


Using semantic roles to improve text classification in the requirements domain

Alejandro Rago^{1,2}  · Claudia Marcos^{1,3} ·
J. Andres Diaz-Pace^{1,2}

© Springer Science+Business Media B.V., part of Springer Nature 2017

Abstract Engineering activities often produce considerable documentation as a by-product of the development process. Due to their complexity, technical analysts can benefit from text processing techniques able to identify concepts of interest and analyze deficiencies of the documents in an automated fashion. In practice, text sentences from the documentation are usually transformed to a vector space model, which is suitable for traditional machine learning classifiers. However, such transformations suffer from problems of synonyms and ambiguity that cause classification mistakes. For alleviating these problems, there has been a growing interest in the semantic enrichment of text. Unfortunately, using general-purpose thesaurus and encyclopedias to enrich technical documents belonging to a given domain (e.g. requirements engineering) often introduces noise and does not improve classification. In this work, we aim at boosting text classification by exploiting information about semantic roles. We have explored this approach when building a multi-label classifier for identifying special concepts, called domain actions, in textual software requirements. After evaluating various combinations of semantic roles and text classification algorithms, we found that this kind of semantically-enriched data leads to improvements of up to 18% in both precision and recall, when compared to non-enriched data. Our enrichment strategy based on semantic roles also allowed

✉ Alejandro Rago
arago@exa.unicen.edu.ar

✉ Claudia Marcos
cmarcos@exa.unicen.edu.ar

✉ J. Andres Diaz-Pace
adiaz@exa.unicen.edu.ar

¹ ISISTAN Research Institute, UNICEN University, Tandil, Argentina

² CONICET, Buenos Aires, Argentina

³ CIC, Buenos Aires, Argentina

classifiers to reach acceptable accuracy levels with small training sets. Moreover, semantic roles outperformed Wikipedia- and WordNET-based enrichments, which failed to boost requirements classification with several techniques. These results drove the development of two requirements tools, which we successfully applied in the processing of textual use cases.

Keywords Text classification · Natural language processing · Knowledge representation · Semantic enrichment · Use case specification

1 Introduction

Over the last decade, Machine Learning techniques have been widely used to help end users categorize documents, identify topics and extract concepts from textual documentation (Kelleher et al. 2015) in domains as diverse as engineering, business and medicine. In particular, in the Requirements Engineering (RE) domain (Nazir et al. 2017), researchers have capitalized on text classification techniques for automating various analyses such as: identifying inconsistencies (Kamalrudin et al. 2011), finding redundancy (Falessi et al. 2013), mapping text to code (Diamantopoulos et al. 2017), and highlighting deficiencies and defects in textual requirements (Femmer et al. 2017; Rosadini et al. 2017), among others. To perform these analyses and obtain good results, it is necessary to comprehend the text and the underlying semantics encoded in the text (Mahmoud and Carver 2015). A typical example in software requirements such as use cases are domain-specific terms or phrases (Roth et al. 2014), which only make sense in the specialized context of the specification. For instance, interactions in use cases described as “register data” and “save information” are equivalent in their meaning (write to a non-volatile medium), even though the words *register* and *save* are not synonyms from a (general) semantic perspective.

Traditionally, text classification techniques use the “bag-of-words” approach (Kelleher et al. 2015) to detect domain-specific information. This approach transforms sentences to a vector space model (VSM) that consists of weighted terms. The VSM then enables the application of well-known classifiers, such as: decision trees, naïve-bayes, or support vector machines, among others. However, there are two main problems with VSM. First, it ignores semantical relations among the terms because it operates by comparing keywords (e.g., abbreviations, synonyms, hypernyms or domain-specific relationships are not considered) (Tommasel and Godoy 2014). Second, classifiers trained with VSM are dependent on the terms of the dataset, and thus, they might not work well with unforeseen instances if the training data is not large (Bai et al. 2010).

To overcome these problems, several researchers have looked at enriching text with concepts derived from encyclopedias or structured thesaurus, like Wikipedia and WordNET (Wang et al. 2009; Bai et al. 2010). The key idea is to exploit the relations between concepts in order to improve classification models. However, in our experience with automated processing of RE documents (e.g., use case specifications) (Rago et al. 2013, 2016c), the semantic enrichment of those

documents with concepts from general-purpose knowledge sources (Mahmoud and Carver 2015) might be counterproductive, because they tend to introduce noise that hinders the construction of good classification models (Egozi et al. 2011). In practice, requirements are written with a “special” lexicon for the sake of stakeholders’ communication (Kamalrudin et al. 2011), using specific terminology to convey interactions between an actor and the system and limiting the number of words to express such behaviors (e.g., *write*, *store* and *save* mean the same in a use case specification but not in a dictionary). For this reason, abusing of semantic enrichment with “general” concepts might reduce the discerning power of technical terms and ultimately degrade the performance of a classifier (Mansuy and Hilderman 2006).

Due the above limitations, our research pursued three main objectives. The first objective is to build a representation to categorize requirements at the sentence level, capable of capturing their functional intention and handling ambiguity at different abstraction levels. The second objective is to explore the classification performance of traditional semantic enrichment techniques, based on knowledge resources such as Wikipedia and WordNET. The third objective is to provide an alternative enrichment technique based on the semantic information conveyed by the predicates of a sentence.

In this context, we report on the development of a classifier for identifying special concepts, called *domain actions*, in textual requirements documents. A domain action (DA) is an abstraction that captures the intention of a given sentence in the context of use cases and provides a deeper understanding of its semantics, for instance, for text categorization purposes. We have defined a hierarchy of DAs and applied it to build multi-label classifiers trained with previously tagged requirements. A novel aspect of our approach is the semantic enrichment of (requirements) datasets with information about *semantic roles*, which is added to the text (instances) via semantic role labeling (SRL) techniques (Szu-ting 2015). The evaluation of several classifier configurations with real requirements showed noticeable precision/recall improvements due to the SRL enrichment. Moreover, the classifiers trained with enriched data learned better prediction models with a modest number of instances (compared to training with plain data). The best-performing classifier—a combination of binary relevance and support vector machine—was then employed as part of two RE analysis tools (Rago et al. 2016a, b), improving their detection capabilities.

The contributions of this work are two-fold. First, we describe a semantic enrichment technique for textual documents that reduces the learning process of a variety of classifiers and boosts their performance. Our results within the RE domain led to precision/recall gains up to 15%. Furthermore, the evaluation revealed that Wikipedia and WordNET were not very effective for improving performance of the classifiers. Second, we present a model of RE-specific concepts that fits with our enrichment technique and supports requirements analyses. We believe that this strategy can be replicated in other engineering domains [e.g., hardware design (Rago et al. 2016c)] for improving text processing analyses.

The rest of the article is organized as follows. Section 2 introduces the categorization of textual requirements as a classification problem, and explain the

role of DAs and SRL in our approach. Section 3 describes our model of DAs in more detail. Section 4 reports on the results of evaluating classifiers with enriched and not-enriched datasets. Section 6 discusses related work. Finally, Sect. 7 gives the conclusions and outlines future lines of work.

2 Background

Analyzing textual requirements is a complex activity (Mund et al. 2015), mainly due to the expressivity and ambiguity of natural language. A challenge in the development of RE tools is how to detect similar (or related) concepts that recurrently appear across several requirements (Roth and Klein 2015; Ménard and Ratté 2016). Such concepts can be common behaviors (e.g., retrieving, obtaining or getting information) or shared properties of interest (e.g., performance considerations that impact on several sentences) which are expressed with a different wording but refer to the same conceptual action. When requirements are processed with automated techniques, a first task is to look at repeated terms and synonyms in the text, which is normally accomplished with natural language processing (NLP). Another task is to try to “understand” (at least, in an approximate way) the intention of requirements. Intention refers to the meaning of a requirement in the particular context of the system being developed, and thus, has semantic connotations.

Given a set of concepts C_1, C_2, \dots, C_n (classes) that, we assume, are associated to some requirements of a set R_1, R_2, \dots, R_m , we can define a semantic analysis of the requirements as a multi-label classification problem. The goal is to categorize each requirement as belonging to one or more classes. These classes can symbolize concepts such as quality-attribute constraints, reusable functionality, interaction types or requirements deficiencies (e.g., using passive voice or negative statements), among others. Depending on the kind of categorization, the sentences from the requirements might need to be associated with multiple concepts. In these cases, a multi-label classifier might be able to tag an individual requirement with more than one concept (e.g., identifying multiple quality attributes). A graphical schema of the classification problem is depicted in Fig. 1. We assume that the classifier is learned based on a training dataset, which consists of requirements already labeled with their corresponding concepts.¹ Then, this classifier can predict likely relevant concepts for new instances of requirements. As usual, each instance is initially converted to a VSM representation, before being processed by the classifier. At this point, it is possible to use a semantic resource such as WordNet or Wikipedia (see Fig. 1) for enriching the instances with the end goal of improving the accuracy of text classification. Nonetheless, traditional text classifiers seldom rely on enrichment techniques in technical domains because of their poor performance.

For instance, let us suppose a simple requirement sentence, and the RETRIEVAL and EXTERNAL COMMUNICATION concepts, as illustrated in Fig. 2. The requirement sentence is an excerpt from a use case scenario, whereas a concept typifies different kinds of interactions between a system and its actors. In this context, we may need

¹ For simplicity, a supervised approach is considered.

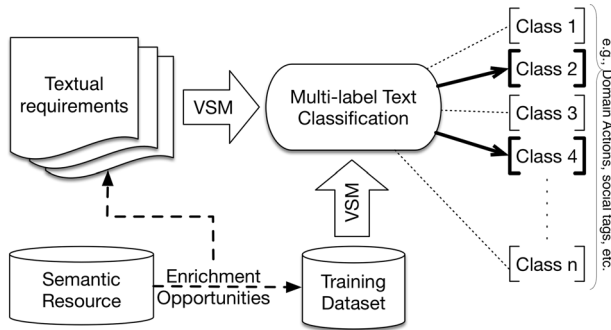


Fig. 1 Graphical schema of requirements classification

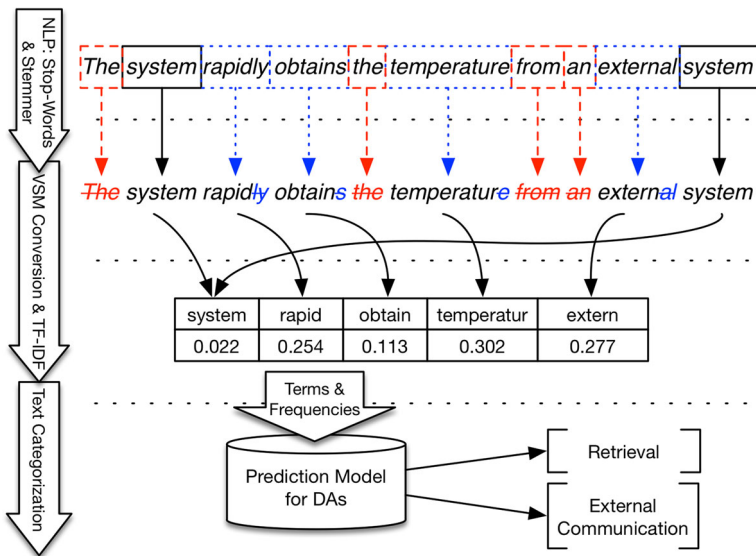


Fig. 2 Processing a textual requirement with a traditional VSM approach

an application capable of automatically identifying such concepts in the text by means of a prediction model in order to better understand their meaning. Given the complexity of the sentence, a single requirement can be associated with one or more types of interactions, justifying the need for a multi-label classifier. For this purpose, a basic NLP pipeline (e.g., stop-words and stemmer) can be used to extract the relevant terms for the vector. The dashed arrows below the sentence depict either removal or trimming operations as the result of applying a stop-words or stemming technique, respectively. Furthermore, the term frequency can be computed using TF-IDF (Kelleher et al. 2015). In Fig. 3, frequencies are shown in the second row of the table below the pre-processed terms. This representation reflects the importance of a word in a collection of documents and can be used as a weighting factor for

