Engineering Applications of Artificial Intelligence 77 (2019) 70–85

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

PAVAL: A location-aware virtual personal assistant for retrieving geolocated points of interest and location-based services

Lorenzo Massai, Paolo Nesi *, Gianni Pantaleo

University of Florence, Department of Information Engineering, Distributed Systems and Internet Tech lab DISIT Lab, Italy[1]

ARTICLE INFO

Keywords:
Virtual personal assistants
Location-aware recommender systems
Natural language processing
User-intent detection
Semantic web technologies
Geographic information retrieval
Geoparsing
Geocoding

ABSTRACT

Today most of the users on the move require contextualized local and georeferenced information. Several solutions aim to meet these trends, thus assisting users and satisfying their needs and preferences, such as virtual assistants and Location-Aware Recommender Systems (LARS), both in commercial and research literature. However, general purpose virtual assistants usually have to manage large domains, dealing with big amounts of data and online resources, losing focus on more specific requirements and local information. On the other hand, traditional recommender systems are based on filtering techniques and contextual knowledge, and they usually do not rely on Natural Language Processing (NLP) features on users' queries, which are useful to understand and contextualize users' necessities on the spot. Therefore, comprehending the actual users' information needs and other key information that can be included in the user query, such as geographical references, is a challenging task which is not yet fully accomplished by current state-of-the-art solutions. In this paper, we propose *Paval* (Location-Aware Virtual Personal Assistant [2]), a semantic assisting engine for suggesting local points of interest (POIs) and services by analyzing users' natural language queries, in order to estimate the information need and potential geographic references expressed by the users. The system exploits NLP and semantic techniques providing as output recommendations on local geolocated POIs and services which best match the users' requests, retrieved by querying our semantic Km4City Knowledge Base. The proposed system is validated against the most popular virtual assistants, such as Google Assistant, Apple Siri and Microsoft Cortana, focusing the assessment on the request of geolocated POIs and services, showing very promising capabilities in successfully estimating the users' information needs and multiple geographic references.

1. Introduction

The recent rapid and growing diffusion of mobile devices and ICT solutions has generated an increasing demand for retrieving specific information on local services in order to fulfill everyday users' information needs. For instance, retrieving information on points of interest (POIs) like local food and drinking, accommodations, events, shopping spots, entertainment, commercial and cultural activities, tourism attractions etc., as well as public administrations and institutional facilities (public transportation, healthcare etc.) has become a common information demand, especially on the move. Therefore, users' needs and requirements are increasingly moving towards Location Based Services (LBS), experiencing a mobile environment which is often characterized by dynamic and contextual information demands (Kumar, 2011). Quite recent studies reported that about 25% of Web searches have local intents (Palacio et al., 2015), and almost 20% of search queries contain geospatial and temporal references, in addition to information related to the search topic (Palacio et al., 2010). These percentages commonly increase when we consider queries performed on the move.

Virtual Personal Assistants (VPA) are designed with the aim of simplifying and improving our way to retrieve POIs, web resources and help managing some daily activities by simply posing natural language queries to intelligent agents. Examples of interaction with a personal assistant are, for instance, asking how to find and reach specific places, search and attend events, manage scheduled activities, receive suggestions and recommendations, provide decision support, interact with social media and commercial vendors and services, etc. (Campagna et al., 2017). In literature, a number of personal assistants and, more specifically, systems addressing the recommendation of POIs from Natural Language Processing (NLP) statements and requests have been produced. Some of them are from the industry such as Google Assistant, Apple Siri, Microsoft Cortana, IBM Watson (Ferrucci et al., 2010) and

* Corresponding author.
E-mail address: paolo.nesi@unifi.it (P. Nesi).

[1] http://www.disit.org, http://www.sii-mobility.org, http://www.km4city.org.

[2] The name Paval is chosen as a permutation of the initials of "Location-aware virtual personal assistant".

https://doi.org/10.1016/j.engappai.2018.09.013
Received 15 January 2018; Received in revised form 7 June 2018; Accepted 16 September 2018
Available online xxxx

Amazon Alexa. Some research efforts have been also made to propose intelligent agents and assistant solutions for narrower domains, such as healthcare, education, entertainment and tourism. On the other hand, though the above-mentioned tools from the industry have access to huge amounts of data, the capabilities for the interpretation of the user needs are often limited. Actually, most of these solutions, although employing NLP and semantic search technologies (Kumar and Reddy, 2017; Nickel et al., 2016) do not provide yet a semantic with a sufficient degree of expressiveness, supporting queries with limited complexity and describing only the most common entity types (Uyar and Aliyu, 2015). Therefore, the above-mentioned systems are not always able to understand the real information need or geographic intent expressed in natural language queries, unless its meaning is explicit. Therefore, the answers provided by current state of the art assistants are frequently a generic resource, such as web pages which are only partially related to the keywords extracted from the text query, thus not always supplying geolocated results. Moreover, indexing and dealing with very large amounts of data (both geographic and descriptive) may also lead to a loss in precision rate for local resources retrieval (Nesi et al., 2016).

Recommender systems provide custom suggestions based on filtering techniques on many different domains, topics and items, although they are typically not designed to directly respond to natural language queries. Only in recent years traditional recommender systems started to take into account multiple dimensions, considering spatial dimension and, specifically, local geographic information as a relevant aspect in the users' information needs. Actually, although there already exists a huge amount of georeferenced data, users are usually interested only in local, nearby contexts and resources (Rodríguez-Hernández et al., 2015). The inclusion of the spatial dimension in recommendation systems allows to obtain more effective suggestions, leading to a quite new application field called Location-Aware Recommender Systems (LARS). LARS applications take into account the spatial properties (locations) of users and/or items.

Geolocated data and geographic entities play an important role in most of our daily activities. Geographic-aware search is especially important for location-based services where a user in a mobile environment might have dynamic and contextual information demands (Kumar, 2011). For this purpose, many solutions for Geographic Information Retrieval (GIR) have been investigated and proposed. This area of Information Retrieval (IR) deals typically with unstructured textual data, exploiting NLP and semantic based techniques for geographic entities extraction (Buscaldi et al., 2006; Nesi et al., 2016), and disambiguation (Buscaldi and Rosso, 2008). Moreover, in order to go beyond what is typically covered in current GIR solutions, it is important to extract explicit geographic information as well as to estimate users' implicit geographical needs. Actually, from past research there is the evidence that only about 50% of queries expressing a geographical intent or need (i.e., queries where the users expect geolocated results) contains explicit location names (Welch and Cho, 2008). Besides, among all the potential named entities or unstructured natural language data which may contain implicit geographic information, different localization capabilities can be addressed. For instance, queries containing keywords like "restaurant" or "cinema" usually imply local, nearby information needs (so that the users' contextual information, e.g., the GPS position, IP location, profile data etc., can be used to better focus on specific local requirements), while other keywords like "hotel" or "highway" do not necessary mean a demand for local resources, so that they may contain an implicit geographic intent with weaker localization capabilities (Yi et al., 2009).

1.1. Related work

Our review of the state of the art is focused on three distinct and related research areas: VPA, LARS and GIR, as defined above.

1.1.1. Virtual personal assistants

In current literature, there is still lack of virtual agents and assistants which fully respond to queries and execute actions and commands in natural language, whereas an effort to build a comprehensive cross-domain virtual assistant has been made by Campagna et al. (2017) with the Almond project, based on Thingpedia, a crowdsourced public Knowledge Base accessing open APIs, Internet of Things and natural language interfaces. Personal assistants may provide a wide range of services and fulfill a high variety of tasks, based on user inputs, on location and other sensors-based information, as well as on the ability to access information from online resources (Madhusudhanan and Subramaniyan, 2016). This approach suggests the development and implementation of modular architectures for this kind of complex systems, integrating different aspects and functionalities such as natural language interfaces for human–computer interaction (Matsuyama et al., 2016), web based search engines, geographic location-awareness, social media management, data analytics and statistical frameworks, semantic technologies (Bellandi et al., 2012), inference and reasoning, decision support (Heredero et al., 2013) and recommendation features (Tavčar, 2016). The most important IT companies have proposed their own solutions, such as Google Assistant by Google, Siri by Apple, Cortana by Microsoft, Watson by IBM and Alexa by Amazon; besides, other open source solutions exist, like Mycroft and Lucida. These tools usually index huge amounts of data (both geographic and descriptive) which, although covering a wide range of domains, may lead to a degradation of precision rates evaluation for local resources retrieval. Research efforts have been recently made to propose intelligent agents and assistant solutions for more restricted and specific domains, such as education (Harvey et al., 2015), entertainment (Gordon and Breazeal, 2015), healthcare (Ahamed et al., 2006) and tourism-based services (Tavčar, 2016). In most of the cases, these systems are not able to understand the meaning of complex phrases, as well as sentences with tacit meaning underneath, and most of the times they provide general and not always geolocated resources as results, such as web pages, behaving like traditional web search engines. To estimate the purpose of a web request, an intent classification from query log analysis is analyzed by Broder (2002) and further refined by Rose and Levinson (2004) which have provided a classification into *navigational*, *informational* and *transactional* queries. According to Jansen et al. (2008) it is also possible to classify 70%–80% of the queries in one of the above categories, with a high degree of confidence, considering non-multiple-intent queries only.

1.1.2. Location-aware recommender systems

Traditional recommender systems usually do not implement NLP features, and they are typically classified into *collaborative filtering*, *content-based* and *knowledge-based*. Collaborative filtering recommender systems provide suggestions and predictions of what a user needs or likes based on the similarity among his/her actions, preferences and feedback with the ones of other users (Bhagwani et al., 2016). Content-based recommender systems generate recommendations and best-matching items based on users' past experiences and preferences (Lops et al., 2010), without involving other users' contextual information. Knowledge-based recommender systems produce advices according to external knowledge resources, users' preferences and the characteristics of the required items or services (Husain and Dih, 2012). According to the literature review, only in the last few years we are assisting to an extension of these tools towards LARS, allowing the production of suggestions which are more focused on meeting users' local needs, thanks to the implementation of specific location-aware features (Noguera et al., 2012). LARS approaches also may often be collaborative, allowing users to submit location-based ratings (Mathur and Bairagee, 2016). Recommender systems providing location-based features have been proposed for several specific domains. In the e-Tourism field Clements et al. (2011) present a solution that suggest touristic POIs and travel plans on the basis of the users' visiting history, exploiting a location similarity model among different locations to plan

a visit to a new place. It uses a set of geotags to measure similarity among locations and has been evaluated only at country and city scale. In the field of e-commerce, Yang et al. (Yang et al., 2008) implemented a LARS for mobile shopping which provide recommendations of vendors that are in the user's neighborhood. CityVoyager (Takeuchi and Sugimoto, 2006) is a recommendation system for local shops based on user's GPS location history. The production of suggestions is based on locations of the shops previously visited by each user. All these systems produce suggestions taking into account profile information, as well as preferences, ratings, feedbacks, interactions with other users etc.; however, this may not always be sufficient to accomplish a generic user's need at the moment, which require a deeper investigation that may be provided, for instance, by NLP based analysis of users' queries.

1.1.3. Geographic information retrieval

LBS provide relevant information, suggestions and recommendations according to the user's current location using geospatial information (GPS position) and intelligent applications (Husain and Dih, 2012), relying usually on GIR solutions. Any typical GIR approach has the goal of identifying and unambiguously associating toponyms extracted from text with geographic locations, being also capable of dealing with Word Sense Disambiguation (WSD) (Palacio et al., 2015). One of the main current gaps in the state of the art which we aim to contribute with the system proposed in this paper is the ability to reliably understand and extract geographic references from an unstructured user query (e.g.: the question "*Where can I find a restaurant near to Baker Street?*" contains a geographic reference to the London metropolitan area, in the UK). Correlated to this issue is the handling of different levels of geographical intents, often conveyed in the expressiveness of natural language (for instance, in a query like "*Which is the nearest bus stop to Piazza San Marco, in the central district of Florence, in Italy?*"). Some efforts have been previously made by Yi et al. (2009), who use language modeling to determine the implicit locations and geographic references in queries, considering only geographic granularity at city-level. Moreover, the GeoCLEF community has proposed, since 2008, a geo-query parsing and classification forum (Mandl et al., 2008), with the aim of retrieving explicit geographic references in users' queries, providing also a classification of query types in order to better contextualize users' natural information needs.

