Keyword-based, Context-aware Selection of Natural Language Query Patterns*

Giorgio Orsi
Politecnico di Milano
P.zza Leonardo da Vinci 32
20133 - Milan, Italy
orsi@elet.polimi.it

Letizia Tanca
Politecnico di Milano
P.zza Leonardo da Vinci 32
20133 - Milan, Italy
tanca@elet.polimi.it

Eugenio Zimeo
Universitá del Sannio
Piazza Guerrazzi 1
82100 - Benevento, Italy
zimeo@unisannio.it

ABSTRACT

Pervasive access to distributed data sources by means of mobile devices is becoming a frequent realistic operational context in many application domains. In these scenarios data access may be thwarted by the scarce knowledge that users have of the application and of the underlying data schemas and complicated by limited query interfaces, due to the small size of the devices.

A viable solution to this problem could be expressing the queries in natural language; however, in applications like medical emergencies, data management systems must obey requirements such as very fast and precise data access which make this solution infeasible.

To reduce the time needed to get answers to user queries, the paper proposes a lightweight, context-aware approach based on the combination of keywords with natural language queries. The method employs ontologies and query patterns to support the users in formulating the most appropriate query for retrieving the desired data. Precision and query efficiency are further improved by focusing searches only to the data which are meaningful w.r.t. the current context, thus supporting the users' situation awareness.

The approach has been integrated in the SAFE system, developed for mobile and Web, and has been applied in cardiology to support medical personnel in emergency interventions on patients affected by chronic cerebro-vascular diseases. Experimental results have shown that the proposed solution significantly reduces the time to get useful data w.r.t. traditional form-based approaches.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Query formulation*; J.3 [**Computer Applications**]: Life and Medical Sciences—*Medical Information Systems*

General Terms

Design, Human Factors, Experimentation

Keywords

Ontology, Keyword Search, Context-Awareness, Emergency Management, Natural Language

1. INTRODUCTION

The availability of small devices able to perform complex computations (e.g., tablets, pads, smart-phones, etc.) is opening the way to new working models characterised by pervasive applications and mobile personal computing systems. In such an environment, there is an increasing need for seamless access to distributed and heterogeneous data sources. However, users have often a very limited knowledge of the information content and organisation underlying their applications, and consequently retrieving useful data in a precise, fast and exhaustive fashion requires either time or an a-priori technological training.

In addition, the compactness of many mobile devices makes it difficult to design effective (i.e., complete and easy to use) user-interfaces to support *data access*, *retrieval* and *querying*. In particular, approaches based on forms and SQL queries, that are very effective with desktop or web applications, may turn users away since keyboards are small or virtual, frequent scrolling is required to reach the desired fields, and small-size screens prevent easy reading of query results. Despite the fact that better user-interface design may partially solve some of these problems we believe that a better solution resides *in a different user-device interaction paradigm.*

A possible course of action consists in allowing users to formulate queries in natural language. However, (semi)automatic generation of queries from natural language must address the problem of recognising the intended meaning of sentences [34], a problem that, despite the many advances obtained by means of quantitative techniques, is computationally very expensive and still requires complex training phases on large text corpora. Moreover, building a complete sentence in natural language and transforming it into adequate queries towards a database introduces two levels of complexity that contribute to increasing the time needed to obtain the desired information.

A more effective approach aims at reducing the problem to keyword-based search, avoiding the burdens of building and processing complete sentences expressed in natural language. However, pure keyword-based solutions are affected by two major problems: (i) most of the information conveyed by natural language is hidden within the structure of the sentences, thus a list of keywords

*This research is framed within the SAFE project funded by Campania Region in the context of Misura 3.17 POR 2000/06, and has also been partially funded by the European Commission, Programme IDEASERC, Project 227977-SMScom. We wish to thank Pierluca Lino and Quirino Zagarese for their invaluable contribution to the SAFE source-code development, Ester Giallonardo and METODA S.r.l. for their support in the definition of the domain ontology and of the system's architecture.

has often an imprecise or ambiguous semantics; (ii) in order to input the keywords, the user must have at least an approximate idea of the available information. These are critical aspects in emergency situations, where speed and precision (in the sense of matching users' objectives) are of paramount importance, especially for naive users.

The paper proposes a lightweight approach to mobile search in emergency situations, based on semantic technologies and supported by context awareness. Ontologies are employed to match the keyword semantics against predefined query patterns, which are prompted to the users for them to identify the most appropriate query. Precision and speed are enhanced by exploiting context-awareness, which suggests a way for tailoring the ontology, and consequently query templates and searches, to the current context.

Our approach is particularly interesting in application scenarios that require flexibility in composing a query and high performance to access data in very short time. One such situation is the treatment of emergency in health-care environments and in particular in cardiology. Here, the fundamental information is represented by the patient's anamnesis that must be (i) accessed very fast and (ii) restricted to only those data that are pertinent to the current emergency situation, in order to avoid death or to affect severely (i.e., irrecoverably) the patient's health.

In such scenarios, each chronic patient is provided with a smart-card (SAFE-Card) that stores (i) medical information about the patient and (ii) suitable context meta-data that affect the way the medical data will be accessed by the system. A central organisation (e.g. a federation of health-care structures) provides access to a federated information system, when needed. This endpoint is reachable through the web by means of a GSM/UMTS connection reserved to emergency services. However, an essential requirement of SAFE is the capability to support the health-care personnel even in conditions of zero-connectivity with the central system.

SAFE prompts the user (a paramedic or a doctor) with a list of parametric query patterns expressed in natural language that are related to the keywords given as input. Once the user has identified the queries that s/he would like to execute, SAFE will answer them by using the data stored into the smart-card or by accessing the central information system (if reachable). Therefore, the approach can be used both in mobile and desktop-based scenarios, as described by the two following use cases:.

`c1` - *mobile emergency*: a team of paramedics is sent to rescue an unconscious patient (owning a smart-card called "SAFE-Card") hit by a heart-stroke in an area with limited connectivity (i.e., no broadband over the cellular network). The paramedics are provided with mobile devices capable of reading the content of the SAFE-Card. With this equipment, they are able to retrieve as much information as possible about the patient's recent anamnesis (e.g., chronic diseases and subjection to heart-strokes) plus any information that could be of interest for handling the emergency – e.g., if the patient has had a cardio-stimulator planted, rather than employing a defibrillator the paramedics must resort to manual heart-massage, otherwise the discharge may damage the cardio-stimulator and the patient dies before reaching the hospital. Note that, in this situation, an easy interaction paradigm is somewhat mandatory since paramedics may have to interact with the device in difficult environments (rain, low visibility, etc.) possibly wearing gloves or single-handed.

`c2`*: - first aid and pre-surgery briefing*: a patient is carried to the first-aid department of a hospital after preliminary intervention in the ambulance during transportation. The paramedics at the hospital use the patient's SAFE-Card to immediately access his or her anamnesis through the SAFE Web application, and to automatically book the needed clinical exams and the surgery room for a possible operation. The exams suggest to proceed with surgery within one day. The team is now discussing before doing surgery on the patient: the anaesthetist needs to access the SAFE Web application through a terminal in the briefing room, to retrieve the information about all the pharmacological treatments prescribed to the patient – e.g., if the patient is under therapy with vasodilator or anti-aggregating chemicals, the administration of any further drug that empowers the effect of these chemicals might lead to massive haemorrhages during the intervention, jeopardising the patient's life.

Both uses cases show the need for novel tools and technologies able to assist and expedite the access data which are meaningful w.r.t a given context. Accordingly, the main scientific contributions of the paper are: (i) an architecture for fast and adaptive keyword-based retrieval of medical information in emergency situations, (ii) a structure for Query Patterns associating a query in natural language with a query in a concrete query language. (iii) a context-aware, semantic ranking algorithm for Query Patterns w.r.t. an ordered set of keywords and (iv) an ontology-based context model suitable for a-priori definition of context-relevant areas in an ontology. An additional and important contribution is the implementation and the integration of the proposed techniques in a real system that supports fast and adaptive keyword-based retrieval of medical information in emergency situations.

We assume that the reader be familiar with relational databases and keyword-based search engines. Moreover, in the following sections we make an extensive use of ontologies. An ontology is a formal, conceptual specification of a domain of interests [16], modelled as a 4-tuple $\mathcal{O} =< N_C, N_R, N_U, N_I >$ where N_C is a set of concept names (i.e., classes or types), N_R is a set of role names (i.e., binary relations between concepts), N_U is a set of attribute names (i.e., binary relations between a concept and a data-type) and N_I is a set of individual names (i.e., constants). In this paper we consider Description Logics (DLs) ontologies [4] structured as a pair $\mathcal{KB} =< T, A >$, where T (the TBox) is a set of terminological axioms (i.e., intensional knowledge) involving elements in $N_C \cup N_R \cup N_U$, while A (the ABox) is a set of facts (i.e., extensional knowledge) about elements in N_I.