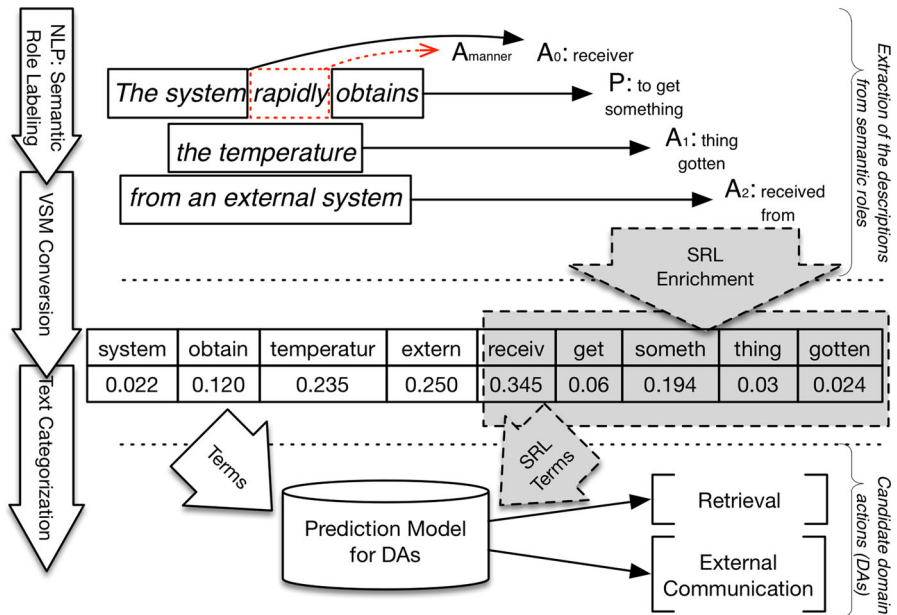


Fig. 3 Processing a textual requirement with an SRL-enriched VSM approach

classification purposes. The importance of a word increases proportionally to the number of times such a word appears in the document, but is offset by the frequency of the word in the collection, helping to adjust for the fact that some words appear more frequently in general. In use case specifications, for instance, the word “system” receives a low score with TF-IDF because it is normally repeated across all the scenarios.

2.1 Domain actions as requirements concepts

When analysts inspect requirements, it is common for them to spot repeated concepts (or patterns) along the textual specifications. This is particularly true in use cases (Hull et al. 2014), which are a RE approach to capture a behavioral contract between the system and its external actors by means of “usage” scenarios. Use cases have two interesting characteristics regarding text processing: (1) the lexicon (i.e., terminology) used to describe behavior is generally limited; and (2) the “story-telling” of scenarios often exhibits a common structure and employs semantically-similar concepts (e.g., inputting data or processing information, among others). These characteristics are often the result of recommended RE guidelines that analysts normally adhere to (Wiegiers and Beatty 2013).

A type of concept relevant to use cases is that of *domain action* (DA). A DA represents an abstraction of a recurrent interaction between the system and its actors, and has an unequivocal semantic meaning in the context of use cases (Sinha et al. 2010). A DA can refer to aspects of data management, user interactions, or

information displaying, among others. In practice, a use case consists of several steps (sentences), each of which can be viewed as (or reduced to) a DA. Table 2 shows several examples of DAs associated to steps of a use case. DAs are relevant to our work, as they are precisely the class labels for our classifier to categorize input requirements. In order to learn a classifier of DAs, an expert must take a large set of textual requirements covering different types of systems, and manually tag each requirement with the corresponding DAs. Furthermore, there are other subtleties when applying DAs to use cases, such as abstraction levels or overlapping classes, which are explained in Sect. 3.

2.2 Enriching text with semantic roles

Building a DA classifier with acceptable performance and applicable to different software domains is not straightforward. Even if a large dataset is available (which is hardly the case), the DA tagging process can be tedious and time-consuming for an expert, because she must consider the different writing styles of the analysts who produced the requirements. A way to cope with these problems is to enrich the text with semantic annotations, being as independent as possible from software domains or human writing styles. For instance, an expert might recognize that the words *sends* and *transmits* mean the same in the context of RE specifications. Unfortunately, using sources such as Wikipedia or WordNET for annotating use case (or even other types of requirements) is not always the best approach, because their concepts are too general and end up “confusing” the classifier (Kehagias et al. 2003; Mansuy and Hilderman 2006; Tommasel and Godoy 2014).

The alternative approach that we investigate in this work is to enrich requirements with *semantic roles* descriptions, which are obtained via semantic role labeling (SRL) (Szu-ting 2015). SRL is the automated classification of text with labels defined in the Proposition Bank (PropBank) (Palmer et al. 2005), a linguistic resource created for adding a layer of predicate-argument information (also referred to as semantic role labels) to syntactic structures within a sentence. Semantic roles tackle a limitation of traditional syntactic analyses, which fail to capture the true meaning of a sentence and its constituents (Palmer et al. 2010). Understanding what the participants are in a sentence and what event is described is very important for computers to make effective use of the information codified in the text.

In PropBank, a set of underlying semantic roles are defined on a verb by verb basis (i.e., the predicate). The roles of each verb are labeled with a special notation, consisting of numbered arguments (A_0 , A_1 , A_2 , A_3 , A_4 and A_5) and adjunct arguments ($A_{Location}$, $A_{Temporal}$, $A_{Negation}$, A_{Manner} , $A_{Direction}$, A_{Modal} , among others). Numbered arguments are specific to each particular verb. For consistency with other taxonomies, A_0 is generally the “Agent” of a sentence whereas A_1 is generally the “Patient” or “Theme”. No other generalizations can be made for the other numbered arguments, that is, those that are specific for each verb. Adjunct arguments are less restrictive and can be applied to any verb. A set of roles associated to a distinct usage of a predicate is called a roleset (or frameset). If a verb is polysemic, it might require having multiple rolesets for each sense. For SRL, this means that a disambiguation task is often performed to select the correct roleset.

Table 1 Example of semantic role labeling

Roleset	$accept_{01} \rightarrow$ “take willingly” $A_0 \rightarrow$ “acceptor” $A_1 \rightarrow$ “thing accepted” $A_2 \rightarrow$ “accepted from” $A_3 \rightarrow$ “attribute”
Example	$[A_0$ The system] $[A_{Modal}$ would] $[A_{Negation}$ not] <i>accept</i> $[A_1$ credentials] $[A_2$ from users who are on leave of absence]

SRL implementations rely on both semantic and syntax information to distinguish between rolesets (for instance, the number of arguments in the sentence). During the tagging procedure, predicates and arguments are often enriched with a “description” field, which captures the meaning of each role for a particular roleset. As an example, let us consider the results of an SRL analysis performed on a sentence extracted from a requirements document, as shown in Table 1. The sentence is divided into six semantic roles, four verb-specific arguments and two adjunct arguments surrounding the main predicate “accept”.

In our work, we leverage on SRL information as a middle ground between requirements and their corresponding DAs, in order to obtain a classifier that provides a good performance when trained with a dataset of moderate size. The text of SRL descriptions can be incorporated to the VSM, as depicted in Fig. 3 (gray area on the right). This process is done by analyzing both the predicate of a sentence and each of its arguments, adding the words that are part of their respective description fields to the word vectors (except for stop-words). This feature helps classification because some semantic roles are shared between words (Selvaretnam and Belkhatir 2016). If a sentence contains multiple predicates, then the analysis is performed one verb at a time, only considering arguments belonging to that individual roleset and ignoring the rest of the sentence.

For instance, let us have a look at sentences #1 and #7 in Table 2. In such sentences, the words used for describing the system behavior are different. Furthermore, despite the fact that the two interactions play a similar function in the scenarios (i.e., to fetch some data), the main verbs *retrieve* and *obtain* are not synonyms according to the thesaurus. However, both the *retrieve* and *obtain* predicates (see the bold words in rows 1–2 in Table 3) have arguments defined as *thing gotten* and *received from*, respectively. An analysis of sentences #4 and #6 exhibits the same pattern, where the verbs *clear out* and *delete* are not synonyms but mean the same in the context of use cases (i.e., to remove data from the system). Again, this relation can be revealed by analyzing the arguments of these predicates, in which the words “remove” and “entity” are shared by the descriptions (see rows 3–4 in Table 3).

SRL basically recognizes predicates and their associated arguments in a sentence, and also classifies those arguments into specific roles (e.g., the agent, the patient, the

Table 2 Association of DAs to use-case sentences

USE-CASE SENTENCE	LOW-LEVEL DA(s)	HIGH-LEVEL DA(s)
1. The system rapidly <i>obtains</i> the temperature from an <i>external system</i>	RETRIEVAL & EXTERNAL COMMUNICATION	DATA & PROCESS
2. The system <i>saves</i> the complaint	CREATE	DATA
3. The system <i>ensures</i> the health unit information is <i>consistent</i>	CALCULATION & VERIFICATION	PROCESS
4. The system <i>clears out</i> the schedule	DELETE	DATA
5. The administrator <i>types in</i> the report number	ENTRY	INPUT/OUTPUT
6. The system <i>deletes</i> the personal record of the patient	DELETE	DATA
7. The system <i>retrieves</i> the list of enrolled courses	RETRIEVAL	DATA

Table 3 Semantic role descriptions of predicates

WORD	PROPBANK ID	SEMANTIC ROLE DESCRIPTIONS			
		PREDICATE	A0	A1	A2
<i>retrieves</i>	retrieve.01	get back	receiver	thing gotten	received from
<i>obtain</i>	obtain.01	get	receiver	thing gotten	received from
<i>clears out</i>	clears.05	remove , leave	clearer	entity leaving or removed	place left
<i>deletes</i>	delete.01	remove	entity removing	thing being removed	removed from

manner, the time, the location, etc.) (Szu-ting 2015). A semantic role identifies the part played by a fragment of the sentence in the context of the main verb. The knowledge held by predicates and arguments are very useful for removing irrelevant information, since it permits to answer questions such as “Who did What to Whom, How, When and Where?” (Palmer et al. 2010). Several tools exist to perform SRL over textual sentences, which internally rely on classifiers trained to predict semantic roles using the PropBank corpus (Palmer et al. 2005). This corpus contains text annotated with semantic propositions, including predicate and argument relations.

3 Modeling domain actions in use cases

Concepts similar to DAs have been discussed by other researchers (Sinha et al. 2010; Kamalrudin et al. 2011; Jurkiewicz and Nawrocki 2015). However, in the context of requirement analysis, these conceptual representations have a number of limitations, namely: (1) the abstraction level is fixed, (2) only one concept per sentence is allowed, and (3) the techniques employed for identifying the concepts do not analyze the context of the verbs.

The first limitation precludes comparisons among DAs at different abstraction levels, which can be helpful in some RE analyses. For example, in Table 2, sentences #1, #2 and #4 are tagged with distinct DA classes (2nd column). These classes are RETRIEVAL, CREATE and DELETE, which refer to the recovery of information, the creation of persistent records, and the removal of information, respectively. An analyst could recognize here an implicit relation among the sentences, because the three DAs denote the manipulation of information. For this reason, we included a more abstract DA class called DATA that reflects this parent-child relation (3rd column).

Regarding the second limitation, we believe that a flexible approach allowing more than one DA per sentence can help to deal with ambiguity. In several projects, a requirement has to be tagged with various DAs because it intermingles two or more behaviors within the same sentence. For example, let us consider sentences #1 and #3. In these two sentences, the dominant DAs are RETRIEVAL and CALCULATION,

referring to the recovery of some information and a computation of some sort. However, a more detailed analysis of the sentences reveals that there might be other DAs at work (see the 2nd and 3rd columns of Table 2). In sentence #1 there is a mention to an *external system*, indicating a EXTERNAL COMMUNICATION DA that is related to the RETRIEVAL DA. In sentence #3, the presence of the word *consistence* could indicate a checking associated with a VERIFICATION DA.

At last, for the third limitation, we argue that the consideration of the context in which verbs occur is also important for addressing ambiguity. An accurate classification of DAs requires the identification of the parts of a sentence affected by the main predicate. For example, the association of sentence #1 with a EXTERNAL COMMUNICATION DA would have not been possible without analyzing contextual words such as *external*. However, the analysis of the word *rapidly* in the same sentence, usually related to system calculations, might deviate the classifier towards other kinds of DA classes. In this regard, we can take advantage of SRL to filter irrelevant terms and include only those arguments that better describe DAs.

