Glossary Term Extraction from the CrowdRE Dataset

Miro Conzelmann, Tim Gemkow

February 13, 2018

Abstract

This paper demonstrates how to automatically extract relevant domain-specific glossary term candidates from a large body of requirements, the CrowdRE dataset [1]. Our approach uses Natural Language Processing (NLP) techniques for extracting term candidates for a glossary and facilitates Information Retrieval (IR) by building an index linking glossary term candidates back to the corresponding requirements. Results indicate that with a combination of linguistic and statistical extraction methods a substantial degree of recall can be achieved, although the possibilities for evaluation are limited due to the lack of an established ground truth for the given dataset. Based on several auxiliary metrics, we show that our approach is a reasonable first building block for a pipeline to support glossary creation.

1 Introduction

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 [2, p. 49]. 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 [3, p. 918].

Best practice in RE holds that a glossary should be developed continuously during the process of requirements elicitation [3, p. 918]. 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 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.

Based on the identification of these term candidates, the next steps in developing a glossary would be to derive definitions for each term candidate, to identify glossary terms with multiple meanings (homonyms) and to identify different glossary terms referring to the same concept (synonyms). However, these processing steps are beyond the scope of this paper.

To extract term candidates, we implement a specific sequence of natural language processing steps (pipeline). We subsequently investigate different approaches to evaluate the quality of the generated list. In this phase, the effect of variations in individual parts of our pipeline on the results is examined in order to optimize the processing steps and facilitate future adaptation of our approach to new datasets.

The final 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 74.5% and a precision of 73.3% is achieved if partial matches are included. Similarly, we achieve a recall of 63.5% in reconstructing tags assigned by the requirements' authors if we include partial matches (as discussed later, an estimation of precision in this respect is not possible).

We conclude that our approach is generally useful to support the first, time-intensive step of extracting glossary term candidates during glossary construction. However, the recall rates are not high enough yet to base glossary construction solely on the extracted terms; possible causes and remedies in this respect are discussed in section 4.

In Section 2 our study questions, study design and setup are described in greater detail. The results concerning the stated questions are presented in Section 3. Section 4 describes remaining limitations, as well as connections of our research to related works. Section 5 concludes by outlining promising next steps in research on automated glossary term extraction in RE.

2 Research Outline

2.1 Study Goals and Questions

This paper aims to address the following research questions:

1. Automated Keyword Extraction: How can we automatically extract relevant glossary term candidates from an existing corpus of requirements in order to support glossary construction?
2. Evaluation: How can the quality of extracted glossary term candidates be evaluated properly?
3. Pipeline Alternatives: How do variations in the extraction pipeline affect the quality of the extracted term candidates?

2.2 The CrowdRE Dataset

The CrowdRE dataset is the result of an empirical study conducted by P. K. Murukannaiah et al [1, 4] 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 and to assign an application domain and any number of tags to each of his or her requirements. The participants formulated the requirements in several stages and were provided with different input stimuli due to the research questions of P. K. Murukannaiah et al. However, these differences are not relevant for our work and all requirements produced during any stage adhere to the same format. In a second phase, 309 new participants rated the requirements formulated by other participants on the three dimensions clarity, usefulness and novelty. Moreover, all workers from both phases also provided data regarding their demographics and conducted a personality test to determine their position on the standard OCEAN scale of personality traits and a standardized test to assess their creative potential. Hence, the CrowdRE dataset includes requirements, 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. All requirements are given in user story format, with role, feature and benefit available as separate strings. We also reference the application domain and the set of tags assigned to each requirement. 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>

  Application domain: <Health>, tags: <food, delivery, dinner, voice>

- 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>

  Application domain: <Energy>, tags: <lights, turn on, night>

- Req. 13: As a <worker>, I want <my smart home to sync with my biorhythm app and turn on some music that might suit my mood when I arrive home from work> so that <I can be relaxed>

  Application domain: <Entertainment>, tags: <music, biorhythm, mood>

Each requirement is assigned exactly one application domain from the list of "Entertainment", "Energy", "Health", "Safety", and "Other". Every author was free to assign any number of tags to a requirement, and was able to formulate these freely without any predefined reference set or list.

We did not use the rating data or the author-specific metadata in the scope of this paper. We investigated, but finally did not use the application domain data since the categories were too broad to be useful for our purpose.