The paper is organised as follows: Section 2 compares SAFE with the literature, analysing the main results of the works more related to the technique we propose. In Section 3 we go into the details of the keyword-based query answering approach adopted in SAFE while Section 4 describes how context-awareness is achieved by means of a context model, and its effect on the SAFE system. Finally, Section 5 is dedicated to the evaluation and experimental setting while Section 6 draws some conclusions and proposes some extensions to this work.

2. RELATED WORK

In SAFE we combine techniques coming from various research fields namely: information retrieval [5], databases [1] and pervasive, context-aware systems [15, 24].

Enabling a simple and intuitive access to databases is the domain of *keyword-based database querying* [13], a research area which is relatively recent for the database community, where the data are accessed through keywords instead of using structured query languages such as SQL. This approach has proved effective when the databases have complex, unknown or evolving schemata since the user does not need to know the logical structure of the underlying database. These techniques mainly result from the hybridisation of databases with *information retrieval* (IR), where keywords are instead used for ranking and retrieve relevant documents from (large) collections. Relevant works on this topic include BANKS [6], DBXplorer [2], Discover [17], SQAK [31] and ObjectRank [18], where the user provides a set of keywords as input and the system returns, as answer, a set of *tuple-trees* i.e., inter-connected tuples that are related to *all and only* the keywords given as input. In all these proposals the returned tuples must match the keywords given as input thus increasing the precision of the answers. However, this semantics is considered too restrictive for most of the applications, since the user might not know the exact names of the entities s/he wants to query and this causes a limited recall especially when the database schema is unknown. In order to overcome this problem, taxonomy- and ontology-based approaches have been proposed. These use a reference semantic structure to enrich (i.e., expand) the set of keywords in order to retrieve also tuples that are not explicitly related to the original keywords (e.g., they are related to hyponyms of the keywords). Alternatively, approaches such as NaLIX [21] use natural language sentences instead of keywords. This reduces the uncertainty resulting from the intrinsic ambiguity of flat keywords but adds the burden of natural language processing, whose computational requirements are still unsuitable for mobile environments. An interesting approach bridging keywords and structured queries is that of QUICK [25], that proposes a *structured-keyword* query language where that explicitly uses domain relationships to relate the keywords. In particular, it is possible to explicitly represent typing (e.g., `John` *`is-a`* `person`) and relationships (e.g., `Jimi Hendrix` *`plays(guitar)`*). However, this language is limited to very particular structures (e.g., it can express only targets of a relation and not their subjects), while SAFE can automatically identify the relations between the keywords using the terms in the ontology. In particular the presence of a relation `affects` in the ontology with domain *disease* and range *system* will lead to suggesting the terms `disease` and `system` (along with their synonyms) whenever the keyword `affect` is typed into the system.

Differently from the above systems, SAFE cannot afford the luxury of inexact answers due to the particular constraints imposed by the emergency setting and, therefore, the queries that are eventually executed on the database are hand-made, structured query patterns. Keywords are used only to produce a ranking over the available query patterns that, in turn, carry a curated natural-language description of the semantics of the query. This eliminates the remaining uncertainty that can be introduced by the keyword matching processing that generates the ranking of the patterns.

Since keyword-based querying is schema-less, a natural issue is *keyword identification* i.e., users must be supported during the formulation of the keywords to avoid errors. In [14], the input keywords are auto-completed by looking-up the entire content of the database, and different scalable algorithm for this task are presented. However, in order to be efficient, these techniques require an additional memory consumption ranging from 200 to 300 megabytes for medium-sized databases ($\sim$500k entries). Even the most space-efficient technique presented in [14] will require around 4 GBs of additional memory to deal with a large medical ontology such as SNOMED [30], that is at the limit of the memory capabilities of current mobile devices. In SAFE we go beyond auto-completion by providing the user with a set of related keywords based on the ontology and on the context of the user. Differently from taxonomy-based approaches we go beyond classic linguistic relationships by exploiting domain-specific relationships such as the property of a concept (i.e., the name or SSN of a patient) and relationships between terms (e.g., the fact that a drug interacts with other substances).

When semantic technologies are combined with IR techniques we speak of *Semantic Search* [20,26,32]. In [19], a reference taxonomy is used to perform keyword-based querying over a database of medical records. Given a keyword, the system computes all the synonyms and hyponyms of the term given as input; these terms are then searched within the database using pre-defined queries. However, due to the storage model adopted for the ontology, this technique requires recursive SQL for ontology querying while, in SAFE, querying can be reduced to answering conjunctive queries that, differently from full SQL, can be answered very efficiently. Si-SEEKER [37] combines the IR engine of a RDBMS with a Semantic Search Algorithm that uses manually-generated textual annotations for the data. In both the previous approaches, only the sub-class relationships are taken into account, thus reducing the domain ontology to a simple taxonomy; we exploit a broader class of ontological relationships (with the aim of improving recall), while keeping a clear understanding of the influence that the expressive power of the adopted ontological language has on query processing efficiency. A related work in this sense is that of Search-WebDB [33] where the keywords are used to generate SPARQL conjunctive queries by exploring an RDF graph; such queries will be then answered over the same RDF graph. This approach is affected by the same ambiguity problem affecting the query generation techniques developed for relational databases since SPARQL is only a syntactic variant of non-recursive SQL. However, the the authors also propose a graph summarisation strategy to improve the graph exploration step required to derive the structured queries from the keywords. Since summarisation is affected by uncertainty, in SAFE we use an a-priori context-based tailoring of the ontology in order to explore the ontology graph efficiently for keyword matching.

The constraints imposed by the pervasive and mobile setting of some SAFE users, as well as the time constraints induced by the emergency situations to be managed, led us to study how to apply *Pervasive Data Management* [24] techniques to focus the search-space of the queries while maintaining a decent response-time. The main solution was found in context-aware techniques.

The notion of *context*, formerly emerged in various fields of research like psychology and philosophy [12], has been recently studied also by the computer science community (in particular in the Knowledge Representation area [11,35]), in order to find suitable models that can be used to embed the contextual dependency in software systems [9]. According to [15], we refer to context as *any information that can be used to characterise the situation of an entity*, where an entity is a person, a place, or an object that is considered as relevant for the interaction between a user and an application. Context-awareness is a very common tool in medical information systems especially when combined with IR techniques: [22] enriches a standard IR engine for medical documents with the UMLS ontology[1] and a context model. The ontology is used as a reference index; the terms in the documents are linked to the resources of the ontology to increase the recall of the retrieval process, while the context-model is used to annotate the documents with meta-data about the context of the document's author. In MIS-earch [28], a search engine for medical information, the context meta-data are extracted from the PHR (Patient Health Record) of the user. In this system, the original query is forwarded to Google and the contextual data are used to further filter and rank the search

[1] http://www.nlm.nih.gov/research/umls/

results. In both [22] and [28], the context meta-data are biased toward either the document author's or the patient's perspective and they fail to represent other contexts of fruition of the information (e.g., the situation or the device capabilities). SAFE uses the information stored in the SAFE Card in a similar way, however its context-model, based on the one presented in [10], is much more expressive than those adopted in the previous two systems and can model all the necessary perspectives (i.e., situations) involving users, systems and applications.

3. FROM KEYWORDS TO QUERIES

SAFE receives as input a *sequence of keywords* and produces, as output, a *ranking over a set of query patterns*, possibly with a suggested assignment for their parameters.

A typical execution flow (Figure 1) starts with an ordered set of keywords $< k_1, ..., k_n >$ given as inputs to one of the SAFE user applications (A). The keywords determine a ranking of the natural-language patterns $< p_1, ..., p_m >$ that are proposed to the user along with a suggested assignment for their parameters (B). The user can then further specify/override the parameter assignments for the patterns $< p_i, ..., p_j >$ selected for execution (C). Afterwards, depending on the SAFE application being used, the corresponding queries $< q_i, ..., q_j >$ are answered using the chunks (D) or using the SPARQL Endpoint (E), in the case of using a mobile device or the Web-based application, respectively.

Keyword interpretation is based on their comparison with the resources of the *domain ontology* and on the *current context* of the user. The following data structures constitute the information backbone on which SAFE relies:

- the *domain ontology* (SAFE-DO) that extends the relational schema of the SAFE-DB by means of additional structures. The domain ontology is defined using OWL2-EL based on the description logic $\mathcal{EL}^{++}$ [3], a DL for which query-answering remains tractable (i.e., PTIME in data-complexity) while providing sufficient expressive power for the representation of medical and biological ontologies [29].

- A set of SAFE *chunks*, context-aware fragments of the domain ontology, which represent the data over which the queries are evaluated on the mobile device. Their ABox instances are retrieved from the SAFE-DB and kept cached on the mobile device. The TBox T_c of each chunk is determined at design-time while the corresponding ABox A_c is retrieved through the SAFE Context Server on-demand and materialised as triples on the file-system of the mobile device.

