Gemkow, Tim; Conzelmann, Miro; Hartig, Kerstin; Vogelsang, Andreas

Automatic glossary term extraction from large-scale requirements specifications

Gemkow, Tim; Conzelmann, Miro; Hartig, Kerstin; Vogelsang, Andreas (2018): Automatic glossary term extraction from large-scale requirements specifications. In: RE 2018. 26th IEEE International Requirements Engineering Conference, August 20 - 24, 2018 - Banff, Alberta, Canada. - New York: IEEE.

Automatic Glossary Term Extraction from Large-Scale Requirements Specifications

Tim Gemkow, Miro Conzelmann, Kerstin Hartig, Andreas Vogelsang
Technische Universität Berlin, Germany
{tim.gemkow,miro.conzelmann}@campus.tu-berlin.de, {kerstin.hartig, andreas.vogelsang}@tu-berlin.de

***Abstract*—Creating glossaries for large corpora of requirments is an important but expensive task. Glossary term extraction methods often focus on achieving a high recall rate and, therefore, favor linguistic proecssing for extracting glossary term candidates and neglect the benefits from reducing the number of candidates by statistical filter methods. However, especially for large datasets a reduction of the likewise large number of candidates may be crucial. This paper demonstrates how to automatically extract relevant domain-specific glossary term candidates from a large body of requirements, the CrowdRE dataset. Our hybrid approach combines linguistic processing and statistical filtering for extracting and reducing glossary term candidates. In a twofold evaluation, we examine the impact of our approach on the quality and quantity of extracted terms. We provide a ground truth for a subset of the requirements and show that a substantial degree of recall can be achieved. Furthermore, we advocate requirements coverage as an additional quality metric to assess the term reduction that results from our statistical filters. Results indicate that with a careful combination of linguistic and statistical extraction methods, a fair balance between later manual efforts and a high recall rate can be achieved.**

***Index Terms*—Requirements Engineering, Natural Language Processing, Glossary Term Extraction, Crowd RE**

I. Introduction and Relevance to RE

Glossaries are an important tool in Requirements Engineering (RE). In general, a glossary provides definitions for the technical terms in a domain. Furthermore, it usually includes domain-specific information about the synonyms, concepts, and related terms as well as examples of their application [1]. Glossaries serve to ensure that all stakeholders have a common understanding of key terms. To this end, it is necessary to have a set of unambiguous terms that clearly define the key concepts used in the requirements to minimize the risk of misinterpretation by any stakeholder [2].

Best practice in RE holds that a glossary should be developed continuously during the process of requirements elicitation [2]. However, in many real-world projects, this effort is not expended initially, but a glossary is still needed in later stages of a project to support requirements comprehension, consolidation, and management. Hence, we assume that glossary creation starts from a body of pre-existing requirements.

Creating glossaries for large corpora of requirements is an expensive process. This paper aims at automating the first key step of extracting relevant candidates for glossary terms from a body of existing requirements. The expected result is

- a machine-generated list of glossary term candidates and
- an index linking from each glossary term candidate to the relevant requirements from which it was acquired.

To extract term candidates, we implement a hybrid approach that first identifies possible terms by a set of natural language processing (NLP) steps, and afterwards reduces the number of terms by two statistical filters. We applied our approach to the CrowdRE dataset [3], which contains about 3,000 crowd-generated user stories for smart home applications. The final generated list contains 326 glossary term candidates. On a subset of 100 requirements for which term candidates have been manually derived by us, a recall of 74.9% and a precision of 73.4% is achieved.

Our paper is the first to report on automatic glossary term extraction on large-scale requirements specifications, represented by the CrowdRE dataset, which is 5 to 10 times larger than datasets that have been used for evaluating related approaches [2][4]. We investigate the effects of single parts of our approach (linguistic extraction and statistical filtering) to the dataset and argue that for large-scale requirements specifications, statistical filters are inevitable although they may reduce recall.

We conclude that our approach is generally useful to support the first, time-intensive step of extracting glossary term candidates from large scale requirements specifications.

