Automatic Generation of Semantic Patterns using Techniques of Natural Language Processing

Pablo Suarez, Valentín Moreno, Anabel Fraga and Juan Llorens

Carlos III of Madrid University, Madrid, Spain

Abstract. Within the discipline of natural language processing there are different approaches to analyze large amounts of text corpus. The identification patterns with semantic elements in a text let us classify and examine the corpus to facilitate interpretation and management of information through computers. This paper proposes the development of a software tool that generates index patterns automatically using various algorithms for lexical, syntactic and semantic analysis of text and integrates the results into other projects in the area of research and other ontological formats. The algorithms in the system implemented various types of analysis in the context of natural language processing, so they can identify grammatical categories and semantic characteristics of words, making up index patterns. The results obtained correspond to a pattern list sorted by frequency of occurrence and take into account intermediate optional elements, which determine its relevance and usefulness to other projects. The developed system proposes a model of generation and storage of patterns, and a control interface that allows the specification of parameters and running reports.

1 Introduction

A pattern is a reusable solution to a recurring problem in software design. The recognition of patterns through domains allows the classification and identification of common solutions to simplify the analysis and understanding of the domains. It is possible with patterns and analysis techniques to classify and interpret texts through computers, thus able to solve many problems in the research areas of artificial intelligence, speech and language recognition and information retrieval [1].

Through software development is possible to integrate the concepts of identifying patterns into the design of a system that automatically generate patterns from large amounts of text or corpus. This system should incorporate methods of natural language processing that allow the classification, identification and analysis of text patterns. Software engineering provides a systematic approach necessary to undertake these research projects using methodologies and design techniques.

2 State of the Art

The methods and objectives for this project enter within the scope of knowledge ma-

Suarez P., Moreno V., Fraga A. and Llorens J..
Automatic Generation of Semantic Patterns using Techniques of Natural Language Processing.
DOI: 10.5220/0004641500340044
In *Proceedings of the 4th International Workshop on Software Knowledge* (SKY-2013), pages 34-44
ISBN: 978-989-8565-76-1

nagement as the solution implies a representation of semantic data through the organization and analysis of information. Knowledge management builds and manages conceptual models that can recover and understand the data from domains, in this case, to make an analysis of different features for the representation of semantic relations and patterns [2]. Semantics is understood as the study of the meaning of linguistic expressions, and corresponds to a major branch of the study of the natural language processing [3]. In the index patterns generation, semantics is included within the orientation of natural language and not of artificial language as usually determined for software engineering projects. The semantic pattern generation is contained into the models defined within the knowledge management approach. The system incorporates grammatical categories, semantic relationships and contexts defined by sentences. Through the generation of these patterns it is possible to develop hierarchical structures and rip taxonomic relationships through the semantic data [4]. The analysis of texts is defined using techniques like "pattern-matching", which are the basis of the pattern generation algorithm proposed in the system.

A. Information Reuse

Reuse in software engineering is present throughout the project life cycle, from the conceptual level to the definition and coding requirements. This concept is feasible to improve the quality and optimization of the project development, but it has difficulties in standardization of components and combination of features. Also, the software engineering discipline is constantly changing and updating, which quickly turns obsolete the reusable components [5]. At the stage of system requirements reuse is implemented in templates to manage knowledge in a higher level of abstraction, providing advantages over lower levels and improving the quality of the project development. The patterns are fundamental reuse components that identify common characteristics between elements of a domain and can be incorporated into models or defined structures that can represent the knowledge in a better way.

B. Natural Language Processing

The Natural Language Processing (NLP) is an approach in artificial intelligence that we have selected for indexing pattern generation, as it brings together several mechanisms that enable effective interpreting of natural language texts. These mechanisms try to solve the ambiguities of language and vocabulary by its semantic, which is precisely the main difficulty facing the corpus analysis [6]. To process the corpus is necessary to use specialized tools in each of the different type of documents and analysis.