3.1 Proposed hierarchy of DAs

To overcome the limitations above, we created a hierarchical model of DAs that allows us to capture the semantics of use cases. The hierarchical feature contributes to improve the performance of text classification (Tsoumakas et al. 2010). Furthermore, the categorization of DAs was treated as a multi-label classification problem in order to deal with ambiguities (as discussed in the previous section for the second limitation) (Kang et al. 2015). Particularly, the aspect of ambiguity for requirements that cannot be easily categorized into a single class has been acknowledged in previous works (Sinha et al. 2010; Sengupta et al. 2015) but not addressed yet. Our model of DAs initially derived from Sinha's work (Sinha et al. 2010), because the abstraction level of their classes fitted well with several semantic analyses for use cases.

We refined Sinha's classes by removing some system-specific classes not suitable for our purposes (e.g., DELEGATE and BROWSE) and renaming some others. We also incorporated classes for recognizing control flow sentences, which contain terminology that is specific to use cases. Overall, we defined 25 DA classes and arranged them in a three-level hierarchy, as depicted in Fig. 4. The upper level of the hierarchy groups DAs for semantically-similar use-case interactions, such as INPUT/OUTPUT, DATA, PROCESS and USE CASE. The middle level introduces more specific information. For example, the top-level DATA DA is refined by two mid-level DAs: READ and WRITE. The bottom-level DAs cover concrete interactions such as: ENTRY, SELECTION, DISPLAY, or NOTIFICATION, among others. A brief description of each DA label is given in “[Appendix](#)” section.

3.2 Compiling a dataset of requirements + DAs

Despite some notions of DAs and the classification of these concepts existed in the literature (Sinha et al. 2010; Kamalrudin et al. 2011; Sengupta et al. 2015), it was not possible for us to conduct a comparative assessment for two reasons. First, the

datasets used in these works were not always publicly available for download. Second, existing datasets would require an extensive adaptation to map the concepts to our DAs and fit the hierarchical and multi-label representation, ultimately biasing the results. We instead defined a new dataset (i.e., a training corpus for different classifiers) using requirements specifications from real software systems that were clearly written (in accordance to the recommended best practices for use cases) and asked a group of experts to tag the sentences according to the DAs they detected in the text. As selection criteria for this corpus, we considered parameters such as application domain (i.e., type of project), programming paradigm, requirements methodology, or availability of the project artifacts, written quality of the specifications and size of the project. Basically, we aim at choosing object-oriented projects from diverse software domains documented with use cases and publicly available for making comparisons. Moreover, we targeted projects clearly written and without grammatical, syntactical or lexical mistakes which may hurt classification tasks. At last, we ensured that the resulting corpus contained a representative number of use case steps associated with every DAs defined in the hierarchy.

For the dataset, we took use case specifications from three real systems,² namely: Health Watcher System³ (HWS), Course Registration System (CRS)⁴ and Collegiate Sports Paging System⁵ (CSPS). HWS is a Web-based system for consulting health notices and registering sanitary complaints. CRS is a system that operates in a university intranet and allows students to enroll in courses and professors to report students' grades. CSPS is a distributed system that notifies students' cell phones and pagers about last-minute sports news. Table 4 shows an overview of the projects and their requirements documentation. The first five rows detail the domain type of the project, number of use cases, textual length, number of sentences and total count of words. The sixth row contains a word cloud that illustrates the most frequent words in each specification.

The preparation of the specifications consisted of manually extracting the relevant text from the original files, preserving the format of the text in the form of enumerated lists and indentation. Tables and images in the original use cases were omitted since they did not contain valuable information from a classification point of view. We also corrected minor grammatical and spelling mistakes in the scenarios, because they might hinder NLP analyses and text classification tasks. The text of the use cases was then broken into sentences, removing irrelevant sections (e.g., overview, included use cases and priorities, among others), and arranged in a ARFF file format where each sentence tagged with a DA is represented as an individual instance. In total, ~ 900 instances were gathered from the functional scenarios of use cases. Afterwards, these sentences were manually tagged with DAs. The tagging procedure was entrusted to a group of Ph.D. students in Computer

² The original requirements documents of the case studies can be downloaded from: <http://www.alejandrorago.com.ar/files/assets/dataset-Source.zip>.

³ <http://www.comp.lancs.ac.uk/~greenwop/tao/>.

⁴ http://sce.uhcl.edu/helm/RUP_course_example/courseregistrationproject/indexcourse.htm.

⁵ <http://sce.uhcl.edu/helm/rationalunifiedprocess/examples/csports/>.

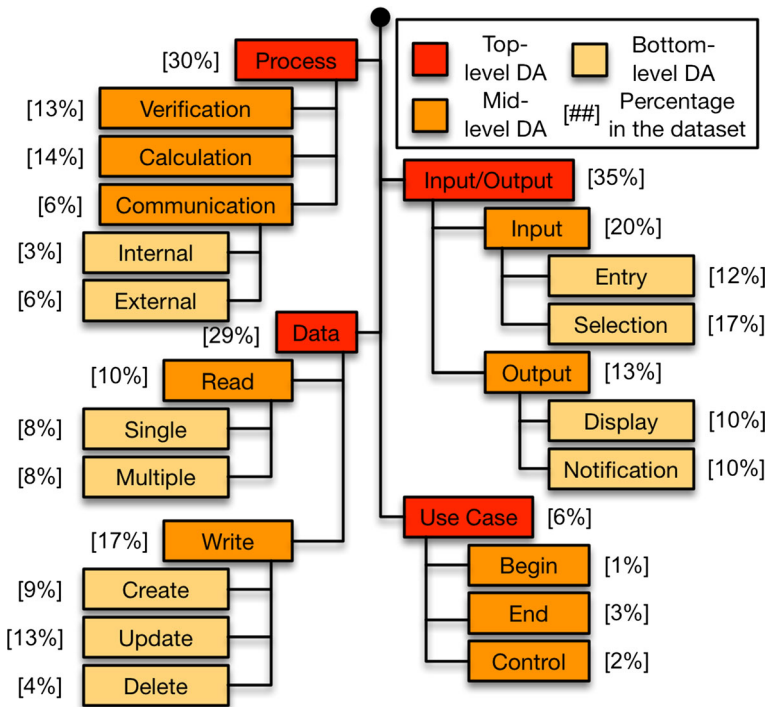


Fig. 4 Hierarchy of domain actions

Science from UNICEN University. We chose doctorate students over pre-graduate students and industry practitioners because they have a strong background (i.e., academic) in requirements processes and its deliverables, they are well motivated and willing to participate in research experiments, as well as they are not biased by previous experiences (e.g., bad requirement practices) acquired in industrial projects. The subjects were given a short training on our model of DAs, and then each subject began the analysis of sentences, assigning one or more DAs per sentence, based on her own criterion. At the end of the exercise, we had a meeting with all the subjects in order to consolidate the individual results and produce the final dataset. The last row of Table 4 shows the five most frequent DAs in each project and its respective distribution percentage.

Although we did not conduct an inter-agreement study of the exercise, we observed that the subjects reached a consensus for the majority of the requirements, that is, they labeled the requirements with the same DAs. Nonetheless, we noticed that for some instances with ambiguity, the subjects had different opinions regarding the DAs involved. In those cases, the solution reached by the subjects was to label the requirements with more than one DA, introducing multi-label instances

in the final dataset.⁶ Afterwards, we conducted an inspection for checking labeling mistakes or ironing out confusions made by the subjects, ensuring the correctness of the dataset. Only a few number of instances (~ 40) had to be revised by the subjects, and those problems were generally solved by removing/adding some DAs. We should note that the size of our dataset is relatively small for classification tasks; however, other RE works have also used datasets of similar size for this purpose and achieved reasonable results (Sinha et al. 2010; Casamayor et al. 2012; Roth and Klein 2015; Nguyen et al. 2015; Sengupta et al. 2015). Moreover, given that our goal is the identification of DAs, we argue that a well-balanced dataset composed of ~ 360 instances (40% of the current dataset) should suffice for training a good classifier.

We made a quantitative analysis of the dataset, and observed that the overall distribution of DAs was balanced (see percentages per level in Fig. 4). Every DA class was used at least in one instance. The number of instances tagged with top-level DAs (i.e., abstract ones) was larger than that of mid-level and bottom-level DAs. This situation was expected because of the hierarchical organization of DAs. An analysis of DA distribution per level revealed the following facts. Instances tagged as INPUT/OUTPUT, DATA and PROCESS represented $\sim 94\%$ of top-level DAs. Instances tagged as USE-CASE were very few in the text ($\sim 6\%$), because they only appeared at the beginning, ending and flow bifurcations of the scenarios. Classes in the mid-level were evenly distributed, with each DA representing $\sim 10\text{--}20\%$ of the dataset. The most frequent classes were INPUT, WRITE and CALCULATION. At the bottom-level, instances were also tagged evenly, representing $\sim 8\text{--}15\%$ of the dataset. The most frequent DA in this level was SELECTION ($\sim 17\%$), whereas the least frequent ones were INTERNAL COMMUNICATION and EXTERNAL COMMUNICATION (~ 3 and $\sim 6\%$).

When analyzing the overlapping nature of our dataset, we observed a reasonable fraction of instances having multiple labels, which corroborated our view of the categorization of DAs as a multi-label problem. As we have previously stated, ambiguity in textual requirements is an expected phenomenon. For instance, Sinha et al. tackled this problem by using a dictionary in which every verb is associated to multiple DAs with a certain degree of confidence (Sinha et al. 2010). However, the requirements in their dataset are tagged with a single DA label/class per instance. Along the same line, Sengupta et al. represented the ambiguity of requirements using multiple labels (Sengupta et al. 2015). In particular, 26% of the verbs in the sentences compiled in their dataset have overlapping semantic categories. Trusting that the procedure described above to tag requirements is correct (it is a subjective task), approximately 16% of the instances in our dataset were simultaneously associated with 2–3 or more bottom-level DAs. This a direct result of the ambiguity of requirements sentences, which are often associated to multiple DAs. Given the hierarchical relations between DAs, we expected to find a large number of labels per instance. If we consider all three levels of the hierarchy at the same time, instances with 2 or 3 labels represented an 87% of the dataset. The rest of the instances were

⁶ The complete dataset can be found at <http://www.alejandrorigo.com.ar/files/assets/dataset-DomainActions.zip>.

Table 4 Fact sheet of the projects in the dataset

Feature	Project	CRS	CSPS
Domain	HWS	University Course Management & Grading Services	Sports Subscription & News Delivery
Use cases	Citizenship Healthcare Control & Health Notices Distribution	8 use cases	11 use cases
Textual length	9 use cases	20 pages	16 pages
Sentences	19 pages	291 sentences	295 sentences
Word count	378 sentences	3670 words	2207 words
Word cloud (not TF-IDF)			
Top-5 DAs (leafs)	CALCULATION: 13.65% VERIFICATION: 10.92% SINGLE RETRIEVAL: 10.58% DISPLAY: 9.21% NOTIFICATION: 8.87%	VERIFICATION: 14.28% SELECTION: 13.95% CALCULATION: 10.96% UPDATE: 8.64% ENTRY: 8.30%	CALCULATION: 19.70% SELECTION: 13.79% VERIFICATION: 12.80% UPDATE: 8.37% ENTRY: 6.89%

tagged with 4–6 or even 9 labels, because they are tagged with two or more top branches of the hierarchy.

4 Evaluation

Several tests were conducted in order to assess the performance of different classifiers of DAs. Our hypothesis is that enriching textual requirements with semantic roles can significantly improve the precision/recall of classifiers. The main objective of the evaluation is to determine the pros and cons of SRL-enriched over Plain-Text classification, as well as to compare these results with other enrichment strategies. In the following sections, we discuss the preparation of the dataset, the experimental methodology and the machine learning algorithms used in the evaluation. At last, we present the results and draw conclusions of the evaluation.

4.1 Dataset preparation

Before training the classifiers, the use-case sentences were converted to a suitable text format for the algorithms. Figure 5 illustrates the activities performed for deriving our datasets. Initially, the textual documentation of each project was pre-processed with stop-words and stemming filters (Phase #1) for all but SRL tasks, which do not work well with this kind of techniques and have to be applied after identifying semantic roles. The goal of this phase is to remove prepositions and articles because they have little value in classification tasks, and to reduce terms to their inflectional forms to have a unique representation for word families, respectively. For the sake of comparison, two datasets were initially built. The first dataset only included plain text (*Plain-Text*), whereas the second dataset was enriched with semantic roles (*SRL-Enriched*). In the *Plain-Text* dataset, we used the original text from the use cases, skipping the enrichment of the requirements. Therefore, the text of the sentence is transformed to a VSM representation, and the resulting words are weighted using a term frequency—inverse document frequency (TF-IDF) technique (Phase #3). Then, the dataset is persisted in ARFF, which is a suitable format for classifiers.

