A Distributed Multi-Session Dialog Manager with a Dynamic Grammar Parser

Vincenzo Catania , Raffaele Di Natale , Antonio Rosario Intilisano and Andrea Longo
Dipartimento di Ingegneria Elettrica Elettronica e Informatica
Università degli Studi di Catania, Catania, Italy
vincenzo.catania@diiei.unict.it

Dipartimento di Ingegneria Elettrica Elettronica e Informatica
Università degli Studi di Catania, Catania, Italy
raffaele.dinatale@diiei.unict.it

Dipartimento di Ingegneria Elettrica Elettronica e Informatica
Università degli Studi di Catania, Catania, Italy
antonio.intilisano@diiei.unict.it

Dipartimento di Ingegneria Elettrica Elettronica e Informatica
Università degli Studi di Catania, Catania, Italy
andrea.longo@diiei.unict.it

Abstract*:*

In recent years the Spoken Language Processing field has been rapidly growing. Several human-machine systems with different characteristics have been developed, most of them without a distributed and scalable architecture. For instance, Olympus framework at the current version does not support multi-session. In addition, it uses the Phoenix parser without a dynamic grammar. In this paper we design a distributed, robust, scalable and multi-session architecture applied to a spoken dialog system along with a dynamic and self-learning Natural Language Understanding (NLU) mechanism. It is based on an extension of the basic Olympus framework. The implementation we propose, called OlympusP2P is built on a distributed and highly context-sensitive framework for applications that aim at interacting with users by using natural languages. OlympusP2P will soon available for researcher and application developer.

Keywords : *Multi-Session, Dynamic Grammar Parser, Olympus*

I. INTRODUCTION

In recent years the Spoken Language Processing field [1] has been rapidly growing. Several human-machine systems with different characteristics have been developed, most of them without a distributed and scalable architecture. For instance, Olympus framework [2] at the current version does not support multi-session. In addition, it uses the Phoenix parser without a dynamic grammar [3]. In this paper we design a distributed, robust, scalable and multi-session architecture applied to a spoken dialog system along with a dynamic and self-learning Natural Language Understanding (NLU) mechanism. It is based on an extension of the basic Olympus framework. The implementation we propose, called OlympusP2P is built on a distributed and highly context-sensitive framework for applications that aim at interacting with users by using natural languages. In order to manage different sessions, our solution introduces a multi-session layer with a partially

decentralized Peer to Peer (P2P) network, which is composed by Olympus Super Peer nodes and Olympus Peer nodes. Olympus Super Peer nodes have a customizable amount of free slots in which we can assign new normal Olympus Peer nodes running Olympus. This makes the multi-session possible and provides the system with the capability of dynamic nodes allocation depending on context necessities or network load. To make Olympus grammar more dynamic, at compile time, the Olympus HUB module [4] receives an event and it informs the parser to load the new Grammar files without stopping and restarting the Parser module. In addition we introduce a self-learning mechanism that checks if unknown words match certain criteria (e.g. synonymous of known words). When one of the criteria is triggered the new word is automatically included in the grammar and recognized by the parser. The proposed architecture is described in terms of robustness, scalability and it is compared to the standard Olympus framework. OlympusP2P is written in C# with Microsoft Windows Communication Foundation (WCF) technology and will soon available for researcher and application developer.

II. The Spoken Dialog Systems

The main goal of a spoken dialog system is to put a human at the center of the interface between the device and the human. The device may provide the human with all the required information or perform a specific request in the way that is most natural for him: speaking. Over time there has been a substantial evolution of these systems: initially they interacted with one user sitting in front of a computer that ran the system, then moving on to solutions based on telephone interface. Today, thanks to the spread of mobile devices you no longer need to be sitting in front of your PC. This then allows these systems to simultaneously interact with millions of users. The applications of a system are numerous: they range from the search of a restaurant, to book a hotel room, to dial a phone number or to listen to music. Since the early 1990s, many spoken dialog systems have been developed in the commercial domain to support a variety of applications in telephone-based services.

