Mobile Cloud Support for Semantic-enriched Speech Recognition in Social Care

Antonio Corradi, *Member, IEEE,* Marco Destro, Luca Foschini, *Member, IEEE,*
Spyros Kotoulas, *Member, IEEE,* Vanessa Lopez, *Member, IEEE,* Rebecca Montanari *Member, IEEE*

Abstract—Nowadays, most users carry high computing power mobile devices where speech recognition is certainly one of the main technologies available in every modern smartphone, although battery draining and application performance (resource shortage) have a big impact on the experienced quality. Shifting applications and services to the cloud may help to improve mobile user satisfaction as demonstrated by several ongoing efforts in the mobile cloud area. However, the quality of speech recognition is still not sufficient in many complex cases to replace the common hand written text, especially when prompt reaction to short-term provisioning requests is required. To address the new scenario, this paper proposes a mobile cloud infrastructure to support the extraction of semantics information from speech recognition in the Social Care domain, where carers have to speak about their patients conditions in order to have reliable notes used afterward to plan the best support. We present not only an architecture proposal, but also a real prototype that we have deployed and thoroughly assessed with different queries, accents, and in presence of load peaks, in our experimental mobile cloud Platform as a Service (PaaS) testbed based on Cloud Foundry.

Index Terms—Mobile Cloud, Semantic Web, Speech Recognition, Natural Language Processing.

1 Introduction

Cloud computing architectures have gained more and more momentum in recent years and most vendors are looking at them to provide feasible solutions for optimal exploitation of their own infrastructures. At the same time, the ever-increasing wireless connection bandwidths, hardware memory, and processing capabilities have boosted the spreading of new mobile devices, such as smartphones, tablets, and netbooks. Notwithstanding their great potential, mobile devices exhibit still strict constraints on local resources and mobile support design typically focuses on saving the most precious energy resource; for this reason, applications that require high amounts of processing power and resources cannot be easily ported to and deployed atop of them.

Mobile cloud, by bringing together mobile and cloud computing areas, can provide a solution to the above issue: it allows to move resource-demanding tasks that require high energy computation from the mobile device to the cloud computing infrastructure, thus enabling the availability of complex applications within edge-devices with limited resources [1], [2]. For instance, Content Based Image Retrieval (CBIR) requires a lot of computing and resources to compute a set of metrics from an image, and then to match them against the metrics of the images stored in a database [3]. By moving the computing and the matching processes to the Cloud, resources of user devices are saved, making CBIR feasible, even in resource-limited devices. Similarly, mobile device antivirus protection can be enabled by shifting the heavy-load file signature matching process into the Cloud [4].

Along that direction, mobile speech recognition represents a very good example of an application area that can greatly benefit of the new Mobile Cloud Computing paradigm. Currently, to reduce resource consumption on mobile devices, mobile speech recognition solutions only provide syntactic-based speech recognition which is useful only in specific cases, such as when the aim is finding out most commonly used keywords or injecting routinely commands to the user's mobile device. No support for text meaning checking is, however, provided, thus hampering the exploitation of mobile speech recognition in a widespread set of application domains, especially the ones that require reliable and complex text analysis and searching, such as social- and health- care ones.

Mobile cloud computing can leverage mobile speech recognition adoption in several application scenarios by allowing designers of mobile speech recognition systems to envision not only syntactic but also semantic-based speech recognition. Semantic Web extends the content of normal data with additional structured information (metadata) that provide semantics support so to allow machines to interpret and deal about data meaning. The Mobile Cloud paradigm offers the advantage of providing a backend for the execution of complex semantic-based text recognition algorithms. There are several factors of complexity stemming from semantic-based text representation. The process of data description has become more challenging with the huge increase and the steady growth of unstructured data from a data engineering point of view: a particular tough case is the management applied to data coming from speech recognition. In fact, if the task consists in giving a limited number of commands to the machine, existing semantics-agnostic systems perform pretty well, but if there is a necessity to create and save data referred to a particular domain or context, checking previously its meaning, the efficiency is usually not enough and users often still prefer to write them manually. Moreover, the Speech Recognition provided to users in these systems usually lists a set of ordered by confidence possible matches, but once the text is returned and it does not represent one of the previous described

keywords, the device is not capable of understanding its meaning at all. For this reason, a tool able to analyze the different proposed matches, understand their meaning, choose the most appropriate one, and finally extract the semantics of what the user really wanted to write down could be very useful.

This paper tackles the above issues by proposing Mobile cloud Support for Semantic-enriched speech recognition in Social Care (MoSSCa), a novel infrastructure that addresses above semantic-enriched speech recognition and scalability issues by melding together the advantages of Semantic Web and mobile cloud computing at the Platform as a Service (PaaS) level. Our proposal has several core original aspects. First, to the best of our knowledge, it is one of the first proposals enabling mobile care workers to exploit semantic-speech recognition during their care delivery activities. As we will describe in the following Section, this domain combines deep technical challenges and very high societal relevance. In addition, the particular application domain has been selected based on first-hand industrial experience. Second, the MoSSCa support allows fine-grained, efficient, and fully-distributed cloud infrastructure monitoring and management of all involved backed components by promptly and proactively triggering effective elastic scaling actions, thus guaranteeing prompt replies even during heavy and peak load conditions. Third, the proposed solution is based on open speech recognition components and the cloud support has been realized and integrated within the new open-source Cloud Foundry PaaS and have been tested in a real cloud deployment based on the open-source OpenStack Cloud IaaS platform; as relevant contribution, the proposed PaaS management platform for dynamic Cloud Foundry load balancing is available for the cloud community as an open-source prototype that can be easily integrated in other different cloud PaaS solutions. Finally, we show the results of an extensive MoSSCa use campaign in which we involved a mix of native and non-native English speakers, and challenged MoSSCa with several different queries for real patients histories so to assess its effectiveness in realistic settings; we also assessed system performances in terms of resource local and distributed resource usage and overall PaaS-enabled mobile cloud support scalability. Obtained experimental results show quantitatively and qualitatively how the exploitation of the PaaS-based MoSSCa infrastructure improves performance for the Social Care area.

The rest of the paper is structured as follows. Section 2 contextualizes our work and motivates its main goals. Section 3 introduces needed background material about the tools and platforms employed for the realization of the MoSSCa support. Section 4 presents MoSSCa architecture and its main component and Section 5 presents main integration and implementation insights. Section 6 reports a thorough evaluation of functional and system-/resource-consumption aspects of the proposed approach. Section 7 reports related works and Section 8 we provide conclusions and future work directions.

2 Mobile Cloud for Social Care

There are several mobile application fields that can benefit from the exploitation of the mobile cloud computing paradigm for their design, development and deployment. Mobile commerce, mobile gaming, social care and mobile healthcare are significant examples along with all other applications where mobile users require searching services (e.g., searching information, location, images, voices, or video clips). This section focuses on the social care application domain by motivating the need for a mobile cloud enabled support system in order to offer advanced services onboard social caregiver mobile devices, such as semantic-enriched speech recognition. Then, this section reports a focused selection of related social care research efforts in the mobile cloud area.

Care spans domains with tremendous *economic impact*: averaged across the members of the Organization for Economic Cooperation and Development (OECD), Social Care and Healthcare account to some 21% and 9.3% of the GDP respectively[1]. Public Safety, Justice and Education, domains that also have a large economic footprint, directly influence and are influenced by Care. The information relevant to Care is deeply *complex*: for healthcare, Nuance reports that LinkBase®[2] contains more than 1 million concepts. Social care depends on information from a very broad domain, from *numerous* relevant organizations (social service administration, educational institutions, homeless shelters and public safety authorities, etc) and differs across administrative boundaries.

In this paper, we are focusing on *care delivery* in the sense of the delivery of services to teams of care workers. Care workers have multiple and very heterogeneous specialisations: they can be nurses, medical assistants, care assistants, social workers or medical doctors. A key business factor driving this research is that a *large proportion of the care workers' time is spent performing tasks outside their unique skillset*. For instance, in a New York hospital, a survey has shown that 9.2 minutes out of a 15-minute doctor's visit were spent on social needs, crowding out clinical care [5]. Cowden et al. [6] report that up to 40% of social worker's time is spent on administrative tasks, depriving them of valuable time with their customers. At the same time, a single social worker may be responsible for thousands of people [5], making the need for timely and efficient information sharing as pressing as ever.

The societal and economic relevance of supporting care workers on the field is not easy. A series of research and technical challenges need to be addressed before a practical system can be deployed in a real-world setting. First, the amount of relevant information for a care worker is large. In order to support care workers on the field, a system would need to access large amounts of data, including medical ontologies, in order to understand medical terms, ontologies for common sense knowledge to understand commands and context, ontologies about social care, to understand various aspects pertaining to the social, psychological and behavioural condition of the person, among many others. In addition, processing this information entails computational challenges. As an example, machine-interpretable voice capability includes speech-to-text processing, result-

1. OECD Factbook 2012, figures for 2011
2. http://www.nuance.com/for-healthcare/resources/clinical-language-understanding/ontology/index.htm

ing in multiple text interpretations of the input. For each text interpretation, we can have multiple meanings of the lexical and grammatical structure. In turn, for the different lexical and grammatical productions, we can have multiple semantic interpretations, which need to be ranked to get the most relevant ones. Obviously, this explosion of the number of interpretations results in a computational explosion that is difficult to handle on a mobile phone. At the same time, the mobile phone may store valuable contextual information that may not be possible to transmit to a server due to legislative restrictions. Indeed, for practical and legal reasons, care workers would need to have immediate access to the selected interpretation, so as to verify its correctness.