In summary, this paper makes the following contributions:

1) A hybrid approach for automatically extracting relevant glossary term candidates by combining linguistic extraction and statistical filtering.
2) A replicable application and analysis of the approach to a large-scale, public dataset (the CrowdRE dataset [3]).
3) A list of manually extracted glossary term candidates for a subset of 100 requirements from the CrowdRE dataset.
4) A list of automatically extracted glossary term candidates for all requirements contained in the CrowdRE dataset.

The last two contributions may help developing a publicly available benchmark for the problem of glossary term extraction. We invite authors of similar approaches to compare their work based on our contributions.

II. The CrowdRE Dataset

The CrowdRE dataset is the result of an empirical study conducted by Murukannaiah et al. [3][5] in 2016. The study investigated the elicitation of requirements in the domain of smart home applications through crowd-sourcing techniques. 300 participants with various backgrounds were asked to formulate requirements for smart home applications through the Amazon Mechanical Turk platform. Each participant was

asked to formulate requirements in form of user stories. In a second phase, 309 new participants rated the requirements formulated by other participants on the three dimensions clarity, usefulness, and novelty. In summary, the CrowdRE dataset includes 2,966 requirements in the form of user stories with meta-information attached to each requirement (e.g., tags, ratings), and meta-information attached to each author (e.g., demographics, OCEAN personality traits).

For this paper, the requirements themselves are the primary data, since we want to extract the relevant glossary terms directly from the textual data. The following examples illustrate the format of this data:

- Req. 11: *As a* worker, *I want* my smart home to be able to order delivery food by simple voice command *so that* I can prepare dinner easily after a long day at work
- Req. 12: *As a* home occupant, *I want* my smart home to turn on certain lights at dusk *so that* I can come home to a well-lit house

We did not use the rating data or the author-specific metadata in the scope of this paper. The dataset does not contain a glossary or definitions of central terms, i.e., the dataset does not contain any *ground truth* for the problem addressed in this paper. We will later explain how we evaluated our approach without this ground truth.

III. Automatic Glossary Term Extraction

A. Criteria for Glossary Terms

The International Requirements Engineering Board (IREB) provides the following definition of a glossary: *"A collection of definitions of terms that are **relevant** in some **domain**. Frequently, a glossary also contains cross-references, synonyms, homonyms, acronyms, and abbreviations."* [6]. Hence, we can derive the following properties of good glossary terms:

- Relevance: A glossary contains only terms that are important for understanding the meaning of the documents (e.g., the requirements) in question.
- Domain specificity: A glossary contains only terms that need to be defined in the domain-specific context, because they are either unique to the domain or because the have a specific meaning in the context of this domain.

B. General Approaches for Glossary Term Extraction

The requirements are provided in the form of natural language. Current research suggests two complementary approaches to identify glossary terms in textual data [7]:

- Linguistic approaches characterize glossary term candidates by syntactic information. For example, some approaches extract the noun phrases from a text as glossary candidates and restrict their further consideration to these candidates [2]. Some authors claim that for technical glossaries 99 percent of all relevant terms for technical glossaries are noun phrases [2].
- Statistical approaches characterize glossary term candidates by the frequency of their occurrence relative to some benchmark, such as their term frequency relative to the text size or their term frequency in relation to their frequency in a larger corpus of documents. Other statistical means include mutual information measures and other factors related to lexical collocation and semantic similarities [8].

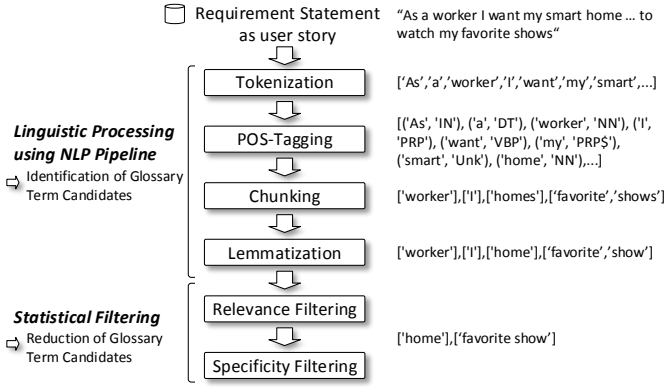

Fig. 1. Hybrid approach to identify and reduce glossary term candidates

In our analysis, we combine both approaches. First, we identify all noun phrases in our requirements as glossary term candidates (analogous to [2]). Subsequently, we use statistical filters to reduce the list of candidates to a reasonable length and quality. The implementation is done via a process pipeline that is described in the next section.

