Robust Requirements Analysis in Complex Systems through Machine Learning

Francesco Garzoli[1], Danilo Croce[1], Manuela Nardini[2], Francesco Ciambra[2], and Roberto Basili[1]

[1] University of Roma, Tor Vergata, Italy
[2] Finmeccanica SELEX Sistemi Integrati, Italy

Abstract. Requirement Analysis (RA) is a relevant application for Semantic Technologies focused on the extraction and exploitation of knowledge derived from technical documents. Language processing technologies are useful for the automatic extraction of concepts as well as norms (e.g. constraints on the use of devices) that play a key role in knowledge acquisition and design processes. A distributional method to train a kernel-based learning algorithm is here proposed, as a cost-effective approach for the validation stage in RA of Complex Systems, i.e. Naval Combat Systems. The targeted application of Requirement Identification and Information Extraction techniques is here discussed in the realm of robust search processes that allows to suitably locate software functionalities within large collections of requirements written in natural language.

1 Introduction

The objectives of Requirements Engineering (RE) include at least the identification of the goals to be achieved by a target system, the operationalization of such goals into services and constraints, and the assignment of responsibilities for the resulting requirements to agents such as humans, devices and software. Different processes are involved in RE, such as domain analysis, elicitation, specification, assessment, negotiation, documentation and evolution. Sources of information are mostly expressed in natural language and require manual analysis: getting high quality requirements is difficult, critical and costly. During a novel system design, all of these phases must be performed, and generally they are carried out without any reuse of old analysis performed over previous systems.

In this scenario, search systems are usually required to help analysts to locate and access the information stored in documents whereas key-word based search may not be sufficient. As an example, when searching for "*attack scenario*" documents containing an expression such as (the verb) "*assail*" may not be found as for the mismatch between the query and the text. A more semantic aware process is needed to increase the benefits of automatic search in RE. Furthermore the validation of design choices could be automatized, e.g. checking the consistency of requirement pre-conditions. However, translating user requirements and problem domain described in natural language into the consistent modeling of the target application is still challenging. According to [1], "*We are not really* *having a problem coding a solution - we are having a problem understanding what solution to code . . . If you focus on requirements and verification and validation, the coding will take care of itself*". Vagueness and ambiguity are the main phenomena that make the natural language used to describe user requirement a challenging task. Consider the complexity of a sentence when it contains clauses and phrases that describe and relate several objects, conditions, events and/or actions.

Natural Language Processing (NLP) approaches gained much interest in the community of Software Engineering, as recent works in this direction suggest. In [2] a similarity measure based on linguistic information is used for clustering correlated software artifacts. In particular, authors explore the effects of mining lexical information about different artifact element, such as Function Names, Parameter Names or Software Comments. In [3], an automatic approach to classify affordances of web services according to the texts describing them is presented.

Regarding Requirement Analysis, Abbot [4] proposes a technique attempting to guide the systematic procedure that compiles design models from textual requirements. While it was able to produce static analysis and design modules, it was nonetheless requiring high levels of user involvement for decision making. Saeki et. Al. [5] illustrates a process of incrementally constructing software modules from object-oriented specifications as obtained by interpreting text requirements. Nouns were considered as classes and their corresponding verbs as methods. These were automatically extracted from the raw textual descriptions but lexical ambiguity problems and hand-coding were striking limitations in the construction of fully reusable formal specifications. In the REVERE [6] system, a summary of requirements from a natural language text is derived. The system makes use of a lexicon to recognize suitable word senses in the texts. However, no attempt to model the system at the functional level is carried out. In [7] natural language analysis is suggested as a possible approach for automatically compile formalized control mechanisms in the requirement specifications. An expressive semantics-based point cuts within a requirement are detected and mapped into the RDL semi-formal description language. The authors suggest that syntactic and semantic analysis of natural language expressions can be made precise enough to support the definition of a flexible composition mechanism for requirements analysis. All those systems, while exploring the applicability of NLP, propose traditional tools for the specific RE context. Most of the traditional limitations of NLP are thus inherited by the above works, namely costly design and development processes, complex maintenance of the large Knowledge Bases necessary for full NL analysis as well as poor portability across domain, systems and scenarios.

In this work we propose statistical learning methods embedded in a large scale natural language processing system in support of RE. The adoption of advanced technique of NLP combined with Machine Learning capabilities, i.e. Statistical Information Extraction, is a crucial advance to improve applicability of this technology on a large scale. Moreover, the effectiveness of acquired information is evaluated in a Information Retrieval scenario, where a robust search engine has been defined to search existing software functionalities in a specific domain through user requirements expressed in natural language.

In the rest of the paper, Section 2 discusses the application of Human Language Technologies in RE. Section 3 proposes the architecture of an automatic system for Requirement Analysis. Section 4 presents the evaluation of the adopted techniques for the Naval Combat Systems requirement analysis.

