Extracting Paraphrases of Technical Terms from Noisy Parallel Software Corpora

Xiaoyin Wang[1,2], **David Lo**[1], **Jing Jiang**[1], **Lu Zhang**[2], **Hong Mei**[2]
[1]School of Information Systems, Singapore Management University, Singapore, 178902
{xywang, davidlo, jingjiang}@smu.edu.sg
[2]Key Laboratory of High Confidence Software Technologies (Peking University), Ministry of Education
Beijing, 100871, China
{zhanglu, meih}@sei.pku.edu.cn

Abstract

In this paper, we study the problem of extracting technical paraphrases from a parallel software corpus, namely, a collection of duplicate bug reports. Paraphrase acquisition is a fundamental task in the emerging area of text mining for software engineering. Existing paraphrase extraction methods are not entirely suitable here due to the noisy nature of bug reports. We propose a number of techniques to address the noisy data problem. The empirical evaluation shows that our method significantly improves an existing method by up to 58%.

1 Introduction

Using natural language processing (NLP) techniques to mine software corpora such as code comments and bug reports to assist software engineering (SE) is an emerging and promising research direction (Wang et al., 2008; Tan et al., 2007). Paraphrase extraction is one of the fundamental problems that have not been addressed in this area. It has many applications including software ontology construction and query expansion for retrieving relevant technical documents.

In this paper, we study automatic paraphrase extraction from a large collection of software bug reports. Most large software projects have bug tracking systems, e.g., Bugzilla[1], to help global users to describe and report the bugs they encounter when using the software. However, since the same bug may be seen by many users, many duplicate bug reports are sent to bug tracking systems. The duplicate bug reports are manually tagged and associated to the original bug report by either the system manager or software developers. These families of duplicate bug reports form a semi-parallel corpus and therefore a good candidate for extraction of paraphrases of technical terms. Hence, bug reports interest us because (1) they are abundant and freely available,(2) they naturally form a semi-parallel corpus, and (3) they contain many technical terms.

Parallel bug reports with a pair of true paraphrases	
1:	connector extend with a straight line in *full screen mode*
2:	connector show straight line in *presentation mode*
Non-parallel bug reports referring to the same bug	
1:	Settle language for part of text and spellchecking part of text
2:	Feature requested to improve the management of a multi-language document
Context-peculiar paraphrases (shown in italics)	
1:	status bar appear in the middle of *the screen*
2:	maximizing window create phantom status bar in middle of *document*

Table 1: Bug Report Examples

However, bug reports have characteristics that raise many new challenges. Different from many other parallel corpora, bug reports are *noisy*. We observe at least three types of noise common in bug reports. First, many bug reports have many spelling, grammatical and sentence structure errors. To address this we extend a suitable state-of-the-art technique that is robust to such corpora, i.e. (Barzilay and McKeown, 2001). Second, many duplicate bug report families contain sentences that are not truly parallel. An example is shown in Table 1 (middle). We handle this by considering lexical similarity between duplicate bug reports. Third, even if the bug reports are parallel, we find many cases of *context-peculiar paraphrases*, i.e., a pair of phrases that have the same meaning in a very narrow context. An example is shown in Table 1 (bottom). To address this, we introduce two notions of *global context-based score* and *co-occurrence based score* which take into account all good and bad occurrences of the phrases in a candidate paraphrase in the corpus. These scores are then used to identify and remove

[1]`http://www.bugzilla.org/`

context-peculiar paraphrases.

The contributions of our work are twofold. First, we studied the important problem of paraphrase extraction from a noisy semi-parallel software corpus, which has not been studied either in the NLP or the SE community. Second, taking into consideration the special characteristics of our noisy data, we proposed several improvements to an existing general paraphrase extraction method, resulting in a significant performance gain – up to 58% relative improvement in precision.

2 Related Work

In the area of text mining for software engineering, paraphrases have been used in many tasks, e.g., (Wang et al., 2008; Tan et al., 2007). However, most paraphrases used are obtained manually. A recent study using synonyms from WordNet highlights the fact that these are not effective in software engineering tasks due to domain specificity (Sridhara et al., 2008). Therefore, an automatic way to derive technical paraphrases specific to software engineering is desired.