C. Hybrid Approach for Glossary Term Extraction

To automatically extract glossary terms, each requirement statement is processed in a sequential pipeline. The individual steps and an example output of each step are displayed in Fig. 1. The implementation of the pipeline is done with Python, using the Natural Language Toolkit (NLTK) framework [9]. Since we have no requirements-specific reference corpus available, we rely in particular on the Treebank corpus of Wall Street Journal articles available through NLTK. As a newspaper corpus, it is characterized by a succinct and relatively formal style which we consider somewhat related to the style of requirements documents.

1) Linguistic Processing: The first part of our approach uses linguistic processing to identify term candidates.

Input: The CrowdRE dataset provides the role, feature, and benefit for each user story as separate items. We do not process them separately but reconstruct the full user story sentence from these three pieces. Using full sentences allows for a better performance of the PoS tagging step later in the pipeline, which relies on the grammatical structure of each sentence.

Tokenization: Each sentence is split into individual tokens while preserving their order. We use the NLTK tokenizer class *nltk.word_tokenizer* that separates tokens on whitespace characters and punctuation. Whitespace gets discarded, while punctuation is returned as a token itself [10]. This is necessary because punctuation can convey important information about sentence structure useful for subsequent tagging.

Part of speech (PoS) tagging: The tokens of each sentence are tagged according to their syntactical position in the text. We use the NLTK Tagger *nltk.tag.tnt* that assigns tags using an established statistical approach to PoS tagging, the Trigrams

'n' Tags (TnT) tagger [11]. It relies on classification through a hidden Markov model, which requires training data for which the correct output is known. We train our classifier on a subset of the included Treebank corpus *nltk.corpus.treebank* of newspaper articles for which expert-annotated tokens with POS tags are available.

Chunking: We rely on the class *nltk.corpus.treebankchunk*, which uses a statistical approach to chunking and is trained with the Treebank corpus data for which the correct parse tree has been determined by human experts. This means that patterns of POS tags that define a noun phrase (NP) are learned from the treebank corpus instead of an explicit grammar definition. The chunker subsequently uses the knowledge obtained from the training data to recognize sequences of PoS tags that are characteristic for noun phrases.

Lemmatization: The identified candidate terms are reduced to their canonical form to simplify statistical analysis. This allows for the aggregation of different forms of the same word to a common glossary term. This is implemented using the *nltk.stem.Wordnetlemmatizer* that goes back to a language-specific database (WordNet) to look up a predefined canonical form for each word. Lemmatization also processes the PoS tag information for each word, since this can occasionally be used to resolve ambiguities in choosing the correct canonical form. We also convert all terms to lower case and remove individual tokens from our candidate terms that match with a list of known "stop words".

2) Statistical Filtering: The second part of our approach uses two statistical filters to reduce the number of term candidates and to increase the relevance and domain specificity.

Relevance Filtering: The first filter eliminates all candidate terms that appear in less than a specific number of requirements to ensure that all concepts included in our glossary are sufficiently relevant for the domain in question. The threshold should be selected to optimize the trade-off between an increase in precision and a possible loss of coverage (more details on that in Section V).

Specificity Filtering: With the specificity filter, we want to ensure that our glossary term candidates are sufficiently specific to the domain. We determine this by comparing the number of occurrences of term candidates in our requirements corpus with the occurrences in a non domain-specific text corpus such as in newspaper articles. Therefore, we eliminate all candidate terms that occur less frequently in our requirements than in a subset of the Treebank newspaper article corpus. This ensures that all glossary terms included in our set have a minimum degree of domain specificity.

