Automatic Word Embeddings-Based Glossary Term Extraction from Large-Sized Software Requirements

Siba Mishra and Arpit Sharma(✉)

Department of Electrical Engineering and Computer Science,
Indian Institute of Science Education and Research, Bhopal, Madhya Pradesh, India
{sibam,arpit}@iiserb.ac.in

Abstract. [**Context and Motivation**] Requirements glossary defines specialized and technical terms used in a requirements document. A requirements glossary helps in improving the quality and understandability of requirements documents. [**Question/Problem**] Manual extraction of glossary terms from a large body of requirements is an expensive and time-consuming task. This paper proposes a fundamentally new approach for automated extraction of glossary terms from large-sized requirements documents. [**Principal Ideas/Result**] Firstly, our technique extracts the candidate glossary terms by applying text chunking. Next, we apply a novel word embeddings based semantic filter for reducing the number of candidate glossary terms. Since word embeddings are very effective in identifying terms that are semantically very similar, this filter ensures that only domain-specific terms are present in the final set of glossary terms. We create a domain-specific reference corpus for home automation by Wikipedia crawling and use it for computing the semantic similarity scores of candidate glossary terms. We apply our technique to a large-sized requirements document, i.e., a CrowdRE dataset with around 3000 crowd-generated requirements for smart home applications. Semantic filtering reduces the number of glossary terms by 92.7%. To evaluate the quality of our extracted glossary terms we manually create the ground truth data from CrowdRE dataset and use it for computing precision and recall. Additionally, we also compute the requirements coverage of these extracted glossary terms. [**Contributions**] Our detailed experiments show that word embeddings based semantic filtering can be very useful for extracting glossary terms from a large body of requirements.

Keywords: Requirements engineering · Natural language processing · Word embeddings · Term extraction · Semantic filter

1 Introduction

Requirements are the basis for every project, defining what the stakeholders in a potential new system need from it, and also what the system must do in order to satisfy that need [8,19]. All subsequent steps in software development are influenced by the requirements. Hence, improving the quality of requirements means improving the overall quality of the software product. A major cause of poor quality requirements is that the stakeholders involved in the development process have different interpretations of technical terms. In order to avoid these issues and to improve the understandability of requirements, it is necessary that all stakeholders of the development process share the same understanding of the terminology used. Specialized terms used in the requirements document should therefore be defined in a glossary. A glossary defines specialized or technical terms and abbreviations which are specific to an application domain. For example, if the system is concerned with health care, it would include terms like "hospitalization", "prescription drugs", "physician", "hospital outpatient care", "durable medical equipment", "emergency services", etc. Additionally, requirements glossaries are also useful for text summarization and term-based indexing.

In order to develop a glossary, the terms to be defined and added need to be first extracted from the requirements document. Glossary term extraction for the requirements document is an expensive and time-consuming task. This problem becomes even more challenging for large-sized requirements document, e.g., [16,17].

This paper focuses on automatic extraction of glossary terms from large-sized requirements documents. A first step in this direction is to extract the candidate glossary terms from a requirements document by applying text chunking. Text chunking consists of dividing a text in syntactically correlated parts of words. Since 99% of all the relevant terms are noun phrases [2,9], we only focus on extracting the noun phrases from a requirements document. Next, we apply a novel word embeddings based semantic filter to remove the noun phrases that are not domain-specific from the set of candidate glossary terms. Word embeddings are capable of capturing the context of a word and compute its semantic similarity relation with other words used in a document. It represents individual words as real-valued vectors in a predefined vector space. Each word is mapped to one vector and the vector values are learnt based on the usage of words. Words that are used in similar ways tend to have similar representations. This means that distance between two words which are semantically very similar is going to be smaller. More formally, the cosine of the angle between such vectors should be close to 1. To compute the similarity scores, we create a domain-specific reference corpus by crawling the home automation (HA) category on Wikipedia. The key idea is to use this corpus to check if the candidate glossary terms extracted by text chunking from a CrowdRE document are domain-specific or not. In other words, if a term in CrowdRE document has been used in a context which is different from the context in which it has been used in the domain-specific corpus for home automation then it needs to be removed from the final set of glossary terms.

We have applied our approach to the CrowdRE dataset [16,17], which contains about $3,000$ crowd-generated requirements for smart home applications. Our detailed experiments show that advantages of this new approach for glossary extraction go in two directions. Firstly, our filter reduces the number of glossary terms significantly. Note that this reduction is crucial for large-sized requirements documents. Secondly, the semantic nature of our filter ensures that only terms that are domain or application-specific are present in the final set of glossary terms. Note that in the case of statistical filtering such terms would be removed from the final set of glossary terms if they have low frequency of occurrence. To the best of our knowledge, this is the first time that a word embeddings based semantic filter has been proposed for automatic glossary term extraction from large-sized requirements documents.

