graz University of Technology
Inffeldgasse 16b/2
A-8010 Graz, Austria
freinfra@ist.tugraz.at

Martin Stettinger
Institute for Software Technology
Graz University of Technology
Inffeldgasse 16b/2
A-8010 Graz, Austria
mstettinger@ist.tugraz.at

Alexander Felfernig
Institute for Software Technology
Graz University of Technology
Inffeldgasse 16b/2
A-8010 Graz, Austria
afelfernig@ist.tugraz.at

Abstract—Assuring quality in software development processes is often a complex task. In many cases there are numerous needs which cannot be fulfilled with the limited resources given. Consequently it is crucial to identify the set of necessary requirements for a software project which needs to be complete and conflict-free. Additionally, the evolution of single requirements (artifacts) plays an important role because the quality of these artifacts has an impact on the overall quality of the project.

To support stakeholders in mastering these tasks there is an increasing interest in AI techniques. In this paper we present two content-based recommendation approaches that support the Requirements Engineering (RE) process. First, we propose a Keyword Recommender to increase requirements reuse. Second, we define a thesaurus enhanced Dependency Recommender to help stakeholders finding complete and conflict-free requirements. Finally, we present studies conducted at the Graz University of Technology to evaluate the applicability of the proposed recommendation technologies.

I. INTRODUCTION

Ensuring quality in a software project is a complex task. On the one hand there are limited resources in software projects. On the other hand there is a huge set of requirements which should be satisfied. Consequently, it must be guaranteed not to waste valuable resources on unnecessary tasks and to implement the most important artifacts first. A software development process can be divided into requirements engineering (RE), architectural and detailed design, implementation, and testing.

The scope of RE tasks is fairly broad as it starts with an unlimited solution space and is used to set borders for the following software development. It also needs to transform high-level objectives to operational prescriptions [1][2].

According to a Gartner report [3], corrections of defects (e.g. conflicting or missing requirements, wrong or incomplete artifact descriptions) are inexpensive during the RE phase but they are very expensive after delivery which makes decisions taken during the RE phase critical for the success of software projects [4]. Undetected errors in the RE phase is a major source of problems in subsequent development phases. These errors trigger not only costs for correcting the offending error, but also generate expenses in related artifacts to this error (e.g. redesign of code, documentation rewrite, and costs of the replacement of software already deployed) [5][6]. Consequently, it is necessary to establish actions which guarantee requirements with high quality, address the stakeholders needs, and have no inconsistencies or errors [7][8].

In most cases, to derive a complete definition of all needed objectives in a software development project, it is necessary to include a variety of different and inhomogeneous stakeholders in the RE process [9]. Within this group of participants it cannot be assumed that everybody has the expertise to write specifications in a formal representation. Thus, it is more attractive to evolve, maintain and discuss requirements in natural language with possibly non-technical customers as this is the only language which can be confidently assumed to be shared among all involved stakeholders in a software development process [1][10][11][12].

Requirements analysts start with ill-defined and in many cases conflicting ideas of what the proposed system is expected to do, and must progress towards a single, detailed, technical specification of the system [2]. Within this process analysts are confronted with non-functional concerns such as safety, security, usability, performance, and so forth which are often conflicting with functional requirements [1]. This fact and the circumstance that requirements are often presented without an explicitly specified structure complicates the RE process [13]. Unfortunately, creating a structure in RE is a labor intensive task as software projects consist of a huge amount of requirements. A complicating factor is that the process of defining a structure involves understanding of the needs of users, customers, and other stakeholders, and also of the context in which the to-be-developed software will be used [2]. Without the existence of formal descriptions, requirements validation usually describes a subjective evaluation of informal or undocumented requirements which requires stakeholders involvement [2].

As mentioned before, a subjective evaluation without any intelligent support is a labor intensive and error prone task. Consequently, facilitating the task by preprocessing the provided data is promising. Previous empirical research explained that the identification of user requirements and the improve-
The contributions of this paper are two-fold: first, we propose two different recommendation techniques to support stakeholders during the elicitation phase of the RE process. Second, we conducted two studies at the Graz University of Technology to evaluate the applicability of these techniques in the RE context.