Export: The results of our pipeline are available as a Python dictionary for further processing. They are also made available in structured output formats as csv and Excel file to support manual inspection. For practical reasons, we also maintain an index for each candidate glossary term that maps the term to the requirements that it originated from. This index is used in the statistical filtering step to determine the number of requirements covered by each candidate term. It is also made available as a result of the process to facilitate later Information Retrieval (IR). This is necessary because a reconstruction of the origin of term candidates without an index can be expensive: Stop word removal and lemmatization imply that simple full text search is not reliable; instead, the pipeline steps would need to be re-run on the original data.

D. Replication Package

A replication package for our study is available on GitHub[1]. It contains (1) the code that implements the proposed NLP pipeline and instructions how to run the analysis, (2) a csv file containing a list of manually identified glossary terms for a subset of 100 requirements of the CrowdRE dataset together with a link to the respective requirements, (3) an xlsx file containing all automatically extracted glossary term candidates together with a link to the requirements in which the terms appear and a quantitative evaluation of the related requirements per term, and (4) an extended report on the effect of variations of the NLP pipeline that we tested in the process of our research.

IV. Evaluation Plan

To evaluate the quality of the extracted glossary term candidates, it is necessary to define suitable quality metrics. A common way to evaluate keyword lists is to work with reference lists and use precision and recall as metrics. In practice, candidate term extraction would generally be followed by a second step, where domain experts formulate authoritative definitions of the relevant terms to create the actual glossary. Although these experts benefit from a concise list of terms, this setup indicates that recall is more critical than precision: Unnecessary glossary term candidates would be identified and removed in subsequent manual processing (although this consumes valuable human effort), while missing glossary term candidates would likely remain unnoticed and thus hurt the quality of the ensuing glossary. Unfortunately, a concise evaluation of precision and recall is difficult in our case, since the CrowdRE dataset does not contain a reference list of correct glossary terms (a *ground truth*).

A. Evaluation of Precision and Recall

Therefore, in a first step, we apply our approach only to a random subset of 100 requirements, for which we have manually identified glossary term candidates based on our intuition of potential glossary terms. We did not generate a full glossary. Instead, we manually generated ground truth data at the same level of abstraction achieved by our pipeline, i.e., relevant terms extracted directly from the requirements' texts. This benchmark dataset allows an estimation of precision and recall for our pipeline applied to this subset of requirements. Of course, this benchmark is not very reliable because it is created by two of the authors who extracted and discussed potential glossary terms.

Besides precision and recall, we are also interested in the characteristics of the approach when applied to large scale requirements specifications. In the light of a missing ground truth for the entire CrowdRE dataset, we therefore advocate requirements coverage as another metric for a glossary's quality.

[1] https://github.com/tgem/crowdre-glossary

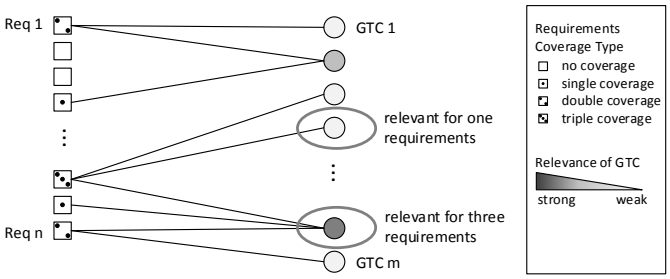

Fig. 2. Linking glossary term candidates and requirements: indicator for a candidate's relevance and its effect on requirements coverage

B. Evaluation of Requirements Coverage

In a second step, we apply our approach to the entire CrowdRE dataset (2,966 requirements) to investigate the number of resulting glossary term candidates and the coverage of requirements by the extracted glossary terms. On a very basic level, we expect a good glossary to contain all relevant domain-specific terms. Additionally, we expect that only very few requirements cannot be related to any glossary term. Therefore, we use the percentage of requirements covered by at least one, two, or three glossary terms, as indicators (see Fig. 2). A higher coverage should indicate a higher recall, as the available glossary terms can describe a bigger fraction of all requirements. However, while higher coverage is generally better, even perfect coverage does not guarantee a specific level of recall.

