Abbreviation-Expansion Pair Detection for Glossary Term Extraction *

Hussein Hasso[1] ✉, Katharina Großer[2][0000−0003−4532−0270], Iliass Aymaz[1], Hanna Geppert[1], and Jan Jürjens[2,3][0000−0002−8938−0470]

[1] Fraunhofer FKIE, 53343 Wachtberg (Bonn), Germany
{hussein.hasso,iliass.aymaz,hanna.geppert}@fkie.fraunhofer.de
[2] University of Koblenz-Landau, 56070 Koblenz, Germany
{grosser,juerjens}@uni-koblenz.de
[3] Fraunhofer ISST, 44227 Dortmund, Germany

Abstract. *Context and motivation:* Providing precise definitions of all project specific terms is a crucial task in requirements engineering. In order to support the glossary building process, many previous tools rely on the assumption that the requirements set has a certain level of quality. *Question/problem:* Yet, the parallel detection and correction of quality weaknesses in the context of glossary terms is beneficial to requirements definition. In this paper, we focus on detection of uncontrolled usage of abbreviations by identification of abbreviation-expansion pair (AEP) candidates. *Principal ideas/results:* We compare our feature-based approach (ILLOD) to other similarity measures to detect AEPs. It shows that feature-based methods are more accurate than syntactic and semantic similarity measures. The goal is to extend the glossary term extraction (GTE) and synonym clustering with AEP-specific methods. First experiments with a PROMISE data-set extended with uncontrolled abbreviations show that ILLOD is able to extract abbreviations as well as match their expansions viably in a real-world setting and is well suited to augment previous term clusters with clusters that combine AEP candidates. *Contribution:* In this paper, we present ILLOD, a novel feature-based approach to AEP detection and propose a workflow for its integration to clustering of glossary term candidates.

Keywords: Requirements Engineering · Glossary Term Extraction · Abbreviation-Expansion Pair Detection · Synonym Detection.

1 Introduction

One of the goals in requirements engineering is to improve an opaque system comprehension into a complete system specification [28]. Activities related to glossary building support that goal, since glossaries serve to improve the accuracy and understandability of requirements written in natural language [3].

* Second author supported by European Space Agency's (ESA) NPI program under NPI No. 4000118174/16/NL/MH/GM.

Cite as: Hasso, H., Großer, K., Aymaz, I., Geppert, H., Jürjens, J. (2022).
Abbreviation-Expansion Pair Detection for Glossary Term Extraction.
In: Gervasi, V., Vogelsang, A. (eds) Requirements Engineering: Foundation for Software Quality.
REFSQ 2022. Lecture Notes in Computer Science, vol 13216. Springer, Cham.
https://doi.org/10.1007/978-3-030-98464-9_6

According to the *International Requirements Engineering Board* (IREB) [12], a glossary is a collection of definitions of terms that are relevant in a specific domain. In addition, a glossary frequently contains cross-references, synonyms, homonyms, and abbreviations [12]. Glossaries serve to enrich the requirement texts with additional important information, which ensures that technical terms are used and understood correctly, and supports communication among project participants [19]. The consequent use of a complete and accurate glossary leads to a more consistent language, resulting in coherent structures for the requirements, which in turn enhances automatic analysability [8,26]. Finally, a glossary can be reused for future projects within the same application domain to facilitate requirements elicitation and analysis [18].

In order to obtain the mentioned benefits, a glossary should be developed during the requirements elicitation phase, which is also compliant to best practices [19,27]. For various reasons, many projects tend to build their glossary after the requirements elicitation phase [1,2,11]. However, this complicates the task, since requirements written without the use of a glossary are more likely to contain imprecise or ambiguous wordings. When multiple terms are used to refer to the same meaning (synonyms), denote specializations (hyponyms), or terms have multiple meanings (homonyms), this presents a major challenge for the identification of glossary terms. Therefore, beforehand, the analyst has to ensure that the terminology is used consistently, e.g. through syntactic or semantic adjustments. This task affects various inter-requirement relations in parallel.

With this paper, we present an approach that encourages the analyst to start with the glossary building, even when the requirements quality still shows weaknesses, and contributes to resolve two tasks in parallel: (1) quality improvement through reduction of lexical variation and (2) glossary term identification. In particular, we integrate the detection of abbreviations and their expansions.

2 Problem Definition

We briefly focus on the main problems to be solved by an automated tool for the identification of glossary terms (GTE) [3,8,11].

First, since 99% of glossary entries are *noun phrases* (NPs) [14,17]:

(A) A GTE tool needs to have an *accurate noun phrase detection.*

Second, as glossaries deal with domain specific terms and omit duplicates:

(B) A GTE tool needs to *filter* detected NPs to *glossary term candidates.*

Considering (A), *Natural Language Processing* (NLP) pipelines for noun phrase detection, e.g., through *chunking approaches*, are shown to be effective [1,2]. As such, in this paper we focus on devising an effective technique for (B). Here, *statistical filters* composed of *specificity* and *relevance* measures, as presented by Gemkow et al. [11], could be used, in which beforehand identification of homonyms, synonyms, and different spelling variants among detected noun phrases is expected to have a positive effect on accuracy. Since we explicitly consider requirement sets with such quality weaknesses, we first focus on:

(B1) A GTE tool needs to *identify and/or merge homonyms, synonyms, hyponyms and different spelling variants* among detected noun phrases.