In the *SRL-Enriched* dataset, textual requirements are complemented by means of a lexical resource (see the greyed boxes of Phase #2). First, the original sentences (without pre-processing) are analyzed with a *Semantic Role Labeling* classifier, identifying the main predicates and their arguments. We used Mate-Tools⁷ to implement the SRL task (Björkelund et al. 2010), mainly because it is a well-documented project with an acceptable performance and accurate results. Nonetheless, more current tools might be used to replace this library in the future (e.g., Alchemy Lite, SENNA or SEMAFOR). Next, irrelevant information in the sentences is removed by using the results of the SRL. The goal of this activity is to take into account only those fragments of text in the sentences coming from predicates and arguments labeled with SRL as A_0 , A_1 or A_2 (Szu-ting 2015). Other

⁷ <https://code.google.com/p/mate-tools/>.

types of arguments, such as A_3 , A_4 , $A_{Temporal}$, $A_{Location}$, A_{Manner} and so forth are less frequent to appear in software specifications and commonly introduce details that can hinder text classification (Uysal and Gunal 2014). To support this claim, Table 5 summarizes a frequency analysis of the arguments with SRL in each case study. In addition, we also believe that the information held within adjunct arguments is not beneficial for identifying domain actions, because these type of arguments are not strictly specific to the predicate. For instance, some steps in the “submit grades” use case of CRS contained temporal information about (student) classes “completed in the previous semester”, which can be filtered by ignoring $A_{Temporal}$ arguments. Third, the description of the semantic roles present in each argument and predicate is appended to the instances as additional information. To this end, we search each “descriptor” label in PropBank⁸ roleset files, available as XML documents. If the predicate or any of the arguments in a sentence is ambiguous and can be mapped to multiple rolesets, we relied on Mate-Tools’ disambiguation features to select the most likely variant. Our implementation actually takes into account the predicate sense in order to determine argument senses. Finally, both the original and the enriched text goes through stop-words/stemmer filters, is transformed to term vectors and a TF-IDF technique is applied before saving the dataset in ARFF format (Phase #3) (Kelleher et al. 2015). We did not apply feature-selection algorithms to the resulting datasets (Badawi and Altincay 2014), because of their modest size.

In order to assess that SRL performs better than other lexical resources for improving text classifiers, we also defined four additional datasets containing WordNET and Wikipedia concepts (Phase #2). Basically, two extended datasets were created for each lexical resource. Initially, the text from the requirements is analyzed with WordNET and Wikipedia looking for matches in the respective lexical resource. The first and third dataset (*WordNET-Concepts* and *Wikipedia-Concepts*) were constructed by searching the words of the sentences in the lexical resources and replacing them with concept identifiers. This means to interchange words for synsets and article entries for WordNET and Wikipedia, respectively. If an individual word is not found in the resource, then that information is lost in the transformation. In WordNET, we also exploited relations between words, expanding each term with its synonyms by taking advantage of how a single concept can represent many words expressing the same meaning. Since losing information is often harmful for text classification, we also considered mixing the words with their corresponding concepts identifiers. This enrichment strategy derived in the second and fourth dataset (*WordNET-enriched* and *Wikipedia-enriched*). The idea is to add the concepts to the already existing words in a sentence, instead of replacing them. Naturally, enriching the text in this way can introduce overlapped information in the datasets, especially with WordNET. This is because many of WordNET synsets are linked to a single word. Fortunately, the classifiers we used are prepared for handling highly correlated features in the instances and not making mistakes in the classification. At last, concept-enriched instances are transformed with VSM and TF-IDF to an ARFF representation (Phase #3).

⁸ <http://verbs.colorado.edu/~mpalmer/projects/ace.html>.

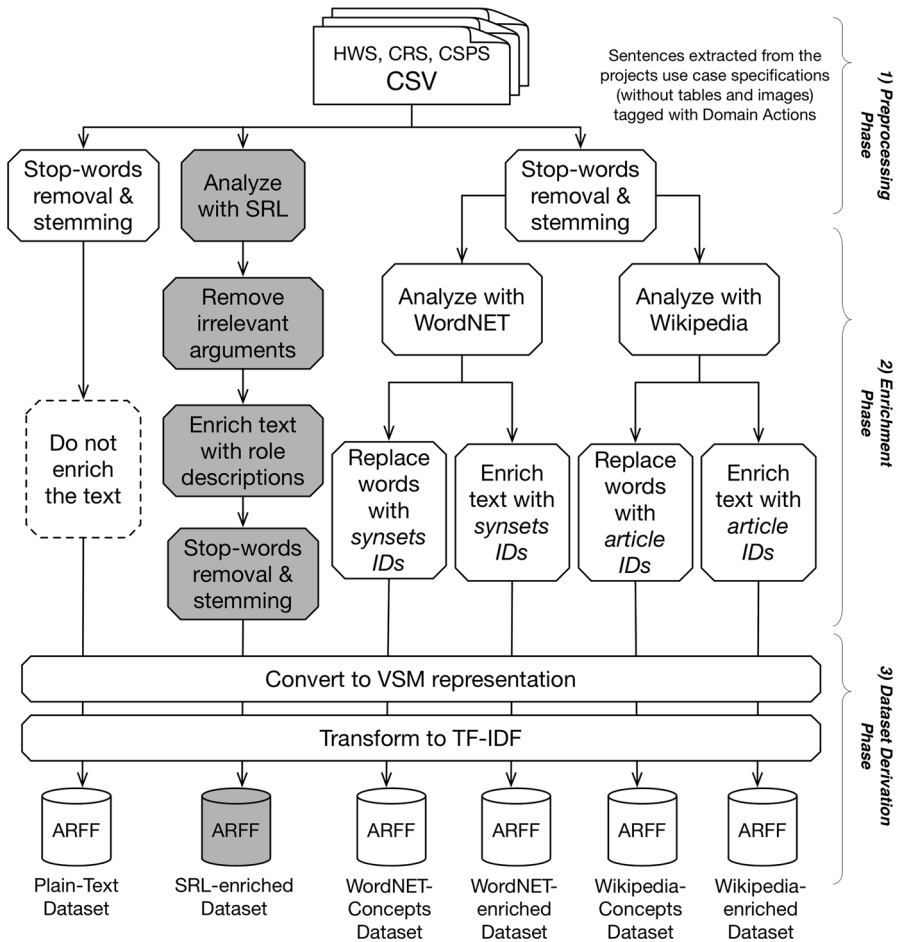


Fig. 5 Procedure for deriving the semantically-enriched datasets

We used the implementation of KATOA for transforming/enriching the text with concepts (Huang 2011). KATOA defines two filters built on top of Weka, called *StringToWordNetVector* and *StringToWikipediaVector*, that convert textual sentences to concept vectors. This idea has been previously explored in recent works (Huang et al. 2012). By using these filters, deriving *WordNET-Concepts* and *Wikipedia-Concepts* out of sentences is straightforward. We also made some modifications to the KATOA implementation to avoid the deletion of existing words in order to create the *WordNET-enriched* and *Wikipedia-enriched* datasets.

4.2 Experimental methodology

We investigated several combinations of supervised single-class classification algorithms with extensions to support multiple classes and labels. We tested three

algorithms for one-class classification implemented in Weka,⁹ namely: *decision trees* (J48), *naïve-bayes* (NB) and *support vector machines* (SMO). We chose these algorithms mainly because they are well-known, easy to use and have been extensively used in the past for text classification tasks. The J48 classifier builds a prediction model in the form of a tree structure composed of decision nodes and leaf nodes (i.e., the classes). Essentially, the classifier recursively breaks down the dataset into smaller subsets while, at the same time, a decision node is added to the tree. The NB classifier is based on the Bayes theorem and permits to categorize data by computing the a-posteriori probability of a class given a set of predictors. Despite its simplicity, NB usually performs well in text classification tasks (Zhang 2004). The SMO classifier is based on the concept that instances in a space model can be separated into classes by hyperplanes. Kernels are used for mapping complex planes from a space to simpler planes in another space. In our evaluation, we decided to use a popular radial basis function (RBF) kernel approximation based on the Euclidean distance between examples, since it has shown good results for classifying bag-of-words datasets.

The J48 and NB classifiers were configured using default parameters, because their performance is not significantly affected by their values in text classification tasks. In the case of SMO, different values of C and γ were considered, until finding optimal parameters for the classifiers. We explored two transformations implemented in Mulan¹⁰ to support multiple labels and classes, namely: *label powerset* (LP) and *binary relevance* (BR) (Zhang and Zhou 2014). Furthermore, since DAs are arranged in a hierarchical structure, we applied an algorithm called *hierarchical multilabel classifier* (HMC) to take into account father-child relations among DAs (Tsoumakas et al. 2010). A grid search for $C \in \{2^{-1}, 2^1, \dots, 2^9, 2^{11}\}$ and $\gamma \in \{2^{-9}, 2^{-7}, \dots, 2^1, 2^3\}$ was performed on both train and test sets (explained later), trying to find a good configuration of the classifier that does not over-fit or under-fit the data. As a result of the search, the optimal parameters obtained were ($C = 2^1, \gamma = 2^{-5}$) for SMO-LP and ($C = 2^5, \gamma = 2^{-5}$) for SMO-BR.

We analyzed classification results with example-based *precision* and example-based *recall* metrics (Zhang and Zhou 2014), which are commonly used in multi-label classification tasks. *Precision* gauges the fraction of labels correctly identified with respect to all the suggested labels, whereas *recall* gauges the fraction of labels correctly identified with respect to all the instances with such labels (Sokolova and Lapalme 2009).

We decided not to use a traditional k-fold cross validation, because the number of folds is tied to the split percentages (e.g., using 10 folds would mean training the classifier with 90% of the instances and testing it with the remaining 10%). Given the nature of DAs, we argue that a relatively low number of instances should be sufficient to correctly predict DAs on the rest of the dataset. Thus, we opted for a repeated sub-sampling validation with fixed splits (Kelleher et al. 2015). In this validation schema, the split ratio for the training and testing data is defined in advance (e.g., 20/80 or 30/70, respectively). Multiple sub-samples with that split

⁹ <http://www.cs.waikato.ac.nz/ml/weka/>.

¹⁰ <http://mulan.sourceforge.net/index.html>.

Table 5 Frequency of SRL arguments

Case study	A_0	A_1	A_2	A_3	A_4	A_{Tmp}	A_{Loc}	A_{Mnr}	A_{Adv}	A_{Neg}
HWS	120	291	47	2	9	8	24	9	6	3
CRS	212	371	43	9	1	50	15	18	37	6
CSPS	143	289	57	2	0	29	3	18	30	10
Total	475	951	147	13	10	87	42	45	73	19

ratio are then computed, ensuring that every instance is used for training and testing purposes. Specifically, we used several train/test splits $\{20/80, 30/70, 40/60, 50/50, 60/40\}$ and we chose ten sub-samples for each split ratio in order to obtain an accurate result. Once sub-samples are established, the model is fit to the training data and the precision/recall is assessed using both training and testing data. This allows us to check for underfitting and overfitting problems and to optimize parameters (e.g., SMO). The results of the sub-samples of a fixed split ratio are finally averaged.

4.3 Results of SRL-enriched classification

Figure 6 depicts the results obtained with several configurations of classifiers. The figure is horizontally organized according to the different configurations, namely: J48-BR, J48-LP, NB-BR, NB-LP, SMO-BR and SMO-LP. The values for the SRL-enriched dataset are shown as piled bar charts. In each chart, the X axis shows the split percentage used for training/testing in our validation, whereas the Y axis shows precision (or recall) values.