2.3 Study Execution

2.3.1 Criteria for Glossary Terms

The International Requirements Engineering Board (IREB) provides the following consensus 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."* [5, p. 14, our emphasis]

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.[1]

2.3.2 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 [6, p. 573][2]:

- Linguistic approaches characterize glossary term candidates by syntactic information. As an example, many authors extract the noun phrases from a text as glossary candidates and restrict their further consideration to these candidates (e.g. [3]). Some authors claim that for technical glossaries 99 percent of all relevant terms for technical glossaries are noun phrases [3].
- 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].

In our analysis, we combine both approaches. First, we identify all noun phrases in our requirements as glossary term candidates (analogous to [3]). Subsequently, we use a statistical filter 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.

2.3.3 NLP Pipeline

Each requirement statement is processed in a sequential pipeline. The individual steps as well as an example output of each step are displayed in figure 1. For each step, we have evaluated different alternatives. For clarity, this overview presents only the final pipeline adopted here; we later discuss and evaluate the possible alternatives.

The implementation of the pipeline is done with Python, using the Natural Language Toolkit (NLTK) [9] framework. NLTK is one of the standard open source libraries available in the domain of NLP that provides all necessary modules to build a solid processing pipeline. On top of the processing modules it contains about 50 text corpora and and lexical resources that can be easily

[1] As an example, "Eigentum" and "Besitz" would need to be included in a glossary for an application supporting civil law suits in Germany since these terms have specific, and distinct, meanings in legal language while considered synonyms in everyday parlance.

[2] A combination of these two approaches is also adopted in [7].

accessed for training, testing, extraction and comparison. 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.

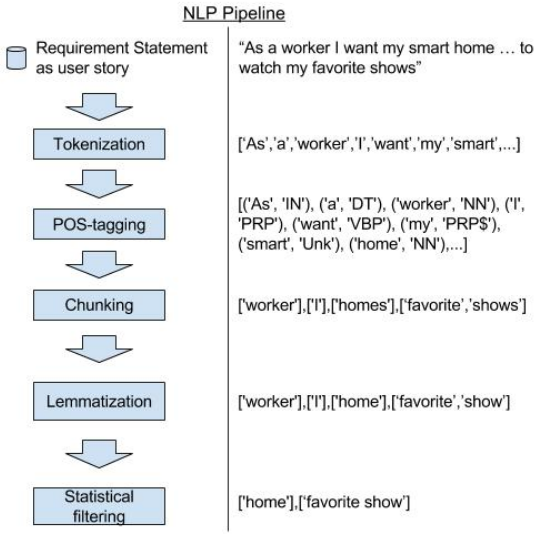

Figure 1: The Processing Pipeline Used to Determine Term Candidates

1. User Story as Input: The CrowdRE dataset provides the role, feature and benefit for each user story as separate items. However, 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.

2. 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, p. 10]. This is necessary because punctuation itself can convey important information about sentence structure useful for subsequent tagging. The tokenizer provides the basic data structure that is necessary for further processing.

3. 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.

4. Chunking: Extracting noun phrases to identify candidate glossary terms.

   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.

5. 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. At the same time, 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". The example in 1 shows the reduction of 'favorite shows' to 'favorite show' in the lemmatization step.

6. Statistical filtering: In order to reduce the number of candidate terms and to increase the relevance and domain specificity, two statistical filters are applied.

   The first filter eliminates all candidate terms that apply to less than five requirements to make sure that all concepts included in our glossary are sufficiently relevant for the domain in question. The threshold of five has been chosen to optimize the trade-off between an increase in precision and a possible loss of coverage (more details in section 3.3, see in particular figure 3).

   In a second step, we eliminate all candidate terms which occur less frequently among our requirements than in a subset of the Treebank corpus newspaper articles. This ensures that all glossary terms included in our set have a minimum degree of domain specificity.

7. Export: The results of our pipeline are available as a Python dictionary for further processing. They are also made available in stuctured 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.

2.3.4 Evaluation

In order to evaluate the quality of the extracted glossary term candidates and to discuss alternative pipeline configurations, 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 [8, p. 10]. In our case they are interpreted in the following way:

- *Precision* is the fraction of relevant terms (terms that occur in the reference list) among all terms extracted by our procedure. A high precision indicates that our approach produces few false positives, i.e. our list is not cluttered with unnecessary terms.
- *Recall* is the percentage of extracted glossary terms among all glossary terms of the reference list. A high recall indicates that our approach produces a comprehensive list, i.e. we do not miss relevant glossary terms.

In a field application, 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 to be defined, this setup indicates that recall is even more critical in our case 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. The process can be compared with a brainstorming session with the aim to generate a broad set of concepts that can be sorted out afterwards.

