Automated Reuse Recommendation of Product Line Assets based on Natural Language Requirements [⋆]

Muhammad Abbas[1,2 (✉)] [0000-0001-6418-9971], Mehrdad Saadatmand[1] [0000-0002-1512-0844], Eduard Enoiu[2] [0000-0003-2416-4205], Daniel Sundamark[2], and Claes Lindskog[3]

[1] RISE Research Institutes of Sweden, Västerås, Sweden
{muhammad.abbas,mehrdad.saadatmand}@ri.se
[2] Mälardalen University, Västerås, Sweden
{muhammad.abbas,eduard.paul.enoiu,daniel.sundmark}@mdh.se
[3] Bombardier Transportation AB, Västerås, Sweden
claes.lindskog@rail.bombardier.com

Abstract. Software product lines (SPLs) are based on reuse rationale to aid quick and quality delivery of complex products at scale. Deriving a new product from a product line requires reuse analysis to avoid redundancy and support a high degree of assets reuse. In this paper, we propose and evaluate automated support for recommending SPL assets that can be reused to realize new customer requirements. Using the existing customer requirements as input, the approach applies natural language processing and clustering to generate reuse recommendations for unseen customer requirements in new projects. The approach is evaluated both quantitatively and qualitatively in the railway industry. Results show that our approach can recommend reuse with 74% accuracy and 57.4% exact match. The evaluation further indicates that the recommendations are relevant to engineers and can support the product derivation and feasibility analysis phase of the projects. The results encourage further study on automated reuse analysis on other levels of abstractions.

Keywords: software product line · reuse recommender · natural language processing · word embedding.

1 Introduction

With the increasing customization needs from customers, quality, and quick delivery of software products are of paramount importance. Meeting this demand requires an effective software engineering process. Software Product Lines (SPL/PL) [21] are created to help achieve quick delivery of quality products by systematically reusing features across variants. These PL features satisfy a set

⋆ This work is funded by the ITEA3 XIVT [25], and Knowledge Foundation's ARRAY Projects.

of standard requirements in a particular domain and are realized by reusable assets. Variations in the assets are introduced to cope with varying customization requirements of the same product.

A common industrial practice to the adoption of SPL is through the incremental development of overloaded assets, which are reused in a clone-and-own manner (e.g., in the railway industry [1]). While clone-and-own reuse is generally not recommended in SPL engineering, it does have some benefits, e.g., the reuse is very high speed with little domain engineering, needs less coordination, and has less adoption cost. In such cases, in the derived products, some assets are reused as-is, while for some, a copy of the asset is modified to address the particular customer requirements. This way of working results in many functional variants of the assets. Companies following clone-and-own based reuse practices faces problems among others, when implementing a new requirements, it might be required to know a) if a similar requirement has already been implemented by a PL asset or its functional variant, and b) if that is not the case, which asset or its functional variant is the closest one that can be modified to realize the new requirement. To achieve the aforementioned objectives, a reuse analysis is performed.

A reuse analysis process may have the following activities (as per [13, 12], summarized in [19]). Note here that we modified the activities to match our industrial partner's practices in the context of a PL: (1) identify high-level system functions that can realize the new customer requirements, (2) search existing projects to shortlist existing owned assets (that implements the system functions), (3) analyze and select from the shortlisted PL assets, and (4) adapt the selected assets to new customer requirements. The current (requirements-level) process for reuse analysis lacks automated support, is time-consuming, and is heavily based on the experience of engineers. Our approach aims to support the first three steps.

We support the reuse analysis and recommendation of PL assets early at the requirements level. The approach proposed in this paper is motivated by an industrial use case from the railway domain, where the same PL engineering practices (clone-and-own based reuse via overloaded assets) are followed. In our partner's company, requirements are written at various levels of abstraction (i.e., customer level, system level, subsystem level, SPL Assets description). Compliance with safety standards requires our partner to maintain the links between customer requirements and PL asset descriptions realizing them. Unfortunately, most of the existing reuse approaches (e.g., [6, 15, 2]) do not make use of this information (existing cases) to recommend reuse of PL assets and their functional variants. In addition, many existing approaches are limited to recovering traces between one requirement abstraction level sharing term-based or semantic similarity. In our case, the customer requirements share less semantic similarity with the asset descriptions.

In this paper, we propose an approach for requirement-level case-based reuse analysis and recommendation of (existing) PL assets. The approach is developed and evaluated in close collaboration with Bombardier Transportation, Sweden.

Our approach uses NLP techniques to clean the input customer requirements, train a word embedding model on the existing requirements, and cluster them. The trained model and clusters can then be used to generate recommendations (five at most, in our case) for PL assets (Simulink models in our case) reuse for new customer requirements.

Contributions. To this end, we make the following contributions: i) a proposed approach for the recommendation of PL asset reuse, ii) an evaluation of different word embedding algorithms on an industrial use case, and iii) a focus group evaluation to report engineer's view on the results.

