2017 24th Asia-Pacific Software Engineering Conference

Identifying Security Requirements Based on Linguistic Analysis and Machine Learning

Tong Li
Beijing University of Technology, China
litong@bjut.edu.cn

***Abstract*—Eliciting security requirements in early stage of system development has been widely recognized as an efficient way for minimizing security cost and avoiding recurring security problems. However, in many projects, security requirements are not explicitly specified but rather mixed with other requirements, requiring precise and fast identification of such security requirements. Although several probability-based approaches have been proposed to tackle this problem, they are either imprecise or domain-dependent. In this paper, we propose a tool-supported method to efficiently identify security requirements, which combines linguistic analysis with machine learning techniques. In particular, we apply a systematic approach to identify linguistic features of security requirements based on existing security requirements ontologies and linguistic knowledge. We automatically extract such features from textual requirements, which are then used to train security requirements classifiers using typical machine learning techniques. We have implemented a prototype tool to support our approach, and have systematically evaluated our approach based on three realistic requirements specifications. The evaluation results show that our approach has promising potential to train classifiers that can classify requirements specifications from different application domains.**

***Index Terms*—security requirements classification, security requirements ontology, linguistic analysis, natural language processing, machine learning, prototype**

I. Introduction

Security breaches have repeatedly caused millions of dollar losses per year to large organizations, and this cost is on the rise [1]. Related research has shown that early security requirements analysis can reduce costs related to software development and maintenance from 12-21% [2]. Nevertheless, due to the lack of security expertise, in many projects, security requirements are not explicitly identified and specified during requirements analysis. Instead, security requirements may implicitly spread throughout different parts of a requirements specification [3]. Manually identifying security requirements from the entire requirements specification is time-consuming and error-prone, arousing the need of automatic analysis.

There are two main challenges to the automatic identification of security requirements. Firstly, there is no commonly accepted ontological definition for security requirements, and different people may interpret security requirements differently. As surveyed by Souag et al., there are eight security requirements ontologies have been proposed, each of which involves a particular combination of different security requirements concepts and targets a specific level of abstraction [4]. For example, some ontologies include security objectives [5], [6], some take into account organizational issues [7], [8], and some define vulnerability as a first-class citizen [9], [10]. Secondly, the inherit ambiguity of natural language makes the automatic identification of security requirements even more difficult. Although recent research has introduced specific templates for documenting security requirements [11], in practice most security requirements are specified without using templates. As a result, it is difficult to determine particular linguistic rules for identifying security requirements.

Existing solutions to the automatic identification of security requirements relies on machine learning techniques. Cleland-Huang et al. mine indicators from requirements specification based on their frequency, which are then used to train non-functional requirements classifiers [12]. Knauss et al. use statistical methods to train security requirements classifiers, considering all words appear in a requirement instead of only keywords [3]. Such approaches classify security requirements based on statistical evidence at the lexical level (e.g., term frequency), but do not really “comprehend” the meaning of requirements, as no syntactic and semantic analysis was performed. As a result, the classifiers trained by their approaches are likely to overfit security requirements in particular application domains. Thus, when the classifiers are applied to analyze a new application domain, which are not included in the training dataset, their performance is normally not acceptable.

In this paper, we argue that successful recognition of security requirements concepts and their interrelationships will contribute to the identification of security requirements. To this end, it is essential to take into account the semantic structure of security requirements descriptions when training security requirements classifiers. As such we propose a hybrid approach to automatically identify security requirements, combining linguistic analysis and machine learning techniques. Fig. 1 shows an overview of our proposal, based on which we summarize specific contributions of this paper as follows:

1) defining a set of linguistic rules of security requirements based on existing ontologies of security requirements and linguistic knowledge. Each rule indicates a particular form of security requirements, which can be automatically matched with textual requirements using a proposed syntactic- and keyword-based method.
2) implementing a prototype tool to automatically extract linguistic features based on the proposed linguistic rules and a set of security requirements-related keywords, and to train security requirements classifiers using typical machine learning techniques.

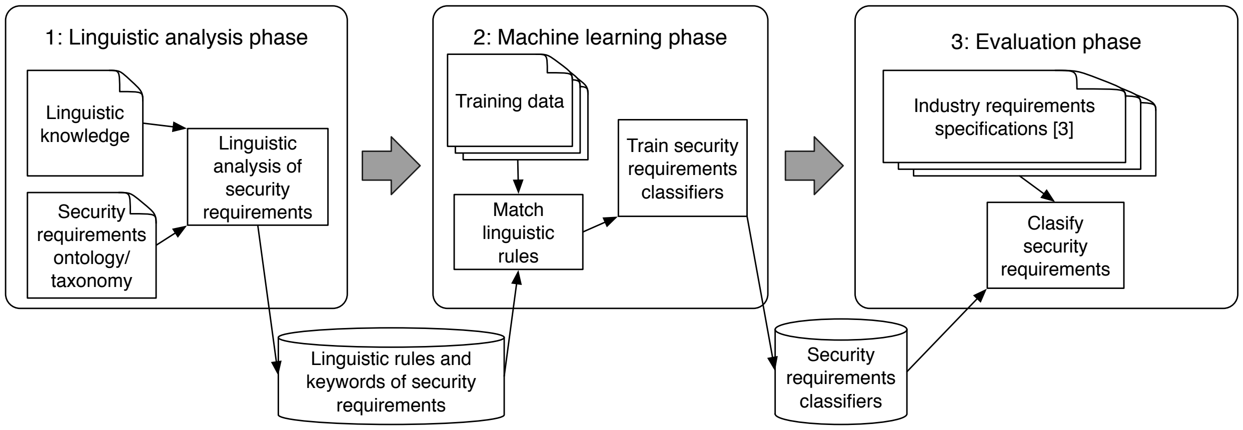
1: Linguistic analysis phase
Linguistic knowledge
Security requirements ontology/ taxonomy
Linguistic analysis of security requirements
2: Machine learning phase
Training data
Match linguistic rules
Train security requirements classifiers
3: Evaluation phase
Industry requirements specifications [3]
Clasify security requirements
Linguistic rules and keywords of security requirements
Security requirements classifiers

Fig. 1: An overview of our proposal

3) systematically evaluating our approach on three industry requirements specifications, the results of which show that our proposal have decent performance over particular datasets and has promising potential to be generalized to classify requirements that come from different application domains.