To detect such relations among domain terms is also beneficial for the building of initial domain models. In order to check whether a given pair of terms is synonymous, homonymous, or hypernymous, the underlying concepts themselves must be disambiguated [14], which requires good knowledge about the relevant domain. Therefore, candidate term pairs still have to be confirmed or rejected by the analyst. To keep the manual effort low, *term clusters* are a suitable method of representation [2]. A cluster of size n can combine $(n(n-1))/2$ term pairs. For example, the *REGICE* tool [2] follows a synonym clustering approach. Yet, only *context-based* (semantic) and *text-based* (syntactic) similarity [34] are considered and *abbreviations* must have been cleaned up and defined beforehand. Homonyms and hyponyms are not explicitly addressed. Yet, they can be spotted as bycatch. However, for homonyms this is only the case for non-disjoint clusters.

For higher recall in synonym detection, additionally *pattern-based* similarity [34] for *controlled abbreviations* can be applied. It refers to clauses where abbreviations are defined by their corresponding expansions using parentheses or keywords such as *"also known as"*, *"abbreviated"* and *"a.k.a."*, e.g.,

- Common Business Oriented Language **abbreviated** COBOL
- AES **(**Advanced Encryption Standard**)**
- Compression / Decompression, **also known as** Codec

More interesting, however, is an algorithm that also supports to resolve *uncontrolled abbreviations*, which are not defined in place when they are used. Uncontrolled abbreviations in requirements are rather common, especially when requirements elicitation is carried out by different persons (in different organizations) and when guidelines for the use of abbreviations are missing or not followed. Abbreviations can be homonymous by having multiple possible expansions within the same requirements set, as they are predominantly used in a project- or domain-specific context, and new projects regularly come up with new word creations. Thus, simple look-up techniques on predefined lists are not sufficient. This leads us to the next problem statement:

(B1.1) A GTE tool needs to exploratorily resolve hitherto unknown abbreviations in comparison to other terms present in the given text.

Since the abbreviation list is part of the glossary, both should be built in parallel. The goal is to enable a specific synonym detection optimized for matching of abbreviations with their expansions, which can be integrated to the clustering in glossary term extraction (GTE) tools. For that, we first compare the accuracy of syntactic and semantic similarity measures with feature-based classification approaches applied to abbreviation-expansion pairs (AEPs). Further, we introduce *ILLOD*, a binary classifier extending the algorithm of Schwartz and Hearst [31]. It checks **I**nitial **L**etters, term **L**engths, **O**rder, and **D**istribution of characters. Finally, we propose how tools like ILLOD can be integrated into the clustering of glossary term candidates.

3 Related Work

For glossary term extraction (GTE), Gemkow et al. [11] showed how to reduce the number of glossary term candidates by using relevance and specificity *filters*. Improving the precision of glossary term extraction like this is important especially for large data-sets. Yet, they do not regulate the possible presence of synonyms and homonyms when determining term frequencies.

Arora et al. [2] argue that *clustering* of glossary term candidates has the advantage to better mitigate false positives from noun phrase detection (A) and to support candidate filtering (B). In addition, their approach provides guidelines on how to tune clustering for a given requirements set, to detect synonyms with high accuracy. They conclude that disjoint clusters should be produced in order to keep the workload for term identification low. In Section 6 of this paper, we look at this from a new perspective.

There are various approaches for the extraction and recognition of *abbreviation-expansion pairs* (AEPs). In addition to *statistical* [24,36] and *rule-based* methods [29,32], there are also *machine learning* methods [35]. Many publications deal with biomedical texts and a few, like Park et al. [25], with the field of computer science. Most work assumes that AEPs are predefined in the text via certain patterns and focus their analyses on the surrounding context of the detected abbreviations, which is also the case for Schwartz and Hearst [31]. In our work, we extend the algorithm *findBestLongForm* presented by Schwartz and Hearst [31] to make it applicable for cross-comparisons where an abbreviation and its expansion may occur in different sentences/requirements and are distributed over the given text. We also show that this extension—ILLOD—can be used beneficially in extraction and identification of requirements glossary terms.

4 Abbreviation Detection

The first step to AEP-matching is the identification of abbreviations. Since "[t]he styling of abbreviations is inconsistent and arbitrary and includes many possible variations" [21], abbreviation extraction is usually achieved by finding *single words* that are *relatively short* and have *several capital letters* [20,31,33]. This way, not only acronyms are addressed, but also other forms of abbreviations.

For this task, we implement a simple detection algorithm. It returns *"true"* for a given word w, if the capital letter portion and the word length exceed respectively fall below specified parameter values, otherwise it returns *"false"*. We test this method on a cleaned list of 1786 abbreviation-expansion pairs known from the field of information technology [7][1] with abbreviations of different styles. We reference this list with L, all abbreviations $a \in L$ with A, and all expansions $e \in L$ with E. To identify suitable parameters for word length and the proportion of capital letters, we perform *F1-optimisation* through an exhaustive search on all possible combinations of the two parameters. The search is conducted in the

[1] For reproduction purposes, this list is also included in the supplemental material [13].

range from 0.01 to 1.0 (in 1/100 steps) for the capital letter portion parameter and from 1 to 20 for the word length parameter. The algorithm is once tested on all $a \in A$ and once on all $e \in E$ to obtain false negative and false positive assignments respectively.

After optimization, with **word-length** <= 13 and **proportion of capital letters** >= 0.29 we achieve Precision = 0.922, Recall = 0.923, and F1 = 0.922. On full written text, to keep such high accuracy, we apply an additional stop word filter sorting out words whose lower case matches a stop word and which only have one uppercase letter, as first letter, e.g. "The", "Any", or "If".

5 Detection of AEP Candidates

For AEP detection, different types of similarity measures are eligible. In a nutshell, words are *semantically* similar if they have the same meaning and *syntactically* similar if they have a similar character sequence [10]. Semantic measures rely on data from large corpora or *semantic nets* —models of terms and their relations, whereas "syntactic measures operate on given words and their characters without any assumption of the language or the meaning of the content" [10]. Finally, *feature-based* similarity rates features that are common to a given pair of words, e.g. the order of certain letters. Below, we compare three different types of classifiers for AEP detection based on these three types of similarity measures.

