User-driven Service Retrieval Platform for Converged Environments

Edgar Camilo Pedraza Alarcón
GIT
University of Cauca
Popayán, Colombia
epedraza@unicauca.edu.co

Julián Andrés Zúñiga Gallego
GIT
University of Cauca
Popayán, Colombia
gzja@unicauca.edu.co

Luis Javier Suarez Meza
GIT
University of Cauca
Popayán, Colombia
ljsuarez@unicauca.edu.co

Juan Carlos Corrales
GIT
University of Cauca
Popayán, Colombia
jcorral@unicauca.edu.co

***Abstract*—Today, there is an abundance of information and services in heterogeneous contexts, such as converged environments (Next Generation Networks) available for end users. However, the developments of Telecommunications and Internet converged services represent a high level of complexity for users without technical skills, since user's requests are represented by complex expressions that describe the required services. Thus, the search and selection of these services depend on the ability of the user to retrieve the most suitable ones, converting this labor in an inefficient task. With this in mind, and in order to improve the time to create convergent services, this paper proposes a novel approach that supports the automatic retrieval of services in converged environments, considering functional and non-functional properties of end-user's requests in natural language. Finally, we present the prototype that implements our proposed architecture and particularly, we describe in detail the defined tasks for natural language processing (NLP).**

***Index Terms*—automatic service retrieval; converged environments; natural language; Telecommunications and Internet services.**

I. Introduction

Service retrieval that accomplishes user requests is understood as an important stage in the composition process [1], [2], this allows the user to find and use a service based on a published description of its functionality or operational parameters [3]. Currently, services retrieval offers new challenges driven by the convergence of Web and telecommunications domains around the IP protocol, enabling the use of diverse and innovative services, regardless of the customer access network [4].

As result, operators and SMEs (Small and Medium Enterprises) that provide applications, services and content, must adapt their IT (Information Technology) infrastructure in order to develop capabilities to create and deploy new converged services with low time to market [5]. Understanding converged services, as the coordination of a range of services from different vendors, such that for end user view, it is a single service [6].

The above mentioned, both with new trends in application environments, where users are important generators of content and applications, opens up towards a new paradigm in which non-technical individuals are able to design and create their own fully customized services by integrating Web and telecommunication components, an activity that years ago was done only by expert developers due to its complexity. This context has led to the development of various projects focused on service retrieval. However, these approaches require that users specify their request with formal expression, which becomes complex for ordinary users, even more in dynamic and varied context, as converged environment (Web + Telco) [7], [8]. In many approaches, the retrieval process is limited to Web services, leaving aside other kind of resources. On the other hand, converged environments also cover resources, such as: data, Telecom capabilities, widgets, APIs, and so on, which facilitates the compliance with the user's request, but the complexity of service retrieval for end users is still very high due to the technical knowledge required.

In order to overcome the aforementioned issues, in this paper, we propose an architecture for automatic service retrieval in converged environments, considering the services functional properties (e.g., inputs, outputs, preconditions and effects) and non-functional properties (e.g., QoS, such as: availability, response time, reputation, etc) requested by the user in natural language (NL), to reduce the complexity in the creation of converged services for end users. In the description of the architecture, we focus on describing with more detail the natural language processing (NLP) mechanisms over the other functionalities because the main objective of this approach is to support the end user, This not only ensures a correct match between the user request and the system results but also provides mechanisms to maintain a relevant quality of the input (request) of the user. Finally, we present an evaluation of each task from proposed NLP module. The preliminary evaluation shows promising results in contrast to related approaches.

Typically, NLP is useful to analyze and produce semantic representations of the user's request, providing needed information to identify functional and nonfunctional properties of services. Therefore, we propose the use of NLP techniques to improve the process of service retrieval from requests made in NL. In this regard, we reduce NLP techniques to a problem of selection of possible terms, that represent the interface or functionality of a service, or terms that are used to describe a general order of execution of services (generic control flow), clarifying that our approach does not cover a formal composition process, but displays the responses of independent retrieved services after its execution in a generic order.

To do this, we adapt existing algorithms and develop rules for selecting terms according to certain conditions and compliance with regular expressions. The result obtained after this process, is a set of terms classified into several categories, the terms are used for obtaining services with their generic execution order. Thus, with the identification of key terms for the selection of services, supported by NLP techniques and adaptation of algorithms for matching services with those terms, it is possible to make an more accurate and automatic retrieval of services available within both Web and telecommunications domains.

The remainder of this paper is structured as follows: in the next section, we review the work related to the different topics that involves the current research. We present the problem statement in Section III. In Section IV we compare our approach with an existing one showing the advantages for the user. Then, in Section V, a high-level description of the proposed architecture is presented. To provide greater clarity an example is discussed in Section VI. In section VII we present a general evaluation of the system and finally in Section VIII we conclude the paper.

II. State of the Art

retrieval can be addressed under two main approaches: syntactic and semantic. The searching of services from a syntactic approach, considers either, interfaces matching techniques (e.g., WSDL, IDL) or keyword searching [9] that require exact matches at the syntactic level between the parameters of service descriptions. This leads to deficient results in the retrieval of service, obtaining services that do not correspond to the initial search criteria. The semantic approach allows the establishment of relationships between concepts that define the functionality of services (functional properties) and additionally, considers formal descriptions constructed by non-functional properties [10], achieving a more precise description of services and improving the quality of results (retrieved services) according to user needs [11]. From the foregoing, and given the nature of the problem, the proposed solution focuses on services retrieval based on lightweight semantics and NLP.

In turn, currently, an extension of previous approaches, is being widely accepted in the Web, where services are described by simple labels (tags) for different users, which corresponds to a classification of collaborative services through labels, without hierarchy or kinship default [12], [13]. A number of studies have shown that a set of tags added to resources, is rich and compact enough to describe and characterize, with a good degree of accuracy, the main concepts they represent [13].

The most relevant related works are described in two sections according to the main components of our proposal: NLP applications and user-driven service retrieval.

A. NLP Application

One of the most popular NLP applications is Siri. Siri is a personal assistant that resides on iPhone 4S and performs a range of tasks understanding natural speech. Siri is based on artificial intelligence and NLP mechanisms, is able to know which application to open based on natural language requests. Working on the operating system level [14], with information from contacts, music library, calendars, reminders, and Apple's data centers, Siri is able to understand the requests and return responses [15]. Siri works with the built-in application of IPhone including weather, stocks, messaging, and others, but leaves aside the retrieval of external resources such as web services and some telecommunication capabilities, in addition, to our knowledge, Siri does not consider non-functional properties.

In [16], the IBM research team developed a supercomputer that performs analysis phases of natural language questions composed by hundreds of algorithms, some of these phases present a similar approach with the proposed ones in this paper. However, it requires a complex system composed of multi-core hardware processor, hardware capabilities that are still difficult to migrate to some end users devices such as PDAs or cell phones.

In the work developed in [17], the authors address the selection of Web services based on requests expressed in natural language, this solution is based on language restrictions, matching the structure of the request with predefined patterns for decomposition into blocks using keywords. Subsequently, the request is processed and transformed into a data flow and control model expressing the general logic of the new service. In addition, the authors propose the use of a common ontology and a NL dictionary, in order to relate textual fragments with functional parameters of such services. This work presents disadvantages when limiting requests to simple sentences.

Based on concepts graphs and conceptual distance measure, a solution is presented in [2]. Its purpose is to calculate the similarity between the user's request, represented by keywords, and services available in a repository. Within the linguistic analysis, different processes are performed: text segmentation, irrelevant word removal, elimination of derivatives (stemming) and grammatical corrections. The authors admit their proposals lack of dynamic adaptation at runtime.

The approach of [10] re-uses the converged services creation environment of project SPICE (Service Platform for Innovative Communication Environment) [18] to facilitate services retrieval with different types of semantic annotations. The author focuses his work on the development of an intelligent agent in charge of analyzing the application in NL to extract semantic information, specifically the goals, from which additional semantic information is derived, as inputs and outputs, which are used to retrieve services and report their order composition. However, retrieval's throughput and processing critically depend on the amount of services stored in the repository.

B. User-Driven Service Retrieval

Some tools and techniques have been developed to enhance resource retrieval (e.g., photos, videos, services) taking into account the information generated by the users, an example is the

Folksonomy Ontology Enrichment Tool (FLOR) [19], which performs an automatic semantic enrichment of collaborative tagging systems, from user generated tags, creating a semantic layer to describe the concepts of those tags and their relations. The tool reuses existing knowledge such as online ontologies indexed in the Watson Semantic Web Gateway and WordNet, the system also performs three phases that guarantees a lexical processing of terms, a sense expansion and a final semantic enrichment. The major drawback is that the tool presents high percentages of incorrect semantic enrichment, mainly due to the semantic expansion phase for the differences in term's definition of Wordnet and the online ontologies.

The prototype developed in [20] is implemented to improve the information retrieval in tag-based systems. Taking as input the tags provided by the end-users, the prototype adds system tags from semantic ontologies (WordNet or MultiWordNet), or tag clusters imported from the Flickr website, to annotate and retrieve videos previously imported from real tag-based system (Youtube). While a generic architecture is developed without delving into the different stages, the key ideas and concepts are useful in directing the enrichment of user-generated tags with sources of knowledge.