In the remaining part of this paper, we first present our linguistic analysis over security requirements in Section II, which results in a set of linguistic rules. After that we detail the method for training security requirements classifiers in Section III. We evaluate the obtained security requirements classifiers on three industry requirements specifications, the results of which are presented in Section IV and discussed in Section V. Section VI presents existing approaches and compares them with our proposal. Finally, we conclude the paper and discuss future work in Section VII.

II. Linguistic Analysis on Security Requirements

To efficiently identify security requirements, we need to train classifiers which understand the meaning of security requirements. Thus, we propose to define linguistic rules for security requirements, establishing linkages from semantics of security requirements to their textual representations. With the help of such linkages, we are able to automatically recognize security requirements concepts via text processing, contributing to the identification of security requirements.

In this section, we first present a core set of security requirements concepts, which are summarized based on existing security requirements ontologies and taxonomies and shed light on various definitions of security requirements. After that we investigate the semantics of such concepts and define 47 linguistic rules for security requirements. Successfully matching such rules can be seen as outstanding features when identifying security requirements. Finally, we present a syntactic and keyword-based method for automatically matching textual requirements with the proposed linguistic rules.

A. Security Requirements Concepts

As surveyed by Souag et al., there are more than 20 security requirements-related ontologies [4]. However, most of these proposals focus on security ontologies in a general sense and are not specialized in security requirements. Instead, Firesmith has proposed a conceptual model, in which the concept of security requirements is defined as a first-class citizen [13]. Specifically, this conceptual model highlights concepts that influence and are influenced by security requirements, reflecting various aspects of security requirements. As a result, this conceptual model can well meet our need, i.e., understanding the comprehensive meaning of security requirements.

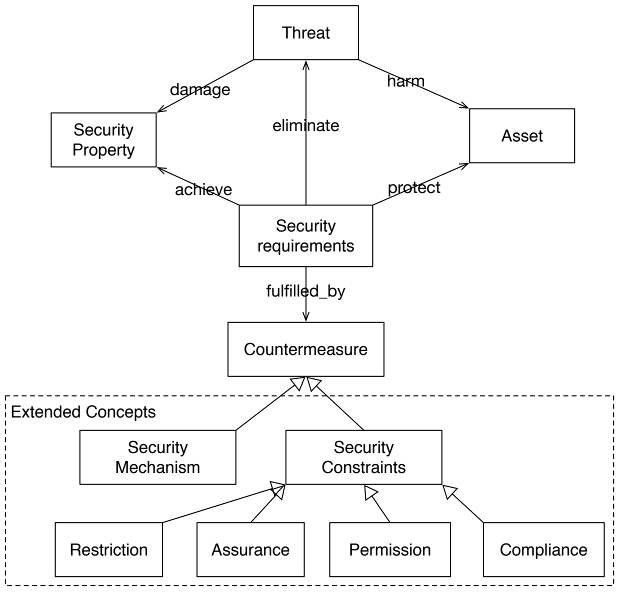

Fig. 2: A revised conceptual model for security requirements

We define a revised conceptual model of security requirements based on Firesmith's work, which is shown in Figure 2.

In particular, an *asset* is anything that has value to an organization (e.g., personal data), which is typically concerned by security requirements. A *security property* specifies a characteristic of security (e.g., confidentiality, integrity, and availability), which indicates the security objective that a security requirement aims to achieve. A *threat* presents an undesired state that an attacker wants to impose on the targeting system (e.g., threat of theft), which harms assets and thus damages corresponding security properties. A *countermeasure* is a solution that is adopted to fulfill corresponding security requirements. In theory, a threat may result in an attack, i.e., they are two different concepts. For example, the threat of theft may result in an actual theft (attack). However, as pointed by Tulloch, in some books and articles, "attack" and "threat" are confounded by being used as synonyms [14]. As such, in this paper, we do not distinguish these two concepts.

As pointed out by Firesmith in [15], most requirements engineers are poorly trained to specify security requirements and often confuse them with security countermeasures. In this paper, we elaborate the concept of security countermeasure to for better distinction. As shown in the bottom part of Figure 2, we consider two sub-classes of countermeasure. A *security mechanism* is a function that is designed to detect, prevent, or recover from a security attack, and thus fulfill one or more security requirements. A *security constraint* is a constraint that is imposed on existing system functions in order to protect assets from harms. In particular, a security constraint can be further categorized as one of the following types: *restriction*, *assurance*, *permission*, and *compliance*. Each of these four types of security constraints is presented in particular syntactic structure, which will be detailed in the next subsection.

B. Linguistic Rules of Security Requirements

Based on the above conceptual model of security requirements and related linguistic knowledge, we have identified a list of typical linguistic rules of security requirements, which are listed in Table I and II. In particular, we first consider atomic rules, i.e., defining security requirements based on an individual security requirements concept. On top of such atomic rules, we further consider their combination, yielding compound rules. These two types of rules are detailed below.

Atomic Rules. In Fig. 2, each concept that is directly associated with the concept of *Security Requirements* can be used to describe security requirements from a particular perspective, indicating a corresponding semantic structure of security requirements. As such, we defined 8 atomic rules according to the conceptual model, i.e., a threat based rule, a asset-based rule, a security property-based rule, and 5 countermeasure-based rules (Table I). For example, as security requirements should eliminate system threats, analysts may express security requirements based on system threats, resulting the first rule in Table I. Specifically, this rule indicates sequential relations among the three security requirements concepts (i.e., terms that are within angle brackets), from left to right.

According to the bottom part in Fig. 2, *Countermeasure* can be specialized as *Security Mechanism* and *Security Constraint*, and the later one can further be specialized into four subclasses. We have defined particular rules for each of these subclassess, which are shown in the bottom row in Table I. For example, the linguistic rule regarding security mechanism specifies a semantic structure that a *Subject Provides* a particular *Security Mechanism*. Note that concepts that are within brackets means they are optional to match.

Compound Rules. The atomic rules shed light on simplest ways of expressing security requirements, while compound rules focus on more complicated cases. Specifically, a compound rule involves multiple security requirements concepts, orchestrating them in an appropriate semantic structure. In such a way, a security requirement is expressed in a more comprehensive manner. By exploring all possible combinations among the 8 security requirements concepts, we have defined 39 compound rules, which are shown in Table II. Specifically, the name of each compound rule consists acronyms of security requirements concepts that are annotated in Table I. For example, the last rule in Table II (i.e., CATS) combines all security requirements concepts, specifying countermeasures that need to provide, assets that need to protect, threats that need to eliminate, and security properties that need to achieve.

