Supporting Analysts by Dynamic Extraction and Classification of Requirements-Related Knowledge

Abstract—**In many software development projects, analysts are required to deal with systems' requirements from unfamiliar domains. Familiarity with the domain is necessary in order to get full leverage from interaction with stakeholders and for extracting relevant information from the existing project documents. Accurate and timely extraction and classification of requirements knowledge support analysts in this challenging scenario. Our approach is to mine real-time interaction records and project documents for the relevant phrasal units about the requirements related topics being discussed during elicitation. We propose to use both generative and discriminating methods. To extract the relevant terms, we leverage the flexibility and power of Weighted Finite State Transducers (WFSTs) in dynamic modelling of natural language processing tasks. We used an extended version of Support Vector Machines (SVMs) with variable-sized feature vectors to efficiently and dynamically extract and classify requirements-related knowledge from the existing documents. To evaluate the performance of our approach intuitively and quantitatively, we used edit distance and precision/recall metrics. We show in three case studies that the snippets extracted by our method are intuitively relevant and reasonably accurate. Furthermore, we found that statistical and linguistic parameters such as smoothing methods, and words contiguity and order features can impact the performance of both extraction and classification tasks.**

Index Terms—**Requirements elicitation, Natural Language Processing, Requirements classification, Weighted Finite State Transducers, Dynamic language models**

I. Introduction

In industrial software development, it is not uncommon for an analyst to be assigned to work on the requirements for a project whose domain is not totally familiar to them, be it because the analyst does not have specific training or previous experience with the project domain, or because the analysts are assigned to an ongoing project. The analyst is faced with the daunting task of becoming familiar with a possibly large amount of documentation that has already been produced. In each case, efficiently interacting with stakeholders (while lacking familiarity with the domain or with pre-existing project artifacts) might pose major challenges.

In this paper, we present a suite of innovative automated techniques aimed at helping the analyst in this scenario. In particular, we envision a system where real-time interaction between an analyst and one or more stakeholders is processed in real-time (we include spoken interaction, via a third-party speech-transcription utility, and written *in-context* interaction, e.g. in chat or quick-turn emails). The system is able to identify a sliding window of active topics in the conversation, and at any stage, recall from a repository of pre-existing documents, information that is relevant for eliciting requirements on the topic at hand. In our scenario, the analyst is assisted in real-time with extracted snippets of the existing documents, which provide context and additional information on the topics under discussion. Such documents might include domain description documents, existing requirements, pending feature requests, and so on — in a word, the entire menagerie of Requirements Engineering (RE) artifacts, as long as they are expressed in Natural Language (NL).

We propose to use both generative models (based on Weighted Finite State Transducers (WFSTs) [1], [2] and statistical Language Models (LMs)) and discriminative models (based on Kernel methods [3] and Support Vector Machines (SVMs) [4]) to extract and classify (by F/NF and by type of NFRs) requirements-relevant knowledge from the existing documents. In contrast to other approaches, we integrate both techniques so that information to present to the analyst is selected based on both its content (i.e. the text detailing the sub-system that is being discussed), and on its role (i.e. the text dealing with the robustness requirements on that sub-system). This integration allows us to improve, at times significantly, over previous approaches. The main contributions of this paper are:

1) We propose a simple novel and dynamic generative model based on the concept of *lexical association*, which is a quantitative measure of the strength of contextual association between two or more words in a corpus. This model leverages the flexibility and efficiency of WFSTs for dynamic modelling and analysis of the incoming text.
2) By modelling requirements-related documents with WFSTs, we propose to use an extended version of the SVM classification approach enriched with non-contiguous string kernel in the context of requirements classification.

We evaluate our solution to the problem of dynamic extraction and classification of requirements-related knowledge by means of three experiments on real-world datasets and compare the results of our classification approach with others published in the literature. The theoretical basis along with the experimental results demonstrate that the proposed solution advances the state of the art in requirements-related term extraction and classification.

In the remainder of this paper, we first provide some background on the current state of the art and, in Section III, formalize the problem we are addressing and establish needed notations and concepts. Then, we present our technical contribution in Section IV; this is followed, in Section V, by an experimental evaluation and comparison. We conclude the paper by discussing threats to validity, with possible extensions and suggestions for future work in Section VII.

II. State of the Art

To better set the stage for our contribution, in this section, we review the salient aspects of existing work on extraction and classification of requirements-relevant terms.

1) Term Extraction: In an early work, Goldin and Berry [5] proposed AbstFinder, a character-based approach to finding non-contiguous common substrings in a corpus, independently of the order of these substrings. AbstFinder neatly addresses the non-contingency of content-carrying terms by using circular shifts, however, it is highly sensitive to the position of terms in a sentence and does not leverage the contextual information implicitly contained in a corpus. Since this work, there has been much active research into corpus-based language learning and analysis in RE. In particular, Sawyer et al. [6] applied a Berry-Rogghes z-score [7] to identify frequent n-grams within a document. However, important and indicative n-grams might not always occur frequently in a document and consequently missed by z-scores and similar statistical methods. Gacitua et al. [8] proposed a Relevance-driven Abstraction Identification (RAI) technique which treats documents as a stream of words and applies the corpora-based frequency profiling approach [9] to rank a domain document words. To identify and rank multi-word terms, they used syntactic patterns (e.g. adjective/nouns, adverb/verbs, and prepositions) and a heuristic weighting approach in which component words in a multi-word term are weighted in descending order from the last word in the term. Likewise, Quirchmayr et al. [10] used lexical and syntactical (i.e. part of speech tagging) characteristics to semi-automatically extract Feature-Relevant (FR) information from natural language user manuals. Following this technique, analysts first conduct a manual revision to ensure syntactical correctness, and then define domain-specific terms which will be used to automatically extract FR information (i.e. a clause compromises at most one subject and one predicate). Lian et al. [11] proposed an approach to explore and highlight requirements knowledge from domain documents. They used a pre-defined set of search terms and measured their density and diversity in fixed-length sequences of words (a.k.a windows).

2) Requirements Classification: There is a growing body of research investigating the effect of using automatic methods for requirements classification [12]–[15]. An efficient categorization of requirements supports analysts to filter relevant information about their ongoing task and enables focused communication and prioritization of requirements [16], [17]. Verma and Kass [18] proposed a Requirements Analysis Tool (RAT) to automatically analyze requirements documents. They used a deterministic finite automata-based approach to parse and provided a set of user-defined glossaries and controlled syntaxes to formulate business rules to standard (i.e. $\langle agent\rangle\langle modal\ verb\rangle\langle action\rangle\langle rest\rangle$) and conditional (i.e. $\langle if\rangle\langle cond\rangle\langle then\rangle\langle rest\rangle$) requirements. They leveraged a pre-defined NFR classification and the Web Ontology Language (OWL) to model and query requirement's relationships and semantics. However, phrasal and semantic analysis steps of this approach need a substantial manual preprocessing effort (e.g. writing sentences in a formalized fashion), which makes them highly dependent on the syntactical form of the requirements.

Cleland-Huang et al. [19], [20] proposed an algorithm that uses the *term* and *inverse document* frequencies to measure the weight score (w) of indicator terms for each NFR Q in the training set. These terms will be used in an indication function $f(w)$ which defines a classification threshold and represents the likelihood that the new requirement belongs to a certain NFR type. In this method, requirements must be processed and reduced to a set of keywords. Also, context-dependent terms such as products and clients' names might appear in the list of indicator terms, which negatively impact the performance of the classifier in highly unbalanced datasets. In the same vein, Rahimi et al. [21] augmented term weight for indicator terms and phrases found in requirements specification to evaluate the mapping between requirements and their associated goals. Also, the output of the indication function in this approach (i.e. the probability score) depends on the lexical content of requirements documents, as the authors assume that these documents are more likely to contain only indicator terms. Likewise, Casamayor et al. [22] used the term frequency-inverse document frequency (TF-IDF) weighting function to transform requirements specification documents into feature vectors. They used the expectation maximization (EM) strategy with the Naïve Bayes algorithm to propose a semi-supervised approach for automatic identification and classification of NFRs aiming at reducing the size of the training dataset. The results of the empirical evaluations on the same dataset used by Cleland-Huang et al. [19] showed that the approach requires less human effort in labeling requirements and it outperformed the K-NN and EM/Naïve Bayes classification methods.

Naïve Bayes and SVMs classifiers are used in several studies [17], [23]–[26] for automatic classification of requirements. As an example, Slankas and Williams [26] proposed a tool-based approach to extract and classify NFRs in requirements specification documents presented in natural language. They used words vectors to parse requirements strings and classify them into 14 categories of NFRs. Among K-NN, Sequential Minimum Optimizer (SMO), and Naïve Bayes [27] classifying techniques used in this work, SMO, a variation of the SVM approach with heuristic training, had the highest effectiveness. Abad et al. [17] used the Stanford NLP [28] and proposed a set of context-based regular expressions to preprocess requirements documents and to increase the performance of various classification approaches (e.g. LDA [29], BTM [30], and Naïve Bayes [27], [31]). After applying their proposed contextual rules, they showed that Binarized Naïve Bayes (BNB), has the highest performance for classifying NFRs compared to other methods used in this study.