The study of the current state of the art highlights the lack of systems exploiting a deep and unambiguous understanding of the semantics of the user's query, either for simple and complex sentences, more conveniently without exploiting user profiling (which may present drawbacks such as the cold start problem, (Hossein et al., 2014)), besides providing a level of information detail that is sufficiently specific for the purposes intended in this paper. Also, according to the direction of LARS systems on meeting users' local needs, a mechanism should be provided for reliably retrieving geographic data at multiple levels of geospatial resolution from the user query.

1.2. Aim of the paper

The system proposed in this paper is Paval (available at https://paval.disit.org/Paval/), a location-aware virtual personal assistant focused on suggesting local POIs and services. Paval is intended to be used mainly on the move, e.g., while driving or in a situation of emergency, giving the user the ability to express an actual need neither knowing what kind of service he/she needs, nor how to reach it (see Fig. 1a). Paval aims at understanding natural language queries exhibiting a generic everyday information need, with the goal of estimating the most suitable kind of service to suggest and to recognize potential geographical references (see Fig. 1b). In order to better contextualize the suggestion of results, Paval focuses on the users' specific local needs. The system provides as output a list of suggested geolocated POIs and/or services which best match the estimated user's necessity (see Fig. 1c). Local POIs and services are retrieved exploiting the Km4City Smart City Knowledge Base (Bellini et al., 2014). The Km4City semantic RDF repository has been designed and implemented at DISIT Lab by integrating open and private data by local Public Administrations of the Florence Municipality and the Tuscany Area, in Italy, together with several heterogeneous kinds of historical and real-time data related to Smart City areas. Some of the areas covered by the Km4City KB are public and private transports, POIs, commercial activities and services, events, public administration and healthcare facilities, detailed local toponyms (streets, roads, squares etc.), as well as several data types provided by sensors, e.g.: air quality monitoring, traffic density, public parking lots status, etc. The Km4City Knowledge Base does not index an amount of data as huge as those handled by some of the major IT companies above mentioned, anyway it provides a very accurate local data coverage and expressiveness (Nesi et al., 2016), also thanks to careful reconciliation and quality improvement on indexed data. The Km4City ontology is also used in a preliminary phase to the execution of Paval for building a semi-supervised custom reference corpus constituted by lists of words semantically related to a subset of the Km4City taxonomy, describing a wide range of local POIs and services. The obtained corpus represents a focused-domain semantic resource which is used by the system as a reference for classifying keywords and keyphrases extracted from the input textual query, in order to estimate the user information need. In this sense, our approach is different from other semantic frameworks based on external knowledge, which rely mainly on reference or training corpora generated from full dumps of online general-purpose and non-specialized resources such as Wikipedia or Dbpedia, often dealing with a high level of noise, unresolved ambiguities etc. The proposed system aims at going beyond the current state of the art by understanding the semantic context and latent user intents expressed in the user query, as well as handling multiple levels of geographic references.

The main aim of the present paper is to answer the questions posed in Fig. 1b through the Paval engine. To our knowledge, after a review of the current state of the art, the main original contributions of our system are the following:

- Understanding the users' query at level of information needs and geographic intent, without relying on user profiling.
- Comprehension of multiple levels of geographic intents, estimating if the input query expresses a user's need in the neighborhoods of the user's position, or instead in proximity of other geolocated references recognized and extracted by the system from query text.
- Use of a custom reference corpus to classify relevant keywords and related synsets extracted from natural language queries, following the semantic hierarchy of the Km4City Knowledge Base, which includes a taxonomy for classes describing city elements, POIs, commercial activities and services, attractions etc.
- Word Sense and context disambiguation based on different weighting of relevant keywords (extracted from the input query) with respect to their Part-Of-Speech (POS), and on a semantic relatedness score assigned to reference corpus terms on the basis of term occurrences extracted from descriptive data contained within the Km4City Knowledge Base.
- Use of the local Km4City Smart City Knowledge Base to retrieve geolocated items (POIs, services etc.) to be provided as results.

The rest of the paper is organized as follows: Section 2 illustrates the functional architecture of the proposed system; in Section 3, the performance analysis is reported together with the validation of the system and the user front-end for validation; finally, Section 4 reports conclusions.

2. External knowledge and system architecture

The architecture of the proposed system exploits NLP and semantic technologies, aiming at understanding the meaning of the user intent, taking into account the possible presence of multiple geographical references and thus estimating if the information demand is intended

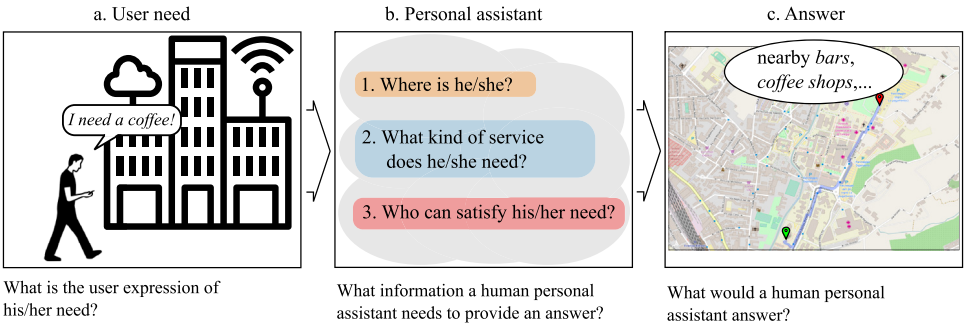

Fig. 1. Fulfillment of an everyday user's information need.

in the neighborhoods of the user, or instead in proximity of other places or geolocated items. Hence, the system input is represented by the user's natural language query and, if allowed, by the user's GPS position. The input can be specified and submitted to the system through a web user interface which implements the basic servlet functionalities and forwards it to the inner processing modules. The detection of geographical references uses NLP techniques for geographic entity parsing (geoparsing), while the retrieval of geolocated data, i.e. the association of a latitude and longitude tuple for each extracted geographic item (geocoding), is performed by querying the semantic Km4City repository, exploiting our Km4City Smart City API (Badii et al., 2017). A list of retrieved POIs and services, which are estimated to best meet and satisfy the user's needs, is provided as output.

2.1. External knowledge and resources

The system exploits the following external knowledge and third-party tools for NLP and semantic expansion-based tasks:

- the Km4City ontology model and data (Bellini et al., 2014) accessed through the Km4City Smart City API (Badii et al., 2017);
- the GATE framework (Cunningham et al., 2002), in particular the ANNIE plugin for natural language analysis, mainly POS-tagging and pattern matching for extracting geographic entities;
- the semantic network Babelnet (Navigli and Ponzetto, 2012) to provide query expansion in the Service extraction phase;
- Wordnet (Fellbaum, 1998), accessed by the Datamuse API interface (Datamuse API), to generate the lists of semantically related terms in the Reference Corpus Generation phase;
- Google Geolocation APIs to provide user GPS geolocalization;
- the Yandex online translator APIs (Yandex).

In addition, the Reference Corpus Generation module is designed and realized (as described in Section 2.1.1) as an original contribution to produce the Target Expansion lists, a reference corpus which is used by the Service Extraction module (Section 2.2.1). The generation and organization of external knowledge and resources employed by Paval is shown in Fig. 2: in the gray area all the external knowledge and resources employed by the Paval architecture are represented. The resources outside the gray area are the static resources generated in the preprocessing phase (which are made available for research purpose at: http://www.disit.org/paval/pavalsources.rar). The yellow arrows denote an API call, while the blue arrows denote the data flow.

2.1.1. Reference corpus generation module (target expansion)

The proposed system firstly involves the creation of a target corpus which allows to semantically describe and contextualize a wide range of POIs and service categories. It is to be noticed that this operation is performed *una tantum*, as a training step dedicated to the creation of the semantic resources later used for the estimation of the information need expressed by the user. The result of this phase is represented by a set of lists containing keywords that are semantically related to the labels of the Km4City ontology classes representing the service categories within which are instantiated local POIs and services, commercial activities, public administration and healthcare facilities, public transportation lines and stops, cultural activities, events etc.

The strategy adopted to build the lists is the following: the module receives as input the label strings of the Km4City classes; each label is then semantically expanded by querying the Wordnet database through the Datamuse API with respect to each Wordnet semantic relation (hyponymy, hyperonymy, synonymy, antonymy, meronymy). The expanded keywords obtained are grouped together in lists. Afterwards, in order to improve robustness for the resulting expanded datasets, keywords are lemmatized, and an additional manual annotation is performed as a quality improvement (e.g., adding semantically relevant keywords obtained from external resources, such as *ad hoc* crawled web sites, which is the case of keywords extracted from restaurant menus that are added to the *Restaurant* service category list).

Using such a semi-supervised process, a set of lists is obtained, constituted by words that are semantically related to the labels of the Km4City classes describing service categories (a total of 528 classes), preserving the semantic hierarchy built-in in the Km4City taxonomy (e.g.: the Km4City *WineAndFood* class is the parent of several child classes, such as *Restaurant*, *Bar*, *SushiBar*, *TakeAway,* etc.; the class *Entertainment* is parent of child classes like *Cinema*, *Pool*, *Discotheque,* etc.). Stop words such as prepositions, conjunctions, articles and punctuation are pruned, and finally the words contained in each list are weighted by assigning them a measure of their semantic relatedness respect to the class label.

As a result, a reference corpus composed by 528 documents, corresponding to the names of the categories of the Km4City taxonomy, is produced and used as target documents where to retrieve the keywords extracted from the user's query in the Service extraction phase (see Section 2.2.1).

2.2. System architecture

The architecture of the system, which is depicted in Fig. 3, articulates into three main modules (the colors of the inner modules in Fig. 3 recall the steps of fulfillment of users' information needs, shown in Fig. 1b):

- The **Service extraction** (described in Section 2.2.1) module is devoted to the estimation of the kind of service which best matches the user's information need, based on NLP and semantic analysis of input text. To this goal, this module deals with the extraction of relevant keywords and POS tags from the user's queries, their semantic expansion and their classification against the semantic resources obtained in the Reference Corpus Generation phase.
- The **Location extraction** (Section 2.2.2) module is committed to detect potential multiple geographic references contained into the user query. If any geographic reference is found, the present module queries the Km4City repository in order to retrieve the corresponding latitude and longitude tuple.

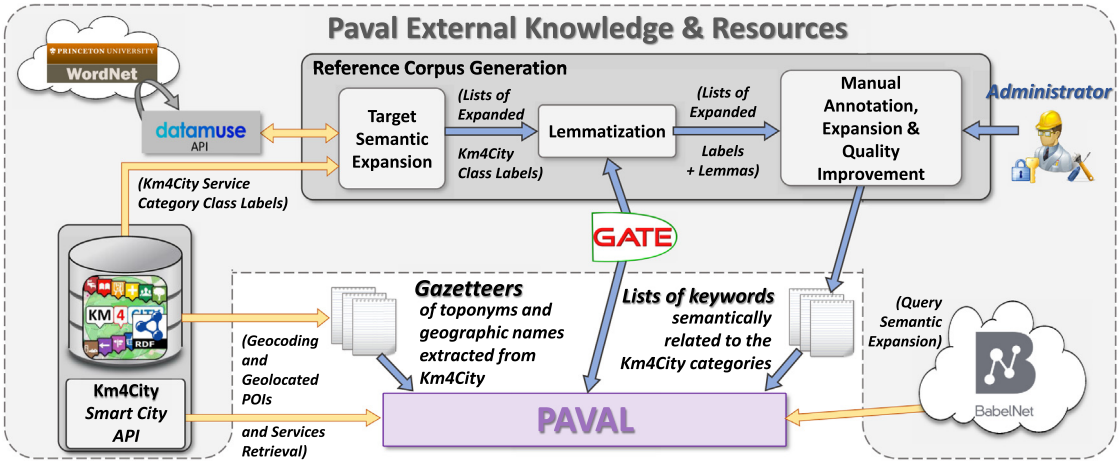

Fig. 2. Generation and organization of external knowledge and resources. . (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

- The **Geolocated data retrieval** (Section 2.2.3) module is dedicated to retrieve the actual local data (POIs, services etc.) to be provided as final results, by querying the Km4City repository on the basis of the kind of service and geographical references estimated by the previous modules. A list of POIs belonging to the estimated service category is provided, ordered by increasing distance with respect to the user's position (if present) or with respect to any geographic reference (if detected by the system).