It should be noted that we show only one compound rule for each countermeasure-related rule in Table II in order to save space. Each of such countermeasure-related rules can actually be expanded to 5 rules according to the definitions of countermeasure-based rules in Table I. For example, the first three concepts of the *CA* compound rule (i.e., the first row in Table II) can be replaced by $< Subject >< Restrict >< Object >$, yielding another *CA* compound rule.

C. Matching Linguistic Rules

To effectively match textual requirements with defined linguistic rules (Table I and II), in this subsection we propose a syntactic- and keyword-based method.

Syntactic-based matching. Each aforementioned linguistic rule specifies a particular order among different security requirements concepts, showing an ideal representation of a security requirement. However, due to the complexity and ambiguity of natural languages, it is common to document security requirements with additional words or phrases, which cannot be exactly matched with the proposed linguistic rules. As a result, we propose to further analyze syntactic structure of each linguistic rule, based on which the rule matching can be done in a more flexible manner.

We here use parse trees to represent the syntactic structure of sentences. A parse tree is an ordered, rooted tree that describes how a sentence can be deduced from its root. For example, Fig. 3(a) shows a parse tree which describes the basic syntactic structure of the threat-based linguistic rule. In particular, *Subject* should be a *NP (Noun Phrase)*, followed by which is a *VP (Verb Phrase)*. Moreover, the *VP* consists of a *VB (Verb)* which can deduce *Eliminate* and a *NP* which can deduce *Threat*.

Based on this basic syntactic structure, we define a corresponding syntactic pattern that characterizes a set of syntactic

TABLE I: Atomic linguistic rules of security requirements

Name	Rules
Threat-based (T)	$< Subject >< Eliminate >< Threat >$
Asset-based (A)	$< Subject >< Protect >< Asset >$
Security property-based (S)	$< Subject >< Achieve >< SecurityProperty >$
Countermeasure-based (C)	$< Subject >< Provide >< SecurityMechanism >$
	$< Subject >< Restrict > [< Object >]$
	$< Subject >< Assure > [< SecureCondition >]$
	$< Subject >< Permit > [< SecureCondition >]$
	$< Subject >< Obey > [< SecurityPolicy >]$

TABLE II: Compound linguistic rules of security requirements

Name	Rules
CA	$< Subject >< Provide >< SecurityMechanism > to < Protect >< Asset >$
CT	$< Subject >< Provide >< SecurityMechanism > to < Eliminate >< Threat >$
CS	$< Subject >< Provide >< SecurityMechanism > to < Achieve >< SecurityProperty >$
AT	$< Subject >< Protect >< Asset > to < Eliminate >< Threat >$
SA	$< Subject >< Protect >< SecurityProperty > of < Asset >$
TS	$< Subject >< Eliminate >< Threat > to < Achieve >< SecurityProperty >$
CAT	$< Subject >< Provide >< SecurityMechanism > to < Protect >< Asset > from < Threat >$
ATS	$< Subject >< Protect >< Asset > from < Threat > to < Achieve >< SecurityProperty >$
CSA	$< Subject >< Provide >< SecurityMechanism > to < Protect >< SecurityProperty > of < Asset >$
CTS	$< Subject >< Provide >< SecurityMechanism > to < Eliminate >< Threat > to < Achieve >< SecurityProperty >$
CATS	$< Subject >< Provide >< SecurityMechanism > to < Protect >< Asset > from < Threat > to < Achieve >< SecurityProperty >$

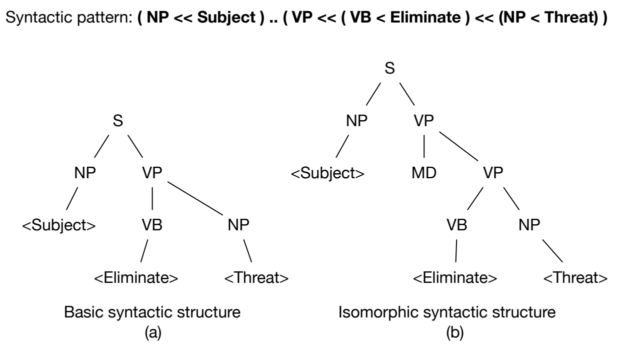
Syntactic pattern: (NP << Subject) .. (VP << (VB < Eliminate) << (NP < Threat))
S
NP
VP
<Subject>
VB
NP
<Eliminate>
<Threat>
Basic syntactic structure
(a)
S
NP
VP
<Subject>
MD
VP
VB
NP
<Eliminate>
<Threat>
Isomorphic syntactic structure
(b)

Fig. 3: The syntactic structure of the threat-based linguistic rule

structures that are isomorphic to the basic one, which is shown on the top of Fig. 3. We use TGrep2 syntax[1] to specify such syntactic patterns, which can be matched automatically. In particular, $A << B$ means A is an ancestor of B, $A < B$ indicates A is the parent of B, and $A..B$ describes A precedes B (in depth-first traversal of tree). Note that $A < B < C$ means A is both the parent of B and the parent of C. According to such TGrep2 syntax, the syntactic pattern defined in on top of the Fig. 3 can match an syntactic structure (i.e., Fig 3(b)) that is isomorphic to the syntactic structure presented in Fig 3(a). Specifically, the isomorphic syntactic structure contains an *NP* which is followed by a *VP*. The *NP* can eventually be deduced to *Subject*, and the *VP* can eventually be deduced to a *VB* and an *NP* which can be directly deduced to *Eliminate* and *Threat*, respectively.

[1]https://web.stanford.edu/dept/linguistics/corpora/cas-tut-tgrep.html

Keyword-based concept identification. In addition to the above syntax matching, another indispensable task is identifying security requirements concepts, serving as the basis for matching linguistic rules. The identification of security requirements concepts requires considerable expertise. In this paper, we adopt a keyword-based approach to automate this task. To this end, we have surveyed existing security taxonomies [16], [13] and security standards [17], [18], based on which we summarize a list of keywords for each security requirements concept, which are shown in Table III.