2 Language Technologies for Requirement Analysis

The robustness recently achieved by NLP technologies makes their applicability in the support the analysis and design of system development very promising. As an example, the reuse of existing technological components during the design stages of new complex systems can be drastically increased whenever a semantic search system from the targeted component repository is available. Such an engine would be able to rely on conceptual notions in the user queries (e.g. functions and norms), as they are originally extracted from technical specification documents, and retrieve components suitable for the design needs and validate them according to their compliancy or *composability*. The role of Human Language Technologies (HLT) in this proactive support to the analyst is clear as it favors the incremental design through reuse. HLT are crucial to support robust and accurate analysis of unstructured texts, and enrich them by semantic meta-data or other kinds of information implicit in the texts. HLT allows extracting the interesting semantic phenomena and mapping them into structured representation of a target domain. When a semantic meta-model is available, for example in form of an existing ontology, HLT allows to locate concepts in the text (irrespectively from the variable forms in which they appear in the free text), mark them according to Knowledge Representation Languages (such as RDF or OWL) thus unifying different shallow representations of the same concepts. In this way semantic annotations of concepts in the text (i.e. automatic semantic indexes) are obtained for the original document, making it more suitable for clustering, retrieval and browsing activities. In synthesis, HLT enables to perform and simplify several advanced functionalities (e.g. semantic and not keyword based search) that are possible over the text. The semantic annotation task, just outlined above, has been largely studied by the NLP community and it is known as Information Extraction (IE), i.e. "*The identification and extraction of instances of a particular class of events or relationships in a natural language text and their transformation into a structured representation (e.g. a database).*" [8]. IE requires typically three stages. In the first, the target information is abstracted and structured set of inter-related categories are designed. These structures are called *templates* and the categories (roles) that need to be filled with information are called *slots*. For example, if we want to extract conceptual information about vessels from specifications, we may be interested in the name but also in the *type of ship* or its *maximum speed*, as well as its *combat system equipment*. Therefore, a SHIP template can be defined as a conjunctive combination of slots such as *name*, *ship type*, *maximum speed* or *combat* *system equipment*. Once the template is given, the text fragments containing relevant information to fill the template slots (i.e. specific values associated to the attributes of a certain SHIP instance) need to be identified in a text. The recognition of textual information of interest results from pattern matching against extraction rules. Finally, in a third phase, whenever the information of interest is identified in the text, its mapping in the suitable (e.g. SHIP) template slot is carried out. The above chain is not trivial and contemporary IE systems[1] are usually integrated with large scale knowledge bases, determining all the lexical, syntactic and semantic constraints needed for a correct interpretation of usually domain-specific texts. Unfortunately, the manual development of these resources is a time-consuming task that is often highly error-prone due to the subjectivity and intrinsic vagueness that affects the semantic modeling process. Knowledge acquisition task is often approached through the use of Machine Learning algorithms to automatically learn the domain-specific information from annotated data [9]. Statistical learning methods [10] assume that lexical or grammatical aspects of training data are the basic features for modeling the different inferences. They are then generalized into predictive patterns composing the final induced model. A statistical language processor is assumed to be able to locate specific instances of a template type (e.g. SHIP) and their slot information in an incoming text. The resulting instantiated template can be employed to populate an existing knowledge base whose semantic schema correspond (or can be mapped) to the template structure. Moreover, reasoning over the extracted information, e.g. identifying relations or dependencies with respect to previous requirements, can be better performed. For example, retrieval of developed components that respond properly to new requirements could be realized as a form of reasoning.

3 Machine Learning for Requirement Analysis

In Requirement Analysis some NLP applications like Information Extraction tasks could be very useful to support people to perform this task practically and in a cost-effective way. Statistical NLP approaches provide domain specific models of target interpretation tasks by acquiring and generalizing linguistic observations. Several Statistical Machine Learning paradigms have been defined to provide robust models that easily adapt across different (and possibly specific) domains. These techniques are the basis to our proposed approach and we will discuss them hereafter. This problem is normally treated as a Statistical Classification problem, where the target is to identifying the sub-population to which new data belong, where the identity of the sub-population is unknown (the test data), on the basis of a training set of data containing observations whose sub-population is known (the training data). In this scenario we may be interested for example to induce a template slot for a candidate text. Support Vector Machine (SVM), as discussed in [11] and [12], represents one of the most known learning paradigm for classification, based on Statistical Learning Theory. Given training instances, each one associated with a class and a set of "features", i.e.

[1] OpenCalais: `http://viewer.opencalais.com/`

the dimensions of the employed geometrical representation of each example, the goal of SVM is to produce a model (based on the training data) which predicts the target values of the test data given only the test data features. In a geometric perspective, SVM classifiers learn a decision boundary between two data classes that maximizes the minimum distance or margin from the training points in each class to the boundary. The notion of distance used in such feature space can be adapted to a specific classification problem to better separate examples. This is explicitly the role of kernel functions [11] aiming to separate the learning task from the representation through a proper although implicit mapping to a newer space more expressive for the target problem.