1.1 Contributions

We propose an automated solution for extracting glossary terms from large-sized requirements documents. Our solution uses state-of-the art neural word embeddings technique for detecting domain-specific technical terms. More specifically, our main contributions are as follows:

- We extract candidate glossary terms by applying text chunking. Next, we propose a semantic filtering technique based on word embeddings to ensure that only terms that are truly domain-specific are present in the final set of glossary terms. This semantic filter is based on the principle that words that are used in similar ways tend to have similar representations.
- We apply our technique to the CrowdRE dataset, which is a large-sized dataset with around 3000 crowd-generated requirements for smart home applications. Our semantic filter reduces the number of glossary terms significantly. More specifically, we reduce the number of glossary terms in CrowdRE dataset by 92.7%.
- To measure the effectiveness of our technique we manually extract the glossary terms from a subset of 100 CrowdRE requirements and use this ground truth data for computing the precision and recall. We obtain a recall of 73.2% and a precision of 83.94%. Additionally, we also compute the requirements coverage of these extracted glossary terms.
- Finally, we discuss the benefits and limitations of word embeddings based semantic filtering technique for glossary extraction.

The remainder of the paper is structured as follows. Section 2 discusses the related work. Section 3 provides the required background. Section 4 explains our approach. We present the results and findings in Sect. 5. Finally, Sect. 6 concludes the paper and provides pointers for future research.

2 Related Work

Word Embeddings for RE. In [4], an approach based on word embeddings and Wikipedia crawling has been proposed to detect domain specific ambiguities in natural language text. More specifically, in this paper authors investigate the ambiguity potential of typical computer science words using a Word2vec algorithm and perform some preliminary experiments. In [5], authors estimate the variation of meaning of dominant shared terms in different domains by comparing the list of most similar words in each domain specific model. This method was applied to some pilot scenarios which involved different products and stakeholders from different domains. Recently, in [15], we have measured the ambiguity potential of most frequently used computer science (CS) words when they are used in other application areas or subdomains of engineering, e.g., aerospace, civil, petroleum, biomedical and environmental, etc. For every ambiguous computer science word in an engineering subdomain, we have reported its most similar words and also provided some example sentences from the corpus highlighting its domain specific interpretation. All these applications of word embeddings to requirements engineering are very recent and only focus on detecting ambiguity in requirements documents.

Glossary Extraction for RE. In [1,7], authors have developed tools, e.g. findphrases and AbstFinder for finding repeated phrases in natural language requirements. These repeated phrases have been termed as *abstractions*. These tools can be used as the basis for an environment to help organize the sentences and phrases of a natural language problem description to aid the requirements analyst in the extraction of requirements. In [18], authors have described an approach for automatic domain specific glossary extraction from large document collections using text analysis. A tool named GlossEx has been used to build glossaries for applications in the automotive engineering and computer help desk domains. In [10], authors described a case study on application of natural language processing for extracting terms from the text written by domain experts, and build a domain ontology using them. In [21], a term extraction technique has been proposed using parsing and parse relations. In this paper, authors have built a prototype dowsing tool, called Dowser which is capable of achieving high precision and recall when detecting domain-specific terms in a UNIX manual page. A text mining technique using term ranking and term weighing measures for the automatic extraction of the most relevant terms used in Empirical Software Engineering (ESE) documents has been proposed in [22]. In [23], authors have developed a procedure for automatic extraction of single and double-word noun phrases from existing document collections. Dwarakanath et al. [3] presented a method for automatic extraction of glossary terms from unconstrained natural language requirements using linguistic and statistical techniques. Menard et al. [12] retrieved domain concepts from business documents using a text mining process. Their approach has been tested on French text corpora from public organizations and shown to be 2.7 times better than a statistical baseline for relevant concept discovery. Recently, Arora et al. [2] have proposed a solution for automatic extraction and clustering of candidate glossary terms from natural language requirements. This technique has been evaluated on three-small sized industrial case studies. Note that in [2] syntactic, e.g., Jaccard, Levenstein, Euclidean and knowledge-based similarity measures, e.g., WUP, LCH, PATH have been used for clustering of glossary terms. More recently, a hybrid approach which uses both linguistic processing and statistical filtering for extracting glossary terms has been proposed in [6]. This technique has been applied to the same CrowdRE dataset which we have used in our paper for experiments.