2.1 Semantic-Enriched Speech Recognition Use Cases

As motivation for our work (and driver for the evaluation section), we consider the scenario where a care worker visits a vulnerable individual. We outline two motivating use-cases: intelligent note-taking and assisted assessments.

Care workers largely rely on case notes. Depending on the expertise of the care worker, case notes may be generic observations such as "Michael has a problematic relationship with Mary" or more domain specific, such as "Michael is complaining about pains in the lower abdomen". Typically, these notes are written using pen and paper, and, in some cases, an information system such as a note-taking program.

Although these notes can give very important insight to care workers, usually they are not sharable, because that would require copying them to a central file. Understanding and (semantically) indexing these notes would improve collaboration and allow care workers to get a complete picture of an individual since it would allow queries such as "give me all information regarding the psychological function of this person".

A key tool for care workers is standardised *assessments* of the strengths, skills, needs and weaknesses of vulnerable individuals. Assessments are essentially questionnaires and corresponding guidelines pertaining to various factors of the individuals life, including biological, financial, behavioural, psychological and social factors. Assessments can take anywhere from 10 minutes to multiple days.

Typically, assessments are conducted using pen and paper. Given recent advances in speech-to-text technology, mobile and Cloud computing, there is a significant opportunity for improvement. A system to be used to automate this process would save care workers much needed time, by automating input while they conduct the assessment. In addition, it would enable self-assessment through the use of standard phones.

As outlined in the previous section, this process is expensive, and necessitates assistance from a Cloud infrastructure in terms both of computation and data storage.

2.2 Mobile Cloud Solutions for Social/m-Health Care

Mobile cloud computing for social care is a new emerging area with no assessed solutions specifically tailored to support the previously described motivating social care delivery scenario. Instead, more research efforts have been directed to provide cloud-enabled mobile healthcare which can be considered a close research field posing similar challenges. The limited computational power, battery life and memory capacity of existing mobile devices significantly reduce their ability to execute resource-intensive health applications, such as patient data analysis, and patient medical data storage. In addition, mobile devices used by patients and physicians can experience loss of connectivity due to network disconnections that hampers real-time medical data accessibility and prevent users from synchronizing medical updates between the mobile device and the back-end healthcare management system.

The integration of mobile health applications and mobile cloud computing is expected to facilitate the deployment of cost-effective, scalable and flexible mobile healthcare systems, to limit the workload of mobile devices by offloading heavy computation on health data.

In particular, mobile cloud computing offers several advantages which include the possibility to offer richer functionalities and services, such as speech recognition, medical video streaming and medical data mining, to improve efficiency in terms of computation, storage, communication and energy and to reinforce reliability. For instance, when the battery of a mobile device dies, the mobile application can still continue running in the cloud without interruption.

Different frameworks and solutions have been proposed to support mobile cloud-enabled healthcare management. In [7] a cloud-based system is presented that enables collecting patients vital data via a network of sensors connected to legacy medical devices, and delivering the data to a medical center cloud for storage, processing, and distribution. The main benefits of the system are that it can provide users anytime real-time data collecting, eliminates manual collection work and the possibility of typing errors, and eases the deployment process. Another cloud computing protocol management system has been proposed in [8] that provides multimedia sensor signal processing and security as a service to mobile devices to relieve mobile devices from executing heavier multimedia and security algorithms in delivering mobile health services. That can also improve the utilization of the ubiquitous mobile device for societal services and promote health service delivery to marginalized rural communities. An interesting cloud initiative called Dhatri is described in [9] which leverage the power of cloud computing and wireless technologies to enable physicians to access patient health information at anytime from anywhere. Similarly in [10] a cloud-based prototype emergency medical system for the Greek National Health Service is described that integrates the emergency system with personal health record systems to provide physicians with easy and immediate access to patient data from anywhere and via almost any computing device while containing costs. Other solutions that witness the benefits of mobile cloud computing for mobile health applications include the Mobile Cloud for Assistive Healthcare infrastructure for assistive healthcare [11], the cloud-enabled WBAN architecture described in [12] that shows the functionality and reliability of mobile cloud computing services, and the mobile cloud telemedicine framework in [13] that takes advantage of the real-time, on-site monitoring capability of Android mobile device and the abundant computing power of the cloud.

3 Background on Speech Recognition Tools and Mobile Cloud PaaS Platform

This section introduces some background knowledge to provide a better understanding of the MoSSCa infrastructure. First, we focus on the speech recognition workflow and available environments and tools to implement it. Then, we introduce the Cloud Foundry open source PaaS that we used to realize the mobile cloud support for elastic MoSSCa semantic-enriched speech recognition service provisioning.

3.1 Semantics-enriched support for Speech Recognition and Natural Language Processing

Automatic Speech Recognition (ASR) can be defined as the computer-driven transcription of spoken language into readable text in real time: it is the technology that allows a computer to identify the words that a person speaks into a microphone or telephone and convert it to written text. The quality of these systems has reached levels adequate to make them very diffused and common in a variety of different contexts. Nowadays, these systems are available on every modern smartphone and they usually produce multiple matches (caused by problems like noise, accents, dialects..) ordered by a confidence level. In order to extract the linguistic and semantic information to identify the best textual option, it is possible to analyse the interdependencies and meaning between the textual matches using pipelines performing Natural Language Processing (NLP), that are commonly used to associate metadata, typically called annotations, to the text. To perform this task there are various available tools, and in this work we decided to employ the General Architecture for Text Engineering (GATE) framework [14] due to its openness and ease of use. In fact, GATE Java-based infrastructure offers a large variety of tools and components able to process human language: from the most simple tokenisers and sentence splitters to more advanced parsers or name entity research tools. There are three particular types of components:

- LanguageResources (LRs): represent entities such as lexicons, corpora, and ontologies;
- ProcessingResources (PRs): represent entities that are primarily algorithmic, such as parsers;
- VisualResources (VRs): represent visualization and editing components that participate in GUIs.

We built a pipeline of PRs, where the most advanced tools rely on the results returned after the sequential execution of the previous resources, i.e., the annotations which include information associated to the text analysed (e.g., tokens, *part-of-speech -POS-*, features), to process and create further annotations using LRs. In particular, the main GATE components used in MoSSCa include the Stanford parser [15] and the *gazetteer*, introduced in the following paragraphs.

The *Stanford parser* is a probabilistic parsing system with data files available for parsing Arabic, Chinese, English and German. This PR acts as a wrapper and translates GATE annotations to and from the data structures managed by the parser itself. In particular, this PR produces a type of annotation called *Dependency*, that represents a grammatical relation between words in a sentence. Stanford Dependencies (SD) are triplets composed by the name of the relation, a governor word and a dependent word. For instance, these type of relations represent which word is, respectively, the main verb, the subject, and the object in a sentence, or which noun an adjective is referring to.

As regards the *gazetteer*, it refers primarily to a LR, namely, a set of lists containing names of entities such as cities, organisations, days of the week, etc. These lists are used to find occurrences of these names in text defined as *entity recognition*; in addition, the same term (i.e. "gazetteer") is also used to refer the PRs that use those lists to find occurrences of the names in text. When a gazetteer PR is running on a document, it will enrich it with specific *Lookup* annotations for each matching string in the text that matches with the entities in the gazetteer LR. In particular, the *OntoRoot* gazetteer was adopted to create these lists of entities from two ontologies representing the Social Care domain.

The first ontology is derived and extended from the Social Care Taxonomy [3] and it provides a controlled vocabulary and hierarchical arrangement of social care topics for browsing, searching and indexing material on Social Care Online. Its extended version consists of 1085 classes, 26 object properties and 3 data properties. In addition the knowledge bases were extended through the use of Human Diseases Ontology [16], a standardized ontology with the purpose of providing the biomedical community with consistent, reusable and sustainable descriptions of human disease terms, phenotype characteristics and related medical vocabulary. The version adopted comprises 8797 classes and 15 object properties.

Finally, in order to glue together the semantic-enriched output obtained from the GATE toolchain with other distributed components, mainly MoSSCa mobile client app we exploit the lightweight Linked data format. Linked data has emerged as a paradigm for information integration across domains and systems [17], that typically adopts the RDF[4] representation. The basic RDF model stores information as a set of labeled edges across nodes with unique, global identifiers. Typically, a standardized triple notation is used, consisting of RDF terms, typically referred to as Subject-Predicate-Object triples. Data are usually stored in a data store (typically referred to as triple store or RDF store) and queried through the SPARQL Protocol and RDF Query Language (SPARQL) query language [18]. Among the several possible availalbe RDF serialization formats, in our work we adopt the JavaScript Object Notation for Linked Data (JSON-LD), a lightweight syntax to serialize Linked Data in the widely diffused JSON format, so to ease the integration with Web-based programming environments, such as the MoSSCa backend exposing Representational State Transfer (RESTful) APIs.

3.2 Cloud Foundry

Cloud Foundry is an open-source Platform as a Service infrastructure, initially developed by VMWare with the aim of promoting data backup and recovery and accelerate

3. http://www.scie.org.uk/publications/misc/taxonomy.asp
4. http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

the entrance into the Cloud [19]. Cloud Foundry, primary written in Ruby, a dynamic, general-purpose object-oriented language, provides a number of frameworks, languages, and ready to use Services to the end users for implementing their web applications.