- a set of contextual specifications representing the different contexts the user can be in. Each context is associated with the chunk TBox T_c which must be considered in order to (i) suggest keywords to the user, (ii) compute the ranking of the queries and (iii) answer the queries. The context is modelled using the description logic $\mathcal{ALCQ}(\mathbf{D})$, that provides constructs like *qualified number restrictions* and others needed for modelling contexts.

- A set of configurable Query Patterns representing the associations between a query in natural language and a query pattern in SPARQL[2], a query language for RDF [3], which is actually a syntactic variant of the Relational Algebra fragment of SQL [23]. In SAFE we consider *conjunctive* SPARQL queries, corresponding to the select-project-join queries of Relational Algebra. Query Patterns are thoroughly explained in Section 3.2

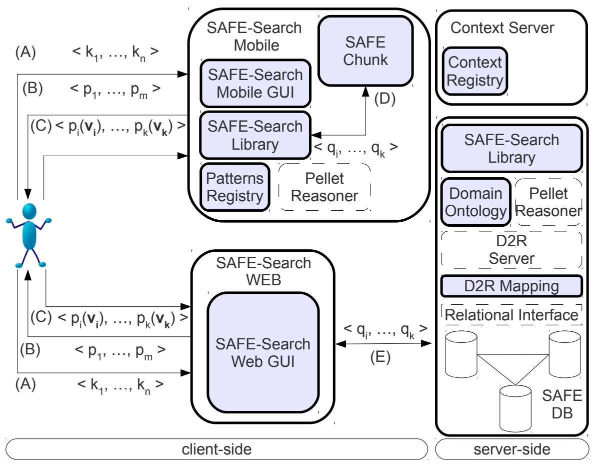


Figure 1: SAFE **Architecture**

In SAFE the queries, in SPARQL, are posed against the domain ontology – through the Web interface – or against the *chunks* – when operating in mobile contexts –. Then, they are translated into equivalent conjunctive queries and computed by resorting to the query-answering service of the Pellet reasoner[4], embedded in SAFE and used every time a query is issued. The SPARQL endpoint is accessed only in two situations: (i) by the SAFE-Web Application in order to answer a query and (ii) by the SAFE Context Server in order to compute the chunk related to a context.

We now discuss in detail (i) how the keywords are "interpreted" by the system, (ii) how they are refined and (iii) how they contribute to the ranking of the query patterns.

3.1 Keyword-to-Ontology Matching

In SAFE, the input keywords are first refined using the ontology as a controlled vocabulary; then, other keywords are suggested to the user to better specify the query. An important aspect of the input mechanism adopted in SAFE is the possibility to specify as keywords both *search terms* (e.g., `patient`, `drug`, etc.) and *parameters* (e.g., `John Smith`, `Diazepam`, etc.). The system will leverage on the ontology and the type of the parameter (e.g., integer, string, etc.) to recognise whether a given keyword should be used to rank the patterns or as a value for a pattern's parameter field.

A sequence of *lemmas* $w = < w_1, w_2, \ldots, w_n >$ is a *keyword* if the lemmas of w are considered as a unique concept by the user, e.g., a user might consider each of the two sequences of lemmas <`heart stroke`> and <`low blood pressure`> as a unique keyword or not. Both the mobile and the web applications in SAFE provide the users with the needed visual support in the GUIs in order to explicitly denote whether a sequence of lemmas is part of a unique keyword. This is different from classic IR where either every lemma is considered as a distinct keyword or compound keywords must be explicitly connected by means of boolean operators (i.e., AND, OR, NOT, etc.) or quotes. The SAFE keyword-interface accepts sets of keywords and uses them to retrieve a corresponding conjunctive query pattern from an existing collection. In this way, we

[2]http://www.w3.org/TR/rdf-sparql-query/
[3]http://www.w3.org/RDF/
[4]http://clarkparsia.com/pellet/

can straightly associate the results of the search to the keywords typed as input.

A user keyword may be typed into the system by the user or picked from a set of suggested keywords usually related to the keywords already provided. The suggestion mechanism is contemporaneous to typing and operates as follows:

Let $\mathcal{L}_{\mathcal{O}}$ be the set of all the labels (i.e., the natural language descriptions) associated with concepts and roles in the domain ontology $\mathcal{O}$ and $\mathcal{K}$ and $\mathcal{S}$ be two ordered sets of keywords. At each moment, $\mathcal{K}$ contains the keywords that have been already "confirmed" by the user while $\mathcal{S}$ contains the keywords that are being *suggested* to the user by the system. The function $pos(k,\mathcal{K})$ denotes the position of a keyword k in $\mathcal{S}$ and it is used to suggest more pertinent keywords first to the user, while $match^{JW}(s,t)$ denotes the Jaro-Winkler [36] distance between two strings s and t.

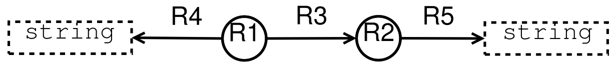
The sets $\mathcal{K}$ and $\mathcal{S}$ are initially both empty. Every time the user inputs a character, the system considers that character as part of a possible keyword w being typed by the user. During the typing process, the function $match^{JW}(w,l)$ is computed between the string w that is being typed in and each of the labels $l \in \mathcal{L}_{\mathcal{O}}$. The system adds to $\mathcal{S}$ the top-m labels l in the ontology for which $match^{JW}(w,l) > t$ – where m (i.e., the number of suggestions) and t (i.e., the similarity threshold) are defined at configuration time – and proposes them to the user; it is worth noting that, during this process, the ontology is used as a controlled dictionary of possible keywords. Whenever the user selects a keyword k from $\mathcal{S}$, the string w is replaced by k that is, in turn, added to $\mathcal{K}$. If the user does not select any of the suggested keywords from $\mathcal{S}$ and confirms the given input string w to be considered as a keyword, the system associates it with the resources in the ontology $\mathcal{O}$ whose labels are the best matches for w. If no $l \in \mathcal{L}_{\mathcal{O}}$ exists such that $match^{JW}(w,l) > t$, then w is interpreted as a possible *value* (i.e., a *hint*) for the parameters of the query patterns and it will not be associated with any resource in the ontology.

For each suggested keyword k, its position in S (i.e., $pos(k,\mathcal{K})$) is computed on the basis of (i) how frequently k has been used in past searches and (ii) how much k represents a pertinent concept w.r.t. the ontology resources associated with the other keywords in $\mathcal{K}$.

The *usage frequency* $freq(k)$ of a keyword k is computed through the analysis of the system's logs. Every time a keyword k is typed into the system, it is logged and added to the statistical base. The value of $freq(k)$ is then computed as the number of times it has been used in a search over the total number of searches executed by the system.

The *pertinence* of a resource r w.r.t a generic set of keywords $\mathcal{K}$ (i.e., $pert(r,\mathcal{K})$) denotes how much r is related to the resources associated with the keywords contained in $\mathcal{K}$.

Assume $\mathcal{K}$={drug, ascriptin} and the following ontology fragment:



where circles represent concepts, arrows represent roles and dotted rectangles represent data-types. The resources (concepts and roles) carry with the following labels: $\mathcal{L}_{R_1}$={drug, pharmaceutical, medicinal}, $\mathcal{L}_{R_2}$={disease, condition, illness, sickness}, $\mathcal{L}_{R_3}$={treat, cure, heal}, $\mathcal{L}_{R_4}$={name}, $\mathcal{L}_{R_5}$={code}. The computation of the pertinence is carried out in three steps:

1. SAFE associates each keyword k with the resource r_k whose label is the best match for k. The association is established by first decorating each $r \in \mathcal{O}$ with a property $m_k(r)$ whose value is $m(k) = max_{l \in \mathcal{L}_k}(match^{JW}(k,l))$ (i.e., the best match), where $\mathcal{L}_k$ is the set of labels associated with the resource r in the ontology. Then, r_k is the resource such that $m_k(r_k) = max_{r \in \mathcal{O}} m_k(r)$. Notice that, whenever a keyword is selected by the user from the ordered set $\mathcal{S}$ of suggestions, the association is certain (i.e., $m(k) = 1$) since the selected keyword comes from those taken from those available in the ontology. With respect to the above example, we associate a property m_{drug} to R_1 with value $v = 1.0$ because the keyword matches perfectly a label from $\mathcal{L}_{R_1}$, while the keyword "ascriptin" (which is the name of a particular cardio-aspirin) is not associated to any of the ontological resources.