All the above mentioned approaches can be broadly classified into linguistic, statistical or hybrid approaches. Linguistic approaches detect glossary terms using syntactic properties. In contrast, statistical approaches select terms based on the frequency of their occurrence. A hybrid approach combines both linguistic and statistical approaches, e.g., [6].

Our work proposes a new approach for selecting glossary terms based on the use of state-of-the art neural word embeddings technique. We use word embeddings and similarity scores to create a semantic filter which selects only those candidate terms that are truly domain-specific. We believe that this paper is a first step forward in the direction of developing advanced semantic filters for glossary term extraction from large-sized requirements documents.

3 Preliminaries

This section introduces some preliminaries that are needed for the understanding of the rest of this paper.

3.1 Word Embeddings

Word embeddings are a powerful approach for analyzing language and have been widely used in information retrieval and text mining. They provide a dense representation of words in the form of numeric vectors which capture the natural semantic relationship of their meaning. Word embeddings are considered to be an improvement over the traditional bag-of-words model which results in very large and sparse word vectors. Out of various word embedding models, the model "Word2vec" developed by the researchers at Google [13] has been used in our work to learn and generate word embeddings from a natural language text corpus. We focus on the model named skip gram negative sampling (SGNS) implementation of Word2vec [14]. SGNS predicts a collection of words $w \in V_W$ and their contexts $c \in V_C$, where V_W and V_C are the vocabularies of input words and context words respectively. Context words of a word w_i is a set of words $w_{i-wind}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+wind}$ for some fixed window size $wind$. Let D be a multi-set of all word-context pairs observed in the corpus. Let $\vec{\boldsymbol{w}}, \vec{\boldsymbol{c}} \in \mathbb{R}^d$ be the d-dimensional word embeddings of a word w and context c. These vectors (both word and context) are created by the Word2vec model from a corpus and are analyzed to check the semantic similarity between them. The main objective of negative sampling (NS) is to learn high-quality word vector representations on a corpus. A logistic loss function is used in NS for minimizing the negative log-likelihood of words in the training set. For more details, we refer the interested reader to [11,13,14]. As the input corpus changes, word embeddings are also updated reflecting the semantic similarity between words w.r.t. new corpus.

Word Similarity Computation. The Word2vec model uses the *cosine similarity* to compute the semantic relationship of two different words in vector space. Let us assume two word embedding vectors $\overrightarrow{\boldsymbol{w}'}$ and $\overrightarrow{\boldsymbol{w}''}$, where $\overrightarrow{\boldsymbol{w}'}$ is a word vector generated for CrowdRE and $\overrightarrow{\boldsymbol{w}''}$ is a word vector for home automation. The cosine angle between these two word embedding vectors is calculated using Eq. (1).

$$cos(\overrightarrow{\boldsymbol{w}'}, \overrightarrow{\boldsymbol{w}''}) = \frac{\overrightarrow{\boldsymbol{w}'} \bullet \overrightarrow{\boldsymbol{w}''}}{|\overrightarrow{\boldsymbol{w}'}||\overrightarrow{\boldsymbol{w}''}|} \tag{1}$$

The range of similarity score is between 0 to 1. A score closer to 1 means that the words are semantically more similar and used in almost the same context. On the other hand, a score closer to 0 means that the words are less related to each other.

3.2 Crowd-Generated Requirements

The CrowdRE dataset was created by acquiring requirements from members of the public, i.e., *crowd* [17]. This dataset contains about 3000 requirements for smart home applications. A study on Amazon Mechanical Turk was conducted with 600 workers. This study measured the personality traits and creative potential for all the workers. A two-phase sequential process was used to create requirements. In the first phase, user stories for smart home applications were collected from 300 workers. In the second phase, an additional 300 workers rated these requirements in terms of clarity and creativity and produced additional requirements.

Each entry in this dataset has 6 attributes, i.e., role, feature, benefit, domain, tags and date-time of creation. Since we are interested in extracting domain-specific terms from this dataset, we only focus on feature and benefit attributes of this dataset. An example requirement obtained from this dataset after merging feature and benefit attributes is as follows: "my smart home to be able to order delivery food by simple voice command, i can prepare dinner easily after a long day at work". For further details, we refer the interested reader to [16,17].

4 Approach

This section discusses the approach used to extract glossary terms from large-sized requirements documents. Figure 1 shows an overview of our approach. The first step includes the process of *data gathering*. In the second step we perform *data preprocessing*. The third step focuses on *extracting the candidate glossary terms* from preprocessed data. In the final step, *semantic filtering of candidate glossary terms* provides us the final set of domain-specific terms. The rest of this section elaborates each of these steps.