Results obtained from this evaluation on two projects (derived products of the Power Propulsion Control (PPC) system's PL) show that our approach can recommend reuse of PL assets given unseen customer requirements with 74% accuracy [4] and 57.4% of exact match percentage [5]. The results obtained using thematic analysis on the focus group transcript show that our approach's recommendations are highly relevant and useful for practitioners.

Structure. The remainder of the paper is structured as follows. Section 2 presents our proposed approach for reuse analysis and recommendation. Section 3 evaluates the proposed approach and state-of-the-art word embedding algorithms in the context of reuse, Section 3.1 discusses the results obtained from the evaluation and the focus group, Section 3.2 discusses threats to validity. Section 4 presents the related work, Finally, Section 5 concludes the paper.

2 Approach

Our approach supports the reuse analysis of PL assets at the requirements-level. The approach has three distinct phases (shown in Figure 1), namely `Pre-Process`, `Training` and `Asset Reuse Recommender`. The first step of the approach (`Pre-Process`) is responsible for cleaning the requirements text for the later steps. Cleaning requirements include the removal of stop words, Part-Of-Speech (POS) tagging, and lemmatization. `Training` takes in existing cleaned customer requirements with their links to assets, realizing them. Clean requirements are used to train a feature extraction model (word embedding) and produce meaningful vectors for the existing customer requirements. The derived vectors are clustered using unsupervised clustering. The `Asset Reuse Recommender` phase takes in unseen cleaned customer requirements and recommends PL assets that can be reused to realize them. This is achieved by inferring meaningful feature vectors for the new requirements, predicting cluster for vectors of new requirements in the existing clusters, retrieving closest neighbors' reuse links to the PL assets or their functional variants, and finally ranking the reused PL assets. To demonstrate our approach, we use a running example (`LOG4J2`) from the SEOSS 33 data-set [22]. The running example is a subset of issues to issues

[4] A reuse recommendation is accurate if the recommended list contains the ground truth.

[5] A reuse recommendation is an exact match if the recommended list contains the ground truth on top.

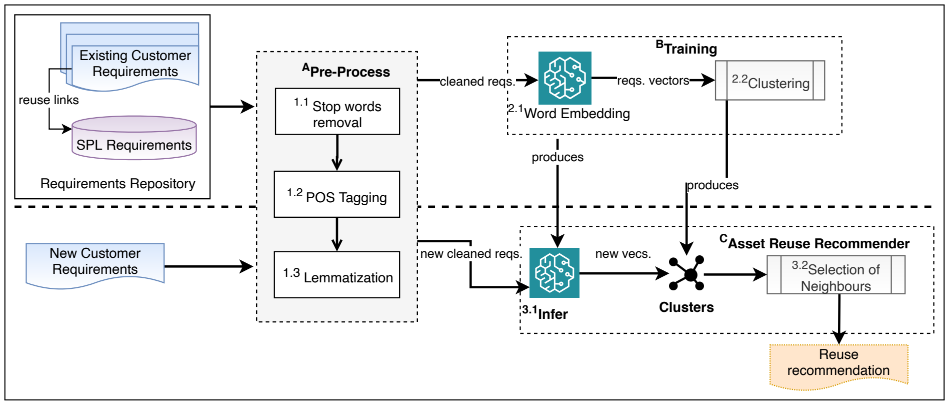

Fig. 1: An overview of the approach for reuse analysis and recommendation

Table 1: The Running example of LOG4J2 and its corresponding issue text

ID	Issue Text	Target
1292	1274 added the encode(LogEvent, ByteBufferDestination) method to Layout. Appenders implementing the ByteBufferDestination interface can call Layout.encode. This enables Layout implementations to process LogEvents without creating temporary objects. This ticket is to track the work for letting RandomAccessFileAppender (and its Rolling variant) implement the ByteBufferDestination interface and call Layout.encode instead of Layout.toByteArray.	1274
1291	1274 added the encode(LogEvent, ByteBufferDestination) method to Layout. The default...	1274
1305	Logging in a binary format instead of in text can give large performance improvements. ...	1397
1424	In non-garbage-free mode, the logged Message objects are immutable and can be simply...	1397
1517	Add ThreadContext.setContext(Map$< String, String >$). Note that we ...	1516
1519	Add API ThreadContext.putAll(Map$< String, String >$). My immediate goal is to be able...	1516
1349	The current ThreadContext map and stack implementations allocate temporary objects....	1516

data-set shown in Table 1. The `Issue Text` column can be mapped (to our data-set) as customer requirements, and the `Target` column can be mapped as PL assets. Note that the running example is just for demonstration purposes and contains implementation details unlike requirements. In this section, we discuss each step of our approach in more detail.