The remainder of this paper is organized as follows. In Section 2 we summarize work related to the techniques used in this paper. In Section 3 we present an overview of IntelliReq which is our web platform to develop and evaluate recommendation technologies. Section 4 describes the two used content-based recommender implemented for our evaluation. In Section 5 we present the findings of the conducted studies and in Section 6 we conclude this work and outline future research directions.

II. RELATED WORK

This Section gives an overview about definitions and approaches related to text evaluation and similarity calculation.

A. Text Document Representation

A commonly used technique is the so called bag-of-words representation. Within this approach the information about paragraphs, sentences, and word orders are removed to make the information more useful for machine learning algorithms [19]. In this bag all non-descriptive words like and or has are defined as stop words and have to be removed. The remaining words are stemmed (reducing inflected or derived words to their stem) and their occurrence is stored in a vector [20]. For example, if the word house and the word houses can be found in a text the stemmed version of both terms is hous and the resulting occurrence is two.

B. Word Sense

To find relations between requirements it is necessary to calculate the similarity of all words contained in the textual description of the involved requirements. It is stated that not the word form, but rather the Word Sense is the relevant participant to define a possible relation between words [21]. In WordNet, for example, these Words Senses are defined as synsets (short for synonymy sets) and can be interpreted as the ambiguity of the word [22][23]. For example, the term cold has a different meaning in the sentence a person is cold as in the sentence a room is cold. To handle this ambiguity synsets can be used instead of terms within the bags-of-word representation. This leads to two benefits: first, the terms are fully disambiguated as the context has been taken into account and should increase precision. Second, equivalent terms can easily be identified as they all reside in the same synset which increases recall [24].

However, the assignment of the correct Word Sense to a term is a challenging task. It is necessary to decide whether to use a simple rule like taking the most frequent used Word Sense found in the used language or to analyze the complete context in which the word under investigation occurs. The second approach is clearly more complex as it needs to calculate the...
probability for all possible word senses of all terms used in a
document [20].
Next, the representation of the Word Sense inside the bag-
of-words has to be chosen. There are basically three concepts
[20]:
- **Add Word Sense**: For each term add a Word Sense. This
  results in an occurrence of at least two, as the original
  term is not replaced.
- **Replace terms by Word Sense**: By replacing the original
  term the minimal occurrence can be one.
- **Word senses only**: The bag-of-words only contains an
  entry for terms where a Word Sense can be found. The
  Word Sense is used to replace the original term. The
  cardinality of this representation is smallest of the three
  presented options.
Alternatively to using already defined Word Senses like
synsets one can discover Word Senses by clustering. With this
approach similar Word Senses are derived from the context
in which they are used. The assumption is that the meaning
of an unknown word can often be inferred from its context.
This approach is meant to cope with the problem that standard
dictionaries miss domain-specific senses of words [25]. On
the other hand, learning-based approaches are very domain-
specific which means that the quality of the classifier drops
precipitously when the same classifier is used in a different
domain [26].

C. Domain Knowledge

Domain knowledge is one crucial factor for high quality
requirements elicitation [27]. A domain thesaurus can be used
to formalize this knowledge and to inheritate a classification
of terms for further processing [17]. Initializing a domain
thesaurus is labor intensive in the factors cost and maintenance
but also contain high-value knowledge. To reduce the initial
effort of creating a domain thesaurus, Wikipedia can be used as
a source of manually defined terms and relationships [28].

III. IntelliReq

Having the need for computer aided software engineering
(see Section I) we decided to develop IntelliReq\(^1\) which is
a web platform for early requirements engineering. Besides
the content-based recommendations discussed in this paper,
IntelliReq also supports group-based recommendations and
stakeholder guidance techniques to improve the quality of the
RE process [29]. Figure 1 shows the latest GUI version of
IntelliReq. We use IntelliReq to evaluate the applicability of
Artificial Intelligence (AI) techniques for requirements engi-
neering. IntelliReq also supports geographically independent
collaborative work which is often necessary when project
stakeholders can not participate in meetings [30]. A main non-
functional requirement for IntelliReq itself was the computa-
tional simplicity of all our recommender algorithms because
our system is designed as an online multi-agent platform with
fast response time.