Unfortunately, a concise evaluation of precision and recall is impossible in our case, since our dataset does not contain a reference list of correct glossary terms (a "ground truth") that could be used as benchmark. Therefore, we test several auxiliary metrics as possible substitute measures.

On a very basic level, we would expect a good glossary to contain all relevant domain-specific terms and hence 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 glossary term, and the

percentage of requirements covered by at least two glossary terms, as indicators. 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.

Moreover, each requirement in the dataset has been assigned a varying number of tags by its author, which identify important concepts or categories related to the requirement. These tags serve to characterize the requirements and are in this respect similar to the glossary terms that we seek. We have therefore investigated the usefulness of these tags as a potential benchmark for our glossary terms. Our initial idea was to measure the degree of agreement between tags and glossary terms on two levels:

- To what extent do our glossary terms match the tags in the dataset?
- To what extent do the requirements assigned to our glossary terms match the requirements assigned to the tags in the dataset?

The second step was meant to identify cases where the tag may use a different word ("intruder" instad of "trespasser"), but still covers exactly the same requirements and might thus express the same concept as one of our terms. For example, requirement 86 "As a home occupant, I want an air purity controller so that it detects pollution and sensors get active" is tagged with "healthy air" and "pollution check", neither of which are used expressly in the statement itself. If our pipeline identifies the same concepts, it will necessarily characterize them by canonical forms of expressions actually used in the requirement.

However, as we discuss in more detail in the results section, tag assignment in the original dataset has varied widely between different requirements' authors and follows no stringent procedure. Therefore, while tags assigned to requirements are important indicators, the fact that a tag has not been assigned to a requirement does not reliably indicate that it is not related to this concept. Therefore, we have been able to use tags only as a minimum benchmark for recall (as we should ideally reconstruct the tags that have been assigned). Comparing the requirements sets associated with each tag, or using the tags also as a measure of precision (in the sense that glossary term candidates not matched by a tag would be false positives), is not possible due to their unsystematic assignment.

Due to these limitations, we have proceeded to manually generate additional data for evaluation purposes. We have hand-coded appropriate terms for a subset of 100 requirements from the dataset. We did not generate a full glossary, however, since our pipeline is not capable of semantic integration of results (cf. the discussion in section 4). Instead, we manually generated ground truth data at the same level of abstraction achieved by our pipeline, i.e. as relevant terms extracted directly from the requirements' texts. As in our pipeline, terms where identified as noun phrases and individual words were reduced to their root form (in particular, by using singular form for nouns), but no terms were used that did not feature in the requirements themselves. This benchmark dataset allows us an estimation of precision and recall for our pipeline applied to this subset of requirements. Our approach is similar, but less ambitious, than that of [3] who use a full manually-constructed glossary as reference list.

Another possible quality measure would be to compare our results with those obtained by an established and successful glossary term extraction tool. We have therefore contacted Mr Chetan Arora and asked wether we could be provided with the tool successfully used in [3]. However, Mr Arora has indicated that he would only make a revised version available to us, which was not received in time for the revision deadline for this paper.

3 Results

3.1 Automated Keyword Extraction

Our method is capable of automatically extracting a set of glossary terms from the complete list of CrowdRE requirements. We identify 326 different potential glossary terms. Table 1 shows the twenty terms that are linked to the largest number of requirements, as well as ten terms linked to the minimum number of requirements for inclusion as a glossary term candidate.

The quality of these results regarding different evaluation criteria is part of the next section. The terms demonstrate the statistical nature of our extraction pipeline, which occasionally misclassifies

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

Table 1: Examples of extracted glossary term candidates

terms and thus also includes some very prominent terms ("able", "safe") that do not actually qualify as noun phrases.

3.2 Evaluation

Our first heuristic for a sufficiently complete set of glossary term candidates is coverage. The result in this regard (fig. 2) seems impeccable: 98.6% of all requirements are covered by at least one glossary term. However, we note 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 look at the percentage of requirements covered by at least two different glossary terms. With 91.3%, this rate 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. Figure 2 expands this procedure up to 5 candidates per requirement and shows the expected behavior.