V. Results

A. Precision and Recall for Small Subset

We manually identified 251 relevant glossary term candidates for the subset of 100 requirements. If we solely apply our linguistic processing steps to these requirements, we automatically identify 258 glossary term candidates containing 189 true positive terms, which accounts for a recall of 75.30% and a precision of 73.26%. Please note that we also count short terms that are included as parts of longer terms of the other set. If we additionally apply the specificity filter, we can exclude two candidates from the automatically extracted list, which leads to a slightly reduced recall of 74.90% and slightly higher precision of 73.44%. Since the relevance filter aims at enhancing glossary term extraction especially for large datasets, we refrain from its application to the small subset of 100 requirements. Experiments showed that it takes beneficial effects on large datasets but can lead to rapid reduction of recall on small datasets.

B. Impact of Statistical Filtering on Entire Datset

In this section, we examine the impact of both statistical filters (relevance and domain specificity) on the number of extracted term candidates and requirements coverage when applied to the entire dataset of 2,966 requirements.

1) Number of glossary term candidates: Our first filter excludes glossary term candidates that relate to a very small number of requirements to ensure the relevance of the identified terms. Fig. 3 shows the impact of both statistical filters on the number of glossary term candidates. Since each candidate must relate to at least one requirement, the relevance filter

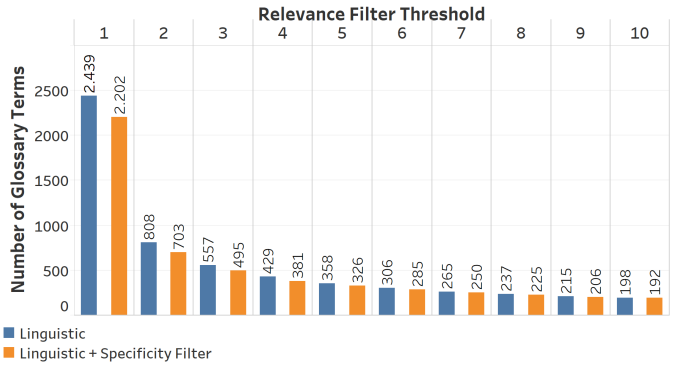

Fig. 3. Number of glossary terms for increasing relevance filter threshold

only activates at a threshold of 2. A threshold of 1 outputs the same candidates as extracted without the relevance filter. The figure reveals that a large number of candidates relate to only one requirement. When eliminating those candidates by applying a relevance filter threshold of 2, the number of candidates reduces enormously from 2,439 to 808 (2,202 to 703 with specificity filter). Higher thresholds lead to a further, less drastic reduction of the candidate list. The specificity filter has a less radical effect. Its application reduces the number of candidates extracted by solely linguistic processing by 4–13%.

2) Requirements Coverage: We advocate requirements coverage as additional heuristic for a sufficiently complete set of glossary term candidates. Fig. 4 depicts the impact of both filters on requirements coverage. Without any statistical filtering, we achieve a coverage rate of 99.76% for single coverage of each requirement. Introducing a relevance threshold comes at a very low price since requirements coverage stays above 98% even for a threshold value of 10. Again, specificity filtering has only a minor impact on coverage loss. Its application diminishes requirements coverage only slightly, e.g., from 91.98% to 91.3% for double coverage and a relevance threshold filter of 5.

The high values for single requirements coverage can be explained by noting that most role descriptions from the user stories ("home owner", "house occupant", "parent") are among the glossary term candidates, which ensures a very high coverage regardless of the specific content of the requirements. It is therefore more reasonable to examine the requirements coverage rate of double or even triple coverage for each requirement. If we analyze double covered requirements, we can observe a slight reduction of coverage, decreasing even more at higher relevance threshold. Applying specificity filtering and a relevance threshold of 5, the coverage rate of 91.3% is still satisfactory but also shows that about one tenth of all requirements could not be linked to any generalized glossary term candidate beyond the role description. Triple coverage of each requirement reduces the overall requirements coverage tremendously, e.g., from 91.3% to only 72.02%.