Focusing on its distributed architecture, Cloud Foundry, to grant the widest possible scalability, reliability, and elasticity, follows some design guidelines, namely, loose coupling with event-driven and non-blocking interactions. Cloud Foundry includes: i) a local database component to store its internal state; ii) a set of internal core components realizing all main PaaS management functions; and iii) an API component acting as a service front-end to export sevice functionalities via interoperable RESTful APIs. Interactions between internal service components are facilitated by a publish/subscribe messaging service based on Not Another Tibco Server (NATS) and acting as common communication bus, whenever possible, and by management functions exposed via external RESTful APIs calls or NATS, especially to limit synchronization costs when frequent interactions are necessary such as for continuous resource monitoring.

Following these main design guidelines, Cloud Foundry consists of five main components: the Cloud Controller, the Health Manager, the GoRouter, the Droplet Execution Agents (DEAs) and Warden container, and a set of Services. Other components include the User Account and Authentication (UAA) server for PaaS costumer authentication, and Stacks to provide a common set of development tools and libraries; for additional details about all the Cloud Foundry components illustrated in Figure 1, we refer interested readers to [19].

The *Cloud Controller* represents the core PaaS system component: this component exposes the main REST interface providing a set of APIs for PaaS clients to access the system. It maintains an internal database with specific tables to register apps, services, service instances, user roles, and more configurations.

The *Health Manager* is a standalone daemon with the aim of retrieving the current applications status and of periodically checking and managing the application to recover it in a safe status in case of faults. For instance, when an application crash happens while it is running, the Health Manager will ask the Cloud Controller to re-start the application instance.

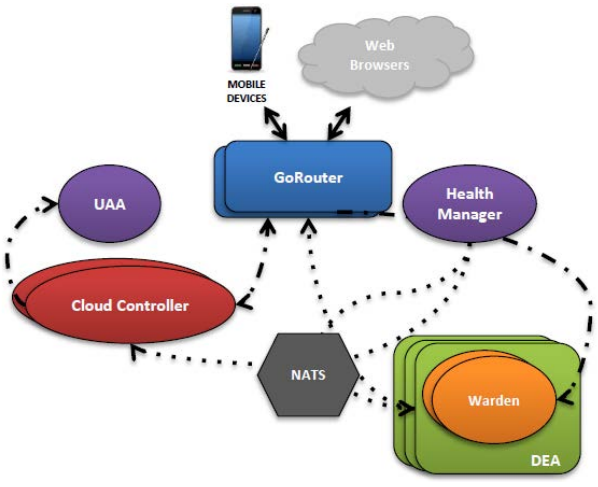


Fig. 1. Cloud Foundry Architecture Overview

The *GoRouter* is the daemon that routes incoming traffic to the appropriate component, usually the Cloud Controller or a running application on a DEA node such as in the case of the MoSSCa backend.

Another essential component is the *DEA* that acts as local agent deployed at each computing node to manage the whole application lifecycle. In Cloud Foundry, applications are wrapped up and deployed as self-contained execution entities called *droplets*. Each droplet contains all needed configuration/binding parameters as well as stop/start scripting logic to de-/activate them. Cloud Foundry support droplet execution isolation via *Warden* containers whose primary goal is to provide isolated environments that can be limited in terms of CPU usage, memory usage, disk usage, and network access by using different kernel resource namespaces [19]. DEA acts as coordinator for all Warden containers running on the same node and realizes several main key functionalities, such as staging and running applications (i.e., droplets), managing their lifecycle of each application instance running it in a separate Warden container, and starting and stopping containers/applications upon requests from the Cloud Controller.

Every component communicates with each other with *NATS*, a lightweight publish-subscribe and distributed queuing messaging system [20]. The NATS client provides asynchronous communication, in order to grant non-blocking behavior when publishing messages through the service. When a component of Cloud Foundry first boots, it subscribes to the NATS server using the IP of the machine in which the messaging server is running and the port on which is listening using specific credentials (a username and a password). It then subscribes to all the subjects its interested in, and publishes messages such as heartbeats and advertising communications.

Cloud Foundry provides a set of third-party components, such as an external DB service, called *Services*. Services are any type of add-on which can be provisioned alongside web applications, such as SQL and no-SQL databases, messaging system, etc. Each Service has a common architecture consisting on two components: a Service Broker that communicates with the Cloud Controller via a set of API for the provision and binding of a service instance; and a Service Node, where the real service processes are running.

Let us conclude this section noting that Cloud Foundry, notwithstanding the management facilities, still lacks a thorough support for elastic scaling of applications at run-time. At the current stage, Cloud Foundry already offers some APIs to scale, either horizontally by adding new application instances or vertically by requiring more resources (e.g., memory and disk space), but those APIs have to be invoked directly by application providers; at the same time, some companies are offering their own autoscale support products for Cloud Foundry. However, an internal Cloud Floundry core component in charge of taking over all needed monitoring and dynamic reconfiguration operations needed to elastically scale applications and making them available to the final users automatically is still missing.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2570757, IEEE Transactions on Cloud Computing

4 MoSSCa Architecture

The design and deployment of a semantic-enriched speech recognition service for social care delivery pose several sociological and technological challenges stemming from the need to cope with different requirements.

From the sociological point of view, social workers should be able to have a more convenient, real-time and reliable experience when using an automated speech recognition service rather then when taking notes or writing assessments by hand in order to convince them to change their habits. In addition, by operating from mobile devices caregivers should rely on efficient speech recognition services that do not drain mobile device's energy impeding them to use their mobile devices for other tasks.

From the technological point of view, since the social caregiver's professional life can be very hectic, they often take crucial information about their patients during their sessions quickly and often using a lot of abbreviations. These confused notes are then usually clear just to the author and not very helpful to the team to plan coordinate actions or therapies. A quick way to gather this type of information in an understandable way is crucial to improve social care delivery results and effectiveness. To speed up interactions with the patient, the most convenient way to insert the information would be to speak directly to the mobile phone summarizing the patient information. This method anyway faces the technical challenge to transform the speech into a reliable and machine-understandable information. Information misunderstanding could produce several health risks for the patient and legally risks for the social worker, thus requiring reliable speech recognition techniques. In addition, without properly designed functionalities, it is impossible for a machine to understand information semantics or contextualizing the information gathered.

To address these requirements, MoSSCa adopts two main guidelines, a *semantic-enriched approach* to speech recognition and a *mobile cloud support*. In particular, the MoSSCa mobile cloud-based architecture is composed of two modules: a client and a server application on the cloud, as represented in Figure 2. The client application must cope with the need for mobility of social carers. For this reason the speech recognition is performed through mobile devices, while the results are analysed first on a server application, where the domain of the speech is represented through ontologies, and, depending on the operation chosen, they are then compared on the mobile device with the context, which consists in the data previously known about the patients. This double check on the data enhances the reliability of both the machine understandable information produced and the text selected as best option. Moreover, let us note that storing the context at the mobile side allows to keep the server side very simple and stateless and to always have local access to patient information even in presence of possible intermittent disconnections.

Splitting the whole process was even almost necessary under many different resource perspectives. First, since ontologies and models (representing the components of the language to be analysed) to perform the NLP tasks may be demanding in terms of storage available at the mobile device. Second, and most important, notwithstanding ever-increasing computing power available on mobile devices nowadays, NLP tasks require processing large models and knowledge bases (such as the two ontologies used in MoSSCa, that could increase for other different applicative domains) and that forced us to exclude the possibility to run at the mobile device (see also experimental results shown in Figure 7). Hence, moving the semantic enrichment workflow execution at a powerful cloud backend, with more resources and tools dedicated, is indeed a strict requirement in these mobile cloud scenarios.

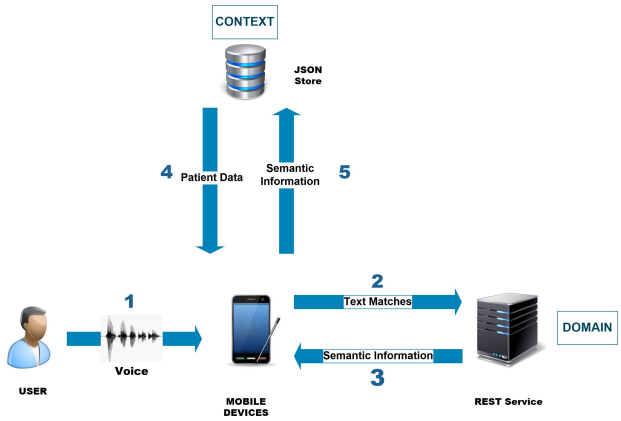


Fig. 2. Cloud infrastructure workflow

4.1 MoSSCa Mobile Client

We designed the mobile application to locally implement several important parts of the speech recognition process, with the design goal of making it resilient to possible intermittent disconnection and idiosyncrasies of the wireless medium. It locally translates the speech into text, then it exploits the powerful cloud backend to remotely execute the semantic enrichment step, but finally the mobile client also locally stores the context of all patients cared by the social worker, by iteratively extending this context with the new semantic-enriched information input by the social carer. In particular, existing context is used to evaluate the semantic data extracted (by the server) from the text, to check if the concepts recognized in the text match with the clinic data already collected in the past about the client. When that match occurs, we define the semantic-enriched speech recognition as valid and we update the local context knowledge base accordingly, otherwise we discard it. For instance, if the user speaks of a disease that she has already suffered, it is considered valid only if there is a match with a RDF triple already present in her context.