Paraphrases can be extracted from non-parallel corpora using contextual similarity (Lin, 1998). They can also be obtained from parallel corpora if such data is available (Barzilay and McKeown, 2001; Ibrahim et al., 2003). Recently, there are also a number of studies that extract paraphrases from multilingual corpora (Bannard and Callison-Burch, 2005; Zhao et al., 2008).

The approach in (Barzilay and McKeown, 2001) does not use deep linguistic analysis and therefore is suitable to noisy corpora like ours. Due to this reason, we build our technique on top of theirs. The following provides a summary of their technique.

Two types of paraphrase patterns are defined: (1) Syntactic patterns which consist of the POS tags of the phrases. For example, the paraphrases "a VGA monitor" and "a monitor" are represented as "DT_1 JJ NN_2" $\leftrightarrow$ "DT_1 NN_2", where the subscripts denote common words. (2) Contextual patterns which consist of the POS tags before and after the phrases. For example, the contexts "in the middle of" and "in middle of" in Table 1 (bottom) are represented as "IN_1 DT NN_2 IN_3" $\leftrightarrow$ "IN_1 NN_2 IN_3".

During pre-processing, the parallel corpus is aligned to give a list of parallel sentence pairs. The sentences are then processed by a POS tagger and a chunker. The authors first used identical words and phrases as seeds to find and score contextual patterns. The patterns are scored based on the following formula: *(n+)/n*, in which, *n+* refers to the number of positively labeled paraphrases satisfying the patterns and *n* refers to the number of all paraphrases satisfying the patterns. Only patterns with scores above a threshold are considered. More paraphrases are identified using these contextual patterns, and more patterns are then found and scored using the newly-discovered paraphrases. This co-training algorithm is employed in an iterative fashion to find more patterns and positively labeled paraphrases.

3 Methodology

Our paraphrase extraction method consists of three components: sentence selection, global context-based scoring and co-occurrence-based scoring. We marry the three components together into a holistic solution.

Selection of Parallel Sentences Our corpus consists of short bug report summaries, each containing one or two sentences only, grouped by the bugs they report. Each group corresponds to reports pertaining to a single bug and are duplicate of one another. Therefore, reports belonging to the same group can be naturally regarded as parallel sentences.

However, these sentences are only partially parallel because two users may describe the same bug in very different ways. An example is shown in Table 1 (middle). This kind of sentence pairs should not be regarded as parallel. To address this problem, we take a heuristic approach and only select sentence pairs that have strong similarities. Our similarity score is based on the number of common words, bigrams and trigrams shared between two parallel sentences. We use a threshold of 5 to filter out non-parallel sentences.

Global Context-Based Scoring Our context-based paraphrase scoring method is an extension of (Barzilay and McKeown, 2001) described in Sec. 2. Parallel bug reports are usually noisy. At times, some words might be detected as paraphrases incidentally due to the noise. In (Barzilay and McKeown, 2001), a paraphrase is reported as long as there is a *single* good supporting pair of sentences. Although this works well for a relatively clean parallel corpus considered in their work, i.e., novels, this does not work well for bug reports. Consider the context-peculiar example in Table 1 (bottom). For a context-peculiar para-

phrase, there can be many sentences containing the pair of phrases but very few support them to be a paraphrase. We develop a technique to offset this noise by computing a *global* context-based score for two phrases being a paraphrase over *all* their parallel occurrences. This is defined by the following formula: $S_g = \frac{1}{n}\Sigma_{i=1}^{n} s_i$, where n is the number of parallel bug reports with the two phrases occurring in parallel, and s_i is the score for the i'th occurrence. s_i is computed as follows:

1. We compute the set of patterns with affixed pattern scores based on (Barzilay and McKeown, 2001).
2. For the i'th parallel occurrence of the pair of phrases we want to score, we try to find a pattern that matches the occurrence and assign the pattern score to the pair of phrases as s_i. If no such pattern exists, we set s_i to 0.