In [21], the work is based on the collaborative tagging of web services, where users annotate indexed Web services with keywords, in order to define structured collaborative tagging, differing between tags: input, output, and behavior of a service. In the proposed technique, the authors define a matchmaker for the Web services retrieval, which involves two stages: classification and service ranking. However, the language used as input is based on a system query, i.e., the user's query is realized in a formal language.

From the previous review, limitations are evident because most of them are based on Web services retrieval with functional preferences, leaving aside in first place, non-functional requirements that may provide more sense to services semantic descriptions, and second, not considering other kind of resources such as Telecommunication capabilities. Also, many works are not end-user centered, which makes them complex for ordinary users, moreover these works are based on formal languages, or they are limited to simple phrases made in natural language, therefore, flexible requests made in natural language are not considered. Apart from [17], the approaches lack in showing the retrieved services in a flow to express the logic of the request, useful for subsequent service composition process. Finally, the automatic retrieval of services in converged environments is a recent topic of research, where, considering all the above features has not yet been developed.

Therefore, we propose a novel end-user centered approach which facilitates the interaction of the user with the system through requests made without formal languages, considering techniques to expand the search in order to increase the accuracy of the services retrieval process and with a recommendation service system. Our approach also considers functional and non functional parameters of services and the arrangement of the retrieved services in a generic flow to express the logic of the request.

III. Problem Statement

One of the key challenges of competitiveness of the existing telecommunications companies is the reduction of time in the process of creating new converged services. For this reason, converged services creation environment has been developed [22], [23], [24], which require formal requests with complex expressions by the developer, in order to compose services that are selected manually. The service manual selection shows that the process of creating converged services depends on the ability of the developer to select the most appropriate services, which is wasteful and inefficient work, since it is difficult for the human capacity to generate compositions that go hand in hand with the growing number of services available on the Web and Telecommunications domains.

In this vein, the creation time of converged services can be improved by reducing the complexity of the activities. Activities such as the selection of atomic services that are part of a composite service, need to be more flexible and agile [25], [26], [27]. To achieve this, it is necessary to define mechanisms to facilitate this process, such as making requests in a simpler and understandable language, with the intention of automating tasks in the identification of requirements and selection of services.

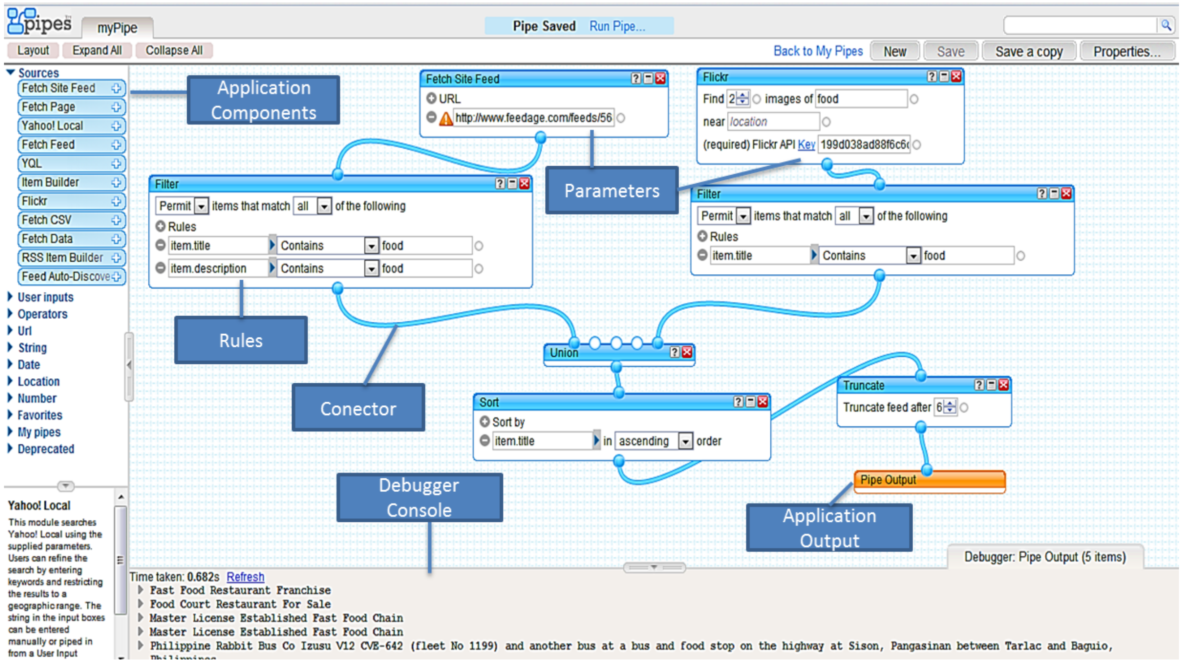
In this scenario, we propose an approach that allows the services retrieval in a converged environment that meets the end-users requirements. Therefore, the following problem statement is formulated:

How to reduce the complexity of services retrieval to end users in a converged environment, by ensuring compliance with their requirements?

IV. Challenges and Solutions

The proposed architecture considers problems found in currently user-driven systems for services retrieval that simplifies this process by allowing users make requests in natural language. To illustrate our proposal, let us consider an application created by Yahoo! Pipes [28], the application required by the user, retrieves food and beverage businesses for sale feeds from a website related with ”food”, also filters the content based on the title and the description that must contain the word ”food” and retrieves two tagged images from Flickr [29] with the word ”food” applying a similar filter. Finally, the application joins the retrieved information and organizes it to publish items for viewing. This process is outlined in Figure 1.

Thus, this application can be summarized to a request like: *”Get pictures of food and places selling food”*. However, for that, in this tool seven components were needed, also it requires additional configuration by the user, such as defining the URL of the feed website, manage filtering rules and define the retrieved information flow between components. It is also necessary that the user knows the different components and the functionality of each one, for example the user must know that Flickr is an image hosting website. The complexity and level



```
Time taken: 0.682s Refresh
▸ Fast Food Restaurant Franchise
▸ Food Court Restaurant For Sale
▸ Master License Established Fast Food Chain
▸ Master License Established Fast Food Chain
▸ Philippine Rabbit Bus Co Izusu V12 CVE-642 (fleet No 1199) and another bus at a bus and food stop on the highway at Sison, Pangasinan between Tarlac and Baguio,
```

Fig. 1. Application created using YahooPipes

of knowledge required for this application is not appropriate for a common end-user.

Addition to the above restrictions, with the increase of online repositories and development of Telecommunication services, high precision service retrieval is required, also it is necessary to develop systems that are increasingly intuitive and easy to handle for users without technical skills, and that allow the automation of tasks in the identification of requirements and selection of convergent services. These challenges can be addressed considering facilities such as the use of natural language, which improves the user interaction with the system.

Also, there are other important features that should be taken into account to meet user requirements, such as speed rates of resource retrieval process or the possibility that can give a tool to allow a user to learn from others users of the tool (social-aware recommendation concept [30]). These features are considered in the system, through collaborative tagging mechanisms [21], which are an alternative solution of ontology-based approaches. Ontology-based approaches are inadequate primarily because the ontologies are expensive in creation and maintenance, also end-users are not involved in the development process of ontology, which fails to use a different vocabulary to request a service. On the other hand, collaborative tagging allows that users can see the annotation made by other users on the resources and hence extract the semantics from annotation of the community, [21]. This approach allows the handling of different kind of resources, which facilitates work in convergent environments. However, it is appropriate to clarify that due to the magnitude of the project, the description of the tagging mechanisms is omitted, and will be presented in future papers.

Our proposal allows the user to make a request as simple as: ”I want to receive pictures of food and places selling food”, to retrieve components that perform this functionality. Additionally, our approach shows to the user how these components are arranged in the flow, which facilitates the interaction of the user with the system through requests made without formal languages. Likewise, our proposal recommends services during and after of the user’s request and uses NLP techniques applied to the request, taking advantage of collaborative tagging and formal knowledge sources; thus, the system retrieves and publishes the outcome of the converted user’s request into a components flow that represents the desired service. A Detailed explanation of the conceptual model and architecture is presented in the next section.

V. Architecture

This section contains a detailed description of the architecture proposed, for automatic services retrieval in converged environments. The architecture is organized in four phases: Natural Language Analysis, Recommendation, Matching and Association. Phases are also composed of internal modules that carry out small process, most of them sequentially organized. The system has as input, user’s request made in NL from a mobile device, and gives finally a service ranking and a generic control flow as output. In section VI, is described in more detailed the results of the system, how act this modules and how a requester can execute them. Below, architecture modules are described with its respective functionality.

- *SpellChecker:* This module receives user’s requests and checks over possible mistakes, underlining misspelled words.
- *Autocomplete:* The module auto completes some words of the request, showing a possible word representing what

the user wants to write.