2.2.1. *Service extraction module*

The Service extraction module is instantiated every time a query is submitted by a user. The working flow of the present module is divided into three steps:

- The first one is the **relevant keywords extraction** phase, which aims at extracting from the user phrase POS-tagged keywords which convey semantic significance to the phrase itself.
- The second step is the **query expansion and disambiguation** phase, in which each keyword extracted in the first step is expanded into a vector of semantically related words called a *synset*, and different weights are assigned to words based on their POS.
- The third step is the **election of a service as the user need**. In this step, the synsets of keywords provided in the previous step are classified against the Target Expansion lists produced in the Reference Corpus Generation module (Section 2.1.1). A majority voting algorithm is implemented to calculate and provide, for each list, a score obtained by summing the weights of all the extracted keywords and their corresponding synset terms which are found to be present in the list itself. Finally, the service category which best matches the user's need is estimated as the label of the list with the highest score.

The combination of these three phases leads to a hybrid approach which improves the retrieval based only on query expansion, providing specific target documents where to retrieve the expanded words. The problem is thus reduced to a ranking classification method, based on the weights of all the expanded keywords within the generated lists. Relevant keywords are extracted from the user query using a GATE pipeline containing the default tokenizer, a sentence splitter, ruleset and lexicon, and the ANNIE plugin which allows to define patterns and rules (through the dedicated Jape library, a Java Annotation Pattern Engine) to be extracted in the analyzed text. The Treetagger annotation tool (Schmid, 1994) is also added to the processing pipeline for POS tagging. Token identification is later used to filter relevant keywords and to associate different weights to different parts of speech in the subsequent phase, which is devoted to service category estimation. Stop words, such as prepositions, conjunctions, articles and punctuation are filtered out.

In order to enrich the expressiveness and the understanding capabilities of the whole framework, the system performs a semantic expansion of all the relevant POS-tagged keywords extracted from the input query, and then the most likely senses are chosen by a weight-based strategy, as described in the following. Such a query expansion procedure is performed exploiting the Babelnet semantic network, automatically expanding each extracted keyword into a distinct synset of semantically related words. Each term composing the synsets is also associated with a different weight, giving more relevance to the keywords originally contained in the query with respect to the other related words, as well as to verbal tokens.

Disambiguation among phrase senses is entrusted mainly by following a two-phase strategy. The first phase is performed *a priori*, in the Reference Corpus Generation phase, by assigning different weights to terms included in the Target Expansion lists based on a semantic relatedness score with respect to the label of the list containing them (Fig. 4).

The semantic score for each word is obtained by extracting the normalized frequencies of all of the target expansion words from local services names, descriptions, commercial sectors, etc. from crawled up-to-date dumps of the Km4City data containing all instances of classes describing POIs and services (Algorithm 1).

Algorithm 1 returns a list of words related to a Km4City category label and the corresponding relatedness score. Our reference corpus is constituted by these lists. A score of the semantic relatedness between a word and its domain can also be obtained using the Babelnet framework, not without some issues: one of them is mapping the Babelnet domains into the Km4City taxonomy; also, the high rate of false negatives which has been observed in our experiments by employing this solution is one of the major reasons that led us to a self-designed strategy. To manage the problem of sense disambiguation of the extracted terms, an alternative disambiguation method based on the Lesk algorithm (Lesk, 1986) has been also developed using Babelnet domains as senses; since this solution did not provide significant improvements to the overall retrieval performance and quality of the system which may justify the corresponding computational cost, it has not been included in the system architecture.

The second phase of our sense disambiguation strategy deals with a POS-based weighting method, consisting in assigning different weights to extracted keywords based on their POS. Specifically, a double weight is assigned to keywords originally extracted from the input query (with respect to the related ones contained in the semantically expanded synsets) and to verbal tokens. Actually, verbs have been found to be particularly useful estimating different action intents associated with the

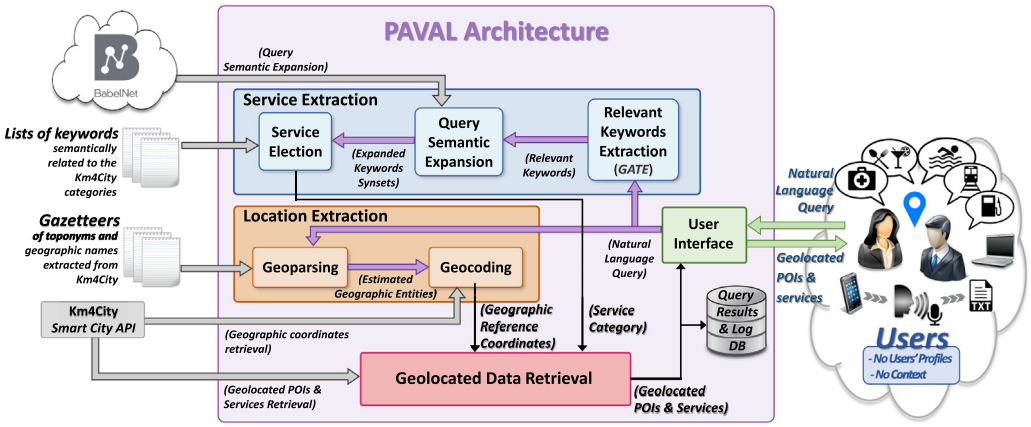

Fig. 3. Paval architecture.

Algorithm 1. generation of the semantic relatedness scores

```
for each category label[i] of Km4City taxonomy {
        add meronyms(label[i] to list[i]);
        add synonyms(label[i] to list[i]);
        add antonyms(label[i] to list[i]);
        add hyponyms(label[i] to list[i]);
        add hypernyms(label[i] to list[i]);
        for each data element k in Km4City category label[i] {
                parse Description field of element k;
                extract NOM, ADJ, VER parts of speech words;
                add extracted words to list[i];
        }
        add specific related words to list[i];
        for each row[j] of list[i] {
                add lemmaOf(word[j]) to list[i];
                merge rows and store the number of occurrences as weights;
                normalize weights between [0,1];
        }
}
```

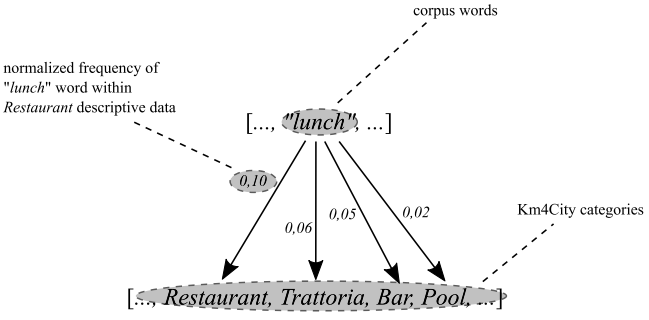

Fig. 4. Example of corpus terms weighting.

expressed information need, thus being also useful to understand and disambiguate the whole context meaning. In the case of a user phrase containing polysemic terms, as well as nouns which are related with more than one service (as a conceptual domain), the role of the verb is crucial to decide the correct kind of service.

For example, let us define *Phrase1*: "*I'd like to buy some meat*"; this is slightly different with respect to the sentence "*I'd like to eat some meat*", which we refer below as *Phrase2*. However, in terms of the service to be retrieved there is a tangible difference, because the first one most likely expresses the need for data within the *Meat and poultry* category, while the second expresses the need for data within

Table 1
Verbal token boost example for service category retrieval.

Restaurant list			*Meat and poultry* list		
Word lemmas	Normalized weight	Boosted weight (**Verbal Tokens** only)	Word lemmas	Normalized weight	Boosted weights (**Verbal Tokens** only)
Buy	**0**	**0**	Buy	**0.10**	**0.20**
Eat	**0.24**	**0.48**	Eat	**0.03**	**0.06**
Meat	0.04	0.04	Meat	0.32	0.32

the *Restaurant* category. To better clarify how the proposed system can handle these situations, let us consider the *Phrase1* and *Phrase2* examples. The details shown in Table 1 are an excerpt taken from the Target Expansion lists obtained in the Reference Corpus Generation phase for the service categories *Restaurant* and *Meat and poultry* and represent the relevant words lemmas contained in both *Phrase1* and *Phrase2* and their corresponding normalized weights (Table 1).

The normalized weight of each word relates to each list obtained through Algorithm 1: e.g. the word *meat* is found to have a normalized weight of 0.04 within the list *Restaurant*; we consider this weight as a score of the semantic relatedness between the word *meat* respect to the word *restaurant*. During the second phase the score of verbs is doubled, together with the score of the words directly extracted from the user query.

In this case, if no boosting strategy was applied to verb weights, the comparison among the classification of terms extracted from *Phrase1*

(here considered without query expansion for simplicity, without loss of generality) and the *Restaurant* list will yield a weight score sum of 0.04, while the match with the *Meat and poultry* list will have a score of 0.42, thus electing the latter list, as expected. However, considering *Phrase2,* the classification of terms extracted from *Phrase2* and the *Restaurant* list will yield a weighted score sum of 0.28, while the match with the *Meat and poultry* list will have a score of 0.35, leading to the election of the *Meat and poultry* list, which would not satisfy the user need expressed by *Phrase2*. By applying a weight boost to verbal tokens, the outcome for the first comparison is the same; however, for the latter comparison it is obtained a score of 0.52 for the *Restaurant* list, and a score of 0.38 for the *Meat and poultry* list, leading to the election of the *Restaurant* list, which is the expected outcome.

The election of the service is performed by computing the sum of the weights of all the expanded query terms that are found into the Target Expansion lists obtained in the Reference Corpus Generation phase (Section 2.1.1). The list with the highest rank (i.e., with the highest weights sum) is considered as the elected service category (represented by the corresponding class in the Km4City Knowledge Base). If more than one category has the same highest score, each one is likewise designated for the final output.

The problem of extracting the information need from the user query and mapping it to a corresponding service thus is treated as a multi-classification problem, using the service categories labels of the Km4City Knowledge Base as classes. The elected class labels are later used in conjunction with the GPS user position (if enabled) or any estimated geographical reference item (estimated as explained in Section 2.2.2) to query the Km4City Knowledge Base (see Section 2.2.3) and provide, as final result, a list of the most relevant (i.e., the nearest) instances (POIs, commercial activities and services etc.) for each elected service category.

2.2.2. Location extraction module

In addition to the estimation of the kind of service that can satisfy the user need, the system also aims at programmatically recognizing geographic references from the user phrase. This module is dedicated to extract sequences of words which may potentially represent geographical locations from the user query (geoparsing phase) and associate them to a latitude and longitude tuple (geocoding phase). These tasks are carried out by following a hybrid approach: firstly, the input text is parsed to detect location candidates. If found, such candidates are searched in the reference Km4City Knowledge Base (specifically, within the sub-graph related to toponyms including streets, roads, squares and other city elements which are detailed at street number resolution), to check whether there is an exact match or not with an actual location name. Whenever this strategy is not able to univocally recognize a geographic location name, more coarse-grained NLP techniques are applied; for instance, this may be the case of incomplete or misspelled toponyms, that may include abbreviations and acronyms (which are frequent, for example, for street names), punctuations etc. Therefore, this methodology provides a dynamic fine-to-coarse technique which uses a stronger or weaker match strategy based on the detail of the detected geographical reference. Once the geoparsing phase is completed and a location is recognized, the geocoding phase provides the corresponding geographical coordinates from the Km4City repository. The module finally outputs the tuple formed by the extracted geographic location and its corresponding latitude and longitude values.