\(^1\)http://www.intellireq.org

Research has been done in clustering requirements to find
dominant themes (topics) to support the assignment of potential
interested stakeholders to this themes [30]. In our work
we focus on the calculation of tensions between requirements,
because we need a ranking of the k-top related requirements
to a requirement under investigation. Although clustering has
not been focused yet, the calculated tensions produced within
IntelliReq can be used as input for a subsequent clustering.
In this paper we discuss the content-based recommendation
support evaluated with IntelliReq, which can mainly be divided
into the following two techniques:
- **Keyword Recommender**: IntelliReq recommends keywords for a new inserted requirement
- **Dependency Recommender**: IntelliReq recommends requirement pairs as dependency candidates based on the
similarity between requirement descriptions

The reason for these two recommender implementations were
two-fold: first, we want to increase the reuse of artifacts which can, according to literature, increase the software quality [8].
This should be done with the Keyword Recommender. Second,
we want to facilitate the management of requirements in the
dimensions completeness, redundancy, and consistency. The
scope of this recommendation is no automatic generation of
dependencies but to recommend requirement pairs with similar
descriptions to the users for further investigation.
We discuss these two recommender implementations in
more detail in Section IV and the results of studies conducted
at Graz University of Technology in Section V.

A. Lexical Semantic Resources

To enhance the calculation of similarity with semantic in-
formation it is necessary to select a lexical semantic resource.
We therefore discuss three different available resources and
motivate our decision for our selection. Large lexical semantic
resources can be categorized into expert-built lexical semantic
resources (ELSR) like GermaNet\(^2\) and collaboratively con-

\(^2\)http://www.sfs.uni-tuebingen.de/GermaNet/
structured lexical semantic resources (CLSR) like Wiktionary⁴. OpenThesaurus⁴ can be located between these two definitions as it is collaboratively constructed, but reviewed and maintained by an administrator who revises all changes made in the database [23][31].

All resources support Hyponymy which declares relationships between words as Hyponyms and Hypernyms. A Hypernym can be characterized as a type-of relationship while a Hyponym is a topic of a set of other terms. For example, we can denote the word Measurement Device as a Hypernym while the term chronometer is a Hyponym to Measurement Device.

OpenThesaurus follows the idea that users should be able to freely contribute to the project. The access to the stored data is available through their web portal, where users can search for synonyms. There is also an API for a web service and the data can be downloaded as a MySQL database dump file. The main focus of OpenThesaurus is to provide synonyms for words. This is based on two reasons: first, the project should be kept simple and second, the most prominent application is OpenOffice which has no strong demand for other relations than synonyms [31].

Comparing these different lexical resources one can say that GermaNet contains the largest amount of taxonomic relations, OpenThesaurus provides the highest number of synonymy relations, and Wiktionary contains the most antonyms and the second most synonyms and hypernymy relations [23].

While OpenThesaurus hardly supports Hyponymy it massively outperforms the other resources in the dimension Synonym relations (see Table I). For that reason we decided to use OpenThesaurus. In OpenThesaurus Word Senses are grouped into Synsets (see Section II). If we consider the example term cold then we get the Synsets with the numbers: 1110, 3834, 3945, 10632, 29416. Table II shows the corresponding terms for the Synsets 1110 and 3834.

<table>
<thead>
<tr>
<th>Lexical Resource</th>
<th>Number of Synonym Relations</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenThesaurus</td>
<td>288,121</td>
</tr>
<tr>
<td>GermaNet</td>
<td>69,097</td>
</tr>
<tr>
<td>Wiktionary</td>
<td>62,235</td>
</tr>
</tbody>
</table>

TABLE II
SYNSETS FOR THE TERM cold

<table>
<thead>
<tr>
<th>Synset ID</th>
<th>Terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>1110</td>
<td>cold, insensible, cruel, icy, cold-hearted, hard-hearted</td>
</tr>
<tr>
<td>3834</td>
<td>cold, fresh, cool, frosty</td>
</tr>
</tbody>
</table>

