# An Easy and Efficient Grammar Generator for Understanding Spoken Languages

A Novel approach to develop a Spoken Language Understanding Grammar for Inflective Languages

Salvatore Michele Biondi, Vincenzo Catania, Ylenia Cilano, Raffaele Di Natale, Antonio Rosario Intilisano
Dipartimento di Ingegneria Elettrica Elettronica e Informatica
University of Catania
Catania, Italy
salvo.biondi@dieei.unict.it, vincenzo.catania@dieei.unict.it, ylenia.cilano@dieei.unict.it, raffaele.dinatale@dieei.unict.it, aintilis@dieei.unict.it

***Abstract— In a Spoken Dialog System, the Spoken Language Understanding component is able to recognize words that were previously included in its grammar. The development of a grammar is a very time-consuming and error-prone process, especially for the inflectional languages, because the developer must manually include all possible inflected forms of a word. As a consequence, the grammar definition files are long and hard to manage. This paper describes a solution for creating a semi-automatic Spoken Language grammar using a morphological generator.***

***Keywords-Spoken Language Understanding; Natural Language Understanding; Spoken Dialog System; Grammar Definition.***

## I. INTRODUCTION

Grammar development in a Spoken Dialog System (SDS) is the process of specifying the words and patterns of words that a speech recognizer should be able to process. The system is able to recognize a user's utterance of "hello" only if that word is included in the grammar. Manual development of domain-specific grammar is time-consuming, error-prone and requires a significant amount of expertise. It is difficult to write a rule–set that has a good coverage of real data without making it intractable [1]. Writing domain-specific grammars is a major obstacle to a typical application developer. This specialization often does not cover any unspecified data and it often results in ambiguities [2].

With this purpose, we suggest that a semi-automatic method for generating grammar contents for inflectional languages is necessary. In order to circumvent the complexity associated with conventional methods for automatic grammar inference for Spoken language, we pursue a different avenue. More precisely, this paper focuses on the simplification of the writing of a grammar, by means of a method that automatically generates the inflected forms of its terms.

This is accomplished by introducing an intermediate grammar that helps generating a simpler and more compact grammar. The development process allows to obtain large amounts of grammar contents starting from a few rows of the intermediate grammar.

In order to test the validity of our solution, a specified grammar editor has been developed. It permits to automatically convert the new grammar format, developed in this work, in the Phoenix grammar [3]. Phoenix represent the Spoken Language Understanding (SLU) module of the Olympus Framework [4].

This paper is organized as follows: the Phoenix grammar format is described in Section II. The proposed grammar format in presented in Section III. Section IV introduces a Grammar generator based on a Morphological Generator for the Italian language and Section V shows an example. Finally, in Section VI, we draw conclusions.

## II. PHOENIX GRAMMAR

The Phoenix parser represents the state of the art of the development of robust Spoken Language interfaces for spoken language applications. Spontaneous speech is often ill-formed and could cause recognition errors. For such a reason, the proposed parser is designed to enable robust parsing and is able to manage this kind of input. The Phoenix parser uses a specific grammar file (.gra) containing context free rules that specify the word patterns corresponding to the token.

The pattern is a combination of words [5] that can be recognizable by the Phoenix parser. The syntax of a token is shown in Fig. 1:

```
# optional comment
[token_name]
        (<pattern a>)
        (<pattern b>)
;
```

Figure 1. Syntax of a token.

A token can also contain other tokens, as Fig. 2, for example:

```
[token_example]
        (word1 [other_token] word2)
;
```

Figure 2. Syntax token example.

 ISBN: 978-1-61208-342-1

This format allows for recognition of several sentences with the combination of different slots and words; furthermore, each token can be reused in many tokens.

In the inflectional languages [6], as in the case of Italian or Romance languages in general, words can occur in several forms, verbs can change their form depending on conjugations and nouns and adjectives depending on declensions. Moreover, suffixes or prefixes can be applied to them.

Thus, inflected forms add complexity to the Phoenix grammar, since they generate multiple different rules with similar patterns. That causes an increase of development time. Moreover, the developer might not include some inflected forms, thus causing the grammar to be incomplete.

## III. A NEW GRAMMAR SCHEMA

The development of a new domain application needs a new Context Free Grammar (CFG) that is able to define the concepts and their relations of such domain.

Alternative approaches learn structures from a set of corpora. However, this process appears too expensive and potentially not exhaustive [7]. Our approach consists of creating a new intermediate grammar that focuses on the meaning of a grammar token rather than on its content.