III. Problem Formalization and Preliminaries

We first define a formalization of the problem we address in this paper. We will then define basic notations of WFSTs and will provide the technical description of WFST operations that will be used during *extraction* and *classification* processes.

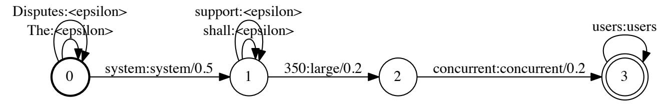


Fig. 1. An example of using WFST to model natural language text. This WFST replaces English and contextual stop-words with empty string ε. It also replaces numbers with a contextual term (contextual replacements are out of the scope of this paper). If we define the weight as the cost of each transition, the lower the weight, the higher the probability of the term in a context. In this example, relevant terms *large* and *concurrent* are receiving lower weights.

A. Defining the Problem

Given a *document repository* $\mathcal{D} = \{d_1, \ldots, d_n\}$, with each d_i representing a document, and a *source stream* (a real-time transcription of an ongoing interview), $\mathcal{S} = \langle s_1, s_2, \ldots s_m \rangle$ with each $s_i = (a_i, b_i)$ representing an *exchange* (where without loss of generality a_i indicates text from the analyst, b_i indicates text from the stakeholder), the problem we address in this paper is to select inside each d_i those textual snippets that are most *relevant* for the most recent part of the conversation happening in $\mathcal{S}$, where "relevant" is understood in intuitive terms, that is according to a (not formally defined) utility function linked to the effectiveness of the elicitation process. Selection of requirements-relevant snippets ($\mathcal{R} = \{r_1, r_2, \ldots r_v\}$) will constitute selecting textual spans which are most likely to be relevant given surrounding context. The context of an occurrence is defined by substrings or subsequences (non-contiguous terms) surrounding it. Each relevant snippet r_j contains a set of requirements-relevant terms $\{t_1, t_2, \ldots, t_k\} \in \mathcal{T}$.

Also, from a predefined list of classes $\mathcal{C} = \{c_1, c_2, \ldots, c_p\}$ we assign a label/class to each selected snippet. Given $\mathcal{D}$ and $\mathcal{S}$ as inputs, the output of our technique will be a tuple $\langle r, c, \{t_1, t_2, \ldots, t_z\} \rangle$, where z is defined by analyst and represents the maximum number of relevant terms that should be highlighted in each extracted snippet.

B. Preliminaries

We briefly describe some of the main theoretical and algorithmic aspects of WFST machines.

a) Weighted Transducers and Automata: A Finite State Transducer (FST) is a finite automaton in which an acceptable path through the initial state to a final state provides a mapping from an input sequence to an output string [32]. Similarly, a weighted transducer is an FST that, in addition to the input and output strings, incorporates a weight into each transition. This weight may present probability, priority, or any other quantities assigned to alternative and uncertain transitions. Figure 1 gives a simple, familiar example of a WFST to model sample requirement "The Disputes system shall support 350 concurrent users". The bold circle represents the initial state and double circle final state. The input and output labels x and y, and weight w of a transition are presented on transition arcs by *x:y/w*. We represent the weight associated with a pair of input and output strings (x, y) modeled by transducer T by $[\![T]\!](x, y)$ (e.g. in Figure 1, $[\![T]\!](350, large) = 0.2$). Given the alphabet Σ, we refer to $|w|$ as the length of a string $w \in \Sigma^*$ and to ε as the empty string (i.e. $|\varepsilon| = 0$).

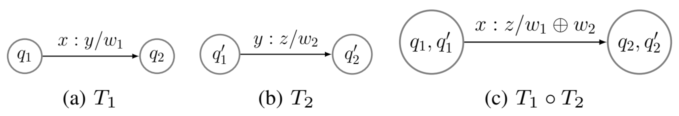


Fig. 2. The composition ($\circ$) operation for detecting substrings (subsequences) of interest with transition rule: $(q_1, x, y, w_1, q_2) \circ (q'_1, y, z, w_2, q'_2) \Rightarrow ((q_1, q'_1), x, z, w_1 \oplus w_2, (q_2, q'_2))$

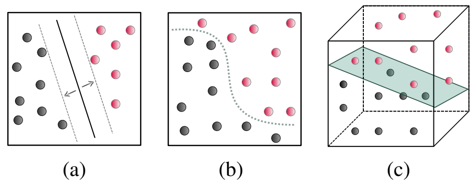
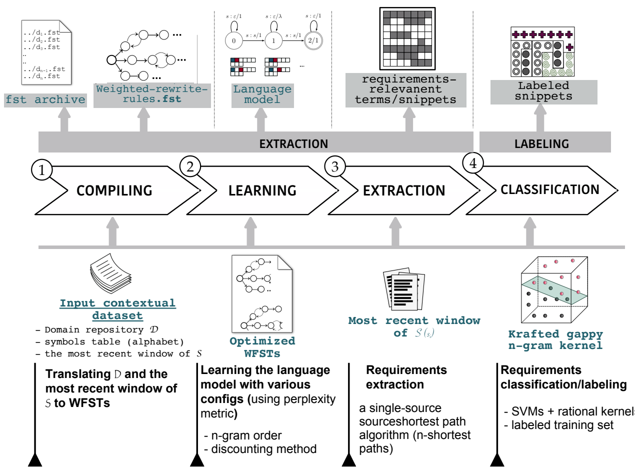


Fig. 3. The application of kernel trick to distinguish non linear separable data points. (a): Linear separable, (b) Non-linear separable, (c) Mapped Space

b) Composition of WFSTs: WFSTs can be composed by a general operation for tying two or more WFSTs together to create a pipeline which can be used to represent statistical models of both generative and discriminative models (e.g. LM and SVMs). As illustrated in Figure 2(a-c), given two WFSTs T_1 and T_2 such that the output alphabet of T_1 coincides with the input alphabet of T_2, composition feeds the output of T_1 into the input of T_2. The composition $T_1 \circ T_2$ is identified by $\oplus$-summing the weights of (x, z) paths, where the weight of all these paths identified by $\otimes$-multiplying the weights of (x, y) and (y, z) paths. Substring y presents the substring of interest appearing in a weighted automata [1], [2].

c) Kernel Methods and Rational Kernels: Figure 3a illustrates a very simple example of distinguishing two different classes (i.e. grey and red circles). One can simply choose a hyperplane to separate the two populations correctly. Among the infinitely many choices available for this hyperplane, the SVM approach determines the hyperplane that maximizes the *margin*, which represents the distance between each population and the hyperplane [4], [33]. However, in practice, the linear separation of the training data is often not possible (Figure 3(b)). One way to address this problem is to use a nonlinear mapping $\Phi : X \rightarrow Y$ which transforms the problem space X to a higher-dimensional space Y with each of the dimensions being combinations of the original features (Figure 3(c)). Even though the space transformation may solve the problem of nonlinearly separable data, taking a large number of dot products in a very high-dimensional space to define the hyperplane may be very costly. *Kernel Trick* is a solution to this problem, which defines a *kernel* as a similarity function (measure). There are various types of kernels for text classification such as word [34], n-gram [35], mismatch [36], spectrum [37], and string [38] kernels which use counts of the common occurrence of subsequences. Kernel K is **rational** when there exists a weighted transducer T such that $k(x, y) = [\![T]\!](x, y)$ [32], for all sequences (i.e requirements expression) x and y, where, $[\![T]\!](x, y)$ denotes the weight associated to a pair of (x, y) [39]. This implies that the objects that are aimed to be classified are weighted automata.



../d1.fst
../d2.fst
../d3.fst
../dn-1.fst
../dn.fst
fst archive
Weighted-rewrite-rules.fst
Language model
requirements-relevant terms/snippets
Labeled snippets
EXTRACTION
LABELING
1 COMPILING
2 LEARNING
3 EXTRACTION
4 CLASSIFICATION
Input contextual dataset
- Domain repository $\mathcal{D}$
- symbols table (alphabet)
- the most recent window of $\mathcal{S}$
Optimized WFSTs
Most recent window of $\mathcal{S}(s_i)$
Krafted gappy n-gram kernel
Translating $\mathcal{D}$ and the most recent window of $\mathcal{S}$ to WFSTs
Learning the language model with various configs (using perplexity metric)
- n-gram order
- discounting method
Requirements extraction
a single-source sourceshortest path algorithm (n-shortest paths)
Requirements classification/labeling
- SVMs + rational kernels
- labeled training set

Fig. 4. The architecture of our proposed extraction/classification process.

