Requirements Dependency Extraction by Integrating Active Learning with Ontology-Based Retrieval

Gouri Deshpande*, Quim Motger†, Cristina Palomares †, Ikagarjot Kamra‡, Katarzyna Biesialska†,
Xavier Franch†, Guenther Ruhe* and Jason Ho§
*dept. of Computer Science
University of Calgary, Calgary, Canada
Email: {gouri.deshpande, ruhe}@ucalgary.ca
†dept. of Service and Information System Engineering (ESSI)
Universitat Politècnica de Catalunya, Barcelona, Spain
Email: {jmotger, cpalomares, biesialska, franch}@essi.upc.edu
‡dept. of Electrical and Software Engineering
University of Calgary, Calgary, Canada
Email: ikagarjot.kamra1@ucalgary.ca
§Blackline Safety. Inc, Calgary, Canada
Email: jho@blacklinesafety.com

***Abstract*—Context: Incomplete or incorrect detection of requirement dependencies has proven to result in reduced release quality and substantial rework. Additionally, the extraction of dependencies is challenging since requirements are mostly documented in natural language, which makes it a cognitively difficult task. Moreover, with ever-changing and new requirements, a manual analysis process must be repeated, which imposes extra hardship even for domain experts.**

Objective: The three main objectives of this research are: 1) Proposing a new dependency extraction method using a variant of Active Learning (AL). 2) Evaluating this AL and Ontology-based Retrieval (OBR) as baseline methods for dependency extraction on the two industrial data sets. 3) Analyzing the value gained from integrating these diverse approaches to form two hybrid methods.

Method: Building on the general AL, ensemble and semi-supervised machine learning, a variant of AL was developed, which was further integrated with OBR to form two hybrid methods (Hybrid1, Hybrid2) for extracting three types of dependencies (requires, refines, other): Hybrid1 used OBR as a substitute for human expert; Hybrid2 used dependencies extracted through the OBR as an additional input for training set in AL.

Results: For two industrial case studies, AL extracted more dependencies than OBR. Hybrid1 showed improvement for both data sets. For one of them, $F1$ score increased to 82.6% compared to the AL baseline score of 49.9%. Hybrid2 increased the accuracy by 25% to the level of 75.8% compared to the AL baseline accuracy. OBR also complemented the AL approach by reducing 50% of the human effort.

***Index Terms*—Requirements dependencies, dependency extraction, active learning, domain ontologies, hybrid method, empirical evaluation, industrial data.**

I. Introduction

Requirement dependencies are an inherent characteristic of software development projects. Carlshamre et al. [7] estimated that about 80% of all requirements are interdependent. Also, a recent survey with practitioners revealed that ignoring dependencies has a detrimental effect on project success [14]. More than 80% of the participants of this survey agreed that dependency extraction is difficult and the implications of ignoring them have an adverse impact on the success of the project. Additionally, 90% of the participants did not use any type of automation to extract or maintain the dependencies [14]. Therefore, the extraction of dependencies is critical for the prioritization of requirements and effective release planning. Yet, requirements are mostly documented in natural language. Hence, the process of extracting requirement dependencies often entails manual effort, which is laborious, cognitively exhaustive and thus error-prone.

Automatic extraction of dependencies through requirements' textual content has recently been a focus of active research in areas as diverse as software testing [1], change analysis [51], requirements prioritization [41] and bug prediction [47]. Dependencies can be analyzed from different angles. For instance, the relationship between requirements can be characterized as basic. i.e., binary: dependent or independent, or described using more factors, such as direction, strength, precedence [5], [13], [39] etc.

In this research, we compare the two baseline methods that take a radically different approach to the requirements dependency extraction problem. The first method utilizes a domain ontology to extract the dependency pairs and their dependency type. The second method uses Ensemble-based Active and Semi-supervised labeling learning [25]. This approach selects a good set of requirement pairs to form a labeled training data set - an active learning approach. Also, it utilizes both labeled and unlabeled data - a semi-supervised approach. This solution aims to use a small labeled data set to achieve a good classifier with high accuracy.

We also propose and explore two different hybrid methods in this paper which could overcome major limitations of the baseline methods. The first hybrid method replaces the human-in-the-loop component of AL with an ontology-based solution.

The second uses dependencies extracted from the domain-specific ontology as training input to AL. These approaches are evaluated with two industrial data sets provided by two IT companies: Siemens, Austria and Blackline Safety Corp., Canada. In this paper, we refer to Siemens, Austria, as Company A and Blackline Safety Corp., Canada, as Company B.

The paper is organized as follows: Section II presents the background on the dependency extraction problem. Section III describes the research method, research questions, baseline methods, industrial data sets, and the design of the whole empirical study in detail. Section IV covers the empirical evaluation of the baseline methods. Section V introduces the two hybrid approaches and reports their evaluation. Section VI discusses the overall results of the study. Section VII details the threats to validity. Section VIII presents the related work. Finally, Section IX provides conclusions and outlines future work.

II. Background

Requirement dependencies establish a relationship between requirements such that "progress of action on one requirement assumes the timely outcome of action on another requirement or the presence of a specific condition" [2]. Some authors refer to them as "interdependencies" [10] [22] to clearly distinguish from relationships between requirements and other artefacts such as code or test suites. Although the study of requirement dependencies is framed in the Requirements Engineering (RE) discipline, other communities have targeted the problem extensively with respect to aspect identification [37] or extracting dependencies among features such as variability modelling [44] and architectural analysis [46].

Research in the area has a long history that studied dependencies related to both syntactic and semantic criteria. While some authors have proposed taxonomies focusing on requirements engineering in general [36] [10], others focused on application areas such as release planning [8]. For example, Dahlstedt [10] provided the classification of most fundamental dependency types such as Requires, Refines, Similar_to, Increases/Decreases_value_of etc. and classified them into Structural and Cost/Value interdependencies.

Recently, Zhang et al. [51] consolidated these taxonomies in the context of change propagation analysis and proposed a model that included nine types of dependencies. In this paper, we are particularly interested in two types of dependencies: *refines*, *requires*, whereas the remaining ones are labelled as *other*. Refinement was defined by Pohl's seminal proposal of taxonomy as "target object [*requirement*] is defined in more detail by another requirement" [36], whereas the requires relationship can be defined according to Carlshamre et al., as "R1 requires R2 to function" [8].

Not only academics but also software industry professionals have a voice on the topic. Deshpande et al. [14] report the results of a recent survey for requirements dependency extraction and maintenance, with 76% responses (out of 70) from practitioners. More than 80% of the participants agreed or strongly agreed that dependency type extraction is difficult in practice; dependency information has implications on maintenance, and ignoring dependencies has a significant impact on project success [14].

Previous studies have explored diverse computational methods that used natural language processing (NLP) [33], fuzzy logic [32], predicate logic [52] and deep learning [18] techniques (see Section VIII: Related work) in the past. By considering their diverse nature, strengths and weaknesses, it is natural to explore the option of combining them into hybrid methods to obtain better results. In this paper, we choose two approaches (elaborated in Section III-C):

- **Ontology:** Ontologies are widely used in RE research (see [11] for a survey). They are a well-suited conceptual artefact to manage knowledge. Particularly, domain ontologies [26] provide a formal representation of a specific domain serving as a means for communication and agreement. Hence, their use in activities such as dependency extraction is a reasonable choice to get domain-specific solutions.
- **Active Learning:** AL is a form of ML in which a learning algorithm interactively queries an *oracle* (typically a human expert) to obtain the desired label for new data points [40]. It has been effective in reducing human efforts in the data analysis process [15] [3]; therefore, it can be considered a good candidate for the dependency extraction problem.