In such a way, it is no longer necessary to write the word pattern of the token, but only the "keyword name" like [**word**,**characteristic**]. The new schema generates a grammar file containing a token and its generated word patterns and it can be reused and edited like a standard Phoenix grammar. The new format description is shown in Fig. 3:

```
Function = SLOT_NAME
{
 [word,characteristic] term1
 [word,characteristic] term2
}
```

Figure 3. New grammar format description.

The grammar slots [5] are defined by the "Function" keyword that defines the slot name (Function = SLOT_NAME). This way, a token is defined as a couple "[word, characteristic]" and is used by the editor to generate the appropriate **word** patterns according to the **characteristic**.

The couple [word,characteristic] is defined as below:

*1) If "word" is a verb, "characteristic" can be replaced with:*

a) "Presente" if the Italian language present form is desired;
b) "Passato" if the Italian language past forms are desired;
c) "Futuro" if the Italian language future forms are desired.

*2) If "word" is a noun or an adjective, "characteristic" can be replaced with:*

a) "Singolare" if the Italian language singular forms are desired;
b) "Plurale" if the Italian language plural forms are desired;

All new forms specified by the characteristic are generated by a Morphological Generator [8].

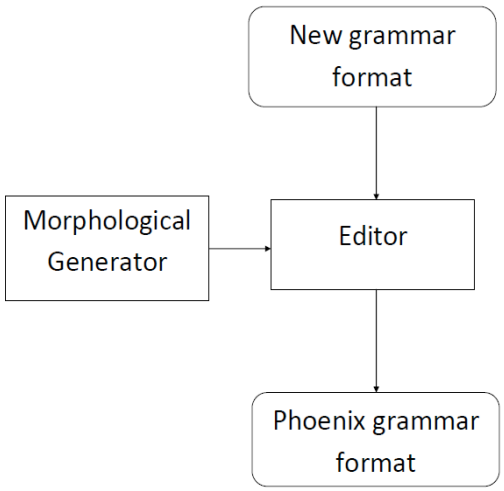


Figure 4. Grammar generation.

Our editor generates a standard Phoenix grammar from the new intermediate grammar, performing the following actions:

- Create a token named SLOT_NAME in which new tokens and terms are included;
- Create a token for each new defined token, in which terms generated by the Morphological Generator are included. The entire process is shown in Fig. 4.

## IV. GRAMMAR GENERATOR

In our test, we used Italian as inflectional language, but different Romance languages can be used. Each inflected form of a verb gives information about mood, tense, number and person. There are also some verbs, nouns and adjectives that are inflected in an irregular manner. In Italian, nouns and adjectives can be altered by adding particular suffixes.

These alterations modify a word's meaning in terms of quantity or quality. This increases the effort in developing an efficient Spoken Language Understanding grammar for a SDS.

In our solution, a Morphological Generator generates all inflected forms of a word for Italian language. Its aim is to help the programmer to generate a complete SLU grammar for a SDS in an easy way.

 ISBN: 978-1-61208-342-1 

## V. EXPERIMENTAL RESULTS

An example is reported to show the advantages obtained by this approach. It shows a grammar developed for a room reservation application. In a typical interaction, the user can express the same concept using a specific word, but in different tenses.

For example, “I want a room” in Italian can be expressed like “Voglio una camera” but also “Vorrei una camera” (I'd like to have a single room.) or “Vorrei una cameretta” (I'd like to have a small room.). Fig. 5 shows an example of grammar:

```
Function = NEED_ROOM_PRESENT
{
    [Volere,Presente] [Camera,Singolare]
    [Desiderare,Presente][Camera,Singolare]
    [Volere,Presente] [Stanza,Singolare]
    [Desiderare,Presente][Stanza,Singolare]
}

Function = NEED_ROOM_FUTURE
{
    [Volere,Futuro] [Camera,Singolare]
    [Desiderare,Futuro][Camera,Singolare]
    [Volere,Futuro] [Stanza,Singolare]
    [Desiderare,Futuro][Stanza,Singolare]
}
```

Figure 5. New grammar format example.

The new grammar consinsts of two parts. The first one, shown in Fig. 6, represents the definition of a grammar slot:

```
[NEED_ROOM_PRESENT]
      [VolerePresente] [CameraSingolare]
      [DesiderarePresente][CameraSingolare]
      [VolerePresente] [StanzaSingolare]
      [DesiderarePresente][StanzaSingolare]
;
[NEED_ROOM_FUTURE]
      [VolereFuturo] [CameraSingolare]
      [DesiderareFuturo][CameraSingolare]
      [VolereFuturo] [StanzaSingolare]
      [DesiderareFuturo][StanzaSingolare]
;
```

Figure 6. Phoenix grammar generated.