A. Pre-Process The pre-processing step of our approach is responsible for cleaning the text of requirements. The approach makes no assumption on the structure of the requirements. However, we do assume that the input requirements are written in English. The input to this step is processed following the steps below:
1.1 Stop words removal. Removal of language-specific stop words is important since most of the NLP models expect clean input. We use the spaCy[6] library for tokenizing the requirements and removing the English stop words from the text of the requirements. We also remove some of the domain-specific stop-words (e.g., system).
1.2 POS Tagging. Utilizing the full features of our pre-process pipeline, we also tag each of the tokens with their POS tags to guide lemmatization.
1.3 Lemmatization. We use the pre-trained English model from spaCy for the

[6] spaCy: Industrial-Strength NLP, https://spacy.io/

Table 2: Pre-processed text of issue 1292

Text: 1274 add encode(logevent bytebufferdestination method layout appender implement bytebufferdestination interface layout.encode enable layout implementation process logevent create temporary object ticket track work let randomaccessfileappender roll variant implement bytebufferdestination interface layout.encode instead layout.tobytearray
Vector: $< 0.66987733, -0.12908074, 0.01017888, 0.01252554 >$

lemmatization of the requirements text. This step is necessary in order to avoid different interpretations of the same word in other forms. First row of Table 2 shows the text of issue 1292 after pre-processing.

B. Training The training step expects clean requirements, and their reuse links to the PL assets realizing them. Each of the cleaned requirement is a training sample for the word embedding model. After training the word embedding model, the step produces vectors for the training set and a model that can be used to infer vectors for unseen requirements. The vectors obtained from the training set are given as an input to the unsupervised clustering algorithm. The unsupervised clustering algorithm (K-Means) is fitted to the input vectors, and the vectors are iteratively clustered to n number of clusters (calculated using the elbow method). The clustering step produces clusters and a model for later use. The running example (other than issue 1517) is used for the training phase.

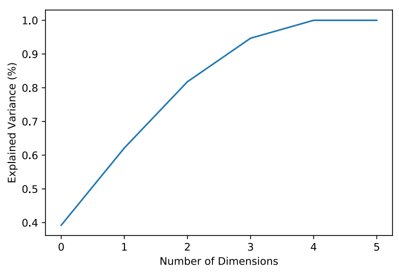

Fig. 2: Selected dimensions Vs. explained variance

2.1 Word Embedding is a set of feature extraction and learning approaches used to extract numeric vectors from raw text. This is done by mapping the words and phrases of the requirements into vectors of real numbers. This allows the application of complex mathematical operations (such as Euclidean distance calculation) on the vectors representing the requirements. The effectiveness of the resultant pipeline is heavily dependent on the choice of the word embedding method. Thus we also presented an evaluation of different pipelines on an industrial data-set. Three different word embedding methods (and their variants) are supported by our approach and can be selected by the end-users. We presented each word embedding method supported by our approach below:

Table 3: Clusters for the running example

Cluster	Members
1	1349 (linked to 1516), 1424 (linked to 1397), 1519 (linked to 1516)
2	1291 (linked to 1274), 1292 (linked to 1274), 1305 (linked to 1397)

Term Document matrix-based word embedding. Our approach's default word embedding is based on the term co-occurrence matrix called Term Frequency Inverse Document Frequency (TFIDF). TFIDF vectors mostly contain redundant features, contributing to the high dimensionality of the vectors. We apply a dimensionality reduction technique called Principal Component Analysis (PCA) to remove the redundant and co-related features from the vectors. The resultant vectors are considered as a final output for clustering. The vectors generated for the running example are of 1051 dimensions. After applying PCA (as shown in Figure 2), 1047 features can be dropped since those features do not contribute to the explained variance. Resultant vector for the issue 1292 is shown in Table 2.

Neural Network-based word embedding. Our approach also supports vectors from the state-of-the-art Doc2Vec algorithm [14]. The Doc2Vec is designed for learning paragraph vectors and can be later used for inferring vectors for unseen paragraphs. Another neural network-based word embedding algorithm supported by our approach is the FastText [3]. FastText is another neural network-based approach for learning word vectors, utilizing character-level information. This makes FastText an ideal choice for domain-specific NLP tasks (such as word embedding for requirements). Note that vectors obtained from neural network-based word embedding usually do not require dimensionality reduction.

2.2 Clustering existing (vectors of) customer requirements aid the case-based recommendation process for the PL asset's reuse. This is done by predicting clusters for new customer requirements in the existing clusters, and the nearest neighbors' top reused PL assets are recommended for reuse. We use the K-Mean algorithm to cluster the vectors of existing customer requirements iteratively. While clustering the existing requirements, we also keep track of the links to the reused PL assets. This step produces clusters containing the vectors of existing requirements. The produced clusters are stored for later use. The running example is clustered into two clusters, shown in Table 3.