IV. Proposed Extraction and Classification Technique

In this section, we start by discussing the intuition behind using *lexical association* and the rationale for its application in extracting relevant terms in requirements-related documents. Then, we provide the technical description of our proposed approach for dynamic control of context-dependency as well as the dynamic calculation of lexical association in existing documents. We conclude this section by describing the process of classifying snippets containing the extracted relevant terms. More precisely, the proposed extraction and classification process is depicted in Figure 4. More application-minded readers can consult Section IV-D first to get a non-technical summary of the extraction scenario and the state of the field, and then read Sections IV-B to obtain more technical details.

A. Rationale

Recall from Section II, most existing work on the extraction of relevant terms from NL text in the context of RE use basic document features such as terms frequency and length, document length and the existence of a term in a repository. Using these features, relevant terms stay independent of other content-carrying terms in the document which contributes to overlooking the context surrounding terms when measuring their relevance. This is a weakness shared by all *bag of words* approaches. To address this problem, in this paper, we use *lexical association* between documents terms, which quantitatively determine the strength of association between two or more words (or terms) based on their co(occurrence) in a corpus [40] and will assign different weights to terms depending on the context they occur in.

The intuition behind using *lexical association* is the basic assumption that a context in which a word is used can often influence its meaning [41]. Thus, in a document, the words that are highly associated with each other and occur together more often than expected by chance have a special function and can be considered as relevant (content-carrying) terms. In contrary, the irrelevant (background) terms will have a very low association with the other terms in a corpus [42].

B. Extraction

To calculate lexical association (i.e. co-occurrence knowledge) we use statistical language models (LMs), which assign probabilities to sequences of words based on their prior history. Using the *chain rule of probability*, we can decompose the probability of any sequence $w_1^n = (w_1 w_2 ... w_n)$ to:

$$P(w_1 w_2 ... w_n) = \prod_{i=1}^{n} P(w_i | w_1^{i-1})$$

where $P(w|h)$ assigns a probability to term hw, considering some history h, and word w [43]. A straightforward maximum likelihood estimate of $P(w|h)$ is given by relative frequency $\frac{count(hw)}{count(h)}$. Since the parameter space of $P(w_1 w_2 ... w_n)$ is too large (i.e. size of the language vocabulary) and there might be some new sentences (or contexts) that have never occurred before, we use *n-gram* language models [35], [44] which approximate h by just the last few words. An ***n*-gram** model is a sequence of n words that approximates the probability of each word only to the last $n-1$ words: "software requirements" and "requirements engineering" in *"software requirements engineering"*, are bigrams (2-gram) and the whole expression represents a trigram (3-gram). While the intuition behind n-gram language models helps to manage the complexity of the probability function, these models are highly dependent on the corpus we use as the document repository (i.e. $\mathcal{D}$) and it underestimates the probability of all possible terms that might occur. Given the fact that the contextual data available for requirements specifications are often not big enough to give us good estimates for the probability of all possible word co-occurrences and considering the creativity of language, there is always the possibility that stakeholders and clients use *requirements-relevant terms* that have not occurred in the training set (i.e. $\mathcal{D}$) but do occur in the test set (i.e. $\mathcal{S}$). For instance, the following repository $\mathcal{D}$:

$\mathcal{D} = \{$The admin should be able to ensure the confidentiality of information and sensitive data of users"$\}$

includes requirements-relevant bigrams $IT = \{$ensure confidentiality, confidentiality information, information sensitive, sensitive data, data users$\}$. However, given $s_j \in \mathcal{S}$ with

$$\begin{aligned} a_j &= \{\text{"Do admins protect information confidentiality?"}\} \\ b_j &= \{\text{"Yes, all private information are confidential."}\} \end{aligned}$$

the probability in $\mathcal{D}$ of both terms "information confidential" and "private information" from s_j are zero, and so is the probability of the entire test set. This is because terms in a_j contain an *unseen context* despite the words appearing in the vocabulary set of $\mathcal{D}$, and terms in b_j contain *unseen words* (i.e. Out-Of-Vocabulary (OOV)). We address these two scenarios as follows:

Unseen Words [OOVs]- To increase the generalizability of our proposed approach we exploit the flexibility of WFSTs by dynamically adjusting the symbol table to map all words in the source stream not found in the domain corpus to an OOV symbol unk.

Unseen Context- To consider all possible variations of a context and to keep our language model from assigning zero to unseen contexts (n-grams), we use **n-gram hierarchy** [1]. In a nutshell this approach takes the view that, sometimes, using less context is a good thing and helps generalizing the context of the n-gram model. In other words, to calculate $P(w_n|w_{n-2}w_{n-1})$ of trigram $w_{n-2}w_{n-1}w_n$ we can instead estimate this probability by using less contexts (i.e bigram probability $P(w_n|w_{n-1})$ or unigram probability $P(w_n)$). To implement the n-gram hierarchy approach we use and evaluate the following techniques:

1) *Backoff:* Following this technique, if the required n-gram has zero instances, we approximate its probability by backing off to (n-1)-gram. We iteratively back off to a lower-order n-gram until we find a term with a non-zero count. For instance, to compute the probability of trigram $w_{n-2}w_{n-1}w_n$ (i.e. w_{n-2}^{n}) we have:

$$P(w_n|w_{n-2}^{n-1}) : \begin{cases} P(w_n|w_{n-2}^{n-1}) & \text{if } C(w_{n-2}^{n}) > 0 \\ P(w_n|w_{n-1}) & \text{if } C(w_{n-2}^{n}) = 0 \wedge C(w_{n-1}^{n}) > 0 \\ P(w_n) & \textit{Otherwise} \end{cases}$$

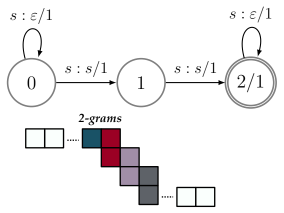
In this paper, we apply and evaluate two backoff methods Katz [45] and Witten-bell [46] (Section V-C3).

2) *Interpolation:* No matter what the frequency of different order n-grams is, by applying this approach we mix the probability of all the n-gram sequences. In other words, we reduce the probability mass from some more frequent terms and give it to the contexts that have never occurred in the dataset [43]. Following presents the general idea of the linear interpolation, combining uni/bi/trigrams, each weighted by κ, where $\sum_i \kappa_i = 1$ [43], [47].

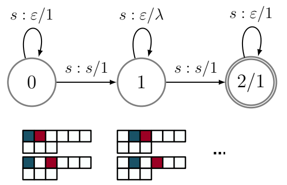
$$P(w_n|w_{n-2}^{n-1}) = \kappa_1 P(w_n|w_{n-2}^{n-1}) + \kappa_2 P(w_n|w_{n-1}) + \kappa_3 P(w_n)$$

Going back to our example, using either of back-off or interpolation techniques, our n-gram model assigns non-zero probabilities to bigrams "private information" and "information confidential". Absolute discounting [48] subtracts a constant discount d from each count and re-distributes the probability mass. Kneser-ney discounting [47] augments absolute discounting by considering the number of *contexts* each word has happened in the domain dataset. By using this approach, we hypothesise that words that happened in more contexts in the past are most likely to appear in some new contexts. For instance, to calculate the probability of term "information confidential" considering the following domain corpus, our model assigns a higher probability to unigram "confidential" than "admin". While "admin" is more frequent compared to "confidential", it is mainly frequent in the term "system admin", while "confidential" has appeared in two different contexts.

[1]This approach is also called as smoothing or discounting in literature [43]



(a) n-gram Kernel



(b) Gappy n-gram Kernel

Fig. 5. Weighted transducers computing the count of expecting (a) all bigrams; (c) all gappy bigrams with a fixed penalty factor λ and maximum gap 1. s represents each symbol (word) $\in \mathcal{D}$.

d_1 = {The system administrator should be able to configure the confidentiality of sensitive information.}
d_2 = {All sensitive confidential data will be secured on the server and only accessible by authorized system administrators.}
d_3 = {Only system administrators can activate the account.}

To identify which language model parameters assign the highest probability to the source stream data (i.e. more accurately predicts the incoming source stream) we used the perplexity score [43], [49]. By LM parameters we mean the n-gram order (i.e. n) and the discounting method (i.e. Katz, Witten bell, absolute, and Kneser-ney). The perplexity of a LM on a test set [2] is defined as the inverse n-gram probability of the source stream data, normalized by the number of words. The lower the perplexity, the higher the ability of the language model in predicting the incoming text. More details about the dynamic application of this parameter in our proposed approach will be explained in Section V.