By taking the average of s_i as the global score for a pair of phrases, we do not rely much on a single s_i and can therefore prevent context-peculiar paraphrases to some degree.

Co-occurrence-Based Scoring We also consider another global co-occurrence-based score that is commonly used for finding collocations. A general observation is that noise tends to appear in random but random things do not occur in the same way often. It is less likely for randomly paired words or paraphrases to co-occur together many times. To compute the likelihood of two phrases occurring together, we use the following commonly used co-occurrence-based score:

$$S_c = \frac{P(w_1, w_2)}{P(w_1)P(w_2)}. \quad (1)$$

The expression $P(w_1, w_2)$ refers to the probability of a pair of phrases w_1 and w_2 appearing together. It is estimated based on the proportion of the corpus containing both w_1 and w_2 in parallel. Similarly, $P(w_1)$ and $P(w_2)$ each corresponds to the probability of w_1 and w_2 appearing respectively. We normalize the co-occurrence to the range of 0-1 by dividing it with the size of the corpus.

Holistic Solution We employ the parallel sentence selection as a pre-processing step, and merge co-occurrence-based scoring with global context-based scoring. For each parallel sentence pairs, a chunker is used to get chunks from each sentence. All possible pairings of chunks are then formed. This set of chunk pairs are later fed to the method in (Barzilay and McKeown, 2001) to produce a set of patterns with affixed scores. With this we compute our global-context based scores. The co-occurrence based scores are computed following the approach described above.

Two thresholds are used and candidate paraphrases whose scores are below the respective thresholds are removed. Alternatively, one of the score is used as a filter, while the other is used to rank the candidates. The next section describes our experimental results.

4 Evaluation

Data Set Our bug report corpus is built from OpenOffice[2]. OpenOffice is a well-known open source software which has similar functionalities as Microsoft Office. We use the bug reports that are submitted before Jan 1, 2008. Also, we only use the summary part of the bug reports.

We build our corpus in the following steps. We collect a total of 13,898 duplicate bug reports from OpenOffice. Each duplicate bug report is associated to a master report—there is one master report for each unique bug. From this information, we create duplicate bug report groups where each member of a group is a duplicate of all other members in the same group. Finally, we extract duplicate bug report pairs by pairing each two members of each group. We get in total 53,363 duplicate bug report pairs.

As the first step, we employ parallel sentence selection, described in Sec. 3, to remove non-parallel duplicate bug report pairs. After this step, we find 5,935 parallel duplicate bug report pairs.

Experimental Setup The baseline method we consider is the one in (Barzilay and McKeown, 2001) without sentence alignment – as the bug reports are usually of one sentence long. We call it *BL*. As described in Sec. 2, *BL* utilizes a threshold to control the number of patterns mined. These patterns are later used to select paraphrases. In the experiment, we find that running *BL* using their default threshold of 0.95 on the 5,935 parallel bug reports only gives us 18 paraphrases. This number is too small for practical purposes. Therefore, we reduce the threshold to get more paraphrases. For each threshold in the range of 0.45-0.95 (step size: 0.05), we extract paraphrases and compute the corresponding precision.

In our approach, we first form chunk pairs from the 5,935 pairs of parallel sentences and then use the baseline approach at a low threshold to ob-

[2] `http://www.openoffice.org/`

tain patterns. Using these patterns we compute the global context-based scores S_g. We also compute the co-occurrence scores S_c. We rank and extract top-k paraphrases based on these scores. We consider 4 different methods: We can use either S_g or S_c to rank the discovered paraphrases. We call them *Rk-*S_g and *Rk-*S_c. We also consider using one of the scores for ranking and the other for filtering bad candidate paraphrases. A threshold of 0.05 is used for filtering. We call these two methods *Rk-*S_c*+Ft-*S_g and *Rk-*S_g*+Ft-*S_c. With ranked lists from these 4 methods, we can compute precision@k for the top-k paraphrases.