C. Asset Reuse Recommender In this final step, the new customer requirements are cleaned following the same pre-process pipeline. The cleaned customer requirements are given as an input to the word embedding model obtained from the training phase. The model infers (computes) a vector for each customer requirements using the learned policy. The computed vectors for all new requirements are given as an input to the K-Means model to predict clusters for the new requirements' vectors in the existing clusters. For each new requirement, the existing closest customer requirement from the predicted cluster is selected, and their reused PL assets are retrieved. The retrieved PL assets (five at most)

Table 4: Generated Reuse recommendations

(a) For the running example

Input: 1517		
Reuse	**Sim. Score**	**Based on**
1516	0.84	1519
1397	0.06	1424

(b) For one industrial requirement

Input: CUS-REQ-450		
Reuse	**Sim. Score**	**Based on**
PL-1249	0.988	CUS-REQ-449
PL-1252	0.939	CUS-REQ-451
PL-906	0.901	CUS-REQ-426
PL-1069	0.879	CUS-REQ-409
PL-1333	0.622	CUS-REQ-377

are then ranked based on the similarity between their source vectors (existing requirements) and the new vector (new requirement). Note that the approach recommends multiple PL assets because one requirement can be satisfied by one or many PL assets. For the running example, issue 1517 (predicted cluster, in this case, is 1) is given as an input to the `Asset Reuse Recommender`, and the output is shown in Table 4a. The `Reuse` column shows the recommended PL assets based on similarity (`Sim. Score`) with the existing customer requirements shown in `Based on` column. The actual (ground truth) reused PL asset is shown in italic in Table 4a. Table 4b shows a real run of the approach on an industrial case.

3 Evaluation

This section presents the evaluation of our approach on a data-set from the railway industry in detail. From a high-level view, we started the evaluation by considering the following four research questions.

RQ1. Which word embedding algorithm produces the most accurate results in the context of PL asset reuse recommendation? We investigate the accuracy of different word embedding algorithms for PL asset reuse recommendation on a real industrial data set.

RQ2. Are pre-trained word embedding models suitable in the context of PL asset reuse recommendation? Since neural network-based word embedding models require large amounts of data for training (which might not be available), we investigate if pre-trained models can be used for reuse recommendations.

RQ3. What is the execution time of different pipelines for reuse recommendations? We aim at identifying the most efficient pipeline in terms of the time it takes to produce recommendations.

RQ4. What are the potential benefits, drawbacks, and improvements in our approach in an industrial setting? This research question captures the engineers' qualitative view on the results of the approach.

Implementation. All the pipelines (shown and discussed in sub-section 3) are implemented in Python 3. The pipelines are configured to process spreadsheets

Table 5: Summary of the pre-processing

Reqs.	**Before**		**After**	
-	**Words**	**AVG. Words**	**Words**	**AVG. Words**
188	4527	24.079	2421	12.877

containing requirements and their reuse links exported from the requirements management tool. For neural network-based word embedding algorithms, we used the Gensim implementation [23], and for TFIDF word embedding and clustering, we used sci-kit learn implementation [20].

Data Collection and Preparation. We used two recently deployed industrial projects (derived products of the PPC product line at the company). The SPL is actively developed in the safety-critical domain by a large-scale company focusing on the development and manufacturing of railway vehicles. A manual reuse analysis was already performed on the customer requirements of these two projects, and therefore the data set contains the ground-truth.

We selected a relevant subset of requirements out of the requirements available in the two documents. We removed the non-requirements (explanation text and headings) inside both documents. The rest of the requirements were further filtered out by excluding 78 requirements having reuse links to a PL asset that is not reused by any other requirement in the data-set. A final set of 188 requirements was reached. The application of the above criteria was necessary to ensure that the training phase uses suitable data. The PL reuse frequency in the date-set is spanning between 2 and 12. The text of the 188 selected requirements was passed through our pre-process pipeline (outlined already in Figure 1). Table 5 reports the total number of words and an average number of words per requirement both before and after pre-processing.

Besides, we conducted a two hours face-to-face focus group session. A focus group instrument was developed containing three topical questions based on RQ4. We recruited a convenience sample of individuals affiliated with our industrial partner's organization. The participants in the focus group were five employees, all with more than ten years of experience. The participants work closely with requirements, bids, and product line engineering during their work hours. Note here that three of the participants were involved in the product derivation and requirements engineering activities in the selected projects. The interview was transcribed and then analyzed using Braun and Clarke's guidelines for thematic analysis [5]. Themes are high-level abstractions of the transcribed data. In our case, these themes were predefined and extracted from RQ4 (i.e., benefits, drawbacks, and improvements). We did a data-driven analysis for the actual thematic analysis without trying to make it fit into an existing theoretical framework. The transcription was coded independently by two authors to encourage diversity of codes, sub-themes, and themes.

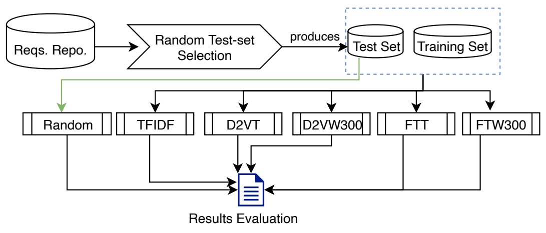

Fig. 3: Pipeline execution process for the quantitative evaluation

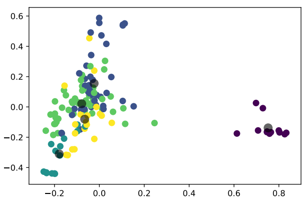

Fig. 4: Clustered requirements (training set) using K-Means

Evaluation Metrics for the Pipelines. We used the standard metric accuracy (A) and exact match percentage (E) for the evaluation of our pipelines. We randomly selected 25% of our data for validation. A recommendation is correct if the recommendations generated by the pipeline contains the ground truth. In our case, accuracy is calculated as the ratio between the total number of correct recommendations and total instances in the test set. In addition, we use a stricter evaluation metric (i.e., exact match percentage). This is calculated using the ratio between the number of exactly correct recommendations (where the ground truth is ranked on the top of the list of recommendations) and the total number of instances in the test set.