Focusing on the internal architecture, with the goal to grant the widest portability the realized mobile application it is mostly based on pure HTML5 and Javascript technologies, but we also had to access some native device functionalities. The design and development of the app, that runs on Android, was heavily influenced by the the Industry Solutions Mobile Core Framework developed by IBM, namely, the IBM Worklight framework. Worklight supports HTML5 and is compatible with all most diffused Javascript frameworks such as jQuery, DOJO, and PouchDB

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2570757, IEEE Transactions on Cloud Computing

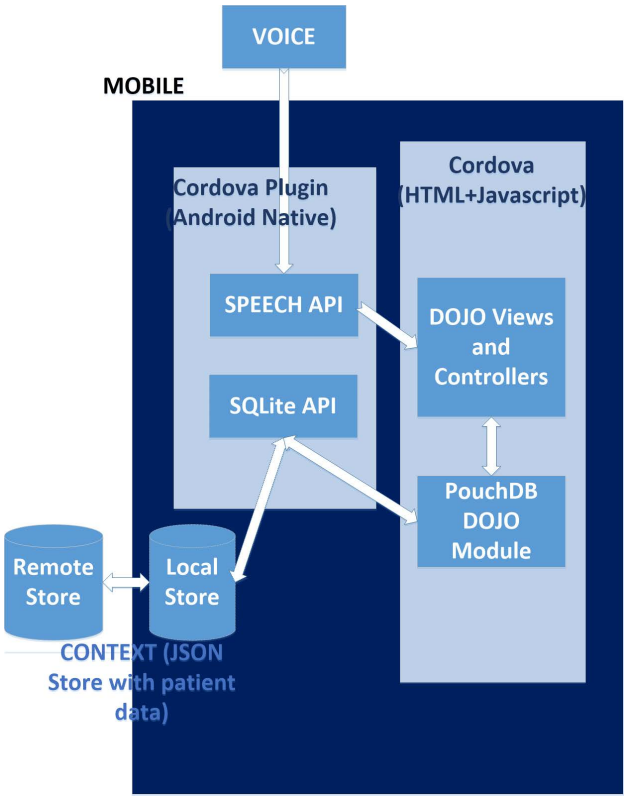


Fig. 3. Mobile architecture

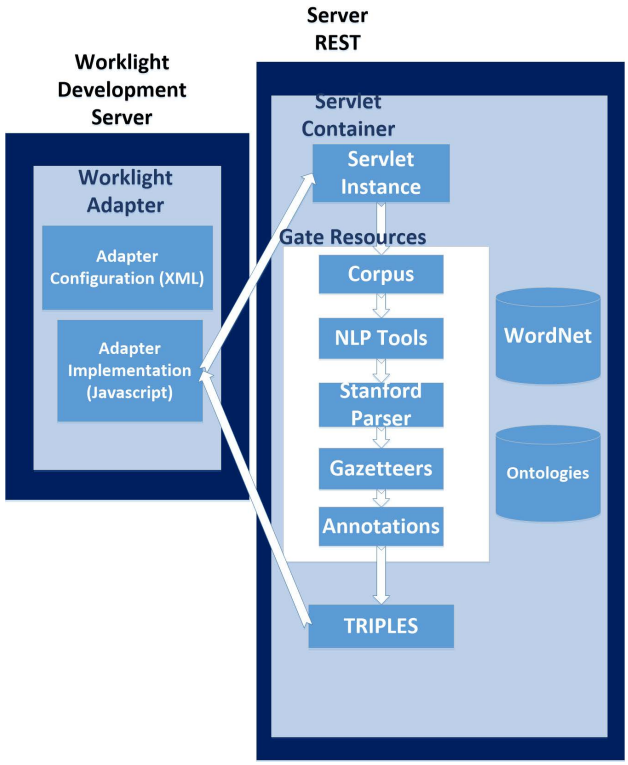


Fig. 4. Backend architecture

to store JSON tuples; through Apache Cordova it also allows to access native device functions via a native-to-Javascript plugin.

Figure 3 shows Mobile Client architecture and all its main components that work in a pipeline. First, we exploit native speech recognition Android API to extract text results, ordered by confidence score, from voice samples; then, according to this order we build the requests to direct to the cloud backend. To perform server invocations we employ the chain of responsibility pattern: the first match is sent to the server application and the results obtained are evaluated on the context data already known about the patient; if the results are valid the invocation ends, otherwise the next match in the confidence level order is sent until a valid result is retrieved or all possible matches are evaluated.

Finally, focusing on context knowledge base storage, semantic-enriched data returned by the server are returned as a JSON-LD document. This RDF format is very handy and can be directly stored in a JSON store; in MoSSCa we opted for PouchDB, a local JSON store that runs inside mobile web browser. Moreover, via Cordova and the SQLite API we integrated PouchDB with the local SQL Lite server so to persist JSON-LD triples for future interactions with the same patient.

4.2 MoSSCa Server and Backend Components

MoSSCa server side is in charge of parsing incoming text and to extract semantic information from it. To foster wide scalability we decided to realize the service as a stateless REST server, by including in it also all needed ontologies and lexical databases, to make it easily replicable and portable. Moreover, in order to obtain effective results we had to integrate at the server side several different tools and several different LRs and PRs by composing them in a complex toolchain as better explained in the following (see Figure 4).

For the development of the REST service we used the Apache Wink framework; in addition, because the mobile application uses IBM Worklight and its optimized Worklight communication format, the backend includes also the Worklight Adapter component (see Figure 4). The REST service receives the text and process it through a NLP pipeline based on GATE; unfortunately, GATE does not natively support pipeline distribution across different hosts, but it supports concurrent execution of thread-safe GATE pipelines by multiple threads. Hence, we designed and configured the REST server with a pool of thread resources to obtain local server scalability, and then we support automatic and elastic vertical scalability through application replication across different DEA as better explained in the next section.

Focusing on the whole request processing workflow, when incoming requests arrive, the service feeds them into the GATE pipeline that performs different concurrent tasks, namely, sentence splitting, tokenisation, POS tagging, morphological analysis, parsing, and name entity recognition through the knowledge bases described in Section 3.1. Then, the Stanford parser recognizes which words represent subjects, predicates, and objects, and gazetteers annotate text with name entities; through the combination of these two types of information it is possible to create triples <subject, predicate, object>with the URIs of the entities recognized

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2570757, IEEE Transactions on Cloud Computing

from the ontologies, in the RDF format then sent back to the Mobile Client as JSON-LD.

Delving into finer details, apart these major steps, the text processing includes also other steps aimed to further increase the effectiveness of the semantic-based recognition. An important one is evaluating semantic matches against the knowledge bases, not only for the received text, but also for synonyms of those incoming words. This is particularly important because while for the classes and entities there are already synonyms described in the knowledge bases adopted for the Social Care domain, the same does not apply for the object properties (the predicates). For this reason, a lexical database was used to retrieve more synonyms for these predicates to increase the possibilities of matches in the knowledge bases. Since the language tested was English, we employed the widely accepted WordNet[5]: when a predicate is found by the Stanford Parser, but the Gazeteer does not produce any result, we retrieve a list of synonyms and use them to replace the predicates until they are all analysed or a valid match for the synonym is retrieved from the ontologies.

4.3 Cloud Foundry PaaS Support for MoSSCa

Semantic-enriched speech recognition asks for an elastic support able to dynamically adapt to the current incoming requests, by replicating the application on unloaded nodes and, eventually, grow and shrink the cluster by need. As anticipated in Subsection 3.2, Cloud Foundry does not completely support all the features required in a dynamic Cloud environment. In fact, Cloud Foundry neither elastically scales up/down, nor is able to execute resource rebalancing in order to exploit more computational power provided by nodes temporarily added to the backend.

To solve the above issues, we have designed our original Cloud Foundry PaaS autoscaling support for MoSSCa shown in Figure 5. The distributed architecture relies on two main components: the Provisioning Engine and the Monitoring Aggregator.

The *Provisioning Engine* is the core component that acts autoscaling manager; it listens for scaling requests from DEA and, interacting with the Monitoring Aggregator, takes informed scheduling decisions about the DEA to choose for the deployment of the new application instance taking care of performing the up(down) scaling procedure by interacting with the Cloud Controller.

The *Monitoring Aggregator* is the module responsible of collecting monitoring information about both physical and PaaS resources. A traditional Cloud Foundry PaaS can collect monitoring information about the resources used by all application instances managed by each DEA, and use them to take scheduling decisions. However, the PaaS level typically has no (cross-layer) visibility of the underlying physical host performances, this is true especially when the PaaS runs atop a virtualized IaaS layer as in the case of Cloud Foundry over OpenStack deployments. In this way, if the new application instances are instantiated on a host with limited (physical/virtual) resources, this can lead to performance degradation without the DEA and PaaS being able to react to it. This calls for an external monitoring

5. http://wordnet.princeton.edu

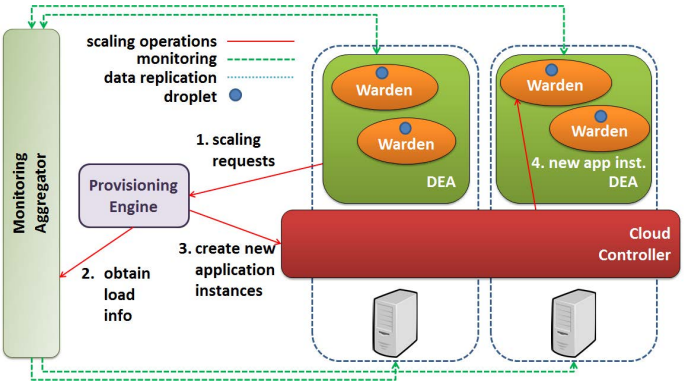


Fig. 5. Cloud Foundry MoSSCa support

facility that periodically collects and tracks metrics also from physical hosts, such as memory or CPU consumption and disk I/O rates.