The model outlined above provides a context-sensitive weight to each of the lexical components of the document. The next step is applying these weighted components to the source stream document to explore the relevant terms in this document. Given source stream $\mathcal{S}$, a weighted transducer S is built using the same symbol table we used for generating transducer D from $\mathcal{D}$. To efficiently capture these requirements-relevant (i.e. domain-specific) terms in real-time, we rank the extracted terms with more probable relevant phrases first. To do this, we use a single source shortest path algorithm applied to the WFST resulted from composing S and D. In a simple word, we first build a language model for $\mathcal{D}$, then recompute a language model for the most recent "window" of $\mathcal{S}$ on every addition to $\mathcal{S}$. The intersection [3] between the two language models returns "relevant" terms (terms that are in the intersection). Finally, we fetch parts of d_i that contain the top n relevant terms:

$$\{t_1, t_2, \dots t_n\} \in \texttt{Shortest\_path}\ ([LM_D \circ LM_S], n)$$

Where n defines the number of top relevant terms (i.e. n shortest paths of the composed WFST). We illustrate this process with a simple example later in Section IV-D.

[2]The test set in this context is the transcribed interview or a new specification text.

[3]Note that intersection ($\cap$) and composition($\circ$) are two different operations in automata theory and they return different results. We used this term to imply the intuition behind our algorithm.

C. Classification/Labeling

As pointed earlier, previous efforts to classify requirements use methods that only work on fixed-size feature vectors, which are difficult to use in dynamic settings as well as in large-scale datasets. Consequently, much effort in the area has focused on feature engineering to produce a crafted list of features [50]–[53]. Even more to the point, non-contiguous language models (i.e *gappy n-grams*), which positively impact the performance of text classifiers [34], [38] have not yet been applied in the context of requirements classification.

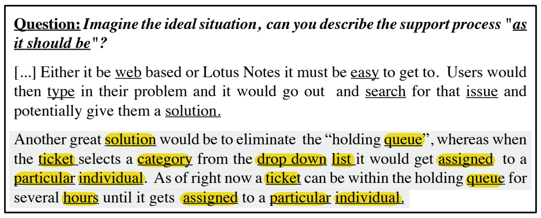
In this paper, we propose to apply the rational kernels [39] method for use with SVMs for classifying requirements. This family of kernels is easy to design and implement and lead to substantial improvements in the classification accuracy [39]. By using a gappy rational kernel (i.e. a *WFST* with ε transitions), requirements specifications that diverge through unrelated word sequences (e.g. products, clients, or end-users' special names) are still likely to contain meaningful subsequences that almost or completely match. Also, adding weights to specific sequences of words is one way of incorporating prior knowledge into the classification algorithm. Our approach is very simple. After extracting text snippets containing requirements-relevant terms that have been identified in the *extraction* phase, the rational kernel (a.k.a similarity function), can be computed efficiently using a general composition operation which is followed by calculating a single source shortest distance from state 0 to find the sum of the weights of all successful paths of the composed transducer. We define two requirements are similar if they share many common substrings or subsequences (non-contiguous/ gappy n-grams). The similarity measure of two transducer E (i.e. extracted from the source stream document) and T (i.e. transducer associated with the training set [4]) can be computed as:

$$K(E,T) = \sum_{x,y} [\![E]\!](x).[\![X]\!](x,y).[\![T]\!](y) = \sum_{x,y} [\![E \circ X \circ T]\!](x,y)$$

Where $[\![T]\!](x,y)$ denotes the similarity measure between two strings x and y. This measure can be defined as the sum of the expected counts in E and T of the matching substrings.

Figure 5a represents an example of transducer X that is designed to compute the expected counts of all bigrams in extracted transducer E. Here is how transducer X does so: State 0 reads a prefix of the bigram x and outputs the empty string ε (i.e. ignoring unmatched prefixes). Then, an occurrence of x is read and output identically. Finally, state 1 reads the rest of the sentence and outputs ε. The number of ways that we can start from state 0 and follow this procedure to reach state 1 (i.e. the accepting state) gives the counts of the specific bigram x. Figure 5b represents a gappy bigram kernel which allows for a gap between the occurrences of two symbols (while keeping the order of terms). In this case, we say two requirements descriptions are similar if

[4]The training set for the classification task includes a set of pre-existing labeled functional and non-functional requirement and it is different from the training set of the *extraction* task

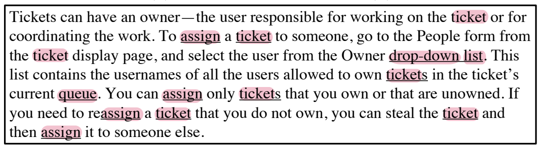


Question: ***Imagine the ideal situation, can you describe the support process "as it should be"?***

[...] Either it be web based or Lotus Notes it must be easy to get to. Users would then type in their problem and it would go out and search for that issue and potentially give them a solution.

Another great solution would be to eliminate the "holding queue", whereas when the ticket selects a category from the drop down list it would get assigned to a particular individual. As of right now a ticket can be within the holding queue for several hours until it gets assigned to a particular individual.

(a) Recent window of $\mathcal{S}$ (s)



Tickets can have an owner—the user responsible for working on the ticket or for coordinating the work. To assign a ticket to someone, go to the People form from the ticket display page, and select the user from the Owner drop-down list. This list contains the usernames of all the users allowed to own tickets in the ticket's current queue. You can assign only tickets that you own or that are unowned. If you need to reassign a ticket that you do not own, you can steal the ticket and then assign it to someone else.

(b) Extracted snippet from $\mathcal{D}$

Fig. 6. Extracted requirements-relevant terms from a transcribed interview question ($s_i \in \mathcal{S}$), using RT Essentials [54] as $\mathcal{D}$.

they share such subsequences (i.e. gappy bigrams). In some contexts, ignoring the gap between content-carrying terms might impact the context of their application. For example in s: {The confidentiality status does not affect the information access} is dealing with information access (rather than the confidentiality (type) of the information). Thus, to penalize the gap/distance between two words of the gappy bigram, we use a fixed penalty factor $0 \leq \lambda < 1$ [38], [39]. The self-loop on state 1, represents the way we consider gaps between bigrams (or n-grams in general). In Section V we investigate how this technique can impact the performance of our labeling/classification task.

Finally, we describe the whole process of proposed extraction/classification approach that we name *Relevance Extraction and Requirements Classification* (RERC) in Algorithm 1.

D. Example

In this section, the extraction method described above is applied to a real-world dataset containing a requirements elicitation interview for a *help desk ticketing system*. As stakeholders might use contextual technical terms when describing a system's features and requirements, we used a book [54] (as domain repository $\mathcal{D}$) which contains both technical and general information about the ticketing system. This book is large enough (224 pages) to be used as a substitution for the volume of text that an analyst might need to review to understand the domain material. We are looking for the lexical association between terms in the two datasets $\mathcal{D}$ (the book) and s_i (the most recent exchange of $\mathcal{S}$, Figure 6a — in practice, a larger window would be used). Following the central assumption of lexical association [41], we also assume that lexical association can be used to semantically relate terms in both document $\mathcal{D}$ and s. Considering the innovative nature of language and diversity of terms that can be used to describe a specific feature, looking for the exact literal match to measure the lexical association between the terms of $\mathcal{D}$ and s is very likely to overlook relevant terms that occur in both documents. For example, after removing stop words and stemming words to their roots, contiguous term *"search issue"* (which appears in s and is a requirement-

ALGORITHM 1: Relevance Extraction and Requirements Classification (RERC) Algorithm

```
Input: (1) domain repository $\mathcal{D} = \{d_1, d_2, ...d_n\}$, (2) source stream documents
$\mathcal{S} = \langle s_1, s_2, ...s_i \rangle$, Training set $T$ //for the classification task

Extraction [Smoothed Language Model]

$ST = \bigcup_{i=1}^{n} ST[d_i]$ //ST represents the symbol table (i.e.
 vocabulary) of the domain repository $\mathcal{D}$

//$n$: the order of the $n$-gram LM
foreach $n \in [1..5]$ do
    foreach $m \in$ [Katz, Witten-bell, Kneser-ney, Absolute] do
        $P[m][n] \leftarrow$ perplexity($n$-gram, $m$, $s$);
        //$s$ represents the most recent window of $\mathcal{S}$
    end
end
//If $n'$-gram and method $m'$ yield the minimum perplexity
$LM_D \leftarrow$ ngrammake( $n'$-gram$_{\mathcal{D}}$, m');
$ET \leftarrow LM_{\mathcal{D}} \circ LM_s$; //ET= the transducer of Extracted Terms
$\mathcal{T} \leftarrow$shortest_path($ET$, $z$); //T= $\{t_1, ..., t_z\}$: z-top ranked relevant terms
//If $r_i$ has the maximum overlap with $\mathcal{T}$
$r \leftarrow r_i \in \mathcal{R}$ //$\mathcal{R}$ is a set of requirements-relevant snippets from $\mathcal{D}$

Labeling/Classification [SVM enhanced with Rational Kernels]

//similarity function K
foreach $n \in \{2, 3, 4\}$, $j \in \{0, 1, 2\}$, and $z \in \{0, 0.5\}$ do
    $LM_k \leftarrow klngram$  order $= n$  max_gap $= j$  $\lambda = z$ // generating
     the transducer which measures lexical association (as the
     similarity measure)
    $N = T_r \circ LM_k \circ T_t$ //$T_r$: transducer of the snippet contains
     relevant terms, $T_t$: transducer related to the documents in
     training set
    $W[N] \leftarrow$ shortest_distance($N$) //to calculate the shortest distance
     from initial states to final state for all common
     subsequences between the training and testing datasets
end

Note: $W[N]$ will be used in combination with SVMs which is out of the scope of this paper

Result: Classified snippet containing requirements-relevant terms: $\langle r, c, \{t_1, t_2, \ldots, t_z\} \rangle$
```