The enrichment of textual requirements with semantic roles led to consistent improvements across the six configurations, for both precision and recall. Some configurations, such as those using the NB classifier and the SMO-LP configuration obtained the best improvements (~ 12 and $\sim 18\%$ for both metrics, respectively). When it comes to the split percentages, we observed that using 20/80 and 30/70 splits was not enough to build good classification models. Without the SRL enrichment, only a few configurations had acceptable precision and recall using a 60/40 split. However, when the SRL enrichment was applied, we observed very good precision and recall values over 40/60 splits (i.e., 40% of the instances used for training and the remaining 60% for testing, averaging the results of ten random but evenly distributed partitioning samples). This result means that the enrichment allowed us to learn classifiers “faster”, in the sense that fewer instances were required (~ 280 instances).

In terms of precision, the best performing configurations were J48-BR and SMO-BR. These configurations obtained an absolute ~ 65 and $\sim 70\%$ precision using a 40/60 split, respectively. These results were boosted to ~ 72 and $\sim 76\%$ precision when the dataset was enriched. We noticed that SMO-LP achieved initially a very low precision (55% at 40/60), which was then much improved with the SRL enrichment ($\sim 68\%$ at 40/60). The NB classifier obtained lower precision values

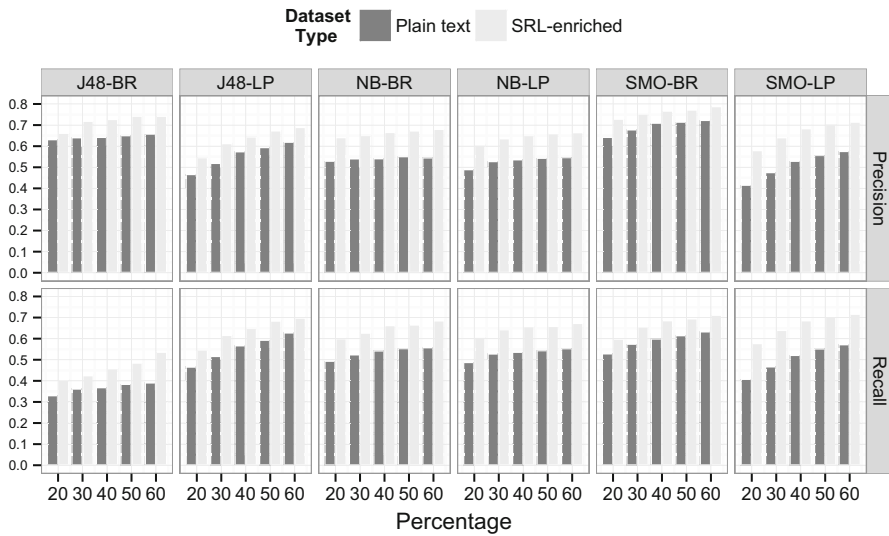


Fig. 6 Experimental results with SRL enrichment

than the other techniques, regardless of the split percentage. The configurations with SMO classifiers obtained the best recall. Both SMO-BR and SMO-LP achieved ~ 60 and $\sim 55\%$ recall at a 40/60 split. However, the enrichment boosted these values to $\sim 70\%$ at 40/60. The J48-BR configuration had a rather poor recall (under $\sim 40\%$ without enrichment and $\sim 50\%$ with enrichment), showing a trade-off with its good precision. In a similar way to the precision analysis, SMO-LP exhibited low recall (55% at 40/60), which was then boosted to $\sim 69\%$ thanks to the SRL enrichment.

4.4 Results of WordNET- and Wikipedia-enriched classification

In addition to the analysis of the two datasets in the previous section, we also explored the performance of enrichment strategies for text classifiers, such as WordNET and Wikipedia concepts (i.e., concept-based vectors instances). To this end, we conducted experiments for assessing the precision and recall measurements of the *WordNET-Concepts* and *Wikipedia-Concepts* datasets. Basically, these datasets have the words of the sentences replaced by their conceptual counterparts in a lexical resource (if such concept exists). We also evaluated the classifiers in the *WordNET-Enriched* and *Wikipedia-Enriched* datasets. These datasets also incorporate WordNET and Wikipedia concepts, but retain the original words in each sentence (i.e., instance vectors which are composed of both words and concepts). Figures 7 and 8 show the results of running the text classifiers on the four dataset variants. Moreover, the figures also show the precision and recall of *Plain-Text* as a reference for comparisons. For the sake of experimental validity, we used the same configurations of single-class classifiers (J48, NB and SMO) and extensions for multi-classes/labels (BR and LP), and we employed a schema with identical fixed-

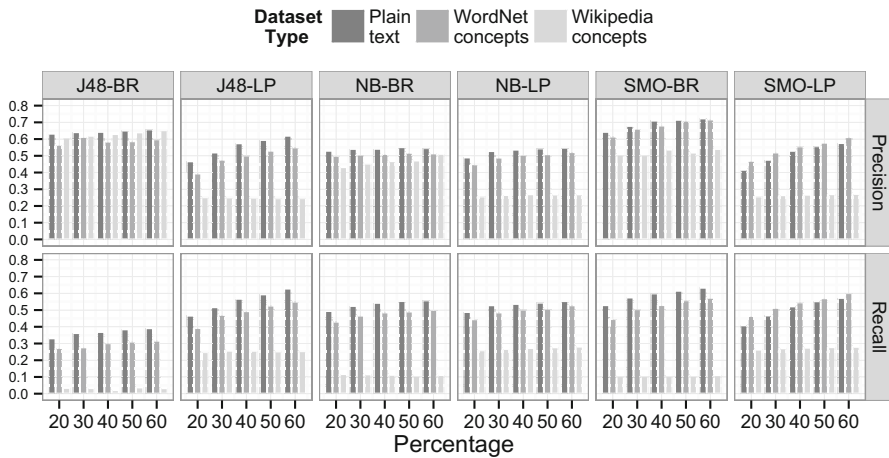


Fig. 7 Experimental results with WordNET and Wikipedia concepts

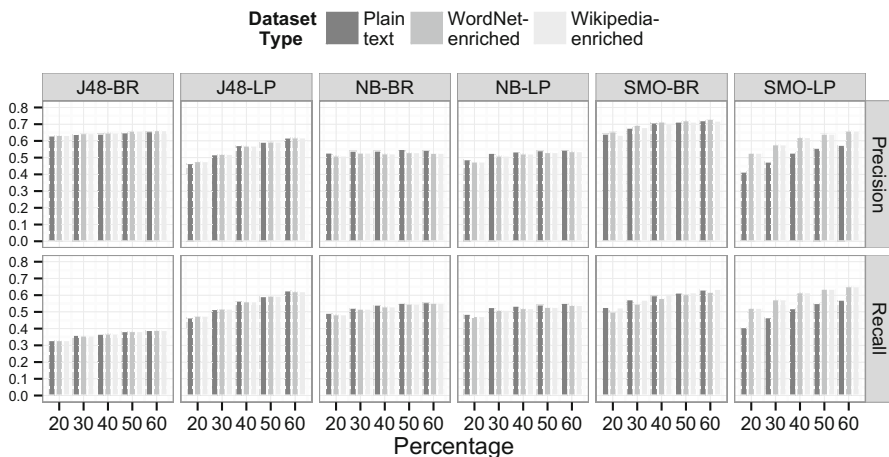


Fig. 8 Experimental results with WordNET and Wikipedia enrichment

splits partitioning to evaluate the datasets. In the piled charts, the X axis represents the six configurations of the classifier and the percentage used for splitting the datasets into training and testing subsets. The Y axis measures the precision and recall obtained by the classifiers to predict domain actions.

The *Wikipedia-Concepts* dataset yielded poor results in almost every configuration. Classifiers using the LP extension, for example, produced worse precision and recall using Wikipedia concepts than using the original text ($\sim 15\text{--}25\%$ drops in both metrics). Alternatively, classifiers using the BR extension achieved an acceptable and sometimes equal precision than with the *Plain-Text* dataset, but at the cost of getting a low recall ($\sim 5\text{--}20\%$). This is an important finding because it

enforces the idea that Wikipedia articles are not good enough on its own to improve the classification of technical documentation, disregarding the classifier chosen and its configuration. The classifiers achieved a better accuracy using *WordNET-Concepts*, but without improving the results of *Plain-Text*.

The transformation of word vectors to WordNET concepts yielded virtually identical results in terms of precision and recall than *Plain-Text*, slightly reducing the precision and recall. We found an exception to this observation in the SMO-LP classifier, in which WordNET led to precision and recall boosts between 3 and 7% for every training/testing split percentage. However, such improvement is apparently an anomaly of this configuration, which presented a low precision and recall with the *Plain-Text* dataset. Nonetheless, the gains obtained with WordNET are still significantly lower to the boost produced by enriching the dataset with SRL in the same configuration ($\sim 18\%$ gains).

The classifiers behaved better with the *WordNET-enriched* and *Wikipedia-enriched* datasets than with the previous concept-based datasets (see Fig. 8). However, this enrichment strategy failed to improve the results of *Plain-Text* by a significant margin. Since the results are very similar to those of the *Plain-Text*, the reader is referred to Fig. 9 to appreciate the precision and recall variations. This chart summarizes the improvements of the enriched datasets (*SRL-Enriched*, *WordNET-Enriched* and *Wikipedia-Enriched*) with respect to traditional text classification (*Plain-Text*). The X axis contains the configurations of single-class classification algorithms, multi classes/labels extensions and the training/testing splits percentages, whereas the Y axis represents the differences in recall and precision between the classifiers using word vectors and its enriched counterparts. A quick look at the precision and recall obtained with the J48-BR and J48-LP configurations shows that WordNET and Wikipedia enrichment neither increased nor reduced the values achieved with the *Plain-Text* dataset. Fortunately, this was not the case of SRL-enriched classification, which produced approximately 8% improvement for J48 in both precision and recall. The results of the NB classifier revealed a pattern in the measures, in which the *SRL-Enriched* dataset consistently

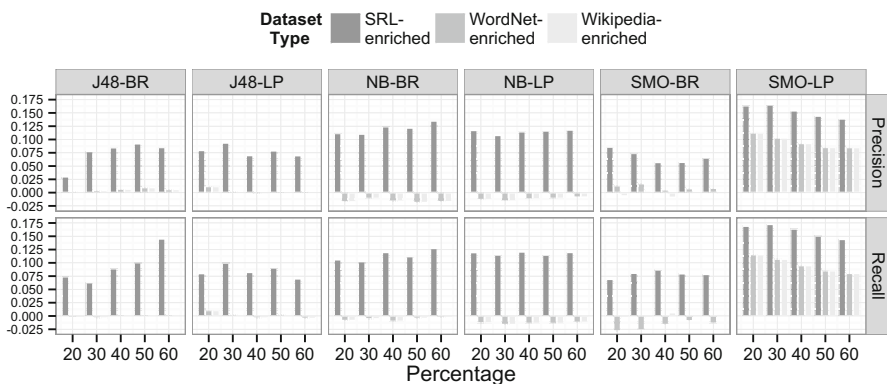


Fig. 9 Accuracy gains and losses due to enrichment

boosted the precision and recall by approximately 12%, while the classifiers were insensitive when using WordNET and Wikipedia enrichments.

Unlike J48 and BR, the SMO classifier is suited to handle a large number of attributes in the dataset, which is a direct consequence of enriching the instance vectors, and thus its performance is very relevant in the evaluation. Despite this characteristic, the SMO-BR configuration exhibited a nearly identical precision and recall with the *WordNET-enriched* dataset to that of *Plain-Text*. Using Wikipedia as enrichment mechanism instead did not produce significant variations in precision and recall for SMO either. Either way, the classification improvements of the *SRL-enriched* dataset in SMO-BR were noticeable lower than that of NB algorithms in terms of both precision and recall (6% vs. 12%). This phenomenon can be attributed to the good precision and recall already obtained by this configuration, which left little room for boosting the metrics. The anomaly observed with the SMO-LP classifier was even more evident in the enriched datasets. Since this configuration underperformed with the *Plain-Text* dataset, using either WordNET, Wikipedia or SRL produced significant gains in precision and recall. For *WordNET-enriched* and *Wikipedia-enriched* datasets, the highest boost of $\sim 10\%$ was obtained with 20/80 splits. The improvements slightly decreased as the training sets of the splits were larger. Training the classifier with the *SRL-enriched* dataset produced a maximum $\sim 18\%$ boost in the precision and recall, outperforming the other enrichment strategies by a factor of two in some of the splits. Even though SMO-LP obtained the largest accuracy boost, this classifier was not the best performing configuration when it comes to absolute measurements. For instance, J48-LP got a similar precision and SMO-BR got a higher recall than SMO-LP.