2. A subsequent phase progressively decorates the *neighbours* $< t_1,\ldots,t_n >$ of r_k in the ontology (i.e., resources directly connected to r_k); for each $j \in \{1,\ldots n\}$, the value of $m_k(t_j)$ is computed as $m_k(t_j) = \sum_{t_i \in T} \tau \cdot m_k(t_i)$ where T is the set of all the resources t_i which are in their turn directly connected to t_j, that is, the neighbourhood of t_j: intuitively, $m_k(t_j)$ registers how much t_j is related to r_k. The decoration process is repeated for n hops, determining a set of resources that represents an area in the ontology that is related to r_k, and thus to the keyword k.

   The value of τ takes into account the types of the resources (classes and properties) and determines an area in the ontology whose resources are somewhat related to the input keywords. In the current version of SAFE, after experimental trials, the values for τ are set as in Table 1

   Table 1: Decoration function (τ) coefficients.

t_i	t_j	τ
concept	attribute	$\frac{m_k(t_i)}{\#number\ of\ attributes\ of\ t_i}$
concept	hyponym	$m_k(t_i)$
concept	hypernym	$\frac{m_k(t_i)}{\#number\ of\ hypernyms\ of\ t_i}$
concept	role	$\frac{m_k(t_i)}{\#number\ of\ roles\ whose\ t_i\ is\ the\ domain}$
attribute	domain	$m_k(t_i)$
role	domain	$\frac{m_k(t_i)}{2}$
role	range	$\frac{m_k(t_i)}{2}$

   Instead, the choice of n is configurable on the basis of the structure of the domain ontology. In general, n is greater than zero and limited by the maximum path length identifiable in the ontology graph. The output of this process for the example above is the set of annotations {(R_1, m_{drug}, 1.0), (R_2, m_{drug}, 0.25), (R_3, m_{drug}, 0.5), (R_4, m_{drug}, 0.5), (R_5, m_{drug}, 0.25)}.

3. The final step computes the pertinence w.r.t. $\mathcal{K}$ of each resource r (not necessarily associated with a keyword) that is simply the average among all the values of the properties m_k (for each $k \in \mathcal{K}$) decorating r.

   $$pert(r,\mathcal{K}) = avg_{k \in \mathcal{K}}(m_k(r)) \tag{1}$$

   With respect to the example above, due to the presence of a single keyword in $\mathcal{K}$ the pertinence values correspond to the values of m_{drug}.

Let now $R_{mathcalS}$ be the set of resources associated with the keywords $k \in \mathcal{S}$. For each $k \in \mathcal{S}$, its position is then determined through the function $pos(k,\mathcal{S})$, which is computed as a linear combination of its frequency and of the pertinence value of the resource r_k associated with k:

$$pos(k,\mathcal{K}) = avg_{r_k \in R_k}(pert(r_k,\mathcal{K})) + \beta(t) \cdot freq(k) \quad (2)$$

where $\beta(t)$ is computed as $\frac{1}{2} - \frac{1}{2 \cdot n(t)}$ and $n(t)$ is the number of different keywords used in searches up to the instant t. Notice that $n(t) > 0$ for each instant $t > t_0$. This factor is used to ensure that the statistical base is large enough before proposing a keyword on the basis of its usage frequency. In this way, the pertinence factor will be predominant in small statistical bases, while the two factors will be asymptotically considered as equally important, while the system collects more keywords. Following the example above, if we assume $\beta(t) = 0$ and $n = 5$, the set of suggested keywords is $\mathcal{S}$={name, treat, cure, heal, condition} while the keyword "ascriptin" will be considered as a parameter value.

We are now going to describe how the pertinence of the resources affects the ranking of the query patterns.

3.2 Ontology-driven Query-Pattern Selection

In SAFE, all the queries are presented to the user in natural language. Each query in natural language is associated with a SPARQL query over the domain ontology through a pattern structure. Each pattern is enriched with suitable meta-data that specify the intended semantics of the natural language query that, in turn, determines whether or not a particular query should be shown to the user and its position in the ranking. Each *query pattern* is specified by the following elements:

NL-Query: encodes the natural-language query and its variables. As an example, consider the situation of the use-case `c1` and the query: "`show the risk factors leading to heart-stroke in John Smith's anamnesis`". The corresponding *NL-Query* is: `Show the risk factors leading to $pat in $id's anamnesis`, where `$pat` and `$id` are variables to be bound.

Variable Bindings: in SAFE, each variable is associated with an XML datatype but it is possible to specify the admissible values to either (i) a default value, (ii) a set of possible values (i.e., enumeration) or (iii) a user-defined function.

Formal Query: encodes the SPARQL query associated with the NL-Query. The two queries share the variables defined in the variable bindings section, and the values fed into the NL-Query are transferred to the SPARQL query before its evaluation. Note that this separation is important since, in general, a very simple NL-Query might correspond to a complex SPARQL query.

Formal Resources: lists the ontology resources that specify the semantics of the NL-Query. These resources are used to determine how much a query is related to a set of keywords and its ranking among the results. The formal resources are also used to decouple the ranking process from the terms explicitly used in the SPARQL query thus making the ranking independent of the structure of the SPARQL query.

By leveraging on the value of *pertinence* associated to the formal resources it is now possible to determine whether a NL-Query should be shown to the user and in which position.

The ranking of a query pattern (denoted by $ran(p)$) is determined by two factors: (i) the pertinence and the number of formal resources defined in the pattern, and (ii) the number and the types of the involved variables (i.e., the pattern parameters).

An easy way to compute $ran(p)$ is to average over the pertinence of the formal resources specified in the query pattern and use this value to rank the patterns; given a set $R_P(p)$ of formal resources assigned to p, the ranking based on average pertinence is computed as:

$$ran(p) = \sum_{r_i \in R_P(p)} \frac{pert(r_i,\mathcal{K})}{|R_P(p)|} \quad (3)$$

The formula above almost does the job, however, our experiments showed that this way of computing the ranking would penalise more general patterns referencing many resources (*broad patterns*) w.r.t. those patterns which reference few resources and are thus more precise (*narrow patterns*). For this reason we add to Equation 3 a normalisation factor as follows:

$$ran^{norm}(p) = \nu \cdot ran(p) \quad (4)$$

where the value of ν is defined as follows:

$$\nu = \frac{1 - |R_P(p)|}{|R_P(p)|} \cdot |R_P(p) \setminus R_k| + |R_P(p)| \quad (5)$$

ν takes into account the number of resources directly associated with any keyword k (denoted by R_k) and mentioned in the set of formal resources of the pattern. It leaves unaltered the ranking when R_k is empty while removes the dependency on the number of referenced resources when the pattern mentions only resources in R_k.

Another situation to be taken care of arises when it is not possible to determine the position of a query pattern in the ranking by relying on the referenced resources only because they get the same value for $ran(p)$. When such conditions occur we rely on the "unmatched" keywords. As already introduced in Section 3.1, when it is not possible to find a resource in the ontology that matches through its labels a keyword given as input, the system considers that string as a possible value for the variables in the pattern (i.e., a *hint*). Hints are matched against the variables defined in the query pattern considering both the number of variables and the associated type. The pattern with the higher number of matches *dominates* (i.e., is shown before) the other patterns.

Once the NL-Queries correspondent to the patterns have been selected, the user chooses which queries have to be executed against the available data represented by the current chunk.

3.3 Adaptive Ranking

In the approach presented so far, given a set of keywords, the ranking of query patterns is determined only by the pertinence of the ontology resources associated to the keywords. However, with a simple extension, it is possible to make the ranking adaptive w.r.t the user interpretation of the input keywords.

This problem is generally known in Information Retrieval as *relevance feedback* [27] and the aim is to involve the user in the retrieval process by allowing him to explicitly denote a document as *relevant* or *un-relevant* w.r.t. the issued query; this information is then used to iteratively improve the retrieval process.

In SAFE, an important user feedback is the selection of a query that is not at the top of the ranking, meaning that the ranking process did not capture precisely the user interpretation of the input keywords. The idea is thus to define an additive correction factor for the ranking which integrates the history of the user's feedbacks during ranking computation, allowing SAFE to learn from the previous user experience. This form of feedback is known as *implicit feedback* since the user is not giving an explicit judgement on the relevance of the retrieved pattern. In addition, no other form of feedback would be acceptable due to the peculiar time-constraints imposed by the emergency scenario where SAFE is adopted.