Our motivation for selecting these two methods is that they are radically different approaches. While AL needs to start training from scratch for every new set of requirements, the use of a domain ontology enables knowledge reuse for projects of the same domain.

III. Research Method

In this section, we present the problem statement and then formulate the two research questions. Additionally, we introduce the two baseline methods in detail. Since this research is completely application-oriented, we describe the two industrial data sets first, followed by the configuration of the two baseline methods and the design of the empirical evaluation.

A. Problem Statement

The requirements dependency extraction problem aims to find and explicitly describe all existing dependencies between pairs of requirements. In this study, we aim to understand the effectiveness of two baseline methods that are variants of Active Learning (AL) and Ontology-based Retrieval (OBR). We analyze the outcomes and further evaluate the hybrid approaches by combining these two diverse methods, which are then evaluated on the two industry data sets. This study focuses on two different dependency types: *requires* and *refines*. All other extracted dependencies, i.e. those dependencies which are not classified as *requires* or *refines*, are subsumed as *other*. This categorization implies that these dependencies would require further analysis to determine their specific type.

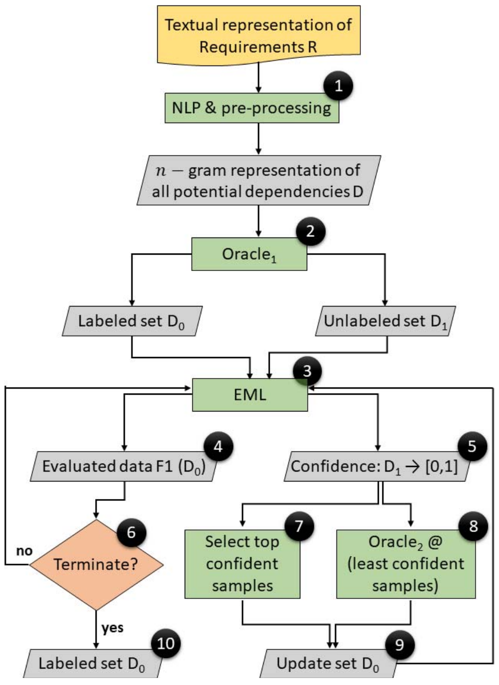
Textual representation of Requirements R
1
NLP & pre-processing
n − gram representation of all potential dependencies D
2
Oracle$_1$
Labeled set D$_0$
Unlabeled set D$_1$
3
EML
4
Evaluated data F1 (D$_0$)
5
Confidence: D$_1$ → [0,1]
6
Terminate?
no
yes
7
Select top confident samples
8
Oracle$_2$ @ (least confident samples)
10
Labeled set D$_0$
9
Update set D$_0$

Fig. 1. Ensemble-based AL method for dependency extraction

B. Research Questions

RQ1: Are AL and OBR valid approaches for requirements dependency extraction from the perspective of industrial applications?
Justification: The two baseline methods have been explored in the existing literature; however, they lack industrial evaluation. From an application perspective, besides precision and recall measure, the amount of effort required to run the methods is an important selection criterion. The results obtained to answer this RQ are not just useful by themselves, but also pave the road to the following second research question.

RQ2: Can AL and OBR be integrated to form an improved dependency extraction method?
Justification: Results from RQ1 provide insights to both baseline methods, which enable us to design a hybrid approach that could reconcile their diverse perspectives and utilize their strengths to yield improved results.

C. Baseline Methods

In this subsection, we describe our AL method in-depth, which is designed specifically for this study and summarize the OBR method, which is already reported in the previous research by Motger et al. [28]. Also, we describe a NLP pre-processor that is common to both the methods.

1) Requirements Dependency Extraction by Active Learning (RD-AL): Figure 1 depicts the workflow and various steps of the specialized AL-based method. We have utilized the general AL method [15], ensemble machine learning [24] and semi-supervised learning (SSL) method, a self-training [25] [49] mechanism[1], to design a variant of AL-based tool which is explained as follows.

- **Initialization:** All the textual requirements information is passed through an NLP pre-processing pipeline first that generates a unigram representation of the textual data fields of all dependency candidates D (Step ❶), which is discussed further in Section III-C3. The output of this pipeline is used by baseline methods to build more complex n-gram representations (bigrams, trigrams, etc.) according to the specific needs of the algorithms. A domain expert - also referred to as $Oracle_1$ - randomly chooses a (typically small) set of potentially dependent requirement pairs and annotates them (Step ❷) to generate a training set.
- **Training using ensemble machine learning**: Thus prepared training set is used for training the machine learner. We utilize yet another ML technique called *ensemble machine learning* (EML) [16] in this step. EML combines individual classifiers' predictions by using their strengths and diluting their weaknesses, which improves the prediction performance of individual classifiers (Step ❸). EML has proven to be effective compared to individual classifiers in problems such as app review classification [21] and software defects prediction [27].
  Our method uses three classifiers: Naive Bayes, Random Forest and Support Vector Machine as they are simple classifiers to work with textual data [30] [21] and provide a probability of an instance belonging to a class which is indicative of the uncertainty (e.g., for a binary classification problem, the closer the probability of a prediction is to 0.5, the higher the uncertainty of classifier's prediction) [15].
- **Choose least confident samples for manual labeling:** The EML learner is trained with an initially small amount of labelled samples which is then used to classify the unlabelled data (Step ❺) applying *pool-based sampling* [40] technique. This is well-suited for sampling the most uncertain data point or instance(s). All requirement pairs in the unlabelled data pool are then evaluated according to their informativeness. Informativeness is computed using *least confidence uncertainty sampling* [40], which selects the instances with the lowest confidence level (probability value) to be labelled next (Step ❺, ❽). In every iteration, n most uncertain predictions are shown to the $Oracle_2$ and labels are acquired (Step ❾).
- **Choose additional labels (a SSL method):** A unlabled pair of requirement for which the classification probability value (confidence value) is higher than the (preset) *confidence* threshold, is chosen to be added to the training set [24] (Step ❼, ❾).
  The selection of $confidence$ threshold and n is a trade-off between the accuracy of the classifier and the effi-

[1]SSL: A learner is first trained with a small labelled data set, and then it is used to classify the unlabelled data. Typically the most confident unlabelled instances, together with their predicted labels, are added to the training set, and the process repeats.

ciency of the trainer. In every iteration, the additional labels selected through SSL and instances labelled by $Oracle_2$ are added to the training set and the process repeats until one of the termination criteria is met.

- **Termination criteria:** AL stops if either a predefined number of iterations is done, or if the degree of improvement is minimal, or if no more unlabelled samples are available (Step ❻, ❿).