The need for implementing Natural Language Processing techniques arises in the field of the human-machine interaction through many cases such as text mining, information extraction, language recognition, language translation, and text generation, fields that requires a lexical, syntactic and semantic analysis to be recognized by a computer [7]. The natural language processing consists of several stages which take into account the different techniques of analysis and classification supported by the current computer systems [8].

1) *Tokenization:* The tokenization corresponds to a previous step on the analysis of the natural language processing, and its objective is to demarcate words by their sequences of characters grouped by their dependencies, using separators such as spaces and punctuation [6]. In this way, the tokenization gets a standard list of tokens that are related to the text in sequential order and establishes the structure for the further analysis. Tokens are items that are standardized to improve their analysis and to simplify ambiguities in vocabulary and verbal tenses. For this reason there are several difficulties to tokenize, as it is possible to find special cases, such as abbreviations, acronyms, numerical data or compounds terms that should have specific rules to allow recognition. Tests have been conducted on corpuses that have reached tokenization accuracy of 99%, as in the Brown Corpus [9].

2) *Lexical Analysis:* Lexical analysis aims to obtain standard tags for each word or token through a study that identifies the turning of vocabulary, such as gender, number and verbal irregularities of the candidate words. An efficient way to perform this analysis is by using a finite automaton that takes a repository of terms, relationships and equivalences between terms to make a conversion of a token to a standard format [10]. Another method of lexical analysis is concerned in the need to analyze unfamiliar words to a specific dictionary. For this, it can be possible to have a morphological model that defines derivation rules of words to identify common terms in different variations. In the latter case it is possible to classify the unknown terms and without morphological relationship with others, as nouns (in the case of proper names) or other special categories (undefined category). Finally, there are several additional approaches that use decision trees and unification of the databases for the lexical analysis but this not covered for this project implementation [11].

3) *Syntactic Analysis:* The goal of syntactic analysis is to explain the syntactic relations of texts to help a subsequent semantic interpretation [12], and thus using the relationships between terms in a proper context for an adequate normalization and standardization of terms. Several problems in this analysis have to do with the coverage of the grammar for a defined language, the structural ambiguities, and the cost of computer processing that depends on the preferred depth of the analysis, therefore, to expedite the processing it is possible to delimit the analysis to sentences or phrases into a limited scope or size. The preferred method for an efficient analysis corresponds to the realization of a superficial analysis that defines into a local scope of information. For this, there are deductive techniques which define a set of grammar rules and that use finite state techniques [13], and inductive techniques, that use machine learning and neural networks on a training corpus [14]. To incorporate lexical and syntactic analysis, in this project were used deductive techniques of standardization of terms that convert texts from a context defined by sentences through a special function or finite automata.

4) *Grammatical Tagging:* Tagging is the process of assigning grammatical categories to terms of a text or corpus. Tags are defined into a dictionary of standard terms linked to grammatical categories (nouns, verbs, adverb, etc.), so it is important to normalize the terms before the tagging to avoid the use of non-standard terms. The most common issues of this process are about systems' poor performance (based on large corpus size), the identification of unknown terms for the dictionary, and ambiguities of words (same syntax but different meaning) [15]. Grammatical tagging is a

key factor in the identification and generation of semantic index patterns, in where the patterns consist of categories not the terms themselves. The accuracy of this technique through the texts depends on the completeness and richness of the dictionary of grammatical tags.

5) *Semantic and Pragmatic Analysis:* Semantic analysis aims to interpret the meaning of expressions, after on the results of the lexical and syntactic analysis. The implementation problems are due to lexical ambiguity of language as homonymous word can have multiple grammatical categories; referential ambiguity, since the meaning of the terms may vary by their pronouns, and the ambiguity of scope, when there are additional words in the context that change the meaning of the related words [6]. The semantic analysis gets the semantic value from the terms and links them to the generated patterns in various forms, storing their value into the Patterns structure. The pragmatic analysis for its part also depends on the previous stages and aims to interpret the words or phrases in a defined context [16]. This analysis not only considers the semantics of the analyzed term, but also considers the semantics of the contiguous terms within the same context. Automatic generation of index patterns at this stage and for this project does not consider the pragmatic analysis.