5.1 AEP Detection with Semantic Similarity Measures

Most methods to semantic similarity need to know queried terms in advance. This applies to knowledge-based methods that rely on lexical databases such as *WordNet* [23] and corpus-based methods such as *Word2vec* [22]. As a result, these methods are not suitable to solve (B1.1). Thus, we chose FastText (FT) [4] as a generic approach and state-of-the-practice technique to assess the suitability of semantic similarity methods for AEP detection. To assign an abbreviation a to a *potential* expansion t in the upcoming evaluation, our simple semantic classifier returns whether

$$cosine\_similarity(embed_{FT}(a), embed_{FT}(t)) \geq threshold,$$

where the *cosine_similarity* for two vectors x and y is defined as $\frac{x^T y}{\|x\|\|y\|}$, and $embed_{FT}$ stands for embedding with FastText.

5.2 AEP Detection with Syntactic Similarity Measures

The second type of classifier uses syntactic similarity measures between a and t. For this, several measures, as summarized by Gali et al. [10], can be used, like *Levenshtein-Distance* (LD)[2], *Jaro-Winkler-Similarity* (JWS), an extension

[2] We do not choose the extended *Damerau–Levenshtein-Distance* as it considers transpositions and LD is therefore more sensitive to changes in the sequence of letters.

of *Jaro-Similarity*, and the *Dice-Coefficient* (DC). However, the use of syntactic similarity measures to detect AEPs is limited. Typically, abbreviations contain only a small proportion of the letters of their respective extensions. E.g., the pair ("ISO", "International Organization for Standardization") has only a share of 3/14 common characters compared in lower case. This is also reflected in Table 1, where the similarities between randomly selected pairs from L are rather low.

Table 1. Syntactic and semantic similarities between randomly chosen AEPs $(a, e) \in L$ *Distance measures d normalized to similarity in $[0, 1]$ by $1-(d(a,e)/max(|a|,|e|))$ [10]

Abbreviation-Expansion Pair (a, e)	**LD***	**JWS**	**DC**	**FT**
(LED monitor, Ligth-emitting diode)	0.15	0.435	0.818	0.30
(Int, integer)	0.286	0.651	0.667	0.20
(PS/2, Personal System/2)	0.235	0.436	0.444	0.19
(IANA, Internet Assigned Numbers Authority)	0.114	0.612	0.316	0.093
(SMM, System Management Mode)	0.136	0.586	0.307	0.142
(U/L, upload)	0.0	0.0	0.444	0.025
(IAP, Internet access provider)	0.042	0.458	0.375	0.06
(CLNS, connectionless network service)	0.0	0.0	0.471	0.076
(MMC, MultiMediaCard)	0.214	0.603	0.333	0.533
(I/O, input/output)	0.083	0.472	0.6	0.147

Table 2. Average syntactic similarities for all $(a, e) \in L$ and $(a, \hat{a})$ with $\hat{a} = potAbb(e)$ *Distance measures d normalized to similarity in $[0, 1]$ by $1-(d(a,x)/max(|a|,|x|))$[10]

Compared Pairs	**LD***	**JWS**	**DC**	**with pre-processing**
(a, e)	0.092	0.309	0.419	no
	0.183	0.637	0.422	yes
$(a, \hat{a})$	0.361	0.422	0.861	no
	0.797	0.896	0.865	yes

To overcome this difficulties, the matching between an abbreviation a and some possible expansion t can be estimated by creating a *potential* abbreviation $\hat{a} = potAbb(t)$ out of the initial letters of the single tokens of t. Similarity is then measured between a and $\hat{a}$. This contraction allows to compare a and t on a homogeneous representation level. Table 2 summarizes the average values of the syntactic comparisons between (a, e) as well as $(a, \hat{a})$ for all pairs $(a, e) \in L$, where $\hat{a} = potAbb(e)$ following the just mentioned contraction approach.

Further, we apply *pre-processing* by converting the string into lower case letters, removing punctuation marks and the stop words "for", "and", "of", "in", "via" and "be". Table 2 shows, that pre-processing and contraction have a positive effect for all three examined measures. For (a, e), the average (normalized) Levenshtein-Distance improves by 0.705, average Jaro-Winkler-Similarity

by 0.587, and the average Dice Coefficient by 0.446. Thus, with $a^c = preprocess(a)$ and $t^c = preprocess(t)$, the second type of classifiers returns whether

$$syntacticSimilarityMeasure(a^c, potAbb(t^c)) \geq threshold.$$

Although abbreviations usually have short length—in our dataset the average after pre-processing is 3.55—it can be assumed that $\hat{a}$ and a still differ in many cases despite pre-processing. For the Levenshtein-Distance, there is a relative difference of 20.3% in average between $\hat{a}$ and a even after pre-processing, which shows that, as assumed [21], the formation and use of abbreviations in computer science is not subject to fixed guidelines/regulations in practice. Even though the average Jaro-Winkler-Similarity and the average Dice-Coefficient-Similarity are close to their ideal value of 1.0, they are potentially prone to many false positive assignments. We address this assumption in Section 5.4.