We implemented this specialized AL method as a tool called *RD-AL*. RD-AL has been implemented in Python utilizing Scikit-learn libraries [35] for voting classifier based ensemble classifier. Ensemble VotingClassifier combines conceptually different machine learning classifiers and uses a majority vote (hard voting) or the average predicted probabilities (soft voting) to predict the class labels. Such a classifier can be useful for a set of equally well-performing models to balance out their individual weaknesses. Since AL requires probability values to pick the most uncertain and confident instances in every iteration, we have utilized *soft voting classifier*, which generates the averaged predicted probabilities for classification in this research.

2) Requirements Dependency Extraction by Ontologies: For the OBR method, we utilize the OpenReq-DD tool that uses domain ontologies [28]. The ontology defines dependency relationships between specific terms related to the domain of the requirements. Using this information, it is possible to apply NLP techniques to extract meaning from these requirements and relations. Further, ML techniques could also be applied for conceptual clustering to classify the requirements into the defined ontology.

OpenReq-DD is exposed as a RESTful service with an API[2] to provide the required data and perform the dependency extraction. The ontology is built using the W3C Web Ontology Language (OWL[3]). The output of the OpenReq-DD service is a set of extracted dependencies as a JSON response.

Figure 2 shows the sequence of steps that OpenReq-DD performs to extract requirement dependencies. After data pre-processing (see Subsection III-C3), the semantic analysis performs two operations: 1) It generates a dependency tree where each node is a token of the input sentence and edges are the relations between parent words and child words; 2) extracts keywords to categorize each requirement, using a TF-IDF_BASED algorithm which uses frequencies to find most relevant, significant keywords among the requirement corpus. Next, the ontological categorization step classifies requirements into the different concepts of the domain ontology, which is the input to the final dependency extraction step, based on matching ontology concepts and relationships with clustered requirement keywords. See [28] for more details.

3) Natural Language Pre-processor Pipeline: To make both methods comparable and avoid bias in the initial step, we have implemented a state-of-the-art NLP pre-processor component which is deployed as a decoupled Java-based tool.

[2]https://api.openreq.eu/dependency-detection/swagger-ui.html
[3]https://www.w3.org/OWL

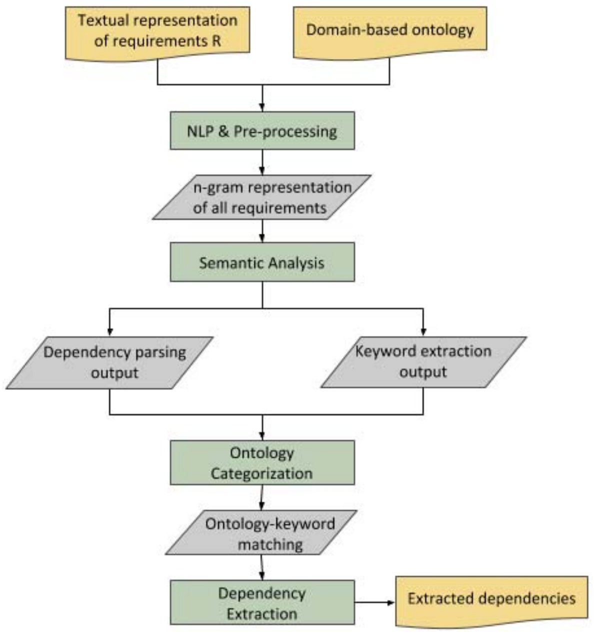
Textual representation of requirements R
Domain-based ontology
NLP & Pre-processing
n-gram representation of all requirements
Semantic Analysis
Dependency parsing output
Keyword extraction output
Ontology Categorization
Ontology-keyword matching
Dependency Extraction
Extracted dependencies

Fig. 2. OpenReq-DD technical workflow representation

This component applies a set of NLP techniques to improve the quality of textual requirements data and to apply a lexical analysis. Thus generated output is then fed to both the RD-AL and the OpenReq-DD tools for further processing.

The NLP pre-processor pipeline implements the following techniques:

- *Noisy Text Cleaning*. A set of 14 sentence cleaning rules which include the removal of non-relevant elements like list pointers or escape sequence characters.
- *Stop-word Removal*. Most common words that are not relevant for the syntactic and semantic analysis of a sentence are filtered out of the requirements text.
- *Standardization*. Words or phrases that are not recognized by standard dictionaries (such as acronyms) are replaced for understandable words.
- *Tokenization*. Each requirement sentence is split into a bag of words or tokens using the Apache OpenNLP toolkit[4].
- *Stemming*. Each token is converted into its root or stem using the KStemmer from the NLP4J toolkit[5].

D. Industrial Data Sets

Research questions are investigated on the two industrial data sets, as summarized in Table I and described below.

1) Company A - Siemens, Austria: This is a multinational conglomerate and the largest industrial manufacturing company in Europe. For this research, we use a collection of Request For Proposals (RFPs) documents of the railway domain. These requirements refer to specific technical features of

[4]http://opennlp.apache.org
[5]https://emorynlp.github.io/nlp4j

TABLE I
INFORMATION ABOUT INDUSTRY DATA INPUT FROM THE TWO COMPANIES A AND B

	Product	# Requirements (n)	# Potential dependent pairs: $n(n-1)/2$	# Annotations acquired	# Ontology classes	Rough ontology construction effort (hrs)
Company A	Train Control System	310	47,985	273	28	5hrs
Company B	Safety Critical Product	93	4,278	191	21	10 to 12hrs

TABLE II
SAMPLE REQUIREMENT PAIR EXAMPLES FROM THE TWO DATA SETS AND THEIR DEPENDENCY TYPE

	Requirement 1	Requirement 2	Dependency
A	TND_2366: All acceptance tests for the ETCS level 2 system shall check the content of the telegrams emitted/received by the RBC and the balises	TND_2145: Shunting routes need not be recorded in the RBC	*Requires*
B	Movement detection setting or enabling movement detection should be initiated from the device's menu and also remotely from the portal	For anti-theft, special locking mechanism will not be provided. If the owners are worried about theft they can bolt it	*Refines*

the railway domain, including the reconstruction, rehabilitation and maintenance of tracks, industrial materials, related legislation, software systems and electronic interlocking installations.

For our study, we selected 310 requirements from the RFPs documents. This selection was assessed by Company A's domain experts, who chose a highly related subset of requirements from the Radio Block Center (RBC) railway subdomain to guarantee the extraction of relevant dependencies. From these 310 requirements, 47,895 requirement pairs were generated, from which 273 pairs are manually annotated (53 as requires, 65 as refines, 30 as others, 125 as non-dependent). These requirement pairs are candidates for the dependency extraction process.

2) Company B - Blackline Safety Corp., Canada: This is a world leader in the development and manufacturing of wireless safety products. For this study, we use the requirements of an area monitoring safety product built to monitor confined and open areas for safety hazards. It is a complex product that uses hardware, firmware, and software technologies. This product has intricate and inconspicuous dependencies in the manufacturing and release levels.

Currently, company B extracts dependencies exclusively manually. The domain knowledge is distributed over different documents and owned by different people, hence, the complete extraction and dependency management process is not only effort intensive but also time consuming. Additionally, there is a risk of missing dependencies. For our study, this company provided a document of 93 requirements which resulted in 4,278 potential requirement pairs. Of these, 191 pairs were manually annotated (43 as requires, 68 as refines, 4 as others, 80 as non-dependent). Table II shows examples from companies A and B.