C. Automated Keyword Extraction

Based on the results above, we decided to include the specificity filter and a relevance filter with threshold value of 5 for the extraction of glossary terms from the entire dataset. This configuration combines a small loss of coverage with a

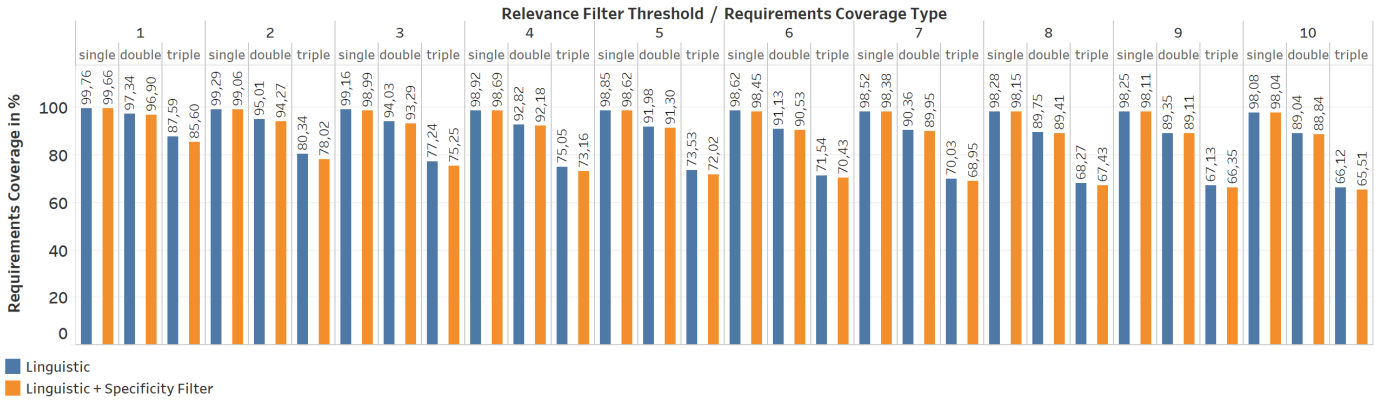

Fig. 4. Requirements coverage for coverage types and increasing relevance filter threshold evaluated on the entire dataset of 2,966 requirements.

TABLE I
EXAMPLES OF EXTRACTED GLOSSARY TERM CANDIDATES

term candidate	no. reqs	term candidate	no. reqs	term candidate	no. reqs
home occupant	1088	water	149	escape	5
home	767	child	146	hair	5
home owner	621	pet owner	138	less energy	5
house	355	person	130	sound system	5
parent	333	temperature	116	drain	5
time	245	food	108	sport fan	5
alert	190	able	107	automatic door	5
door	175	safe	95	fall	5
light	170	phone	93	dirty	5
energy	165	day	92	rural home oocupant	5

forceful reduction in the size of our glossary. If we execute our hybrid glossary term extraction approach with these settings using the CrowdRE dataset, we identify 326 glossary term candidates. Table I shows the twenty terms that are linked to the largest number of requirements and the ten terms linked to the minimum number of requirements for inclusion as a glossary term candidate. The terms demonstrate the statistical nature of our extraction pipeline, which occasionally misclassifies terms and thus also includes some very prominent terms (“able”, “safe”) that do not actually qualify as noun phrases.

VI. DISCUSSION

A. Comparison of Statistical Filters

In general, our approach corroborates that established means for term extraction through NLP can be successfully applied to glossary term candidates in a large-scale requirements specification. Since our two criteria for good glossary terms are relevance and domain specificity, we encoded them as statistical filters. Our evaluation shows that the statistical filter for domain-specific terms has a much smaller impact in terms of filtered terms than the relevance filter. Moreover, the effect size remains small even if we increase the specificity further: For testing purposes, we included only terms that are at least twice as frequent in our corpus than in the Treebank corpus. Even this more restrictive threshold led only to the exclusion of another 18 terms from our candidate list of 326 terms.