5.3 AEP Detection with Feature-Based Similarity Measures

The third type of classifier is represented by *ILLOD*, an extension of the algorithm *findBestLongForm* [31] that we implemented in *Python*. Whether (a, t) is a candidate AEP is decided by ILLOD solely on the basis of features of a and the words in t. Thus, it is a feature-based approach, although each feature is identified using conditional rules. Algorithm 1 specifies ILLOD in pseudo-code:

Algorithm 1: ILLOD

```
1 a^c = preprocess(a); t^c = preprocess(t);
2 if check_initial_letters(a, t) then
3 |   return True ;
4 else if check_initial_letters(a^c, t^c) then
5 |   return True ;
6 else if check_order(a^c, t^c) and compare_lengths(a^c, t^c) and
    check_distribution(a^c, t^c) then
7 |   return True ;
8 else
9 |   return False ;
```

The method $check\_initial\_letters(a, t)$ examines for all letters in a whether they correspond to the initial letters of the words in t. Thus, the calls in lines 2 and 4 check intuitively if the expansion fits the abbreviation, but have difficulties with pairs like ("QnA", "Questions and Answers"). To solve this, in line 6 additional features are evaluated:

$check\_order(a, t)$ examines if the order of the letters in a can also be found in t and if the initial letters of a and t correspond. Based on Schwartz and Hearst [31], we compare the letters in backward reading direction to favour an even distribution of the letters over the words of the expansion.

compare_lengths(a, t) checks whether the length (count of letters) of a is $\geq$ the number of words in t. This sorts out pairs like ("A", "Advanced Configuration and Power Interface"), based on the assumption that a should reference as many words in t as possible.

check_distribution(a, t) tests if the letters from a, if present in t, are uniformly distributed over the words in t, to sort out pairs like ("SMS", "Systems Network Architecture") or ("PaaS", "Palo Alto Research Center").

5.4 Evaluation of the Approaches on a Synthesized Data-Set

To estimate the accuracy of AEP detection approaches, a data-set D is needed that contains incorrect and correct AEPs. For this purpose, we compiled D as $D = L \cup S$, where L corresponds to the list from Section 4 and S consists of the pairs (a, e) in which a random element e from E was assigned to a given abbreviation a, not matching the real abbreviation of e. To be more formal, the set S can be described as $S = \{(a, e) \mid a \in A, e \in E, (a, e) \notin L\}$.

While $|L| = 1786$, S grows to $|S| = 2\,710\,125$. S could be reduced by filtering to pairs with identical initial letter. However, since L contains AEPs in which the initial letters differ, this option is discarded. Since we aim to test on a *balanced* data-set, where the proportion of abbreviations among all terms approximately corresponds to that in requirement texts, we test the presented approaches on different $D_\alpha = L \cup S_\alpha$, where $S_\alpha \subset S$ is randomly chosen from S each time, under the condition that $|S_\alpha| = \alpha * |L|$. To obtain an estimate for α, we extract 3195 NPs from 1102 requirements from ten projects of the *PURE* data-set [9]. To increase the recall, all words (not only words in NPs) are checked by our extraction rules from Section 4. In total, we extract 138 abbreviations and therefore estimate $\alpha = 3195/138 = 23.152$. Since requirement sets vary in use of abbreviations, several values for α (8, 16, 24, 48, 72) are considered. To avoid disadvantages for classifiers based on syntactic and semantic similarity, threshold values are F1-optimized for all α, given as *thold* in Table 3.

Table 3. F1 performance of AEP detection for different α. *Sem* (FT) corresponds to the semantic classifier in Section 5.1, *Syn* corresponds to the different variants of the syntactic classifier in Section 5.2 and *Feat* (ILLOD) corresponds to the feature-based classifier in Section 5.3. Best thresholds are given in the *thold* columns.
*Normalised *LD*: $LD^*(a, t) = 1 - (LD(a^c, potAbb(t^c))/max(|a^c|, |potAbb(t^c)|))$ [10]

		$\alpha = 8$		$\alpha = 16$		$\alpha = 24$		$\alpha = 48$		$\alpha = 72$	
Classifier		**F1**	***thold***	**F1**	***thold***	**F1**	***thold***	**F1**	***thold***	**F1**	***thold***
Sem	**FT**	0.287	*0.13*	0.191	*0.13*	0.146	*0.16*	0.088	*0.16*	0.064	*0.18*
Syn	**LD***	0.861	*0.55*	0.841	*0.54*	0.825	*0.52*	0.780	*0.70*	0.776	*0.68*
	JWS	0.874	*0.73*	0.849	*0.79*	0.831	*0.79*	0.800	*0.79*	0.778	*0.84*
	DC	0.841	*0.75*	0.811	*0.79*	0.789	*0.77*	0.746	*0.82*	0.723	*0.85*
Feat	**ILLOD**	**0.948**	-	**0.942**	-	**0.937**	-	**0.917**	-	**0.900**	-

The results in F1-scores summarized in Table 3 show that the FastText-based classifier performs poorly. This might be because a word embedding obtained from FastText can only inaccurately represent a certain word sense if the corresponding abbreviation has multiple expansions with heterogeneous meanings. Classifiers based on syntactic similarity measures have F1-scores between 72 and 87% — on average 80%, but are outperformed by ILLOD, which has between 90 and 94%. In the majority of cases, ILLOD achieves higher precision and recall at the same time. With increasingly larger α ($= 48, 72$), a weakening of the precision for ILLOD becomes apparent. Here, it is surpassed by the LD classifier. However, ILLOD is able to retain the best F1-score across all α thanks to its consistently high recall in particular. While Table 3 only states F1-scores, these more detailed results can be obtained within supplemental material [13].

5.5 Evaluation of the Approaches on a Requirements Data-Set

We evaluate the practicability of ILLOD for the intended use case with requirements from 15 projects comprised in a *PROMISE* [30] data-set [5,37]. In order to simulate their uncontrolled usage, 30 undefined abbreviations are inserted as replacements for written-out terms into various requirements. Only terms that appear in at least two requirements are abbreviated in at most one of those. No further guidelines for abbreviation are followed and different styles, not only acronyms, are used. This is performed by an independent person, not involved in this work and without the knowledge of the authors.