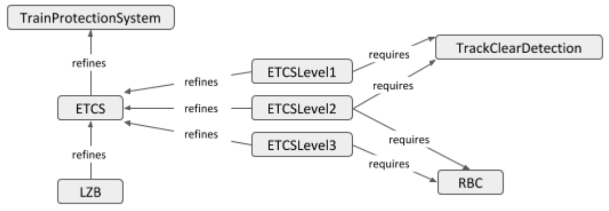

Fig. 3. Railway domain ontology example of Company A

E. Configuration of the Two Baseline Methods

1) Configuring the Ensemble-based AL Approach: For classifier model training, we randomly chose 20% of the data and retained it as the test set; the remaining 80% is used as the training set. The same test set (unseen data) is used in each iteration of AL to compute the classification ($F1$) scores. There are three possible stopping criteria to terminate the learning process [40]: 1) Desired classification accuracy is obtained or accuracy start to degrade, 2) Labelling budget is exhausted, 3) There are no more unlabelled requirement pairs. We chose to terminate if the accuracy does not improve over a certain number of iterations (up to 10).

In order to strike a balance between the classifier's accuracy and trainer's efficiency, we set the number of annotations per iteration to three (n=3). Also, to minimize the addition of high confidence dependent pairs to the training, we set the $confidence$ threshold to 0.9.

2) Configuring the OBR Approach: We built domain ontologies for both the products[6] in collaboration with the respective product's domain experts. For Company A, the ontology was developed collaboratively by some of the authors of this paper who are experts in ontologies in general, and domain experts. Keyword elements and relationships were established based on an analysis of the data set provided by Company A. The result takes the form of a formatted ontology file (using OWL syntax).

Figure 3 shows an excerpt of the ontology for Company A, comprising a small subset of key concepts and their dependency relationships, including two dependency types: *requires* and *refines*. We could see how this ontology conveys the necessary information to automatically extract the dependency among requirements TND_2366 and TND_2145 listed in Table II, considering the relationship "ETCSLevel2 *requires* RBC".

[6]Due to confidentiality reasons, we refrain from providing ontology details for Company B

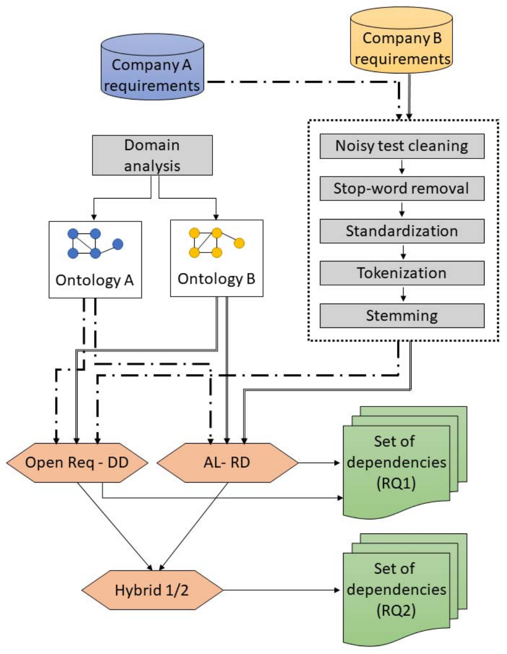

Fig. 4. Research design

The design and elaboration of the domain-specific ontology for Company A (consisting of 28 classes) took 5hrs of effort as it was designed alongside ontology experts from the development team. Company B's ontology (consisting of 21 classes) was also developed in collaboration with the domain expert of this product over 12hrs in multiple iterations and meetings.

F. *Design of the Empirical Investigations*

The overall design of the study has been outlined in Figure 4. We use Company A and Company B requirements as the two industrial data sets to build two independent empirical evaluation scenarios. These include a set of pre-processed requirements using the preprocessor pipeline and a domain ontology file designed by domain experts from each company. This input has been in three different evaluation set-ups: (a) The OpenReq-DD baseline evaluation, (b) The AL-RD baseline evaluation, (c) The two different Hybrid approaches, discussed in detail in Section V. For each one of the three evaluation scenarios, we provide two independent empirical analysis using Company A and Company B data sets. Each combination provides a set of dependencies as a result.

RQ1 is devoted to analyzing the performance of the two baseline methods for the two industrial data sets whereas RQ2 is for the evaluation of the two hybrid methods stemming from AL and OBR approach.

IV. Empirical Evaluation of the Baseline Solution Methods (RQ1)

In this section, we present the evaluation, results and analysis for RQ1. The results are explained through accuracy values which are supported with qualitative and comparative analysis of the two methods from an efficiency perspective.

For the evaluation, we have used 10 times 10-fold cross-validation. Each of the classifier statistics has been analyzed using precision, recall and $F1$ measures.

The evaluation results are summarized in Table III. $F1$ is computed for the test data (unseen data). The standard deviation (STD) has been specified for the cross-validation score to provide information regarding the variance between multiple runs. Special care has been taken to have balanced data sets to ensure an unbiased analysis.

The ensemble-based AL approach (see Figure 1) generates the final output in two stages. Firstly, a coarse classification of dependent and independent requirements is performed. Secondly, only the dependent requirements pairs are used further to classify into fine-grained dependency types in the second stage.

A. *Results Company A*

As summarized in Table III, OpenReq-DD extracted 1,608 dependencies. Among them, there were 501 *refines* dependencies, 1,107 *requires* dependencies and zero *other*. The 10×10 cross-fold validation showed an average $F1$ score of 85% and a standard deviation (STD) of $\mp$0.07.

The RD-AL tool was fed with all 47,985 potentially dependent pairs of requirements. Of these pairs, 273 pairs were randomly selected and annotated by a domain expert. The multi-class classification stage, RD-AL extracted 1,656 *requires* dependencies, 1,758 *refines* dependencies and 4,871 *other* dependencies. The multi-class classification $F1$ score was 72% and the 10×10 cross-fold validation average was a 92% $F1$ score with $\mp$0.02 STD.

B. *Results Company B*

The OpenReq-DD tool extracted 154 *refines* and 44 *requires* dependencies. The 10×10 cross-validation resulted in an average $F1$ score of 76% with a $\mp$0.21 STD. The RD-AL tool was fed with all potentially dependent pairs of requirements generated out of 93 requirements. Of these, 200 pairs were randomly picked and annotated by a domain expert to form a seed training set. In the second stage of fine-grained multi-class classification, the tool extracted 871 *refines*, 416 *requires* and 65 *Other* dependencies. Once again, the operation was terminated when the accuracy plateaued at 49.87% $F1$ score, while the 10×10 cross-validation accuracy showed 86% accuracy and $\mp$0.14 STD.

C. *Analysis*

Due to the varying size of the two industry data sets, the amount of effort and time taken for executing the RD-AL approach were different. Company A's data set needed about 3hrs of domain expert's time of which 0.5hrs was to provide the annotations in every iteration. Similarly, Company B's data set needed one hour of domain expert's time including 0.25hrs for annotation.