4.5 Discussion of results

There were two main lessons learned from the evaluation of the classifiers. First and foremost, the SRL enrichment strategy produced a noticeable improvement for the classification of DAs. Second, traditional enrichment strategies failed to enhance the classification results when applied to requirements documentation. Regarding the second statement, the accuracy of the classifiers did not increase by using WordNET and Wikipedia, partially confirming our belief that these kind of enrichment techniques are not suitable for technical documents such as software requirements specifications. The strategy of replacing word vectors for concept vectors derived from Wikipedia articles and WordNET synsets (synonyms sets) produced poor results, achieving a lower precision and recall than the *Plain-Text* dataset. The combination of words and Wikipedia/WordNET concepts did not boost accuracy either. In Wikipedia, the enriched representation did not improve the classification because most articles/categories available in this resource are broadly-scoped and unrelated to software concerns, and failed to increase discriminative power of the dataset. In WordNET, the enriched representation incorporated fine-grained concepts that failed to complement the requirements mainly due to their overlapping with existing words in the instances.

Conversely, the SRL enrichment strategy consistently boosted the accuracy of all the DA classifiers, increasing the precision and recall from as low as $\sim 6\%$ to a

higher $\sim 18\%$, depending on the configuration. The idea of expanding the requirements with semantic role descriptions worked well, uniformly improving the performance of the classifiers for the different partitioning splits. However, we could not appreciate a correlation between the partitioning splits and the gains of the *SRL-enriched* datasets. That is, we conjectured that the improvement produced by the SRL enrichment was more evident when the training set is smaller. Surprisingly, we observed that SRL boosted the recall and precision by a similar margin independently of the size of the training subset. Another important observation of the *SRL-enriched* dataset is that even in the most demanding learning scenarios (e.g., using a 30/70 or 40/60 partitioning split), the best-performing classifiers were able to obtain acceptable absolute precision and recall measures, which is a necessity to develop tool-supported techniques that capitalize on the results of the classification. This characteristic would essentially allow a developer to create more complex and rich applications by relying on the “good” results of SRL-enriched classifiers.

5 Applications to RE tools

The classifiers of DAs have applications in diverse types of requirements analyses. In previous works (Rago et al. 2016a, b), we relied on the DA classifier during the processing of use case specifications with two purposes: (1) identifying latent concerns often related to quality attributes, and (2) for finding duplicate functionality written in more than one document.

The first application dealt with the identification of concerns “buried” in the use case scenarios and “scattered” across many documents. Use cases normally have textual specifications that describe the interactions between the system and external actors. However, since use cases are specified from a functional perspective, concerns that do not fit well this decomposition criterion are kept away from the analysts’ eye and might end up intermingled in multiple use cases. These *crosscutting concerns* (CCCs) are generally relevant for analysis, design and implementation activities and should be dealt with from early stages. Unfortunately, identifying such concerns by hand is a cumbersome and error-prone task, mainly because it requires a semantic interpretation of textual requirements. To ease the analysis of CCCs, we developed an automated tool called *REAssistant* (Rago et al. 2016a) that is able to extract concepts in the text and localize quality-attribute information from the specifications to reveal candidate CCCs, helping analysts to reason about them before making important commitments in the development. The *REAssistant* tool is implemented as a set of Eclipse plugins that provide special views for visualizing CCCs at different levels of granularity.

The second application that takes advantage of our DA classifier aimed at spotting functional behaviors duplicated in the use cases. In spite of existing guidelines for writing use cases, industrial use cases do not often meet the standards of what it is considered a “good” use case model and often exhibit signs of unwanted/unnecessary redundancy. Duplicating functionality is the action of repeating the description of some interactions between the system and actors.

Although duplication is not always a quality defect, and it might be there for the sake of readability of non-technical stakeholders, the issues entailed to lack of modularity and abstraction can have a profound (negative) effect on the developers conducting activities such as effort estimation, project planning, architectural design, change impact analyses and evolution management. Unfortunately, finding duplicate functionality in multiple specifications is a cumbersome, arduous and error-prone activity for the analysts. For this reason, we developed a tool called *ReqAligner* (Rago et al. 2016b) that helps analysts to identify duplicate behaviors in use cases and provides guidelines to mend those defects (duplications) in an automated fashion. This tool is also materialized as a set of Eclipse plugins that can display the duplication in the use cases and assist them to refactor their text.

The sub-sections provide more details about the internal components of both tools, with a focus on the role of the DA classifier for its operation. The reader is referred to Rago et al. (2016a, b) for further information about the tools.

5.1 The *REAssistant* tool

The *REAssistant*¹¹ tool tries to identify latent concerns by understanding the meaning of use case interactions and relating them with well-known crosscutting concerns (e.g., performance, security, persistence). Initially, the tool adds NLP-generated annotations to the text. These annotations consist of sentences, tokens, arguments and predicates. Next, the tool runs the DA classifier to recognize the kind of interaction described in each use case step. Once the text analyses are completed, analysts can query the resulting annotations by means of searching rules. The analysts' role is to define concern-specific queries codified in terms of the NLP annotations to search for crosscutting concerns. The queries can also take advantage of DAs to unveil crosscutting relations, which represent functional requirements affected by the concerns. Finding these relations requires a semantic interpretation of the use case steps and its linkage with the concerns, which would be impractical to do without concepts like DAs. *REAssistant* comes loaded with a predefined ruleset of CCCs, but these rules can be easily customized by analysts. The queries codify knowledge about concerns and how they relate semantically to natural language expressions, and were defined by experienced analysts to cover a wide range of software domains. There are two types of queries: i) *direct queries*, responsible for detecting a CCC; and ii) *indirect queries*, for detecting DAs that are potentially related to that concern. Direct queries are focused in localizing explicit references to a particular CCC, for example, the word "server" or "database". Complementary, indirect queries are focused in finding more subtle associations that come from a semantic interpretation of the use cases. Figure 10 illustrates a PERFORMANCE rule composed of three queries. Query #1 would find parts of the text related to PERFORMANCE through the analysis of token lemmas such as "response" and "second", similarly to keyword-based approaches. Queries #2 and #3 make use of domain actions to reveal indirect impacts, looking for actions such as "calculation" and "process".

¹¹ <https://code.google.com/p/reassistant/>.

DIRECT QUERY #1

```
select S from [#Sentence#] as S, [#Token#] as T
where for T (lemma = 'response' or lemma = 'second' or lemma = 'time' or stem =
'delay' or lemma = 'throughput' or lemma = 'latency' or lemma = 'deadline')
where T.begin >= S.begin where T.end <= S.end
```

INDIRECT QUERY #2

```
select S from [#Sentence#] as S, [#DomainAction#] as DA
where for DA (label = 'Calculation')
where DA.begin >= S.begin where DA.end <= S.end
```

INDIRECT QUERY #3

```
select S from [#Sentence#] as S, [#DomainAction#] as DA, [#Token#] as T
where for DA (label = 'Process')
where for T (lemma = 'result' or lemma = 'value')
where T.begin >= S.begin where T.end <= S.end
where DA.begin >= S.begin where DA.end <= S.end
```

Fig. 10 Queries for searching a *Performance* concern (REAssistant)

We carried out an evaluation of the rules in *REAssistant* with three case-studies. In the experiments, the use of indirect rules (codified only in terms of keywords and NLP properties) achieved a poor recall. This means that the majority of the sentences associated with a relevant CCCs went undetected. However, once the best-performing DA classifier (SMO-BR) was incorporated into the NLP pipeline and the tool searched the concerns with DA-based indirect rules, the recall was boosted by a 40%, whereas precision decreased by just a 5%. This improvement in absolute recall percentages is attributed to the codification of more comprehensive rules in terms of domain actions, which allowed the engine to significantly raise the retrieval of sentences affected by concerns. Such rules would be overly complicated to define by using mere keywords. Using DAs, the rules were simpler to codify and more compact than those written with keywords. It is worth noting that the (accurate) detection of DAs is vital here, and a bad classification leads to poor results in the identification of CCCs. In absolute scores, the number of sentences of the largest case-study initially found by keyword-based rules went from 86 to 206 when using DA-based rules (out of 273).

5.2 The *ReqAligner* tool

The *ReqAligner*¹² tool was developed to find duplicate functionality in textual use cases by semantically comparing the scenarios using algorithms similar to those used in bioinformatics for matching DNA strings. Similarly to *REAssistant*, *ReqAligner* leverages on a domain-specific classifier of *semantic actions* and, based on such knowledge, employs a *sequence alignment* technique for finding suspiciously similar functionalities in the use cases. Basically, the tool works by transforming use case scenarios to sequences of symbols (i.e., DAs) and applying an alignment algorithm to them. The role of the DA classifier with SRL enrichment is pivotal in *ReqAligner*, because it allowed us to summarize and compare the textual contents of the scenarios, as exemplified in Fig. 11. A nice advantage of using the

¹² <http://ucrefactoring.googlecode.com/>.

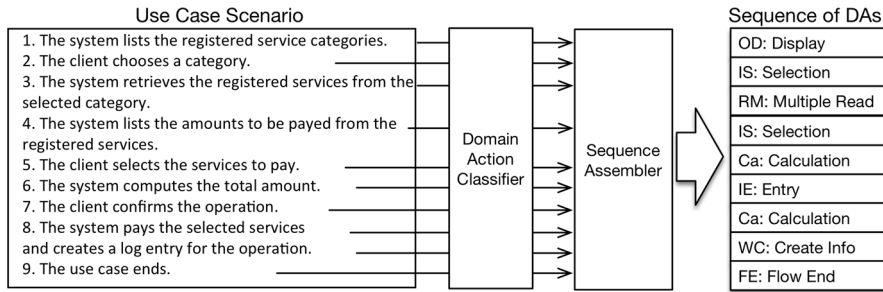


Fig. 11 Transformation of a use case scenario to a sequence of DAs (ReqAligner)

DA classifier is that it can link more than one DA per sentence, because it operates at the level of predicates. This is particularly helpful for different interactions of use case specification combined into a single step using disjunctions/conjunctions. As seen in the figure, each interaction is mapped to a single symbol in the sequence, representing a bottom-level DA. Mid- and top-level domain actions are not considered in the construction of sequences. In the case of interactions tagged with multiple DAs at the same time (ambiguous requirements), the algorithm only maps the most relevant bottom-level DA to the sequence. Once duplicate functionality is found, the tool can suggest UML relationships that might help analysts to remove duplications in the text and ultimately improve the overall requirements model.

Internally, *ReqAligner* assembles the sequences (or chains) of DAs after the semantic analyses of use-case steps is completed. The resulting sequences synthesize a summarized view of a use case scenario, which enables the comparison of scenarios based on their overall semantics (that is, according to their intention and meaning). At this point, the different sequences are processed and matched pairwise by means of a customized *sequence alignment* (SA) technique (Compeau and Pevzner 2015). The sequence alignment basically compares the DAs of two different sequences and aims at finding a sub-chain that maximizes a given similarity function. When two sequences go under analysis, the comparison depends on two predefined parameters: the substitution matrix and the penalization table, which we adapted to the use cases domain. The substitution matrix defines the similarities between the use-case steps (according to their DA). The penalization table defines adjustments (i.e., penalties) the aligner should apply to the similarity score when there are gaps between the matched sequences. If some textual portions of a pair of use cases were successfully aligned, it means that we have one or more candidate duplications in the functional specification. Based on these results, the tool applies a series of heuristics to determine the kind of problem and the most likely UML relationship that can help to refactor (and thus, improve) the use cases. In Rago et al. (2016a), we also reported on the results of an empirical evaluation of *ReqAligner* with five publicly available case-studies that produce promising results.

6 Related work

The use of classifiers in textual documentation has drawn the attention of many engineering researchers in the past few years. On one hand, there has been an increasing interest in improving engineering processes such as software development by applying classification techniques. The goal of these works is to tag software documentation with useful concepts that will ultimately simplify and streamline their analysis. On the other hand, many researchers have also investigated strategies for enriching textual documentation with semantic resources such as WordNET and Wikipedia. Essentially, these strategies aim at complementing the documents with additional information that will help to increase the accuracy of the classifiers. The following sub-sections discuss some relevant works that have addressed these lines of research.