We read in the modified data-set as CSV-file. In independent runs, first abbreviations, as described in Section 4, and then *ordinary terms without undefined abbreviations* (OT), obtained through noun-chunk extraction [2], are gathered. In the next step, ILLOD is used to determine AEP-candidates by pairwise comparison of the abbreviations with the ordinary terms. The pairs created this way are then merged to *AEP groups*—clusters of exactly one abbreviation and all its potential expansions. For the modified PROMISE data-set, ILLOD creates 115 term tuples, combined to 51 AEP groups. Subsequently, this list of all determined AEP candidates is compared with the actual replacements. As a result, the extraction approach detects 29 of 30 inserted abbreviations and ILLOD is able to indicate the correct expansions for 25 of them.

We performed the same experiment with the semantic and syntactic classifiers. The results show that the other classifiers generate more than twice as many term tuples (AEP candidates) compared to ILLOD in order to indicate the correct expansion for fewer abbreviations—at maximum 22. Detailed results can be found again in the supplemental material [13].

6 Integration into Clustering Workflow

On the lines of Wang et al. [34], the preceding results confirm that different types of synonyms require different adapted approaches to calculating similarity, in particular for AEP-detection. Before we describe how AEP-specific methods

like ILLOD can be integrated into clustering of GTE tools, it is necessary to discuss how to ensure that the clusters created are meaningful and useful.

Arora et al. [2] create an ideal cluster solution from a given domain model against which the clusters obtained by different clustering algorithms have to be measured/compared/evaluated. We adopt this guiding principle in order to find a good strategy for the integration of ILLOD. We do not intend to evaluate different clustering algorithms, but rather to show how two already optimized clustering results—one for ordinary terms according to Arora et al. [2] and one for AEP groups—can be merged. To do so, some theoretical considerations on how ideal clusters can be constructed for this are required.

6.1 Ideal Clustering Solution

Arora et al. [2] create ideal clusters around a single concept c from the domain model, where the clusters also contain variants of terms that are conceptually equivalent to c and terms that are related to c according to the domain model.

Terms within individual AEP groups have a different relation to each other—indicating that two terms can be used as an expansion/definition for the same, as yet undefined, abbreviation. Thus, AEP groups differ in type from the ideal clusters of Arora et al. [2]. As AEP groups are designed to indicate probable ambiguities, they should not be separated in an ideal cluster solution.

As the ordinary terms within the individual AEP groups do not have to be conceptually related to each other according to the domain model, we must assume that they are distributed over the different clusters of the ideal clusters.

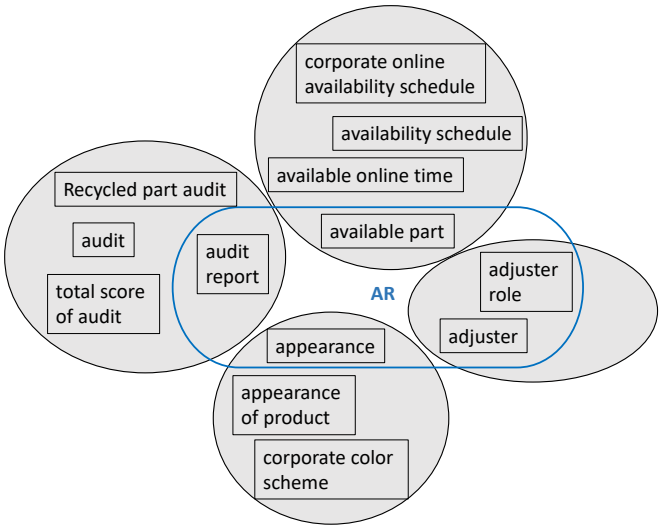

Fig. 1. Glossary term clusters of ordinary terms (grey) and overlay cluster for abbreviation *"AR"* and its possible expansions (blue) for a *"vehicle parts finder system"*.

This leads to the conclusion that AEP groups in an ideal cluster solution must be considered as so-called overlay clusters, which implies that the AEP groups are included as additional clusters. Fig. 1 shows an example for this, based on the requirements from a *"Vehicle Parts Finder System"* part of the PROMISE data-set [5] as project #5.

6.2 GTE Processing Steps

Considerations from the previous section lead us to propose the approach for the integration of ILLOD into a given GTE tool, as outlined in Fig. 2.

First, abbreviations are extracted from the given text, as described in Section 4 and then reduced to only consider yet undefined ones. Further, general glossary term candidates are extracted, e.g., through noun-chunking, and then cleaned from the abbreviations to a set of ordinary terms. ILLOD is then used to cluster abbreviations with their potential expansions into AEP groups, while a general synonym clustering approach, such as presented by Arora et al. [2], is used to cluster the ordinary terms.

As AEP groups are added into the final cluster solution in the last step, this will produce overlapping clusters. To evaluate the solutions generated by this approach, in addition to an ideal cluster solution, a metric is required to determine the score of agreement between an overlapping clustering solution and an overlapping ground truth—the ideal cluster solution. The *OMEGA-Index* Ω, a metric based on pair counts, introduced by Collins et al. [6], can achieve this.

Another argument for generating disjoint clusters of ordinary terms in the second-last step, besides the ones given by Arora et al. [2], is indicated by Ω. It shows the difficulty of making overlapping cluster solutions more similar to the ground truth clustering. For calculation, Ω uses the *contingency table* C. The entries $c_{i,j} \in C$ indicate the number of all pairs that appear in exactly i clusters in the solution and in exactly j clusters in the ground truth. A necessary condition to increase Ω between a generated cluster solution and a given ground truth is to modify the cluster solution so, that their agreement (sum of all diagonal values in C) is increased and their disagreement (sum of all values outside the diagonal) is decreased. Finally, an enlargement of the matrix would cause only a linear increase in the number of agreement fields, while the number of disagreement fields increases quadratically. Therefore, we propose the combination of disjoint clustering with separately calculated AEP group overlay clusters as introduced in Section 6.1.