OpenReq-DD approach is solely based on the ontology, thus, the efforts and time needed for its construction was a one-time activity. Conversely, RD-AL approach needs greater

TABLE III
BASELINE RESULTS FOR THE TWO APPROACHES

	Annotated data	RD-AL	OpenReq-DD
Company A	273 pairs	**Refines:**1,758 **Requires:**1,656 **Other:**4,871 **F1**:72.0% **10×10:**92%(∓0.02)	**Refines:**501 **Requires:**1107 **10×10:**85%(∓0.07)
Company B	191 pairs	**Refines:**871 **Requires:**416 **Other:**65 **F1:**49.9% **10×10:**86%(∓0.14)	**Refines:**154 **Requires:**44 **10×10:**76%(∓0.21)

effort since a domain expert was actively involved in the process. However, the findings indicate that the number of dependencies extracted by RD-AL was higher compared to OpenReq-DD. RD-AL uses a top-down approach where it considers all the potential pairs of the requirements and uses an oracle to acquire additional knowledge. In contrast, OpenReq-DD tool used a bottom-up approach using ontology as its direction to achieve the same objective. As a result, the dependencies extracted by the OpenReq-DD tool are smaller in number (higher chances that a few of the dependencies could be missed), whereas RD-AL extracted them in a larger numbers (higher chances of false positives).

RQ1: For both case studies, RD-AL was extracting more dependencies than OBR. The results naturally lead to the evaluation of the hybrid approach of the two methods, which could complement each other to improve performance and reduce efforts.

V. Two Hybrid Solution Methods and their Empirical Evaluation (RQ2)

This section elaborates the two hybrid approaches and their evaluation. Ontologies in general help capturing the domain-specific knowledge which can be further analysed to provide a deeper analysis of requirements documents [45] [20] [43]. However, constructing an ontology is time-consuming [19] and has been proven to be difficult to automate for repeated use in the advent of changing and evolving software products [20].

Conversely, AL works on the uncertainty sampling (active sampling) instead of random sampling selection strategy [40]. Hence, AL seeks to minimize the human effort required for training a classifier by intelligently selecting an unlabelled sample for labelling via uncertainty sampling over multiple iterations. However, manual annotation by an oracle and seed train set are the most important aspects of the AL approach. Hence, these can have immense impact on how fast the AL could converge to stopping criteria [15] [24].

Also, there have been studies which evaluate alternatives to $oracles$ (which are typically domain experts otherwise) to overcome the noise that a single human expert could add due to fatigue, boredom, inconsistency etc. Such as crowdsourcing or multiple annotators etc. [40].

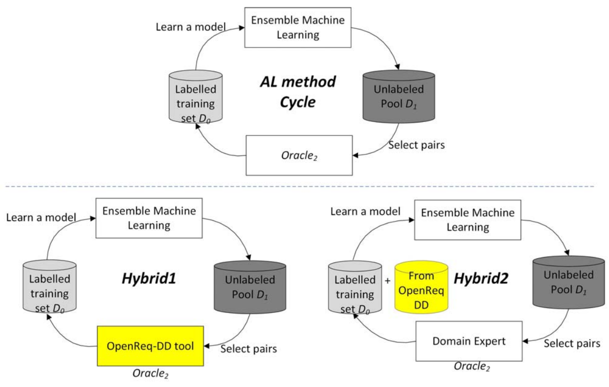

Fig. 5. Comparison of AL with two hybrid approaches

Building on the strength of domain-specific ontologies and the power of AL, in this subsection, we discuss two hybrid approaches and leverage on their respective benefits. These approaches use ontology reasoning as (i) a substitute to the human expert ($Oracle_2$: Figure 1) and (ii) as an extension for the training set for running the AL approach. Although the AL tool was developed in Python and OBR in Java, python wrapper was developed to communicate among the two tools through API. Figure 5 shows how the OBR tool was used as a plug-in into the AL tool to construct the hybrid approaches.

A. OBR as Oracle for Classification (Hybrid1)

In the AL approach (Figure 1), $Oracle_2$ (domain expert) assigns a label to each of the least confident requirements pairs in every iteration. The Hybrid1 approach replaces the human expert by the OpenReq-DD tool as a $Oracle_2$. At each iteration of the AL cycle, the least confident requirement pairs are automatically sent to the OpenReq-DD tool using its REST interface to autonomously predict whether the requirement pairs are dependent or not (stage 1) and which type of dependency are they classified to (stage 2). In this way, we examine the effectiveness of the tool support for classification.

B. Dependencies Extracted from the Domain-specific Ontology as Training Input to AL (Hybrid2)

To create a synergy between the two baseline methods, Hybrid2 uses the dependencies extracted using ontology-retrieval (Figure 2) method as an additional input for the training set (D_0, in Figure 1). This is achieved by running a prior analysis using the OpenReq-DD baseline approach, and this output is then fed as the labelled training set for the AL-RD baseline method. As a result, initial labelled data turns out to be a combination of the human expert annotation process and the automatic OpenReq-DD classification output.

As a consequence, the chances of learning are improved by gathering input from two different types of oracles: human expert and ontology retrieval. As shown in Figure 5, this approach compares with the AL method described in Figure 1.

TABLE IV
HYBRID1 RESULTS: OPENREQ-DD TOOL AS AN ORACLE IN RD-AL APPROACH

	Annotated data	RD-AL output
Company A	273 requirement pairs (positive samples)	**Refines:**1,783 **Requires:**5,448 **Other:**854 **F1:**75.0% **10×10:**96%(∓0.02)
Company B	191 requirement pairs (positive samples)	**Refines:**102 **Requires:**231 **Other:**219 **F1:**82.6% **10×10:**92%(∓0.17)

TABLE V
HYBRID2 RESULTS: TRAINING SET COMBINED WITH DEPENDENCIES FROM OPENREQ-DD TOOL

	Annotated data	RD-AL output
Company A	273 pairs + 1609 OpenReq-DD dependencies (positive samples)	**Refines:**4,292 **Requires:**7,530 **Other:**1,151 **F1:**76.7% **10×10:**95%(∓0.02)
Company B	191 pairs + 198 OpenReq-DD dependencies (positive samples)	**Refines:**585 **Requires:**311 **Other:**207 **F1:**75.8% **10×10:**92%(∓0.05)

C. Results Company A

Tables IV and V provide a summary of the results explained in this section. The tables describe the impact of variants of hybrid methods which try to strike synergy between RD-AL and OBR. For Hybrid1, where OpenReq-DD replaced the human oracle in RD-AL, the $F1$ score increased from 72.0% to 75.0% when compared to the baseline method. The 10×10 cross-validation analysis reported a 96% $F1$ score with STD of ∓0.02.

For Hybrid2, where the output from the OpenReqDD-tool (1,608 dependencies) was added to the training set, showed an $F1$ score of 76.7% which is about 3% higher than the baseline accuracy. The 10×10 cross-validation $F1$ score improved to 95% from 92% with a STD of ∓0.02. The human effort required by Hybrid2 is a combination of the effort required by OBR and the human effort of the AL baseline method, which include the design of the ontology and the feedback provided to the active learner cycle as $Oracle_2$.

D. Results Company B

For Hybrid1, the $F1$ score showed an increase to 82.6% compared to the baseline 49.9%. Also, the 10×10 validation score showed an improvement of 6% to become 92%.

For Hybrid2, when the output from the OpenReq-DD (191 dependencies) was added to the existing training set (200 dependencies), the accuracy increased by 25% to the level of 75.8% in the multi-class classification compared to baseline accuracy. of the RD-AL tool at 49.87%. The 10×10 cross-validation accuracy was 92% with STD of ∓0.05.