III. MULTISESSION

While the main goal of the recent Spoken Dialog Systems was to make the interaction between user and system as natural as possible, the new goal is to provide a multi-session interaction with multiple users: in fact, in a real condition, the system can dialog with multiple users by using multi-sessions. In this paper, we introduce a novel approach for modeling and managing the multi-session dialogue in a spoken dialogue system able to support a consistent number of concurrent interactions and to adapt its knowledge taking into account what globally happened and happens inside the several sessions. These goals are achieved by using the P2P architecture that manages multiple instances of Olympus, and also the ability to globally share a self-learning grammar updated at real time by what happens in the individual sessions.

IV. Olympus

Olympus is a complete framework for implementing spoken dialog systems created at Carnegie Mellon University. Olympus incorporates the Ravenclaw dialog manager [5], which supports mixed-initiative interaction, as well as components that handle speech recognition (Sphinx), understanding (Phoenix) and generation (Rosetta). Olympus uses a Galaxy [4] message-passing layer to integrate its components and supports multi-modal interaction. As shown in Fig. 1, the Galaxy architecture is a constellation of Galaxy *Servers*, which communicate with each other through a central Galaxy *Hub*.

Figure 1: Olympus architecture

In *Olympus*, most of the agents are Galaxy *Servers*. Specific functions, for instance dialog planning, input processing, output processing, error-handling, etc. should be encapsulated in subcomponents with well-defined interfaces that are decoupled from domain-specific dialog control logic. Olympus users are able to inspect and modify each of these components individually, towards their own ends. For each Olympus application we need to specify **resources** like grammar, language model and dialog task specification. These resources serve as a basis of knowledge that is necessary to understand and to interact with users.

V. RELATED WORKS

Olympus Framework does not natively support multiple sessions. Our implementation, Olympus P2P introduces a new architecture that enhances Olympus Framework's features while adding a multisession support with shared resources. Take into consideration the following aspect: a single Olympus instance represents a single point of failure; a single Olympus instance is not able to "increase" or "decrease" the number of available sessions in function of the user's needs and availability of hardware/software resources and it is not able to replicate the active sessions in function of new users requests; the client must know the address of each Olympus session.

Table 1 shows the comparison between Olympus P2P and Olympus/Raven Claw. Both implementations are Multi-Modal, Multi-Domain, Multi-Language and open source. Olympus P2P introduces features such as Multi-Session and Self-Learning Dynamic Grammar. These two last features introduce the concept of shared resources. The system-user interaction is enhanced because the common resource can be updated by all active sessions. This creates new self-learning scenarios in which a common resource is continuously updated by different sessions.

Tool	Multi-Session	Self-Learning Dynamic Grammar	Multi-Modal	Multi-Domain	Multi-Language	Shared Resources	Open Source
OlympusP2P	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Olympus/Raven Claw	No	No	Yes	Yes	Yes	No	Yes

Table 1: Comparison between OlympusP2P and Olympus/RavenClaw

VI. OLYMPUS P2P

Olympus P2P concurrently serves more users allowing the sharing of resources among all sessions. If one of the Olympus sessions is able to update its knowledge during the interaction with the user, then all other

sessions will benefit from it. Moreover, allocating dynamically new Olympus sessions depends on the number of users. This becomes necessary in large scales systems. Our goal is not only to manage multiple, scalable and robust instances and to face the growing demand for sessions by users, but also to use the parallelism of several active sessions to share useful resources in order to improve the user interaction.

The new component, implemented by a WCF Technology is based on a partially decentralized P2P Network. Olympus Super Peer nodes play an important role in this network, carrying out specific functions, such as a distributed search and Olympus session discoveries among multiple peers around the network. This is completely transparent to the client application. The grammar and other resources are shared by all Olympus Super Peer nodes, unlike client-server architectures in which the server shares all contents and resources. P2P is more reliable since the central dependency is removed. Failure of one peer does not affect the work of other peers. Specifically we have two types of nodes, as shown in Fig. 2:

- **Olympus Peer (OP)**: This is the interface to the client application (client login, client logout, exchange of information). It contains one or more Olympus active instances and it is connected to a single Olympus Super Peer.
- **Olympus Super Peer (OSP)**: This deals with the research of active sessions in the network. It knows at least another OSP in the network. It maintains a list of active peers with its active sessions and it can perform a query only to the known OSP.