Formally, the $SVM^{multiclass}$ schema described in [13] is applied[2] to implicitly compare all class and select the most likely one, using the multi-class formulation described in [14]. The algorithm thus acquires a specific function $f_y(x)$ for each class $y \in \mathcal{Y}$, with $|\mathcal{Y}| = k$. Given a feature vectors $x \in \mathcal{X}$ representing a novel requisite, $SVM^{multiclass}$ allows to predict a specific class $y^* \in \mathcal{Y}$ by applying the discriminant function $y^* = \arg\max_{\mathbf{y} \in \mathcal{Y}} f_y(x_i)$, where $f_y(x) = w_y \cdot x$ is a linear classifier associated to each y.

3.1 A General Adaptive Architecture for Advanced RA

In this section we present the architecture of a requirement analysis system. It handles Requisite Documents and automatically extracts the information needed in the generic requirement management phase. In the next session, we will discuss how this system can be employed in a real use case as the underlying requirement management phase of a Combat Management System (CMS) is presented. This system processes semi-structured documents, i.e. written in natural language, and enriches texts with linguistic information employed by other modules. Then, all sentences expressing one or more requisites are retrieved and the target information is extracted. Interesting slots of the *templates* modeling different concepts in the requirement analysis are filled. These template instances are used to populate the *Requirement Repository* that can be later easily accessed by the analyst. Moreover, the system also analyzes the extracted information in order to recognize/acquire existing dependencies among different requisites.

In Figure 1, the overall architecture is shown and the interaction between the different functions that contribute to the main workflow, as well as their interactions and dependencies are reported. On the top of the architecture, the basic Natural Language Processing (NLP) chain is foreseen. This module carries out different NL steps needed to analyze a document, extracting all linguistic information useful to later modules in the chain. This includes steps such as a Sentence Splitter, a Part-of-Speech tagger, a Name Entity recognizer, a Word Sense Disambiguation module and a Syntactic Parser. These modules are based on different knowledge bases, modeling different aspects of the overall RA process:

[2] `http://svmlight.joachims.org/svm_multiclass.html`

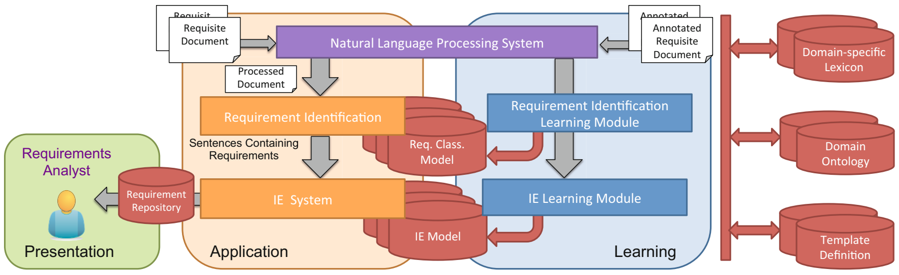


Fig. 1. Requisite Analysis System Architecture

– *Domain specific Lexicon*: it contains the specific domain dictionaries providing lexical information about the application domain, e.g. involved entities and acronyms.
– *Domain ontology*: it provides an ontological model of the application domain, as well as an abstraction of the requirements (i.e. the template for the Information Extraction activity). Moreover it provides the relations among different requirements, e.g. dependency rules among pre-conditions and post-conditions that enable the reasoning.
– *Template Definition*: it represents the repository of different templates involved in the IE activity, that are domain specific (as the ontology), but possibly more specific than the concepts or relations in the domain ontology.

According to our machine learning perspective, each module performs the corresponding task according to a model of the domain that has been automatically previously acquired from real data. These are requirement documents that have been previously annotated by the analysts, with the same information the IE system is expected to precisely detect in future texts. The general architecture is thus divided in different main blocks to distinguish models directly employed in the (on-line, i.e. interactive) Requirement Analysis Application workflow from the ones employed in the (*off-line*) Learning workflow. In the *Application Block* the following modules process requirement as follows:

– *Requirement Identification Module*: This module performs the analysis of documents that are enriched with linguistic information in order to suitably locate sentences containing concepts (and relations) of interest in the requirements analysis domain.
– *Information Extraction (IE) System*: Once a specific requisite is found, the extraction of its relevant information is carried out as a slot-filling process over the existing templates. Once a template is filled, it is made available (i.e. it populates) the Requirement Repository for the analysis.

In a machine learning perspective, each module performs the corresponding task according to a model of the domain that must be automatically acquired from annotated data. In this view, the second block in the architecture of Figure 1, i.e. the *Learning* block, is dedicated to the acquisition of the individual IE models. Finally, the *Presentation* block is responsible for the interaction with the analysts in (1) accessing the extracted information as well as (2) in providing feedback to the system in form of acceptance or rejection of some of its decisions.

4 Semantic Technologies in a Real Application Scenario

New Generation Naval Combat Systems are very complex systems based on a sw component able to manage all the Combat System Equipment (CSE) in different mission scenarios: the Combat Management System (CMS) as in [15]. The main objective of the CMS is to enable the Command Team to manage the ships CSE to conduct the missions in the scenarios. The CMS is mainly composed of a real time component (C2S) which provides the Combat System with facilities for the management of short term activities (Conduct of action); and a Command Support System (CSS), which provides the Combat System with facilities for the management of medium and long term activities in the conduct of operational tasks. From the functional point of view the C2S is decomposed into application segments that allow the system to perform the following functions:

1. global tactical picture compilation,
2. warfare missions conduction in different domains (Air, Surface, Submarine, Land) at platform and force level,
3. Tactical Data Link exchange data functions.