Procedure. To answer our first three research questions, we executed our approach with different word embedding algorithms and included both term document matrix-based and neural network-based word embedding algorithms. We also include a random recommender as a pipeline. Two neural network-based pre-trained models (trained on Wikipedia documents) are also included in our evaluation to answer RQ2. Each pipeline (other than random) is given the same randomly selected 75% of the 188 requirements as the training set and the rest of the 25% of the data is used to validate the pipeline. Figure 3 shows the execution process of the resultant pipelines. Due to randomness involved in the algorithms, each pipeline is executed 15 times.

All the pipelines were executed on an Apple MacBook Pro, 2018 with Intel Quad-Core i7 Processor (2.7 gigahertz) and 16 gigabytes of primary memory. We further discuss the execution setup for each pipeline in the remainder of this section.

The *Random* pipeline was configured to randomly generate (for each requirement in the test set) five unique reuse recommendations from the list of 50 PL assets.

The *TFIDF* pipeline was configured to the maximum (0.5) and minimum (6) term frequencies. We considered the n-gram ranging from 1 to 8. The TFIDF pipeline is configured to build the matrix on the training set. The produced vectors (of 2750 dimensions) from the training-set are reduced with PCA configured to automatically select the top features (i.e., 95% or more of the explained variance of the data is captured by the selected dimensions). The reduced vectors of

Table 6: Summary of pipelines validation

Stats.	Random		**TFIDF**		D2VT		D2VW300		FTT		FTW300	
%→	**A**	**E**	**A**	**E**	**A**	**E**	**A**	**E**	**A**	**E**	**A**	**E**
AVG.	11.73	2.24	**74**	**57.4**	10.71	2.24	65.2	41.45	56.53	43.30	59.38	44.35
SSD	5.01	2.93	3.87	4.26	4.59	1.67	3.72	3.42	4.80	4.75	6.25	5.25
VARS	25.17	8.60	14.99	18.21	21.10	2.81	13.86	11.71	23.07	22.57	39.12	27.57

Table 7: Average time taken for pipelines (RQ3)

Stats.	PP	Ran.	**TFIDF**		D2VT		D2VW300		FTT		FTW300		C	RG
-	-	-	**T**	**I**	**T**	**I**	**L(s)**	**I**	**T**	**I**	**L(s)**	**I**	-	-
AVG.	12.94	0.08	5.55	0.62	18.5	0.04	4.04	3.55	67.04	0.70	261.9	1.46	1.55	17.6
SSD	0.44	≈0	0.33	0.06	5.36	≈0	0.49	0.98	2.63	0.02	2.60	0.33	1.09	1.12
VARS	≈0	≈0	0.11	≈0	28.75	≈0	0.24	0.97	6.92	≈0	6.76	0.10	1.20	1.25
TT	-	-	**38.26ms**		50.63ms		4.07s		99.8ms		261.9s		-	-

86 dimensions from the PCA are stored and clustered using K-Means preserving the reuse links to PL assets. We used the Elbow method to compute the number of clusters using the vectors obtained from the PCA. Five clusters are used in all pipelines. Figure 4 shows the K-Means clusters of the data-set (with cluster's centers in light black color), showing only two dimensions of the training vectors obtained from one run. For each requirement in the test-set, the vectors from the TFIDF are generated and are reduced by PCA. The reduced vectors are plotted in the existing clusters produced by K-Means (during training), and at-most five recommendations for reuse are generated by looking into the reuse links of closest neighbors.

For *Doc2VecTraining (D2VT) and Doc2VecWiki300 (D2VW300),* a model is trained with a vector size of 300. The model is configured to consider a minimum term frequency of two. A pre-trained Doc2Vec model from the Gensim data[7] is also considered for this evaluation. The model has a vocabulary size of 35556952, and a vector size of 300. The cleaned customer requirements are vectorized using both of these configurations.

For *FastTextTraining (FTT) and FastTextWiki300 (FTW300),* a model is trained with a vector size of 100. This model is configured to consider a minimum term frequency of one. We also considered a pre-trained model with a vector of size 300. The pre-trained model is trained on one million words from Wikipedia on the sub-words level. The obtained requirements vectors are clustered, and recommendations for reuse are generated.

3.1 Results and Discussion

To answer the stated research questions, we executed each pipeline 15 times using the same training and test set (selected randomly in each run). We also performed a focus group session with engineers. This section presents the answers to all of our research questions with essential bits in *Italic*.