4.1 Data Gathering

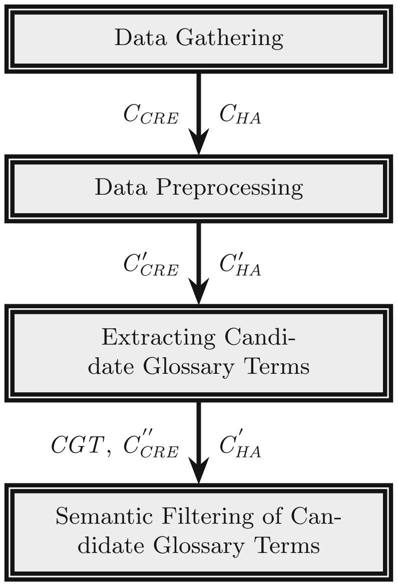
CrowdRE. For each user story in the CrowdRE dataset, we merge the feature and benefit attributes to obtain a single textual requirement. This is done by using a comma (,) between the text present in two attributes and a full stop (.) to terminate the requirement. Let C_{CRE} denotes the CrowdRE corpus obtained after applying the above mentioned transformations.

Requirements-Specific Reference Corpus. We use some standard web scraping packages available in python[1] to crawl and build the corpus of home automation domain. Let C_{HA} denotes the home automation corpus obtained by Wikipedia crawling. C_{HA} has been built by retrieving the web pages from "Wikipedia home automation" (HA) category[2], which has a tree structure. Wikipedia categories group together pages on similar subjects. Categories are found at the bottom of an article page. They support auto linking and multi-directional navigation. For our case, the maximum depth used for subcategory traversal is 2. This is primarily because increasing the depth results in extraction of less relevant pages from Wikipedia. For the sake of completeness, we have crosschecked all the results (data extraction for the home automation Wikipedia category) with the help of a widely used Wikipedia category data extraction tool known as PetScan[3]. PetScan (previously CatScan) is an external tool which can be used to find all the pages that belong to a Wikipedia category for some specified criteria.

4.2 Data Preprocessing

This step involves transforming raw natural language text into an understandable format. All the steps of data preprocessing have been implemented using the Natural Language Toolkit (NLTK)[4] in Python. The NLP pipeline used in data preprocessing is shown in Fig. 2. The textual data (sentences) of each corpus are broken into tokens of words (*tokenization*) followed by the cleaning of all special symbols, i.e., alpha-numeric words. Note that tokenization preserves the syntactic structure of sentences. Next, we convert all the words to lowercase (*lowering of words*) followed by the removal of noisy words defined for the English language[5] (*stop word removal*). The tokens of each sentence are tagged according to their syntactical position in the text. The tagged tokens are encoded as 2-tuples, i.e., (PoS, word), where PoS denotes the part of speech. We have used the NLTK (pos_tag)[6] Tagger, which is a perceptron tagger for extracting PoS tags. A perceptron part-of-speech tagger implements part-of-speech tagging using the averaged, structured perceptron algorithm. It uses a pre-trained pickled model

1 https://selenium-python.readthedocs.io/.
2 https://en.wikipedia.org/wiki/Home_automation.
3 https://petscan.wmflabs.org/.
4 https://www.nltk.org/.
5 https://www.ranks.nl/stopwords.
6 https://www.nltk.org/_modules/nltk/tag.html.



Data Gathering
C_{CRE} C_{HA}
Data Preprocessing
C'_{CRE} C'_{HA}
Extracting Candidate Glossary Terms
CGT, C''_{CRE} C'_{HA}
Semantic Filtering of Candidate Glossary Terms

Fig. 1. Semantic approach for glossary term extraction.

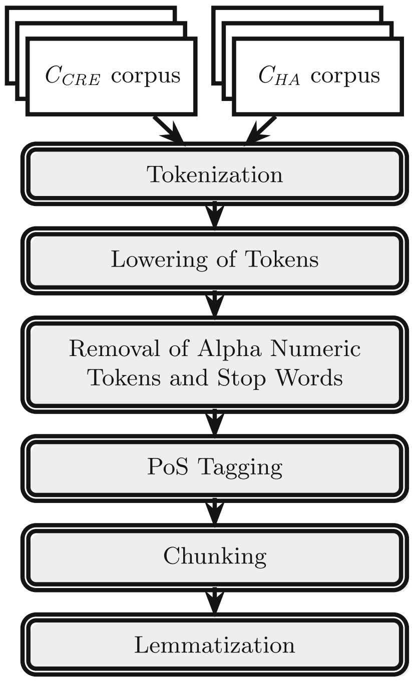


Fig. 2. NLP pipeline.

by calling the default constructor of the PerceptronTagger class[7]. This tagger has been trained and tested on the Wall Street Journal corpus. After extracting the tags, we apply text *chunking* which consists of dividing a text in syntactically correlated parts of words. Finally, we lemmatize[8] the generated chunks (*lemmatization*) which removes the inflectional endings and returns the base or dictionary form of a word, i.e., *lemma*. For example, after the lemmatization step *books* becomes *book* and *cooking* becomes *cook*. Let C'_{CRE} and C'_{HA} be the new corpora obtained after applying these steps. Lemmatization is important because it allows for the aggregation of different forms of the same word to a common glossary term.