relevant term) never occurs in $\mathcal{D}$, but it occurs in a context with the same meaning: *"[...] does a quick search for all of the new or open Macintosh issues that are currently unowned"*. Also, the language model created for the book returns zero for the probability of predicting the sentence of s which contains the term *"Lotus Notes"*, as this term never occurs in the book. However, this sentence contains a genuine requirement-related information about the platform of the system which should be *web-based* in this context (e.g. *"[...] this ensures that anyone with a web browser can use the system [...]"* from $\mathcal{D}$). We handle out-of-vocabulary (OOV) words' problem by simply replacing all words that do not appear in $\mathcal{D}$ with unk when modeling s, and handle the problem of order and contiguity by using a hierarchy of n*-gram* language models when modeling both $\mathcal{D}$ and s documents. If either OOVs or non-contiguous words appear in a relevant context in s, our approach would perfectly detect the same context in $\mathcal{D}$. Thus, the extraction would be treated as finding all common terms which appear in relevant *contexts* and our algorithm identifies the relevance of terms by computing the intersection between the two hierarchical language models ($\mathcal{D}$ and s). It then extracts the snippet which has the highest lexical association with the most recent window of $\mathcal{S}$. For example, *"queue"* appears as a requirement-relevant term in both $\mathcal{D}$ and s. This term occurred 128 times in 50 different contexts in $\mathcal{D}$, but our extraction algorithm returns the snippet in which *"queue"* appears in the same context as s (Figure 6b). By looking at this snippet, the analyst can come up with some follow up questions about ticket assignment and its ownership methods which have not been discussed in the interview so far, possibly eliciting more information.

V. Experimental Evaluation

A. Method

Our main assumption is that *lexical association*, as approximated by our n-gram hierarchy and gappy n-gram kernels (with decay factor), is a useful indicator of semantic relatedness. To evaluate this assumption, we designed and implemented three experiments as follows:

Experiment #1 (E$_1$): As discussed in Section III-A, the successful extraction of snippets from domain repository $\mathcal{D}$ means that the selected snippets have the highest degree of overlap with the top-ranked extracted relevant terms. Thus, we hypothesized that lexical association can be used as a means to indicate the relevance of extracted snippets and, following the central assumption of lexical association [41], the snippet with the highest level of overlap with extracted terms is the most relevant for the purpose of understanding and analyzing the content of the recent window of the incoming text ($s \in \mathcal{S}$). In this experiment, we pose the null hypothesis: [H_0- *lexical association has no impact on the relevance of extracted snippets*] and use a publicly available industrial dataset to test this hypothesis.

Experiment #2 (E$_2$): This experiment mimics a situation in which an analyst is in a real-time conversation with a client. In this experiment, the same hypothesis (as in Experiment #1) was tested using a publicly available dataset containing transcribed interviews for exploring the requirements of a Help Desk Ticketing system.

Experiment #3 (E$_3$): In this experiment, we evaluate the effectiveness of using gappy n-gram kernels as a similarity measure (which is based on lexical association), in requirements classification. Additionally, we are interested in measuring how well our *lexical-association* based method performs compared to the classification methods used in [17], [19], [24].

B. Evaluation Metrics

1) Edit Distance: Given the identified snippets, to test the null hypothesis H_0, we need to compute the overlap (lexical association) between the reference set of relevant terms and the relevant terms included in the corresponding extracted snippets. By a reference set, we mean a sequence of relevant terms from s required for understanding and analyzing its content. Our reference for a correct set of relevant terms associated with each snippet are the first author of the paper and an external research assistant, who are experienced analysts and familiar with the problem. To remove any threat of biased decisions, we use the *kappa* statistics [55], [56] to measure the magnitude of agreement between them. The calculated kappa coefficient for this task is 0.92, which shows an *almost perfect* [56] agreement for this task.

The most popular metric for measuring the similarity between sequences of words and more specifically between *short* strings is *string edit distance* [57], [58]. This distance is the minimum number of edit operations (e.g. substitutions, insertions, and deletions) to transform the extracted sequence to the reference sequence. For instance the distance between extracted term "confidential information day" and reference term "access confidential information" is two because one insertion (access) and one deletion (day) are required. To calculate the edit distance, we use the Levenshtein edit distance [57], [59], which, incidentally, we implemented as a further transducer.

2) Precision/Recall: We evaluate the performance of our proposed requirements classification/labeling technique using the standard metrics precision (P), recall (R), and F score. The recall is the percentage of the correct answers which are retrieved, whereas precision is the percentage of the retrieved instances that are correct [60], [61]. F score considers both P and R and defines the overall effectiveness of the classification method. Formally, these measures are given by:

$$P = \frac{TP}{TF + FP}, \qquad R = \frac{TP}{TP + FN}, \qquad F = 2\frac{P \cdot R}{P + R}$$

C. Experimental Design

1) Datasets: The experiments E$_{1-3}$ are conducted on the following datasets:

Dataset*₁* $[E_{1,3}]$**:** We used a published dataset by the U.S. Department of Transportation (U.S. DOT), obtained from [62]. In addition to contextual data ($\mathcal{D}$), this dataset provides a repository for both high-level and low-level requirements governing the design of Clarus, a system to monitor environmental and road conditions. The full dataset consists of 700 (300 high level + 400 low-level) requirements.

Dataset*₂* $[E_2]$**:** This dataset was provided by ThyssenKrupp Presta Steering Group USA. The $\mathcal{S}$ component of this dataset is made up of 18 transcribed requirements elicitation interview questions obtained from two separate interviews (of 60 minutes each) to explore the requirements of a Ticketing system [5]. The answer to each question can map to one or more requirements (32 functional and 14 non-functional requirements in total). We obtained $\mathcal{D}$ from the full source of a textbook on ticketing systems, RT Essentials [54]. RT is a high-level open-source ticketing system. The book is large enough to be used as the domain document and its subject is representative of the technical domain that the analyst might need to understand.

Dataset*₃* $[E_3]$**:** This dataset is obtained from the OpenScience tera-PROMISE repository[6] and consists of 625 labeled natural language requirements (255 FRs and 370 NFRs).

2) Preprocessing and WFST Transformation: We utilized the *tm* [63] and *Snowball* [64] packages in R to remove punctuations, English/predefined stop words and to stem words to their roots (e.g. confidentiality, confidential $\rightarrow$ confidenti, based on the popular Porter's stemmer [65]).

3) Learning and Extraction: After transforming $\mathcal{D}$ to a set of WFSTs, we applied perplexity measure to find the most probable LM for each pair of $\langle s_i, \mathcal{D}\rangle$. In particular, we built 20 static language models for $\mathcal{D}$ and indexed each model with $\langle m_j, n\rangle$, where $m_j \in \{$Katz, Witten-bell, Absolute, Kneser-ney$\}$ and denotes the discounting method, and $n \in \{1, 2, 3, 4, 5\}$ the order of each LM. LM$_{\langle m_j, n\rangle}$, where discounting method m_j and order n generate the minimum perplexity, will be composed with the most recent "window" of $\mathcal{S}$ on every addition to $\mathcal{S}$. To explore and indentify relevant terms, we use the top m ranked paths of the WFST generated from LM$_{\langle m_j, n\rangle} \circ$ LM$_s$. This process involves a straightforward implementation of a generalization of the Dijkstra algorithm [67] (i.e. the n-best strings problem [66]).

[5]This dataset is publicly available by ThyssenKrupp Presta Steering Group

[6]https://terapromise.csc.ncsu.edu/!/#repo/view/head/requirements/nfr

TABLE I

DETAILS OF THE DATASET AND COMPARISON RESULTS FOR THE CLASSIFICATION TASK

NFR	**CQ$_1$**		**CQ$_2$**	
	#Te	**#Tr**	**#Te**	**#Tr**
Us	20	56	15	52
Op	20	51	15	47
Pe	15	35	15	37

(a) Classification datasets

Reference	**Us**	**Op**	**Pe**
Cleland-Huang et al.[19]	0.25	0.20	0.38
Abad et al. [17]	0.86	0.84	**0.93**
Kurtanovi and Maalej [24]	0.74	0.71	0.82
Our approach	**0.89**	**0.88**	0.91

(b) Comparison results (E$_3$)

To provide this flexibility to analysts to manage the number of relevant terms (as it might need to be changed based on the size of the s "window"), we used a parametric value which can be changed during the application of the method.