Additionally, to measure the potential of the ontology to replace the domain expert as an oracle (Hybrid1), intersection of the annotations, provided by the domain expert (in the baseline results), with OBR was conducted. The results showed that the Company A and B's tests had 56% and 50% overlap, respectively.

E. Analysis

Firstly, when comparing Hybrid1 and Hybrid2, the former generated better results for Company B, whereas the latter showed better results for Company A. Secondly, Hybrid1 results indicate that the domain-specific ontology could cover for approx. 50% of the efforts of a domain expert. More concretely, this implies that OpenReq-DD does not extract all the dependencies present in the data set and provides partial knowledge of the complete representative data of the unlabelled data set. Conversely, considering the sum of the estimated effort required by $Oracle_2$ in every iteration during RD-AL execution (as reported in Section IV), Hybrid1 reduces half of the human effort required for dependency extraction for these two companies.

Lastly, Hybrid2 outcomes show more promising results. There is a clear improvement in the $F1$ score: 3% for company A at 76.7% and an increase of 25% for company B at 75.8%. This disparity could be attributed to the diverse nature of these two data sets. While one is a pure software-based product, the other belongs to a more complex and multi-component project. However, the results indicate that OBR output helped classifier to learn the representative of the data set with this approach.

RQ2: For Hybrid1, the $F1$ score showed an increase to 82.6% compared to the baseline 49.9%. For Hybrid2, the accuracy increased by 25% to the level of 75.8% in the multi-class classification compared to baseline accuracy. OBR also complemented the AL approach by reducing 50% of the human effort.

VI. DISCUSSION OF RESULTS

Despite its exploratory nature, this study offers some interesting insight into the relevance of the two baseline methods to the real-world (industry) data sets. The findings show that our AL-based implementation tends to create false positives. However, this could be a positive aspect when weighed against the adverse impact it could have had on the overall product success (at least in the real-world context) if dependencies were overlooked or omitted. Additionally, the ontology-based approach appeared to be conservative in the extraction process; however, this does not claim external validity.

Although there are different approaches to extract requirements dependencies, there is limited or no evidence of their suitability in real-world settings. In fact, we think that there are no easy answers to the question: how different methods perform under different circumstances? Through this research, we hope to answer this question and make progress, even though it may appear minuscule, towards reaching the bigger goal.

Based on the results from just the two industrial case studies, we do not claim to answer the fundamental question, "Which method works better and in what circumstances?". Instead, we argue that the industrial perspective goes beyond simple $F1$ measurement and includes other crucial aspects that are especially related to effort and impact. Thus, we could draw one such strong conclusion from this research, which is that creating ontologies becomes more valuable, the more it is re-used subsequently in different ways. However, measuring this value was outside the scope of our investigation.

The return of investment (ROI) is defined as the ratio of the current value of the investment and the cost of investment. ROI $\geqslant 1$ indicates that the value (in a given time interval) exceeds the cost of investment and thus is profitable. Consequently, we believe that even a small improvement in extracting requirements dependencies is valuable considering the adverse impact of missing some of them.

In summary, the investment into the hybrid approach is justified by the investigation of ROI: How much rework effort is saved and quality improvement is achieved from the cost of extracting more number of requirements dependencies? The answer is context-specific, but we expect a bigger than one ROI from investing in a hybrid extraction method, in particular, in the case of acquiring additional training samples to generate better models.

Ontologies are intended to provide knowledge engineers with reusable pieces of declarative knowledge, which can be – together with problem-solving methods and reasoning services – easily assembled into high-quality and cost-effective systems [31]. Ontology is created by synthesizing the knowledge elicitated from domain experts into comprehensible mapping.

While it can not be determined if a specific hybrid system integrating ontology-based techniques in the AL cycle is better than an alternate approach, empirical evaluation proves that both baseline methods improved when integrated to counter-balance the main weaknesses of each one of them.

Finally, as the product manager of Company B summarized the value addition of the proposed hybrid methodology, " The safety-critical functionalities of the product and the evolution of the requirements as part of the product incremental development cycles exert tremendous pressure to identify as many dependencies as possible. Hence, decisions at the fine granular level have a larger impact on quality, cost, resource utilization and time-to-market".

VII. Threats to Validity

A. Internal validity

1) Our EML model comprises of the basic and most used ML algorithms in text-based research. However, EML can be enhanced to incorporate additional algorithms and tested for its impact on the classifier performance.
2) RD-AL utilizes the least confidence measure to compute the uncertainty index. It remains to be seen how other measures, such as Min Margin and Label entropy [40], could alter the results.
3) We used the SSL method to select and add a small portion of the most confident sample (step ❼, Figure 1) data to the training set in every iteration. However, some of these requirement pairs may be wrongly labelled which could have an impact on the classifier performance.
4) Hyper-parameter tuning for the machine learners has not been explored in this study. Since this needs additional computational time and resources and thus was not feasible to be carried out in every iteration of AL. However, we do not rule out its impact on the results.
5) Similarly, to have the same measures, we chose $F1$ in both companies but in the case of Company A, since they are more interested in minimizing false negatives, other values of F_k would be a more precise measure for their business case.
6) In this study, synonymy evaluation is not used. Exploring intelligent mechanisms such as semantic similarity using WordNet database could have a significant impact on NL procedures like ontology categorization.
7) Data annotation was carried out by domain experts from companies A and B. Thus, we do not rule out the adverse impact of bias in the initial training set.
8) We did not consider directionality in the dependency set. Additionally, we did not consider the feature engineering process of extracting word and sentence-level features. While these two treatments could have improved the prediction power of our approach, reported results are not compromised.

B. Construct validity

The level of detail and the completeness of the ontology are known to have a substantial impact on the overall retrieval results. It remains open how complete these ontologies for the two industry data sets are. As a matter of fact, ontology design was not specifically focused on a three-type dependency approach and, hence, the generic *other* dependency type is not mapped in the ontologies. A refinement iteration of this third generic type and its design on the ontology could have an impact on the OBR baseline method results. Additionally, we suspect that the language-based noise in the raw data in the form of grammar and semantics could have an impact in general on the outcomes.

C. External validity

The experiments carried out for this research are for just two diverse and different data sets. Hence, the results cannot be generalized. Large scale empirical studies, either on industry or open-source software repositories, could benefit to arrive at more general conclusions.

VIII. Related Work

A. Approaches for Requirement Dependency Extraction

Dependency extraction has been explored as a special case of traceability in the past [10] [6]. J.N.och Dag et al. [33] explored "similar" dependency identification based on similarity measures. Chichyan et al. [9] analyzed the mechanism of how the semantics of unstructured requirements can be evaluated to identify the dependency types using NLP techniques.

Recently many studies have explored requirement dependencies and utilized NLP and ML to a great extent on requirements artefacts [18] [13] [5] [39]. Guo et al.' s study [18] is limited to extract the trace and do not explore the structural type of dependencies, which is a focus of our research.

Deshpande et al. [12] [13] used NLP and ML methods to extract dependencies. Weakly supervised learning methods were explored and various machine learners were utilized in this study on a public data set [17]. Samer et al. [39] analyzed small industry data sets and utilized Latent Semantic Analysis to extract the dependency types. However, these studies lack emphasis on domain-specific knowledge and need a large number of training sets.