First, SAFE computes the value $\Delta = ran^{norm}(p_t) - ran^{norm}(p_s)$, representing the difference between the ranking values of the top-ranked pattern and of the selected pattern. We believe Δ can be assumed as a measure of the "gap" between the system's and the user's interpretation of the input keywords. The additive correction factor is then computed as follows for a given pattern p, a given set of keywords $\mathcal{K}$ and the current time instant t:

$$exp(p, \mathcal{K}, t+1) = \eta \cdot \Delta + exp(p, \mathcal{K}, t) \tag{6}$$

where $exp(p, \mathcal{K}, 0) = 0$ and $\eta \in (0, 1]$ is the *learning rate* which determines how fast the SAFE system adapts to the user feedbacks. Also the value of η is determined experimentally, since a high learning rate could make the system too much subjected to user mistakes during the selection of queries, while a low learning rate could lead to a system that adapts with difficulty to the user feedbacks. The resulting formula for the determination of the ranking of a query pattern is the following:

$$ran^{exp}(p) = ran^{norm}(p) + exp(p, \mathcal{K}, t) \tag{7}$$

The techniques presented so far have been inspired by classical techniques adopted in Information Retrieval such as *query reformulation* and *expansion* [5]; however, in SAFE the keywords are eventually connected to the entities in the ontology and, therefore, to the formal resources present in the query patterns. This direct link between keywords and patterns makes the SAFE ranking much more precise w.r.t. traditional IR since the keywords are used more likely as indexes for the query patterns than as queries such is the normality in IR. In addition, traditional relevance-feedback techniques such as the Rocchio's algorithm [27] are known to be ineffective when the keywords' answer-sets are inherently disjunctive (e.g., <`Patient`, `Drug`>). This problem originates from the fact that in Rocchio-style algorithms, similar objects are traditionally assumed to belong to the same *cluster* identified by a keyword. On the contrary, in our approach the correction factor takes into account the entire set of keywords given as input thus limiting this problem.

4. THE ROLE OF CONTEXT

Since most of the queries are answered over the chunks rather than through the federated information system, the correct design of the different T_c's (i.e., the chunk's TBoxes) is of extreme importance in order to avoid too frequent context-switching and, as a consequence, deficiencies in accessing the wanted information during an emergency. Moreover, since SAFE's main aim is the fast and precise retrieval of medical information in order to provide the needed decision support to health-care professionals during medical emergencies, the prior identification of the subset of the available data that is relevant to a given situation (or *context*) may provide a decisive means to reduce the search-space during the computation of the ranking for query patterns.

We first introduce the structure of the context model and then proceed to the description of how context-awareness affects the ranking of the query patterns.

4.1 The Context Model

It has been recognised [11] that knowledge has a contextual component, and that this component may be of use to extract and present the relevant chunks of knowledge, thus allowing for information filtering, focusing and reduction. Differently from other approaches to context-aware system design [9], we believe that the context in which information is managed is orthogonal with respect to what we might call "object information", and that as such it should be treated. Consequently, we represent contexts by means of a *context model* which is completely independent of the information space, and whose relationship with it is clearly stated.

In a common-sense interpretation the context is perceived as a set of variables whose values may be of interest for an agent (human or artificial) because they influence its actions. As an example, the following attribute-value pairs:

```
role=paramedic, situation=car-crash, topic=equipment
```

may be used to characterise the context of a paramedic who takes in a car-crash rescue operation and has to select the appropriate equipment. However, given an application, not all the combinations of variables and value assignments are necessarily meaningful, e.g., the following pairs:

```
role=paramedic, situation=surgery, topic=equipment
```

characterise the unlikely situation of a paramedic taking part in a surgical operation.

In general, the precise definition of the *valid* combinations is obtained through a *context model* [8]. Within SAFE, we adopt the context model known as *Context Dimension Tree (CDT)* [10], formalized as an ontology. This model allows the application designer to specify all the possible (meaningful) contexts related to the application situation, including the association between each context and the part of the domain ontology which is relevant for that context.

Figure 2 shows the graphical representation of a CDT, modelling the possible contexts of SAFE. In this example context is analysed with respect to the dimensions (drawn as black nodes) which are common to most applications: the `role`, representing the user's role (e.g., `General Practitioner`, `Lab technician`, `Hospital personnel`, etc.), the `situation` he/she may be in, the `query interface` and the `topic` of interest. A dimension value (drawn as a white node) can be further analysed with respect to different viewpoints, generating further (sub-)dimensions in the tree structure. A value can be further specified by means of parameters (circled nodes) such as an identifier or a date. A context is a subtree of the CDT, obtained by appropriately choosing a set of (sub-)dimension values. The CDT designer is in charge of establishing which dimensions are appropriate for the current application domain and of specifying the correspondence between each context and the portion of the domain ontology that is relevant to it (called context-aware chunk).

A context-aware chunk or, simply, a *chunk* is a pair $C = < T_c, A_c >$ where T_c is a subset of the terminological axioms of the domain ontology's TBox, while A_c is a subset of the domain ontology's ABox consistent with T_c (i.e., A_c is a model for T_c).

We represent the context model for SAFE in ontological terms introducing the following structures:

- the *context-vocabulary* defines the vocabulary (i.e., the meta-model) used to build the context models. This vocabulary is application-independent.
- the *context-model* is an instantiation of the context-vocabulary and defines the context model for the given application. In particular, the context-model specifies the (possibly hierarchical) context dimensions for the specific application, along with their possible values.
- the *contexts* are instantiations of the context-model and represents valid (i.e., consistent with the context model) contexts for a particular application.

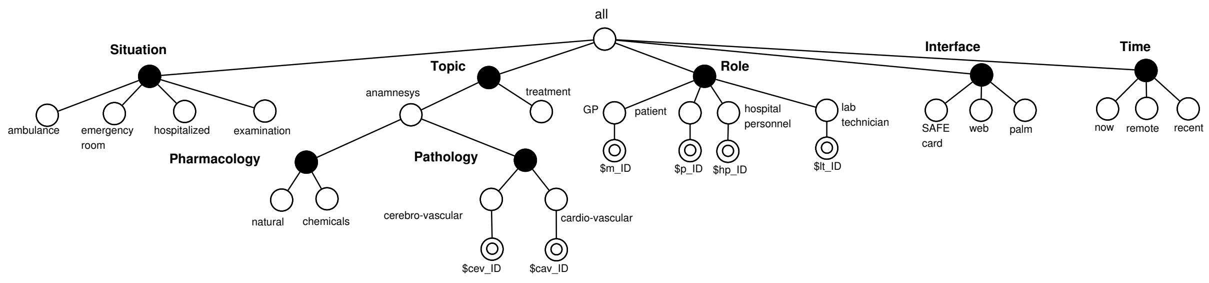


Figure 2: The Context Dimension Tree (CDT) for SAFE

- the *relevant-areas* represent the associations between a valid context c and the corresponding subset (T_c) of the domain ontology.

The context-vocabulary provides the building blocks for modelling concepts as: (i) *dimensions* and their values (e.g., `time=recent`, `role=GP`, `pathology=cardio-vascular`) and (ii) *parameters* and their values (e.g., `$m_ID=65403`, `$cardv_ID=heart-stroke`, etc.), in a word, to represent the structure and semantics of the context model. The full specification of the context-vocabulary is given in Table 2.

The concepts named as `Dimension` and `Parameter` represent the super-classes for all the possible dimensions and parameters of a context model, while the concepts named `ActualDimension` and `ActualParameter` model value assignments for dimensions and parameters. In a given model, the values will be instances of the concepts `Value` and `xsd:AnyType` (i.e., a generic XML Schema data-type) or their sub-classes; a `Context` is then defined as something for which there exists one or more dimension assignments.

The context-model ontology uses the resources of the context-vocabulary ontology to build an application-dependent context model. Consider, as an example, the fragment of the SAFE context model shown in Table 3. Statements of the form of (1) and (2) define the formal dimensions (e.g., `topic`) and the formal parameters (e.g., `cev_ID`), while statements of the form of (3) constrain the valid value assignments for each dimension (e.g., `situation` may assume the value `ambulance` or `emergency-room` but not `anamnesis`. All the values also inherit from the vocabulary concept `Value` (see (4)). Statements of the form of (5) and (6) define the valid assignments for dimensions and parameters, constructing also the hierarchical structure of the context model since each dimension assignments has a reference to his parent in the model; finally statement (7) defines the structure of a `SAFEContext` as a restriction of the `Context` defined in the context-vocabulary. In this example, a `Context` for SAFE is one that may assume (among the others) as actual dimension the `topic` and hence also one of its specialisations such as `pathology`. Besides the `topic` dimension and its values and sub-dimensions, the context-model ontology contains the other dimensions of the CDT of Figure 2 along with their values.

It is now clear that the context-model ontology supports the representation of a certain (finite) number of valid contexts which correspond to different consistent ABoxes (i.e., models for the TBox) of the context-model ontology. An example of context consistent with the SAFE context model and the use-case `c1` of Section 1 is shown in Table 4. Here we define the context for the use-case `c1`, where the paramedic in the ambulance is accessing the recent anamnesis of the patient through a mobile device. `c1` is constituted by five dimensions (`time`, `role`, `topic`, `interface` and `situation`). The last column of Table 4 assigns, to each dimension, the value it takes in `c1`.