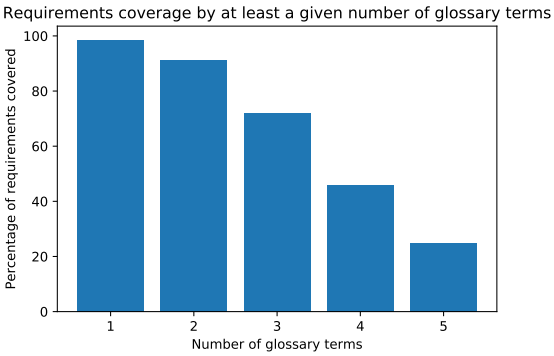

Figure 2: Coverage of Requirements by Minimum Number of Relevant Glossary Term Candidates

Then we compared our glossary term candidates with the respective tags attached to each requirement in the dataset. To prevent a flawed comparison, we previously use the same kind of filtering on the tags that we also use for the glossary terms (e.g. if we only include terms that cover at least five different requirements, our comparison also includes only those tags that cover at least five different requirements).

The number of glossary terms (326) is substantially larger than the number of tags (192) after this normalization. There is a substantial degree of overlap between both categories. 74 tags from the reference list are also identified as candidate terms by our procedure which indicates a precision around 38%. Moreover, five additional terms are included as parts of longer tags, and 43 tags are included as parts in longer glossary term candidate expressions identified by us, further increasing the degree of overlap. We generally count these partial matches as successful links as well, since requirements experts in the following consolidation step can correct whether to use the longer or shorter expression for the glossary with relative ease. Under this assumption, we achieve a recall rate of 63.5% with regard to the tags.

Regarding the precision it is important to understand whether the extracted term candidates that are not matched by tags should be considered as false positives. As the tags were assigned by requirements' authors without specific guidance and with no predefined reference list, it turns out that they have not been used systematically enough to support such an evaluation. To verify the comprehensiveness of tag assignment, we have taken the set of terms that were identified both by our procedure and by the tagging, and have investigated whether they cover the same requirements. As it turns out, there are errors in both directions. As expected, tags are sometimes applied to requirements were they fit thematically but are not present in the explicit wording; these cases are missed by our procedure. But tags are also missing from requirements were they would clearly apply. As a prominent example, our procedure assigns the glossary term candidate 'temperature' to 116 requirements from which it is missing as tag. Manual inspection reveals that our procedure is generally correct in assigning the term, e.g. to requirement 31 "As a home occupant I want my smart home to adjust the *temperature* based on the weather outside so that I can save on my energy bill by not using the AC unnecessarily.". Hence, while assigned tags bear important meaning, the absence of tags cannot be reliably interpreted in this dataset.

For a subset of 100 requirements, relevant glossary term candidates have been manually identified to provide a further basis for evaluation. Note that this process did not intend to create a valid glossary in that no semantic consolidation of term candidates was conducted. Instead, we only conducted the steps of extracting and normalizing term candidates from the requirements themselves, in line with the steps that we propose to automatize in this paper. If we apply our procedure only to these 100 requirements, we automatically identify 102 out of 251 terms extracted manually (40%). Moreover, as when matching with the tags, a key point seems to be the diverging length of identified term candidates. If we also count short terms that are included as parts of longer terms of the other set, we achieve a recall of 74.5% of the assigned terms, similar to the degree of matching between our term candidates and the predefined tags. Because our coding has been conducted more systematically, this reference set can also be used for a measure of precision. In this respect, it indicates that our 169 glossary terms automatically extracted from the 100 requirements is too large: If we include partial matches, only 73.3% of the automatically extracted terms also feature in the manually derived list.

3.3 Altering Pipeline Parameters

In order to find out how the choice and parameters of the pipeline modules influence the outcome, we compared major alternatives for each step.

Tokenization

Tokenization of English-language sentences into individual tokens is a well-understood problem, for which available standard solutions generally provide good results. A possible point of contention is the treatment of contractions such as "can't", "won't" and "they're". Our default tokenizer splits these contractions into two tokens, but leaves them otherwise unchanged, i.e. "can't" becomes ("ca","n't") and "they're" becomes ("they","re") [10, p. 11]. In principle, this is undesirable because not all of the resulting tokens are valid words. We therefore implemented a preprocessing step that uses regular expressions to expand common contractions to their long forms ("cannot","they are") prior to tokenization. However, empirically, the difference is not relevant: Because the selected terms do not contain contractions, the final pipeline results are almost identical with and without the preprocessing. Only a single term is added to the result list, which did not appear convincing on manual inspection ("move"). Based on this finding, we ultimately chose to omit the preprocessing step from our approach, although it might be needed when applying our method to different data.

Part of Speech Tagging

A variety of approaches exists for PoS tagging, of which most rely on training an appropriate statistical model but differ in the nature of this model. We have adopted a so-called TnT tagger, which internally uses a second order Markov model to decide about the most likely tag of each token given the training data [10, p. 104].