B. Ontology-based Approaches

Ontologies are used in different requirements engineering activities [11]. As part of a holistic approach to requirement analysis, Verma and Kass [45] used a semantic graph expressed in OWL to represent a core requirements ontology. The dependencies are just one type ("affects") and are discovered using SPARQL queries. Other works [43] [50] use a domain ontology to create a requirements dependency graph and support some analysis. None of these papers provide empirical evidence on their benefits nor make any attempt to combine ontologies with ML techniques.

Guo et al. [19] proposed the use of manually generated ontologies in a related RE problem, namely term mismatch problem in trace retrieval solutions. The construction of an ontology followed a guided approach by augmenting the ontology with existing traceability knowledge. To alleviate the considerably high cost of constructing such an ontology, the authors suggested that an ontology created through leveraging trace links for one project can be next re-used in other projects.

In a similar vein, Li and Cleland-Huang [23] combined general and domain-specific ontologies to trace requirements with better accuracy than standard information retrieval techniques. To this end, the authors used a syntax tree and considered only noun phrases (representing mostly identifiers' names) and verbs (representing actions) for computing similarity scores between source and target artefacts. Nevertheless, the approach devised by Li and Cleland-Huang [23] lacks higher-level reasoning, as it is unable to capture more sophisticated concepts from the ontology.

Assawamekin et al. [4] utilized an ontology as a knowledge management mechanism to automatically generate traceability relationships. The major limitation of the solution is the inability to handle complex grammatical patterns and requirement sentences written in a different way.

C. Active Learning-based Approaches

Although AL has been explored to classify clone anomaly reports [24], test reports [48] and user models [29], it has not been explored in dependency extraction so far. In the recent past, V. T. Dhinakaran et al. [15] showed that AL could reduce the supervision effort without compromising classification accuracy. Similarly, C. Arora et al. [3] showed that the AL approach could improve the accuracy of automated domain model extraction. Inspired by these results, we have chosen AL and tried to enhance it further.

IX. Conclusions and Future Work

In this study, we proposed a variant of AL, that combined AL, ensemble and SSL to extract requirement dependencies. We also compared it with the OBR approach for two industry data sets. Results showed that AL extracted more dependencies compared to OBR, thus, we designed two hybrid approaches to evaluate how well these baseline methods could complement each other to yield nuanced results. Hybrid1 results showed that it is possible to reduce the human effort required in the AL while improving the reliability of the classification output. Hybrid2 demonstrated that conservative dependency extraction results could be used as input for AL to improve results and provide visibility to new and undetected dependencies.

Requirement dependencies extraction is a difficult task, and no single solution is expected to solve the problem. In fact, there is no "solution" to this problem. Instead, a few existing industrial studies confirmed that each evolutionary improvement could help to improve the product development process from a real-world perspective. Our research on industry data sets shows that improvement not only refers to the formal accuracy ($F1$ value) but also includes the decreasing effort to extract dependencies and the ability to support knowledge management of the company. The latter aspects are hard to quantify but are proven highly relevant [38].

In the future, we will evaluate the impact of the size and completeness of the ontology on the dependency extraction. We expect a trade-off between the effort invested into ontology creation and the benefit generated; therefore, the final effort invested in the ontology may depend on context factors of the organizations and their projects. Existing approaches for ontology construction in RE [34] could be customized to this cost-effectiveness objective using some estimation model for ontology engineering [42]. In another direction, we plan to invest in tuning AL further to find out the classifier's optimized performance based on the characteristics of the problem.

Acknowledgment

This research was partially supported by the Natural Sciences and Engineering Research Council of Canada, Discovery Grant RGPIN-2017-03948, by the Catalan Agencia de Gestión de Ayudas Universitarias y de Investigación (AGAUR) through the FI PhD grant, and by GENESIS project, funded by the Spanish Ministerio de Ciencia e Innovación under project TIN2016-79269-R. The evaluation has been done in collaboration with Blackline Safety and Siemens Austria.

References

[1] M. Abbas, I. Inayat, M. Saadatmand, and N. Jan. Requirements dependencies-based test case prioritization for extra-functional properties. In *2019 IEEE International Conference on Software Testing, Verification and Validation Workshops, ICST Workshops 2019, Xi'an, China, April 22-23, 2019*, pages 159–163, 2019.

[2] M. D. Aias Martakis. Handling requirements dependencies in agile projects: A focus group with agile software development practitioners. In *Proceedings of the 7th International Conference on Research Challenges in Information Science (RCIS)*, pages 1–11. IEEE, 2013.

[3] C. Arora, M. Sabetzadeh, S. Nejati, and L. Briand. An active learning approach for improving the accuracy of automated domain model extraction. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 28(1):4, 2019.

[4] N. Assawamekin, T. Sunetnanta, and C. Pluempitiwiriyawej. Ontology-based multiperspective requirements traceability framework. *Knowledge and Information Systems*, 25(3):493–522, 2010.

[5] M. Atas, R. Samer, and A. Felfernig. Automated identification of type-specific dependencies between requirements. In *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 688–695. IEEE, 2018.

[6] A. Aurum and C. Wohlin, editors. *Engineering and Managing Software Requirements*. Springer-Verlag, 2005.

[7] P. Carlshamre, K. Sandahl, M. Lindvall, B. Regnell, and J. N. och Dag. An industrial survey of requirements interdependencies in software product release planning. In *Proceedings Fifth IEEE International Symposium on Requirements Engineering*, pages 84–91. IEEE, 2001.

[8] P. Carlshamre, K. Sandahl, M. Lindvall, B. Regnell, and J. N. och Dag. An industrial survey of requirements interdependencies in software product release planning. In *5th IEEE International Symposium on Requirements Engineering (RE 2001), 27-31 August 2001, Toronto, Canada*, pages 84–93, 2001.

[9] R. Chitchyan, A. Rashid, P. Rayson, and R. Waters. Semantics-based composition for aspect-oriented requirements engineering. In *Proceedings of the 6th International Conference on Aspect-oriented Software Development*, pages 36–48. ACM, 2007.

[10] A. G. Dahlstedt and A. Persson. Requirements interdependencies: State of the art and future challenges. In *Engineering and Managing Software Requirements*, pages 95–116. Springer, 2005.

[11] D. Dermeval, J. Vilela, I. I. Bittencourt, J. Castro, S. Isotani, P. H. da S. Brito, and A. Silva. Applications of ontologies in requirements engineering: a systematic review of the literature. *Requirements Engineering Journal*, 21(4):405–437, 2016.

[12] G. Deshpande. Sreyantra: automated software requirement inter-dependencies elicitation, analysis and learning. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 186–187. IEEE, 2019.

[13] G. Deshpande, C. Arora, and G. Ruhe. Data-driven elicitation and optimization of dependencies between requirements. In *2019 IEEE 27th International Requirements Engineering Conference (RE)*, pages 416–421. IEEE, 2019.

[14] G. Deshpande and G. Ruhe. Elicitation and maintenance of requirements dependencies: A state-of-the practice survey. https://ispma.org/elicitation-and-maintenance-of-requirements-dependencies-a-state-of-the-practice-survey/. Accessed: 12 Dec 2018.