³http://de.wiktionary.org/
⁴http://www.openthesaurus.de/

B. Language Versions

We evaluated the Keyword Recommender within an empirical study conducted during the course Object-oriented Analysis and Design at Graz University of Technology. As not all course members had German language skills we used English as description language for requirements. On the opposite the second empirical study was conducted with German native speakers as we did not want to risk a bias based on lack of language skills when participants should evaluate the natural language processing capacity of IntelliReq. We therefore developed an English version for the evaluation of the Keyword Recommender and a German version for the evaluation of the Dependency Recommender.

IV. RECOMMENDATION

A. Keyword Recommender

The Keyword Recommender is designed to support the English language and uses a simple generation for the bag-of-words. First, all stop words are removed from the text and the remaining words are transferred to a lower case representation which is stemmed using the Porter Stemming Algorithm⁵. The stemmed version and the original version are stored into a lookup table.

To facilitate the reuse of requirements IntelliReq provides a filtered by keyword functionality. With this functionality users can select a keyword from a list to set the filter. Our keyword recommender stores all keywords used to annotate the requirements and the stemmed version of the keywords. Each time a new requirement is inserted, all words of the requirements subject and description are stemmed and compared to the internal list. If an already stored keyword is equal to one word of the new provided data, the recommender returns the originally used keyword for annotation. With this approach we want to minimize the cardinality of words used to annotate the requirements in the database. For example, take the term database. In the group without the keyword recommendation users use the keywords database and databases. Supported by the Keyword Recommender users get the keyword database as replacement for databases proposed.

We also used a knowledge thesaurus to enhance the quality of the recommendation. This thesaurus was manually created to support the definition of software applications in the domain of recommender technologies for tourism. We therefore defined word synonyms for a controlled subset of terms and declared only one word sense for each term. For example, we defined the words: tourist, client, customer, holidaymaker, vacationer as synonym and these words did not occur in any other synonym list. Using this domain-specific recommender it is possible to increase the efficiency of the Keyword Recommender like it is proposed in literature [28].

B. Dependency Recommender

For our Dependency Recommender we start the generation of the bag-of-word similar to the implementation of the

⁵http://snowball.tartarus.org/algorithms/porter/stemmer.html
Keyword Recommender. First, the text is stripped of all stop words and transferred to the lower case representation. Next, the tokens need to be converted in a comparable form. Instead of using a stemmer, the Dependency Recommender uses a mapping table to get the base forms of the terms. For this purpose we take the Morphy\(^8\) data file which consists of a list of German terms with all inflected forms and grammatical properties and generated a SQL table with the data. With this table a query for the German plural Häuser (houses) will result in the singular ‘Haus’ (house). A normal stemmer would have problems to get both versions in a comparable version as it only truncates the word and do not replaces the character ä with the character a. The Morphy data set contains 368,175 associations between inflected and base forms. Furthermore, these base forms facilitate the lookup in the OpenThesaurus database because the original dataset contains no stemmed forms. Next, we store the data inside the bag-of-words in the database because the original dataset contains no stemmed forms. The calculation of the similarity will result in the highest possible tension with the value of 2 as the Matches are equal to the cardinality.

To clarify Formula 1 we discuss the following three situations:

- (a) Strong match: There are a lot of equal Word Senses between the two terms under investigation. This will lead to a high impact on the calculation of the similarity.
- (b) Weak match: Only a few Word Senses are equal between the two terms under investigation. The association between these two terms will only have a small impact on the calculation of the similarity.
- (c) No match: None of the Word Senses of the two terms are equal. The association between these two terms will have no impact on the calculation of the similarity.

For example, we take a look at the two terms cruel and cool from Table II. We can calculate a cardinality of 6 for the term cruel and a cardinality of 4 for the term cool. As there is only one matching Synset we can calculate the tension as \(\frac{1}{6} + \frac{1}{4} = \frac{5}{12}\) which is very low. Of course, two equal terms will result in the highest possible tension with the value of 2 as the Matches are equal to the cardinality.