6) *RSHP:* it is a model of information representation based on relationships that handles all types of artifacts (models, texts, codes, databases, etc.) using a same scheme. This model is used to store and link generated pattern lists to subsequently analyze them using specialized tools for knowledge representation [17]. Within the Knowledge Reuse Group at the University Carlos III of Madrid RSHP model is used for projects relevant to natural language processing.

3 Description of the System

This section indicates the system features and data structures that will represent the generated information from the corpus, using containers and descriptions through various examples. Then it includes functions and algorithms that will use the defined data structures, and an activity sequential diagram that defines the flow of execution of the algorithms and functions.

A. Data Structures

1) *Pattern Definition:* A pattern is defined in the system as a tuple that may contain grammatical categories (C) or subpatterns (P), in where the number or ID associated with the element points to a specific category or pattern. The patterns relate to the texts of the corpus through the grammatical category represented, so a pattern that contains subpatterns can be split up to be composed only of grammatical categories. This creates a binary tree structure for pattern reporting.

2) *Tokens:* The patterns are generated and stored across three data structures defined: Tokens, Map and Patterns. Tokens correspond to a vector that sequentially stores the elements of the texts that could be grammatical categories or patterns. The system has only one Tokens structure in which the algorithms will by replacing the grammatical

categories with patterns depending on the defined iteration levels. Storing Tokens is the first step in the analysis of corpus to generate patterns.

3) *Map:* The map is a collection of all combinations of grammatical categories and patterns that are stored in the Tokens table by a defined criterion. The simplest criterion is considered in this project and it corresponds to the identification of a pattern as a pair of adjacent elements. The pattern sequential order is no more relevant for the further analysis.

4) *Patterns:* The data structure of Patterns or unique patterns corresponds to a list of items found in the Map grouped by frequency. This structure will contain the patterns found in the texts according to the defined generation criteria, and it will store them in frequency order, which indicates the relevance of the pattern. The most important patterns are the most frequent on the original text. In addition to the frequency, Patterns will store the substitution level of each element to know in which iteration was generated. Patterns with optional elements between are stored in a separate structure with a direct reference to Patterns.

B. Systems Requirements and Scope

To meet the proposed objectives it has been defined for the project the following requirements, based on technological capabilities and compatibility of the results:

- Develop a program in C # that groups different pattern generation algorithms.
- The algorithms must be able to analyze texts, generate sentences, tokens and patterns.
- The system must support and expand the RSHP model using its same data model.
- The system must have export functions to other hierarchical compatible formats mainly for thesaurus representation formats.

In this way the system should be able to take different corpus in natural language, analyze and categorize them to generate a final list of patterns ordered by frequency. In where the generated data should have a standard format to be able to extend to a model based on graphs as RSHP, and to be exportable to other hierarchical compatible formats.

C. Activity Diagram

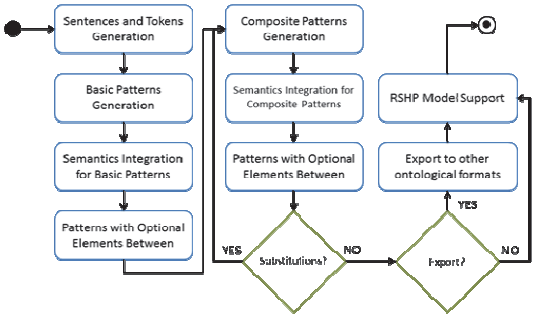

Fig. 1. Activity diagram of the system.