Results The comparison among these methods is plotted in Figure 1. From the figure we can see that our holistic approach using global-context score to rank and co-occurrence score to filter (i.e., *Rk-*S_g*+Ft-*S_c) has higher precision than the baseline approach (i.e., *BL*) in all ks. In general, the other holistic configuration (i.e., *Rk-*S_c*+Ft-*S_g) also works well for most of the ks considered. Interestingly, the graph shows that using only one of the scores alone (i.e., *Rk-*S_g and *Rk-*S_c) does not result in a significantly higher precision than the baseline approach. A holistic approach by merging global-context score and co-occurrence score is needed to yield higher precision.

In Table 2, we show some examples of the paraphrases our algorithm extracted from the bug report corpus. As we can see, most of the paraphrases are very technical and only make sense in the software domain. It demonstrates the effectiveness of our method.

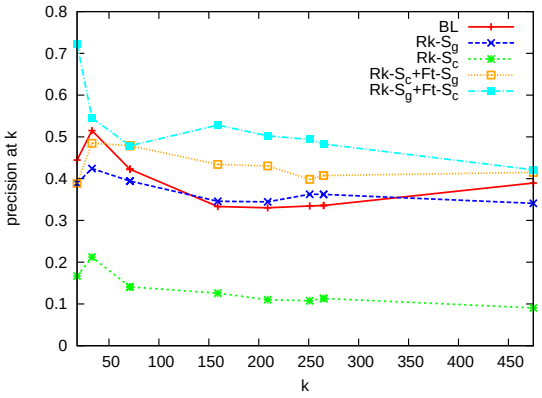

Figure 1: Precision@k for a range of k.

5 Conclusion

In this paper, we develop a new technique to extract paraphrases of technical terms from software bug reports. Paraphrases of technical terms have been shown to be useful for various software engineering tasks. These paraphrases could not be obtained via general purpose thesaurus e.g., WordNet. Interestingly, there is a wealth of text data, in particular bug reports, available for analysis in open-source software repositories. Despite their availability, a good technique is needed to extract paraphrases from these corpora as they are often noisy. We develop several approaches to address noisy data via parallel sentence selection, global-context based scoring and co-occurrence based scoring. To show the utility of our approach, we experimented with many parallel bug reports from a large software project. The preliminary experiment result is promising as it could significantly improves an existing method by up to 58%.

the edit-field	↔	input line field
presentation mode	↔	full screen mode
word separator	↔	a word delimiter
application	↔	app
freeze	↔	crash
mru file list	↔	recent file list
multiple monitor	↔	extended desktop
xl file	↔	excel file

Table 2: Examples of paraphrases of technical terms mined from bug reports.

References

C. Bannard and C. Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *ACL: Annual Meet. of Assoc. of Computational Linguistics*.

R. Barzilay and K. R. McKeown. 2001. Extracting paraphrases from a parallel corpus. In *ACL: Annual Meet. of Assoc. of Computational Linguistics*.

A. Ibrahim, B. Katz, and J. Lin. 2003. Extracting structural paraphrases from aligned monolingual corpora. In *Int. Workshop on Paraphrasing*.

D. Lin. 1998. Automatic retrieval and clustering of similar words. In *ACL: Annual Meet. of Assoc. of Computational Linguistics*.

G. Sridhara, E. Hill, L. Pollock, and K. Vijay-Shanker. 2008. Identifying word relations in software: A comparative study of semantic similarity tools. In *ICPC: Int. Conf. on Program Comprehension*.

L. Tan, D. Yuan, G. Krishna, and Y. Zhou. 2007. /*icomment: bugs or bad comments?*/. In *SOSP: Symp. on Operating System Principles*.

X. Wang, L. Zhang, T. Xie, J. Anvik, and J. Sun. 2008. An approach to detecting duplicate bug reports using natural language and execution information. In *ICSE: Int. Conf. on Software Engineering*.

S. Zhao, H. Wang, T. Liu, and S. Li. 2008. Pivot approach for extracting paraphrase patterns from bilingual corpora. In *ACL: Annual Meet. of Assoc. of Computational Linguistics*.