[7] https://github.com/RaRe-Technologies/gensim-data

RQ1*: Accuracy Results.* Table 6 shows the average accuracy and exact match results (from 15 runs) for each pipeline (shown in Figure 3). The best accuracy and exact match ratio percentage are shown in bold text in Table 6. To summarize each run of each pipeline, we presented some of the descriptive statistics in the `Stats.` column. A sample standard deviation (SSD), and sample variance (VARS) is presented to give insights on the runs. *Automated reuse analysis using term document matrix-based pipeline (TFIDF) outperformed all other approaches in terms of accuracy (74%) and exact match percentage (57.4%).*

The second best word embedding algorithm (in terms of accuracy) in our context is the pre-trained Doc2Vec model (D2VW300). However, the exact match percentage score for the FastText pre-trained model is higher than the D2VW300 pipeline. It is also important to note that the Random pipeline outperformed the D2VT pipeline. This is because the neural network-based approaches require a huge data-set for learning. In cases where the data-set is small, *self trained models for word embedding should be avoided.* However, the FastText's self-trained (FTT) pipeline performed significantly better than the D2VT pipeline. This is because the FTT model utilizes sub-word information for learning, and this makes it more accurate than other self-trained pipelines.

RQ2*: Pre-Trained Models' Results.* Table 6 includes results from two pre-trained models (`D2VW300 & FTW300`). Our results suggest that the use of pre-trained models might be good in cases where the data-set is small. The second best pipeline for reuse recommendation is the pre-trained Doc2Vec pipeline (D2VW300). *Pre-trained models in automated reuse analysis produced more accurate results than self-trained models.* This is because of less data provided to the self-trained models. In many cases, transfer learning might be an ideal choice and is one of our future focuses.

RQ3*: Execution Time Results.* Table 7 shows the time taken by each pipeline. `Stats.` column shows average time (AVG.), Sample standard deviation of time (SSD), and sample variance of time (VARS). Pre-processing time (PP), each pipeline execution time, clustering time (C), and recommendation generation time (RG) are also shown in the table. All the values (except model loading) are the time taken values per requirement and are in milliseconds (ms). The time for model loading (L) is the time taken (in seconds) to load the pre-trained model. The time for the Random (Ran.) pipeline is the average time taken (ms) per recommendation. The total time (TT row) represents the total average time per requirement for the pipeline execution ($PP + Pipeline + C + RG$). *TFIDF is the most efficient pipeline (in terms of execution time per requirement) for PL asset reuse analysis.*

Note that the Random pipeline takes even less time than the TFIDF pipeline, but produces inaccurate results. Our results also suggest that the FastText based pipelines might take significantly more time than other pipelines. However, this does not limit the application of FastText based pipelines in practice (since the model is loaded once per run).

RQ4*: Focus Group Results.* To answer this research question, we performed a thematic analysis of the data obtained from the focus group. In the first 15 minutes, our approach with the TFIDF pipeline was presented and executed on the data-set. After the pipeline generated the recommendations for the test-set), the participants reviewed six cases (i.e., one exact match, one with ground truth ranked down in the list, and four incorrect cases). After the participants reviewed the recommendations and their similarity scores, the participants were asked to discuss how accurate these recommendations are and what are the potential benefits, drawbacks, and improvements in the use of our approach. The rest of this section presents the findings related to three themes (also in Table 8).

Table 8: Identified sub-themes for the main themes.

Theme	**Sub-Theme**
1. Benefits	1.1 Aid in the automation of PL reuse analysis.
	1.2 Relevant and useful recommendations.
	1.3 Quicker reuse analysis.
2. Drawbacks	2.1 Lack of proper documentation.
	2.2 Effective selection of the training set.
	2.3 Tool integration in development processes.
3. Improvements	3.1 Threshold tuning for similarity values.
	3.2 Careful consideration of cases for training.

Theme 1. Participants found that our approach is beneficial for automating the process of reuse analysis. Currently, this analysis is manual, and the use of a tool that recommends reuse can be very useful for quicker analysis. Several participants stated the pipeline could provide relevant and useful recommendations early on in the development process and can avoid redundancy of assets.

Theme 2. Several challenges has been identified during the focus group. According to the participants, the lack of proper documentation for how this approach works and the underlying algorithms used for recommendations can hinder the adoption of such a tool. Another challenge mentioned by the participants was the effective selection of the training set of requirements. When asked how the practitioners would integrate the reuse analysis method into their process setting, we got varying answers depending on the team setting. However, all practitioners saw the full potential of automated reuse analysis if integrated into their existing tool-chains.

Theme 3. After reviewing the six cases, the participants discussed the accuracy results. For two cases, the recommendations were considered correct, relevant, and useful. In four other cases, the recommendations given by our approach were considered wrong. Note here that for these cases, the similarity values were consistently low. Participants suggested that the pipeline should not recommend reuse in cases where the recommendations are based on a similarity value below a certain threshold. For one of these cases, our approach accurately recommended the PL asset, but participants stated that it was the wish of the customer to include a relevant alternative PL asset. In addition, participants recommended that the pipeline should not include such cases for training.