6.1 Identification of domain-specific concepts in software documentation

Many authors have stressed the need of recognizing domain-specific concepts in textual documents. However, current approaches detect the concepts using either ad-hoc techniques or classifiers with VSM transformations. Sinha et al. (2010) introduced the notion of semantic actions in use cases to support completeness, structural and flow checks. Sinha's research is the basis for our hierarchy of DAs. Basically, they implemented a pipeline of linguistic analysis engines for understanding textual use cases and defined an extensible architecture to process requirements in multiple languages. The novel contribution of their work is a domain annotator that maps commonly occurring verbs to a set of pre-defined semantic classes, such as: INPUT, OUTPUT, READ, WRITE, GIVE, GET, UPDATE, QUERY or DELEGATE, among others. However, Sinha's implementation relies on a dictionary of verbs and static reference values (for verb associations), disregarding contextual information and requiring a tailored dictionary per system. Furthermore, another limitation of Sinha's taxonomy is the fact that it contains classes of different abstraction levels, such as UPDATE and DELETE vs. WRITE. Kamalrudin et al. (2011) developed an approach for assuring consistency, completeness and correctness of textual use cases, which links the use cases to abstractions called essential interactions. An essential interaction is an important key phrase (in the context of use cases) labeled with a meaningful abstract term. For instance, phrases with predicates such as "indicate", "choose" and "select" are linked with an abstract interaction named CHOOSE. The authors compiled a library of essential interactions that support a variety of application domains. Examples of abstract interactions in the library include CHOOSE, OFFER, VIEW, REQUEST, IDENTIFY, CALCULATE or CONFIRM, among others. Requirements are tagged with these abstract interactions by means of text-matching techniques. Although Kamalrudin's abstractions seem similar to our DAs, we believe that DAs work at a higher abstraction level mainly because many of our classes at the bottom level often include multiple essential interactions (e.g., "enter", "complete" and "identify" can be considered an "Input" DA) and essential interactions are often linked to a very limited number of words (1-to-1

mapping). Additionally, essential interactions only capture fine-grained behaviors whereas our hierarchy of DAs distinguishes the relationships between low, mid and high-level actions that are mapped to a single word. Jurkiewicz and Nawrocki (2015) analyzed use case scenarios to detect missing events. To do so, they recognize associations between use-case scenarios and classes of activities by combining NLP techniques with a special classifier. Jurkiewicz's activities are very similar to Sinha's actions and to our DAs. A limitation common to the three approaches above is their lack of support for semantic enrichment, which might lead to problems of ambiguity or synonyms.

Recent works in the Software Engineering field have employed advanced NLP techniques. Sengupta et al. (2015) developed an approach for deriving a semi-formal model from textual requirements in order to make quality assessments and integrity checks. Essentially, a domain-specific ontology is used for semantically categorize commonly occurring verbs into high-level actions (i.e., intentions), relational verbs (part-whole and part-of) and low-level actions (i.e., concrete behaviors). Each category is further refined into classes that indicate the semantic meaning of an individual verb, such as POLICY, IS-A, EXECUTE, CREATE, INPUT/OUTPUT, PERMIT, CONDITION or VIEW, among others. These classes are very similar to our DAs, but lack of a proper hierarchical organization (for example, display actions labeled as VIEW should be considered a special case of I/O). Moreover, the approach has a limited support for handling ambiguous requirements, even though the authors acknowledged that over 25% of their requirements dataset belong to two or more classes. In our work, we have addressed this situation by modeling the problem as a multi-label classification task. In addition, a dependency parser is used for extracting subject-verb-object structures and semantic role labeling is employed to determine agents, themes and objects. Unfortunately, they applied an ad-hoc SRL technique which only extracts a subset of roles by taking advantage of Stanford dependencies. In many cases, this is a simplification of the SRL problem as defined in PropBank or FrameNet. Another limitation of such implementation is that this lightweight SRL does not provide information about the semantic roles the predicates and arguments are playing in a sentence. In our approach, we take advantage of the information available in PropBank to enrich text classification with semantic descriptions of the roles. Another promising research for comparing existing requirements by means of transformations to more formal models is given by Roth et al. (2014) and Roth and Klein (2015). Initially, they defined an ontology that captures general requirements concepts from different domains. The first level of the ontology contains two classes for distinguishing THINGS from OPERATIONS. The former class is further refined in ACTOR, OBJECT and PROPERTY, whereas the latter is refined in ACTION, EMERGENCE, STATUS and OWNERSHIP. There are also relations between classes which describe and constrain interactions among them, such as HAS_ACTOR, ACTS_ON or OWNS, among others. Roth also proposed using statistical techniques from semantic role labeling to classify textual requirements and map them to ontology concepts and relations. The underlying idea resembles the steps of a traditional semantic role labeling task, which analyzes the text progressively through incremental steps, namely: identifying the concepts, assigning their type, determining related concepts and labeling relations between pairs of concepts.

Similarly to our work, the text is processed by a series of NLP modules, such as POS-tagging, lemmatization and dependency parsing. An important limitation of Roth's work is the level of detail of the ontology, which only describes generic concepts of use cases and does not have support for telling domain-specific interactions apart. The representation introduced in our work make it possible to recognize fine-grained interactions at the lowest level of the DA hierarchy.

6.2 Document enrichment for improving classification accuracy

Several works have studied semantic enrichment strategies for text classification. Some early works showed improvements in the overall accuracy of text classification algorithms. However, the importance of having domain-specific resources for enriching technical documents has been acknowledged. Bloehdorn and Hotho (2006) explored the use of ontologies in text classification tasks in domains such as news, medicine and agriculture, obtaining good results. Nonetheless, the authors noted that the improvement is dependent on the enrichment source chosen, which has to be aligned with the domain of the datasets. Egozi et al. (2011) presented an approach that enriches textual representations with concept-based features extracted from Wikipedia. This approach was used in information retrieval tasks for transforming/enriching both documents and queries with semantic concepts and performing searches in the enriched space. The premise is that by using concepts the retrieval should be less dependent on the specific terms encoded in the documents. Nonetheless, the authors had to resort to aggressive feature selection techniques to avoid "noisy" concepts and improve the initial accuracy of retrieval. Llorens et al. (2013) explored the application of semantic knowledge to improve the identification and classification of temporal expressions and events. Particularly, they used lexical semantics and semantic roles for enriching textual information and coping with time-related ambiguities. Semantic roles provide valuable information for telling temporal arguments apart from non-time related arguments. However, the improvements obtained experimentally could not be replicated for their classification, because according to the authors the semantic enrichment was not sufficiently discriminative in their domain. Wang and others also explored the use of Wikipedia for text classification. In Wang and Domeniconi (2008), a semantic kernel is used to embed background knowledge derived from Wikipedia and enrich textual representations. Some initial evaluations lead to slightly better results than bag-of-words representations in newspapers, forums and movies datasets. In Wang et al. (2009), Wikipedia was also applied to text classifiers for news datasets. The authors discovered that over-enriching the data can lead to worse results than the baseline. Bai et al. (2010) extracted information from WordNET to improve text classification, obtaining slightly better results than the bag-of-word baseline. Navigli et al. (2011) also explored WordNET concepts for classifying and disambiguating Wikipedia articles. Essentially, they created semantic models for the articles derived from WordNET synsets, their synonymys and glosses, obtaining better results than the baseline. Li et al. (2012) presented a technique that uses WordNET enrichment for creating category profiles that allows to classify unlabeled documents, avoiding the need of labeled datasets and hence fully automating text classification tasks. Rooney et al.

(2014) tackled the problem of textual entailment by enriching sentences with diverse features extracted from semantic resources, such as dependency parsing, semantic role labeling, and WordNET relationships (e.g., synonyms, antonyms, meronyms, among others). As a result, they found several redundant features that hinder classification. Tommasel and Godoy (2014) proposed a semantic enrichment of tags for the classification of Web resources, using both Wikipedia and WordNET. Comparisons against simple tag-based representations showed that certain configurations with WordNET slightly improved classification tasks, but this trend could not be verified with Wikipedia.

7 Conclusions and future work

In this article, we have presented an enrichment technique based on semantic roles for improving text classification, with applications to the RE domain. Particularly, the inclusion of SRL helped to categorize domain-specific abstractions called DAs in textual use cases. We approached the categorization of DAs as multi-label, hierarchical classification problem. An empirical evaluation of several configurations of classifiers on a requirements dataset showed promising results due to the SRL enrichment. Specifically, we observed classification improvements up to $\sim 18\%$ in both precision and recall for some configurations. We also noticed that classifiers required fewer instances to learn good classification models. In addition, the evaluation showed that WordNET and Wikipedia enrichments are not enough to improve the classification of technical documentation, such as requirements specifications. From a theoretical viewpoint, our research shows that role-based features, such as predicate and argument descriptions, can be beneficial for text classification purposes. An experimental evaluation in a technical domain, such as use case specifications, revealed accuracy gains and little to none precision loss. This result is interesting because enrichment strategies based on general-purpose resources have difficulties to achieve a good accuracy in this type of documentation. Thus, domain-specific or semantic enrichment techniques need to be investigated in order to boost accuracy. A practical implication of our work is a demonstration that an SRL enrichment allows classifiers to perform better in cases of unknown textual requirements. The strength of SRL comes from its reliance on common knowledge encoded in semantic roles descriptions. Since applications like *REAssistant* or *ReqAligner*, which rely on the classifier of DAs, demand a high accuracy in order to produce good results, alternative enrichment techniques like the one presented in this article are very important in tool-supported engineering activities. Along this line, another practical implication is the conjecture that both the DA model and the SRL enrichment are applicable to other kinds of software documents.

As future work, we will further evaluate the SRL technique with other requirements datasets, and compare it against other approaches based on subsets of Wikipedia or WordNET. We plan to upgrade the semantic role tagger used in this work with more recent and better SRL tools. We will also investigate whether feature selection algorithms can enhance the results of the multi-label classifier. Finally, we will explore other applications of our DA classifier to aid complex

requirements analyses. Some promising ideas are (1) the adaptation of the hierarchy of domain actions to other types of requirements specifications such as user stories, and (2) the detection of design decisions in software architecture documents and traceability links between the architecture and requirements.

Acknowledgements This work was partially supported by ANPCyT (Argentina) through PICT Project 2015 No. 2565. The authors are grateful to the doctoral students that helped to manually tag the sentences of the case-studies with DAs. The authors would like to make a special mention to Paula Frade, Miguel Ruival, German Attanasio and Rodrigo Gonzalez for testing the DA classifier and helping us to make adjustments to the implementation. The authors also thank the anonymous reviewers for their feedback that helped to improve the quality of the manuscript.

Appendix: Description of domain actions

- **Process:** Represents interactions that involve CPU-demanding activities (of a system).
 - **Verification:** Covers interactions associated from checks of user input, validation of stored information and consistency of data.
 - **Calculation:** Covers interactions associated to the analysis of information and the synthesis of new results.
 - **Communication:** Covers all types of interaction with subsystems or foreign software/hardware.
 - **Internal:** Groups interactions linked to data sharing with subsystems.
 - **External:** Groups interactions linked to data sharing with other systems.

- **Data:** Represents interactions that involve data-related activities, such as persistence and caches operations.
 - **Read:** Covers interactions associated to retrieval of data.
 - **Single:** Groups retrieval interactions of single values, often linked to parameters and object representations.
 - **Multiple:** Groups retrieval interactions of many tuples of information, often materialized as a complex query.

 - **Write:** Covers interactions associated to the storage of data, by either adding, modifying or removing.
 - **Create:** Groups interactions aimed at incorporating new information to the system.
 - **Update:** Groups interactions aimed at altering pre-existing information in the system.
 - **Delete:** Groups interactions aimed at removing information from the system.

- Use Case: Represents interactions commonly used in use case scenarios to manage the execution flow.
 - Begin: Groups interactions frequently used to denote the start of a use case flow.
 - End: Groups interactions frequently used to denote the end of a use case flow.
 - Control: Groups interactions to denote the jump from one use case step to another.

- Input/Output: Represents interactions that involve the communication between the system described and human actors (or other systems).
 - Input: Covers interactions associated to the feeding of information to the system.
 - Entry: Groups interactions related to feeding in data via physical/virtual interface.
 - Selection: Groups interactions related to choosing data from a list of options.
 - Output: Covers interactions associated to the delivery of information to end users.
 - Display: Groups interactions related to the presentation of data on a physical/virtual display.
 - Notification: Groups interactions about status changes or warning messages.