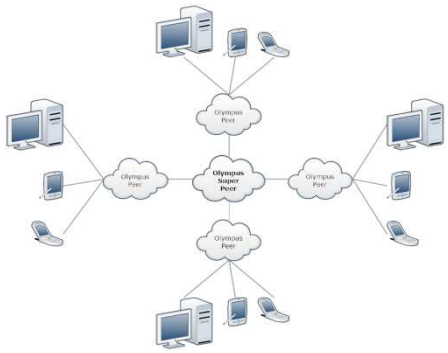

Figure 2: Olympus Super Peer and Olympus Peers network

A. OlympusP2P Architecture

In this section we explain the architecture in detail. For every OSP we have a set of OPs that are connected with a number of Olympus active sessions which serve a set of clients. Each client will only know a limited set of OPs in the network (the list can be centralized or available to the client at the time of installation). After that the authentication has been performed, if the OP contacted by the client has free sessions of the type specified by the client, then it accepts the connection. If the OP does not have free sessions, it performs a query to the OSP which will query the other OPs. The client will communicate through the same OP contacted and it will receive the URL of the OP with a free active session. The search is performed among a "network of OSPs" with techniques derived from flooding broadcast. The messages are exchanged only between the same OSP. Furthermore, each OSP contains the grammar used by its OP, which through the mechanism explained below can be updated in every iteration. Fig. 3 shows how the OSP at each update informs the other OSPs in the network and its own OPs that the grammar has changed. Finally the compiled grammar will be downloaded and

updated. In this way all sessions in the network will benefit from real-time updates of grammar in the current active sessions.

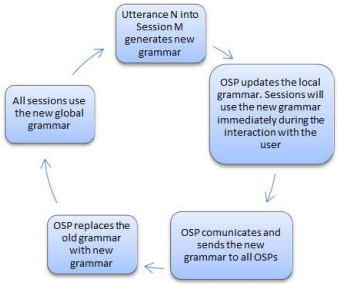

Figure 3: Grammar updating mechanism

As shown in the Fig. 4, the idea is to create an additional software layer between application clients and Olympus, with the main goal to increase the active available sessions.

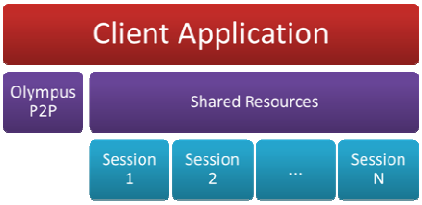

Figure 4: Olympus P2P Architecture

These goals are made possible because of the P2P architecture that manages multiple instances of Olympus and the ability to globally share a self-learning grammar updated in real time by what happens in the individual sessions.

B. Self-learning Dynamic Grammar

The introduction of speech recognition systems and spoken language generators has led to a different approach where users can have a “natural” conversation with the system. In these systems the parser has a central role, typically being the bond between the speech recognizer software and the language-processing unit. The basic role of the parser is to give a meaning to user inputs. Many parsing algorithms have been developed enclosing syntactic and semantic rules inside the grammar. The pursuit of always having a better and more natural interaction brought the development of more complex grammars. The production of an efficient, self-learning and auto generating grammar requires a high amount of human and temporal resources. Different algorithms and approaches have been adopted in the literature of natural languages. These can be clustered in different categories according to different approaches or different grammar types. For instance, the Olympus framework uses the Phoenix parser which was developed by the University of Colorado in 2002 to develop easy and robust NLU systems. As Minker and Chase [6] explained, the Phoenix parser is based on case grammar which analyze the syntactic structure of utterances by studying the combinations of verbs and their related nouns. The main characteristics of case grammars are the possibility of easily sharing the syntactic patterns and grouping related concepts or features together. As shown in Fig. 5 the Phoenix parser is based on frames. Every input word is mapped into a sequence of case frames composed by a set of slots. Each slot has a specified context-free grammar (CFG) that describes the possible content of the slot. The parsing mechanism consists on a

comparison between given input words and frames, according to the rules defined in the grammar. At the end of the process the chart consists of a sequence of slots containing the semantic parse tree of the sequence of input words.