- *Tokenizer:* It has as input the NL request and from this, it obtains words, phrases or symbols called tokens.
- *Filter Words:* is the responsible for removing non-sense words by comparing with a set of words previously identified.
- *Words Tagging:* tags words according to its grammatical category (e.g., she ”pronoun”, loves ”verb”, animals ”noun”).
- *Named Entity Recognition (NER):* It classifies the words into ”functional” or ”control” categories. In ”functional” exists also two different categories: ”input-output” and ”behavior”.
- *Semantic Analyzer:* in charge of semantic disambiguation process of input words, by selecting the correct meaning of a word according to request’s context.
- *Normalization tags:* functions as a stemmer, obtaining their corresponding lemmas from the input words. (e.g., *doggie* or *dogs* becomes *dog*).
- *Coreferencer:* Associates possessive pronouns with the reference subject of the sentence (e.g., *”Make a call to Mary and then send a present to her”* - relates *her* with *Mary*).
- *Non Functional Requirements Recommender:* It searches non-functional parameters in the repository from functional request previously written by user.
- *Service Recommender:* It searches request-service information in the repository obtained from prior inputs of users.
- *Matcher Functional Requirements:* It obtains from cluster services, the first rank, by matching functional requirements.
- *Ranking Generator:* It obtains the final ranking of services considering, if exists, non-functional requirements, of services, such as QoS.
- *Services:* conformed by Web and Telecommunications domain services, which can be conceptually organized by their functional properties.
- *Folksonomy:* It reflects through tags the collective intelligence of a crowd or a community (*wisdom of the crowds*) in giving meaning to available resources (Power Tags) (e.g. QoS, Telco, IT, among many others), supporting functional (internal categories input-output and behavior) and non-functional properties descriptions.
- *Flow Ranking Repository:* It stores an association of service (tags) obtained at the end of the semantic matching phase.
- *Non-functional Repository:* It stores an association of non-functional and functional parameters of cluster’s services.
- *Generic Flow Generator:* It generates an generic control flow, based on keywords taken from the user’s request processed.
- *Flow-Ranking Associator:* It associates the flow obtained by the *Generic Flow Generator* module with the ranking output of the matching semantic phase, obtaining the final output of the architecture.

Below we provide a more detailed description of the different stages and processes that take place in the modules of the proposed architecture (Figure 2).

A. Phase of Natural Language Analysis (NLA)

The development of the first phase can be performed through some techniques like: Gate-NLP, Open-NLP, Apache UIMA [31], [32], among others, which implement modules mentioned above. Taking into account diverse specifications and criteria such as performance, documentation and extension, we have selected the technique called GATE (General Architecture for Text Engineering [29]). This is a suite tool developed at the University of Sheffield and is a Lesser General Public License. This technique also offers an architecture that contains functionality for plugging in all kinds of NLP software.

In the same way, Gate is a platform that contains an information extraction system named A Nearly New Information Extraction System (ANNIE). This has a set of diverse modules like: sentences splitter, tokenizer, part of speech (POS) tagger, gazetteer, named entities and a co-reference tagger. Moreover, ANNIE also provides functionalities as: information extraction through Gate GUI, or can be a starting point for more specific tasks, such as machine translation, information retrieval or the one done in this project. On the other hand, Gazetteer is an important and unknown task that together with JAPE (Java Annotation Patterns Engine) are the elements used for detecting entity lists such as organizations, places, names, dates among others. In this sense, JAPE allows the establishment of grammatical rules in order to obtain words annotations, while Gazetteer contains a set of lists with words classified.

It is important to consider that GATE provides a comprehensive set of elements similar to the modules showed in this paper, however, the proposed approach adapts the modules supported by GATE and enriches them with external modules added as semantic analyzer, filtering of words, the auto-complete and spell checker module, as well as the novel approach oriented towards the use of natural language processing, to recover services which have not been developed in depth previously.

AAlready immerse in the architecture, we must assume initially that a user makes a request from his/her mobile device in NL. While user is typing the request, two modules are evaluating this input: in one side, the *SpellChecker* module is inspecting for misspelled words. If one word is identified as incorrect, the system underlines the word to warn user about his mistake. On the other hand, the *Auto-complete* module is constantly inspecting all the words typed, when it identifies a coincidence, between first letters of a word and a set of words stored in a repository, it will suggest a list of terms to the end user; then he can decide if he wants to auto-complete the word with the recommendations or not.

Then sentences of the request are received by the *Tokenizer* module, where tokenization operation starts, it obtains simple lexical units from complex sentences, by removing existing

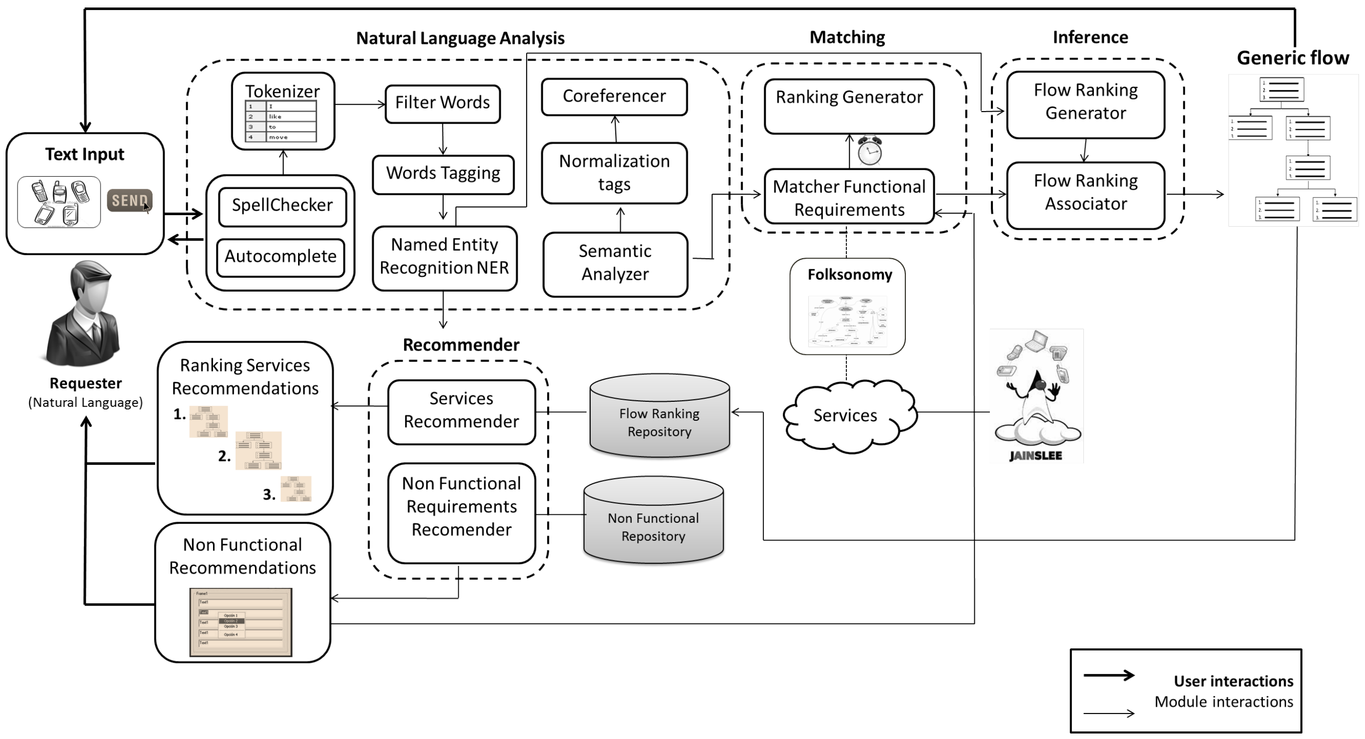


Fig. 2. Architecture of the System.

spaces or punctuation marks, admiration marks, exclamation marks and question marks. Additionally, this module corrects simple lexical errors that may arise in the request; those ignored by the user, when entered the request, or those ignored when *SpellChecker* module warned, i.e., all misspelled word that are easily identifiable. Afterward, the sentences are processed through *Filter Words* module, here, the system removes all the words that have coincidence with a set of words that have been store previously and considered as unimportant. Later, the sentence passes through *Words Tagging* module, where is pretended to classify words of the sentence according to their grammatical category. The module also aims to undertake an analysis based on linguistic rules, trying to identify and compensate syntactic and structural errors.

Once completed these operations, the request is more consistent, but remains complex. Therefore, we established the *Named Entity Recognition* module, which performs a classification between ”Control” and ”Functional” words according to its meaning, from which, control words are directed to *Flow Ranking Generator* module, whereas functional words are directed to *Semantic Analyzer* and *Recommender* modules. In this sense two categories have been established: input-output and behavior, both executed to identify special features of services. To perform words identification in each category, was necessary to establish some rules that allowed words category selection, e.g., numeric types are considered as input-output category. For the previous case, any number in the request would be classified and taken into account as input-output feature of service. This is achieved by name entities transducer, where a set of rules (JAPE) are defined.

Once all the words have been selected in different categories, it is important to consider the semantic ambiguity, for which the *Semantic Analyzer* module, identifies the correct sense according to its context, i.e., it identifies the correct sense of a word with multiple meanings in a sentence. All the possible senses are extracted from Wordnet and the way we choose the correct meaning, is through Lesk algorithm [30]. Once having the correct meaning, the most important word in the definition is selected; this term will be used in the matching phase. In *Normalization tags* module, the system obtains the respective lemmas, from all the words coming from the previous module. Lemma is the canonical form of a word, and it is used as an aid in the subsequent phase of the system. In the final stage of linguistic analysis, the *co- reference* module establishes a relationship between a possessive pronoun with the main subject of the sentence. This is done in order to understand different relationships of one subject in the user's request.