Have a closer look at this table, we adopt particular strategies to identify different types of concepts based on their characteristics. In particular, we extract domain-independent keywords for *Security Property*, *Security Mechanism*, and *Threat* as they are typically described using common taxonomies. On the other hand, it is impossible to identify keywords for *Asset* and *Subject*, as such concepts are domain-dependent. Instead of matching exact keywords, we analyze the grammatical properties of such concepts. Specifically, we consider noun phrase (NP) as candidates of an asset or a subject. Note that we have not defined keywords for optional concepts (e.g., *Security Policy*), and we currently do not consider those concepts when

TABLE III: Keywords of security requirements concepts

Entity	Keywords
Security Property	security, identification, authentication, authorization, integrity, immunity, anonymity, confidentiality, privacy, availability, non-repudiation, accountability, business continuity, reliability
Security Mechanism	auditing, access control, proof, key, lock, pin, detection, validation, verification, awareness, education, training, back-up, clear desk, clear screen, clock synchronization, cryptography, disposal, log, monitoring, segregation, separation, password, screening, protection, compliance, restriction, assurance, permission
Threat	fraud, error, bug, vulnerability, risk, fault, leakage, misuse, virus
Asset	NP
Subject	NP
Relation	**Keywords**
Eliminate	eliminate, get rid of, stop, reduce, avoid
Protect	protect, defend, save
Achieve	achieve, obtain, satisfy, fulfill
Provide	provide, implement, have, deploy
Restrict	restrict, limit
Assure	assure, ensure
Permit	permit, allow, admit
Obey	obey, comply, subject to

matching corresponding rules, leaving for future work. For relations, as they are expressed in terms of verbs, we argue that synonyms of such verbs can be used as indicators (keywords) for recognizing corresponding relations.

It is worth noting that when using the above keywords to identify concepts, we will consider the stemmed version of those keywords. For example, *validation*, *validate*, or *validated* will be stemmed as *valid* for matching. In addition, the keywords presented in Table III are summarized from literature, which are by no means complete and can be revised and improved in the future.

III. Training Security Requirements Classifiers

Based on the previous linguistic rules, we propose to extract particular features of textual security requirements, which can reflect the essence of security requirements. We then use those features to train security requirements classifiers using typical machine learning techniques. In this section, we first introduce a set of features we create for training security requirements classifiers, and then we describe how to automatically obtain values of those features from textual requirements specifications.

A. Feature Extraction

Using specialized knowledge to create features has been widely accepted as an efficient approach. Our proposed linguistic rules are defined based on both security and linguistic knowledge, which are thus ideal candidates to be used as features of security requirements. Specifically, we argue that requirements which can be matched with one or multiple proposed linguistic rules are much more likely to be security requirements than those cannot be matched. As a result, we first define 47 features, each of which corresponds to a proposed linguistic rule. If the a linguistic rule can be matched with a requirement, the corresponding feature will be set as 1, otherwise it will be set as 0.

Keywords have been used as indicators for classifying Non-Functional Requirements (NFRs) [19], which shows promising results. Such NFR keywords are domain-general, and thus can be used to train classifiers that work well in different domains. However, as reported in [19], obtaining such NFR keywords is a non-trivial task. Since our approach has systematically summarized a list of security requirements keywords based on security requirements concepts (i.e., Table III), we propose to include each of these keywords as a feature for training security requirements classifiers. In other words, if a requirement contains one or multiple such keywords, indicating the requirement may involve specific security requirement concepts, then it is likely to be a security requirement. We set the value of a feature as 1 if its corresponding keyword is identified, otherwise we set it as 0.

B. Text Processing

We have developed a prototype tool to automatically determine values of the above features over textual requirements. The original data to be processed is a list of requirements, each of which is an individual sentence (more details of the input data will be described in the next section), while the output is a featured dataset that will be used for training.

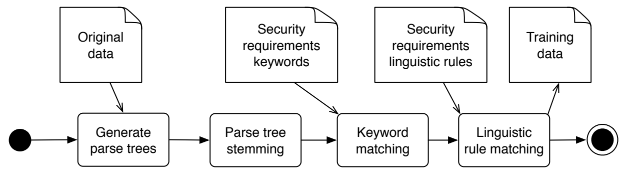

Fig. 4: The process of generating training dataset

The process of generating the training dataset is shown in Fig. 4, which consists of four steps:

- **Generate parse trees.** In order to match the linguistic rules, we first analyze the syntactic structure of each requirement by generating its corresponding parse tree. To this end, we leverage Stanford Parser[2], which is a Java implementation of a lexicalized (Probabilistic Context-Free Grammar) PCFG parser.
- **Parse tree stemming.** To determine whether a requirement contains a keyword, we need to ensure that a word in different inflected forms can be recognized as the same word. To this end, we reduce inflected (or sometimes derived) words to their word stem, i.e., stemming.
- **Keyword matching.** We calculate keyword-based features by matching requirements with each keyword. In addition, once a keyword is found in a requirement, we

[2]https://nlp.stanford.edu/software/lex-parser.html

TABLE IV: Statistics of the three requirements specifications

Specification	Requirements	Security requirements	Percentage
ePurse	124	83	67%
CPN	210	41	20%
GP	176	63	36%

then replace the keyword with its corresponding security requirements concept according to Table III, contributing to the linguistic matching.

- **Linguistic rule matching.** Based on the generated parse trees, we then examine whether they comply with our proposed linguistic rules. In particular, we use Tregex[3] to match syntactic patterns in the parse trees. Based on the matching results, we can then obtain values of each corresponding feature, and eventually build a featured training dataset.

C. Machine Learning Algorithms

We leverage tools and APIs from Weka[4], which covers a broad spectrum of machine learning algorithms and can be easily integrated into our prototype tool. In particular, we first apply some data filters for data pre-processing, then make use of various types of classification algorithms to train security requirements classifiers, and finally evaluate classifiers using their cross validation APIs.

IV. Evaluating Security Requirements Classifiers

In this section, we first briefly describe the dataset that is used in our experiments. Then we present a group of research questions that we aim to address in this paper. According to such questions, we design a series of experiments, and eventually report our experiment results. It is worth noting that we here evaluate our approach as a whole, i.e., treating the execution of our approach as a black box and focusing on the eventual classification results.

A. Security Requirements Dataset

We focus on the security requirements dataset that has been used in [3], which consists of three industrial requirements documents: the Common Electronic Purse specification (ePurse), the Customer Premises Network specification (CPN), and the Global Platform specification (GP).