4) Classification: This section illustrates and examines the application of the rational kernels describes in Section IV to requirements classification. We did a series of experiments on Datasets$_{1,3}$ using rational kernels to answer the following classification questions (CQs) about the performance of our algorithm in identifying various types of NFRs, in particular usability, operability, and performance requirements [7]:

CQ1- Domain knowledge: Does combining the recently discussed requirement with its corresponding extracted relevant snippet impact the performance of the classifier?

CQ2- N-gram kernel characteristics: How is the performance of a classifier affected by: (1) removing the contiguity constraint; (2) the length of the n-gram; and (3) adding cost (decay factor λ) to gaps in non-contiguous n-gram kernels?

Table I(a) indicates the details of the datasets we used for training and testing the classification tasks. We used the OpenFST library for the implementation of the rational kernels used in our classification task, and the LIBSVM [69] library to combine these kernels with SVMs.

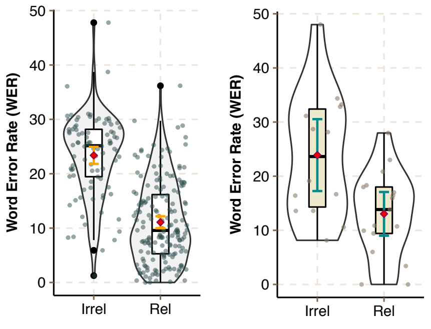
D. Results

In the following, the results of experiments E$_{1-3}$ on tasks extraction and classification are discussed and interpreted.

1) Extraction: In terms of the overall performance of our proposed extraction technique, of 269 low-level requirements descriptions in Dataset$_1$, experiment #1 matched 196 (73%) descriptions to a relevant snippet (i.e. the corresponding high-level requirement). Similarly, of 31 transcribed interview questions in Dataset$_2$, experiment #2 assigned 20 (64%) questions to a relevant extracted snippet from $\mathcal{D}$ (i.e. RT Essentials [43]).

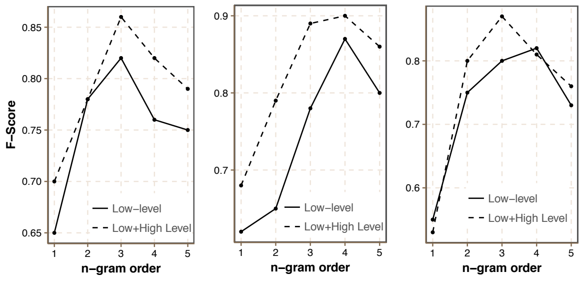
With regard to the accuracy of extracted relevant terms (i.e. lexical association) in extracted snippets, the mean word error rate for E$_1$ is 16.9% (standard deviation (σ): 9.1%, 95% CI mean: 1.1%) and for experiment E$_2$ is 19.3% (σ: 10.1%, 95% CI mean: 3.7%). The higher WER for E$_2$ is probably due to

[7]Due to the very few instances of other subclasses of NFRs in this dataset and to be robust to within-class imbalance problem [68], in this experiment, we focused only on usability, performance, and operability NFRs.



(a) E1 (b) E2

Fig. 7. 95% confidence interval of sample means for WER and relevant and irrelevant snippets: (a): E_1 on 269 high-level requirements and (b): E_2 on 31 transcribed interview answers.



(a) Usability (b) Performance (c) Operability

Fig. 8. CQ1- The impact of using extracted requirements-relevant knowledge on the performance of the classifier.

dataset $\mathcal{D}$ we used to generate the training LM. Interestingly, however, while this dataset is taken from a more general domain, we achieved an accuracy of 81.7% (95% confidence interval $\pm$ 3.7%) for this task.

In terms of the impact of lexical association on the relevance of extracted snippets from $\mathcal{D}$, as we could not confirm the normality of the distribution of our edit distance data (using *Q-Q* plots [70]), we used the non-parametric Kruskal-Wallis test to examine the null hypothesis posed in Section V-A. With p-values at 95% significance or greater ($\leq$*0.05*) we can reject the null hypothesis that the lexical association makes no significant difference on the relevance of extracted snippets. The results of our statistical tests reject this hypothesis for Experiment #1 at *p-value= 2.2e-10* and for Experiment #2 at *p-value=0.02*. Moreover, by looking at Figure 7(a-b), we can see that the 95% confidence interval of sample means for WER of extracted relevant snippets is consistently lower than irrelevant snippets, which implies that *lexical association* can be used as an indicator of relevant snippets.

Finding ❶ *"Lexical association makes a significant impact on the relevance of extracted requirements-relevant snippets and can be used as an indicator of relevance in these snippets.*

2) Classification: To answer *CQ1*, we used the results of Experiment #1 and applied a simple n-gram kernel used with SVMs to both low and high-level requirements. Figure 8 (a-c) shows the results of using low-level requirements alone, compared to using low-level and high-level in combination as the input of the classifier (for three requirements types: usability, performance, and operability). These results show that the classification performance of applying n-gram SVMs to low-level requirements combined with their corresponding high-level description is consistently better than that applied to only low-level requirements. From this, we conjecture that using high-level requirements descriptions for the classification task may provide more contextual information for the similarity measure function, which positively impacts its performance. **Finding ❷**: *"Applying the classification technique to the high-level, instead of the low-level requirements (one-best snippet) can improve the performance of the classifier. This is inline with the literature (e.g. [71], [72]) that show augmentation in data-space can boost the performance of classifiers and reduce overfitting"*.

Table II presents the results of our experiments for the classification task associated with CQ2. It gives the performance of the classification approach as a function of the order (length) and the contiguity of the n-gram kernel as well as the decay factor assigned to each gap. The decay factor allows controlling the number of gaps that are allowed in the kernel function [34]. While for $\lambda = 0$, gaps have no impact on the sequence similarity in our kernel function, for $\lambda = 0.5$, gaps add more cost to corresponding paths (which will impact the classification decision when calculating the shortest distance algorithm). From these results we conclude: **Finding ❸** *"The performance of the classifier varies with respect to the configuration of its similarity measure (kernel). In particular, we found that: (3-4)-gram non-contiguous subsequences are more effective than other configurations in capturing the similarities between strings."*

Regarding the number of gaps in non-contiguous subsequences, our results show that the F score peaks at $max_{\mathrm{g}} = 1$ for all three types of NFRs; and at $\lambda > 0$ (i.e. *0.5*) for usability and performance categories. Further increase in the number of gaps (max_{g}) can degrade the power of the similarity function (kernel) and substantially decreases the recall. We speculate that this may be partly due the similar behaviour of unigrams and n-grams with $max_{\mathrm{g}} > 1$.

Although the kernels we used to perform the classification tasks do not incorporate any knowledge about the dataset being used (except removing English and contextual stop words), it still captures contextual information. It seems that the non-contiguity of subsequences in the similarity function can better detect the inherent polysemy characteristic of natural language. Finally, Table I(b) compares our classification results with some previous work which used the same dataset to classify NFRs. The results show that non-contiguous n-gram kernel with $\lambda > 0$ outperforms the techniques used in previous work, except one NFR from [17]. This could be due to the pre-processing step applied in [17] containing a set of manually-developed, ad-hoc transformation rules for each type of NFRs. However, our approach obtains contextual semantics based on the lexical association between terms and does not apply any manually defined contextual rules. **Finding ❹**: *"The results of our experiments indicate that the application of SVMs with gappy n-gram kernels and with a non-zero decay factor can provide an effective alternative to the existing methods used*

TABLE II
CLASSIFICATION PERFORMANCE CONSIDERING VARIOUS n-GRAM KERNEL CHARACTERISTICS (CQ2) USING DATASET3

		Usability				Operability				Performance			
max_g		g_0	g_1		g_2	g_0	g_1		g_2	g_0	g_1		g_2
			λ_0	$\lambda_{0.5}$	λ_0		λ_0	$\lambda_{0.5}$	λ_0	λ_0	λ_0	$\lambda_{0.5}$	λ_0
2-gram	P	0.82	0.85	0.85	0.78	0.87	0.89	0.87	0.88	0.90	0.91	0.92	0.89
	R	0.76	0.70	0.76	0.71	0.66	0.68	0.68	0.62	0.72	0.74	0.76	0.71
	F	0.79	0.77	0.80	0.74	0.75	0.77	0.76	0.73	0.80	0.82	0.83	0.79
3-gram	P	0.90	0.93	0.94	0.85	0.89	0.91	0.90	0.88	0.96	0.98	0.98	0.89
	R	0.82	0.77	0.85	0.73	0.78	0.75	0.77	0.67	0.80	0.83	0.85	0.75
	F	0.86	0.83	**0.89**	0.79	0.83	0.83	0.83	0.76	0.87	0.90	**0.91**	0.81
4-gram	P	0.86	0.88	0.89	0.79	0.89	0.94	0.92	0.91	0.89	0.92	0.89	0.90
	R	0.80	0.71	0.73	0.76	0.83	0.83	0.82	0.73	0.77	0.79	0.81	0.73
	F	0.83	0.79	0.80	0.77	0.86	**0.88**	0.87	0.80	0.83	0.85	0.85	0.80

in previous works for requirements classification."