To further enhance the recommendation of candidates for dependencies between two requirements we exploit the knowledge about the topics of requirements. This knowledge is explicitly provided by stakeholders as they define the requirement names. To clarify this approach we discuss the example shown in Figure 2. Both requirements Height Determination and Speed Measurement mention the third requirement Internal Memory in their description. As all three requirements use the two terms Internal and Memory, the similarity calculation would recommend them as equal similar to each other. Our assumption is that there is a higher tension between terms used in one requirement as topic than between terms only used in the description. We therefore doubled the value for Word Senses used in requirements topics for our similarity calculation.

### C. Reduce Dimension

In a next step we want to reduce the dimension of keywords used for the recommendation. We therefore use Apache OpenNLP\(^9\) which is a toolkit for natural language processing. With this toolkit we filter out all terms which are not classified as noun. The remaining terms are then used for the calculation of the similarity and can be presented as explanation for the recommendation.

---

\(^8\)http://www.danielnaber.de/morphologie/

\(^7\)Note that this cardinality is document independent as these sets are defined in OpenThesaurus

\(^9\)https://opennlp.apache.org/
D. Calculating Requirements Similarity

For the calculation of the similarity between two requirements in a project three well known measurements can be used: the Dice, Jaccard, and Cosine coefficients. The Dice coefficient can be found in Formula 2, which is a variation of the Jaccard coefficient “intensively” taking into account keyword commonalities [16][32].

\[ \text{sim}(r_a, r_b) = \frac{2 \times |\text{Keywords}_A \cap \text{Keywords}_B|}{|\text{Keywords}_A| + |\text{Keywords}_B|} \] (2)

Instead of user defined keywords we used the calculated tensions from Formula 1 to enhance the calculation with the knowledge derived from OpenThesaurus.

V. Empirical Studies

This Section covers the results of the studies conducted at our University with the proposed recommender.

A. Keyword Recommender

Within the scope of a study conducted at the Graz University of Technology we evaluated the quality of the aforementioned recommendation approaches. The empirical study has been conducted within the course Object-oriented Analysis and Design (N=39 software teams of size 5-6; 15.45% female, 84.55% male). In this context the teams had to create 20 requirements for a software project⁹. The creation of requirements was defined as an collaborative task and the students were encouraged to reuse¹⁰ requirements already defined by other groups.

To evaluate the effectiveness of the Keyword Recommender we randomly assigned the development teams to two groups. The first group had no recommender, while the second group was supported by the Keyword Recommender described in Section IV. Both groups used an own database to store requirements. For example, if a team of the first group generated a requirement all teams from the first group could access this requirement and were able reuse it. Teams from the second group could not see or reuse requirements from the first group and vice versa.

²The assignment for all student teams was to develop a web based hotel recommender
¹⁰Note that a reuse is not a link to a requirement. Instead the requirement is cloned and the reuse team links to the new version of the requirement. Therefore, any changes done to an reused requirement did not affected the original requirement and vice versa.

All different study groups had an interface to browse through the requirements presented as an unsorted list. Additionally, teams could filter the input by selecting keywords.

Evaluation: When evaluating the reuse behavior between the two study groups we could identify a significant increase (t-test, \( p < 0.05 \)) of reuse activity throughout the teams in the study group with Keyword Recommender. Table IV shows the results of our evaluation. From this result we derived that teams with a keyword recommendation used more often the same keywords to annotate requirements. For example, two different requirements were annotated by the keyword database instead of database and databases¹¹. Also using an example from our initial domain-specific thesaurus, users did not need to search requirements for tourist and client separately. As these words were characterized as synonyms in the domain used for the evaluation, the Keyword Recommender proposed the keyword tourist each time the term client or tourist was contained in a requirement description. This resulted in a shorter filter list of keywords which was used to facilitate the search through the list of reusable requirements. For example, teams with this advantage did not need to evaluate requirements found with tourist and databases. They retrieved all related requirements with a single click. One drawback of this approach was that it always uses the first written keyword. For example, if there is a typo in the written keyword (e.g. databasesO), this incorrect diction will be used as a recommendation for similar keywords with the same stemmed version. Of course, using a thesaurus and a correction tool could solve this problem, but we used a very simple approach without any typo correction.