The initial geoparsing phase is carried out by defining the following Jape rules, executed in the GATE pipeline and integrated with specifically designed logic, through which the proposed system is able to detect and manage multiple geographic user intents at different resolutions (municipality, district, street and road element, up to street number resolution):

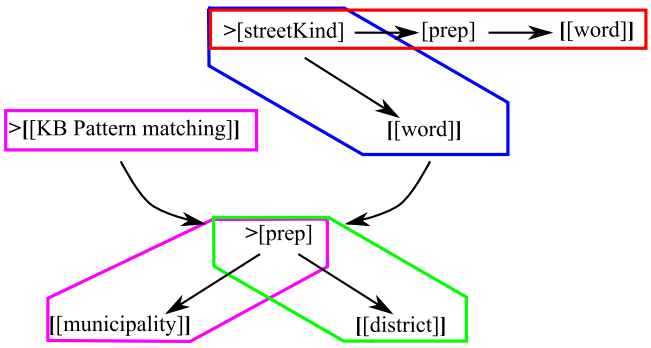

Fig. 5. Location patterns defined to estimate geographical locations.

- The *FindLocation* rule seeks for location candidates at higher spatial resolution (at district or street level), that is a generic named urban entity or city element, such as a street, a road, a square etc., looking up for an exact pattern match between a geolocation candidate, extracted from the input query text, and a gazetteer of street toponyms extracted from the Km4City repository (in particular, the regional street graph, that is the repository sub-graph including the names of all the streets, roads, squares and road elements in the Tuscany region, detailed at street number level). Custom rules for the detection of district names are also defined: in this case, a dedicated gazetteer is created, containing all districts pertaining to the Florence metropolitan area and the Tuscany region, each one provided with a manually assessed punctual latitude and longitude tuple.
- The *FindPlace* rule seeks location candidates at lower spatial resolution (at municipality and city fractions level), by looking for patterns composed by a place preposition (*at*, *in*, *on* etc.) or direction preposition (*to*, *towards* etc.) followed by the name of a Tuscany municipality, which is obtained looking up a dedicated gazetteer, extracted from the Km4City repository. A similar strategy is applied to detect fractions names.

If this detection strategy is not able to find any match, due to misspelled location names, or in case they are partially expressed (also with abbreviations, acronyms etc.) or missing, then the geographic reference is estimated by applying more coarse-grained rules and patterns. More specifically, the system aims at associating sequences of words (called *n*-grams) of the input text to the names of geolocated entities contained in the reference Knowledge Base, up to *3*-grams. In Fig. 5, all the possible evolutions of a location pattern from an initial node (>) triggering the pattern to a final node (**[[ ]]**) are depicted.

The possible initial nodes of a location pattern are **[streetKind]**, **[prep]** and **[KB Pattern matching]**. Each sequence, represented by a path from an initial node to a final node that are connected by arcs, corresponds to a different kind of location pattern, each of which is listed in the following:

- The patterns defined in the Jape rules, above described in this section, are represented in Fig. 5 by the green and the purple sequences, which may be in sequence or not.
  - The **[KB Pattern Matching]** node does not represent an actual pattern, rather it expresses the above described *FindLocation* rule, which aims at finding exact matches between candidate geographic items extracted from input query and the geographic toponyms extracted from the Km4City Knowledge Base (KB).
  - The following pattern is adopted in the above described *FindPlace* Jape rule:

    [prep] + **[municipality]**,

where **[prep]** is a preposition contained in a dedicated gazetteer and **[municipality]** is the Tuscany municipalities gazetteer.

– A similar strategy is applied to detect fractions names, by using the following pattern:

[prep] + [district]

where **[district]** is any word which can be recognized by the Km4City Smart City API as a geographical reference and **[prep]** is equivalent to the one in the former case. Examples for this kind of patterns are: "*Via Dante Alighieri*", "*in Pisa*", "*Via Dante Alighieri in Novoli*".

- The pattern implemented for the two-tokens location detection applied to bi-grams extracted from input query text (highlighted in blue in Fig. 5) is the following:

[streetKind] + [word]

where **[streetKind]** is a word representing the kind of street (e.g.: *via*, *viale*, *piazza*, *vicolo*, etc., which are the Italian urban nomenclatures respectively corresponding to the English terms *road*, *avenue*, *square*, *alley*, etc.) contained in a dedicated gazetteer. The **[word]** token is a potential geographic reference name. It is actually estimated by the system to be the name of a location only if it matches one of the geographic items populating the Km4City Knowledge Base. To this goal, the Knowledge Base is queried through the Km4City Smart City API, which also supports fuzzy retrieval. Examples of sentences following this pattern are: "*Via Alighieri*", "*Via Dante*".

- The pattern implemented for the three-tokens location case (highlighted in red in Fig. 5) is the following:

[streetKind] + [prep] + [word]

where **[streetKind]**, **[prep]** and **[word]** are equivalent to the ones described in the two-tokens case. An example of a sentence following this pattern is "*Via degli Alighieri*".

Adding the *n*-grams analysis approach for geoparsing permits to improve the geographic reference retrieval, providing a modular technique and allowing the system to handle different degrees of expressiveness for geographical references conveyed in the user query. The system is capable of recognizing region cities, fractions districts, roads, city elements and toponyms in the Tuscany area, and associating them to a corresponding geolocation. Moreover, the system can also recognize partial or misspelled toponyms (for instance, street names expressed with the surname only, instead of the complete name, which is a frequent practice in common language).

Once the initial geoparsing phase has recognized one or more geolocated entities, these ones are searched into the Km4City Knowledge Base through the Smart City API, in order to retrieve and associate a corresponding latitude and longitude tuple. Depending on whether the geoparsing process has extracted one or more locations, and depending also on their spatial resolution, the Km4City Smart City API runs different SPARQL queries on the Km4City repository. For instance, in case the user query contains multiple geographic intents, e.g. a street name (extracted as a high spatial resolution geographic item by the *FindLocation* Jape earlier rule) and the pertaining municipality (extracted as a low spatial resolution geographic item by the *FindPlace* Jape), the SPARQL query results are filtered retrieving only the location (a street in this case) comprehended in the requested municipality. If a single high-resolution geographic item is extracted in the geoparsing phase, without a lower-resolution geographic reference, this leads often to have ambiguities in determining the exact location. When such cases occur, the system returns the location in the nearest municipality, according to the user GPS position (if enabled). This is a disambiguation technique for handling and trying to resolve possible homonymy cases, which are frequent for names of places belonging to different nearby municipalities or districts (e.g.: the toponym "*Main Street*" is found in more than 7000 cities in the U.S. only, according to a Census Bureau research (Census Bureau, 1993)). If the output of the geographic coordinates retrieval is empty, then the extracted location is considered as a false toponym. This may be the case, for instance, of phrases like "*Via di qui*", which contains the Italian street/road nomenclature "*Via*", which is a polysemic word meaning also "*away*"; actually, such a sentence means "*Go away from here*".

Finally, after the geocoding phase is completed, an array is obtained containing the estimated service category, any recognized geolocated entities and their corresponding geographic coordinates. The array fields are then used as filters to query the Km4City repository, in order to retrieve actual geolocated POIs and services belonging to the elected service category, as detailed in the next section.

2.2.3. Geolocated data extraction module

Once the service category, corresponding to the extracted user need, is elected by the Service Extraction module (Section 2.2.1) and the latitude/longitude tuple is extracted by the Location Extraction module (Section 2.2.2) the system can proceed to query the Km4City Knowledge Base to retrieve and provide, as final results, a list of geolocated POIs, commercial activities and services belonging to the elected service category. These items are ordered by increasing distance with respect to the detected geographic reference (if estimated), or with respect to the user GPS position (if enabled). This is the case when the system has not recognized any geographic location in the input query, as well as if it has detected false toponyms in the geocoding phase. This methodology also provides a quick toponym disambiguation strategy: in case of multiple toponyms with the same name, the nearest toponym to the extracted location is retrieved. Eventually, if either user's GPS position is not available nor manually inserted in the input web interface (Section 3.1), the output results are not ordered by geographical position. The retrieval of the linked open data corresponding to commercial activities and services is presented in the output web page.

3. Validation

In this section, a quantitative evaluation of Paval capabilities to correctly retrieve and satisfy a practical user's information need is presented. Currently, the system relies on the Km4City repository for retrieving data, that is a local Knowledge Base covering the Florence metropolitan area and the Tuscany region, in Italy. However, Paval is easily adaptable and scalable to different environments and data sets. A data harvesting step for the validation process has been made on the territory by requesting users to pose several natural language queries about any information needs and requirements, oriented to the retrieval of local geolocated POIs and services. The service is accessible through a self-designed web user interface (Section 3.1). The collected data is used to separately evaluate each module of the architecture (Section 3.2) and to validate Paval against the most popular virtual assistants (Section 3.3).

3.1. Validation user interface

A dedicated front-end has been developed to provide an intuitive interface for gathering validation data, through which interviewed users have been able to submit their queries and to visualize the retrieved POIs (see Section 3.2 for details about the validation methodology used). The User interface deals with handling users' queries, sending text and contextual data (user's GPS position) to the processing modules and presenting the result data as output.

The expected user input for the system is a natural language query supporting the following five languages: Italian, English, French, German and Spanish, which can be selected through the relative option available by the rightmost button on the home page. The search form is

managed by the web interface available at https://paval.disit.org/Paval. The user query can be either a sentence or question containing a user-need for a geolocated POI, activity or service. The input phrase can also contain multiple geographical references at different level of street, square, etc., as well as at level of municipality, fraction, or district in the domain of the above described Km4City Knowledge Base. The Knowledge Base mostly contains resources which are in Italian language and thus geographic references included in the user query should most suitably be in Italian (i.e. *"I'm looking for a bar near piazza del Duomo"*). Currently, the covered area is the Municipality of Florence and, partly, the Tuscany Region (which is still under development, in conjunction with other Italian metropolitan areas). Other input elements can be collected through the web interface, that is the GPS location of the user (retrieved by using the Google Geolocalization APIs). The user position can also be manually specified by the user inserting the geographical coordinates in the proper input field; if specified, such coordinates are taken as a reference for the user position (for example if the web browser or mobile device used cannot retrieve or enable the user GPS location).

The output results are shown in the Query Results web page (Fig. 6), which is only accessible after a successful outcome of the computation. Results are ordered by increasing distance with respect to the detected geographic reference (if expressed by the user in the input query); if no geographic location is expressed or recognized, the results are ordered by increasing distance with respect to the user position (if allowed by the web browser). In this page, some metadata of the retrieved items are also presented, like service name, address, the type of commercial activity and a brief description (if present in the repository). To provide a more intuitive feedback of the retrieved data, the coordinates of the recovered local activities are used to show them as marks on a geographic map (Fig. 6). Under the map, the detected geographic reference (if any), its coordinates and the elected Km4City service categories are recalled. On the top of the output web page are also found the input text box and the buttons to submit a new query and to change the query language.

3.2. Validation data and methodology

A dataset of 1264 user queries (in Italian) is collected through the user interface proposed in Section 3.1. These queries are submitted to Paval through a specifically designed automatic procedure for query execution, and the results are evaluated respect to the aim of the principal components of the Paval architecture: the service extraction (Section 3.2.1), the location extraction (Section 3.2.2) and the geolocated data extraction (Section 3.2.3). In this way, we aim at assessing separately the different modules of the system. The partial validation of each level of the architecture is provided; the outcome of each level involves the user experience thus to build a ground truth for each level the validation queries have been posed to real users and the outcomes have been stored for comparison with the Paval outcomes. The overall degree of agreement between the human validators is analyzed through the Fleiss' kappa (Fleiss, 1971). For each level of the validation a comparison of the evaluation metrics among the main assisting tools is reported.