These three requirements specifications contain 124, 210, 176 pieces of requirements, respectively. Most of individual requirements consist of only one sentence, several others contain two sentences. Knauss et al. have identified 83, 41, 63 pieces of security requriements out of the three requirements specifications [3], respectively. All such information is presented in Table IV, and the three specifications are available online [5].

[3]https://nlp.stanford.edu/software/tregex.html

[4]http://www.cs.waikato.ac.nz/ml/weka/

[5]http://www.se.uni-hannover.de/pages/en:projekte_re_secreq

There are several benefits of using this dataset. Firstly, the requirements specifications come from industry, successfully classifying which can well demonstrate the utility of our approach. Secondly, the three specifications involve three different application domains, allowing us to evaluate the generalization of classifiers that are trained based on particular domains. Thirdly, Knauss et al. have trained Bayesian classifiers over this dataset, serving as the benchmark of our experiments.

B. Research Questions

RQ1. Which kind of machine learning algorithms works best for our approach?

RQ2. To which extent classifiers trained by our approach can classify security requirements? Does our approach perform better than the benchmark?

RQ3. To which extent classifiers trained by our approach can be generalized to classify security requirements in other domains? Is our approach better than the benchmark?

C. Experiment Design

To address the above research questions, we design three experiments accordingly.

Experiment 1. There are many off-the-shelf machine learning algorithms, which have been categorized into specific types. In this experiment, we focus on four particular types, including bayes-based, tree-based, function-based, and rule-based. We pick up two particular algorithms from each of these types, respectively, i.e., Naive Bayes (NB), BayesNet (BN), Logistic Model Tree (LMT), J48, Sequential minimal optimization (SMO), Logistic, Decision Table (DT) and PART. A full list of algorithms that are provided by Weka can be found online[6]. In particular, we separately apply those algorithms to train classifiers based on each requirements specification, and then compare their classification results.

Experiment 2. Based on the results of the first experiment, we adopt the best algorithm and further compare its performance with the benchmark. Given the three requirements specifications, we prepare seven training datasets by enumerating all their combinations, except for the empty one. Then we apply our approach on each of these seven datasets, and compare the classification results with the benchmark.

Experiment 3. In order to assess the generalizability of trained classifiers, we need to ensure the training data and test data belong to different domains, i.e., different requirements specifications. Specifically, we prepare six training datasets by traversing all combinations of the three requirements specifications, except for the empty one and the union of the three specifications. Thus, classifiers trained from these six datasets will be applied to classify security requirements from other domains, i.e., the requirements specifications that are out of the training datasets. The classifications results will then be compared with the benchmark.

[6]https://weka.wikispaces.com/Classifiers

D. Evaluation Metrics.

We use standard metrics in the field of information retrieval, including *precision*, *recall*, and *f-measure* [20]. In order to ensure the validity of our evaluation, we leverage k-fold cross validation [21] to avoid biases in division of dataset. For one thing, this technique can systematically create disjointed training data and test data based on a given dataset. For another, this technique can minimize the likelihood of data overfitting, which happens when classifiers are only trained based on particular training data.

When applying this technique, the dataset will be randomly divided into *k* parts with equal size, *k-1* parts of which are used for training and the remaining part is used for evaluating the trained classifier. This procedure will be iterated with different "*k-1* parts" each time. The choice of *k* is a trade-off. Choosing lower *k* makes the evaluation tasks less complicated and faster, but it implies less variance and may introduce more biases. Thus, we should experimentally determine *k* to ensure unbiased results while minimize the complexity of our evaluation. According to [22], [3], 10 is a decent value of *k* for this purpose. It is worth noting that cross validation cannot be applied in experiment 3, because this experiment requires the test data and training data strictly belong to different domains.

E. Results

Experiment 1. We separately adopt the aforementioned eight classification algorithms to train security requirements classifiers on each of the three requirements specifications. Fig. 5-7 present evaluation results on the three requirements specifications, respectively, which show significant variances among different domains. The average recall on the ePurse requirements specification (hereafter ePurse) is 0.8, while the average recall on CPN and GP are only 0.56 and 0.65, respectively. Especially, the recall of NB has a sharp drop from 0.93 (ePurse) to 0.32 (CPN) and 0.4 (NB).

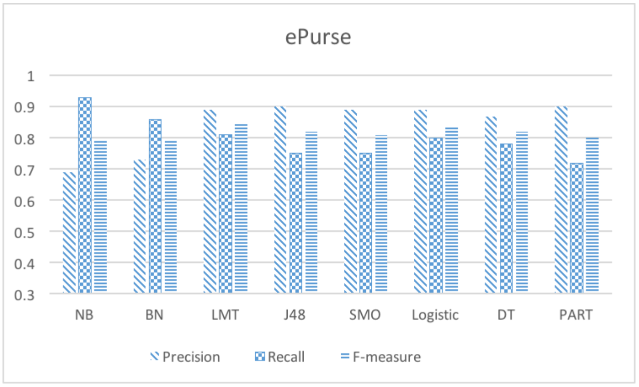

Fig. 5: Classification results of different algorithms on ePurse dataset

Overall, the bayes-based algorithms (i.e., NB and BN) have the worst and most unstable performance. The tree-based algorithms (i.e., LMT and J48) and function-based algorithms (i.e., SMO and Logistic) have similar performances, better than the rule-based algorithms (i.e., DT and PART)