4.3 Extracting Candidate Glossary Terms

Since we are interested only in noun phrases (NPs), let GT be the set of all lemmatized NPs obtained from C'_{CRE}. Similarly, NPs have been extracted from the Wikipedia corpus, i.e., C'_{HA}. Let TW be the set of all lemmatized NPs obtained from C'_{HA}. Finally, we compute the set of NPs on which the semantic

[7] https://www.nltk.org/_modules/nltk/tag/perceptron.html#PerceptronTagger.
[8] https://www.nltk.org/_modules/nltk/stem/wordnet.html.

filtering needs to be applied. In other words, we identify those NPs which are common to both GT and TW. Let $CGT = GT \cap TW$.

4.4 Semantic Filtering of Candidate Glossary Terms

After computing the set CGT, C'_{CRE} is transformed into a novel corpus C''_{CRE} by replacing each occurrence of a NP that appears in the set CGT with a modified version of the NP. This modified version is obtained by prefixing and suffixing the NP by an underscore character. For example, the word *system* is replaced by *_system_*. This transformation helps us in distinguishing the context of a given noun phrase. Continuing the previous example, the word *system* denotes that it is being used in the context of home automation, and *_system_* denotes that it is being used in the context of CrowdRE.

Next, we use the Word2vec model to produce word embeddings and for computing semantic similarity scores of NPs present in CGT. The goal of this step is to check if each noun phrase of CGT has been used in a similar context in both C''_{CRE} and C'_{HA} or not. Learning of word embeddings is facilitated by joining the two corpora, i.e, $C''_{CRE} \cup C'_{HA}$ which is given as an input to the Word2vec model. We set the dimension ($d = 100$), the window size ($wind = 10$), and the minimum count ($c = 1$) for all the experiments. Note that several rounds of experiments have been performed to identify the most suitable Word2vec parameters for this case study. As mentioned earlier, the Word2vec model uses cosine similarity to compute the semantic relationship of two words in vector space. The final set of glossary terms includes only those candidate terms which have a similarity score greater than or equal to 0.5. This value has been selected based on our experiments with the corpora.

Table 1. Descriptive statistics of the corpora.

Data	Type	Total size	Total sentences	Total chunks (NPs)
C_{CRE}	Corpus	2,966 (R)	2,966	4,156
C_{HA}	Corpus	1,196 (P)	64,25,708	64,480

5 Results and Discussions

This section presents the results of our detailed experiments. The semantic approach for glossary term extraction has been implemented in Python 3.7 and executed on Windows 10 machine with Intel Core-i5-7500 CPU, 4 GB DDR3 primary memory and a processor frequency of 3.40 GHz. The first row of Table 1 reports the number of CrowdRE requirements used in our experiments and total number of unique chunks (NPs) extracted from these requirements. Here, R denotes the textual requirements. The second row of this table reports the number of Wikipedia pages crawled for (HA) category to build the domain-specific reference corpus and total number of unique chunks extracted from this corpus. Here, P denotes the Wikipedia pages. The detailed report of our experiments including modified CrowdRE dataset, ground truth, final set of extracted glossary terms and similarity scores can be found in this repository[9].

Table 2. Some examples of manually extracted glossary terms.

Req Id	Textual requirements	Glossary terms
R1	My smart home to be able to order delivery food by simple voice command, i can prepare dinner easily after a long day at work	Smart home, order delivery food, simple voice command, voice command, dinner, day at work
R2	My smart home to turn on certain lights at dusk, i can come home to a well-lit house	Smart home, home, certain lights, light, dusk, house
R3	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, i can be relaxed	Smart home, biorhythm app, some music, music, mood, home, work
R4	My smart home to to ring when my favorite shows are about to start, i will never miss a minute of my favorite shows	Smart home, favorite show

5.1 Ground Truth Generation

Ground truth is used for checking the results of machine learning for accuracy against the real world. For glossary term extraction, ground truth generation involves manual creation of correct glossary terms by domain experts or by a team of experienced requirements engineers. Since CrowdRE dataset does not contain a reference list of correct glossary terms and it is not possible to create the correct glossary terms manually for a large body of requirements, i.e., 3000 requirements, we have manually created the ground truth for a subset of 100 requirements. This ground truth allows us to assess the performance of our approach by computing precision and recall. A total of 250 glossary terms have been manually extracted from a subset of 100 CrowdRE requirements. Note that this ground truth also includes the glossary terms (except role descriptions which do not appear in our subset of 100 requirements) that were manually selected by the authors in [6]. Some examples of manually extracted glossary terms have been shown in Table 2.