All of above modules allow an easy identification of keywords, which will represent user's requests. This stage offers to users, greater flexibility in the use of language, allowing establishing a wider range of possibilities. On the other hand, the phase identifies conditional words (e.g., *if, then, later*) and order words (sequence) (e.g., *first, second*), used in Generic Flow Generator module, mentioned at association phase.

B. Recommender phase

This phase is composed of two modules and both are executed while the matching phase performs the search of services through functional requirements. In this way, the phase initially is going to show recommendations about generic control flow with services to the user. If the user selects one recommendation, the execution of the two remaining phases would be avoided, and straightaway the process finished. Otherwise, the process continues in the matching phase. It is worth mentioning, that there is an estimated time of response, where users can select or not, any diagram recommendation. In case the time is finished, the system will understand that the user has not accepted any recommendation. The module in charged for the above task is the *Service Recommender*, which obtains a list of generic control flow with services of the repository from obtained keywords classified as functional. The second module in this phase is called *Non Functional Requirements Recommender*, this searches non-functional parameters from the *Non Functional Repository* using the obtained keywords classified as functional, the result is shown to the user, and if one or more are selected, it becomes the input of the *Ranking Generator* module, else nothing happened.

C. Matching Phase

At this stage there are two important modules, responsible for selecting the most appropriate and accurate service, according to a set of input terms: Initially, is important to consider all the described service tags, because they are semantically enriched, using different knowledge sources like Wordnet and Flickr. Thereby, is obtained a biggest set of terms that describe the services. In the same way, all input terms are semantically enriched, this process facilitates the matching process described below.

Matcher Functional Requirements use both, the terms enriched that describe services, which also are obtained from a repository, and the terms obtained from the input request. This module makes a syntactic comparison between mentioned terms, and obtains as a result an ordered list of the services that present more syntactic coincidences. In other words, this module is in charge of functional matching requirements to obtain the first service's ranking, it also considers some words classifications (input-output, behavior) and services description, that make possible the service selection process.

The second module refers to the *Ranking Generator* which use a weighting algorithm, i.e., once the first ranking of services is obtained, this module tuned the ranking with non-functional parameters, taken from the *Recommender* phase. The *Ranking Generator* module uses a timer, which provides a time out for an entry of those parameters, here user must entry those parameters which consider important to complement its request. If he/she doesn't entry anything, the system only will consider functional parameters and the ranking established would be the presented one before, otherwise, the system will obtain those non-functional parameters selected by the user.

Often users don't select the most appropriate non-functional parameters, for this reason the system will give to each non-functional parameter a weighing in the ranking generation. Although all parameters are going to be considered, the highest weighing is for those selected by end user. In summary, the ranking of non-functional parameters is obtained taking into account the selected and non-selected parameters and their respective service values, then is generate a list of services with the best non-functional values. Finally, through a factor named tuning factor, is obtained the final ranking which joins functional and non-functional parameters. The tuning factor gives greater or lesser weight to a given ranking, depending on the configuration system input made by user. This phase is also related with a Service Logic Execution Environment SLEE, specifically JAIN-SLEE, which is characterized as an execution environment with low latency, high performance and synchronously event orientation. With this is intended that, once services are discovered and ranked, is necessary to have an execution interface to each detected service, in the case where they were going to be executed in the association phase.

We argue that requests can be enriched with non-functional parameters, in order to provide optimum results that best fit to end user requirements, since such properties are very important for better understanding of the user's request, because they represent features such as quality, efficiency, availability, etc.

D. Association phase

This is the last stage of the system and begins with the *Generic Flow Generator* module. It receives as input, terms classified as *Control words* obtained from NER through *NLA* phase. The module infers a basic structure of ordered operations that represents the basic control flow, useful for the service composition process, which is performed once the most relevant atomic services have been retrieved. The *Flow-Ranking Associator* module receives as inputs the service ranking from the semantic matching phase and the basic control flow (obtained from the previous module) in order to generate the services generic control flow (stored in *Flow Ranking Repository*) which in turn represents the output of the entire system.

VI. Case Study

This section describes the functionality of the proposed architecture through an example that details each of the phases of the general process for automatic service retrieval in converged environments through natural language requests. To do so, consider the following situation. Suppose that an executive requires a service to coordinate meetings and obtain meetings reports. The executive requests from his cell phone by using natural language, the following: *"I want to receive traffic reports of Bogotá via messages, minutes before the meeting and if I have not made it to the meeting, I want to receive audio content of the decisions taken.*"

A. Information Retreival with Natural Language Analysis

1) Tokenizer: Considered as the first NLP operation to be performed in processing of the request (result 1 in the Figure 3), the result obtained by identifying each "atomic" unit is as follows. *"I - Want - to - receive - traffic - reports - of - Bogotá - via - messages - , - minutes - before - the - meeting - and - if - I - have - not - made - it - to - the - meeting - , - I - want - to - receive - audio - content - of - the - decisions - taken "*

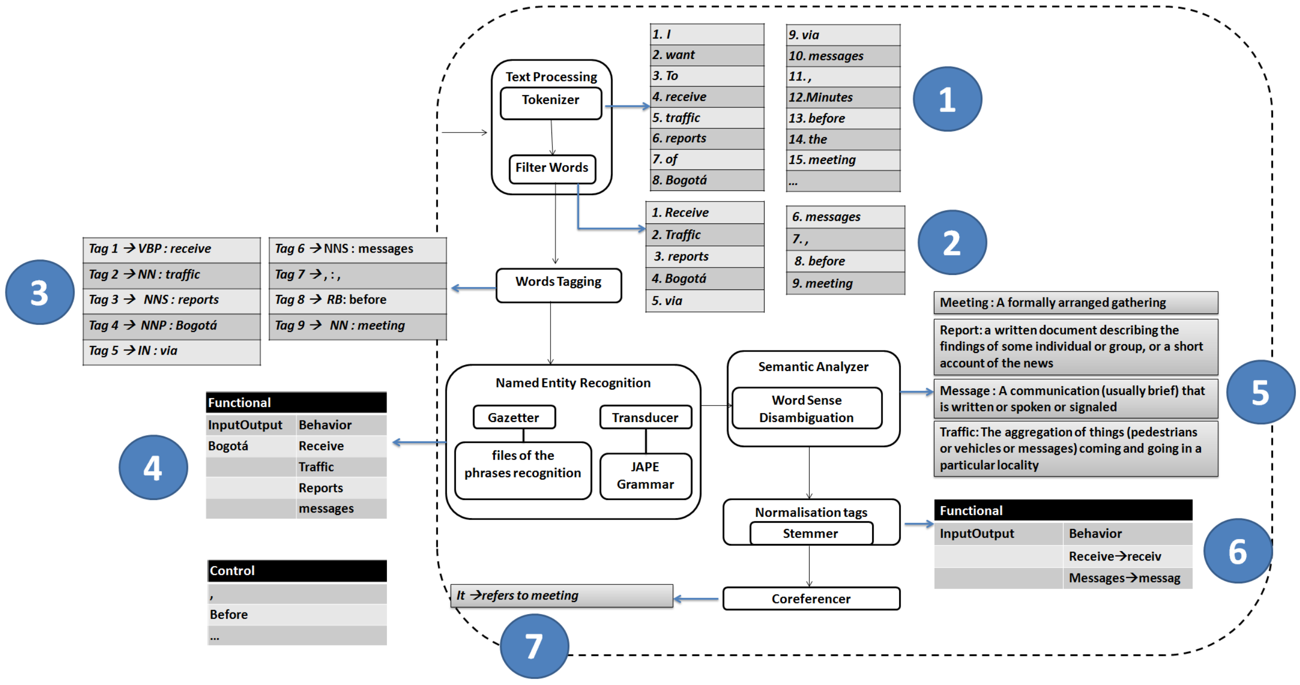


Fig. 3. Example of the Natural Language Analysis Module.

2) Filter Words: Following the idea of [17], it is necessary to remove common words of the request or words that do not add significant meaning, to not create noise in the final result, removing the following words: *"I, want, to, of, have, to, I, want, to, of"* Thus, the result is (part of the significant terms are shown as result 2 in the Figure 3): *"receive - traffic - reports - Bogotá - via - messages - , - minutes - before - the - meeting - and - if – not - made - it -the - meeting - , - receive - audio - content - the - decisions - taken"*

3) Words Tagging: Given the possibility of ambiguous words, it is necessary to perform a tagging process to help clarify this ambiguity. Thus, the words from the previous module are classified according to the grammatical category handled by GATE annotations [33]. The result obtained after performing the POST procedure (Part of Speech Tagging) is (result 3 in the Figure 3) shown in the Table I.

4) Named Entity Recognition: In the process to extract word entities with the annotated words resulting from the previous module, the system classifies words into functional or control categories, for the classification of functional categories, the terms are annotated based on service interface, i.e., through its inputs, outputs and behavior (service functionality). Therefore, if the term corresponds to an input or output parameter, then it is annotated as *"inputoutput"*, and if the term corresponds to a behavior parameter, it is annotated as *"behavior"*. The Gazetter makes the classification for matches between the terms obtained from the previous module and terms obtained from the Gazetter lists, the matching terms are annotated with *majortype* annotations, which define the type of the list, i.e., the categories *"control"*, *"inputoutput"* and *"behavior"* and *minortype* (specific annotations), that represent subcategories as: location, person, number, devices, food, etc. For this case in particular, the Gazetter annotates the word Bogotá with a majortype *"inputoutput"* and minortype *"location"*, and annotates the terms *"messages"*, *"traffic"*, *"reports"* as part of the behavior category without additional subcategories. In addition, the gazetter finds that the word *"before"* is in the list called control.lst so it is also annotated with a majortype *"control"*.