3.2.1. Service extraction evaluation

The goal of this section is to evaluate the capabilities of Paval estimating the user-need and classifying it into the appropriate Km4City ontology taxonomy label.

The quantitative evaluation for the service estimation level is made adopting the standard IR metrics of precision and recall. The precision metric assesses the capabilities of the system to retrieve relevant documents respect to the user query, while recall measures the capabilities of the system to retrieve all the relevant documents in the collection. The F-measure evaluation is also provided, which is expression of the harmonic mean of precision and recall, respectively. These metrics are calculated taking into account the number of True Positives (TP), False Positives (FP), False Negatives (FN) and True Negative (TN) outcomes.

$$Precision = \frac{TP}{TP+FP} \qquad Recall = \frac{TP}{TP+FN}$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

$$F\text{-}measure = 2 * \frac{Precision * Recall}{Precision + Recall}$$

To provide an association between such metrics and our observations the below methodology is adopted:

- A reference classification (ground truth) for the service level is built annotating by 5 human assessors the queries with correct outcome at level of service, based on their experience. Such annotation is averaged (Gwet, 2014) by service type and mapped to the most comparable Km4City ontology category.
- A **TP** is considered whenever a query effectively expresses a user-need which can be fulfilled by Paval and the system elected the correct service; to estimate the fulfillment of the information need the validators experience stated within the ground truth must be taken into account. Each query outcome at level of service is thus compared with the corresponding ground truth and a match is considered a TP.
- If the information contained in the Paval outcome does not match with the validators' expectations of fulfilling the expressed need contained in the ground truth, a misclassification error (a **FP**) is reported; a FP is considered also in the cases of an elected service while there was none expressed in the ground truth.
- The occurrence of a **TN** represents the cases of queries which do not express a user need (the service is not present in the ground truth) and it is correctly not estimated by the system.
- A **FN** is found whenever the system is not able to estimate any service corresponding to the user need, while the human validator actually expects a not-null or not-empty value.

To evaluate the overall degree of agreement between the validators, and thus to get an estimation of the legitimacy of using the employed ground truth, the Fleiss' kappa is assessed. Using $N = 1264$ queries the degree of agreement κ among $n = 5$ validators on $k = 4$ categories of TP, FP, TN and FN is obtained using the formula:

$$\kappa = \frac{\bar{\mathrm{P}} - \bar{\mathrm{P}}_{\mathrm{e}}}{1 - \bar{\mathrm{P}}_{\mathrm{e}}}$$

where the means:

$$\bar{\mathrm{P}} = \frac{1}{N}\sum_{i=1}^{N}\left(P_i\right); \quad \bar{\mathrm{P}}_{\mathrm{e}} = \sum_{j=1}^{k}\left(P_j^2\right)$$

are obtained calculating p_j, that is the proportion of all queries $q_{1,1}, \ldots, q_{N,N}$ which were classified as the jth category, and P_i, that is the extent to which raters agree for the ith query, as:

$$p_j = \frac{1}{Nn}\sum_{i=1}^{N} q_{i,j}, \quad \forall j \in \{1, \ldots, k\}$$

and

$$P_i = \frac{1}{n(n-1)}\left[\left(\sum_{j=1}^{k} q_{i,j}^2\right) - n\right], \quad \forall i \in \{1, \ldots, N\}.$$

The Fleiss' kappa is found to be:

$$\kappa = 0.81$$

which is indicative of an adequate inter-rater agreement respect to the service level.

The evaluation of the service extraction module (Table 2) has produced the following results:

The stabilization of the assessed measures towards the reported values varying the number of queries is shown in Fig. 7.

A comparative evaluation at level of service is performed on the validation queries with respect to Google Assistant, Apple Siri and Microsoft Cortana, in order to compare the effectiveness of the comprehension of the user-need by our retrieval method and some state of the art tools. The

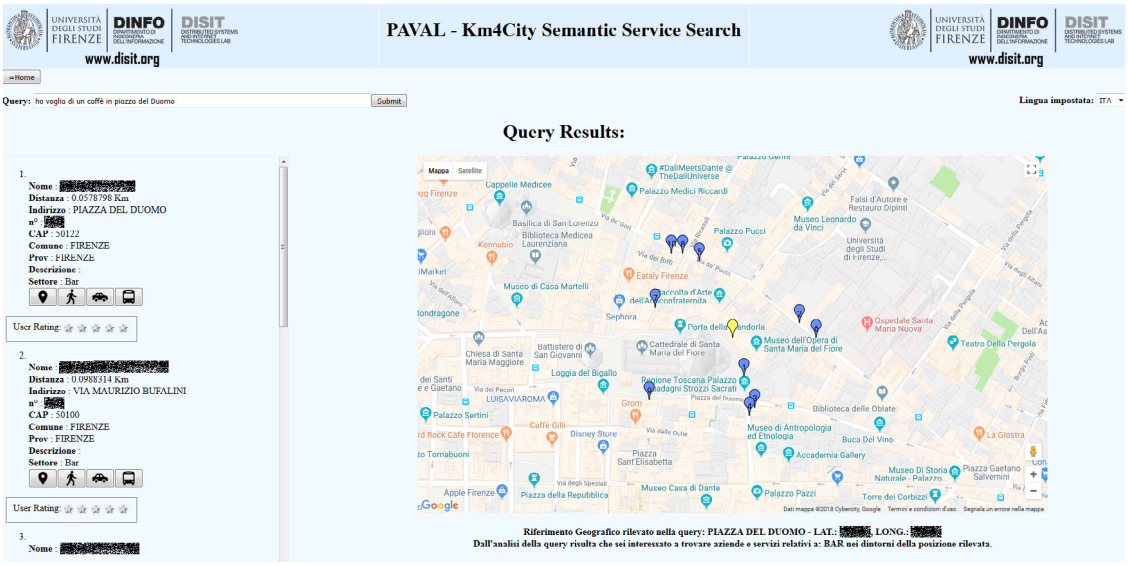

Fig. 6. Output web page.

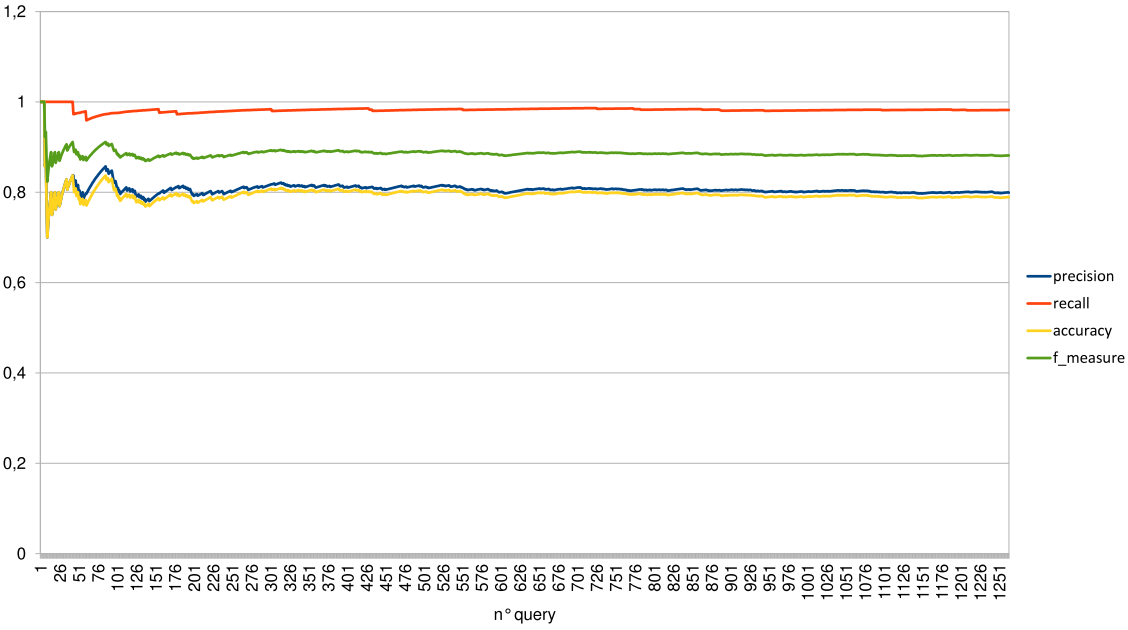

Fig. 7. Service level evaluation plot.

Table 2
Service level evaluation.

Precision	0.799
Recall	0.982
F-measure	0.881

Table 3
Service level comparison between main personal assistants and Paval.

Personal assistant	Precision	Recall	F-measure
Paval	**79.90%**	**98.20%**	**88.10%**
Google Assistant	77.74%	28.34%	40.75%
Apple Siri	60.07%	64.42%	62.29%
Microsoft Cortana	63.51%	14.43%	22.92%

same queries are posed to the aforementioned tools and the evaluation methodology is similar to the one described for assessing our system.

The results of the comparative evaluation at level of service has produced the results, shown in Table 3.

The low values found for the recall measure can be explained because the evaluated assistants are general purpose systems (which Paval is currently not) and the result-set for those assistants is often a set of websites related only to the syntactic form of queries exhibiting not-explicit user-intents.

The validation of the Paval service extraction module is publicly available at: http://www.disit.org/paval/pavalsources.rar.

3.2.2. Location extraction evaluation

The goal of this section is to evaluate the capabilities of Paval detecting the presence of a geographic reference within the user query and associating it to a latitude and longitude tuple. Furthermore, if no geographic reference is contained within the query, the engine should be able to infer the locality of the request assigning to the query a nearby latitude and longitude tuple.

The association between the metrics described above and our evaluation goal is the following:

Table 4
Location level evaluation.

Precision	0.890
Recall	0.935
F-measure	0.912

Table 5
Service level comparison between main personal assistants and Paval.

Personal assistant	Precision	Recall	F-measure
Paval	**89.05%**	**93.58%**	**91.26%**
Google Assistant	87.98%	33.12%	42.02%
Apple Siri	68.68%	65.84%	64.22%
Microsoft Cortana	72.26%	19.39%	23.63%

- A reference classification (ground truth) for the location level is built evaluating by 5 human assessors the validation queries and annotating the correct geographic reference, if expressed within the query, and the user position if no geographic location is expressed within the query. Since the user position and the positions extracted by Paval are in the form of (*latitude, longitude*) tuples, all the coordinates have been provided to validators in the form of street and municipality.
- A **TP** is considered whenever a query contains an explicit geographic reference and it is correctly referenced by Paval to a (*latitude, longitude*) tuple. A TP is associated to the presence of a geographic location in the ground truth which is also expressed within the query.
- A **FP** is considered whenever a query contains an explicit geographic reference and it is associated by Paval to a wrong (*latitude, longitude*) tuple. The wrong association reflects the presence of a geographic location in the ground truth which is not the same expressed within the query (at street and municipality level). A FP is considered also in the cases when a geographic location is extracted by the system while there was none within the user query.
- **TN**s are considered in those cases when a query do not exhibit geographic references and Paval does not return any extracted location (the ground truth contains the user latitude and longitude as user position); however, in order to provide geolocated results to better meet the user's need, the location is set as the user position gathered by the system, if available.
- A **FN** is found whenever the system estimates the location user-need in the vicinity of the user, while there was an explicit geographic reference expressed in the user query.

The evaluation of the location extraction module produced the results given in Table 4.

The stabilization of the assessed measures towards the reported values varying the number of queries is shown in Fig. 8.

To evaluate the capabilities of Paval in extracting the geographic references from the validation queries a comparative evaluation at this level is performed with respect to Google Assistant, Apple Siri and Microsoft Cortana, posing to each assistant the geographic reference only, whenever contained within the query.

The results of the comparative evaluation at level of service has produced the results shown in Table 5.