Each functional requirement of CMS is allocated to Computer Software Configuration Items (CSCI). CSCIs communicate exchanging data over the ship network through a common application layer. The communication principles are different according to the relationships among the components that communicate each other. Independently of the communication model the strategy is that each sw component shares data with the other system components to enable them, i.e. allow them to carry out their own functionalities. A key aspect for managing the overall CMS complexity is the design and description of CSCI interactions in terms of data each component has to *publish* for the benefit of the users. A system like the CMS has a large number of users, a large number of connections to CSE and heavy requirement on the processing applications that must be executed in real-time. Further constraints are given by the demanding performances, security requirements and by the incorporation of Off-The-Shelf software. It is clear how this class of systems needs clear requirement description and management throughout its entire system life cycle. The introduction of Semantic Technologies such as IE (as described in previous sections) in the Requirement Analysis process of naval CMS is bringing significant benefits in different phases of project life cycle. In particular, the application of Machine Learning techniques in the initial phase of the project has allowed filling the gap between the contractual technical specifications and system design description. Thanks to the machine learning method, the tool illustrated in this paper automates the process of recognition of several inferences about system components directly from the texts that characterize them, i.e. the requirements.

4.1 Experimental Evaluation

Machine-learning techniques for requirement analysis described in the previous Sections have been implemented in a Requirement Analysis System, according to the Architecture shown in Figure 1. The resulting adaptive system has been applied to a real scenario, i.e. the requirement analysis of a Naval Combat Systems, focusing on the SW system, namely the Combat Management System (CMS). This Section provides the empirical evaluation of system functionalities, such as the *Requirement Identification* (RI) and *Information Extraction* (IE) as applied to the CMS requirement analysis. Requisites here refer to different aspects of the CMS, such as *Functional Requirements* (FNC) or *Performance requirements* (PRF). The dataset adopted in our tests is made of 4,727 annotated requirements, related to three different scenarios, called *EAU, FREMM* and *NUM*. Each requisite has been labeled according to one of the five requisite types, which are specific aspects of the resulting system, such as FNC or PRF, as shown in Table 1.

Table 1. Requisite Types

ABBR	Type	Number
NFC	Non-Functional Requirements	74
DCC	Design and Construction Constraints	288
OPR	Operator requirements	2,587
PRF	Performance Requirements	249
FNC	Functional requirements	1,529
Total		4,727

The *Requirement Identification* system of Figure 1 has been trained to recognize and characterize requirements. The module applies Support Vector Machine classification to associate each requirement its suitable specific class, reflected into the corresponding type. Different models of observable text properties allowed to investigate different linguistic information and to identify the most informative representations for the learning algorithm:

- The *Bag-of-Word* (*BoW*) model mainly accounts for the lexical information: requisites are mapped into sets of words, neglecting word order, i.e. syntactic information.
- The *N-gram of Words* (*N-Words*) model provides a first form of grammatical information, by mapping short word sequences into n-grams of words.
- A *Bag-of-Word* and *N-gram of Part-of-Speech* (*N-POS*) introduces grammatical information as it attaches part-of-speech to n-grams, by further generalizing the sequences of words in the textual requisite.
- The *Comprehensive* (*BoW + N-Words + N-POS*) model accounts for all the previous information, i.e. as it includes Bag-of-Words, n-grams of Words and n-grams of Part-of-Speeches.

The objective of the experiments is also to measure and compare the adaption capabilities of SVM classifiers to different scenarios: the idea is that SVMs should be able to induce meaningful classification models from the data available in a specific scenario, i.e. the *in-domain* scenario, but also provide accurate predictions even when applied to different, i.e. *out-of-domain*, scenarios.