In addition, we extended the DEA, deployed at each physical node, to collect performance indicators at the application layer, such as request processing delays and number of correctly served incoming requests. Starting from those applicative information, the DEA periodically monitors the situation locally and if detects that the load is over (below) an upper (lower) threshold, it sends a scaling request to the Provisioning Engine. Monitoring operations on physical hosts allow the Provisioning Engine detect load conditions effectively, even those caused by other VMs potentially running on the same physical host.

5 MoSSCa Implementation

5.1 Context-Aware Speech Recognition

In order to disambiguate the text from the speech recognition, the text was analysed first on the backend application through the knowledge bases that represent the domain and then on the mobile application through the retrieved JSON-LD document that represent the context (the patient history). Nevertheless it is not always so straightforward to bridge the gap between the user vocabulary and the entities that need to be retrieved from the knowledge bases. In fact, many times in common language we use unclear subjects as pronouns or more complex cases as anaphoras, words whose meaning depends on previous ones. For instance for a machine it is very difficult to understand the sentence "He has diabetes which is caused by malnutrition".

To resolve the ambiguity about the generic subjects on the backend application there is a default value that tries to guess who the user is referring to when personal pronouns are used. Since the use case tested was the one related to Social Care this value was setted to the generic class that describes patients in the knowledge bases. In particular the subject of the previous sentence "He" will produce the generic *ibm:scv#patients* class. This guess was possible thanks to some checks on the candidates triples before creating them that are going to be explained in the next section. If the triple make sense and it is really returned to the mobile application, the subject of the triple is then replaced with the patient that was chosen by the user such as *ibm:scv#Vincent* during the tests with the previous example.

To cope with anaphoras instead a large variety of Dependencies types were analysed to understand which one is the entity referred by subjects like "that" or "which" that are used as relative pronouns or conjunctions. This is useful to understand in the previous sentence that the real subject that "which" is referring to is "diabetes".

5.2 Semantic Text Disambiguation

To create the semantic triples we have to analyze the annotations produced by the Stanford parser and the OntoRoot gazetteer. First, we consider the grammatical relations represented as Dependencies: in fact, the Stanford parser produces some relations such as *nsubj(has, he)* that indicates that "he" is the subject of the verb "has" or *dobj(has, diabetes)* that selects "diabetes" as the object of the verb "has". These relations are useful to retrieve the subject and the predicate of the triples, but the concept of object in RDF is ambiguous due to the huge number of possible types represented as dependencies. To cope with that, after retrieving the subject and predicate of a candidate triple, all the words connected to the predicate are analyzed to verify if a Lookup annotation; we search for an annotation derived from the ontologies such as *(sentence="diabetes", URI: "ibm:scv#diabetes", type: "class")* that indicates which class corresponds to a specific piece of text. If one is found, the candidate triple is ready for further checks to verify its meaning.

Indeed before creating the final triple, the infrastructure verifies, through the properties domain and range of each predicate described in the ontologies, if the subjects and objects are admissible for the predicate recognized in each candidate triple. These two properties describe which (type of) values are admissible as subject and object, considering the inferences over subsumption-based type hierarchy. If the entities retrieved are valid, the triple is not discarded. This check is fundamental to resolve the guess on generic subjects previously described and to evaluate synonyms of the predicate through WordNet as described in Subsection 4.2.

As a final consideration, let us outline that MoSSCa support for semantic-based speech recognition allows to manage specific diseases. The system depends on the ontology coverage to annotate the relevant terms in the text, giving a semantic meaning to the triples that are extracted from it. If the object (or subject) of a triple is not annotated as a known type (e.g., a disease) the triple cannot be semantically disambiguated or verified. This, however, does not exclude to consider additional diseases. The coverage of new diseases can be extended by adding new ontologies and extending the existent ones. For instance, Life Science[6] is a well-represented domain on the Linked Open Data and our system can benefit from reusing these Web-wide wealth of resources, rich in meaning and structure.

5.3 Autoscaling Components and Integration with Cloud Foundry

We implemented the Monitoring Aggregator with the goal of granting the widest possible visibility about all used and available resources at the different cloud protocol stack levels. The Monitoring Aggregator is heavily based on our previous work on scalable and semantic-enabled Dargos and SMACS monitoring frameworks [21], [22], and integrates also with the widely adopted opensource Zabbix monitoring tool [23]. Through the RESTful APIs exposed by Dargos, SMACS, and Zabbix server, the Monitoring Aggregator retrieves and makes available monitoring information, such as memory and CPU utilization at physical and IaaS levels as well as application performance parameters. In particular, the Monitoring Aggregator maintains the correspondence between DEAs and the physical host; every time the Provisioning Engine requests monitoring information about DEAs, it can also obtain monitoring data of all VMs and physical hosts.

For the implementation of the Provisioning Engine, we took advantage of the features of the highly flexible, pluggable and open-source Cloud Foundry. All communications are based on NATS by subscribing to existing topics, such as for interactions with the Cloud Controller and GoRouter, but also introducing new topics as needed, such as for monitoring data exchange with the Monitoring Aggregator and autoscale control messages with the DEA and the Cloud Controller. Focusing on application instance activation function, we created a new control topic called *instance.scale*. Whenever the CPU or memory load of a certain application instance is over/under a threshold, configurable by the user, the DEA interacts with the Provisioning Engine to init the up/down-scaling process by publishing a scaling request on the instance.scale topic. Then, the Provisioning Engine obtains monitoring data from the Monitoring Aggregator, decides the DEA where the new application instance will be instantiated by choosing the least loaded one from the perspective of PaaS, IaaS, and physical indicators (it uses a simple linear combination of collected performance monitoring indicators), and sends the scale up/down command to the Cloud Controller, that has been extended to listen on our new instance.scale topic.

Of course, up/down-scaling processes can take some time to be completed. Cloud Controller coordinates with the target DEA to push (i.e., download) the droplet, only if needed because not already present there, and to start the application instance in a separate Warden container; then, target DEA notifies other components, especially the GoRouter through the *router.register* topic about the availability of a new up-and-running application instance. Thereafter, new incoming MoSSCa requests will be forwarded by the GoRouter to all MoSSCa application instances, including the new one, by using the default round robin strategy. Let us note that more complex load balancing strategies are possible, but we preferred to use the default round robin policy, without implementing also active load balancing at runtime, because it was sufficient to scale our semantic-enriched speech recognition application due to the stateless nature of the MoSSCa backed. In addition, we made minimal changes on purpose to maintain the widest possible portability and interoperability of our autoscale support with the vanilla Cloud Foundry distribution.

6 Experimental Evaluation

MoSSCa research, development and deployment have been

6. http://lod-cloud.net/state/

driven by an industry scenario within the context of the IBM Curam solution[7]. To test the performance of MoSSCa, we ran an extensive set of both functional and performance tests. From a technical perspective, our mobile cloud testbed environment consists of a various Samsung I8190 smartphones featuring a dual core ARM Cortex-A9 processor running at 1 GHz and 1 GB RAM running Android 4.1, and an OpenStack platform deployed on 3 physical Linux servers, each equipped with 2 16-cores AMD Opteron 6376 processor at 2.3 GHz and 32 GB RAM, connected through three 1 Gbps LANs, and running Linux Ubuntu 12.04. Focusing on the Cloud Foundry infrastructure, it has been deployed atop OpenStack and consists of 8 VMs, each running a separate CloudFoundry component: GoRouter, UAA, CloudController, HealthMonitor, NATS, Syslog Aggregator, and 2 DEAs. Moreover, other 3 VMs were deployed to provide services used by Cloud Foundry components, namely a Postgres database, a NFS server, and etcd. Each VM is equipped with 1 VCPU, 2GB RAM and 20 GB HDD. From a social perspective, we first executed some seminal tests that involved a wide set of colleagues. Those tests demonstrated a high correlation between the nationality (and especially, English v.s. non-English native speakers) and speech recognition performances obtainable with MoSSCa, while there was a small variance between performances obtained for the different people with the same nationality. Consequently, we decided to restrict our study to a smaller set of 11 volunteers who were subjected to a series of longer tests; 5 of them are native English speakers and 6 are not: we asked them to speak to the support system 19 queries extracted from two patients histories. In particular, the 19 proposed queries are different: there are 6 queries composed only by subject, 9 by predicate and object, and 4 more complex ones with more complements and passive queries. An example of a query of the first group is ”he has committed vandalism”, one of the second group is ”john suffer from dementia that was complicated by alcohol misuse” and one of the third group is ”the patient is supported by job”.

Delving into finer details, we analyzed several different aspects; in particular, we first focused on functional aspect of MoSSCa semantic-enriched speech recognition and tested that:

- it enhances the performances of the speech recognition system,
- it is quick enough to justify its use instead of normal handwritten notes,
- it retrieves the correct semantic information associated to what the user said,
- it is able to check on domain and range properties to improves overall performances.

Then, we focused on MoSSCa mobile cloud support performances and assessed that:

- it performs well in a single node settings deployment in the case of both single users and larger groups,
- it is able to elastically autoscale and absorb load peaks by requiring reasonably short times for PaaS dynamic reconfiguration.