The validation of the Paval location extraction module is publicly available at: http://www.disit.org/paval/pavalsources.rar. The validation methodology which we applied so far allows to evaluate the user query respect to the service category and locality of the retrieved items, relying only on each item's belonging to the ontology class matching the user-need and not on specific data elements, which is the aim of Section 3.2.3.

3.2.3. Geolocated data extraction evaluation

The goal of this section is to evaluate the capabilities of Paval retrieving a consistent set of local businesses fulfilling the user-need expressed within a query respect to the levels of service, geographic location and distance of the local business. At the present level of the validation the distance of each data element from a geographic reference which may be expressed in user queries must be considered, together with the evaluation of the extracted service and of the extracted location to constitute a measure of the data relevance respect to the user query. The ranking function of Paval involves the distance: the ordering of the retrieved items within the result-set depends on the distance from the geographic references extracted from the user query; the better ranked results are thus the nearest to such position.

To correctly evaluate the Paval engine capabilities taking into account the ranking of the results, the metrics assessed in Sections 3.2.1 and 3.2.2 are not fairly comprehensive, thus the software Trec Eval is employed. Trec Eval is a software used in the context of the Text REtrieval Conference to provide a collection of metrics to evaluate the quality of the ranking of the documents retrieved by a search engine system compared with a ground truth. The two parameter files needed by the software to evaluate a search engine must contain respectively the similarity scores and rankings of the data retrieved by the system under evaluation and the relevance scores of the ground truth (for the same validation queries). In order to provide the relevance measures required by Trec Eval, a reference classification for the data level is built posing a sample of 50 queries from the validation query dataset to 5 human validators. Then, the validators are requested to compile a list of local businesses which they considered as relevant, based on their experience, for each query, ordered by distance. Each element of the ground truth is chosen by validators according to the fulfillment of their user-need at levels of service, geographic location and distance of the local business. Being the data elements ranked by distance from the extracted location, evaluating the distance is equivalent to evaluate the ranking. The ranked lists provided by each validator are joined and for each query a ground truth list is obtained ordering by distance and trimming at the first 10 data elements. To evaluate the inter-rater agreement among the validators participating to the data level ground truth construction, the Fleiss' kappa defined in Section 3.2.1 is assessed and is found to be:

$$\kappa = 0.68$$

indicating a lower, though adequate, inter-rater agreement respect to the service level.

A smaller subset of the validation dataset has been considered for this validation, since the manual creation of the needed ground truth has been a quite long and time-consuming process, in order to allow each validator to carefully annotate several hundreds of real world POI, local business and city services for all the 50 queries.

The similarity scores required for the output parameter of Trec Eval have been collected by the user interface (see Fig. 6), through the user rating element which is present for each ranked data element in the form of stars from 1 to 5. Such user evaluation is stored and used as a similarity score for each document retrieved.

Through Trec Eval the DCG (Discounted Cumulative Gain) measure (Järvelin and Kekäläinen, 2000) has been evaluated. The DCG is generally used to compare the performance of ranking functions and it is defined as following:

$$DCG_p = \sum_{i=1}^{p} \left(\frac{2^{rel_i} - 1}{\log 2(i+1)} \right)$$

The DCG is an evaluation of multi-grade relevance judgments and rankings of the results and it is assessed in the present evaluation to capture the relevance of a result respect to its position within the result-set. Fig. 9 shows the degradation of relevance respect to the number of results taking into account each result's ranking as discount factor.

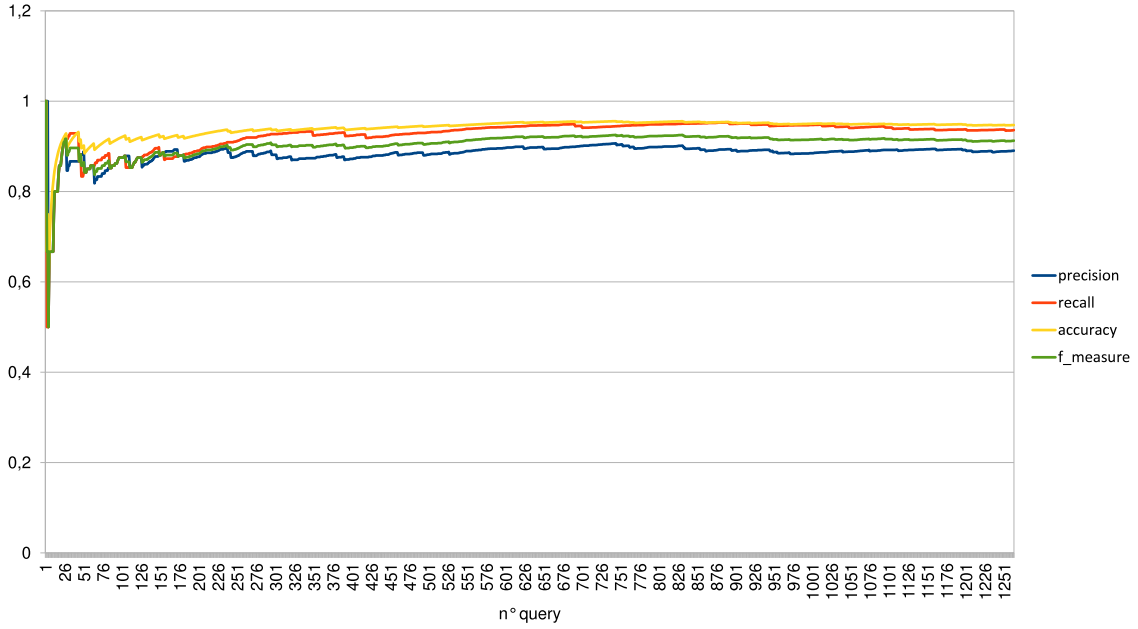

Fig. 8. Location level evaluation plot.

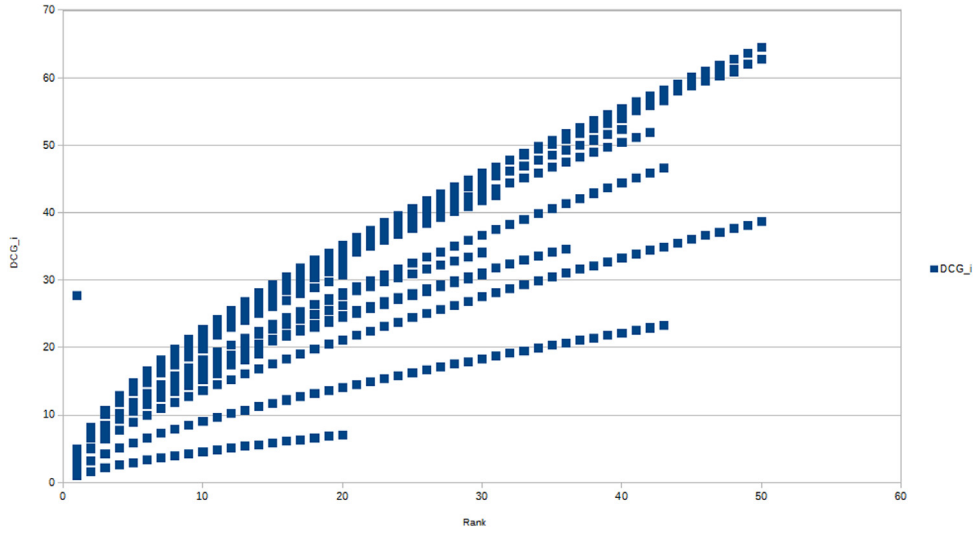

Fig. 9. Paval DCG evaluation.

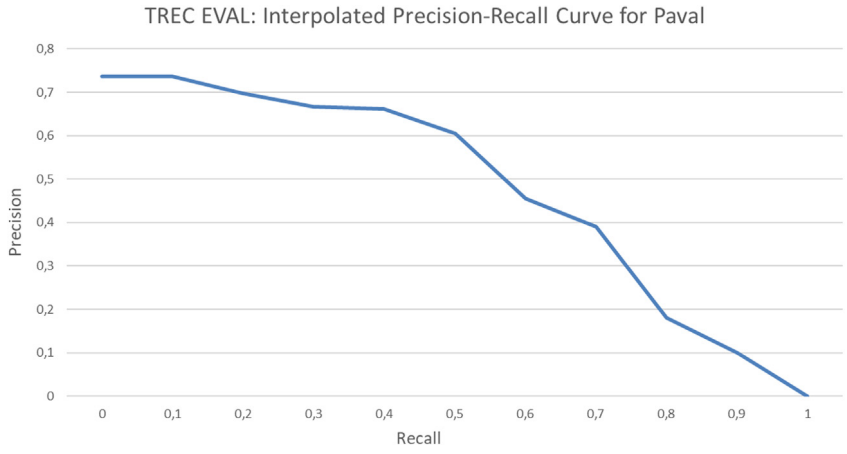

Fig. 10. Trec Eval Interpolated precision/recall curve.

Additional metrics are evaluated by Trec Eval, such as the 11-points interpolated average precision–recall, where precision is measured at the 11 recall levels of $k \cdot 0,1$, where k is integer and $0 \leq k \leq 10$. The curve is depicted in Fig. 10.

The precision value for a given recall (Rec_i) is the mean over all queries of the maximum precision over the relevant elements found by the system with a recall equal or superior to Rec_i. Thus, for $Rec_i = 0$, the precision is the mean of the maximum precisions for all the queries.

The validation dataset at level of data is available at: http://www.disit.org/paval/pavalsources.rar

A comparative evaluation has been made, in this case between Paval and Google Assistant on the 50 queries subset of the validation dataset. In order to focus the validation on the capabilities of the assessed systems to retrieve relevant items (in our case, POIs, local business and city services) instead of relying on their different knowledge bases, which could affect the assessment performance, we considered as reference a portion of the ground truth. Such portion is obtained intersecting, through an automatic procedure, the instances of Google Assistant and Paval KB (Km4City), finding a 61.3% overlay.

The result is shown in Fig. 11, representing the precision–recall curves obtained from the output of the Trec Eval framework for both the assessed systems.

The Trec Eval performance comparison have been analyzed in two different configurations, considering a different dimension (defined as N in the following) for the result set of the two engines: in the former case (Fig. 11a), we consider the first 10 results for each query ($N = 10$); in the latter (Fig. 11b), we consider the first 5 results ($N = 5$). It can be noticed that, in the second case, the precision of both systems degrades more rapidly at increasing recall. This may be due since we consider a smaller result set, thus it may occur that less relevant elements may be retrieved (assuming the same relevant elements are contained in the ground truth). Another aspect worth to be noticed is that the precision at 0 recall (i.e.: the mean of the maximum precisions for all the considered queries) does not significantly change for Google Assistant (about 70%), while increases for Paval (from about 73% to about 88%), with decreasing N. This may show a good capability of Paval in returning relevant elements in the first positions of its retrieved result list. In both cases ($N = 10$ and $N = 5$), Paval shows a higher performance, in terms of interpolated precision–recall, than Google Assistant.

3.3. English validation

The system can accept as input and process queries in English language through the same interface described in Section 3.1 by changing the language option which can be found on the input web page, therefore a quantitative evaluation is performed also for English language. Since all the interviewed users are Italian, all the collected test queries are in Italian language; moreover, since our Km4City Knowledge Base is designed and realized in Italian, in order to assess Paval performances for English and other supported languages, it would be necessary to translate all the classes and instances of the Km4City ontology. This would be a very long and costly process, and furthermore this approach may result to be significantly language-dependent. In order to provide support for other languages, the Paval web input interface is provided with an option which allows users to choose a language among the supported ones (see Section 3.1). When a language which is different from Italian is chosen, the Yandex online translation APIs are called and the input query is translated in Italian before being submitted to the Paval engine. Therefore, in order to assess Paval performances for the English language, the collected 1264 Italian queries were translated in English and then submitted to the Paval engine with the English language option activated. To assess Paval performances and give a quantitative measure for the capabilities of Paval annotating the user query with the correct service in English language, the service level only is considered. To unbind the English validation queries from potential automatic translation errors, the English data-set has been evaluated by a native speaker before being submitted to the system. Following the same approach used in the validation for the Italian language, the results show a precision of 74.67%, a recall of 96.24% and an F-measure of 84.57%, which are comparable with the ones obtained for the Italian language validation, though showing that Paval performs better respect to other assistants using Italian language. The study assessed for English language provided slightly lower measures for the Paval engine, while showing much better results for the other evaluated engines; this may be due to the fact that the automatic translation strategy employed may sometimes provide erroneous translations. The main results of this study are presented in Table 6.