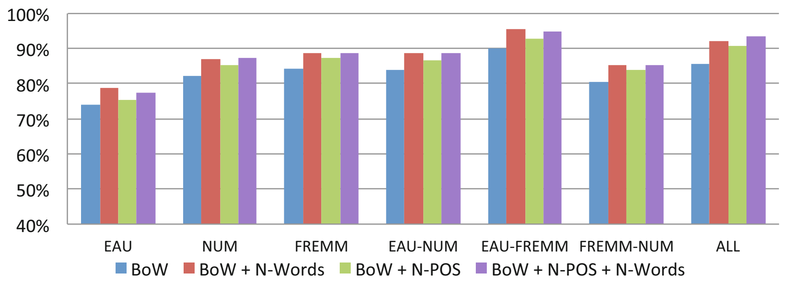


Fig. 2. Requisite Classification Results

Figure 2 reports the classification results, in terms of *accuracy*, i.e. the percentage of correctly classified requisites. Different colors reflect the different adopted feature models. The first three histograms provide results when classifiers are trained over requisite from one single scenario (i.e. EAU, NUM and FREMM respectively) and applied to the other remaining scenarios. The second group of histograms shows results when classifiers are trained over two scenarios (i.e. (EAU-NUM), (EAU-FREMM) and (FREMM-NUM)) and applied to requisites in the single remaining scenario. Finally, the last group shows results from an *in-domain* setting, when the 80% of requisites from all scenarios are used to train classifiers, while the remaining 20% are used as test set. In all experiments, SVM parameters are estimated over an held-out 20% of the training data. Results, especially when lexical and grammatical features are considered, i.e. the *BoW + N-words + N-POS* model, are very good and an accuracy higher of 93% is achieved. Moreover, the system robustness is very promising, as accuracy higher than 85% is reached even in *out-of-domain* tests. Errors refer to reasonable and genuinely ambiguous cases. For example, the system labels as FNC both requisites "*The CMS shall display the progress of each engagement.*" and "*The CMS shall display single manoeuvre request within ...*", although this latter is associated to OPR. Once a specific requisite is located, the *Information Extraction* (IE) System (Fig. 1) carries out the extraction of its relevant information, as a slot-filling process over the reference templates. Templates are automatically generated from the analysis of a *Domain Ontology*, which provided a model of the application domain as well as an abstraction of individual requirement types. These types ontologically determine different *capabilities*, i.e. desired characteristics of a target system. Moreover, the ontology provides hierarchies that group capabilities according to their semantics and the expected grain of analysis. Coarse grained capabilities refer to high level system characteristics, such as such Resource Management, that in turn groups together several fine-grained capabilities. These latter specialize the considered aspects, e.g. Navigation Radar(NAV), that specializes the notion of *Resource Management in Navigation Radar systems*. The IE system is asked to associate a requisite like "*The CMS shall monitor information transmitted by the Navigation Radar*" to the NAV template, recognizing its finer-grained aspect. The database of *Templates*, defined by the ontology, includes 65 templates that correspond to the range of the function mapping each requisite to its corresponding template. The high number of target class makes this task very challenging with respect to the previous requisite identification problem.

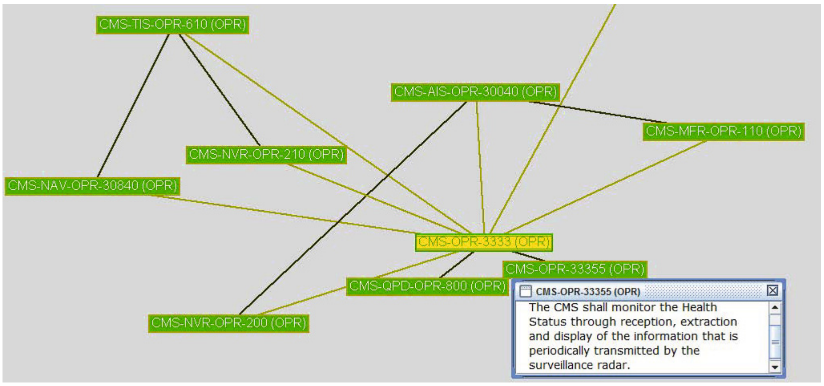
SVM classifiers have been employed even in this task: parameter estimation has not been employed, to prove the low dependence of the learning algorithm from external parameters; instead, results are reported as mean accuracy and (negligible) standard deviation in parenthesis. Requisites are here represented similarly to the previous task, thus employing the *BoW* model, that consider only lexical information, and the *BoW + N-Words* model that consider also the shallow syntactic information of the requisites. In term of percentage of requisite correctly covered by a template we have a classification results 87,61% (1,16%) with **Bow** models and 88,5% (1,46%) with **BoW + N-Words** model. Even in this evaluation, results show an accuracy of 88% proving the IE system as a largely applicable process.

4.2 Retrieval in Large Repositories of Software Documentation

In this section the contribution of the proposed approach for Requirements Analysis (RA) is investigated in an Information Retrieval scenario to improve the software reusability. In order to retrieve a piece of software or any other existing functionality that satisfies a specific user requirement, a Requirement Analyst usually retrieves existing documentation through a search engine through specific term-based queries. In a Ad-hoc Retrieval scenario [12], the quality of the retrieved material is strictly dependent from the expressed query that reflects user needs. In this section we instead define a robust search engine to enable the Requirement Analyst the retrieval of existing software functionalities by expressing software requirements in natural language.

The contribution of the proposed architecture is shown here to enable a more conceptual kind of search. The idea is that requirements determine complex queries can be processed by our RA software and used to retrieve existing software *compatible* with the functionalities expressed by the user requirements. More formally, the user express a requisite $r_i \in R$ in order to retrieve one of the specific functionalities $f_1^i, \ldots, f_{n_i}^i$ satisfying r_i. We denote the set of all $\{f_j^i | j = 1, ..., n_i\}$ as F_i and $n_i = |F_i|$. As an example, given a requisite r "*The CS shall provide facilities for Human Computer Interface presentation*", we would like to retrieve the implemented functionalities satisfying the specific need, such as the functionality f "*The CMS shall provide the following facilities at CMS consoles : screens, pointing device, keyboards, MFKA, service settings.*". In fact, in this case, the requisite r is satisfied by f, because it expresses the available facilities to interact with the software system. We collected all the pairs $RF = \{\langle r_i, F_i \rangle\}$, where r_i is a system requirement and F_i is the set of the corresponding functionalities satisfying r_i, denoted by $\{f_1^i,\ f_2^i,\ \ldots\ ,\ f_{n_i}^i\}$.