To complement the Gazetter, we use Java Annotation Patterns Engine (JAPE). JAPE provides ?nite state transduction over annotations based on regular expressions [34]. For example, we define that tokens tagged as verb by Gate (VB, VBD, VBG, VBN, and VBZ [33]) and without *majortype* annotations provided by the Gazetter, are annotated as a behavior term. The terms *"receive"* and *"meeting"*, fulfill with this rule and are annotated as behavior terms. Another rule defined with JAPE is that tokens of type *punctuation* are listed as terms of control (e.g., commas are grouped in this category).

Finally, considering the order of words, blocks of terms are created in a separated way by the annotations of control, the result is shown in the Table II.(The first Block with the control terms are represented as the result 4 in the Figure 3).

5) Semantic Analyzer: At this point, we detail the sense disambiguation of the words classified as functional, based on dictionary definitions [35], using the Lesk algorithm [36]. This algorithm is based on the assumption that words that co-occur in a phrase tend to share the same topic. In our proposal, the disambiguation process of a term is performed with the Lesk algorithm and WordNet 1.7, which takes as input the words from the NER module and counts the number of common words between the terms from the NER module and each

TABLE I
WORDS TAGGING RESULTS

No	Tag Type	Tag
1	VBP	receive
2	NN	traffic
3	NNS	reports
4	NNP	Bogotá
5	IN	via
6	NNS	messages
7	,	,
8	RB	before
9	DT	the
10	NN	meeting
11	CC	and
12	IN	if
13	RB	not
14	VBD	made
15	PRP	it
16	DT	the
17	VBG	meeting
18	VBP	receive
19	JJ	audio
20	NN	content
21	DT	the
22	NNS	decisions
23	VBN	taken

sense of the ambiguous word given by the dictionary. Finally, the highest scoring sense (the greater number of common words) is selected. For this example, ”report” can be verb or noun, and may have several meanings: *a written document describing the findings of some individual or group, a short account of the news, the act of informing by verbal, a sharp explosive sound (especially the sound of a gun firing), card a written evaluation of a student's scholarship and deportment,* etc. Thus, from which the system determines the first choice as relevant to this case. This and other disambiguated terms are shown in the result 5 of the Figure 3. Consequently, the first term of type noun, from the correct sense given by the algorithm, is added to the blocks previously created to improve the search in the matching module; therefore, in the example, the word ”document”, which correspond to the sense of the term ”report”, is added to the terms to be used in the matching and is also added to the category that belongs the term ”report”, i.e., the category behavior.

TABLE II
NAMED ENTITY RECOGNITION RESULTS

Block	Category	Terms
1	Functional-Behavior	Receive
		Traffic
		Reports
		Messages
	Functional-InputOutput-Location	Bogotá
	Control	,
2	Functional-InputOutput-Time	Before
	Control	Minutes

6) Normalization Tags: The system uses the Porter algorithm inside GATE, provided through the snowball project [37] to make the process of stemming. The Porter algorithm removes affixes using rules and lists of words following the pattern of algorithms known as affix removals [38]. Although the algorithm used in the system is one of the most simple, has proven to be as good as other algorithms in assessments of *precision* and *recall* for information retrieval [39].

In the example, the terms of interest to normalize are those classified as *behavior* or *inputoutput*. These terms represent the request made by the executive. The rules used by the algorithm are based on the *”measure”* (m) of the word, which corresponds to the number of vowels that are followed by a consonant character; for example, the term *”receive”* when m is greater than 1, the algorithm replaces the suffix e by null, resulting *”receiv”*. Other terms such as *”messages”* and *”minutes”* can accomplish several conditions of the algorithm: initially the S suffix of these terms is replaced by null, resulting ”message” and *”minute”* terms respectively, where the final result given by the algorithm are the terms *”messag”* and *”minut”*. Other terms are treated similar depending on their fulfillment with the rules defined by the algorithm, some of these terms are shown as the result 6 in the Figure 3. Finally, the terms are added to the blocks as the terms of the semantic analyzer.

7) Coreferencer: With this module, the system generates additional annotations of type co-reference, such annotations indicate the pairs pronoun/entity, where the entity is the antecedent that refers to the pronoun. In the example, *”it”* in the context *”if I have not made it to the meeting”* has a notation (Behavior type ENTITY_MENTION_TYPE with matches in the position [118, 125]) that corresponds to the word *”meeting”* indicating that the pronoun *”it”* refers to *”meeting”*.

B. Recommender

1) Service Recommender: The terms classified as functional are used to search recommendations in the *Flow Ranking Repository*, from previous results made by other users. A feature has been added to the system, common in lot of web tools and services. It consists in a simple autocomplete feature which takes as input the terms stored in the Flow Ranking Repository to recommend terms of service while the user types the request. Consider that while the user texts *”want to receive traff”*, the system will suggest the option *”traffic”* according to previous requests made by other users stored in the repository Another kind of recommendation comes after sending the request to the system, considering the case for the existence of some match with the words: *”traffic reports”* the result shown to the executive is a list of generic flows with services, outcome from previous requests by other users of the system (see Figure 4).

If the executive doesn't select any recommendation, so the remaining processing continues

2) Non Functional Requirements Recommender: This recommender searches non functional parameters from the *Non Functional Repository* using the terms classified as functional, the result considering the case for the existence of matches with the words: ”traffic reports” is as follows: *”Precision, real-time, cost ...”* The executive chooses the option ”precision” and

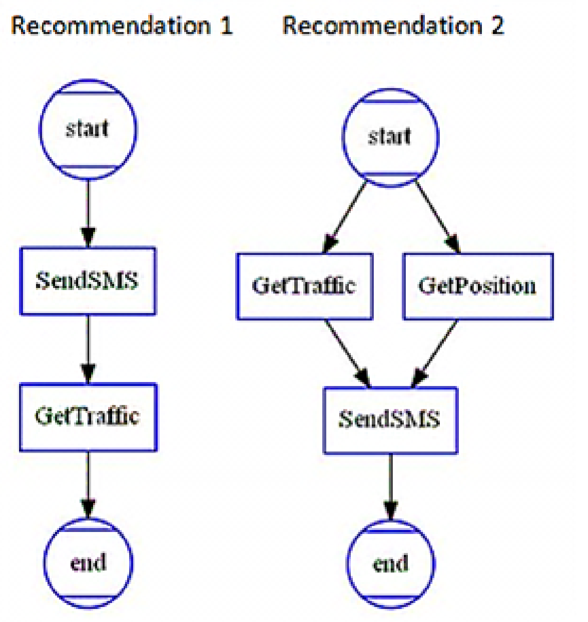


Fig. 4. Results of the Service Recommender Module

it is accepted by the system.

C. Semantic comparison between the processed request and Service Cluster

1) Matcher functional requirements: The input for this stage is represented by the output given by the NLA module (Table III). At this point, we consider that social software

TABLE III
OUTPUT GIVEN BY THE NLA MODULE

Block	Category	Terms
1	Functional-Behavior	Receive
		Traffic
		Reports
		Messages
		Receive
		Messag
		Document
		Communication
		Aggregation
		Report
	Functional-InputOutput	Bogotá
2	Functional-InputOutput	Minutes
		Minut
		Time
3	Functional-Behavior	Meeting
		Gathering
		Meet
4	Functional-Behavior	Meeting
		Meet
		It
5	Functional-Behavior	Receive
		Audio
		Content
		Taken
		Receiv

has some disadvantages (ambiguity, variation of terms, and flat organization of tags). Works as [40] exceed the limitations with

TABLE IV
RELEVANT SERVICES FOR THE RANKING PROCESS

Block	Service Name	Description
1	PostItService	Creates messages and receive new answers.
	ReportingService	Provides report data and network usage data.
	QueueService	Offers a reliable, highly scalable hosted queue for storing messages as they travel between computers.
	DealService_v1	Webservice to receive newest deals, top deals, categories, and so on.
	TopLabService	Answers on messages.
	USWeather	Gest five day weather report for a given zipcode.
2	Airport	Gets Airport Code, CityOrAirport Name, Country, Country Abbrv, CountryCode,GMT Offset Runway Length in Feet, Runway Elevation in Feet.
	HeaderTimeService	Returns the current time.
	StrikeIronRealTimeQuery	Retrieves informattion for the stocks that are traded on the Island ECN marketplace.
3	CommitteeMeetingService	Information on committee meetings of the Washington State Legislature.
4	CommitteeMeetingService	Information on committee meetings of the Washington State Legislature.
5	PostItService	Creates messages and receive new answers.
	Perceval_VoIP_manage	Downloads message audio file.
	AWSECommerceService	Exposes Amazon's product data and e-commerce functionality.

semantic technologies, so that knowledge sources enrich the tagspaces semantically. In [41], the approach addresses the fact that one tag can sometimes be mapped to different Wordnet synsets, where each synset corresponding to one meaning of the tag. Following the former idea, our system enriches the terms obtained from the *Natural Language Analyzer* module, expanding the search with sources of knowledge (Flickr and Wordnet).