7 Discussion

In the following, we discuss limitations and potential threats to validity [15] of our ILLOD approach to AEP detection, its evaluation, as well as considerations on its strengths and its integration to glossary term candidate clustering.

Repeatability We provide our source code and data-sets, as well as additional evaluation data [13] for replication.

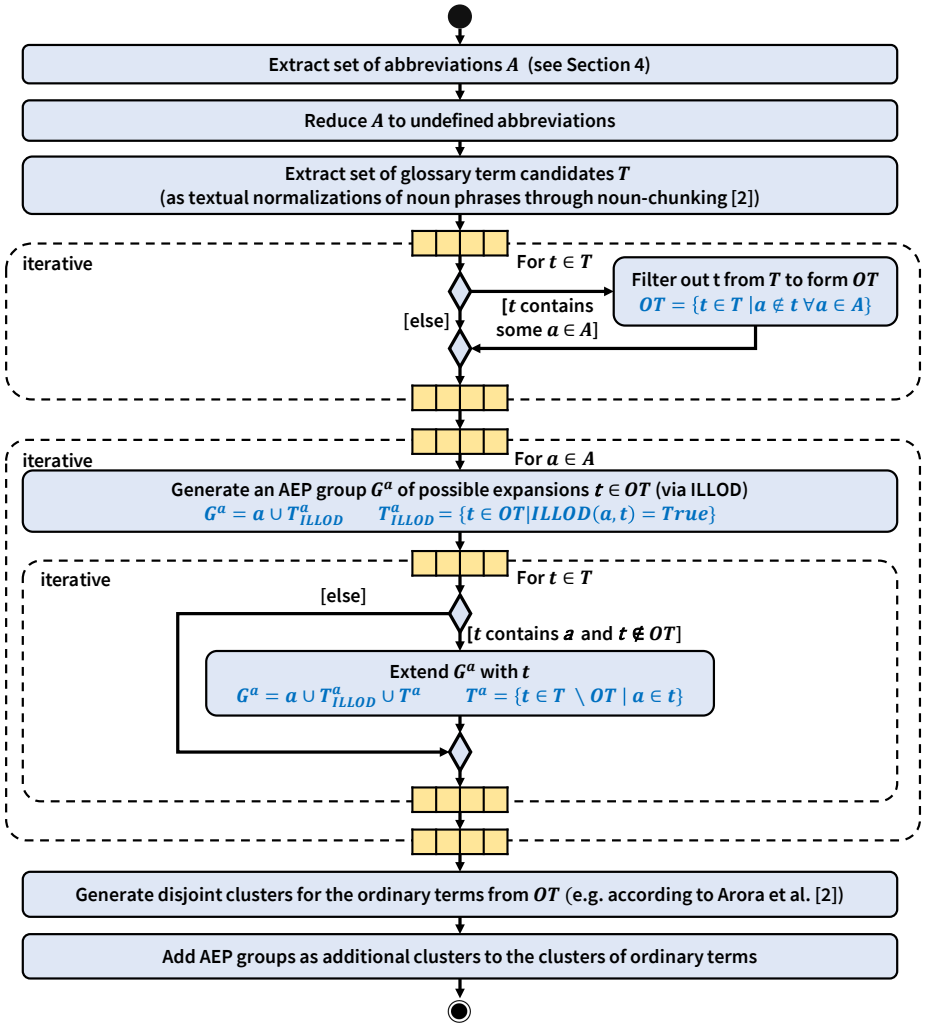

Fig. 2. Proposed workflow for the integration of ILLOD into a given GTE tool

Construct Validity Regarding (B1.1) threats are neglectable, as we directly work on extracted terms obtained via well known and reliable NLP techniques, and parameters for the identification of abbreviations are retrieved from real world examples and can be adjusted to fit domain specific peculiarities. Towards the more general (B1), homonyms and hyponyms are not detected explicitly. Yet, the analyst might be enabled to spot some during manual inspection of the clusters, although in general this problem needs to be addressed in a separate solution. However, focus of this work is on abbreviations, as defined in (B1.1).

Internal Validity To minimize the risk of threats, we tested the similarity measures and classifiers with the same cleaned list of defined abbreviations and under several portions α of abbreviations within the text. Semantic and syntactic similarity measures, are tested with different thresholds.

External Validity Parameters and features for abbreviation detection might be context and language specific. E.g., two examples from the German armed forces to abbreviate a unit of organization and an employment title exceed the limits of our detection: First, ("SABCAbwGSchAufg", "Schule ABC-Abwehr und Gesetzliche Schutzaufgaben") is with 15 letters longer than our limit of 13. Second, ("Schirrmstr", "Schirrmeister") has with 0.1 a too low portion of capital letters—this is presumably typical for simply truncated words. It shows that parameters need to be adjusted or some specific rules have to be added for domains with notably different guidelines. The list we used [7] is open community built without guidelines and thus heterogeneous abbreviation styles not limited to acronyms. Yet, it is domain specific. Further, we only used English terms. Parameters and accuracy might vary for other languages, e.g. in German rules for noun-splitting differ. However, parameters can be easily adapted through optimization on other data-sets. Similar, features evaluated by ILLOD can be easily adapted to domain specific patterns. Yet, the tests on the PROMISE data-set with requirements from 15 projects from different domains, indicate some general applicability. We plan to verify our approach on further data-sets in future research.