To associate a generic f to a given r, we first exploit the vector representation described in Section 4.1 that reflects the generic notion of semantic text similarity [16,17]. This is a geometric representation of textual meaning based on the the set of lexical and grammatical features expressed in the resulting weighted vector that establishes a variety of latent semantic relations between requirements: the closer two requirements are in the representation space, the stronger is their semantic relation.



The CMS shall monitor the Health Status through reception, extraction and display of the information that is periodically transmitted by the surveillance radar.

Fig. 3. Conceptual Graph employed to evaluate the semantic similarity function

A first use of this representation is the Graphical User Interface (GUI) to the database, that allows to analyze complex semantic relationships between individual requisites, such as the redundancy. As shown in Figure 3, individual requisites are represented through a conceptual graph where edges between vertices express weighted semantic similarity relationship between two instances. In Figure 3 the graph of requisites closer to the requisite **CMS-OPR-3333**, i.e. "*The CMS shall monitor the equipment Status through reception, extraction and display of the information that is periodically transmitted by the surveillance radar ...*" is shown. Notice how the most similar text is **CMS-OPR-33355**: "*The CMS shall monitor the Health Status through reception, extraction and display of the information that is periodically transmitted by the surveillance radar.*". This confirms that the captured notion of similarity well reflects rich semantic relations. This relationship instance is in fact a form of *textual entailment* [18], i.e. the directional relationship between a text pair $\langle T, H \rangle$, made by T, i.e. the entailing "*Text*", and H, i.e. the entailed "*Hypothesis*". It is usually stated that T entails H if a human that reads T (assuming it to be true) would accept that H is most likely true. This definition is somewhat informal but model an underlying useful form of commonsense knowledge for human expert.

The way a graph is built depends on the distance metrics established within the underlying vector space. Given a requisite r_i, the short texts describing functionalities f_j can be ranked according to their semantic similarity with the specific r_i, modeled through the cosine similarity sim between the corresponding vectors $\overrightarrow{r_i}$ and $\overrightarrow{f_j}$: $sim(r_i, f_j) = \frac{\overrightarrow{r_i} \cdot \overrightarrow{f_j}}{||\overrightarrow{r_i}|| \cdot ||\overrightarrow{f_j}||}$

In this evaluation, we considered a set of 290 requisites and 1,474 functionalities of the Combat Management System (CMS). The mean number of functionalities for each r was 5, with a standard deviation of about 3.5. For each r_i, the set F_i specifies all functionalities realizing r_i: these are the gold standard, i.e. the set of texts expected to be retrieved by the analyst querying by r_i. As they are short texts, individual r_i as well as f_j are modeled according to the *Comprehensive* model, i.e. the *BoW + N-POS + N-Words* vector representation, as it achieves the best results in the RA discussed in Section 4.1.

The information acquired during the RA phase is here exploited in order to define a *Re-ranking* phase: the ranking provided by the semantic similarity function is thus adjusted to filter out all those functionalities that do not share the same characterization of the target requirement r_i, i.e. the same *type* and *capability*, as discussed in Section 4.1. Four different retrieval strategies are applied, giving rise to four IR systems:

- **NoFilter**: for each r_i, the most similar f_j are retrieved and ranked according to *sim*: no filter is applied.
- **Type**: the ranking provided by *sim* is grouped in two lists: the first, ranked higher, is made by functionalities sharing the same type of r_i and a second list including the remaining f_j whose type is different. In this way functionalities f_j of the same type of r_i are always ranked first than the other ones.
- **Capability**: the two lists are created as before with respect to the capability assigned to the target r_i, so that functionalities with the same capabilities of r_i are ranked first;
- **Type+Capability**: the ranking provided by *sim* is modified as before according to the sharing the both type *and* capability of r_i.