A possible explanation for this is that the relevance filter, which is applied first, already removed most unspecific terms. However, if we change the order of our filters, the basic result still holds: If we filter first for domain specificity, 237 terms get removed from the initial list of 2,439 terms in this step, but another 1,876 are removed by the relevance filter (with a threshold value of 5). Thus, while there is an overlap between both filters, the relevance threshold is substantially more discriminating in our case. This points to an interesting characteristic that seems to differentiate requirements documents from more general corpora: Requirements are already highly focused thematically and contain very little unrelated or tangential concepts (at least after stopword removal).

B. Relevance of Statistical Filters for Large Specifications

Some authors argue that any kind of filtering in glossary term construction should be avoided [2], since missed concepts are considerably more damaging than effort in manually screening candidate terms. We generally agree with this line of argument, however, our results show that approaches without any kind of filtering result in a large set of term candidates when applied to large-scale requirements specifications (2,439 terms in our case). We assume that in many cases, it is not efficient to maintain a glossary with over 2,000 entries. In fact, we are convinced that, when constructing a glossary, a human analyst is influenced by the size of the document(s) to which the glossary relates. For smaller specifications (below 500 requirements), the analyst might follow the strategy to define a large fraction of noun phrases that occur in the requirements. For larger specifications, however, the human analyst would quickly recognize that this strategy leads to a large number of glossary terms. Therefore, a human analyst would focus on terms that appear in a significant part of the underlying corpus (relevance) and that are unique to the domain (specificity). Therefore, we argue that for term extraction from smaller specifications (below 500 requirements), it might be reasonable to maximize recall. However, for larger specifications, statistical filtering is necessary to mimic the changed selection strategy of a human analyst.

C. Limitations of our approach

Our approach successfully generates a reasonable list of glossary term candidates for the CrowdRE dataset. Nevertheless, there are two major limitations to the approach presented here: We cannot aggregate different glossary term candidates based on semantic meaning, and our possibilities to evaluate

the quality of our results remain limited by the lack of a comprehensive accepted ground truth.

Conceptually, an important task of any glossary is to define the canonical terms used to refer to a given concept, and to regulate both the use of synonyms (by clarifying the preferred terms) and homonyms (by clearly disambiguating different meanings and, typically, recommending alternative terms that are clearly distinct). However, our approach can only reduce morphological variance through lemmatization. Any form of semantic synonyms are ignored and the respective terms treated as separate entities; any form of semantic homonyms are likewise ignored and incorrectly grouped together as a single glossary term candidate. Therefore, an important future extension would be to provide support for the consideration of semantic similarity (e.g., by investigating linguistic datasets such as WordNet (cf. [12]) or by through a word vector approach (cf. [13]).

A further avenue to improve the evaluation of our results would be to process the CrowdRE dataset with existing term extraction tools, and to systematically compare their results with those from our pipeline.

D. Related Work

Hybrid approaches, such as ours, have also been successful in previous research on glossary term extraction. For example, Barker et al. [14] built a pipeline that is similar to ours: As linguistic processing, PoS tags are assigned to feed a skimmer (similar to a chunker). The skimmer implements a parser to construct noun phrases based on sequences of PoS tags. Then statistical filters are used to process the extracted noun phrases based on their frequency distribution. In contrast to precision and recall, their main evaluation metric is a relevance rating of automatically extracted terms by a number of human analysts. The terms extracted by their approach achieved an average relevance rating of 0.47 on a scale from 0 (bad) to 2 (good). Other examples of hybrid approaches are proposed by Park et al. [15] and Dwarakanath et al. [4]. Note, however, that all mentioned approaches use a linguistic (rule-based) approach to chunking rather than a statistical chunker. In our case, the statistical approach has proven superior to chunking. It would be interesting to determine the conditions under which one or the other is more reliable.