Initially, we experimented with a simpler sequential chain of taggers, the so-called backoff tagger: If the token is assigned a unique tag in the training data, it is assigned this tag. Otherwise,

the tagger checks the bi- and then the trigram containing this word. If it has a unique PoS tag in the training data given this context, it is assigned the appropriate tag. Finally, if trigram tagging also fails, the word is assigned the default tag of 'NN' as the most common PoS tag (see [10, p. 94f] for this approach). This simpler model leads to a very substantial loss of quality in the pipeline results compared with the adopted solution. With the simpler tagger, the number of identified glossary terms increases by about 40%, and only 28% of the identified terms are shared between this result and our main solution. Closer manual investigation reveals that this expansion in the result set is primarily due to chunking failures, which result from the deteriorated PoS tagging. As an example, based on the simpler tagger, ('home', 'occupant', 'I', 'want', 'shower') and ('home', 'occupant', 'I', 'want', 'refrigerator') are incorrectly identified as noun phrases and subsequently kept as glossary terms (only the 'I' is removed by a later correction step). Conversely, ('home', 'occupant') itself is no longer identified as a relevant term when using the backoff tagger. We therefore prefer the TnT tagger.

Thus, as expected, part-of-speech tagging is a crucial step in our pipeline and our results are highly vulnerable to deficiencies in the tagger.

Chunking

Because we focus on noun phrases, we use the PoS information for identifying them. As presented above, we use a statistical chunking method that extracts chunks based on patterns detected in an annotated corpus of reference data.

A key alternative would be to use rule-based partial parsing ("chunking") of our PoS-tagged sequences of tokens. In this approach, the parser identifies sequences of tokens whose PoS tags match a specific regular expression as noun phrases. We have implemented a slightly modified version of the pattern successfully used by [12], which identifies noun phrases by matching POS tags to the regular expression <DT>?(<JJ>(<CC><JJ>)*|(<NN>|<NP>|<NPS>))*(<NN>|<NP>|<NPS>).

The change in chunking method has substantial consequences for the outcome of our pipeline. The total number of glossary term candidates is reduced to 228 (-98). The new term set achieves a recall of 52.6% including partial matches regarding the tags (-10.9%), and a recall of 57.8% including partial matches in reconstructing our manually generated terms for 100 requirements (-16.7%). At the same time, due to the smaller result set, the precision on the 100 requirements is substantially improved to a level of 85.8% including partial matches (+12.5%). Hence, the choice of chunking method represents a genuine trade-off between a more concise and a more complete list of glossary term candidates.

Since we assume that our list is provided as input to domain experts for glossary construction, we consider the effort of discarding a number of unnecessary term candidates less important than the danger of missing important concepts. We have therefore chosen to prioritize recall over precision, and to maintain the statistical approach to chunking.

Lemmatization

For a comprehensive but compact dictionary, it seems inevitable to link different linguistic variants of the same concept back to some canonical form of the respective term. Since we are concerned with noun phrases, the most important part of this process is to reduce singular and plural forms of a word to the same root.

When omitting this step in our pipeline, the total number of identified glossary terms increases by 10 to 336. Moreover, under the hood, the quality of terms suffers considerably: The glossary now includes singular and plural terms as distinct entities in a number of cases (e.g. "activity"/"activities" and "light"/"lights"). Conversely, a considerable number of relevant concepts are no longer present since their different forms could not be aggregated and each form separately failed to meet the statistical thresholds for inclusion. As an example, while "automatic" is still present, the specific concepts of "automatic door" and "automatic light" are no longer present in the index. This result confirmed our decision to reduce candidate terms to a common root form.

The major alternative available to our approach is to use stemming, which uses predefined rules to cut back ("stem") the endings of words to create a common root. For the English language, several well-developed systems of stemming rules are available. However, the key disadvantage of these approaches is that the stemmed root forms may not themselves be valid words, whereas lemmatization guarantees that the result is a valid word and hence suitable as a glossary term. This is the main reason why we continue to prefer lemmatization for the task at hand.

For testing purposes, we have implemented Porter stemming as an alternative. As a result, the size of our index increases by only 6 terms. There is no comparable loss of quality as experienced when using no root form reduction at all. However, the actual words used for the glossary terms are changed for 61.2% of the glossary entries, as the stemming form differs from the lemmatized form. As examples, the entry 'home occupant' is replaced by 'home occup'. We have therefore decided to continue using lemmatization.

Statistical Filtering

We apply statistical filtering to ensure relevance and domain specificity of our glossary term candidates.