[9] https://github.com/SibaMishra/Automatic-Glossary-Terms-Extraction-Using-Word-Embeddings.

5.2 Precision and Recall

To evaluate the quality of our term extraction technique we compute precision and recall on this subset of 100 CrowdRE requirements. Precision gives us the fraction of relevant instances among the retrieved instances. On the other hand, recall gives us the fraction of relevant instances that have been retrieved over the total amount of relevant instances. As mentioned earlier, to compute the precision and recall we have manually extracted glossary terms from the subset of 100 requirements. We consider this set of manually extracted terms as ground truth. On applying our linguistic processing steps to these 100 requirements we extract 269 candidate glossary terms. Using our word embeddings based semantic filter, we reduce the number of glossary terms from 269 to 218. This set of final extracted glossary terms has 183 true positive terms. Note that we also count short terms that are included as parts of longer terms of the other set. These true positive terms lead to a recall of $\frac{183}{250} = 73.2\%$ and a precision of $\frac{183}{218} = 83.94\%$. These results indicate that our approach manages to strike a balance between the number of extracted glossary terms and recall rate.

5.3 Automated Glossary Term Extraction

Without applying our word embeddings based semantic filtering, the text chunking algorithm returns a total of 4156 candidate glossary terms when applied to the entire CrowdRE dataset, i.e., 2966 requirements. Since the number of glossary terms obtained is very large, we apply our semantic filter to reduce it by removing terms that are not domain-specific. Using our semantic filter we reduce the number of glossary terms significantly, i.e., from 4156 to 304. This means that the number of glossary terms gets reduced by 92.7%. Table 3 presents some

Table 3. Examples of final extracted glossary terms and their similarity scores.

Glossary terms	Similarity score
Automatic door	0.9161
Audio system	0.8517
Air conditioner	0.8042
Blood pressure monitor	0.8091
Comfortable temperature	0.9418
Entertainment system	0.8553
Electric blanket	0.9722
ipad	0.6262
Personal computer	0.6067
Smart light	0.9081
Smart alarm clock	0.9534
Smart card	0.5502

examples of glossary terms extracted by applying word embeddings based semantic filter. The second column of this table shows the similarity score, i.e., score obtained by computing the cosine similarity with the same word from the home automation Wikipedia corpus.

5.4 Coverage

In [6], requirements coverage has been advocated as another metric for a glossary's quality. Roughly speaking, the definition of coverage is the extent to which something is addressed, covered or included. In the context of glossary term extraction for software requirements, coverage gives us the percentage of requirements that are covered by the terms present in the glossary. It is important to note that high coverage rate does not necessarily means high quality glossary. For example, it is possible to achieve a very high coverage rate by including common words or terms that appear frequently in a requirements document even if they are not domain or application-specific terms. For CrowdRE dataset, without semantic filtering we obtain a total of 4156 glossary terms with a coverage rate of 99%, i.e., 2937 of 2966 requirements are covered by these glossary terms. On applying the semantic filter number of glossary terms reduces to 304 and the corresponding coverage rate is 51.88%, i.e., 1539 of 2966 requirements are covered by this new set of glossary terms. This reduction in coverage rate can be attributed to the following two reasons. Firstly, common nouns or noun phrases and terms which are not domain-specific but do appear frequently in requirements document would not be part of the final set of glossary terms obtained after applying semantic filter. Secondly, unlike [6] we do not include the content of role descriptions attribute of CrowdRE dataset as part of every requirement. Since the same role description can be part of many user stories it ensures a high coverage regardless of the specific content of the requirements. Some example role descriptions from CrowdRE dataset are as follows: student, cook, driver, parent, mother, wife, manager, adult, pet owner, nanny and husband. From these examples it is easy to see that role description does not give any useful information about the actual contents of a requirement. Semantic filtering reduces the number of glossary terms by 92.7% while coverage rate is reduced by 47.6%. In other words, the number of glossary terms gets reduced roughly by less than a factor of 14 whereas the coverage rate is reduced by less than a factor of 2. Since for a large-sized requirements document the glossary is required to be restricted to a manageable size, we believe that our approach is very effective in achieving a huge reduction in glossary size with a much smaller impact on coverage rate.