B. Dependency Recommender

To evaluate the dependency detection approach we created a set with 30 requirements for a sport watch during a brainstorming session. The requirements covered functionality such as internal memory, training evaluation, connectivity to a PC system, and sensor measurements like heart rate. The evaluation of potential dependencies between the requirements was not covered in the brainstorming.

In a next step we applied our Dependency Recommender on the set of requirements to generate a ranked list of potential dependencies. According to Formula 3 with \( n = 30 \) (number of requirements in our project) there exist 435 possible dependencies between two requirements. We also discovered a significant decrease of the calculated similarity value after the top

<table>
<thead>
<tr>
<th>Group</th>
<th>Keyword Recommender</th>
<th>Reuse Requirement</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No</td>
<td>141</td>
<td>39.83%</td>
</tr>
<tr>
<td>2</td>
<td>Yes</td>
<td>213</td>
<td>60.17%</td>
</tr>
<tr>
<td>Total</td>
<td>-</td>
<td>354</td>
<td>100%</td>
</tr>
</tbody>
</table>

¹¹The annotation with database and databases could be found in the study group without the Keyword Recommender
20 recommendations. To evaluate the quality of the calculated recommendation we conducted a second study at Graz University of Technology (N=23 participants; 8.69% female, 91.31% male). As we were only interested in evaluating the quality of the recommendation we printed out the recommendations and presented them to study participants offline. We want to point out that the Dependency Recommender is integrated in the online version of IntelliReq and recommendations are calculated on demand without the necessity for precedent offline calculations.12

During the study the participants had to decide if the recommended requirement pairs are worth a look for the task of finding dependencies. Our approach was not designed to automatically find dependencies, but to support stakeholders with a preselection of interesting dependency candidates for evaluation. Finally, the participants were asked to rate the overall usefulness of the recommendation on a 5-point scale with 1 (very useful) to 5 (not useful). The results can be found in Figure 4. Based on the decrease of the calculated similarity value we presented only the best 20 recommendations as we assumed a high increase in false-positive recommendations based on the low calculated similarity values [16]. This should prevent a bias of the overall satisfaction which would occur by showing many recommendations with very low calculated similarity.

Figure 5 shows the result of the acceptance evaluation for the first 20 recommendations. The value agree can be seen as true-positive, while disagree counts for false-positive. The dimensions true-negative and false-negative were not covered as for this the participants would have to evaluate all 435 possible dependency candidates to find pairs for good recommendation not already presented by the Dependency Recommender.

\[
allPossibleRec = n \times (n - 1)/2
\]  

To evaluate the study result we define a recommendation as accepted if more than 75% of the participants agreed on their usefulness (see Formula 4). Using this Formula we see that 70% of the first 10 recommendations were accepted. Although the quality of the recommendations deteriorates for the next 10 recommendations we can still notice that 60% of the recommendations were supported.

12Note that in the current version only the German language is supported for the similarity calculation.
We enhanced our *Keyword Recommender* with domain knowledge in the form of a manually generated thesaurus and defined one *Word Sense* for each term. During evaluation of this approach we discovered a significant increase of reuse activity by software development teams using our *Keyword Recommender*. Furthermore, we conducted a study to evaluate the quality of our similarity measurement technique used in the *Dependency Recommender*. We took 20 recommendations from 435 possible combinations between requirements with the highest calculated similarity score and presented them to study participants. Although our proposed approach is rather trivial, 13 of 20 recommendations were accepted by study participants. Also, the perceived usefulness of this kind of recommendation was rated high with an average of 2.17.

In order to further improve the quality of dependency detection mechanisms in *IntelliReq*, approaches from natural language processing [34] and text mining [35] have to be combined with content-based approaches currently included in *IntelliReq*. We also want to evaluate the applicability of micro-tasks within groups of stakeholders (as used, for example, in *Amazon Mechanical Turk*) to generate and maintain domain-knowledge thesaurus to enhance our content-based recommendation techniques [36].

**REFERENCES**