Conclusion Validity To mitigate threats, the modifications to the PROMISE data-set as well as the evaluation of detection results is conducted by an external independent person without exposure of details to the authors.

The considerations on cluster integration are based on related work [2] and initial experiments on optimization of different clustering algorithms with the OMEGA-Index. However, we plan to substantiate this in future experiments. Based on our findings, the proposed workflow has the following advantages:

(1) By using AEP groups, we avoid to decide which pair of terms belong together automatically, which is a challenging problem according to Jiang et al. [16].
(2) AEP groups have ergonomic as well as procedural advantages:
   (a) The analyst is motivated to build the list of abbreviations in parallel.
   (b) The analyst has direct insight into how an abbreviation could be expanded alternatively, as alternative expansions are likely to be encountered in the same cluster, and thus the analyst gets another opportunity to reduce ambiguities.
(3) Since the AEP groups are added to the generated cluster of ordinary terms in a post-processing step from a clustering point of view, the AEP groups ensure that unknown abbreviations and proposed expansions are placed in the same cluster, regardless of the clustering algorithm.
(4) Adding additional AEP groups lead to a final result with overlapping clusters, but mitigates the disadvantages of such, as these additional clusters are of different type than those of the ordinary terms.
(5) Using a feature-based approach to AEP detection, as ILLOD, provides high flexibility to adjust to domain specific rules, as new rules can easily be added.

We further conducted preliminary experiments with *hybrid* approaches to AEP detection, combining different types of classifiers. For example, to check the *initial letter equivalence* rule contained in ILLOD in a pre-processing step for all syntactic measures. This leads to increased accuracy for this type of classifier, as can be learned from the detailed evaluation data [13]. However, due to the nature of feature-based approaches of combining and potentially weighting different rules/features, it appears to be more plausible, to potentially integrate syntactic measures as additional rules here, rather than to outsource other features to excessive pre-processing.

8 Conclusions

Early glossary building and synonym detection is relevant to reduce ambiguity in requirements sets, e.g. through definition of preferred terms [14]. We demonstrate that different types of synonyms [14] need different treatments in detection. In particular, classical syntactic and semantic similarity measures perform poorly on abbreviations, as we show with our experiments in Section 5. With our ILLOD tool, we present a new feature based approach to AEP detection, which outperforms those classic approaches. It is also more flexible, as rule sets can be easily adapted to context specific characteristics, e.g. guidelines or other languages. Initial experiments indicate that investigation of hybrid approaches might be promising, though. We further propose how to integrate groups of abbreviations and their potential expansions to clusters of ordinary glossary term candidates as additional separate type of clusters.

This enables analysts to build the abbreviation list in parallel to the glossary and start this process early already on preliminary requirements. Further, we assume our approach not only to be relevant for early harmonization of requirements document terminology, but also if glossary and abbreviation list have to be built over several documents spanning multiple project phases and/or involved organizations and domains. In addition, different clusters for different synonym types can support the building of synonym groups for glossaries or thesauri with cross references [14] and context specific grouping as well as domain models.

References

1. Arora, C., Sabetzadeh, M., Briand, L., Zimmer, F.: Automated checking of conformance to requirements templates using natural language processing. IEEE Transactions on Software Engineering **41**(10), 944–968 (2015). https://doi.org/10.1109/TSE.2015.2428709
2. Arora, C., Sabetzadeh, M., Briand, L., Zimmer, F.: Automated extraction and clustering of requirements glossary terms. IEEE Transactions on Software Engineering **43**(10), 918–945 (2017). https://doi.org/10.1109/TSE.2016.2635134
3. Bhatia, K., Mishra, S., Sharma, A.: Clustering glossary terms extracted from large-sized software requirements using FastText. In: 13th Innovations in Software Engineering Conference, Formerly known as India Software Engineering Conference (ISEC'20). pp. 1–11 (2020). https://doi.org/10.1145/3385032.3385039

4. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics **5**, 135–146 (2017). https://doi.org/10.1162/tacl_a_00051
5. Cleland-Huang, J., Settimi, R., Zou, X., Solc, P.: Automated classification of non-functional requirements. Requirements Engineering **12**(2), 103–120 (2007). https://doi.org/10.1007/s00766-007-0045-1, http://ctp.di.fct.unl.pt/RE2017//downloads/datasets/nfr.arff
6. Collins, L.M., Dent, C.W.: Omega: A general formulation of the rand index of cluster recovery suitable for non-disjoint solutions. Multivariate behavioral research **23**(2), 231–242 (1988). https://doi.org/10.1207/s15327906mbr2302_6
7. Computer Hope: Computer acronyms and abbreviations, https://www.computerhope.com/jargon/acronyms.htm, visited on 10/16/2021
8. Dwarakanath, A., Ramnani, R.R., Sengupta, S.: Automatic extraction of glossary terms from natural language requirements. In: 21st IEEE International Requirements Engineering Conference (RE'13). pp. 314–319. IEEE (2013). https://doi.org/10.1109/RE.2013.6636736
9. Ferrari, A., Spagnolo, G.O., Gnesi, S.: Pure: A dataset of public requirements documents. In: 25th IEEE International Requirements Engineering Conference (RE'17). pp. 502–505 (2017). https://doi.org/10.1109/RE.2017.29
10. Gali, N., Mariescu-Istodor, R., Hostettler, D., Fränti, P.: Framework for syntactic string similarity measures. Expert Systems with Applications **129**, 169–185 (2019). https://doi.org/10.1016/j.eswa.2019.03.048
11. Gemkow, T., Conzelmann, M., Hartig, K., Vogelsang, A.: Automatic glossary term extraction from large-scale requirements specifications. In: 26th IEEE International Requirements Engineering Conference (RE'18). pp. 412–417. IEEE (2018). https://doi.org/10.1109/RE.2018.00052
12. Glinz, M.: A Glossary of Requirements Engineering Terminology. Tech. rep., International Requirements Engineering Board IREB e.V. (5 2014)
13. Hasso, H., Großer, K., Aymaz, I., Geppert, H., Jürjens, J.: AEPForGTE/ILLOD: Supplemental Material v(1.5). https://doi.org/10.5281/zenodo.5914038, https://doi.org/10.5281/zenodo.5914038
14. ISO: 25964-1: Information and documentation — Thesauri and interoperability with other vocabularies — Part 1: Thesauri for information retrieval. ISO (2011)
15. Jedlitschka, A., Ciolkowski, M., Pfahl, D.: Reporting Experiments in Software Engineering, pp. 201–228. Springer (2008). https://doi.org/10.1007/978-1-84800-044-5_8
16. Jiang, Y., Liu, H., Jin, J., Zhang, L.: Automated expansion of abbreviations based on semantic relation and transfer expansion. IEEE Transactions on Software Engineering (2020). https://doi.org/10.1109/TSE.2020.2995736
17. Justeson, J.S., Katz, S.M.: Technical terminology: some linguistic properties and an algorithm for identification in text. Natural language engineering **1**(1), 9–27 (1995). https://doi.org/10.1017/S1351324900000048
18. Kiyavitskaya, N., Zeni, N., Mich, L., Berry, D.M.: Requirements for tools for ambiguity identification and measurement in natural language requirements specifications. Requirements engineering **13**(3), 207–239 (2008). https://doi.org/10.1007/s00766-008-0063-7
19. van Lamsweerde, A.: Requirements engineering. John Wiley & Sons, Inc. (2009)
20. Larkey, L.S., Ogilvie, P., Price, M.A., Tamilio, B.: Acrophile: an automated acronym extractor and server. In: 5th ACM conference on Digital libraries. pp. 205–214 (2000). https://doi.org/10.1145/336597.336664

21. Merriam-Webster: What is an abbreviation?, https://www.merriam-webster.com/dictionary/abbreviation, visited on 10/17/2021
22. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
23. Miller, G.A.: WordNet: a lexical database for English. Communications of the ACM **38**(11), 39–41 (1995). https://doi.org/10.1145/219717.219748
24. Okazaki, N., Ananiadou, S.: A term recognition approach to acronym recognition. In: COLING/ACL'06 Main Conference Poster Sessions. pp. 643–650. ACM (2006)
25. Park, Y., Byrd, R.J.: Hybrid text mining for finding abbreviations and their definitions. In: Conference on empirical methods in natural language processing (2001)
26. Park, Y., Byrd, R.J., Boguraev, B.K.: Automatic glossary extraction: Beyond terminology identification. In: 19th International Conference on Computational Linguistics (COLING'02). vol. 1, pp. 1–7 (2002). https://doi.org/10.3115/1072228.1072370
27. Pohl, K.: Requirements Engineering. Springer (2010)
28. Pohl, K.: The three dimensions of requirements engineering. In: Bubenko, J., Krogstie, J., Pastor, O., Pernici, B., Rolland, C., Sølvberg, A. (eds.) Seminal Contributions to Information Systems Engineering: 25 Years of CAiSE, pp. 63–80. Springer (2013). https://doi.org/10.1007/978-3-642-36926-1_5
29. Pustejovsky, J., Castano, J., Cochran, B., Kotecki, M., Morrell, M.: Automatic extraction of acronym-meaning pairs from MEDLINE databases. In: MEDINFO'01. pp. 371–375. IOS Press (2001). https://doi.org/10.3233/978-1-60750-928-8-371
30. Sayyad Shirabad, J., Menzies, T.: PROMISE software engineering repository. School of Information Technology and Engineering, University of Ottawa, Canada (2005), http://promise.site.uottawa.ca/SERepository/
31. Schwartz, A.S., Hearst, M.A.: A simple algorithm for identifying abbreviation definitions in biomedical text. In: Biocomputing 2003, pp. 451–462. World Scientific (2002). https://doi.org/10.1142/9789812776303_0042
32. Sohn, S., Comeau, D.C., Kim, W., Wilbur, W.J.: Abbreviation definition identification based on automatic precision estimates. BMC bioinformatics **9**(1), 402–412 (2008). https://doi.org/10.1186/1471-2105-9-402
33. Song, M., Chang, P.: Automatic extraction of abbreviation for emergency management websites. In: 5th International Conference on Information Systems for Crisis Response and Management (ISCRAM). pp. 93–100 (2008)
34. Wang, Y., Manotas Gutiérrez, I.L., Winbladh, K., Fang, H.: Automatic detection of ambiguous terminology for software requirements. In: International Conference on Application of Natural Language to Information Systems. pp. 25–37. Springer (2013). https://doi.org/10.1007/978-3-642-38824-8_3
35. Yeganova, L., Comeau, D.C., Wilbur, W.J.: Identifying abbreviation definitions machine learning with naturally labeled data. In: 9th International Conference on Machine Learning and Applications. pp. 499–505. IEEE (2010). https://doi.org/10.1109/ICMLA.2010.166
36. Zhou, W., Torvik, V.I., Smalheiser, N.R.: ADAM: another database of abbreviations in MEDLINE. Bioinformatics **22**(22), 2813–2818 (2006). https://doi.org/10.1093/bioinformatics/btl480
37. Zou, X., Settimi, R., Cleland-Huang, J.: Improving automated requirements trace retrieval: a study of term-based enhancement methods. Empir Software Eng **15**(2), 119–146 (2009). https://doi.org/10.1007/s10664-009-9114-z, http://ctp.di.fct.unl.pt/RE2017//downloads/datasets/nfr.arff