[15] V. T. Dhinakaran, R. Pulle, N. Ajmeri, and P. K. Murukannaiah. App review analysis via active learning: reducing supervision effort without compromising classification accuracy. In *2018 IEEE 26th International Requirements Engineering Conference (RE)*, pages 170–181. IEEE, 2018.

[16] T. G. Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[17] A. Ferrari, G. O. Spagnolo, and S. Gnesi. Pure: A dataset of public requirements documents. In *Requirements Engineering Conference (RE), 2017 IEEE 25th International*, pages 502–505. IEEE, 2017.

[18] J. Guo, J. Cheng, and J. Cleland-Huang. Semantically enhanced software traceability using deep learning techniques. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, pages 3–14. IEEE, 2017.

[19] J. Guo, M. Gibiec, and J. Cleland-Huang. Tackling the term-mismatch problem in automated trace retrieval. *Empirical Software Engineering*, 22(3):1103–1142, 2017.

[20] J. Guo and J.-C. Huang. Ontology learning and its application in software-intensive projects. In *Proceedings of the 38th International Conference on Software Engineering Companion*, pages 843–846, 2016.

[21] E. Guzman, M. El-Haliby, and B. Bruegge. Ensemble methods for app review classification: An approach for software evolution (n). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 771–776. IEEE, 2015.

[22] V. Kulshreshtha, J. T. Boardman, and D. Verma. The emergence of requirements networks: the case for requirements inter-dependencies. *IJCAT*, 45(1):28–41, 2012.

[23] Y. Li and J. Cleland-Huang. Ontology-based trace retrieval. In *2013 7th International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE)*, pages 30–36. IEEE, 2013.

[24] D. Lo, L. Jiang, A. Budi, et al. Active refinement of clone anomaly reports. In *2012 34th International Conference on Software Engineering (ICSE)*, pages 397–407. IEEE, 2012.

[25] S. Ma, S. Wang, D. Lo, R. H. Deng, and C. Sun. Active semi-supervised approach for checking app behavior against its description. In *2015 IEEE 39th Annual Computer Software and Applications Conference*, volume 2, pages 179–184. IEEE, 2015.

[26] M. McDaniel and V. C. Storey. Evaluating domain ontologies: Clarification, classification, and challenges. *ACM Comput. Surv.*, 52(4):70:1–70:44, 2019.

[27] A. T. Mısırlı, A. B. Bener, and B. Turhan. An industrial case study of classifier ensembles for locating software defects. *Software Quality Journal*, 19(3):515–536, 2011.

[28] Q. Motger, R. Borrull, C. Palomares, and J. Marco. Openreq-dd: A. requirements dependency detection tool. In *Joint Proceedings of REFSQ-2019 Workshops, Doctoral Symposium, Live Studies Track, and Poster Track co-located with the 25th International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2019), Essen, Germany, March 18th, 2019*, 2019.

[29] P. K. Murukannaiah and M. P. Singh. Platys: An active learning framework for place-aware application development and its evaluation. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 24(3):1–32, 2015.

[30] M. Nayebi, M. Marbuti, R. Quapp, F. Maurer, and G. Ruhe. Crowd-sourced exploration of mobile app features: A case study of the fort mcmurray wildfire. In *The 39th International Conference on Software Engineering Companion*. ACM, 2017.

[31] R. Neches, R. E. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, and W. R. Swartout. Enabling technology for knowledge sharing. *AI magazine*, 12(3):36–36, 1991.

[32] A. Ngo-The and M. O. Saliu. Fuzzy structural dependency constraints in software release planning. In *The 14th IEEE International Conference on Fuzzy Systems, 2005. FUZZ'05.*, pages 442–447. IEEE, 2005.

[33] J. N. och Dag, B. Regnell, P. Carlshamre, M. Andersson, and J. Karlsson. A feasibility study of automated natural language requirements analysis in market-driven development. *Requirements Engineering*, 7(1):20–33, 2002.

[34] I. Omoronyia, G. Sindre, T. Stålhane, S. Biffl, T. Moser, and W. D. Sunindyo. A domain ontology building process for guiding requirements elicitation. In *Procs. of the 16th International Working Conference on Requirements Engineering: Foundation for Software Quality REFSQ*, volume 6182 of *Lecture Notes in Computer Science*, pages 188–202. Springer, 2010.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[36] K. Pohl. *Process-Centered Requirements Engineering*. Research Studies Pre, 1996.

[37] J. C. Ribeiro and J. Araújo. Asporas: A requirements agile approach based on scenarios and aspects. In *Proceedings of the 7th International Conference on Research Challenges in Information Science (RCIS)*, pages 313–324, 2008.

[38] I. Rus, M. Lindvall, and S. Sinha. Knowledge management in software engineering. *IEEE software*, 19(3):26–38, 2002.

[39] R. Samer, M. Stettinger, M. Atas, A. Felfernig, G. Ruhe, and G. Deshpande. New approaches to the identification of dependencies between

requirements. In *31st International Conference on Tools with Artificial Intelligence, ICTAI 2019, Portland, USA, November 4-6, 2019*, ICTAI '19. ACM, to appear, 2019.

[40] B. Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[41] F. Shao, R. Peng, H. Lai, and B. Wang. Drank: A semi-automated requirements prioritization method based on preferences and dependencies. *Journal of Systems and Software*, 126:141–156, 2017.

[42] E. P. B. Simperl, C. Tempich, and Y. Sure. Ontocom: A cost estimation model for ontology engineering. In *International Semantic Web Conference*, pages 625–639. Springer, 2006.

[43] H. A. Soomro, Safeeullah et al. Ontology based requirement interdependency representation and visualization. In *International Multi Topic Conference*, pages 259–270. Springer, 2013.

[44] K. P. Stan Buehne, Guenter Halmans. Modelling dependencies between variation points in use case diagrams. In *International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ)*, pages 59–70, 2003.

[45] K. Verma and A. Kass. Requirements analysis tool: A tool for automatically analyzing software requirements documents. In *International semantic web conference*, pages 751–763. Springer, 2008.

[46] A. Vogelsang. Feature dependencies in automotive software systems: Extent, awareness, and refactoring. *Journal of Systems and Software*, 160, 2020.

[47] J. Wang and Q. Wang. Analyzing and predicting software integration bugs using network analysis on requirements dependency network. *Requir. Eng.*, 21(2):161–184, 2016.

[48] J. Wang, S. Wang, Q. Cui, and Q. Wang. Local-based active classification of test report to assist crowdsourced testing. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, pages 190–201, 2016.

[49] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*, pages 189–196, 1995.

[50] D. S. Yudi Priyadi, Arif Djunaidy. Requirements dependency graph modeling on software requirements specification using text analysis. In *1st International Conference on Cybernetics and Intelligent System (ICORIS)*, pages 221–226. IEEE, 2019.

[51] H. Zhang, J. Li, L. Zhu, D. R. Jeffery, Y. Liu, Q. Wang, and M. Li. Investigating dependencies in software requirements for change propagation analysis. *Information & Software Technology*, 56(1):40–53, 2014.

[52] W. Zhang, H. Mei, and H. Zhao. A feature-oriented approach to modeling requirements dependencies. In *13th IEEE International Conference on Requirements Engineering (RE'05)*, pages 273–282. IEEE, 2005.