Our primary filter is to exclude glossary term candidates that relate to only a very small number of requirements. In this way, we want to ensure the relevance of the identified terms. As figure 3 shows, a large number of term candidates relates only to a small number of requirements. Thus, these terms serve more as descriptors for these requirements than as indication of concepts important for the domain, and are excluded even by low thresholds. As thresholds gets higher, the additional reduction in the number of retained terms diminishes quickly.

Moreover, introducing a small threshold comes at a very low price with regard to requirements coverage. Taking the percentage of requirements covered by at least two glossary terms as our primary measure, figure 4 documents the small loss in coverage for low thresholds. The loss would be even smaller for coverage by at least one glossary term, which stays above 98% even for a threshold value of 10.

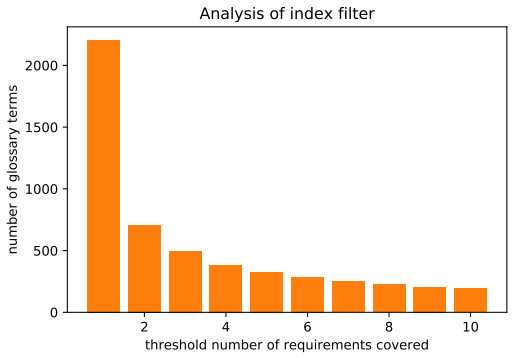

Figure 3: Number of Terms Depending on Threshold Number of Requirements Covered by Retained Terms

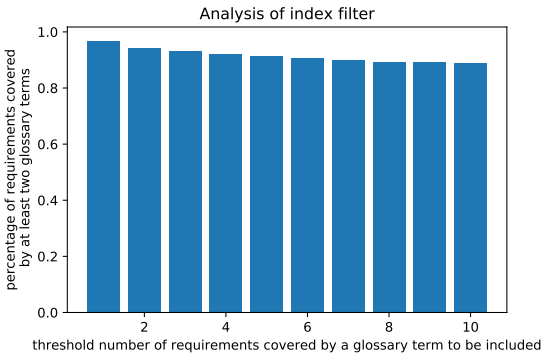

Figure 4: Requirements Coverage by at Least Two Candidate Terms Depending on Filter Threshold

Based on these results, we stayed with the decision to include only glossary terms relating to at least five requirements, which combines a small loss of coverage with a forceful reduction in the size of our glossary.

With a second filter, we want to ensure that our glossary term candidates are sufficiently specific to the domain. It is important to note that we want to characterize whether term candidates

are specific for our entire corpus (as compared to other texts), not for individual requirements. Being only one sentence, the individual requirements are too short to meaningfully derive term frequencies. As soon as a term candidate appears once in a requirement, it can be assumed to be relevant to it.

To determine which term candidates are characteristic for our entire set of requirements, a standard approach would be the so-called term frequency-inverse document frequency (td-idf) metric. However, this requires the availability of a larger corpus of different texts (e.g. entire requirements documents) for which the inverse document frequency can be derived. Since these were not available, we have adopted a simpler approach with the same general principle: We take an existing corpus of newspaper sentences from the Treebank data, and extract term candidates from it using our pipeline. Subsequently, we compute the ratio of the number of requirements linked to a term candidate and the total number of requirements (for our original data) and the ratio of the number of sentences linked to a term candidate and the total number of sentences (for the Treebank data). We discard any glossary term candidate that is less frequent by this metric in our data than in the Treebank data due to lack of domain specificity. In total, 32 term candidates are excluded in this way (e.g. 'worker', 'stock', 'question', 'product'). Compared to the reduction achieved by the relevance filter, this is only a small amount. 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 (fig. 4); even this extreme threshold led only to the exclusion of another 18 terms form our candidate list.

A possible explanation for this could be that the previous filtering step 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 (instead of 32) get removed in this step, but another 1877 are removed only later by the relevance threshold. Thus, while there is an overlap between both filters, the relevance threshold is substantially more discriminating in our case.

Some authors argue that any kind of filtering in glossary term construction should be avoided [3, p. 919], since missed concepts are considerably more damaging than effort in manually screening candidate terms. However, given the large size of our result set without any filtering, we remain convinced of an intermediate approach where some filtering is done to reduce excess terms, but the filtering thresholds are set rather low to prioritize recall over precision.

4 Discussion

4.1 Unexpected results

In general, our approach corroborates that established means for term extraction through NLP can be successfully applied to glossary term candidates in a requirements corpus. Since our two criteria for good glossary terms are relevance and domain specificity, we did not initially expect that the second statistical filter for domain-specific terms would have such a small impact on the overall result of our pipeline. 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). Therefore, when extracting terms from requirements documents that belong to a single domain, our experience suggests that an additional filtering for domain specificity is barely necessary and can yield only very limited benefits.