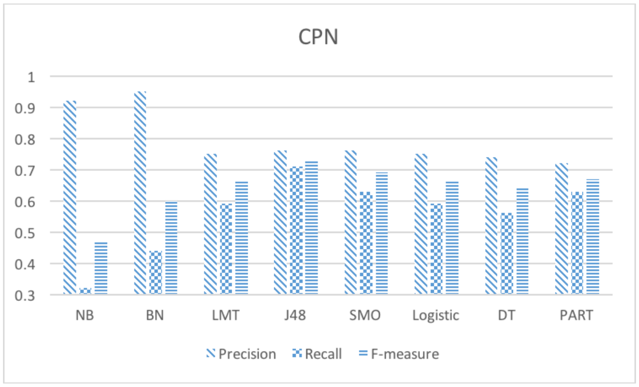

Fig. 6: Classification results of different algorithms on CPN dataset

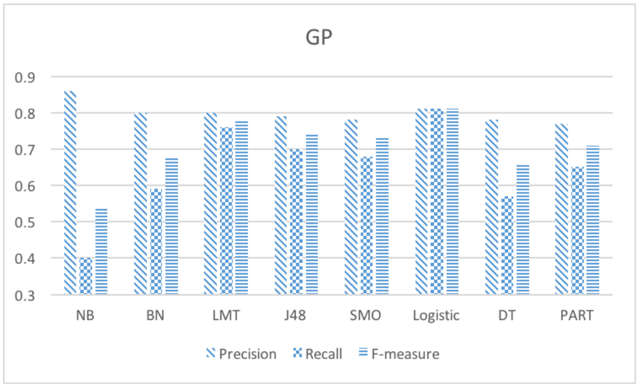

Fig. 7: Classification results of different algorithms on GP dataset

As emphasized by Berry et al., for CASE tools (or in general automated analysis tasks), recall is more important than precision [23]. When classifying security requirements, the recall should be at least 0.7 [3]. Among the eight classification algorithms, only J48 can reach at least 0.7 recall on all the three requirements specifications, and its average precision and recall are among the best. As such, we choose to use J48 in our approach for training security requirements classifiers.

Experiment 2. After selecting J48 as the default classification algorithm, we then evaluate the performance of our approach on various datasets, i.e., all combinations of the three requirements specifications, except for the empty one.

We show the classification results of our approach in Table V, and compare them with the benchmark. Specifically, in each cell, we first present the benchmark results, following by which are our results.

Overall, our approach produces decent results in the sense that the precision and recall can reach at least 0.7 in all datasets. However, the benchmark approach performs better than ours. Especially, the average recall of the benchmark is 0.17 higher than ours, among which the CPN dataset reveals the biggest difference (as highlighted in Table V). Such differences are more obvious as visualized in Fig. 8, where solid triangles indicate benchmark results and hollow rectangles stands for our results.

TABLE V: Compare our approach with the benchmark

Dataset	Precision	Recall	F-measure
ePurse	0.83/0.9	0.93/0.75	0.88/0.82
CPN	**0.98/0.76**	**0.95/0.71**	**0.96/0.73**
GP	0.81/0.79	0.92/0.7	0.86/0.74
ePurse+CPN	0.81/0.77	0.93/0.74	0.87/0.76
ePurse+GP	0.8/0.82	0.96/0.83	0.87/0.82
CPN+GP	0.82/0.71	0.87/0.72	0.85/0.71
ePurse+CPN+GP	0.79/0.76	0.91/0.8	0.84/0.78
average	0.83/0.79	0.92/0.75	0.88/0.77

Remark: benchmark results are presented first

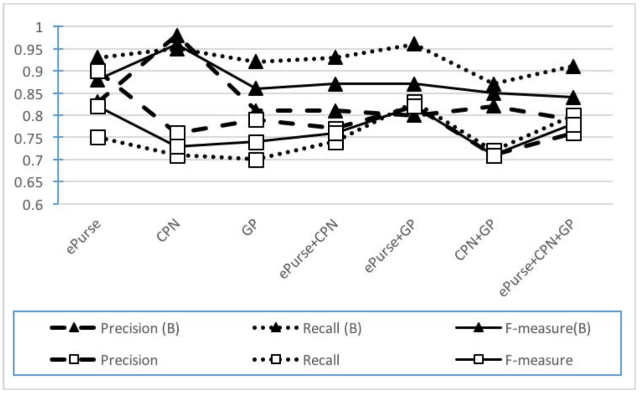

Fig. 8: Comparison on classifying requirements in same domains

Experiment 3. In this experiment, we prepare training data using one or two of the three requirements specifications, and test the trained classifiers using the remaining requirements specifications. Table VI presents the evaluation results in the format *"benchmark results/our results"*. Note that in the first column, the requirements specifications presented in parentheses are the test data.

On average, the precision and recall of our approach can go beyond 0.6, which is not perfect. Considering the difficulties in classifying requirements from completely different domains, we think the results are acceptable and our approach has shown the promising potential for dealing with such classification tasks. Comparing our results with the benchmark, our approach performs significantly better. The average prevision and recall of our approach are 0.18 and 0.11 higher than the benchmark, respectively. Especially, in one third cases (as highlighted in Table VI), the f-measures of our approach are at least 0.3 higher than the benchmark.

Summary and interpretation of evaluation results. We have answered our research questions via the three experiments. In general, both tree-based and function-based classification algorithms can produce reasonable results, among which J48 has the best performance (**RQ1**). Although our approach can reach decent precision and recall (at least 0.7) in all datasets, the benchmark approach can do better than us (**RQ2**). On the other side, the classifiers created by our approach have show promising potential to be generalized to

TABLE VI: Comparison on classifying requirements in different domains

Dataset	Precision	Recall	F-measure
ePurse(CPN)	**0.23/0.52**	**0.54/0.78**	**0.33/0.63**
ePurse(GP)	0.43/0.5	0.85/0.78	0.57/0.61
CPN(ePurse)	0.99/0.95	0.33/0.49	0.47/0.65
CPN(GP)	0.29/0.85	0.19/0.17	0.23/0.29
GP(ePurse)	**0.72/0.94**	**0.48/0.82**	**0.58/0.88**
GP(CPN)	0.29/0.5	0.65/0.76	0.4/0.6
ePurse+CPN(GP)	0.51/0.52	0.56/0.43	0.53/0.47
ePurse+GP(CPN)	0.26/0.47	0.85/0.76	0.4/0.58
CPN+GP(ePurse)	**0.84/0.95**	**0.31/0.75**	**0.46/0.84**
average	0.51/0.69	0.53/0.64	0.44/0.61

Remark: benchmark results are presented first

classify other domains, which can achieve 0.69 precision and 0.64 recall on average. In this respect, our approach behaves significantly better than the benchmark approach (**RQ3**).

Due to the inherit complexity, security requirements can be incorrectly classified even by human, and there are inevitable noisy data in the datasets (discussed in next section). Thus, one plausible explanation for the best performance of J48 is because J48 is comparatively more robust to noise.