The relevant areas define the domain ontology resources that are "relevant" for a given context (i.e., the TBox T_c of a chunk). Basically, each context `c` is assigned to the fragment T_c of the ontology's TBox that has the property of containing all the resources needed to answer the user's queries when `c` is active. This assignment is manually-defined by an expert at design-time and stored in the relevant areas ontology. The association of a context to the corresponding T_c is constructed by means of the property `inContext` of the context-vocabulary ontology.

4.2 Context-Aware Pattern Ranking

Relevant areas can be exploited to improve the ranking of query patterns. At any moment, during the execution of the SAFE system, there exists a single *active context*. As already said, each context defines a precise set of resources (concepts and properties in the corresponding chunk's TBox) taken from the domain ontology that are somewhat related to the queries that a user may want to execute while operating in a given context; let us call this set of resources $Rel(c)$ where c is the considered context. The ranking of a query pattern p is thus adapted as follows:

$$ran^{ctx}(p) = \nu^{ctx} \cdot ran(p) \cdot \mid Rel(c) \cap R_P(p) \mid \qquad (8)$$

where the correction factor ν^{ctx} adapts the multiplicative factor ν used in Equation 4 in order to take into account the relevant-area. ν^{ctx} is computed as follows:

$$\nu^{ctx} = \frac{1 - \dfrac{Rel(c) \cap R_P(p)}{R_P(p)}}{\dfrac{Rel(c) \cap R_P(p)}{R_P(p)}} \cdot \mid R_P(p) \setminus R_k \mid + \frac{Rel(c) \cap R_P(p)}{R_P(p)} \qquad (9)$$

Roughly speaking, the value of the ranking depends now on the fraction of resources that are, at the same time, referenced by the pattern and mentioned in the chunk associated with the active contexts. Whenever a pattern does not reference any resource inside the relevant-area, the computed ranking is equal to zero and the pattern will be shown at the bottom of the ranking. On the contrary, when the pattern references only resources in the relevant-area, the ranking is equal to the sum of the values of pertinence of the referenced resources plus one. As it can be seen, the new ranking prefers patterns whose referenced resources are in the w.r.t patterns whose referenced resources are sources of pertinence.

4.3 Chunk Computation

Table 2: The context-vocabulary ontology

ActualDimension	$\sqsubseteq$	$\exists_{=1}$formalDimension.Dimension $\sqcap$ $\exists_{=1}$dimensionValue.Value $\sqcap$ $\exists_{=1}$hasDimension^{-}.ActualDimension
ActualParameter	$\sqsubseteq$	$\exists_{=1}$formalParameter.Parameter $\sqcap$ $\exists_{=1}$parameterValue.*AnyType* $\sqcap$ $\exists_{=1}$hasParameter^{-}.Value
Context	$\sqsubseteq$	$\exists_{=1}$hasDimension.ActualDimension

Table 3: A fragment of the SAFE context model

(1a) Dimension(situation)	(3a) Situation(ambulance)	(4a) Situation $\sqsubseteq$ Value
(1b) Dimension(topic)	(3b) Situation(emergency-room)	(4b) Topic $\sqsubseteq$ Value
(1c) Dimension(pathology)	(3c) Topic(anamnesis)	(4c) Pathology $\sqsubseteq$ Value
	(3d) Topic(treatment)	
(2a) FormalParameter(cev_ID)	(3e) Pathology(cardioVascular)	
(2b) FormalParameter(cav_ID)	(3f) Pathology(cerebroVascular)	
(5a) ActualSituation $\sqsubseteq$	ActualDimension $\sqcap$ $\exists$dimensionValue.Situation $\sqcap$ $\exists$formalDimension.{situation} $\sqcap$ $\exists$hasDimension^{-}.(ActualDimension $\sqcap$ $\exists$dimensionValue.{all})	
(5b) ActualTopic $\sqsubseteq$	ActualDimension $\sqcap$ $\exists$dimensionValue.Topic $\sqcap$ $\exists$formalDimension.{topic} $\sqcap$ $\exists$hasDimension^{-}.(ActualDimension $\sqcap$ $\exists$dimensionValue.{all})	
(5c) ActualPathology $\sqsubseteq$	ActualDimension $\sqcap$ $\exists$dimensionValue.Pathology $\sqcap$ $\exists$formalDimension.{pathology} $\sqcap$ $\exists$hasDimension^{-}.(ActualTopic $\sqcap$ $\exists$dimensionValue.{anamnesys})	
(6a) ActualParCerebroVascular $\sqsubseteq$	ActualParameter $\sqcap$ $\exists$parameterValue.XSDString $\sqcap$ $\exists$formalParameter.{cev_ID} $\sqcap$ $\exists$hasParameter^{-}.{cerebroVascular}	
(6b) ActualParCardioVascular $\sqsubseteq$	ActualParameter $\sqcap$ $\exists$parameterValue.XSDString $\sqcap$ $\exists$formalParameter.{cav_ID} $\sqcap$ $\exists$hasParameter^{-}.{cardioVascular}	
(7) SAFEContext $\sqsubseteq$ Context $\sqcap \ldots \sqcap$ $\exists$hasDimension.ActualSituation $\sqcap$ $\exists$hasDimension.ActualTopic $\sqcap \ldots$		

Table 4: SAFE Context Configuration for `c1`

SAFEContext(c1)		
hasDimension(c1, c1_time)	ActualTime(c1_time)	timeValue(c1_time, recent)
hasDimension(c1, c1_role)	ActualRole(c1_role)	roleValue(c1_role, paramedic)
hasDimension(c1, c1_pathology)	ActualPathology(c1_pathology)	dimensionValue(c1_pathology, cardioVascular)
hasDimension(c1, c1_situation)	ActualSituation(c1_situation)	dimensionValue(c1_situation, ambulance)
hasDimension(c1, c1_interface)	ActualInterface(c1_interface)	dimensionValue(c1_interface, palm)

The context plays a predominant role also in the definition of the chunk TBoxes. As discussed in Section 1, SAFE queries are mostly answered over the chunks rather than through the SPARQL Endpoint; each chunk is constructed using one or more queries over the domain ontology that retrieve the chunk's ABox A_c.

These queries are intentionally *wide*, in the sense that their are not meant to answer to a query but, rather, to retrieve all the instances from the SPARQL Endpoint that constitute a model for the resources defined in T_c. If the T_c's have been defined correctly, at any moment the data needed to answer the queries will be contained in some chunk associated with the active contexts. Whenever the active context changes, the mobile device signals a context-switch to the system; the new chunks will be downloaded by querying the system and stored on the mobile device for subsequent querying.

It is important to note that, whenever the chunk is computed, the entire domain ontology and all the instances in the SAFE-DB are used; the answers are then complete in the sense of logical completeness. As a consequence, while the chunk's TBox is a projection of the resources in the domain ontology, all the facts (included those implicit in the knowledge base) have been made explicit in the chunk. As an example, consider a small domain ontology with only one class hierarchy constituted by a concept `Person` and two sub-classes `Physician` and `Paramedic`. Let T_c be the relevant area for some context `c` containing just the hierarchy between the classes `Person` and `Physician`. The A_c corresponding to T_c will not represent the class `Paramedics` but will contain all the instances of persons (paramedics included) because A_c has been computed using the domain ontology and not just T_c.

5. EXPERIMENTATION

In this section, the SAFE approach is evaluated by discussing the results of an empirical experiment based on a first implementation of the system. The section is divided into three parts: the first one describes the testbeds, the second one presents the method used for the experiment and its setting, and the third part reports on the resulting times and their break down.

5.1 Testbeds

SAFE has been implemented both for the Web, by resorting to recent technologies for enterprise applications, and for Maemo Linux[5] mobile devices such as the Nokia Smartphones N810 and N900. Due to the absence in literature of similar implementations, we decided to compare the SAFE approach with a Web-based re-implementation of a traditional form-based application currently used by the cardiology unit of a public hospital that collaborated in the development and the experimentation of the system. For this reason, only the Web version (which is the SAFE worst case) of the semantics-based GUI was adopted for the experimentation in a scenario similar to the one described in the `c2` use-case of Section 1.

The overall SAFE implementation is composed of the following components (the first three ones are from third-parties):

- A federated relational database, SAFE DB, containing (among the others) data about patients, health-care personnel, health-care structures, diseases, procedures etc.
- An instance of the D2R-Server [7] along with its SPARQL endpoint exposing a domain ontology which describes the

[5]http://maemo.org/

resources needed in SAFE. The D2R-Server acts as an interface between the SAFE DB and the SAFE user applications, virtualising the DB as an RDF graph.