D. System Algorithms

The algorithms in the system are defined by modules, which are independent from each other but reusable in different parts of the pattern generation process. However, it is defined a sequence of execution of algorithms that use necessary structures generated by other algorithms. Therefore, for reading the corpus and to generate patterns, is a must to execute the algorithms in the sequential order defined in the activity diagram. Just in the case of having structures with pre-filled records the execution could move to further algorithms.

1) *Sentences and Tokens Generation:* The first algorithm to execute consists in the token generation from texts of corpus. For this is required an intermediate step to generate sentences from texts and then passing the sentences into a normalize function that generates the final tokens. The sentences are defined by symbols of demarcation as dots or commas separating the sentences within the analyzed text. The algorithm also ignores line breaks, so this does not affect to the delimitation of a sentence.

- Read a line from the corpus.
- Generate sentences from the line by the delimiter characters considering previous sentences and fragments.
- Read again the lines until the end of the file.
- Read a generated sentence.
- Normalize the sentences according to their context and get a list of tokens with properties.
- Store the obtained data to Tokens structure.
- Read again the sentences until the last sentence.

2) *Basic Patterns Generation:* The basic patterns are the first generation of patterns that take for its execution the adjacent tokens criteria. The adjacent tokens are tokens analyzed and grouped in pairs in sequential order through the entire data structure Tokens. As is the first algorithm that runs, it scans the items in Tokens that currently apply only to grammatical categories, so it will not include subpatterns. The basic pattern generation stores all possible combinations of adjacent tokens in the Map structure, and then, the final identification of patterns uses an algorithm to sort the generated patterns through a dictionary object grouped as unique elements and ordered by frequency. Within the basic patterns generation algorithm it can be executed the functions to identify semantics and patterns with optional elements between.

- Read the generated tokens.
- Group them in pairs and save the Map.
- Read again the tokens until the end of the structure.
- Read the patterns stored in the Map.
- Save the patterns with a frequency counter in a dictionary.
- Read again map patterns until completion.
- Sort the dictionary by the counter and save the patterns to the Patterns structure.

3) *Composite Patterns Generation*: The composite patterns correspond to the later generations of patterns that are formed by substitution levels into the Tokens structure. The algorithm consists of two parts, the first part makes modifications to the Tokens structure by replacing tokens or subpatterns with patterns (through a "pattern-matching" technique), and a second part that implements a similar process to the basic

pattern generation, that generates a map and list of patterns stored in the Patterns structure. The algorithm execution repeats itself depending on the number of substitutions defined by the system's user. If there are no limits defined, the system realizes all the substitutions until there are no more possible substitutions. In this case, there will be a great pattern that groups the entire corpus, but linked to all subpatterns as a tree root.

- Read the generated tokens and patterns.
- Check if there are matches of patterns into the Tokens structure and make all the possible substitutions.
- Get two new patterns from the new tuple formed by pattern's adjacent elements and save them into the Map.
- Read again the patterns and tokens until the end of the structure.
- Read the patterns stored in the Map.
- Save the patterns with a frequency counter in a dictionary.
- Read again map patterns until completion.
- Sort the dictionary by the counter and save the patterns to the Patterns structure.

4) *Pattern Generation with Optional Elements Between:* The patterns with optional elements between are composed by a tuple pattern that can contain any elements between (grammatical categories or subpatterns). To generate these patterns, the system takes the basic and composite patterns and compares them to the current Token structure by a pattern matching that considers intermediate elements. Subsequently, these matches are stored in a map and through a dictionary the algorithm sorts and stores them in the Patterns table. This algorithm is integrated with the basic and composite pattern generation to seize the execution time.

- Read the generated tokens and patterns.
- Check if there are matches or patterns into the Tokens structure and save the patterns matched and their between elements into the Map.
- Read again the patterns and tokens until the end of the structure.
- Read the patterns stored in the Map.
- Save the patterns with a frequency counter in a dictionary.
- Read again map patterns until completion.
- Sort the dictionary by the counter and save the patterns to the Patterns structure.