The differences between the benchmark approach and our approach can be explained as follows. The benchmark approach focuses on analyzing frequently occurred terms in security requirements, based on which they establish probability models for prediction. In other words, it actually mines potential documentation patterns of security requirements from a particular company, or in general, a particular application domain. Since each application domain may use domain-specific terms and have domain-specific ways of documenting security requirements, the benchmark approach can produce good classification results even if it does not consider semantics of each requirements sentence. On the other hand, the benchmark approach is very likely to step into the problem of overfitting. As the frequent terms may significantly vary among different domains, the classifiers trained by the benchmark approach based on one particular application domain may have very poor classification results in another domain, such as illustrated in experiment 3.

Our approach does not delve into a particular application domain. Instead, we focus on the security semantics that should be expressed by security requirements. Specifically, we analyze such security semantics through security keywords and linguistic rules, which are actually domain-general. As a result, classifiers trained by our approach can produce reasonable results when applied to other application domains. The main challenge of our approach is the variance of security keywords and linguistic rules adopted in different security requirements specification. Consequently, the current list of security keywords and linguistic rules proposed in this paper is by no means complete, which affects the performance of our approach, as evaluated in experiment 2.

V. Discussions

A. Challenges

A major challenge of our study is the reliability of the classified security requirements specifications. Different people, even security experts, can have different conceptual models of security requirements, and thus have different definitions on security requirements. Unfortunately, there is no unique definition that is widely accepted by academic researchers. For example, as summarized by Haley et al., security requirements can be represented as non-functional requirements, context, as well as constraints [24]. As a result, when people classify security requirements, their results can be biased.

In our study, we use the security requirements specifications that have been manually classified by Knauss et al. The ePurse requirements specifications contains detailed information about the manual classification. In particular, five security experts are involved in the manual classification procedure, and the final classification results are subject to experts' votes. Two phenomena can be easily observed, illustrating the aforementioned problem. Firstly, there are many conflicts among security experts, i.e., sec *VS.* nonsec. Secondly, sometimes a security expert cannot really tell whether a requirement is a security requirement, i.e., unknown. As a result, many requirements cannot be clearly determined based on the vote. In particular, the ePurse specification initially contains 200 requirements, but only 124 can be classified for training. It is unclear how the other two requirements specifications are classified. If they are classified by another group of experts, the adopted classification criteria might be different, imposing a reliability threat to validity.

B. Aspects for Improvement

Our proposed security keywords and linguistic rules have shown their usefulness in classifying security requirements, but they are by no means complete and should be further enhanced. Firstly, we can take into account more linguistic knowledge and explore more linguistic rules. For example, dependency analysis can reveal dependency relations among different terms, helping us to better analyze semantics of security requirements. Secondly, we can enrich the set of security keywords by systematically mining more security taxonomies and security catalogs, e.g., IT-Grundschutz Kataloge [25] and NIST Special Publication 800-53 [26]. Lastly, in order to avoid the shortcoming of strict keyword matching, we plan to use $word2vec$ to perform fuzzy matching based on the semantic distance between words.

VI. Related Work

The benchmark approach that is compared in this paper is apparently the most relevant study [3]. In addition to the comparison we have presented throughout the paper, we further argue that these two approaches are applied in different contexts and serve different purposes, i.e., they can complement each other. In particular, if the to-be-classified requirements specifications are within the same application domains as the training requirements specifications, then the benchmark approach should be applied. On the contrary, if the to-be-classified requirements specifications come from a brand new application domain, our approach can contribute to this classification problem.

Cleland-Huang et al. have proposed a probability-based approach for the automatic classification on various non-functional requirements (NFRs), including security requirement [19], [12]. Their approach also mines frequent terms as indicators for classification. However, the more types of NFRs to classify, the more ambiguity and complexity are introduced. Overall, although the recall of their approach can reach 0.8, the precision is only around 0.21. In addition, their approach is applied on requirements specifications that are produced by master students in courses, which are less practical than the industry requirements specifications we used in this paper. It is worth mentioning that, as a precursor to this work, Cleland-Huang et al. have tried a keyword-based approach for classifying security and performance requirements. In particular, they extract a list of keywords from softgoal interdependency graphs (SIGs) [27]. However, their approach does not consider security/performance conceptual models when extracting the keywords, and thus the keyword extraction might not be systematic. Moreover, they have not take into account any linguistic rules. Eventually, their approach can achieve 0.7 recall, but only 0.32 precision.

There are a large amount of studies on mining requirements from text. To name a few, Kaiya and Saeki propose to use domain ontology as domain knowledge for requirements elicitation, which is similar to our approach [28]. However, their approach relies on formal ontology and does not incorporate machine learning techniques. As such, their approach requires significant efforts to establish formal ontologies. Moreover, although their approach can reach almost 1 precision, their recall is only around 0.2. Morales-Ramirez et al. propose to mine Open-Source Software (OSS) forums in order to identify discussants intentions, i.e., "raw requirements" [29]. They adopt similar techniques as ours, i.e., combining linguistic rules (speech act theory) with machine learning techniques. However, the performance of their approach is not as good as ours, and they have not evaluated their approach across different application domains. Massey et al. propose to automatically mining requirements from policy documents using Latent Dirichlet Allocation (LDA) [30]. Their approach shows that the established topic models can indicate whether a document contains software requirements expressed as privacy protections or vulnerabilities. Such topic modeling techniques can be ideal complements to our linguistic-based analysis.

Saeki et al. have created security ontology for an application domain (SOAD), which contains security concepts of higher qualities [31]. In particular, their ontology cover security concepts, such as assets, threats, objectives and security functional requirements, which can help us to further improve the set of security keywords.

VII. Conclusions and Future Work

In this paper, we propose a hybrid approach to automatically identify security requirements, which combines linguistic analysis with machine learning techniques. In particular, we systematically examine existing security requirements ontologies to establish a revised security requirements conceptual model, according to which we define a set of linguistic rules that are normally used to describe security requirements. Then, we recognize distinguishing features of security requirements based on such linguistic rules and a set security keywords that are extracted from literature. These features are then used to train security requirements classifiers using typical machine learning algorithms. We have implemented a prototype to automatically extract linguistic features and train classifiers, with the help of which we have systematically carried out a series of experiments to evaluate our approach. The evaluation results show that our approach has promising potential to train classifiers that can be generalized to classify requirements from different application domains. In this respect, our approach performs far beyond the benchmark approach.

As for future work, we plan to further improve and enrich the linguistic rules proposed in this paper in order for better classification results. In addition, we aim to incorporate cutting-edge NLP techniques (e.g., word2vec) into our approach to enhance the strength of linguistic analysis. Lastly, we are seeking to further evaluate our approach on more realistic requirements specifications, and ideally to support industry practices in reality.