In this way, the limitations of social software are partially overcome. Hence, from the terms obtained, the system classifies services: a service is classified as relevant or not depending on whether it shares some terms of input, output or behavior (service interface) with the terms of the expanded user search. Once the algorithm for classification of services is applied, a relevant set of services to each block is obtained. The services used in this example are taken from seekda information system [42], the result is presented in the Table IV.

Note that, the classification component uses the obtained services for the ranking process. Thus, the algorithm gives more importance to those services that match both input-output and behavior parameters with the user's search terms than those that only match one of the two categories. The result of

TABLE V
RANGING OFF SERVICES CONSIDERING ONLY FUNCTIONAL PARAMETERS

Block	Service Ranking
1	1. ReportingService 2. QueueService 3. DealService_v1 4. PostItService 5. TopLabService 6. USWeather report
2	1. HeaderTimeService 2. StrikeIronRealTimeQuery 3. Airport
3	1. CommitteeMeetingService
4	1. CommitteeMeetingService
5	1. Perceval_VoIP_manage 2. PostItService 3. AWSECommerceService

TABLE VI
RANGING OFF SERVICES CONSIDERING ONLY FUNCTIONAL PARAMETERS

Block	Service Name
1	1. ReportingService 2. DealService_v1 3. QueueService 4. PostItService 5. USWeather 6. TopLabService
2	1. StrikeIronRealTimeQuery 2. HeaderTimeService 3. Airport
3	1. CommitteeMeetingService
4	1. CommitteeMeetingService
5	1. Perceval_VoIP_manage 2. PostItService 3. AWSECommerceService

the ranking process considering only the functional parameters is shown in the Table V.

2) Ranking Generator: The generated ranking is subjected to a tuning based on the non-functional parameters chosen by the user. In this case, the executive chose the parameter *"precision"* of three available parameters (precision, real-time and cost). The final ranking has two components, the original ranking and the new ranking function generated with non-functional parameters, this function gives more weight to the services that have the precision as a non-functional parameter. As result we obtain the new reordered ranking (Table VI).

It is worth noting that the words are stored as terms of service interface, in order to speed up the fulfillment of future requests.

D. Retrieval-based Association

1) Flow Ranking Generator: The input for the final stage are the control terms that separate the service blocks retrieved in the previous module: *, (Control) - before (Control) - and (Control) - if (Control) - not (Control) -, (Control)*. These terms are mapped to the most primitive workflow patterns described in [43]. The sequential pattern with the terms like *and, before* or commas, draws blocks one after the other in the order in which they are defined. The parallel pattern with therms like *or*, draws the blocks in the same level and share the same root. The other pattern that we consider was the conditional pattern with therms like *if*, the pattern draws a block and bellow it, follows a parallel pattern to the others blocks.

As a result, the system creates a generic flow composed by five blocks, separated by control terms detected by the system:

Block 1 Separator Term: , (Control).

Block 2 Separator Term: before (Control).

Block 3 Separator Terms: and (Control) - if (Control) - not (Control).

Block 4 Separator Term: , (Control).

The result is shown in the Figure 5.

The reason why we choose to make a flowchart of basic components instead of standardized diagrams as UML, was because we consider that the flowchart is simpler to understand for end users. The generated graph has simple conventions: it represents the start and end nodes with a circle, the blocks of services that follow sequential or parallel patterns are represented as rectangles and the block with a conditional pattern is drawn as a rhomb.

2) Flow Ranking Associator: In this phase, the system replaces the generic blocks generated in the previous phase, for the block of retrieved services, with the exception of the start and end nodes. The result obtained is shown in the Figure 5. The conditional blocks, as the other blocks show the user the mapping of its queries in a generic flow, but the execution of the services is performed only in two ways: sequential and parallel. In the case of conditional blocks, the system executes the first service of the block in a sequential manner, the other blocks also execute the first service following one of the patterns. The execution of the services is performed by two different clients developed: one for web services, which selects the operation most similar to the service name and creates default values for the input; the other one for Telecommunication capabilites sends a Soap message to a Service Logic Execution Environment (SLEE) with the name of the service, and this executes the service with a default agent. In both cases, the responses are returned to the system. This way the user can see its logic request translated to a diagram and the response of the services. The services response is stored in a file indicating the name of the service, its response and the flow pattern that followed (sequential or parallel), this file and the graphic file are sent to the cell phone to be visualized by the user. This way, the user can retrieve services available on dynamic and varied context, see its functionality through the result after its execution and be useful to perform a formal composition process.

3) Result Storage: The terms are stored keeping a relationship with the services that were retrieved with the flow generated, given the possibility of future requests, in order to reduce processing time.

VII. EXPERIMENTAL STUDY

In this section, we present a perception of the quality of output according to the input, through two different evaluations of the first architecture phase (Analysis of Natural Language)

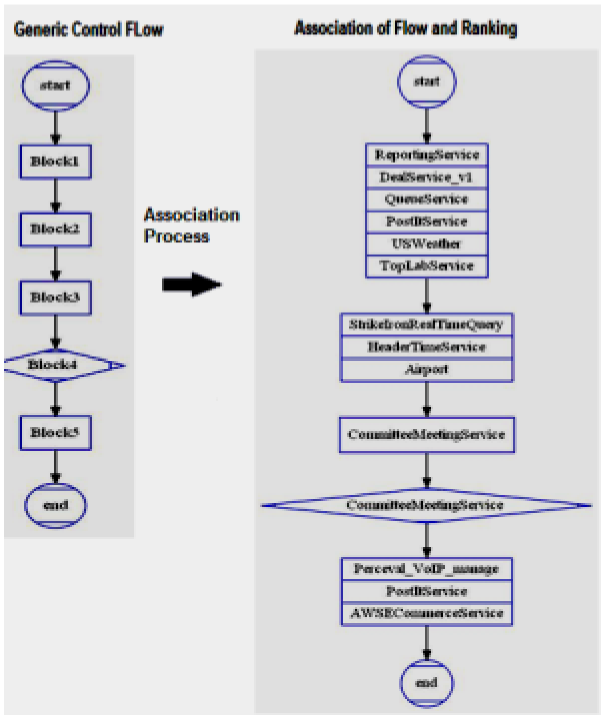


Fig. 5. Generic Control Flow

to determine also the viability of our proposal. Initially to the first evaluation, we took the most important module for the analysis of natural language requests; this module was *Named Entity Recognition* and was chosen, taking into account different features such as adaptation and relevance in the phase. The assessment was focused on measuring the accuracy of the system to classify entities that have already been defined by experts. Based on the measures defined in [44], [45]: *Precision* is the ratio of the number of correct words annotated as the entity by the system and the number of annotations of the entity given by the system, and the *Recall* is defined as the ratio between the number of correct words annotated as the entity by the system and the number of annotations of words as a specific entity given by the user. To develop the evaluation was used a corpus, obtained from different descriptions of the Google Play services. We use as a search parameter key words that were on the lists of the Gazetter. A Java program was implemented, to create the corpus from the android services description. The program removes useless characters leaving 7355 words.

The *Behavior* entities considered are: *message, call, maps, email, music, send, video, messenger* and *sms*; the *Input/Output* entities considered are: *location, person, number* and *time*; and the *Control* entities: *before, however, if, or*, and *while*.

With this in mind, experts who evaluated the system manually attributed annotations to the entities that describe the service interface (Input-Output, Behavior). Such assignments were performed according to the types of entities or words specified. The results for the NER evaluation are shown in Table X, where: *Entity Made by the System* (EMS), *Correct Entity by the System* (CES) and *Entity Given by the User* (EGU) The second one is a general evaluation about user's

TABLE VII
NER EVALUATION RESULTS

Entity	Precision	Recall	EMS	CES	EGU
message	1	0,67	4	4	6
call	1	1	9	9	9
maps	1	1	14	14	14
email	1	0,67	6	6 9	
music	1	0,89	17	17	19
send	1	0,94	18	18	19
video	1	0,52	11	11	21
messenger	0,9	0,36	10	9 25	
sms	0,74	1 23	17	17	
location	0,17	0,01	6	1	78
person	0,06	1	117	7	7
number	0,62	1	141	88	88
time	0,47	0,19	38	18	92
before	1	1	3	3	3
however	1	1	1	1	1
if	1	0,67	4	4	6
or	0,97	1	39	38	38
while	1	1	2	2	2

satisfaction, unlike to the *precision* and *recall* evaluation presented before. At first, we collect a set of forty five (45) requests in natural language from several Web users, collected from a survey that asked them, about the possible requests they would do to a retrieval service system in their mobile devices. In this vein, is important to consider that system will only understand readable and meaningful requests; however, in the case where a meaningless or misspell sentences be entered, the system will try to process them, in order to deliver consistent results to the user's request. All requests were categorized in high, medium and low scope, i.e., according to its content. *Low scope* represents requests without sense or without real services which cannot be implemented because its complexity (e.g., How do I fix this problem?), *medium scope* refers to incomplete or ambiguous service requests, which can be executed, but not in an ideal way (e.g., How do I get to "x site"?), and finally *high scope* request, are those which contains an adequate and reliable information of services (as the request shown in the example given). From former categories were selected randomly eight (8) answers, which became in input's system. To each request was obtained a set of properties and characteristics (e.g., word's grammatical category, number of identified services, etc.) which were being given by the system. To the evaluation, two important aspects were considered. In one side, we looked at classification of input words, organized in these groups: *Behavior, Input/Output* and *Control*. In the other side, we review the number of services identified and the words included in them.