Some statistics related to coverage have been reported in Table 4. First row of this table presents the number of glossary terms that appear only once in the CrowdRE dataset. Similarly, i^{th} row of this table indicates the number of glossary terms that are present in i unique requirements from the CrowdRE dataset. As expected when i increases the number of glossary terms starts decreasing. The only exception to these findings is the 2^{nd} row of this table where the number of glossary terms increases.

Table 4. Number of requirements covered by the extracted glossary terms.

Number of requirements	Glossary terms
1	48
2	98
3	46
4	23
5	18
6	10
7	8
8	5
9	6
10	4

To highlight the fact that our approach does not remove infrequent domain or application-specific terms, we have compiled a set of some example technical terms which have been extracted by our semantic filter (see Table 5). First column of this table reports some examples of extracted glossary terms that appear only once in the CrowdRE dataset. Similarly, second column of this table includes some examples of extracted glossary terms which appear only twice in the dataset.

Table 5. Examples of extracted glossary terms that appear only once or twice in CrowdRE.

Glossary terms (1)	Glossary terms (2)
Smart sensor	Motion sensor
Fingerprint scanner	Smart fridge
Smart tag	Smart tv
Amazon	Ideal temperature
Wireless speaker	Room thermostat
Rfid chip	Facial recognition
Carbon monoxide detector	iphone

5.5 Advantages of Our Approach

A major advantage of our approach is that only those technical terms are added to the final set of glossary terms which are truly domain-specific. This is primarily because we use word embeddings and similarity scores for detecting domain-specific terms. For example, terms that occur very frequently in the requirements document but are not domain or application-specific will not be part of the final set of glossary terms. In contrast, term extraction approaches which only use statistical filtering would include these terms in the final set of glossary terms. Additionally, our filtering technique can be used to significantly reduce the number of glossary terms for large-sized requirements documents.

Another advantage of this approach is that if the number of glossary terms needs to be reduced further, this can be done easily by selecting a higher similarity threshold used for labeling a term as relevant/domain-specific. Our semantic filtering technique can also be combined with other filtering techniques, e.g., statistical and hybrid. Finally, this technique can be used to detect multiple terms (NPs) with the same meaning, i.e., synonyms. This is helpful as we do not need to define synonyms as separate glossary terms in the requirements document. Similarly, terms having the same spelling but different meanings, i.e., homonyms can be detected. This would allow us to define these terms as separate candidate glossary terms.

5.6 Limitations of Our Approach

A major limitation of our approach is that for every application domain it requires a corresponding domain-specific reference corpus which is used by Word2vec model to filter the candidate glossary terms. This could be an issue for new application domains where a reference corpus cannot be built due to the unavailability of large-sized domain-specific documents. Moreover, even for application domains where it is possible to create a reference corpus, there are no specific guidelines for selecting the source from where the corpus should be generated. For example, it is possible that for a particular application domain multiple sources of relevant data are available, e.g., Wikipedia, existing requirements documents, product brochures, handbooks, etc. In this case, it is not clear which of these documents need to be mined or crawled to generate the reference corpus which gives us the most accurate results. For home automation domain, reference corpus created by Wikipedia crawling gives us good results but this may not be true for other application domains.

Another issue with semantic filters is that for very large-sized documents, generation of word embeddings and computation of similarity scores for thousands of NPs may take a long time to complete. For home automation domain, we were able to run the experiments on a laptop but this may not be true for other application domains.

6 Conclusions and Future Work

This paper proposes an automatic approach for glossary term extraction from large-sized requirements documents. The first step of our solution extracts candidate glossary terms by applying text chunking. In the second step, we use a semantic filter based on word embeddings to reduce the number of glossary terms. This semantic filter ensures that domain-specific terms are not removed from the set of candidate glossary terms. We apply our technique to a large-sized requirements documents with around 3000 crowd-generated requirements. Our experiments show that word embeddings based semantic filtering can be very useful for extracting glossary terms from a large body of existing requirements. This research work can be extended in several interesting directions which are as follows:

– Implement a tool that takes as input the large-sized requirements document and automatically mines the Web to build a requirements-specific reference corpus. In the next step, it should automatically extract the set of candidate glossary terms by applying our word embeddings based semantic filter.
– Extend this technique to automatically extract the glossary terms from a large body of natural language requirements for software product lines [20].
– Compare the effectiveness of Word2vec semantic filter with other word embedding techniques, e.g., GloVe and fastText.
– Come up with some guidelines to determine how to create the domain-specific reference corpus.