3.2 Validity Threats

In this section, we present the validity threats following guidelines proposed by Runeson and Höst [24].

The requirements-level similarity might not be the best predictor of actual software reuse. To mitigate the internal validity threat, we verified that requirements-level similarity could be used to recommend PL asset reuse. This initial verification was performed by the use of topic modeling, where we verified that requirements sharing common topics are indeed realized by common PL assets. We also reviewed the current process of reuse analysis at our industrial partner's company and found that requirements-level similarity plays a significant role in reuse analysis.

Our results are based on data from one company using a data-set of 188 requirements created by industrial engineers. To mitigate potential external validity threats, we based our evaluation on requirements from two different projects, originated from different sources. Even if the number of requirements can be considered quite small, we argue that having access to real industrial requirements created by engineers working in the safety-critical domain can be representative. More studies are needed to generalize these results to other systems and domains. In addition, our work is based on the assumption that traceability links are maintained between assets and requirements. This assumption limits the applicability of our approach.

Finally, we address the threats to the reliability of our results by providing enough details on the setup of each pipeline. Our results are also based on 15 runs of each pipeline to address the randomness involved in the process.

4 Related Work

The related work in this area can be classified into three different lines of research. This section summarizes each class of related work.

Feature Extraction. Over the years, a huge amount of research has been focused on feature model extraction and feature recommendation for SPL [16]. Most of these approaches look for commonalities and variability to suggest features and extract feature diagrams. Public documents are being used for mining common domain terminologies and their variability(e.g., [9]). These approaches focus on aggregating natural language requirements to extract a high-level system feature model. Latent semantic analysis (LSA) is used to calculate the similarity between requirements pairs [27]. This approach clusters the requirements based on similarity and extracts feature trees. The Semantic and Ontological Variability Analysis (SOVA) [11] uses semantic role labeling to calculate behavioral similarity and generate feature model. In addition, bottom-up technologies for reuse (BUT4Reuse [17]) are used for reverse engineering variability from various artifacts.

Requirements Reuse. Another class of approaches [10] are focused on deriving and structuring a generic set of requirements that can be reused for PL and configured for derived products. For example, Zen-ReqConfig [15] uses models to aid in the structuring of requirements, reuse, and configuration at the requirements level. Arias et al. [2] proposed a framework for managing requirements that uses a defined taxonomy to aid the reuse of requirements in an SPL. Pacheco *et al.* proposed an approach for functional requirements reuse focused on small companies with no PL architecture [19]. In addition, Moon *et al.* [18] proposed a systematic method for deriving generic domain requirements as core reusable assets by analyzing the existing legacy requirements.

Traceability. Another set of approaches are focused on traceability link recovery [4]. These approaches recommend possible traceability links between different artifacts (e.g., requirements, source code). Yu *et al.* proposed an approach to recover traceability links between features and domain requirements [28]. The approach extracts and merge application feature models and establish their traceability links. Wang *et al.* proposed a co-reference-aware approach to traceability link recovery and also proposed an algorithm for automated tracing of dependability requirements [26]. IR-based approaches for traceability link recovery are well-known solutions to trace recovery (e.g., [7]. These approaches mostly uses term-document matrix-based similarity for tracing requirements.

The approaches classified in Feature Extraction & Requirements Reuse categories are focused on extracting feature models and domain requirements. On the other hand, the approaches included in the Traceability category are closely related to our approach since these are focusing on establishing the links between requirements and other artifacts. Compared to our work, the traceability approaches are not directly recommending the reuse of PL assets in a PL context but can be tailored for reuse recommendation. In addition, existing approaches also do not make use of existing cases for reuse reasoning. To the best of our knowledge, we are the first to support the reuse analysis of PL assets in a context of a clone-and-own reuse process.

5 Conclusion

In this paper, we proposed an automated approach for requirements-level reuse analysis of product line assets. The approach uses existing customer requirements to recommend possible PL assets that can be reused to implement new requirements. We evaluated our approach in the railway industry. The results show that the proposed approach was able to recommend reuse of PL assets with 74% average accuracy and 57.4% exact match ratio. We also presented an evaluation of five different pipelines with varying word embedding algorithms, which demonstrated that the document matrix-based word embedding algorithm performed significantly better than other pipelines. We also found that the self-training of the neural network-based word embedding algorithm should be avoided if the data-set is small. In such cases, transfer learning should be performed with existing pre-trained models. Our results also show that the Doc2Vec pre-trained model performed better than the FastText's pre-trained model. In terms of the practicality of the pipelines, we found that the maximum end-to-end execution time of the approach is around 262 seconds. The validation of the results in our focus group session with five engineers also confirmed the applicability of such pipelines in practice. In particular, results shows that the approach automates the reuse analysis with highly relevant reuse recommendations.

Our future work includes an empirical evaluation of requirements-level reuse recommenders. We aim to investigate the teams performing reuse analysis with and without a reuse recommender. Investigating Bidirectional Encoder Representations from Transformers (BERT [8]) model for requirement-level reuse is also one of our future focus.