Acknowledgements

The work is supported by National Natural Science Foundation of China (Grants No.91646201), Beijing Natural Science Foundation P.R.China (Grants No. 4173072), and BJUT Startup Fundings (Grants No.040000514117054).

References

[1] L. Ponemon, "Cost of data breach study: Global analysis," Poneomon Institute sponsored by IBM, Tech. Rep., 2015.

[2] K. S. Hoo, A. W. Sudbury, and A. R. Jaquith, "Tangible roi through secure software engineering," *Secure Business Quarterly*, vol. 1, no. 2, p. Q4, 2001.

[3] E. Knauss, S. Houmb, K. Schneider, S. Islam, and J. Jürjens, "Supporting requirements engineers in recognising security issues," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 2011, pp. 4–18.

[4] A. Souag, C. Salinesi, and I. Comyn-Wattiau, "Ontologies for security requirements: A literature survey and classification," in *International Conference on Advanced Information Systems Engineering*. Springer, 2012, pp. 61–69.

[5] F. Massacci, J. Mylopoulos, F. Paci, T. T. Tun, and Y. Yu, "An extended ontology for security requirements," in *Advanced Information Systems Engineering Workshops*. Springer, 2011, pp. 622–636.

[6] A. Kim, J. Luo, and M. Kang, "Security ontology for annotating resources," in *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer, 2005, pp. 1483–1499.

[7] S. Fenz and A. Ekelhart, "Formalizing information security knowledge," in *Proceedings of the 4th international Symposium on information, Computer, and Communications Security*. ACM, 2009, pp. 183–194.

[8] H. Mouratidis, P. Giorgini, and G. Manson, "An ontology for modelling security: The tropos approach," in *Knowledge-Based Intelligent Information and Engineering Systems*. Springer, 2003, pp. 1387–1394.

[9] G. Elahi, E. Yu, and N. Zannone, "A modeling ontology for integrating vulnerabilities into security requirements conceptual foundations," in *Conceptual Modeling-ER 2009*. Springer, 2009, pp. 99–114.

[10] B. Tsoumas and D. Gritzalis, "Towards an ontology-based security management," in *Advanced Information Networking and Applications, 2006. AINA 2006. 20th International Conference on*, vol. 1. IEEE, 2006, pp. 985–992.

[11] M. Rudolph, D. Feth, J. Doerr, and J. Spilker, "Requirements elicitation and derivation of security policy templatesan industrial case study," in *Requirements Engineering Conference (RE), 2016 IEEE 24th International*. IEEE, 2016, pp. 283–292.

[12] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc, "Automated classification of non-functional requirements," *Requirements Engineering*, vol. 12, no. 2, pp. 103–120, 2007.

[13] D. Firesmith, "Specifying reusable security requirements," *Journal of Object Technology*, vol. 3, no. 1, pp. 61–75, 2004.

[14] M. Tulloch, *Microsoft encyclopedia of security*. Microsoft Press,, 2003.

[15] D. Firesmith, "Engineering security requirements," *Journal of Object Technology*, vol. 2, no. 1, pp. 53–68, 2003.

[16] D. G. Firesmith, "A taxonomy of security-related requirements," in *International Workshop on High Assurance Systems (RHAS'05)*. Citeseer, 2005.

[17] I. Committee, "Iso/iec 27000:2012 information technology – security techniques – information security management systems – overview and vocabulary," *Online: http://www.27000.org/*, 2012.

[18] *ISO/IEC 27002, Information technology - Security techniques - Code of practice for information security management*, ISO/IEC) Std.

[19] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc, "The detection and classification of non-functional requirements with application to early aspects," in *Requirements Engineering, 14th IEEE International Conference*. IEEE, 2006, pp. 39–48.

[20] R. Baeza-Yates, B. Ribeiro-Neto *et al.*, *Modern information retrieval*. ACM press New York, 1999, vol. 463.

[21] R. R. Picard and R. D. Cook, "Cross-validation of regression models," *Journal of the American Statistical Association*, vol. 79, no. 387, pp. 575–583, 1984.

[22] F. Chantree, B. Nuseibeh, A. De Roeck, and A. Willis, "Identifying nocuous ambiguities in natural language requirements," in *Requirements Engineering, 14th IEEE International Conference*, 2006, pp. 59–68.

[23] D. Berry, R. Gacitua, P. Sawyer, and S. Tjong, "The case for dumb requirements engineering tools," *Requirements engineering: Foundation for software quality*, pp. 211–217, 2012.

[24] C. B. Haley, R. Laney, J. D. Moffett, and B. Nuseibeh, "Security requirements engineering: A framework for representation and analysis," *Software Engineering, IEEE Transactions on*, vol. 34, no. 1, pp. 133–153, 2008.

[25] I. Münch, "It-grundschutz-kataloge 2007," *Datenschutz und Datensicherheit - DuD*, vol. 32, no. 3, pp. 215–215, Mar 2008. [Online]. Available: http://dx.doi.org/10.1007/s11623-008-0034-7

[26] E. A. Nist, "Nist special publication 800-53 revision 3 recommended security controls for federal information systems and organizations," *CreateSpace, Paramount, CA*, 2012.

[27] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-functional requirements in software engineering*. Springer Science & Business Media, 2012.

[28] H. Kaiya and M. Saeki, "Using domain ontology as domain knowledge for requirements elicitation," in *Requirements Engineering, 14th IEEE International Conference*. IEEE, 2006, pp. 189–198.

[29] I. Morales-Ramirez, A. Perini, and M. Ceccato, "Towards supporting the analysis of online discussions in oss communities: A speech-act based approach," in *Forum at the Conference on Advanced Information Systems Engineering (CAiSE)*. Springer, 2014, pp. 215–232.

[30] A. K. Massey, J. Eisenstein, A. I. Antón, and P. P. Swire, "Automated text mining for requirements analysis of policy documents," in *Requirements Engineering Conference (RE), 2013 21st IEEE International*. IEEE, 2013, pp. 4–13.

[31] M. Saeki, S. Hayashi, and H. Kaiya, "Enhancing goal-oriented security requirements analysis using common criteria-based knowledge," *International Journal of Software Engineering and Knowledge Engineering*, vol. 23, no. 05, pp. 695–720, 2013.