4.2 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. Moreover, these limitations interact with each other. 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. Several potential avenues are conceivable for this:

- Linguistic datasets, such as WordNet, contain information on words that can serve as synonyms for each other. This data could be used to detect glossary term candidates that are likely synonyms for each other (cf. [13] on using WordNet data for semantic similarity determination).
- It would also be possible to compare the requirements from which the glossary term candidates were generated, for example through a word vector approach. Glossary term candidates that were extracted from very similar requirements contexts might also be candidates for similar meanings (cf. [14] for using word embeddings to measure semantic similarity).

Given current best practices in this respect, the most promising approach will likely combine automated and human processing, where the techniques mentioned above generate hypotheses about semantic relationships which could be presented to domain experts for validation. [3] proposes a similar approach by directly clustering candidate terms and subsequently presenting these clusters to human experts.

Note that an approach to semantic integration would also greatly facilitate overcoming our second limitation, the lack of an authoritative ground truth for glossary terms from our dataset. Currently, it is very difficult to interpret mismatches between glossary terms and predefined tags in the dataset, since it is entirely possible that these still represent the same concepts but simply use different words for it. To a lesser extent, this also creates difficulties in comparing our hand-coded set with the generated data. Adding a processing step to consolidate semantic meaning, which could be applied to the data used for comparison as well, could therefore also increase the reliability of our measurements. This, in turn, would allow to more precisely investigate further options to tune and improve the processing pipeline.

Moreover, it would also be useful to explicitly model hierarchical relationships between glossary term candidates in such a step. Recall that a recurring problem with evaluating our data was that different methods pick noun phrases of different length, and there is no simple answer to the question whether e.g. "garage door" or "automated garage door" is the appropriate level for glossary terms. However, if we could explicitly code one as a subset of the other, we could more reliably compare result sets that make different choices in this respect, and could potentially also contribute to the development of useful models of the relationship between domain concepts.

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

Finally, our ability to use more complicated statistical models to decide about the domain-specificity of glossary term candidates has been hampered by the lack of specific corpora for comparison. Our experience suggests that this is only a minor limitation since the terms present in the requirements already seem relatively focused. Nevertheless, it would be interesting to see whether a more advanced technique could further increase our precision. If, for example, a sufficiently large corpus of requirements documents from different domains were available for reference, we could employ a traditional term frequency-inverse document frequency rating in our statistical filtering step.

4.3 Related Work

As already discussed, in order to classify different approaches of glossary term extraction Pazienza et al [8] suggest to separate the approaches by their means of classification. They identified three ways to approach the problem: pure linguistic approaches that extract terms on the basis of linguistic properties only, purely statistical approaches that determine relevant terms on the basis of occurrence frequency and other statistical measures and finally hybrid approaches that combine the two. As they also provide a comprehensive overview over these different methods, their framework is a useful starting point to situate our paper in the research literature.

Since our approach takes advantage of both linguistic tagging and chunking as well as statistical filtering, it can be classified as a hybrid approach. Concerning the linguistic part, we cover the two main modules identified in [8], the PoS Tagger and the "recognizer" (i.e. the chunker in our case). However, our pipeline does not include a named entity recognizer that would extend the

chunker. Considering the statistical means the paper presents a large set of potential metrics. It differentiates between measures of unithood (which express strength or stability of syntagmatic collocations) and measures of termhood (which express how much a linguistic unit is related to domain-specific concepts), and between measures for the degree or the significance of an observed relationship. We use only a small amount of these metrics, implicitly in the statistical chunker for unithood determination and explicitly in our two frequency-based statistical filters. However, deploying more complex statistical calculations would also need improved metrics to gauge their impact. In terms of evaluation our approach is close to what Pazenza et al [8, p. 10] describe as "reference list evaluation". However, as outlined above, we remain limited by the nature of our reference lists.