7. http://www-03.ibm.com/software/products/en/social-programs

User\Query	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	Total
American	1,00	0,75	0,90	0,67	0,85	1,00	1,00	0,86	1,00	0,00	0,17	0,13	0,88	1,00	0,60	0,86	1,00	1,00	0,44	
				+,11																0,11
Irish1	1,00	0,75	0,90	0,56	1,00	1,00	1,00	0,86	0,60	0,67	0,67	0,88	0,50	1,00	0,80	0,86	0,00	0,45	0,78	
																				0,00
Irish2	0,60	0,00	0,40	0,67	1,00	1,00	1,00	0,86	0,60	1,00	1,00	0,75	1,00	0,83	0,80	1,00	0,00	1,00	1,00	
														+,17			+1,			1,17
Irish3	1,00	0,75	1,00	0,44	1,00	1,00	0,67	0,86	1,00	1,00	1,00	0,88	1,00	0,83	0,80	1,00	1,00	1,00	1,00	
																				0,00
Irish4	1,00	0,75	0,80	0,00	1,00	1,00	1,00	0,71	1,00	0,50	0,67	0,63	1,00	1,00	1,00	1,00	1,00	0,45	0,89	
																			+,11	0,11
Greek	0,90	0,75	0,40	0,56	0,54	0,50	1,00	0,57	0,60	0,83	0,83	0,75	0,75	0,33	0,60	0,57	1,00	0,64	0,11	
																				0,00
Italian1	0,90	0,50	0,60	0,33	0,77	1,00	1,00	0,86	0,60	1,00	1,00	0,88	1,00	0,83	0,60	1,00	0,00	0,91	0,78	
				+,44										+,17	+,4		+1,			2,01
Italian2	0,10	0,00	0,60	0,56	0,85	-0,33	1,00	0,71	0,60	0,50	0,83	0,75	0,13	0,50	1,00	1,00	1,00	0,55	0,44	
												-,25								-0,25
Italian3	0,20	0,00	0,80	0,44	0,92	0,67	1,00	0,57	1,00	0,67	0,67	0,50	0,50	1,00	0,60	0,71	1,00	0,82	0,33	
	+,1									+,17										0,27
Lebanese	0,80	0,75	1,00	0,44	0,77	1,00	0,33	1,00	1,00	0,83	0,83	0,88	0,63	0,33	0,80	1,00	1,00	1,00	1,00	
				+,33							-,17									0,17
Portuguese	1,00	0,75	0,80	0,11	0,92	0,00	1,00	0,43	0,80	0,67	0,83	0,75	0,75	0,83	0,60	0,86	0,00	0,73	0,67	
								+,57	+,2					+,17	+,2	+,14				1,28
Total	0,10	0,00	0,00	0,89	0,00	0,00	0,00	0,57	0,20	0,17	-0,17	-0,25	0,00	0,50	0,60	0,14	2,00	0,00	0,11	

Fig. 6. Word Accuracy Improvements with Context Update

Starting from the functional tests, to evaluate speech recognition we adopted the most diffused metric in this area: the *word accuracy*. Given a reference sentence, word accuracy is defined as *1-WER*, where WER stands for Word Error Rate and is expressed as *wrong_word_guesses/all_words_in_reference_sentence*, where *wrong_word_guesses* includes errors due to wrong substitutions, insertions, and deletions. To highlight the improvements caused by the support system, the matrix in Figure 6 shows the values representing the word accuracy of the matches obtained with and without the semantic-enriched support. In the first column, we present the nationalities of the volunteers, while the first row is used to distinguish the 19 queries chosen to test the system. For each user, the first row represents the values of the match that the system would have chosen without the support, relying just on the confidence scores. Usually the semantic support agrees with these scores, but when a disagreement occurs, the second row represents the word accuracy changes obtained with the matches chosen by the support system. The scores are then decorated with a colored scale from red (low word accuracy) to green (high word accuracy) to highlight the differences between the recognized speech and the reference; light blue colors, instead, refer to improvements (the darker the better) due to our semantic-enriched support. As it is visible from this test, when the support system chose different matches it caused almost always an improvement in the word accuracy. In particular, the average values calculated for each user in the last columns highlights how much the not natives English speakers benefit more than natives mother tongue ones from the use of this semantic support. In fact, the first ones obtained an average improvement of 0.58 compared to the 0.28 obtained by the latter ones.

Figure 7 represents in logarithmic scale the time (ms) necessary to parse the triples. The goal of this analysis is to confirm the need of our mobile cloud approach; we quantified the performance gap between our *mobile cloud* implementation and a *mobile only* implementation. The *mobile only* implementation, based on the same libraries used at the server side, is specifically implemented to execute only locally at the Android smartphone. In Figure 7, we compared it with the *mobile cloud* implementation for the time necessary to perform about 500 invocations, which are produced by the multiple tries necessary to parse the 209 queries (19 for each one of the 11 users). There is a signif-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2570757, IEEE Transactions on Cloud Computing

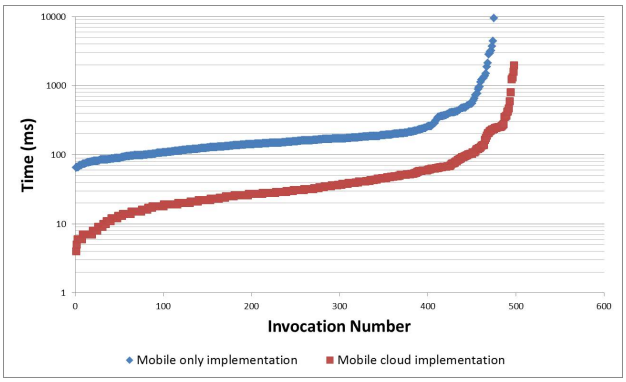


Fig. 7. Time to parse triples on server

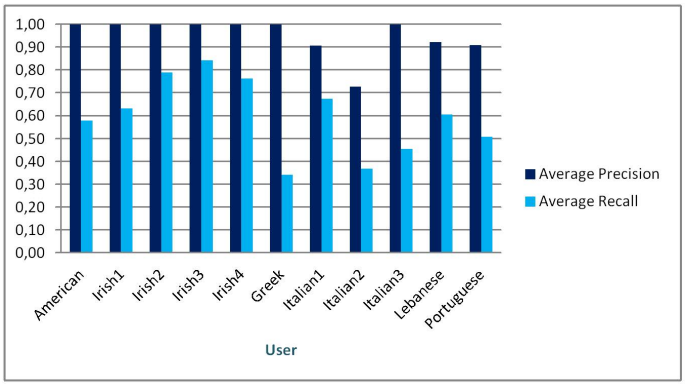


Fig. 8. Average precision and recall per user

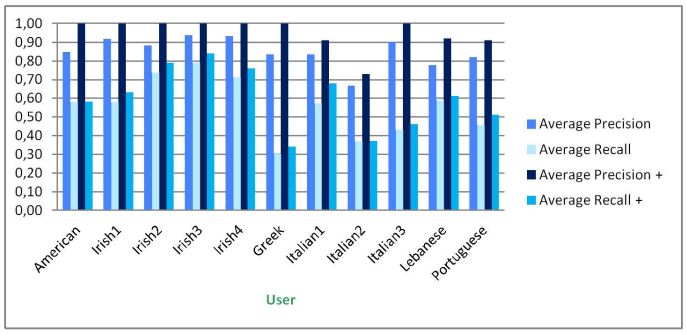


Fig. 9. Semantics improvement with domain and range check

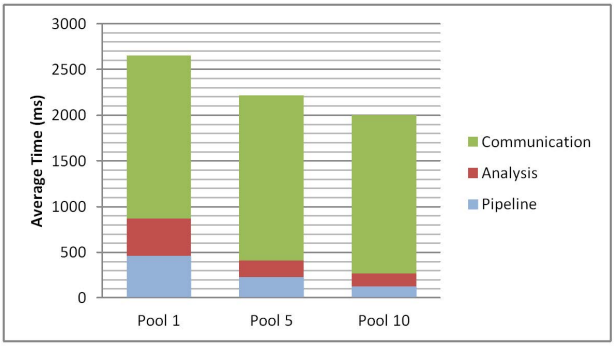


Fig. 10. Time to parse triples with 50 concurrent queries

icant difference between the mobile only implementation, with times that go from 60ms to 1s, and the mobile cloud implementation, that typically presents times from 5ms to 200ms. About the outliers, the mobile cloud implementation ones are always below 2s, while in the case of the mobile only implementation there are cases over 2s up to 10s, thus making worst cases intolerable. Hence, following results are all based on the mobile cloud implementation.

About semantic correctness, we adopt *precision* and *recall* metrics, as usually defined in Information Retrieval, and shown in Figure 8. Focusing on the different performances between precision and recall, they are mainly due to the fact that *recall* depends mostly from the speech recognition performance that is highly affected by different accents and by the performance of the third-party libraries used for speech recognition. *Precision*, instead, is more influenced by the ability of effectively using semantics to process recognized speech, and depends on original MoSSCa components and on the usage of good knowledge bases from which the triples are created. Overall, obtained results are good, in particular for precision.

Since the system is highly configurable, it is possible to turn off the check on domain and range. That characteristic makes it possible an easy measurement of the improvement of the quality of the semantic information produced, as in Figure 9. The bar chart visualizes for each user the averages values of *precision* and *recall*. In particular, the first two bars represent the case without our validation, while the following two the performances previously described: precision improvement is more than 10% in most cases.

Focusing on system performances, instead, we first tested scalability of a single application instance, by loading it with 50 multiple concurrent invocations and for different pipeline (thread) pool sizes, respectively, 1, 5, and 10. Figure 10 shows that when the number of concurrent queries becomes significant, it is fundamental to increase the pool size to lower response times. In particular, we use different colors to show different times: the average time necessary to obtain a pipeline from the pool and use it to process the text in blue, the one to analyze the annotations produced and build the triples in red, and the time necessary for the communication between the client and the server in green. As one can see, the times that improved are the red and blue, while the communication time that depends on the length of the sentence and the quality of the wireless cellular link introduces a (typically long) constant threshold. Even with a limited pool size of 10, the backend is always able to complete the semantic-enrichment process within 300ms and to reply within 2s, thus confirming the usability of the proposed solution for social workers.