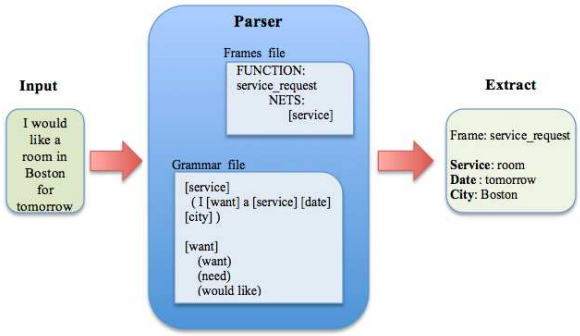

Figure 5: Parser pipeline

1) Description

The Olympus framework helps to design, to implement and to test spoken dialog systems and conversational agents. The Phoenix parser needs a pre-compiled grammar; therefore any spoken dialog system built over Olympus uses a grammar which cannot be easily updated. The grammar compiler creates the output compiled grammar files from a set of grammar specification files (.gra, .class). The system has to be stopped and the grammar has to be re-compiled every time that a new word is added to the system knowledge. In a research environment this is acceptable; however in the industrial field this mechanism is tedious especially for those companies who want the service always available without delays. Thanks to our mechanism, when a new word is added to the system knowledge, all active sessions in the network will automatically use the new grammar. Our implementation, OlympusP2P, introduces a new dynamic grammar mechanism based on a new compiling method along with a self-learning algorithm and it allows the grammar to be updated without having to stop the system. The idea is to build a new apposite Olympus module that has the responsibility of updating and then compiling the grammar. This new module, like other Olympus modules, can be implemented as a Galaxy server. Therefore the entry point can be reached by using an IP address and a port number. The communication between the new module and the Phoenix parser can be accomplished in two ways: by using direct sockets between them or by using the Olympus HUB module as a dispatcher of events. In the first case, the overhead of the communication does not affect the dispatching of events by the HUB module; therefore the Olympus framework performances are not affected. In the second case, we use the basic Olympus framework mechanism of sending frames to the HUB module. When the grammar specification files are updated and re-compiled, the new module informs the HUB that a new compiled grammar is available, and then the HUB informs the Phoenix parser to load the new grammar. This solution introduces two new frames to be dispatched over the Olympus framework network. Realistically, at the beginning these new frames will be exchanged with a high frequency due to the fact that it is more likely that there are new words not contained in the grammar. The frame exchange will occur more rarely when the grammar reaches a steady state. Therefore at a regime state, this solution does not affect the general overhead in terms of performance and latency of messages and so the system maintains an acceptable response time.

2) Algorithm

Our implementation changed the way in which the grammar is loaded by the Phoenix parser, making it possible to load the grammar and reinitialize all the structure currently in the memory at run time. In this way the system can continuously update the grammar without having to stop and restart its activities. Hence, even while conversing with a user the grammar can be updated and all the next input utterances will be processed with the new grammar. The pursuit of self-learning grammar has characterized a big portion of the natural language processing and in general the artificial intelligence field. Many unsupervised, semi-supervised or supervised algorithms have been developed in the past years. Our approach represents a simple way to achieve the goal of populating basic grammar notions in a specific context. The purpose of our implementation is to enlarge the initial grammar database with all the new terms that are somehow related to those already in the grammar specification files. The basic idea is to correlate these words according to some criteria (for example, synonyms). When one or more criteria are triggered, the self-learning grammar module updates the grammar files. The set of criteria that we chose will affect the grammar update frequency rate. Therefore, the less strict the criteria are, the less frequently the grammar will be updated. Instead, if we set weak criteria the grammar will be updated too often and there will be a higher chance to introduce words that do not make sense. Starting from a given input word we retrieve a list of synonyms by using online web services, which provide a set of synonyms through a simple GET request. It is possible to select one or more web services and to use the union or the intersection of their results.

Fig. 6 describes the algorithm supposing that we want to populate the grammar according to synonyms rules. When an unknown word is synonymous of a word that is already present in the knowledge, it will be added to the grammar. This mechanism is carried out by processing the parser output for each input sentence. By analyzing the Phoenix parser output we create a list of words that are not recognized. Since we do not know if one or more words on the list are correlated between each other, we perform a combination of all of them.

```
Input:
      List of utterances U = {u1, u2, …, un}
      Starting Grammar
Output:
      Updated Grammar
Algorithm:
FOR EACH cycle
      compile(grammar)
      FOR EACH inputUtt ϵ U do
            unknownWords <- parse(inputUtt)
            wordsToProcess <- combination(unknownWords)
            FOR EACH word ϵ wordsToProcess
                  Synonyms[word] <- getSynonyms(word)
                  FOR EACH synonymous ϵ Combination(Synonyms)
                        newUtt <- replace(inputUtt, word, synonymous)
                        unknownWordsNew <- parse(newUtt)
                        IF(unknownWordsNew.Size() <
                        unknownWords.Size())
                        Update(grammar, word, synonymous)
                        END IF
```