Once taking all output system, ten human experts, with high knowledge in this subject, evaluated the outputs obtained by system. Experts were selected from research group named GIT (Group of Telematic Engineering) from Cauca's University, who work in the area of telematic services.

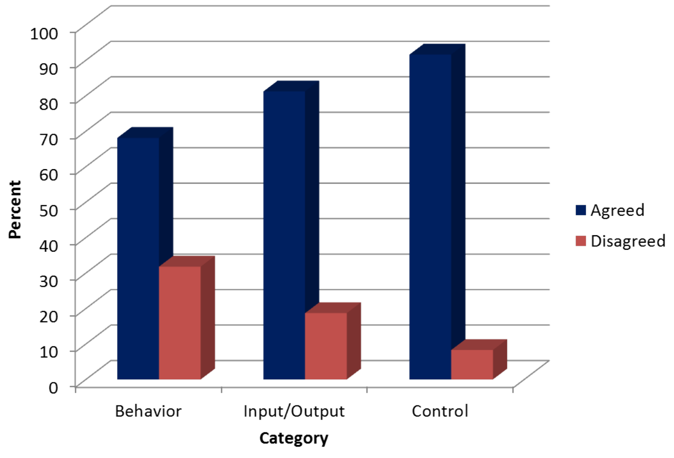


Fig. 6. Word classification satisfaction

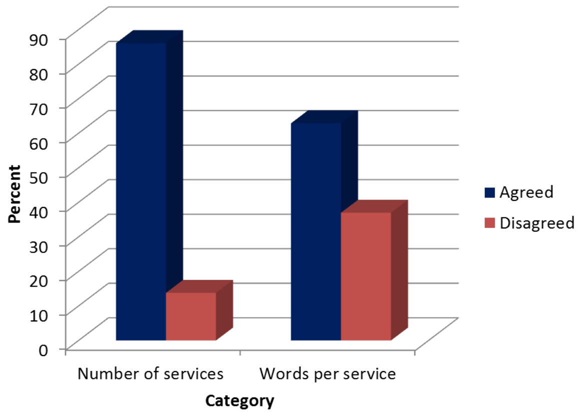


Fig. 7. Service classification satisfaction

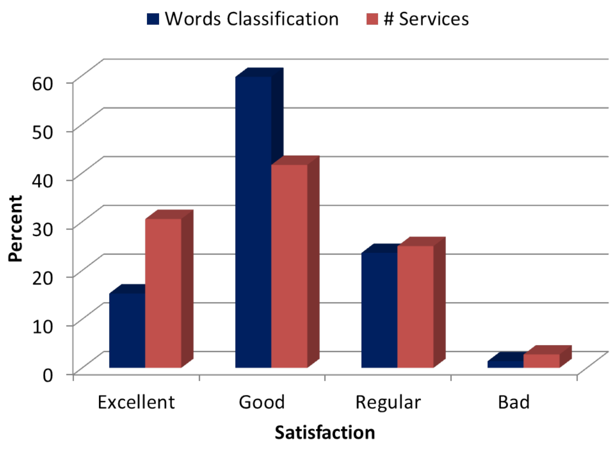


Fig. 8. General Evaluation

Historical examples of this kind of evaluation methodologies are exemplified by ”the fiuma comparative evaluation of parsers of French (Abeillé), ”the fi or competition of morphological analyzers for German” or ”the Morpholympics (Hauser)” [39]. Thus, in order to determine trends of system's outputs, a survey was developed and applied to the above mentioned experts, in December of 2011. The survey applied to experts consisted of eight questions; each one represented the processing of only one request, and shows the classification of words from the sentences and the identification of the number of services. Two different sections can be recognized in the survey. The first one identifies words from input sentences that can be classified into *behavior, input/output* and *control* categories: *Control* words, *Behavior* words and *Input/Output* words. Experts have chosen, if they were agreed or disagreed with each proposed classification system. Percents of satisfaction are showed in Figure 6, where it is possible to show that higher percent of agreed selection was in Control category with a 91.6% and the lower one was presented in Behavior category with 68.1%. Also, in Input/Output category the 81.2% were agreed with the proposed selection. In the same way, experts were asked if they were agreed or disagreed with the number of services identified and the words detected in each service. Results in Figure 7, showed that 86.1% of experts considered that the number of services identified were optimums, while 13.8% not, and on the other hand 62.9% contemplate as adequate the classification of words in each service against a 37.0%. The other section of the survey, evaluates in general terms, whether the system is consider as excellent, good, regular or a bad system, according to all identification, selection and classifications parameters that were used. For this, the Figure 8 shows both outcomes, degree of satisfaction in number of services and words classification. In both cases was observed that most of the experts considered the system as good with a total of 41.6% and 59.7% respectively and only a minority considered a bad system with 2.7% and 1.3%. From this study we are able to determine if word's classification was optimums or not. Thus, results showed that *Behavior* classification had the lowest level of acceptance by experts, demonstrating the need to improve all the set of words and rules that govern and compose this category. This also implies that for subsequent tasks, specifically to development of matching phase, we must establish a higher percent of weigh to all *Input/Output* words, because they obtained better acceptance than *behavior*. Regarding identification of *Control* words, the percent of agreed is high, ensuring that control flow will be correctly established. In general terms the categorization of a good system by majority of experts, shows the adequate performance of the system, although can be improved in some aspects like rules definition, Gazetters lists, patterns and others related to increase exactitude levels.

VIII. Conclusion

This study presented a novel approach which facilitates the service retrieval process in converged environments, reducing complexity and habitual restrictions on incoming requests. Thereby, inexperienced users can express their needs in natural language, unlike other solutions that use templates restricting user's expression. The system also takes into account from user request, functional and non-functional parameters, to enrich and give higher accuracy to retrieval service process.

The proposed system, processes requests made from mobile devices, and delivers a generic control flow as output, which contains the most suitable services to the users. From user's request analysis, we obtained a set of words classified in three different groups: Behavior, Control and Input/Output, which are used into matching module, to compare with characterization services tags. In the same way the folksonomy-based matchmaking, allowed an improvement of the system, by adapting to dynamic vocabulary and reducing the one used by end users.

In the proposed experimental study, where was assessed the quality of the outputs system, trends among experts revealed good acceptance in most of evaluated aspects, all above 55%. However, the survey also showed that Behavior classification must be improved to increase performance and accuracy of service retrieval.

The study of diverse techniques and related works, showed that implementing natural language analysis for service selection process, is efficient since it enhances and facilitates the processes, reducing the time required, complexity and increasing the number of eligible users. On the other hand, the implementation of a recommendation phase and auto-complete module, facilitated the understanding and interaction with the user, since it leads to formal elements in an informal request, without establish limitations in capturing requirements.

The proposed semantic words enrichment, increased accuracy in syntactic matching process, because when users enter words in a request, often they do not match the terms that describe services in the repository. Finally, the inclusion of non-functional properties in the service retrieval process, allows obtain not only the most related services to requests users, but also ensures that these services are the most optimal in terms of availability, response time documentation and other non-functional parameters.

IX. Future Work

As a complementary work, we can consider the enhancement of natural language phase, considering a module with more specific rules, to obtain higher accuracy of retrieval words. Natural Language Analysis phase could be also improved adding a module which detects meaningless request or incoherent and misspelled requests, to filter them and avoid the whole retrieval process of the system. As future work, the prototype can be extended to other language like Spanish, German, Mandarin, etc. Also the number of services and their respective descriptions (Functional and Non-functional properties) can be expanded involving other types of repositories which can enrich the current records.

As shown in the description, the system only executes services in an isolated manner, so for future work could be consider the execution of services taking into account the interfaces of the services such as, input/output features, respecting the order indicated in the generic control flow.

The generation of control flow, can be improved taking into account service interfaces. Finally for future work can be established and developed the other phases that composed the architecture as recommender phase, matching phase and association phase, to complete all functionality for which it was designed.

Acknowledgment

The authors would like to thank Universidad del Cauca, COLCIENCIAS and TelComp2.0 Project for supporting the Research of Camilo Pedraza and Julián Zúñiga

References

[1] E. C. Pedraza, J. A. Zuniga, L. J. Suarez Meza, and J. C. Corrales, "Automatic service retrieval in converged environments based on natural language request," in *SERVICE COMPUTATION 2011*, ser. 978-1-61208-152-6, Rome, Italy, September 25 2011, pp. 52–56.

[2] F.-C. Pop, M. Cremene, M.-F. Vaida, and M. Riveill, "On-demand service composition based on natural language requests," in *Sixth International Conference on Wireless On Demand Network Systems and Services.*, ser. WONS'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 41–44. [Online]. Available: http://dl.acm.org/citation.cfm?id=1688899.1688905

[3] A. Bandara, "Semantic description and matching of services for pervasive environments," Ph.D. dissertation, University of Southampton, 2008. [Online]. Available: http://eprints.ecs.soton.ac.uk/16403/

[4] ITU-T, "Series y: Global informationinfrastructure, internet protocol aspectsand next-generation networks," INTERNATIONAL TELECOMMUNICATION UNION, Tech. Rep., 2001.