VI. THREATS TO VALIDITY

Internal validity. While we used the kappa coefficient to assess the reliability of ce dataset for measuring the edit distance, the estimated kappa coefficient itself could be due to chance [55], which poses a threat to the validity of our results. To address this threat, we calculated the p value of the resulted coefficient. This value ($<$0.05) shows that the estimated agreement is not due to chance [8]. We can also discount any history or maturation threat, as the technique is entirely automated and the computations do not rely on previous state.

External validity. Our experimental results only apply to the three datasets we have used. We do not claim that those datasets are representative of all situations. However, the main contribution of this paper is the technique in itself, which due to the intuitive nature of the "requirements-relevant" concept, must to a certain extent rely on experimenting on each given context before generalizing claims. More precise and extensive case studies are needed to fully address this threat by better scoping the applicability.

Construct validity. As we only used the final outcome of the extraction approach, the method we used to evaluate this approach might pose a threat to construct validity. One may ask, other parameters such as the order of n-gram language models or using a different discounting method (from the one used to generate the final result) might impact the error rate (or the edit distance) differently. However, theoretical considerations [73] and a series of studies conducted by Klakow and Peters [74] show that the word error rate (WER) and perplexity are linearly correlated and are related by a power law. As, during the learning process, we used the perplexity measure to tune the LM of each incoming dataset and considered potential factors for this task, this threat is mitigated in our evaluation method.

Researcher bias. In some experiment we relied on human judgment. One of the judges was an author of the present paper; however, a second judge was not involved in this research, and the inter-subject agreement was quite strong, so we trust our results were not unduly influenced.

[8]Note that this p value only shows that the estimated kappa is not due to chance and does not test the strength of the agreement level.

Content validity. The metrics (Section V-B) we used to evaluate our approach pose a potential threat to content validity. Future studies might be conducted with different metrics to ensure that all interesting dimensions have been considered.

VII. CONCLUSION AND FUTURE WORK

We have presented a technique to dynamically extract requirements-relevant knowledge from existing documents, in order to assist analysts by surfacing relevant information from documental sources during an interactive interview. We evaluated this technique by conducting experiments on three different datasets and used the WER parameter to measure the effectiveness of our approach. The results of our experiments show that the proposed technique is an effective and feasible approach for extracting requirements-relevant knowledge.

On the technical side, we also proposed to use non-contiguous n-gram kernels in the context of requirements classification and applied rational kernels combined with SVMs to implement this technique. Although these kernels do not incorporate any contextual information (apart from the preparation process) they can still capture semantic information. The results of our experiments show that non-contiguous medium length n-grams with decay factor$>$ 0 better capture the similarity between strings that contiguous or short (bigram) non-contiguous n-gram subsequences, which improves over the previous state of the art at the NFR classification task.

Our work advances current techniques by combining both generative and discriminating models to support analysts in their requirements elicitation tasks in real-time.

Although the paper includes a fairly varied evaluation of the application of the proposed approach, there are still some factors (e.g. human interface factors) that might impact the application of this technique. Thus, one possible direction for improvement is developing a tool that can perform the extraction/classification tasks for any industrial contexts, to be able to gather users' feedback on real cases. To this end, we have implemented a prototype tool that we intend to use for a field study.

The techniques we have developed in this work can also be applied to different problems. For example, a dynamically updated language model like the one we have used for finding relevant snippets, could be used instead to identify unexplored areas of a domain (i.e. issues in a change requests repository which are not addressed by a requirements document), or to match incoming bug reports or feature requests to commit messages in open source projects. We intend to explore these possibilities in future work.

References

[1] M. Mohri, F. Pereira, and M. Riley, "Weighted Finite-State Transducers in Speech Recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 69 –88, 2002.

[2] M. Mohri, "Weighted finite-state transducer algorithms. an overview," in *Formal Languages and Applications*, C. Martín-Vide, V. Mitrana, and G. Păun, Eds. Springer Berlin Heidelberg, 2004, pp. 551–563.

[3] V. D. Sánchez A, "Advanced Support Vector Machines and Kernel Methods," *Neurocomputing*, vol. 55, no. 1-2, pp. 5–20, 2003.

[4] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *Machine Learning: ECML-98*, C. Nédellec and C. Rouveirol, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 137–142.

[5] L. Goldin and D. M. Berry, "Abstfinder, a prototype natural language text abstraction finder for use in requirements elicitation," *Automated Software Engineering*, vol. 4, no. 4, pp. 375–412, 1997.

[6] P. Sawyer, P. Rayson, and K. Cosh, "Shallow Knowledge as an Aid to Deep Understanding in Early Phase Requirements Engineering," *IEEE Transactions on Software Engineering*, vol. 31, no. 11, pp. 969–981, 2005.

[7] G. Berry-Rogghe, "The computation of collocations and their relevance in lexical studies," *The computer and literary studies*, pp. 103–112, 1973.

[8] R. Gacitua, P. Sawyer, and V. Gervasi, "On the effectiveness of abstraction identification in requirements engineering," in *2010 18^{th} IEEE International Requirements Engineering Conference*, 2010, pp. 5–14.

[9] P. Rayson and R. Garside, "Comparing corpora using frequency profiling," in *Proceedings of the Workshop on Comparing Corpora - Volume 9*, ser. WCC '00, Association for Computational Linguistics, 2000, pp. 1–6.

[10] T. Quirchmayr, B. Paech, R. Kohl, and H. Karey, "Semi-automatic software feature-relevant information extraction from natural language user manuals," in *Requirements Engineering: Foundation for Software Quality*, P. Grünbacher and A. Perini, Eds., Springer International Publishing, 2017, pp. 255–272.

[11] X. Lian, M. Rahimi, J. Cleland-Huang, L. Zhang, R. Ferrai, and M. Smith, "Mining requirements knowledge from collections of domain documents," in *2016 IEEE 24th International Requirements Engineering Conference (RE)*, 2016, pp. 156–165.

[12] N. A. Ernst and J. Mylopoulos, "On the Perception of Software Quality Requirements during the Project Lifecycle," in *Requirements Engineering: Foundation for Software Quality*, R. Wieringa and A. Persson, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 143–157.

[13] N. Niu and S. Easterbrook, "Extracting and Modeling Product Line Functional Requirements," in *2008 16th IEEE International Requirements Engineering Conference*, 2008, pp. 155–164.

[14] A. Mahmoud and G. Williams, "Detecting, Classifying, and Tracing Non-functional Software Requirements," *Requirements Engineering*, vol. 21, no. 3, pp. 357–381, 2016.

[15] E. Knauss and D. Ott, "Automatic Requirement Categorization of Large Natural Language Specifications at Mercedes-benz for Review Improvements," in *Proceedings of the 19th International Conference on Requirements Engineering: Foundation for Software Quality*, ser. REFSQ'13, Essen, Germany: Springer-Verlag, 2013, pp. 50–64.

[16] C. Duan, P. Laurent, J. Cleland-Huang, and C. Kwiatkowski, "Towards Automated Requirements Prioritization and Triage," *Requirements Engineering*, vol. 14, no. 2, pp. 73–89, 2009.

[17] Z. S. H. Abad, O. Karras, P. Ghazi, M. Glinz, G. Ruhe, and K. Schneider, "What Works Better? A Study of Classifying Requirements," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, 2017, pp. 496–501.

[18] K. Verma and A. Kass, "Requirements analysis tool: A tool for automatically analyzing software requirements documents," in *The Semantic Web - ISWC 2008: 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26-30, 2008. Proceedings*. Springer Berlin Heidelberg, 2008, pp. 751–763.

[19] 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.

[20] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc, "The Detection and Classification of Non-Functional Requirements with Application to Early Aspects," in *14th IEEE International Requirements Engineering Conference (RE'06)*, 2006, pp. 39–48.

[21] M. Rahimi, M. Mirakhorli, and J. Cleland-Huang, "Automated Extraction and Visualization of Quality Concerns from Requirements Specifications," in *2014 IEEE 22nd International Requirements Engineering Conference (RE)*, 2014, pp. 253–262.

[22] A. Casamayor, D. Godoy, and M. Campo, "Identification of non-functional requirements in textual specifications: A semi-supervised learning approach," *Information and Software Technology*, vol. 52, no. 4, pp. 436 –445, 2010.

[23] E. Knauss, D. Damian, G. Poo-Caamao, and J. Cleland-Huang, "Detecting and Classifying Patterns of Requirements Clarifications," in *2012 20th IEEE International Requirements Engineering Conference (RE)*, 2012, pp. 251–260.

[24] Z. Kurtanović and W. Maalej, "Automatically classifying functional and non-functional requirements using supervised machine learning," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, 2017, pp. 490–495.