The idea of hybrid approaches is not uncontested: Arora et al. [2] argue that in the RE domain, statistical filtering should not be used for glossary terms because statistical filters have the purpose to increase precision with the risk to decrease recall. In their approach without any statistical filters, they achieved a recall of more than 90% for three investigated specifications with a precision between 20 and 50%. They argue that for glossary term extraction in RE, recall should be rated much higher with precisions (cf. [16]), and therefore, their approach refrains from any statistical filters. As discussed in Section VI, we agree with this argument for smaller specifications. Consistently, Arora et al. [2], as well as Dwarakanath et al. [4] have evaluated their approaches on requirements specifications with 108–380 requirements. When applied to larger specification, as in our case, this strategy results in a large number of glossary term candidates. This observation is backed up by the work of Park et al. [15] who extracted over 9,000 terms from a document with over 200,000 words.

VII. Conclusion

Our results show that it is possible to automatically extract a reasonable list of glossary term candidates from a large-scale requirements corpus, which can substantially facilitate subsequent expert work. We have already outlined possibilities to further support expert assessment by also preprocessing likely semantic links, and believe that this is a promising avenue for future research.

Our very basic approach can easily be modulated and extended by more sophisticated filters if the limitation in evaluation data can be overcome. Moreover, it should be noted that our pipeline itself does not rely on any specifics of the CrowdRE dataset. Our approach should therefore generalize well to new datasets, and we suggest that this could also yield further valuable insights. Applying our approach to another dataset for which a reliable ground truth is known might also help to overcome the remaining uncertainties in our ability to evaluate the results.

References

[1] I. Sommerville and P. Sawyer, *Requirements Engineering: A Good Practice Guide*. New York, USA: John Wiley & Sons, Inc., 1997.

[2] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer, "Automated extraction and clustering of requirements glossary terms," *IEEE Transactions on Software Engineering*, 2017.

[3] P. K. Murukannaiah, N. Ajmeri, and M. P. Singh, "Toward automating crowd RE," in *25th IEEE International Requirements Engineering Conference (RE)*, 2017.

[4] A. Dwarakanath, R. R. Ramnani, and S. Sengupta, "Automatic extraction of glossary terms from natural language requirements," in *21st IEEE International Requirements Engineering Conference (RE)*, 2013.

[5] P. K. Murukannaiah, N. Ajmeri, and M. P. Singh, "Acquiring creative requirements from the crowd: Understanding the influences of individual personality and creative potential in crowd RE," in *20th IEEE International Requirements Engineering Conference (RE)*, 2016.

[6] M. Glinz, *A Glossary of Requirements Engineering Terminology*. International Requirements Engineering Board, 2014.

[7] C. Jacquemin and D. Bourigault, "Term extraction and automatic indexing," in *The Oxford Handbook of Computational Linguistics*. Oxford University Press, 2003, pp. 599–615.

[8] M. T. Pazienza, M. Pennacchiotti, and F. M. Zanzotto, "Terminology extraction: An analysis of linguistic and statistical approaches," in *Knowledge Mining*. Berlin, Germany: Springer, 2005.

[9] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, 2009.

[10] J. Perkins, *Python 3 Text Processing with NLTK 3 Cookbook*. Packt Publishing Ltd, 2014.

[11] T. Brants, "TnT – A statistical part of speech tagger," in *6th Applied Natural Language Processing Conference (ANLP)*, 2000.

[12] R. Richardson, A. F. Smeaton, and J. Murphy, "Using wordnet as a knowledge base for measuring semantic similarity between words," in *Proceedings of AICS conference*, 1994.

[13] A. Ferrari, B. Donati, and S. Gnesi, "Detecting domain-specific ambiguities: an NLP approach based on wikipedia crawling and word embeddings," in *25th IEEE International Requirements Engineering Conference Workshops (REW)*, 2017.

[14] K. Barker and N. Cornacchia, "Using noun phrase heads to extract document keyphrases," in *Advances in Artificial Intelligence*, 2000.

[15] Y. Park, R. J. Byrd, and B. K. Boguraev, "Automatic glossary extraction: beyond terminology identification," in *19th International Conference on Computational Linguistics*, 2002.

[16] D. M. Berry, "Evaluation of tools for hairy requirements and software engineering tasks," in *25th IEEE International Requirements Engineering Conference Workshops (REW)*, 2017.