[5] D. Moro, D. Lozano, and M. Macias, "Wims 2.0: Enabling telecom networks assets in the future internet of services," in *Proceedings of the 1st European Conference on Towards a Service Based Internet*, ser. ServiceWave '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 74–85.

[6] ITU-T, "Converged services framework functional requirements and architecture," INTERNATIONAL TELECOMMUNICATION UNION, Tech. Rep., 2006.

[7] J. M. E. Carlin and Y. B. D. Trinugroho, "A flexible platform for provisioning telco services in web 2.0 environments," in *Fourth International Conference on Next Generation Mobile Applications, Services and Technologies.* Washington, DC, USA: IEEE Computer Society, 2010, pp. 61–66. [Online]. Available: http://dx.doi.org/10.1109/NGMAST.2010.23

[8] G. Bond, E. Cheung, I. Fikouras, and R. Levenshteyn, "Unified telecom and web services composition: problem definition and future directions," in *Proceedings of the 3rd International Conference on Principles, Systems and Applications of IP Telecommunications*, ser. IPTComm '09. New York, NY, USA: ACM, 2009, pp. 13:1–13:12. [Online]. Available: http://doi.acm.org/10.1145/1595637.1595654

[9] S. Hagemann, C. Letz, and G. Vossen, "Web service discovery - reality check 2.0," in *Proceedings of the Third International Conference on Next Generation Web Services Practices*, ser. NWESP '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 113–118. [Online]. Available: http://dx.doi.org/10.1109/NWESP.2007.31

[10] S. Kirati, "A demonstration on service compositions based on natural language request and user contexts," Master's thesis, Norwegian University of Science and Technology (NTNU), Department of Telematics, June 2008.

[11] E. Al-Masri and Q. H. Mahmoud, "Discovering the best web service: A neural network-based solution." in *SMC*. IEEE, 2009, pp. 4250–4255.

[12] A. V. Riabov, E. Boillet, M. D. Feblowitz, Z. Liu, and A. Ranganathan, "Wishful search: interactive composition of data mashups," in *Proceedings of the 17th international conference on World Wide Web*, ser. WWW '08. New York, NY, USA: ACM, 2008, pp. 775–784. [Online]. Available: http://doi.acm.org/10.1145/1367497.1367602

[13] K. Bischoff, C. S. Firan, W. Nejdl, and R. Paiu, "Can all tags be used for search?" in *Proceedings of the 17th ACM conference on Information and knowledge management*, ser. CIKM '08. New York, NY, USA: ACM, 2008, pp. 193–202. [Online]. Available: http://doi.acm.org/10.1145/1458082.1458112

[14] Apple. (2011, November) Getting to know siri. [Online]. Available: http://media.wiley.com/product_data/excerpt/80/11182992/1118299280-40.pdf

[15] (2012) iphone user guide. [Online]. Available: http://manuals.info.apple.com/en_US/iphone_user_guide.pdf

[16] IBM. (2011, August) What is watson? [Online]. Available: http://www-03.ibm.com/innovation/us/watson/what-is-watson/index.html.

[17] A. Bosca, F. Corno, G. Valetto, and R. Maglione, "On-the-fly construction of web services compositions from natural language requests." *JSW*, vol. 1, no. 1, pp. 40–50, 2006.

[18] C. Christophe, "Specification of pro-active service infrastructure for attentive services," SPICE, Tech. Rep., 2007.

[19] S. Angeletou, "Semantic enrichment of folksonomy tagspaces," in *Proceedings of the 7th International Conference on The Semantic Web*, ser. ISWC '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 889–894. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-88564-1_58

[20] M. Magableh, "A generic architecture for semantic enhanced tagging systems," Ph.D. dissertation, Montfort University, 2011.

[21] G. Maciej, C. Giacomo, P. Marcin, and G. Maria, "Wscolab: Structured collaborative tagging for web service matchmaking." in *WEBIST (1)*, J. Filipe and J. Cordeiro, Eds. INSTICC Press, 2010, pp. 70–77.

[22] S. E. Antonio Javier, "Open platform for user-centric service creation and execution," Telefónica I+D (ES), University of Valladolid (ES), Davidov, (BG), Ericsson (ES), Huawei (CN), IRIS (IT), JBoss (CH), Alcatel (DE), NEC (PT), Politecnico di Torino (IT), Portugal, Telecom Inovação (PT), Telecom Italia (IT), University of Madrid (ES), Tech. Rep., 2008.

[23] C. Christophe, "Service platform for innovative communication environment," France Telecom (F), Alcatel (F), DoCoMo Communications Laboratories Europe (D), Telefonica (ESP), Telecom Italia (I), Telenor (NOR), Siemens (D,A), Ericsson (NL), Nokia (FIN), Stichting Telematica Instituut (NL), NEC Europe (UK) , Bull (F), Fraunhofer FOKUS (D), University of Kassel (D), Alma Consulting Group (F), University of Brussels (B), IRIS (I), Neos (I), University of Surrey (UK), Norvegian University of Science and Technology (NOR), Politecnico di Torino (I),Telekomunikacja Polska (POL), Tech. Rep., 2008.

[24] N. Jorg, L. Klostermann, F. loannis, S. Ulf, d. R. Frans, and O. Ulf, "Ericcson composition engine, next-generation in." Ericcson, Tech. Rep., 2009.

[25] ITU-T, "Enhanced telecom operations map (etom) the business process framework," INTERNATIONAL TELECOMMUNICATION UNION, Tech. Rep., 2004.

[26] H. Jiejin, "A practical approach to the operation of telecommunication services driven by the tmf etom framework," Master's thesis, Universitat Poliècnica de Catalunya, 2009.

[27] TmForum. (2010) Information framework (sid)in depth. TmForum. [Online]. Available: http://www.tmforum.org/InformationFramework/6647/home.html.

[28] Yahoo. (2012) Pipes. [Online]. Available: http://pipes.yahoo.com/pipes/

[29] (2012) Flickr de yahoo. [Online]. Available: http://www.flickr.com/Flic

[30] A. Maaradji, H. Hacid, R. Skraba, A. Lateef, J. Daigremont, and N. Crespi, "Social-based web services discovery and composition for step-by-step mashup completion," in *Proceedings of the 2011 IEEE International Conference on Web Services*, ser. ICWS '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 700–701. [Online]. Available: http://dx.doi.org/10.1109/ICWS.2011.122

[31] H. Cunningham, "Gate, a general architecture for text engineering," *Computers and the Humanities*, vol. 36, pp. 223–254, 2002, 10.1023/A:1014348124664. [Online]. Available: http://dx.doi.org/10.1023/A:1014348124664

[32] Apache. (2010) Apache opennlp developer documentation. [Online]. Available: http://opennlp.apache.org/documentation/manual/opennlp.html

[33] GATE. (2011) Part-of-speech tags used in the hepple tagger. [Online]. Available: http://gate.ac.uk/sale/tao/splitap7.html

[34] (2011) Developing language processing components with gate version 6. [Online]. Available: http://gate.ac.uk/sale/tao/splitap7.html

[35] N. Iulia, "Desambiguación semántica automática," Ph.D. dissertation, University of Barcelona, 2002.

[36] M. Lesk, "Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone," in *Proceedings of the 5th annual international conference on Systems documentation*, ser. SIGDOC '86. New York, NY, USA: ACM, 1986, pp. 24–26. [Online]. Available: http://doi.acm.org/10.1145/318723.318728

[37] Snowball. (2011) Stemming algorithms. [Online]. Available: http://snowball.tartarus.org/

[38] W. B. Frakes, "Term conflation for information retrieval," in *Proceedings of the 7th annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '84. Swinton, UK, UK: British Computer Society, 1984, pp. 383–389. [Online]. Available: http://dl.acm.org/citation.cfm?id=636805.636830

[39] D. A. Hull, "Stemming algorithms: a case study for detailed evaluation," *J. Am. Soc. Inf. Sci.*, vol. 47, no. 1, pp. 70–84, 1996.

[40] P. Mika, "Ontologies are us: A unified model of social networks and semantics," *Web Semant.*, vol. 5, no. 1, pp. 5–15, Mar. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.websem.2006.11.002

[41] D. Laniado, D. Eynard, and M. Colombetti, "A semantic tool to support navigation in a folksonomy," in *Proceedings of the eighteenth conference on Hypertext and hypermedia*, ser. HT '07. New York, NY, USA: ACM, 2007, pp. 153–154. [Online]. Available: http://doi.acm.org/10.1145/1286240.1286282

[42] Seekda. (2012) Web service search. [Online]. Available: http://webservices.seekda.com/search

[43] P. Ahana, "Workflow : Patterns and specifications," Master's thesis, Indian Institute of Technology, 2007.

[44] P. Patrick, C. Stéphane, and H. Lynette, "Principles of evaluation in natural language processing," *TAL*, vol. 48, p. 23, 2007.

[45] M. King, "Evaluating natural language processing systems," *Commun. ACM*, vol. 39, no. 1, pp. 73–79, January 1996. [Online]. Available: http://doi.acm.org/10.1145/234173.234208