Different strategies are evaluated according to standard IR evaluation metrics: *Precision* (P), *Recall* (R), *F-measure* ($F1$) and *Mean Average Precision* (MAP). Precision is expressed as $P = \frac{tp}{tp+fp}$, where tp is the number of the relevant functionalities retrieved, and fp is the number of the not relevant functionalities retrieved. Recall is expressed as $R = \frac{tp}{tp+fn}$, where fn is the number of the relevant functionalities not retrieved. While Precision estimates the capacity to retrieve correct functionalities, Recall is more interesting in this scenario as it measures system capacity to retrieve all existing functionalities; in many cases, it is more important to retrieve all existing software instead of spending more time reading useless documentation. F-measure consider both aspects as it is estimated as the harmonic mean of Precision and Recall: $F1 = \frac{2 \cdot P \cdot R}{P+R}$

Finally, MAP provides a single accuracy measure across different recall levels. MAP is based on the oracle given by $RF=\{\langle r_i, F_i \rangle\}$ that are pairs of a requisite r_i and a functionality set F_i. Every requisite r_i also corresponds to a ranked list of retrieved functionalities, ordered according to the similarity function *sim*. Let F_i^k be the list of retrieved functionalities f_j^i from the top result (i.e. f_1^i, ranked as the closest by the system) to the f_k^i that corresponds to the position where k-th members of the functionalities in F_i results all returned. In this way, the

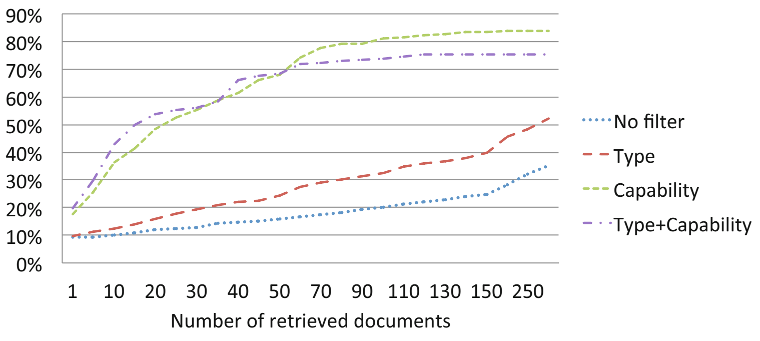


Fig. 4. System Recall

number of true positive functionalities for a requirement is exactly k. Then, the *Mean Average Precision* at level k of recall is denoted as:

$$MAP@k = \frac{1}{|RF|}\sum_{i=1}^{|RF|}\frac{1}{|F_i^k|}\sum_{k=1}^{|F_i^k|}P(F_i^k)$$

where $|F_i^k|$ obviously denotes the number of relevant functionalities for a given requisite r_i that varies with i, and $P(F_i^k)$ expresses the precision against the first k true positives for r_i. The results in terms of Recall with respect to increasing number of retrieved functionalities are shown in Figure 4 for different retrieval strategies. Moreover, Table 2 report evaluation figures for varying size of the retrieved document lists.

Table 2. Accuracy for different IR strategies and sizes of returned functionalities

		1	5	10	15	20	25	30	35	40	45	50
	P	0,158	0,036	0,021	0,016	0,014	0,012	0,011	0,010	0,010	0,009	0,008
No filter	R	0,092	0,093	0,102	0,107	0,122	0,124	0,128	0,142	0,147	0,152	0,160
	F1	0,116	0,052	0,035	0,028	0,026	0,022	0,020	0,019	0,018	0,017	0,016
	P	0,175	0,046	0,029	0,024	0,022	0,020	0,018	0,017	0,016	0,015	0,015
Type	R	0,097	0,112	0,126	0,141	0,157	0,178	0,196	0,211	0,219	0,226	0,242
	F1	0,125	0,065	0,047	0,042	0,038	0,036	0,033	0,032	0,030	0,029	0,028
	P	0,407	0,151	0,106	0,085	0,073	0,065	0,060	0,056	0,052	0,051	0,049
Capability	R	0,176	0,255	0,363	0,415	0,484	0,525	0,553	0,588	0,615	0,659	0,680
	F1	0,246	0,189	0,165	0,141	0,127	0,116	0,108	0,102	0,097	0,094	0,091
	P	0,488	0,167	0,113	0,089	0,073	0,066	0,060	0,055	0,050	0,047	0,044
Type+Capability	R	0,196	0,296	0,430	0,497	0,538	0,553	0,561	0,585	0,661	0,677	0,683
	F1	0,280	0,214	0,179	0,150	0,128	0,118	0,109	0,101	0,093	0,088	0,083

Moreover, in Table 3 the results of MAP are reported: rows correspond to different strategies while columns report different MAP values obtained when k is fixed to 1, 2 and 5, respectively. When no filter is applied, results are quite low, especially in term of Recall: when 50 functionalities are retrieved only 20% are usually relevant for the user. The high-level filter, represented by the *Type* strategy, improves results even if the difference is not very relevant. The Capability information produces a considerable improvement: when 50 functionalities are retrieved, more than 70% of them are relevant for the user. The filter that considers both type and capability is quite effective when few items are retrieved, as confirmed by the highest value of MAP achieved for lower levels of k.

Table 3. Mean Average Precision

Filter	**MAP@1**	**MAP@2**	**MAP@5**
No filter	0.139	0.126	0.117
Type	0.149	0.142	0.135
Capability	0.305	0.284	0.273
Type+Capability	0.368	0.354	0.347

Finally, a qualitative analysis of the retrieval accuracy can be carried out by studying some examples of returned functionalities. We queried the IR service by a system requirement r such as "*The CS shall provide facilities for Human Computer Interface presentation*". Table 4 shows the retrieved functionalities obtained by applying the combined filter (*type* and *capability*) to the input r. It is clear that the returned functionalities have a quite good relevance for the queries requisite, as the first hits in the Table show. Moreover, the quality of the relevance decrease along with the ranking: the third returned hit is much less relevant than the first two, despite properly respond to the requirement.