- The Pellet OWL 2 reasoner.
- The SAFESearchLib, implementing the algorithm for semantic search and for managing query patterns and contexts. It exploits a SPARQL endpoint to submit queries to the D2R server.
- The SAFE-Mobile, implementing the keyword-based graphical query interface for mobile devices and the related application logic.
- The SAFE-Web/Semantics, implementing the keyword-based interface for Web-based desktops, and a scalable application logic hosted by an application server (JBoss[6]). The client-side implementation is based on the SEAM[7] and OpenLaszlo[8] frameworks, to offer a GUI for search (form- and semantics-based, respecively). The connection between OpenLaszlo views and the application logic is based on the framework EU4RIA[9].
- The SAFE-Web/Forms, implementing a pre-existing application of the hospital. It is characterised by a number of forms that collect and group cohesive data extracted from a remote DB. Users can navigate among these forms by selecting tabs.
- The SAFE Context Server, implementing the context-switch services needed to generate a suitable chunk when requested by the mobile application in the case of changes of the operative contexts. The context specifications are sent to the SAFE Context Server, which accesses a registry containing the associations between the context and a suitable SPARQL query able to retrieve the corresponding chunk from the domain ontology, stored in the mobile device.

5.2 Experimental setting

The objective of the experiment was the evaluation of the time for accessing to specific medical information of a patient in a realistic scenario, starting from off-line queries expressed in natural language that emulate the questions that the medical personnel would ask the patient during an emergency intervention (see Table 5).

The experiment involved ten people, with a sufficient knowledge of the domain but without a previous knowledge of the two systems they would use. The storage of query patterns was initially populated with one hundred queries whereas the database contained information related to about one thousand patients. The domain ontology adopted for the experiment is composed of 35 classes, 55 object properties, 77 data properties.

Since the aim of the paper is not improving query-processing performance but increasing the ability of the system to quickly suggest the best queries to be submitted to the system in order to obtain the desired data, we adopted a minimal deployment on a client/server system. As for any other client/server application, scalable servers will be of paramount importance when many users will use the system at the same time. Note that the appropriate scalability level can be easily achieved by replicating the application logic for each user and exploiting a scalable hardware with server replicas.

[6]http://www.jboss.org/
[7]http://seamframework.org/
[8]http://www.openlaszlo.org/
[9]http://eu4ria.googlecode.com

Table 5: Natural Language Queries

ID	Query
1.	Find phone contacts of the patient
2.	Verify whether the patient has risk factors tied to the family anamnesis
3.	Verify whether the patient uses drugs
4.	Verify whether the patient registered high stress situations
5.	Verify whether the the patient is overweight
6.	Verify whether the patient is a smoker
7.	Show possible heart problems in infancy age
8.	Show possible heart problems related to the last years
9.	Show possible recent heart problems
10.	Find anamnesis tied to the habits of the patient
11.	Show correlations between patient's pathologies and the family anamnesis
12.	Verify whether the patient is under pharmacological therapy
13.	Find addresses to use for contacting the patient's relatives
14.	Verify whether the patient frequently uses alcohol
15.	Show all the diseases occurred to the patient during his/her infancy
16.	Show the diseases occurred in the last months
17.	Show all the diseases occurred to the patient
18.	Show possible diseases occurred in the last months
19.	Find the contacts of all the doctors who follow the patient

However, to understand the effects of the different algorithms implemented, besides measuring the overall time (*aT*) for accessing data, we inserted some probes in the code in order to break down this time in its main components:

- *Thinking time* (*thT*) is the time users spend deciding the actions they have to perform onto the application GUI to obtain the desired results.
- *Keyword-to-pertinence time* (*kpT*) is the time from the typing of the first character of a keyword to the assignment of scores to the concepts that are pertinent with the keyword itself.
- *Score and rank time* (*srT*) is the time spent to assign a score to each query pattern stored in the system and to arrange the order of the queries from the highest score to the lowest one.
- *Query execution time* (*qeT*) is the time from the submission of a SPARQL query to the retrieval of the data from the DB.
- *Communication time* (*coT*) is the time spent for the communication between the client and the server.

To reduce the overhead due to reasoning, semantic indexing is applied every time the ontology changes. The indexing is used to pre-compute the pertinence coefficients of each concept of the ontology.

5.3 Evaluation

Each person received five queries, randomly extracted from the pool of queries shown in Table 5 and started to search for the desired information by using the first form-based application and then the semantics-based one. A timer was started before reading each query and stopped at the end when the desired information was found.

During the experiment, the people behaved in different ways: (1) with the form-based application, they navigated among tabs to reach the right view; (2) in the other case, they inserted one or more keywords to obtain the desired queries in natural language and, after the selection of one or more of these queries, the desired information. Whenever additional information was needed, more than one pattern and the related SPARQL queries were executed in parallel.

In 65% of cases, the selected queries were found on top of the ranked list; in 25% of cases, users found the query in the second position whereas in the remaining 10%, the desired query was shown after the second position but always in the list of query patterns visualised without any scrolling. As concerning adaptation, it is worth noting that the convergence towards the top score depends on the learning coefficient (η): a query is moved to the top of the list in at least 1/η iterations.

In several cases people were not able to find the searched information with the form-based application, whereas with the semantics-based approach data were found in a precise and more focused fashion, by showing only the results tied to the selected query patterns.

The main quantitative results of the experiment are shown in Table 6 that also reports on the access time breakdown for semantic search. The presence of zeros in columns kpT and srT for some rows means that in those cases users found the desired queries without typing any keyword. However, also in other cases both kbT and srT are always very short. More time (about 1 sec) is spent for querying data through a SPARQL query while the main contribution is due to thT.

Table 6: SAFE **Human-Computer Interaction Evaluation**

Query ID	Form aT (s)	Semantics aT (s)	thT (s)	kpT (ms)	srT (ms)	qeT (ms)	coT (ms)
Candidate 1							
7	72	30	28.7	0	0	1231	60
10	57	11	10.1	0	0	803	78
2	16	4	2.6	1.6	25.9	1258	69
5	27	24	22.9	0	0	962	87
17	32	58	56.8	1.6	66	1093	69
Candidate 2							
1	10	5	3.3	1.6	59	1535	66
3	122	11	10.2	0	0	713	75
13	17	6	5.2	2.6	34.5	695	69
16	41	24	2.3	1.3	59	1039	75
18	44	16	1.5	1.3	58	581	84
Candidate 3							
3	111	21	20.2	0	0	667	81
7	40	21	19.8	0	0	1153	75
11	26	12	11.2	1.7	79	658	72
14	8	10	9.2	0	0	662	114
15	31	18	16.7	1.8	91	1146	72
Candidate 4							
4	110	11	9.7	0	0	1199	84
6	20	12	11.1	0	0	704	96
12	37	10	9.17	1.9	40.9	715	75
17	25	38	36.8	1.9	110	977	108
19	71	7	5.9	3.9	49.9	982	84
Candidate 5							
1	27	9	8.1	1.6	21.8	765	72
7	33	22	21.1	1.3	118	704	87
8	21	35	34.3	0	0	620	84
10	67	25	23.8	0	0	1139	81
14	23	15	14.3	0	0	603	75
Candidate 6							
3	164	24	23	0	0	859	87
6	54	13	11.9	1.5	19.5	1018	78
9	22	34	32.8	0	0	1044	123
12	55	7	5.9	1.4	25.1	959	78
13	25	8	6.8	1.3	19.5	1117	87
Candidate 7							
3	117	20	19.2	0	0	683	84
7	37	18	17.1	1.2	105.5	668	78
11	19	35	34.1	0.9	87.1	685	108
16	47	22	20.7	1.1	113	1087	81
9	20	16	14.9	0.6	17.6	988	78
Candidate 8							
7	62	26	24.7	0.6	76.3	1099	84
14	46	35	33.9	0	0	928	87
11	55	18	16.8	0.6	31.6	1082	81
18	42	7	5.9	0.5	30.1	992	93
1	23	7	5.9	0.6	18.5	982	81
Candidate 9							
19	36	18	16.8	0.5	103	1030	78
12	32	9	7.7	0.6	78.7	1110	87
6	36	27	25.8	0	0	1059	81
4	11	9	7.8	0.6	31.2	1077	93
1	22	5	4.2	0.6	18.9	606	105
Candidate 10							
3	88	5	4.3	0	0	633	81
17	48	18	17.1	0.7	18.6	756	78
9	41	8	6.9	0	0	1027	93
11	27	69	68.2	0.5	27.6	661	81
13	21	17	16.3	0.6	20.6	606	78

For the form-based search, the overall time could be broken in thinking and navigation (over tabs), query execution and communication times. Since query execution time is very small for simple queries on medium-size databases, and the communication time is almost the same as for semantic search, the most relevant contribution is the thinking and navigation time (which almost coincides with the overall access time shown in Table 6 - II column).