Hybrid approaches have also been successful in previous research on glossary term extraction. As an example, Barker et al [15] built a pipeline that is basically similar to ours: As linguistic processing, PoS tags are assigned using a dictionary lookup. These are feed into a skimmer (similar to a chunker) in a second step. The skimmer implements a parser to construct noun phrases which uses a purely grammatical approach to detect noun phrases based on sequences of PoS tags. Then statistical filters are used to process the extracted noun phrases based on their frequency distribution. The distributions are used to construct a rating and finally thresholds are defined in a second filtering step. In contrast to the usual precision and recall their main evaluation metric is the Kappa-Score that involves human judges. The research already referred to by Park et al [12] similarly uses a hybrid approach. Note, however, that both of these papers have chosen a linguistic (rule-based) approach to chunking rather than a statistical chunker in their mixture of methods, while the statistical approach to chunking has proven superior in our case. Hence, it would be interesting to determine the conditions under which one or the other is more reliable. Moreover, the very idea of hybrid approaches is not uncontested: Arora et al [3] forcefully maintain that in the RE domain, statistical filtering should not be used for glossary terms, although this means that human expert filtering needs to be supported in other ways (e.g. by clustering in their study).

Zou et al. [16] show that extracted glossary terms can serve as a step in a larger process and propose enhancement heuristics for the automatic generation of traceability links that rank potential links higher when they are related to glossary terms. This shows that glossary term candidate extraction may also serve as a building block for solving other automation problems in RE beyond glossary construction as such.

5 Conclusion

As our results show, it is possible to automatically extract a list of candidate glossary terms from a mid-sized 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, while we use tag data for evaluation, it should be noted that our extraction pipeline itself does not rely on any specifics of the CrowdRE dataset and processes only the textual requirements data. 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] Pradeep K. Murukannaiah, Nirav Ajmeri, and Munindar P. Singh. Acquiring creative requirements from the crowd: Understanding the influences of individual personality and creative potential in crowd RE. In *Proceedings of the 20th IEEE International Requirements Engineering Conference (RE)*, pages 176–185, Beijing, September 2016. IEEE Computer Society.

[2] Klaus Pohl and Chris Rupp. *Basiswissen Requirements Engineering.* dpunkt, 2015.

[3] Chetan Arora, Mehrdad Sabetzadeh, Lionel Briand, and Frank Zimmer. Automated extraction and clustering of requirements glossary terms. *IEEE Transactions on Software Engineering*, 2017.

[4] Pradeep K. Murukannaiah, Nirav Ajmeri, and Munindar P. Singh. Toward automating crowd re. In *Requirements Engineering Conference (RE), 2017 IEEE 25th International*, pages 512–515. IEEE, 2017.

[5] Martin Glinz. *A Glossary of Requirements Engineering Terminology.* International Requirements Engineering Board, 2014.

[6] Christian Jacquemin and Didier Bourigault. Term extraction and automatic indexing. In *The Oxford Handbook of Computational Linguistics*, pages 599–615. Oxford University Press, 2003.

[7] Anurag Dwarakanath, Roshni R Ramnani, and Shubhashis Sengupta. Automatic extraction of glossary terms from natural language requirements. In *Requirements Engineering Conference (RE), 2013 21st IEEE International*, pages 314–319. IEEE, 2013.

[8] Maria T. Pazienza, Marco Pennacchiotti, and Fabio M. Zanzotto. Terminology extraction: An analysis of linguistic and statistical approaches. In *Knowledge Mining*, pages 255–279, Berlin, Germany, 2005. Springer.

[9] Steven Bird. Nltk: The natural language toolkit. In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 69–72, Linguistic Data Consortium, University of Pennsylvania,Philadelphia PA 19104-2653, USA, 2006. Proceedings of the COLING/ACL.

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

[11] Thorsten Brants. Tnt - a statistical part of speech tagger. In *Proceedings of the 6th Applied Natural Language Processing Conference ANLP 2000.* ANLP, 2000.

[12] Youngja Park, Roy J Byrd, and Branimir K Boguraev. Automatic glossary extraction: beyond terminology identification. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics, 2002.

[13] Ray Richardson, Alan F Smeaton, and John Murphy. Using wordnet as a knowledge base for measuring semantic similarity between words. In *Proceedings of AICS conference*, pages 1–15, 1994.

[14] Alessio Ferrari, Beatrice Donati, and Stefania Gnesi. Detecting domain-specific ambiguities: an nlp approach based on wikipedia crawling and word embeddings. In *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, pages 393–399. IEEE, 2017.

[15] Ken Barker and Nadia Cornacchia. Using noun phrase heads to extract document keyphrases. In *Advances in Artificial Intelligence*, pages 41–52, Montreal, Quebec, Canada, 2000. 13th Biennale Conference of the Cannadian Society for Computational Studies of Intelligence.

[16] Xuchang Zou, Raffaella Settimi, and Jane Cleland-Huang. Improving automated requirements trace retrieval: A study of term-based enhancement methods. *Empirical Softw. Eng.*, 15:119–146, 2010.