Figure 6: Selg-learning grammar algorithm based on synonyms

For instance, if the list of unknown words contains the words *look* and *for*, we will process the words *look*, *for*, and *look for*. A list of synonymous for each of these terms is retrieved from external resources and stored along with the original word. Each synonymous of each unknown word is replaced in the original utterance and given to the Phoenix parser as input. If the number of parsed words of the new utterance is higher than the number of parsed words with the original sentence, then we have found a synonymous. The synonymous is then written in the corresponding file in the grammar. Fig. 7 shows how, step-by-step, the new module adds words that are not directly correlated to the knowledge. For instance, the verb *look for* is not seen as possible synonymous of *want*, but when the system discover *seek* as possible synonymous of *want*, then the word *look for* will be also discovered as synonymous of *want.*

Figure 3: Synonymous correlation between verbs

VII. FUTURE WORK

Based on this architecture, it is possible to develop future architectures to share resources between peers across multiple sessions. Turn grounding deals with **non-understandings**, i.e. turns in which the system believes the user spoke an utterance but is unable to extract any semantic information out of it. This can happen when the speech recognition hypothesis is completely meaningless or when it is not parsable by the NLU Phoenix Parser, or when the parse cannot be interpreted in the current dialog context. In addition to the normal grounding strategies [7], we could propose to the user a grounding model based on the experience of other users in other sessions (for instance the most common phrases). This will be based upon an analysis of all the log updated in real time or on what is going on in other sessions. This can be realized thanks to the sharing of all logs updated in real time between all OSP.

Moreover, the simple grammar mechanism is not bound to a specific scenario and it can be applied in different systems, and also with different criteria. For instance slots containing a list of cities or zip codes can be populated step-by-step using external resources that checks if a given city or zip code is valid.

VIII. CONCLUSION

In this paper we described OlympusP2P, an open source framework based on a distributed and highly context-sensitive framework along with a dynamic and self-learning NLU mechanism for applications that aim at interacting with users by using natural language. We depicted OpympusP2P's architecture explaining how it is able to support multiple contemporary sessions and the possibility to share a common self-learning dynamic grammar between the clients involved in the sessions.

ACKNOWLEDGMENT

This research was supported by research grants of 2011-2013 POR-FESR Sicilia.

REFERENCES

[1] Cheongjae Lee, Sangkeun Jung, Kyungduk Kim, Donghyeon Lee, and Gary Geunbae Lee, "Recent Approaches to Dialog Management for Spoken Dialog Systems," Department of Computer Science and Engineering Pohang University of Science and Technology (POSTECH) Pohang, Republic of Korea, 2010

[2] Dan Bohus, Antoine Raux, Thomas K. Harris, Maxine Eskenazi, and Alexander I. Rudnicky, "Olympus: an open-source framework for conversational spoken language interface research", in *NAACL-HLT '07: Proceedings of the Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies*, 2007

[3] Ward, W., Issar, S., "Recent improvements in the CMU spoken language understanding system", in *Proceedings of ARPA Human Language Technology Workshop*, Plainsboro, NJ, 1994.

[4] Joseph Polifroni, and Stephanie Seneff, "Galaxy-II as an Architecture for Spoken Dialogue Evaluation", Proc. LREC, 725-730, Athens, 2000

[5] Dan Bohus, Alexander I. Rudnicky, "The RavenClaw dialog management framework: Architecture and systems", *Computer Speech and Language*, vol. 23, no. 3, 2009

[6] Minker, Chase, "Evaluating Parses for Spoken Language Dialogue Systems", in First International Conference on Language Resources abd Evaluation – Proc. Workshop on The Evaluation of Parsing Systems, May 1998

[7] Bohus, D. & Rudnicky, A. 2002, November. Integrating multiple knowlege sources for utterance-level confidence annotation in the CMU Communicator spoken dialog system. (Tech. Rep. No. CMU-CS-02-190). Pittsburgh, Pennsylvania: School of Computer Science, Carnegie Mellon University.