Table 4. Example of retrieved functionalities

r	*The CS shall provide facilities for Human Computer Interface presentation*
f_1^r	*The CMS shall provide similar controls and means of interaction with all displays, i.e. they should be the same where possible and consistent otherwise.*
f_2^r	*The CMS shall provide the following facilities at CMS consoles : screens, pointing device, keyboards, MFKA, service settings.*
f_3^r	*The CMS shall display alerts on primary view area*

5 Conclusions

While Semantic Technologies show a large set of promises in the Defense System Engineering domain, they are usually very demanding from the point of view of complexity in design, optimization and maintenance. Traditional (i.e. Knowledge-based) HLTs approaches are in this class of technologies. The results achieved in Statistical Natural Language Processing by the adoption of robust and accurate Machine Learning algorithms allowed to increase the applicability of these methods in several domain, from Business Analysis, Web Communication as well Security. In this paper, a general architecture for large scale and adaptive Requirement Analysis has been presented. Its application to Requirement Analysis in the specific Defense System Engineering domain is evaluated and discussed. The main idea is to combine requirement classification and IE for automation of most of the validation stages related to system behaviors. The system is currently experimented in a specific scenario of Combat System Equipment, applied to the management of the design and description of the Computer Software Configuration Items interactions. The application of semantic technologies in the Defense System Engineering domain has shown its potentials in the

Requirement Analysis process. In particular it can support significant cost reduction, products' quality enhancement as well as the improvement of Engineering processes.

References

1. Cook, D.: Evolution of programming languages and why a language is not enough to solve our problems (1999), http://lsc.fie.umich.mx/juan/Materias/FIE/Lenguajes/Slides/Papers/Evolution.html
2. Corazza, A., Di Martino, S., Maggio, V., Scanniello, G.: Combining machine learning and information retrieval techniques for software clustering. In: Moschitti, A., Scandariato, R. (eds.) EternalS 2011. CCIS, vol. 255, pp. 42–60. Springer, Heidelberg (2012)
3. Bennaceur, A., Johansson, R., Moschitti, A., Spalazzese, R., Sykes, D., Saadi, R., Issarny, V.: Inferring affordances using learning techniques. In: Moschitti, A., Scandariato, R. (eds.) EternalS 2011. CCIS, vol. 255, pp. 79–87. Springer, Heidelberg (2012)
4. Abbott, R.J.: Program design by informal English descriptions. Communications of the ACM 26(11), 882–894 (1983)
5. Saeki, M., Horai, H., Enomoto, H.: Software development process from natural language specification. In: Proceedings of the 11th International Conference on Software Engineering, New York, NY, USA, pp. 64–73 (1989)
6. Sawyer, P., Rayson, P., Garside, R.: Revere: Support for requirements synthesis from documents. Information Systems Frontiers 4(3), 343–353 (2002)
7. Chitchyan, R., Rashid, A., Rayson, P., Waters, R.: Semantics-based composition for aspect-oriented requirements engineering. In: Proceedings of AOSD, pp. 36–48. ACM, New York (2007)
8. Grishman, R.: Information extraction: Techniques and challenges. In: Pazienza, M.T. (ed.) SCIE 1997. LNCS, vol. 1299, pp. 10–27. Springer, Heidelberg (1997)
9. Manning, C.D., Schütze, H.: Foundations of statistical natural language processing. MIT Press, Cambridge (1999)
10. Gildea, D., Jurafsky, D.: Automatic Labeling of Semantic Roles. Computational Linguistics 28(3), 245–288 (2002)
11. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, New York (1995)
12. Basili, R., Moschitti, A.: Automatic Text Categorization: from Information Retrieval to Support Vector Learning. Aracne (2005)
13. Joachims, T., Finley, T., Yu, C.N.: Cutting-plane training of structural SVMs. Machine Learning 77(1), 27–59 (2009)
14. Crammer, K., Singer, Y.: On the algorithmic implementation of multi-class SVMS. Journal of Machine Learning Research 2, 265–292 (2001)
15. Ciambra, F., Nardini, M.: Naval combat system design: System engineering approach and complexity management. In: Proceedings of INCOSE, France (2004)
16. Agirre, E., Cer, D., Diab, M., Gonzalez-Agirre, A.: Semeval-2012 task 6: A pilot on semantic textual similarity. In: *SEM 2012: The First Joint Conference on Lexical and Computational Semantics, Montréal, Canada (2012)
17. Mihalcea, R., Corley, C., Strapparava, C.: Corpus-based and knowledge-based measures of text semantic similarity. In: AAAI 2006 (2006)
18. Dagan, I., Glickman, O.: Probabilistic Textual Entailment: Generic Applied Modeling of Language Variability. In: Learning Methods for Text Understanding and Mining (January 2004)