Finally, we assessed the performance of our autoscaling support by overloading it with 500 concurrent requests per second. During these tests, we monitored CPU usage for different application instances for the upscale situation as in Figure 11; we obtained similar results, not shown here due to space limitation, for downscale. We set the CPU threshold to 50% on the DEA. At the beginning there is one only active application instance, then, at second 43s, after sampling an overload situation for some times (to avoid triggering

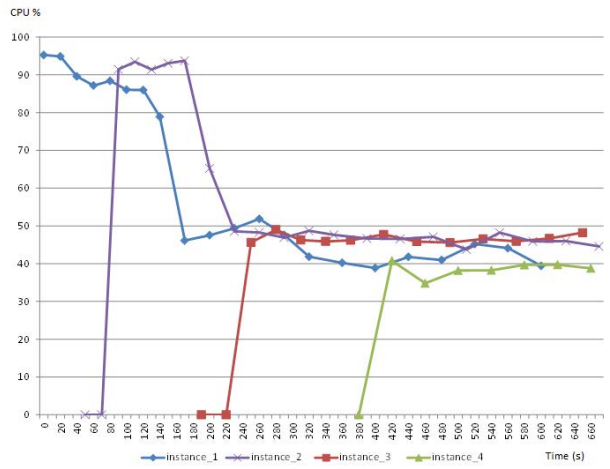


Fig. 11. MoSSCa support and CPU load redistribution with autoscaling.

autoscaling actions too aggressively, especially in presence of short load peaks, DEA waits for 3 multiple consecutive alarms for 3 consecutive periodic samples, every 10 seconds), the DEA triggers the autoscaling process and a second instance is activated and incoming load is redistributed; the same situation occurs, respectively, at seconds 183s and 373s. As regards the duration of the upscale operation, it depends on the actions the Cloud Controller has to take. When the droplet is already present at the DEA node, the application instance starting process is typically very fast and requiring always less than 5 seconds (this is the case in the experimental results shown in Figure 11). When the droplet has to be downloaded through the standard Cloud Foundry push operation, instead, more time is needed, typically up to 50 seconds for the download of all the code and related ontologies and all related LRs. Finally, some time is also required by the GoRouter to configure and apply the routing rule to direct traffic in a round robin fashion also to the new application instance; that is the reason why, after the startup, the application instance CPU load remains low for a certain time span. In any case, our support is able to automatically scale the backend by granting correct and prompt processing times, always below 300ms, and obtained autoscale provisioning times are compatible with the semantic-enriched speech recognition requirements.

7 Related Work

In the last years several projects have been carried out that address the different issues involved by automatic speech recognition, such as speech-into-text translation, text annotation, text comparison, storage, and searching. However, to the best of our knowledge, no comprehensive infrastructures supporting all the functionalities of MoSSCa and targeted at mobile devices have been proposed. The majority of solutions differ mainly on the annotation technique adopted and on the source of the text that can be either multimedia files, simple text files or text obtained from speech recognition. For that reason and because text annotation is also a fundamental management aspect for an automatic speech recognition system, we focus the state of art on available annotation solutions. In particular, we restrict the focus to semantic-based text annotation solutions that have proved their effectiveness for texts coming from various sources, while for an extensive survey on approaches and tools for ontology-based information extraction, we refer interested readers to [24].

In particular, we will first discuss the very few projects exploiting ontology-based annotation techniques for texts obtained from speech recognition and then the annotation solutions for multimedia files. We will also present, at the end of the section, solutions for textual files that, in our opinion, provide useful insights for research in the speech recognition field.

One interesting use of an ontology-based annotator that works on speech recognition is proposed in [25]. This project focuses on the problem of choosing the best hypothesis generated by a tool performing speech recognition. For that purpose they present an algorithm that uses an annotator to calculate a score representing the semantic coherence of text recognized from speech. The algorithm uses a graph representation of an ontology to annotate concepts identified in the text and analyse their dependencies. This is done through the is-a hierarchy representation of the ontology and it is enhanced by parent relationships, so scores evaluate distances between concepts recognized in the graph, and these scores can weight the different matches produced by the speech recognition tool and to choose the most appropriate one.

An annotator whose scope is to provide high level information for audio files analysing the speech recognized in them is proposed in [26]. Keywords are recognized from a vocal source and returned as a list with the related confidence. These entities are then associated to the files to allow information retrieval based on content. In particular, they develop an automatic content annotation system that works on the assumptions that the Knowledge Base (KB) of several domains is described by ontologies. In addition, there is a keyword spotting system trained with some pre-annotated files through phonetic decoding and proper keyword spotting.

An annotator for multimedia files (audio/video) is proposed in [27] where annotations are organized in multiple and sequential tiers, and the final one encloses the annotations derived from an ontology called General Multimedia Ontology. Differently from previous one, it does not provide a automatic annotation system but it is up to the user to annotate manually the content of the files choosing the opportune entities from the ontology.

Interesting annotation solutions for textual files include the proposals described in [28] and [29]. In [28] authors considered NLP process, like the POS annotation, as the task of choosing the appropriate tag word from an ontology of categories. Their framework focuses on the annotation of anaphoric relations, words whose meaning depends on previous one in the text. In particular, the solution analysed permits users to choose a document, load an ontology and then selecting manually part of the text annotate the corpus with its entities. Therefore instance of a certain concept can be annotated with grammatical information. When users annotates an entity, they can specify whether it refers to a set of concepts previously annotated proposed by the framework. In this way, erroneous annotations of relations

TABLE 1
Comparison of semantic-enriched text analysis solutions

Reference	Type	Relation-Aware	Annotatic	Corpus
[25]	Automatic	No	No	TASR
[26]	Automatic	No	No	A
[27]	Manual	No	Yes	A/V
[28]	Manual	No	No	T
[29]	Automatic	No	Yes	T
[30]	Automatic	Yes	No	T

are excluded.

On the other hand, [29] provides an automatic solution for annotating text corpus with ontologies: it developed a plugin for the framework GATE, that will be described for its influence on our work. This plugin called OWLExporter allows to analyse text corpus and generate entities based on an ontology that are used to populate the ontology itself. That is done through the use of two ontologies: one representing the domain and one that is used for NLP analysis.

This type of process is usually called Ontology Learning and Population (OL&P) and an approach is in [30], where the problem of Knowledge Extraction from text is faced by adaptation and reusing Boxer, a linguistic tool based on Discourse Representation Theory (DRT) that generates formal semantic representation of text based on event semantics. Since this tool produces a logical form from natural language that is not compliant with the usual formats of Semantic Web Data, a set of rules for transforming Boxer output to OWL/RDF ontologies is designed.

The remainder of this section summaries the main common characteristics observed in the previously analysed ontology-based annotators as shown in Table 1.

The first fundamental characteristic (*Type* in Table 1) is the capability of providing *annotations automatically* or relying *on manual ones* inserted by the user. Beyond the difference of quickness of the two different methods, its importance is stressed in [31] that reports the experience with a manual annotator ontology based for web pages. During the experiment among 30 users initially willing to provide information about their web pages just 15 succeeded and a lot of annotations inserted were syntactically incorrect. The second characteristic (*Relation-Aware*) is represented by the capability of recognizing just *single concepts* in the corpus or discovering even *relations connecting them*. The third characteristic (*Annotations*) is the capability to support the building of ontology annotations based on *previous annotations of other nature* (typically NLP annotations). The fourth and last characteristic (*Corpus*) report the *type of the corpus annotated*, such as (*T*)ext, Text from ASR (*TASN*), (*A*)udio, and (*V*)ideo.

8 Conclusions and Future Work

The exploitation of the Mobile Cloud Computing paradigm can leverage the design and deployment of several mobile applications whose potential is currently limited by the strict constraints on mobile devices. Mobile speech recognition represents a significant mobile application example that can take advantage of the Mobile Cloud to extend its functionalities by offloading intensive memory and computation tasks to the cloud. This paper proposes MoSSCa as a novel mobile cloud-enabled speech recognition framework for social care delivery that can provide semantic-enriched text recognition, hardly feasible on mobile devices without a mobile cloud support architecture. In particular, MoSSCa exploits the mobile cloud infrastructure to enrich text obtained from the speech with semantic content and on the basis of semantic annotations to allow mobile devices to interpret and reason about the meaning of the text and the context the text relates to. Social workers can obtain with MoSSCa a more reliable, effective, and device's battery-saving experience that can reduce their diffidence in relying on speech recognition instead of taking notes. An extensive set of experiments have been conducted that prove the semantic correctness, performance, and scalability of MoSSCa.

This work paves the way to a new generation of mobile speech recognition systems able to provide not only syntactic but also semantic-based text recognition in several application domains. We have demonstrated the effectiveness of MoSCCa for the Health and Social Care domain, but our proposed framework can be considered a general purpose mobile cloud-enabled speech recognition system. In fact, the knowledge bases and the main settings are easily suitable for different use cases and MoSSCa adoption could then be very useful in a large variety of different situations not related to the Social Care specific field.