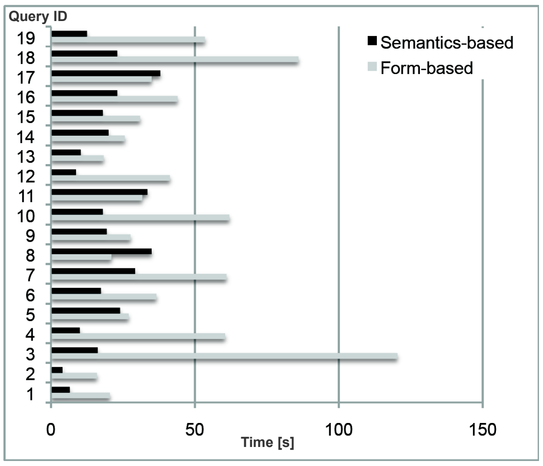


Figure 3: Form-based vs. Semantics-based searches

Figure 3 shows the average time for each natural language query. In most cases, semantics enabled people to reach the desired information more rapidly.

Additional experiments were conducted with larger sets of query patterns to assess the system scalability. The measures obtained with 500 and 1000 queries showed a growth approximately linear of the score and ranking time. This proves a good scalability of the system and its applicability, since the number of query patterns used is sufficiently large to satisfy most of the real scenarios.

6. CONCLUSIONS AND FUTURE WORK

The paper presented an approach for accessing – in a fast, precise and exhaustive fashion – data from personal or centralised information systems in pervasive environments. The approach has been integrated in an innovative system, SAFE, developed for supporting health-care personnel during emergency situations. The system exploits domain ontologies to access data with semantic inference. The ontologies are tailored, on the basis of the application working context, through a context model which is also used to identify the *chunk* of knowledge (ontology and data) that is stored into the personal smart-cards of the patients.

The results obtained by our experiments show that SAFE significantly improves the time to access the desired data in emergency situations for many kinds of queries. It is worth noting that the experiment was conducted in the worst case for the semantic search approach, due to constraints related to the legacy application used by the hospital.

A more beneficial environment is the mobile one: in this case, in fact, the approach based on keywords and natural language patterns is much more effective if compared with the navigation based on tabs, scrolling or other techniques, which are very difficult to use in the small working spaces offered by personal devices.

SAFE was very appreciated by the specialised medical personnel who was involved in the definition and experimentation of the system. Their feedbacks helped us to improve the system and suggested further extensions for future work: (i) highlighting the terms of interest in the results; (ii) adopting user profiles since they can be useful to store the history of pattern selections to incrementally improve the ranking of the SPARQL queries; (iii) deriving SPARQL queries automatically from queries in natural language.

7. REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] S. Agrawal, S. Chaudhuri, and G. Das. DBXplorer: A system for keyword-based search over relational databases. In *Proc. of the 18th Intl Conf. on Data Engineering*, pages 5–16, 2002.

[3] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of the 19th Intl Joint Conf. on Artificial Intelligence*, pages 364–369, 2005.

[4] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider. *The description logic handbook: theory, implementation and applications*. Cambridge University Press, Cambridge, United Kingdom, 2003.

[5] R. Baeza-Yates and B. Ribeiro-Neto. *Modern information retrieval*. Addison-Wesley, 1999.

[6] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using BANKS. In *Proc. of the 18th Intl Conf. on Data Engineering*, pages 431–440, 2002.

[7] C. Bizer and R. Cyganiak. D2R Server: Publishing relational databases on the semantic web. In *Proc. of the 5th Intl Semantic Web Conf. (Poster)*, 2006.

[8] C. Bolchini, C. Curino, G. Orsi, E. Quintarelli, R. Rossato, F. A. Schreiber, and L. Tanca. And what can context do for data? *Comm. of the ACM*, 52(11):136–140, 2009.

[9] C. Bolchini, C. Curino, E. Quintarelli, F. Schreiber, and L. Tanca. A data-oriented survey of context models. *SIGMOD Record*, 36(4):19–26, 2007.

[10] C. Bolchini, C. Curino, E. Quintarelli, F. A. Schreiber, and L. Tanca. Context information for knowledge reshaping. *Journal of Web Engineering and Technology*, 5(1):88–103, 2009.

[11] P. Brézillon and S. Abu-Hakima. Using knowledge in its context. *AI Magazine*, 16(1):87–91, 1995.

[12] M. Chalmers. A historical view of context. *Computer Supported Cooperative Work*, 13(3):223–247, 2004.

[13] S. Chaudhuri and G. Das. Keyword querying and ranking in databases. In *Proc. of the 35th Intl Conf. on Very Large Data Bases*, pages 1658–1659, 2009.

[14] S. Chaudhuri and R. Kaushik. Extending autocompletion to tolerate errors. In *Proc. of the 35th Intl Conf. on Management of Data*, pages 707–718, 2009.

[15] A. Dey. Understanding and using context. *Personal Ubiquitous Computing*, 5(1):4–7, 2001.

[16] T. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–200, 1993.

[17] V. Hristidis and Y. Papakonstantinou. DISCOVER: Keyword search in relational databases. In *Proc. of the 28th Intl Conf. on Very Large Data Bases*, pages 670–681, 2002.

[18] H. Hwang, V. Hristidis, and Y. Papakonstantinou. ObjectRank: A system for authority-based search on databases. In *Proc. of the 26th Intl Conf. on Management of Data*, pages 796–798, 2006.

[19] A. Kementsietsidis, L. Lim, and M. Wang. Supporting ontology-based keyword search over medical databases. In *Proc. of the American Medical Informatics Association Symp.*, pages 409–413, 2008.

[20] Y. Lei, V. Uren, and E. Motta. SemSearch: A search engine for the semantic web. In *Proc. of the 15th Intl Conf. on Knowledge Engineering and Knowledge Management*, pages 238–245, 2006.

[21] Y. Li, H. Yang, and H. Jagadish. Nalix: an interactive natural language interface for querying xml. In *Proc. of the 31st Intl Conf. on Management of Data*, pages 900–902, 2005.

[22] D. Martins, L. Santana, M. Biajiz, A. do Prado, and W. de Souza. Context-aware information retrieval on a ubiquitous medical learning environment. In *Proc. of the Symp. on Applied computing*, pages 2348–2349, 2008.

[23] J. Pérez, M. Arenas, and C. Gutierrez. Semantics and complexity of SPARQL. *ACM Trans. Database Systems*, 34(3):1–45, 2009.

[24] F. Perich, A. Joshi, T. Finin, and Y. Yesha. On data management in pervasive computing environments. *IEEE Trans. on Knowledge and Data Engineering*, 16(5):621–634, 2004.

[25] J. Pound, I. Ilyas, and G. Weddell. Expressive and flexible access to web-extracted data: a keyword-based structured query language. In *Proc. of the 26th Intl Conf. on Management of Data*, pages 423–434, 2010.

[26] J. A. Royo, E. Mena, J. Bernad, and A. Illarramendi. Searching the web: From keywords to semantic queries. In *Proc. of the 3rd Intl Conf. on Information Technology and Applications*, pages 244–249, 2005.

[27] G. Salton. *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice-Hall, Inc., 1971.

[28] J. Silva and J. Favela. Context aware retrieval of health information on the web. In *Proc. of the 4th Latin American Web Cong.*, pages 135–146, 2006.

[29] B. Smith, W. Ceusters, B. Klagges, J. Köhler, A. Kumar, J. Lomax, C. Mungall, F. Neuhaus, A. Rector, and C. Rosse. Relations in biomedical ontologies. *Genome Biology*, 6(5):R46, 2005.

[30] K. Spackman, K. Campbell, and R. Côtè. Snomed rt: a reference terminology for health care. In *Proc. of the AMIA Annual Fall Symp.*, pages 640–645, 1997.

[31] S. Tata and G. M. Lohman. SQAK: doing more with keywords. In *Proc. of the 28th Intl Conf. on Management of Data*, pages 889–902, 2008.

[32] T. Tran, P. Cimiano, S. Rudolph, and R. Studer. Ontology-based interpretation of keywords for semantic search. In *Proc. of the 6th Intl Semantic Web Conf.*, pages 523–536, 2007.

[33] T. Tran, H. Wang, S. Rudolph, and P. Cimiano. Top-k exploration of query candidates for efficient keyword search on graph-shaped (rdf) data. In *Proc. of the 25th Intl Conf. on Data Engineering*, pages 405–416, 2009.

[34] H. Turtle. Natural language vs. boolean query evaluation: a comparison of retrieval performance. In *Proc. of the 17th Intl Conf. on Research and Development in Information Retrieval*, pages 212–220, 1994.

[35] X. Wang, D. Zhang, T. Gu, and H. Pung. Ontology based context modeling and reasoning using OWL. In *Proc. of 1st Intl Workshop on Context Modelling and Reasoning*, pages 18–22, 2004.

[36] W. Winkler. The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Census Bureau, 1999.

[37] J. Zhang, Z. Peng, S. Wang, and H. Nie. Si-SEEKER: Ontology-based semantic search over databases. In *Proc. of 1st Intl Conf. on Knowledge Science, Engineering and Management*, pages 599–611, 2006.