References

1. Aguilera, C., Berry, D.M.: The use of a repeated phrase finder in requirements extraction. J. Syst. Softw. **13**(3), 209–230 (1990)
2. Arora, C., Sabetzadeh, M., Briand, L.C., Zimmer, F.: Automated extraction and clustering of requirements glossary terms. IEEE Trans. Softw. Eng. **43**(10), 918–945 (2017)
3. Dwarakanath, A., Ramnani, R.R., Sengupta, S.: Automatic extraction of glossary terms from natural language requirements. In: 21st IEEE International Requirements Engineering Conference (RE), pp. 314–319, July 2013
4. Ferrari, A., Donati, B., Gnesi, S.: Detecting domain-specific ambiguities: an NLP approach based on Wikipedia crawling and word embeddings. In: 25th IEEE International Requirements Engineering Conference Workshops (REW), pp. 393–399, September 2017
5. Ferrari, A., Esuli, A., Gnesi, S.: Identification of cross-domain ambiguity with language models. In: 5th International Workshop on Artificial Intelligence for Requirements Engineering (AIRE), pp. 31–38, August 2018
6. Gemkow, T., Conzelmann, M., Hartig, K., Vogelsang, A.: Automatic glossary term extraction from large-scale requirements specifications. In: 26th IEEE International Requirements Engineering Conference, pp. 412–417. IEEE Computer Society (2018)
7. Goldin, L., Berry, D.M.: AbstFinder, a prototype natural language text abstraction finder for use in requirements elicitation. Autom. Softw. Eng. **4**(4), 375–412 (1997). https://doi.org/10.1023/A:1008617922496
8. Hull, M.E.C., Jackson, K., Dick, J.: Requirements Engineering, 2nd edn. Springer, Heidelberg (2005). https://doi.org/10.1007/b138335
9. Justeson, J.S., Katz, S.M.: Technical terminology: some linguistic properties and an algorithm for identification in text. Nat. Lang. Eng. **1**(1), 9–27 (1995)
10. Kof, L.: Natural language processing for requirements engineering: applicability to large requirements documents. In: Workshop on Automated Software Engineering (2004)

11. Levy, O., Goldberg, Y.: Neural word embedding as implicit matrix factorization. In: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2. NIPS 2014, pp. 2177–2185 (2014)
12. Ménard, P.A., Ratté, S.: Concept extraction from business documents for software engineering projects. Autom. Softw. Eng. **23**(4), 649–686 (2015). https://doi.org/10.1007/s10515-015-0184-4
13. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
14. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2. NIPS 2013, pp. 3111–3119 (2013)
15. Mishra, S., Sharma, A.: On the use of word embeddings for identifying domain specific ambiguities in requirements. In: 2019 IEEE 27th International Requirements Engineering Conference Workshops (REW), pp. 234–240 (2019)
16. Murukannaiah, P.K., Ajmeri, N., Singh, M.P.: Toward automating crowd RE. In: 25th IEEE International Requirements Engineering Conference (RE), pp. 512–515, September 2017
17. Murukannaiah, P.K., Ajmeri, N., Singh, M.P.: Acquiring creative requirements from the crowd: understanding the influences of individual personality and creative potential in crowd RE. In: 24th IEEE International Requirements Engineering Conference (RE), pp. 176–185, September 2016
18. Park, Y., Byrd, R.J., Boguraev, B.K.: Automatic glossary extraction: beyond terminology identification. In: COLING 2002: The 19th International Conference on Computational Linguistics (2002). https://www.aclweb.org/anthology/C02-1142
19. Pohl, K.: Requirements Engineering - Fundamentals, Principles, and Techniques, 1st edn. Springer, Heidelberg (2010)
20. Pohl, K., Böckle, G., van der Linden, F.J.: Software Product Line Engineering: Foundations. Principles and Techniques. Springer, Heidelberg (2005). https://doi.org/10.1007/3-540-28901-1
21. Popescu, D., Rugaber, S., Medvidovic, N., Berry, D.M.: Reducing ambiguities in requirements specifications via automatically created object-oriented models. In: Paech, B., Martell, C. (eds.) Monterey Workshop 2007. LNCS, vol. 5320, pp. 103–124. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89778-1_10
22. Romero, F.P., Olivas, J.A., Genero, M., Piattini, M.: Automatic extraction of the main terminology used in empirical software engineering through text mining techniques. In: Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement. ESEM 2008, pp. 357–358 (2008)
23. Zou, X., Settimi, R., Cleland-Huang, J.: Improving automated requirements trace retrieval: a study of term-based enhancement methods. Empir. Softw. Eng. **15**(2), 119–146 (2010). https://doi.org/10.1007/s10664-009-9114-z