Table 6
Quality measures comparison between main personal assistants and Paval (English).

Personal assistant	Precision	Recall	F-measure
Paval	74.67%	**96.24%**	**84.57%**
Google Assistant	**75.24%**	48.92%	58.77%
Apple Siri	72.28%	70.65%	71.45%
Microsoft Cortana	66.81%	27.53%	38.36%

Table 7
Classification of the validation queries.

Class	Definition of the class	Tot.
Type 1	Direct request of a full or partial Km4City service category label. i.e. *"I need a restaurant", "Bed and breakfast around me"*	25.60%
Type 2	Direct request of the name of a precise local business i.e. *"Take me to 'Da Mario'"*	2.93%
Type 3	Queries not exhibiting the service name as the user need i.e. *"I want to eat spaghetti", "My stomach hurts"*	74.40%
Type 4	Presence of precise geographical reference i.e. *"I need a restaurant in via dell'Oriuolo", "Hotels in Piazza del Carmine"*	35.12%
Type 5	Presence of partial or misspelled geographical reference i.e. *"Eat in Piazza del Dpomo", "Bar near piazza P. Leopoldo"*	16.32%
Type 6	Presence of multiple geographical references i.e. *"I want to read a newspaper near via Dante Alighieri in Pisa"*	5.24%
Type 7	Presence of geographical references i.e. *"Find a place to eat near ...", "I want to drink something in via ..."*	59.16%
Type 8	Queries inside Florence municipality i.e. *"Museums near piazza della Signoria"*	69.71%
Type 9	GPS localization allowed i.e. Given authorization from the browser: *"I need a restaurant nearby"*	100%
Type 10	Not transactional queries i.e. *"When did Garibaldi die?", "Find a video on YouTube"*	2.31%

The results in terms of precision, recall and F-Measure show that Paval performs mostly better than current state of the art personal assistants retrieving geolocated POIs and location-based services when the input is a natural language query. However, it is to be noticed that the goal of the present work, and thus also of this validation, is to assess the fulfillment of users' information needs and requirements oriented to the retrieval of geolocated POIs and services. In this sense, our system is focused on a more restricted domain, with respect to the tools presented in the comparative evaluation; for this reason, also evaluation metrics used to assess Paval's performances sound more restrictive, in these regards, for the other assistants. Actually, the latter have access to far larger amounts of data, besides including users' profiles, applications and social media access and management etc., so that they not always supply geolocated data as results. Instead, most of the times they provide generic web search, although in some cases they can perform a wider range of actions, including managing applications, schedule events, reminders etc. This can partly explain the large gap that sometimes is found in the recall rate among Paval and the other assistants, despite such generic web resources output by the other assistants assessed

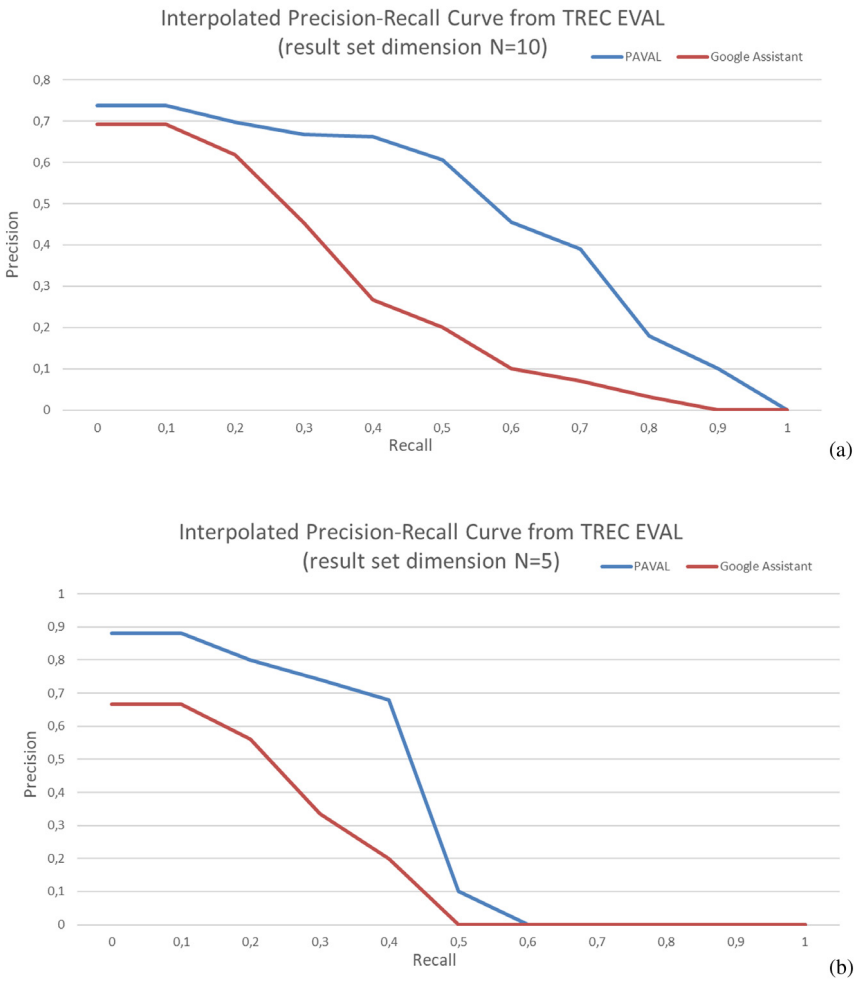

Fig. 11. Comparison between Paval and Google assistant for different result set dimensions $N = 10$ (a), $N = 5$ (b).

are considered as TP in the present validation, whenever they contain information that could satisfy the expressed user's need through the suggestion of a geolocated POI or service. We believe that providing geolocated data, when it is possible, is an added value for accomplishing user's practical needs on the spot, or on the move.

3.4. Validation data classification

It is also provided a classification of the types of validation queries (Table 7), and the cases with the corresponding query ratio (with respect to the total number of queries) have been identified in Table 7:

It is to be noticed that the sum of the percentages exceeds 100% because some query types overlap (i.e. a query containing both a service category label and multiple geographical references is classified either as Type 1 and Type 6). For further details the complete log table containing all validation data and evaluation metrics is available at:http://www.disit.org/paval/pavalsources.rar .

Based on the above classification, precision, recall and F-Measure metrics are evaluated for every class, each of which refers to a number of the validation queries of the same type, with the purpose of highlighting strengths and weaknesses of each evaluated virtual assistant. The following study is referred to the Italian language validation dataset, and provided the results shown in Tables 8–10:

The study shown in Tables 8–10 highlights the main weaknesses of these assistants in understanding the user need in queries which do not exhibit an intent at level of service category (Type 3) and in queries exhibiting a partial or misspelled geographic reference (Type 5); in the range of these categories Paval performs better than the other assistants. On the other hand, the above assistants provide more accurate answers than Paval in queries in which occurs the precise name of a local business (Type 2) and in queries which are not exhibiting a user need for a commercial activity (Type 10), although such strengths are not always noticeable by our study because the number of harvested queries of those kinds is low and, in some cases, such queries also contain the name of the service category to retrieve. Useful information is represented also by the main types of commercial activities correctly retrieved by Paval, which are mainly included within the *Wine and Food* and the *Shopping and Service* commercial sectors (see Fig. 12).

The classification of the kind of queries highlights the need for including a functionality for the understanding of POIs as geographical references, currently not recognized by our logic as georeferenced points (i.e. *"I'm at the city library and I need..."*). This problem could be resolved by assigning a geographical relevance score to sequences of bigrams and trigrams constituting the query.

4. Conclusions

In this paper the Paval framework has been presented, designed and realized as a location-aware virtual personal assistant for suggesting local POIs and location-based services from users' natural language queries. The principal aim of the presented work is to estimate and accomplish the information need and potential multiple geographic references expressed by the user query. The Paval system exploits NLP and semantic technologies, relying also on external knowledge like the

Table 8
Precision evaluation by query type classification.

Virtual assistant	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7	Type 8	Type 9	Type 10
Paval	**93.63%**	28.54%	**85.72%**	78.45%	74.57%	80.12%	**81.58%**	81.67%	**78.84%**	32.34%
Google Assistant	87.06%	92.12%	77.34%	**87.35%**	**89.78%**	**84.64%**	79.83%	**83.54%**	78.42%	**100.00%**
Apple Siri	70.14%	71.58%	60.23%	60.29%	50.10%	58.67%	56.69%	74.48%	60.01%	85.50%
Microsoft Cortana	78.45%	**98.42%**	67.56%	73.25%	64.23%	49.74%	74.44%	61.98%	76.40%	**100.00%**

Table 9
Recall evaluation by Query Type Classification.

Virtual assistant	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7	Type 8	Type 9	Type 10
Paval	**98.80%**	97.12%	**85.38%**	**95.10%**	**94.27%**	**88.64%**	**95.12%**	**99.75%**	**98.34%**	**100.00%**
Google Assistant	49.74%	**100.00%**	24.34%	37.63%	37.08%	46.2%	38.78%	23.54%	25.67%	57.67%
Apple Siri	78.47%	88.37%	74.44%	56.85%	75.56%	74.68%	64.74%	58.56%	68.19%	25.00%
Microsoft Cortana	34.56%	84.67%	5.58%	28.00%	19.11%	16.76%	18.98%	10.56%	10.47%	68.87%

Table 10
f-measure evaluation by Query Type Classification.

Virtual assistant	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7	Type 8	Type 9	Type 10
Paval	**96,15%**	44,12%	**85,55%**	**85,98%**	**83,27%**	**84,16%**	**87,83%**	**89,81%**	**87,52%**	48,87%
Google Assistant	63,31%	**95,90%**	37,03%	52,60%	52,48%	59,77%	52,20%	36,73%	38,68%	73,15%
Apple Siri	74,07%	79,09%	66,59%	58,52%	60,25%	65,71%	60,45%	65,57%	63,84%	38,69%
Microsoft Cortana	47,98%	91,03%	10,31%	40,51%	29,46%	25,07%	30,25%	18,05%	18,42%	**81,57%**

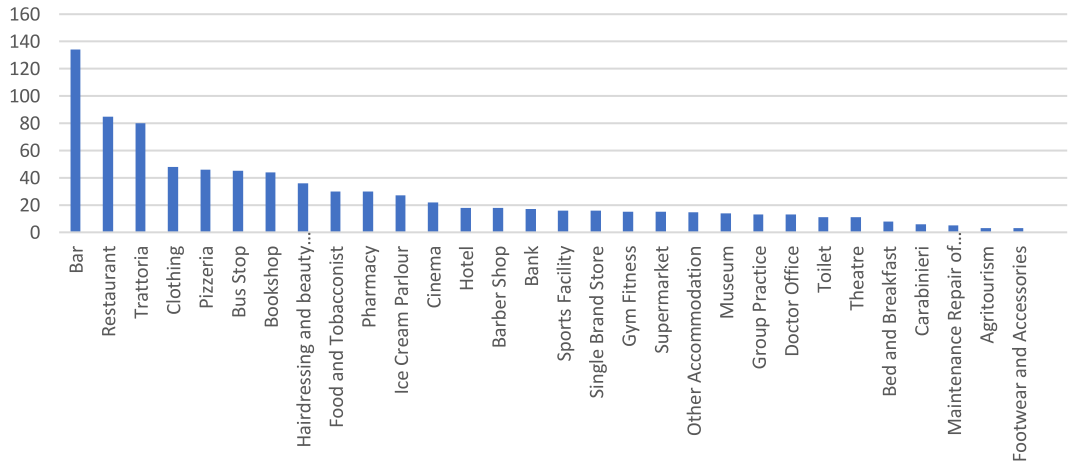

Fig. 12. Distribution of commercial service categories correctly retrieved by Paval over the 1264 user queries dataset collected for validation.