[25] Y. Ko, S. Park, J. Seo, and S. Choi, "Using Classification Techniques for Informal Requirements in the Requirements Analysis-supporting System," *Information and Software Technology*, vol. 49, no. 11, pp. 1128 –1140, 2007.

[26] J. Slankas and L. Williams, "Automated Extraction of Non-functional Requirements in Available Documentation," in *2013 1st International Workshop on Natural Language Analysis in Software Engineering (NaturaLiSE)*, 2013, pp. 9–16.

[27] E. Frank and R. Bouckaert, "Naïve Bayes for Text Classification with Unbalanced Classes," *Knowledge Discovery in Databases: PKDD 2006*, pp. 503–510, 2006.

[28] D. Klein and C. D. Manning, "Accurate Unlexicalized Parsing," in *Proceedings of the 41st annual meeting of the association for computational linguistics*, 2003.

[29] D. M. Blei, "Probabilistic Topic Models," *Commun. ACM*, vol. 55, no. 4, pp. 77–84, 2012.

[30] X. Yan, J. Guo, Y. Lan, and X. Cheng, "A Biterm Topic Model for Short Texts," in *Proceedings of the 22Nd International Conference on World Wide Web*, ser. WWW '13, ACM, 2013, pp. 1445–1456.

[31] D. D. Lewis, "Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval," in *Machine Learning: ECML-98: 10th European Conference on Machine Learning Chemnitz, Germany, April 21–23, 1998 Proceedings*, C. Nédellec and C. Rouveirol, Eds. Springer Berlin Heidelberg, 1998, pp. 4–15.

[32] M. Mohri, "Finite-state Transducers in Language and Speech Processing," *Comput. Linguist.*, vol. 23, no. 2, pp. 269–311, 1997.

[33] C. Cortes, P. Haffner, and M. Mohri, "A machine learning framework for spoken-dialog classification," in *Springer Handbook of Speech Processing*, J. Benesty, M. M. Sondhi, and Y. A. Huang, Eds. Springer Berlin Heidelberg, 2008, pp. 585–596.

[34] N. Cancedda, E. Gaussier, C. Goutte, and J.-M. Renders, "Word-sequence Kernels," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1059–1082, 2003.

[35] M. Damashek, "Gauging similarity with n-grams: Language-independent categorization of text," *Science*, vol. 267, no. 5199, pp. 843–848, 1995.

[36] C. S. Leslie, E. Eskin, A. Cohen, J. Weston, and W. S. Noble, "Mismatch string kernels for discriminative protein classification," *Bioinformatics*, vol. 20, no. 4, pp. 467–476, 2004.

[37] A. Ben-Hur and W. S. Noble, "Kernel methods for predicting protein-protein interactions," *Bioinformatics*, vol. 21, no. suppl$_1$, pp. i38–i46, 2005.

[38] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins, "Text Classification Using String Kernels," *Journal of Machine Learning Research*, vol. 2, pp. 419–444, 2002.

[39] C. C.P.H. M. Mohri, "Rational Kernels," *International Journal of Foundations of Computer Science*, vol. 14, no. 6, pp. 957–982, 2003.

[40] P. Pecina, "Lexical Association Measures and Collocation Extraction," *Language Resources and Evaluation*, vol. 44, no. 1, pp. 137–158, 2010.

[41] Z. S. Harris, "Mathematical structures of language," 1968.

[42] P. Goyal, L. Behera, and T. M. McGinnity, "A Context-Based Word Indexing Model for Document Summarization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 8, pp. 1693–1705, 2013.

[43] D. Jurafsky and J. H. Martin, *Speech and language processing*. Pearson London: 2014, vol. 3.

[44] C.-Y. Lin and E. Hovy, “Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics,” in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, ser. NAACL ’03, Association for Computational Linguistics, 2003, pp. 71–78.

[45] S. Katz, “Estimation of probabilities from sparse data for the language model component of a speech recognizer,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, no. 3, pp. 400–401, 1987.

[46] I. H. Witten and T. C. Bell, “The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression,” *IEEE Transactions on Information Theory*, vol. 37, no. 4, pp. 1085–1094, 1991.

[47] R. Kneser and H. Ney, “Improved Backing-off for M-gram Language Modeling,” in *1995 International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, 1995, 181–184 vol.1.

[48] H. Ney, U. Essen, and R. Kneser, “On structuring probabilistic dependences in stochastic language modelling,” 1, vol. 8, 1994, pp. 1 –38.

[49] A. K. Massey, J. Eisenstein, A. I. Antn, and P. P. Swire, “Automated Text Mining for Requirements Analysis of Policy Documents,” in *2013 21st IEEE International Requirements Engineering Conference (RE)*, 2013, pp. 4–13.

[50] J.-M. Davril, E. Delfosse, N. Hariri, M. Acher, J. Cleland-Huang, and P. Heymans, “Feature Model Extraction from Large Collections of Informal Product Descriptions,” in *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2013, ACM, 2013, pp. 290–300.

[51] E. Boutkova and F. Houdek, “Semi-automatic Identification of Features in Requirement Specifications,” in *2011 IEEE 19th International Requirements Engineering Conference (RE)*, 2011, pp. 313–318.

[52] E. Guzman and W. Maalej, “How do users like this feature? a fine grained sentiment analysis of app reviews,” in *2014 IEEE 22nd International Requirements Engineering Conference (RE)*, 2014, pp. 153–162.

[53] T. Johann, C. Stanik, A. M. A. B., and W. Maalej, “Safe: A simple approach for feature extraction from app descriptions and app reviews,” in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, 2017, pp. 21–30.

[54] J. Vincent, *RT Essentials*. ” O’Reilly Media, Inc.”, 2005.

[55] A. J. Viera, J. M. Garrett, *et al.*, “Understanding Interobserver Agreement: The Kappa Statistic,” *Fam Med*, vol. 37, no. 5, pp. 360–363, 2005.

[56] J. R. Landis and G. G. Koch, “The Measurement of Observer Agreement for Categorical Data,” *biometrics*, pp. 159–174, 1977.

[57] V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals,” in *Soviet physics doklady*, vol. 10, 1966, pp. 707–710.

[58] U. Y. Nahm and R. J. Mooney, “Text mining with information extraction,” in *Proceedings of the AAAI 2002 Spring Symposium on Mining Answers from Texts and Knowledge Bases*, 2002.

[59] E. S. Ristad and P. N. Yianilos, “Learning String-edit Distance,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 5, pp. 522–532, 1998.

[60] D. M. Berry, “Evaluation of tools for hairy requirements and software engineering tasks,” in *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, IEEE, 2017, pp. 284–291.

[61] T. Saracevic, “Evaluation of evaluation in information retrieval,” in *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR ’95, ACM, 1995, pp. 138–146.

[62] A. Ferrari, G. O. Spagnolo, and S. Gnesi, “PURE: A Dataset of Public Requirements Documents,” in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, 2017, pp. 502–505.

[63] I. Feinerer and K. Hornik, “tm: Text Mining Package,” *R package version 0.5-7.1*, vol. 1, no. 8, 2012.

[64] D. Meyer, K. Hornik, and I. Feinerer, “Text Mining Infrastructure in R,” *Journal of statistical software*, vol. 25, no. 5, pp. 1–54, 2008.

[65] M. F. Porter, “An Algorithm for Suffix Stripping,” *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[66] M. Mohri and M. Riley, “An Efficient Algorithm for the N-Best-Strings Problem,” in *Seventh International Conference on Spoken Language Processing*, 2002.

[67] T. H. Cormen, *Introduction to Algorithms*. MIT press, 2009.

[68] J. Laurikkala, “Improving identification of difficult small classes by balancing class distribution,” in *Artificial Intelligence in Medicine*, S. Quaglini, P. Barahona, and S. Andreassen, Eds., Springer Berlin Heidelberg, 2001, pp. 63–66.

[69] C.-C. Chang and C.-J. Lin, “LIBSVM: A Library for Support Vector Machines,” *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, 2011, ISSN: 2157-6904.

[70] M. B. Wilk and R. Gnanadesikan, “Probability plotting methods for the analysis for the analysis of data,” *Biometrika*, vol. 55, no. 1, pp. 1–17, 1968.

[71] Y. Elrakaiby, A. Ferrari, P. Spoletini, S. Gnesi, and B. Nuseibeh, “Using Argumentation to Explain Ambiguity in Requirements Elicitation Interviews,” in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, 2017, pp. 51–60.

[72] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, “Understanding data augmentation for classification: When to warp?” In *Digital Image Computing: Techniques and Applications (DICTA), 2016 International Conference on*, IEEE, 2016, pp. 1–6.

[73] S. F. Chen, D. Beeferman, and R. Rosenfeld, “Evaluation metrics for language models,” 1998.

[74] K. Zechner and A. Waibel, “Minimizing Word Error Rate in Textual Summaries of Spoken Language,” in *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, ser. NAACL 2000, Association for Computational Linguistics, 2000, pp. 186–193.