The second part, shown in Fig. 7, defines each token including their word patterns. A more detailed explanation along with the source code (output.gra file) is given in [9].

The initial grammar, consisting of 21 rows, generates a 140-row-long Phoenix grammar that allows the SLU module to recognize a large set of utterances.

This way, the developer focuses his attention on the meaning of an intermediate-grammar token and not on its content.

```
#Tag Auto Generated          #Tag Auto Generated
[VolerePresente]             [StanzaSingolare]
      (voglio)                     (stanza)
      (vuoi)                       (stanzaccia)
      …                            ...
;                            ;

#Tag Auto Generated          #Tag Auto Generated
[CameraSingolare]            [VolereFuturo]
      (camera)                     (vorro')
      (cameraccia)                 (vorrai)
      ...                          ...
;                            ;

#Tag Auto Generated          #Tag Auto Generated
[DesiderarePresente]         [DesiderareFuturo]
      (desidero)                   (desiderero')
      (desideri)                   (desidererai)
      ...                          ...
;                            ;
```

Figure 7. Token definition generated.

Furthermore, the developer does not need to write all possible forms (mood, tense, person, etc.), some of which could be difficult to predict. The advantage of the generated grammar is the ability to easily simulate and predict the large variety of interactions that can occur.

## VI. CONCLUSION AND FUTURE WORK

This paper proposed a solution to simplify and reduce the amount of writing of the SDS grammar of inflectional language. This method reduces the effort to produce a grammar for a SDS. The SDS used for our tests is the Olympus framework.

An editor has been developed for the translation of the new simple grammar format in the Phoenix grammar format. The editor uses a Morphological Generator to obtain all possible inflected words that are used to create grammar tokens.

The proposed solution will be integrated in our major project called Olympus P2P [10], which is concerned with the upgrading and updating of an SDS grammar by means a Peer to Peer Network to share new grammar tokens.

## ACKNOWLEDGMENT

The authors were supported by the Sicilian Region grant PROGETTO POR 4.1.1.1: "Rammar Sistema Cibernetico programmabile d’interfacce a interazione verbale".

## REFERENCES

[1] H. M. Meng and K-C. Siu, “Semtiautomatic Acquition of Semantic Structures for Understanding Domain-Specific Natural Language Queries”, IEEE Tran. Knowledge & Data Eng., pp. 172–181, vol. 14(1), 2002.

 ISBN: 978-1-61208-342-1 

[2] Y. Wang and A. Acero, "Grammar learning for spoken language understanding," *Automatic Speech Recognition and Understanding, 2001. ASRU '01. IEEE Workshop on*, pp.292-295, 2001.

[3] W. Ward, "Understanding spontaneous speech: the Phoenix system," *Acoustics, Speech, and Signal Processing, 1991. ICASSP-91, 1991 International Conference on*, 14-17 Apr 1991, pp.365-367 vol.1.

[4] D. Bohus, A. Raux, T. K. Harris, M. Eskenazi and A. I. Rudnicky, *Proceedings of the Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies*, NAACL-HLT-Dialog '07 (Association for Computational Linguistics, Stroudsburg, PA, USA, 2007), pp. 32–39.

[5] Phoenix Parser User Manual, http://www.ontolinux.com/community/phoenix/Phoenix_Manual.pdf (last visited: 22 November 2013).

[6] M. Haspelmath and A. D. Sims, Understanding Morphology 2nd edition. London: Hodder Education, 2010.

[7] S. Knight, G. Gorrell, M. Rayner, D. Milward, R. Koeling and I. Lewin, "Comparing grammar-based and robust approaches to speech understanding: a case study", EUROSPEECH 2001 Scandinavia, 7th European Conference on Speech Communication and Technology, 2nd INTERSPEECH Event, Aalborg, Denmark, September 3-7, 2001, pp. 1779-1782.

[8] V. Catania, Y. Cilano, R. Di Natale, V. Mirabella and D. Panno, "A morphological engine for Italian language", ICIEET 2013: 2nd International Conference on Internet, E-Learning & Education Technologies, 2013, pp. 36-43, vol. 12(1).

[9] Source code, http://opensource.diit.unict.it/vctsds/GrammarEditor.zip (last visited: 22 November 2013).

[10] V. Catania, R. Di Natale, A. Longo and A. Intilisano, "A distributed Multi-Session Dialog Manager with a Dynamic Grammar Parser", 2nd International Conference on Human Computer Interaction & Learning Technologies, 2013, pp. 1-9, vol. 8(2).

 ISBN: 978-1-61208-342-1 