Km4City Knowledge Base to retrieve geolocated data. The generation and use of a reference corpus representing the semantic expansion of the ontology taxonomy labels improves current user-intent estimation techniques, which rely mainly on query expansion. The user-intent estimation is also improved proposing a novel term weighting strategy based on boosting of verbal parts of speech. The geoparsing method implements a reliable fine-to-coarse strategy based on the level of detail of the detected geographical reference and allows the detection of multiple references. The system is not directly relying on data thus resulting easily scalable and the performance does not degrade with larger inputs. The proposed system has been validated against the most popular virtual assistants, such as Google Assistant, Apple Siri and Microsoft Cortana, focusing the assessment on the request of geolocated POIs and services, showing very promising capabilities in successfully estimating the users' information need and multiple geographic references. The evaluation is performed by using a corpus of requests provided by real users during the validation. The same requests are posed to all assistants to estimate precision, recall and F-measure. The used corpus is accessible by other researchers to be used in the future as a benchmark. As a conclusion, the adopted corpus is focused on the smart city domain and in this domain the presented Paval assistant results better ranked with respect to the general-purpose systems.

Acknowledgments

The authors would like to thank the MIUR Smart City national founding, Italy SCN_00112, the University of Florence, Italy and companies involved for co-founding in the Sii-Mobility project. Km4City is an open technology of research of DISIT Lab.

References

Ahamed, S.I., Sharmin, M., Ahmed, S., Haque, M.M., Khan, A.J., 2006. Design and implementation of a virtual assistant for healthcare professionals using pervasive computing technologies. E & I Elektrotech. Inform. 123 (4), 112–120.
Amazon Alexa. Available at: https://developer.amazon.com/alexa.
Apple Siri. Available at: https://www.apple.com/ios/siri/.
Badii, C., Bellini, P., Cenni, D., Difino, A., Nesi, P., Paolucci, M., 2017. Analysis and assessment of a knowledge based smart city architecture providing service APIs. Future Gener. Comput. Syst. 75, 14–29.
Bellandi, A., Bellini, P., Cappuccio, A., Nesi, P., Pantaleo, G., Rauch, N., 2012. Assisted knowledge base generation, management and competence retrieval. Int. J. Softw. Eng. Knowl. Eng. 32 (8), 1007–1038.
Bellini, P., Benigni, M., Billero, R., Rauch, Nadia., 2014. Nadia Rauch Km4City ontology building vs data harvesting and cleaning for smart-city services. J. Vis. Lang. Comput. 25, 827–839.
Bhagwani, D., Pandey, H., Gupta, N., Sharma, Y., 2016. Geolocation based recommender system. Int. J. Adv. Res. Comput. Sci. Softw. Eng. 6 (4), 283–287.
Broder, A., 2002. A taxonomy of web search. SIGIR Forum 36 (2), 3–10.

Buscaldi, D., Rosso, P., 2008. A map-based vs. knowledge-based toponym disambiguation. In: Proc. 5th Int. Workshop on Geographical Information Retrieval, GIR-2008, CIKM-2008, Napa Valley, USA, October, pp. 19-22.

Buscaldi, D., Rosso, P., Peris, P., 2006. Inferring geographical ontologies from multiple resources for geographical information retrieval. In: Proc. 3rd Int. Workshop on Geographical Information Retrieval, GIR-2006, SIGIR, Seattle, WA, USA, August 10, pp. 52–55.

Campagna, G., Ramesh, R., Xu, S., Fischer, M., Lam, M.S., Almond: The architecture of an open, crowdsourced, privacy-preserving, programmable virtual assistant, I:n Proc. of the 26th International Conference on World Wide Web, pp.341-350, Perth, Australia — April (2017) 03-07. ISBN: 978-1-4503-4913-0. http://dx.doi.org/10.1145/3038912.3052562.

US Department of Commerce, Bureau of the Census, Geography Division. Census and You. Washington, DC, 1993. Available online at: http://www.usd116org/profdev/ahtc/lessons/PlautFel09/scans/2009_07_09/StreetNamesCensus.pdf.

Clements, M., Serdyukov, P., De Vries, A.P., Renders, M.J.T., 2011. Personalised travel recommendation based on location co-occurrence, CoRR, arXiv:1106.5213.

Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., 2002. GATE: a framework and graphical development environment for robust NLP tools and applications. In: Proc. of the 40th Anniversary Meeting of the Association for Computational Linguistics, ACL '02.

Datamuse API. Available online at: http://www.datamuse.com/api/.

Fellbaum, C. (Ed.), 1998. WordNet. An Electronic Lexical Database. The MIT Press.

Ferrucci, David, Brown, Eric, Chu-Carroll, Jennifer, Fan, James, Gondek, David, Kalyanpur, Aditya A., Lally, Adam, William Murdock, J., Nyberg, Eric, Prager, John, Schlaefer, Nico, Welty, Chris, 2010. Building Watson: An overview of the Deep QA project. AI Mag. 31, 59–79.

Fleiss, J.L., 1971. Measuring nominal scale agreement Among many raters. Psychol. Bull. 76 (5), 378–382.

Google Assistant. Available at: https://assistant.google.com/.

Gordon, M., Breazeal, C., 2015. Designing a virtual assistant for in-car child entertainment. In: Proc. of the 14th Int. Conference on Interaction Design and Children. ACM, pp. 359–362.

Gwet, K.L., 2014. Handbook of Inter-Rater Reliability, LLC Fourth Edition Ed. Advanced Analytics.

Harvey, P.H., Currie, E., Daryanani, P., Augusto, J.C., 2015. Enhancing student support with a virtual assistant. In: Proc. Of Int. Conference on E-Learning, E-Education, and Online Training. Springer, pp. 101–109.

Heredero, G.G., Penmetsa, H., Agrawal, V., Shastri, L., 2013. Activity context-aware system architecture for intelligent natural speech based interfaces. In: Proc. Of the Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence, pp. 21–35.

Hossein, M., Shahraki, N., Bahadorpour, M., 2014. Cold start problem in collaborative recommender systems: Efficient methods based on ask-to-rate techniques. J. Comput. Inf. Technol. 22 (2), 105–113.

Husain, W., Dih, L.Y., 2012. A framework of a personalized location-based traveler recommendation system in mobile application. Int. J. Multimedia Ubiquitous Eng. 7 (3).

Jansen, B.J., Booth, D.L., Spink, A., 2008. Determining the informational, navigational, and transactional intent of Web queries. Inf. Process. Manage. 44 (3), 1251–1266.

Järvelin, K., Kekäläinen, J., 2000. IR evaluation methods for retrieving highly relevant documents. In: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, pp. 41–48.

Kumar, 2011. Relevance and Ranking in Geographic Information Retrieval.

Kumar, G.K., Reddy, K.P.K., 2017. Cortana (intelligent assistant). Int. J. Sci. Eng. Technol. Res. (IJSETR) 6 (4), 698–701.

Lesk, Michael, 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In: Proc. of the 5th Annual International Conference on Systems Documentation, SIGDOC '86. ACM, New York, NY, USA, pp. 24–26.

Lops, P., De Gemmis, M., Semeraro, G., 2010. Content-based recommender systems: State of the art and trends. In: Recommender Systems Handbook. Springer, pp. 73–105.

Lucida Open Source Personal Assistant. Available at: http://lucida.ai/.

Madhusudhanan, R., Subramaniyan, D., 2016. Artificial intelligence – Making an intelligent personal assistant. Int. J. Res. Eng. Technol. 4 (6), 9–14.

Mandl, T., Gey, F., Di Nunzio, G., et al., An evaluation resource for geographic information retrieval. In: Proc. of the 6th Language Resources and Evaluation Conference. LREC 2008, Marrakech, Morocco, 28-30 May 2008.

Mathur, S., Bairagee, N., 2016. A survey paper on location aware recommender system. Int. J. Adv. Res. Comput. Sci. Softw. Eng. 6 (7), 331–335.

Matsuyama, Y., Bhardwaj, A., Zhao, R., Romero, O.J., Akoju, S.A., Cassell, J., Socially-Aware Animated Intelligent Personal Assistant Agent. In: Proc. of the SIGDIAL 2016 Conference, 224–227, Los Angeles, USA, 13–15 September 2016.

Microsoft Cortana. Available at: https://www.microsoft.com/windows/cortana.

Mycroft Open Source Personal Assistant. Available at: https://mycroft.ai/.

Navigli, R., Ponzetto, S., 2012. Babelnet: The Automatic Construction, Evaluation and Application of a Wide-Coverage Multilingual Semantic Network. In: Artificial Intelligence, vol. 193, Elsevier, pp. 217–250.

Nesi, P., Pantaleo, G., Tenti, M., 2016. Geographical localization of web domains and organization addresses recognition by employing natural language processing, pattern matching and clustering. Eng. Appl. Artif. Intell. 51, 202–211.

Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E., 2016. A review of relational machine learning for knowledge graphs. Proc. IEEE 104 (1), 11–33.

Noguera, J.M., Barranco, M.J., Segura, R., Martinez, L., 2012. A Location-Aware Tourism Recommender System Based on Mobile Devices. World Scientic, pp. 34–39, Chapter 7.

Palacio, D., Cabanac, G., Sallaberry, C., Hubert, G., 2010. On the evaluation of geographic information retrieval systems. Int. J. Digit. Libr. 11 (2), 91–109.

Palacio, D., Derungs, C., Purves, R.S., 2015. Development and evaluation of a geographic information retrieval system using fine grained toponyms. J. Spat. Inf. Sci. 11, 1–29.

Rodríguez-Hernández, M.C., Ilarri, S., Trillo-Lado, R., Hermoso, R., Location-aware recommendation systems: Where we are and where we recommend to go. In: Proc. Of First ACM RecSys Workshop on Location-Aware Recommendations, LocalRec'15, Vienna, Austria, September 19, 2015.

Rose, D.E., Levinson, D., Understanding user goals in web search. In: Proc. of the 13th Int. conference on World Wide Web (WWW 2004), New York, NY, 17–22 May 2004, pp. 13–19.

Schmid, H., 1994. Probabilistic Part-of-Speech Tagging Using Decision Trees. In: Proc. of the Int. Conference on New Methods in Language Processing, Manchester, UK.

Takeuchi, Y., Sugimoto, M., 2006. CityVoyager: An outdoor recommendation system based on user location history. In: Ubiquitous Intelligence and Computing. In: Lecture Notes in Computer Science, vol. 4159, Springer, pp. 625–636.

Tavčar, A., 2016. Recommender system for virtual assistant supported museum tours. Informatica, Int. J. Comput. Inform. 40, 279–284.

Thingpedia, Thingpedia by Stanford University. Available online at: https://thingpedia.stanford.edu/.

TREC IR systems evaluation software. [Online] Available at: http://trec.nist.gov/trec_eval/.

Uyar, A., Aliyu, F.M., 2015. Evaluating search features of google knowledge graph and bing satori. Online Inform. Rev. 39 (2), 197–213.

Welch, M.J., Cho, J., Automatically identifying localizable queries. In: Proc. Of ACM SIGIR, pp. 507–514.

Yandex online translator APIs. Available at: https://translate.yandex.com/.

Yang, W.S., Cheng, H.C., Dia, J.B., 2008. A location-aware recommender system for mobile shopping environments. Expert Syst. Appl. 34 (1), 437–445.

Yi, X., Raghavan, H., Leggetter, C., 2009. Discovering users' specific geo intention in web search. In: WWW '09: Proc. of the 18th Int. Conference on World Wide Web. ACM, New York, NY, USA, pp. 481–490.