References

1. Abbas, M., Jongeling, R., Lindskog, C., Enoiu, E.P., Saadatmand, M., Sundmark, D.: Product line adoption in industry: An experience report from the railway domain. In: 24th ACM International Systems and Software Product Line Conference. ACM (2020)
2. Arias, M., Buccella, A., Cechich, A.: A Framework for Managing Requirements of Software Product Lines. Electronic Notes in Theoretical Computer Science **339**, 5–20 (2018)
3. 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)
4. Borg, M., Runeson, P., Ardö, A.: Recovering from a decade: a systematic mapping of information retrieval approaches to software traceability. Empirical Software Engineering **19**(6), 1565–1616 (2014)
5. Braun, V., Clarke, V.: Using thematic analysis in psychology. Qualitative research in psychology **3**(2), 77–101 (2006)
6. Dag, J.N.o., Gervasi, V., Brinkkemper, S., Regnell, B.: A linguistic-engineering approach to large-scale requirements management. IEEE Softw. **22**(1), 32–39 (Jan 2005)
7. De Lucia, A., Oliveto, R., Tortora, G.: Adams re-trace: Traceability link recovery via latent semantic indexing. In: International Conference on Software Engineering. p. 839–842. ICSE '08, ACM, New York, NY, USA (2008)
8. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
9. Ferrari, A., Spagnolo, G.O., Orletta, F.D.: Mining Commonalities and Variabilities from Natural Language Documents. In: International Software Product Line Conference. pp. 116–120. ACM, Tokyo, Japan (2013)
10. Irshad, M., Petersen, K., Poulding, S.: A systematic literature review of software requirements reuse approaches. Information and Software Technology **93**(September 2017), 223–245 (2018)
11. Itzik, N., Reinhartz-Berger, I., Wand, Y.: Variability Analysis of Requirements: Considering Behavioral Differences and Reflecting Stakeholders' Perspectives. IEEE Transactions on Software Engineering **42**, 687–706 (2016)

12. Krueger, C.W.: Software reuse. ACM Computer Surveys **24**(2), 131–183 (1992)
13. Lam, W., McDermid, T., Vickers, A.: Ten steps towards systematic requirements reuse. In: International Symposium on Requirements Engineering. pp. 6–15 (1997)
14. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. CoRR **abs/1405.4053** (2014), http://arxiv.org/abs/1405.4053
15. Li, Y., Yue, T., Ali, S., Zhang, L.: Enabling automated requirements reuse and configuration. Software and Systems Modeling **18**(3), 2177–2211 (2019)
16. Li, Y., Schulze, S., Saake, G.: Reverse engineering variability from natural language documents: A systematic literature review. In: Proceedings of Software Product Line Consference. vol. 1, pp. 133–142. Sevilla, Spain (2017)
17. Martinez, J., Ziadi, T., Bissyandé, T.F., Klein, J., Traon, Y.L.: Bottom-up adoption of software product lines: a generic and extensible approach. In: International Conference on Software Product Line, SPLC 2015, Nashville, TN, USA, July 20-24, 2015. pp. 101–110 (2015)
18. Moon, M., Yeom, K., Chae, H.S.: An approach to developing domain requirements as a core asset based on commonality and variability analysis in a product line. IEEE Transactions on Software Engineering **31**(7), 551–569 (2005)
19. Pacheco, C., Garcia, I., Calvo-Manzano, J.A., Arcilla, M.: Reusing functional software requirements in small-sized software enterprises: a model oriented to the catalog of requirements. Requirements Engineering **22**(2), 275–287 (2017)
20. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
21. Pohl, K., Böckle, G., van Der Linden, F.J.: Software product line engineering: foundations, principles and techniques. Springer Science & Business Media (2005)
22. Rath, M., Mäder, P.: The SEOSS 33 dataset — Requirements, bug reports, code history, and trace links for entire projects. Data in Brief **25** (2019)
23. Řehůřek, R., Sojka, P.: Software Framework for Topic Modelling with Large Corpora. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. pp. 45–50. ELRA (May 2010)
24. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering **14**(2), 131–164 (2009)
25. Schlingloff, H., Kruse, P.M., Saadatmand, M.: Excellence in variant testing. In: Proceedings of the 14th International Working Conference on Variability Modelling of Software-Intensive Systems. VAMOS '20, ACM (2020)
26. Wang, W., Niu, N., Liu, H., Niu, Z.: Enhancing automated requirements traceability by resolving polysemy. In: International Requirements Engineering Conference, RE 2018. pp. 40–51. IEEE (2018)
27. Weston, N., Chitchyan, R., Rashid, A.: A framework for constructing semantically composable feature models from natural language requirements. In: International Software Product Line Conference. pp. 211–220 (2009)
28. Yu, D., Geng, P., Wu, W.: Constructing traceability between features and requirements for software product line engineering. In: Proceedings - Asia-Pacific Software Engineering Conference, APSEC. vol. 2, pp. 27–34. IEEE (2012)