Encouraged by initial results, we are currently working along some principal directions. In order to use the support in real Social Care situations it is necessary to ascertain the security of patient data now contained in the JSON store, and possibly replicated on the network. Depending on the country where the application will be used, there are very different severe rules to protect personal health data. Moreover, other improvements can derive from increasing the coverage of the knowledge bases adopted. That is possible, by extending the ontologies, to search synonyms for both the subjects and objects especially with different knowledge bases. Last but not least, the system now works for English, but it could be extended to other languages.

9 Acknowledgments

We want to thank the European Commission for cofounding the FP7 Large-scale Integrating Project (IP) Mobile Cloud Networking (MCN) project (grant agreement no. 318109), the CIRI ICT, Center for ICT technology transfer of the University of Bologna (grant POR FESR Emilia-Romagna 2007-2013), and the M.Sc. student Leo Gioia for his help in the implementation of the mobile-only prototype.

References

[1] X. Jin and Y.-K. Kwo, "Cloud assisted p2p media streaming for bandwidth constrained mobile subscribers," *16th International Conference on Parallel and Distributed Systems*, pp. 800–806, 2010.

[2] L. Li, X. Li, S. Youxia, and L. Wen, "Research on mobile multimedia broadcasting service integration based on cloud computing," *International Conference on Multimedia Technology (ICMT)*, 2010.

[3] Z. Yang, I. S. Kamata, and A. A., "Nir: Content based image retrieval on cloud computing," *Intelligent Computing and Intelligent Systems*, pp. 556–559, 2009.

[4] J. Oberheide, K. Veeraghavan, E. Cooke, J. Flinn, and J. F., "Virtualized in-cloud security services for mobile devices," *Proc. Workshop on Virtualization in Mobile Computing*, 2008.

[5] R. Onie, P. Farmer, and H. Behforouz, "Realigning health with care," *Stanford Social Innovation Review*, vol. 10, pp. 28–35, 2012.

[6] S. Cowden and A. Pullen-Sansfacon, *The ethical foundations of social work*. Routledge, 2014.

[7] C. Rolim, F. Koch, C. Westphall, J. Werner, A. Fracalossi, and G. Salvador, "A cloud computing solution for patient's data collection in health care institutions," *Proceedings of the 2nd International Conference on eHealth, Telemedicine, and Social Medicine*, February 2010.

[8] M. Nkosi and F. Mekuria, "Cloud computing for enhanced mobile health applications," *Proceedings of the 2010 IEEE 2nd International Conference on Cloud Computing Technology and Science*, November 2010.

[9] G. Subrahmanya, V. Rao, K. Sundararaman, and J. Parthasarathi, "Dhatri – a pervasive cloud initiative for primary healthcare services," *Proceedings of the 2010 14th International Conference on Intelligence in Next Generation Networks (ICIN)*, October 2014.

[10] V. Koufi, F. Malamateniou, and G. Vassilacopoulos, "Ubiquitous access to cloud emergency medical services." *Proceedings of the 2010 10th IEEE International Conference on Information Technology and Applications in Biomedicine (ITAB)*, November 2010.

[11] D. B. Hoang and L. Chen, "Mobile cloud for assistive healthcare (mocash)," *inServices Computing Conference (APSCC)*, pp. 325–332, 2010.

[12] J. Wan, C. Zou, S. Ullah, C.-F. Lai, M. Zhou, and X. Wang, "Cloud-enabled wireless body area networks for pervasive healthcare," *IEEE Network*, vol. 5, pp. 56–61, September 2013.

[13] W. Xiaoliang, G. Qiong, B. Liu, Z. Jin, and Y. Chen, "Enabling smart personalized healthcare: A hybrid mobile-cloud approach for ecg telemonitoring," *Biomedical and Health Informatics, IEEE Journal*, vol. 18, pp. 739–745, May 2014.

[14] H. Cunningham, V. Tablan, A. Roberts, and K. Bontcheva, "Getting more out of biomedical documents with gate's full lifecycle open source text analytics," *PLOS Computational Biology*, 2013.

[15] M.-C. de Marneffe, B. MacCartney, and C. D. Manning, "Generating typed dependency parses from phrase structure trees," in *LREC*, 2006.

[16] W. A. Kibbe, C. Arze, V. Felix, E. Mitraka, E. Bolton, G. Fu, C. J. Mungall, J. X. Binder, J. Malone, D. Vasant, H. E. Parkinson, and L. M. Schriml, "Disease ontology 2015 update: an expanded and updated database of human diseases for linking biomedical knowledge through disease data." *Nucleic Acids Research*, vol. 43, pp. 1071–1078, 2015.

[17] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data-the story so far," *International Journal IJSWIS*, vol. 5, no. 3, pp. 1–22, 2009.

[18] E. Prud'hommeaux and A. Seaborne, "Sparql query language for rdf," http://www.w3.org/TR/rdf-sparql-query/, Jan. 2008.

[19] "Cloud foundry," http://www.cloudfoundry.org/about/index.html.

[20] "Nats messaging (nats)," http://docs.cloudfoundry.org/concepts/architecture/messaging-nats.html.

[21] J. Povedano-Molina, J. M. Lopez-Vega, J. M. Lopez-Soler, A. Corradi, and L. Foschini, "Dargos: A highly adaptable and scalable monitoring architecture for multi-tenant clouds," *Future Generation Computer Systems*, vol. 29, no. 8, pp. 2041–2056, 2013.

[22] A. Portosa, M. M. Rafique, S. Kotoulas, L. Foschini, and A. Corradi, "Heterogeneous cloud systems monitoring using semantic and linked data technologies," *Proceedings of the 14th IEEE/IFIP International Symposium on Integrated Network Managementon*, May 2015.

[23] "Zabbix: An enterprise-class open source distributed monitoring solution," http://www.zabbix.com/.

[24] D. C. Wimalasuriya and D. Dou, "Ontology-based information extraction: An introduction and a survey of current approaches," *Journal of Information Science*, vol. 36, no. 6, pp. 306–323, 2010.

[25] I. Gurevych, R. Malaka, R. Porzel, and H.-P. Zorn, "Semantic coherence scoring using an ontology," *NAACL '03 Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, vol. 1, pp. 9–16, 2003.

[26] J. Tejedor, R. Garcia, M. Fernandez, F. J. Lopez-Colino, F. Perdrix, J. A. Macias, R. M. Gil, M. Oliva, D. Moya, J. Colas, and P. Castells, "Ontology-based retrieval of human speech," *Database and Expert Systems Applications, 2007. DEXA '07. 18th International Workshop on*, pp. 485–489, September 2007.

[27] Y. D. Artem Chebotko, "An ontology-based multimedia annotator for the semantic web of language engineering," *International Journal on Semantic Web and Information Systems (IJSWIS)*, vol. 1, 2005.

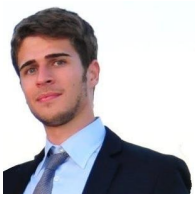
[28] S. H. Philipp Cimiano, "Ontology based linguistic annotation," *3 Proceedings of the ACL 2003 workshop on Linguistic annotation: getting the model right*, vol. 19, pp. 14–21, 2003.

[29] R. Witte, N. Khamis, and J. Rilling, "Flexible ontology population from text: The owlexporter," *International Conference on Language Resources and Evaluation (LREC)*, pp. 3845–3850, May 2010.

[30] F. Draicchio, A. Gangemi, V. Presutti, and A. G. Nuzzolese, "Fred: From natural language text to rdf and owl in one click," *The Semantic Web: ESWC 2013 Satellite Events*, vol. 7955, pp. 263–267, 2013.

[31] M. Erdmann, A. Maedche, H. P. Schnurr, and S. Staab, "From manual to semi-automatic semantic annotation: About ontology-based text annotation tools," *Proceedings of the Coling 2000 workshop on semantic annotation and intelligent content*, August 2000.

Antonio Corradi (M) graduated from University of Bologna, Italy, and received MS in electrical engineering from Cornell University, USA. He is a full professor of computer engineering at the University of Bologna. His research interests include distributed systems, pervasive and heterogeneous computing, context-aware services, network management, mobile agent platforms. He is member of IEEE, ACM, and Italian Association for Computing (AICA).



Marco Destro graduated with honors in Computer Engineering at the University of Bologna, Italy. He spent 6 months in the IBM Smarter Cities Research Laboratory developing his master thesis. His main interests include Semantic Web, Mobile Development, Distributed computing and Data Mining.

Luca Foschini (M) graduated from University of Bologna, Italy, where he received PhD degree in computer science engineering in 2007. He is now an assistant professor of computer engineering at the University of Bologna. His interests include pervasive computing environments, system and service management, context-aware services, and management of Cloud computing systems. He is member of IEEE and ACM.

Spyros Kotoulas (M) is a Research Scientist at the Smarter Cities Technology Center (SCTC) working on large-scale cataloguing, processing and integration of semi-structured data. His research interests lie in data management for semi-structured data, Semantic Web, Linked Data, reasoning with Web data, flexible data integration methods, stream processing, peer-to-peer and other distributed systems.

Vanessa Lopez (M) graduated with a degree in computer engineer from the Technical University of Madrid (UPM) and received a PhD degree from KMi (Open University) where she was a researcher at KMi (Open University) from 2003. Since 2012, she is a researcher at IBM Research Ireland. Her research interests include Linked Data technologies to support data integration and solutions for harnessing urban and web data as city knowledge.

Rebecca Montanari (M) graduated from the University of Bologna, Italy, where he received a Ph.D. degree in computer science engineering in 2001. She is now an associate professor at the University of Bologna. Her research interests include policy-based networking and systems/service management, context-aware service management, security management mechanisms, and tools in both traditional and mobile systems.