References

- Badawi, D., & Altincay, H. (2014). A novel framework for termset selection and weighting in binary text classification. *Engineering Applications of Artificial Intelligence*, 35, 38–53. <https://doi.org/10.1016/j.engappai.2014.06.012>.
- Bai, R., Wang, X., & Liao, J. (2010). Extract semantic information from wordnet to improve text classification performance. In T. H. Kim & H. Adeli (Eds.), *Advances in computer science and information technology, Lecture notes in computer science* (Vol. 6059, pp. 409–420). Berlin: Springer. https://doi.org/10.1007/978-3-642-13577-4_36.
- Björkelund, A., Bohnet, B., Hafdell, L., & Nagues, P. (2010). A high-performance syntactic and semantic dependency parser. *23rd International conference on computational linguistics: Demonstrations (COLING '10)* (pp. 33–36). Beijing: Association for Computational Linguistics.
- Bloehdorn, S., & Hotho, A. (2006). Boosting for text classification with semantic features. *Advances in Web Mining and Web Usage Analysis*, 3932, 149–166. https://doi.org/10.1007/11899402_10.
- Casamayor, A., Godoy, D., & Campo, M. (2012). Functional grouping of natural language requirements for assistance in architectural software design. *Knowledge-Based Systems*, 30, 78–86. <https://doi.org/10.1016/j.knosys.2011.12.009>.
- Compeau, P., & Pevzner, P. (2015). *Bioinformatics algorithms: An active learning approach* (2nd ed.). San Diego: Active Learning Publishers.

- Diamantopoulos, T., Roth, M., Symeonidis, A., & Klein, E. (2017). Software requirements as an application domain for natural language processing. *Language Resources and Evaluation*. <https://doi.org/10.1007/s10579-017-9381-z>.
- Egozi, O., Markovitch, S., & Gabrilovich, E. (2011). Concept-based information retrieval using explicit semantic analysis. *ACM Transactions on Information Systems (TOIS)*, 29(2), 8:1–8:34. <https://doi.org/10.1145/1961209.1961211>.
- Falessi, D., Cantone, G., & Canfora, G. (2013). Empirical principles and an industrial case study in retrieving equivalent requirements via natural language processing techniques. *IEEE Transactions on Software Engineering*, 39(1), 18–44. <https://doi.org/10.1109/TSE.2011.122>.
- Femmer, H., Fernandez, D. M., Wagner, S., & Eder, S. (2017). Rapid quality assurance with requirements smells. *Journal of Systems and Software*, 123, 190–213. <https://doi.org/10.1016/j.jss.2016.02.047>.
- Huang, L. (2011). *Concept-based text clustering*. Doctoral thesis, The University of Waikato, Hamilton.
- Huang, L., Milne, D., Frank, E., & Witten, I. H. (2012). Learning a concept-based document similarity measure. *Journal of the American Society for Information Science and Technology*, 63(8), 1593–1608. <https://doi.org/10.1002/asi.22689>.
- Hull, E., Jackson, K., & Dick, J. (2014). *Requirements engineering* (3rd ed.). Berlin: Springer.
- Jurkiewicz, J., & Nawrocki, J. (2015). Automated events identification in use cases. *Information and Software Technology*, 58, 110–122. <https://doi.org/10.1016/j.infsof.2014.09.011>.
- Kamalrudin M., Hosking J. G., & Grundy, J. (2011). Improving requirements quality using essential use case interaction patterns. In *ICSE'11, Hawaii* (pp. 531–540). <https://doi.org/10.1145/1985793.1985866>.
- Kang, S., Cho, S., & Kang, P. (2015). Multi-class classification via heterogeneous ensemble of one-class classifiers. *Engineering Applications of Artificial Intelligence*, 43, 35–43. <https://doi.org/10.1016/j.engappai.2015.04.003>.
- Kehagias, A., Petridis, V., Kamburlasos, V. G., & Fragkou, P. (2003). A comparison of word- and sense-based text categorization using several classification algorithms. *Journal of Intelligent Information Systems*, 21(3), 227–247. <https://doi.org/10.1023/A:1025554732352>.
- Kelleher, J. D., Namee, B. M., & D'Arcy, A. (2015). *Fundamentals of machine learning for predictive data analytics: Algorithms, worked examples, and case studies* (1st ed.). Cambridge: MIT Press.
- Li, J. Q., Zhao, Y., & Liu, B. (2012). Exploiting semantic resources for large scale text categorization. *Journal of Intelligent Information Systems*, 39(3), 763–788. <https://doi.org/10.1007/s10844-012-0211-x>.
- Llorens, H., Saquete, E., & Navarro-Colorado, B. (2013). Applying semantic knowledge to the automatic processing of temporal expressions and events in natural language. *Information Processing & Management*, 49(1), 179–197. <https://doi.org/10.1016/j.ipm.2012.05.005>.
- Mahmoud A., & Carver, D. (2015). Exploiting online human knowledge in requirements engineering. In *23rd International requirements engineering conference (RE'15)*, IEEE (pp. 262–267). <https://doi.org/10.1109/RE.2015.7320434>
- Mansuy T., & Hilderman, R. J. (2006). A characterization of wordnet features in Boolean models for text classification. In *5th Australasian conference on data mining and analytics (AusDM'06)* (Vol. 61, pp. 103–109).
- Ménard, P. A., & Ratté, S. (2016). Concept extraction from business documents for software engineering projects. *Automated Software Engineering*, 23(4), 649–686. <https://doi.org/10.1007/s10515-015-0184-4>.
- Mund, J., Fernandez, D. M., Femmer, H., & Eckhardt, J. (2015) Does quality of requirements specifications matter? Combined results of two empirical studies. In *ACM/IEEE international symposium on empirical software engineering and measurement (ESEM'15)* (pp. 1–10). <https://doi.org/10.1109/ESEM.2015.7321195>.
- Navigli, R., Faralli, S., Soroa, A., de Lacalle, O., & Agirre, E. (2011). Two birds with one stone: Learning semantic models for text categorization and word sense disambiguation. In *20th ACM international conference on information and knowledge management (CIKM'11)* (pp. 2317–2320). <https://doi.org/10.1145/2063576.2063955>.
- Nazir, F., Butt, W. H., Anwar, M. W., & Khan Khattak, M. A. (2017). *The applications of natural language processing (NLP) for software requirement engineering—A systematic literature review* (pp. 485–493). Singapore: Springer. https://doi.org/10.1007/978-981-10-4154-9_56.
- Nguyen, T. H., Grundy, J., & Almorisy, M. (2015). Rule-based extraction of goal-use case models from text. In *10th Joint meeting on foundations of software engineering (FSE'2015)* (pp. 591–601). <https://doi.org/10.1145/2786805.2786876>.

- Palmer, M., Gildea, D., & Kingsbury, P. (2005). The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1), 71–106. <https://doi.org/10.1162/0891201053630264>.
- Palmer, M., Gildea, D., & Xue, N. (2010). *Semantic role labeling. Synthesis lectures on human language technologies*. San Rafael: Morgan & Claypool.
- Rago, A., Marcos, C., & Diaz-Pace, A. (2013). Uncovering quality-attribute concerns in use case specifications via early aspect mining. *Requirements Engineering*, 18(1), 67–84. <https://doi.org/10.1007/s00766-011-0142-z>.
- Rago, A., Marcos, C., & Diaz-Pace, A. (2016a). Assisting requirements analysts to find latent concerns with REAssistant. *Automated Software Engineering*, 23(2), 219–252. <https://doi.org/10.1007/s10515-014-0156-0>.
- Rago, A., Marcos, C., & Diaz-Pace, A. (2016b). Identifying duplicate functionality in textual use cases by aligning semantic actions. *Software and Systems Modeling*, 15(2), 579–603. <https://doi.org/10.1007/s10270-014-0431-3>.
- Rago, A., Marcos, C., & Diaz-Pace, A. (2016c). Opportunities for analyzing hardware specifications with NLP techniques. In *3rd Workshop on design automation for understanding hardware designs (DUHDe'16), design, automation and test in Europe conference and exhibition (DATE'16)*, Dresden, Germany.
- Rooney, N., Wang, H., & Taylor, P. S. (2014). An investigation into the application of ensemble learning for entailment classification. *Information Processing & Management*, 50(1), 87–103. <https://doi.org/10.1016/j.ipm.2013.08.002>.
- Rosadini, B., Ferrari, A., Gori, G., Fantechi, A., Gnesi, S., Trotta, I., & Bacherini, S. (2017). Using NLP to detect requirements defects: An industrial experience in the railway domain. In *chap 23rd International working conference REFSQ 2017*, Essen, Germany, February 27–March 2, 2017, Proceedings (pp. 344–360). Springer International Publishing. https://doi.org/10.1007/978-3-319-54045-0_24.
- Roth, M., & Klein, E. (2015). Parsing software requirements with an ontology-based semantic role labeler. In *1st Workshop on language and ontologies at the 11th international conference on computational semantics (IWCS'15)* (pp. 15–21). London, United Kingdom.
- Roth, M., Diamantopoulos, T., Klein, E., & Symeonidis, A. (2014). Software requirements: A new domain for semantic parsers. In *Workshop on semantic parsing at the conference of the association for computational linguistics (ACL'14)* (pp. 50–54). Baltimore, MD.
- Selvaretnam, B., & Belkhatir, M. (2016). A linguistically driven framework for query expansion via grammatical constituent highlighting and role-based concept weighting. *Information Processing & Management*, 52(2), 174–192. <https://doi.org/10.1016/j.ipm.2015.04.002>.
- Sengupta, S., Ramnani, R. R., Das, S., & Chandran, A. (2015). Verb-based semantic modelling and analysis of textual requirements. In *8th India software engineering conference (ISEC'15)* (pp. 30–39). <https://doi.org/10.1145/2723742.2723745>.
- Sinha, A., Paradkar, A., Takeuchi, H., & Nakamura, T. (2010). Extending automated analysis of natural language use cases to other languages. In *18th IEEE international requirements engineering conference (RE'10)* (pp. 364–369). <https://doi.org/10.1109/RE.2010.52>
- Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4), 427–437. <https://doi.org/10.1016/j.ipm.2009.03.002>.
- Szu-ting, Y. (2015). *Robust semantic role labeling*. United States: LAP Lambert Academic Publishing.
- Tommassel, A., & Godoy, D. (2014). Semantic grounding of social annotations for enhancing resource classification in folksonomies. *Journal of Intelligent Information Systems*. <https://doi.org/10.1007/s10844-014-0339-y>.
- Tsoumakas, G., Katakis, I., & Vlahavas, I. (2010). Mining multi-label data. In O. Maimon & L. Rokach (Eds.), *Data mining and knowledge discovery handbook* (pp. 667–685). Boston, MA: Springer. https://doi.org/10.1007/978-0-387-09823-4_34.
- Uysal, A. K., & Gunal, S. (2014). The impact of preprocessing on text classification. *Information Processing & Management*, 50(1), 104–112. <https://doi.org/10.1016/j.ipm.2013.08.006>.
- Wang, P., & Domeniconi, C. (2008). Building semantic kernels for text classification using wikipedia. In *14th ACM SIGKDD international conference on knowledge discovery and data mining (KDD'08)* (pp. 713–721). <https://doi.org/10.1145/1401890.1401976>.
- Wang, P., Hu, J., Zeng, H. J., & Chen, Z. (2009). Using wikipedia knowledge to improve text classification. *Knowledge and Information Systems*, 19(3), 265–281. <https://doi.org/10.1007/s10115-008-0152-4>.

- Wieggers, K., & Beatty, J. (2013). *Software requirements* (3rd ed.). Developer best practices. Redmond, WA: Microsoft Press.
- Zhang, H. (2004). The optimality of naive bayes. In V. Barr & Z. Markov (Eds.), *17th International Florida Artificial Intelligence Research Society conference (FLAIRS 2004)* (pp. 562–567). Miami Beach, FL: AAAI Press.
- Zhang, M. L., & Zhou, Z. H. (2014). A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8), 1819–1837. <https://doi.org/10.1109/TKDE.2013.39>.