5) *Semantics Integration:* The patterns integrate semantics from their grammatical categories that contain a semantic code. To include semantics, the patterns are read and for the two elements of the tuple it is checked if it has a verb type grammatical category. If the verb type has a semantic code it is saved into the Patterns structure, if not, the pattern takes its semantic from the verb itself. This algorithm is integrated with the basic and composite pattern generation to seize the execution time. There are 4 possible scenarios of semantic integration with the generated patterns:

- Pattern with semantics by having a verb type grammatical category with a semantic code.
- P34 = (C22, C11) = (NOUN, VERB) and VERB has a semantic code 50.
- Pattern with semantics by having a verb type grammatical category getting the semantic from the verb itself.
- P45 = (P1, C11) = (P1, VERB) and VERB not belong to a semantic group.
- Pattern without semantics for not having a verb type grammatical category.

- P25 = (C138, C22) = (DEFINITE ARTICLE, NOUN).
- Pattern without semantic for being composed of two sub-patterns.
- P265 = (P25, P22)

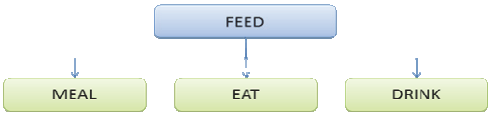

Fig. 2. Example of a semantic verb group.

6) *Export Functions:* It is possible to use the generated patterns structure and data to export them into other ontological formats. For this, a file is created with the list of the generated patterns including its grammatical categories or sub-patterns, which can be interpreted by the natural language processing tools like Knowledge Manager. Optionally, it is possible to generate a file that includes the patterns and its description of the semantic scenarios.

7) *RSHP Support:* After the pattern generation with semantics and optional elements between, the generated data is copied to a special format defined in the RSHP database model to permit reusability in other similar projects. For each pattern is verified that it has patterns with optional elements between, and only the pattern with the highest abstraction level is copied to the RSHP database, ignoring the lower abstraction levels. In other words, the patterns with higher amount of optional elements between are taken into account in the integration with RSHP model as they group patterns with less or equal to zero optional elements.

Example: P10 = (2, 3) and this pattern has its variants with optional elements between:

- O1 (P10) = (2, [1,4], 3)
- O2 (P10) = (2, [1], 3)
- O3 (P10) = (2, [5], 3)

The patterns O1 and O3 are integrated in the RSHP model, as O1 has higher level of abstraction that P10 and O2, and O3 has higher level of abstraction than P10 and do not have optional elements equal to O1 and O2.

4 Testing

The system incorporates a console interface that displays the status of the components and the execution of the algorithms, indicating the progress through percentages. Then it displays the number of generated items and proceeds with the various iterations of the algorithm of the composite patterns generation. The tests presented below represent the execution and reading of the Brown Corpus.

There are two ways to view the results of the execution, first, through console by printing selected patterns, and second, through browsing the generated tables in the database. The pattern printing consist in getting the properties of the selected pattern by its identifier, displaying the breakdown of subpatterns or grammatical categories, the substitution level, semantic data and the associated frequency counter.

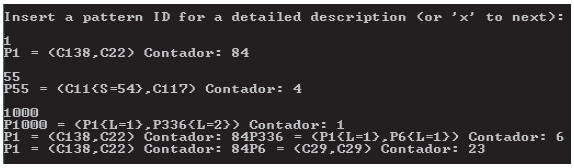

Fig. 3. Screenshot of the activity of the pattern printing function.

5 Results

The execution of the system generated 64,575 delimited sentences from almost the 1 million words that the Brown Corpus contains. From the sentences it was generated 546,352 terms that consist in a list of normalized tokens. From the terms the basic patterns generated were 5,058 and the basic patterns with optional elements between were 411,626 (contemplating every possibility for 2 optional elements between).

The generated data is stored into the Patterns structure, which is accessible from the console reports or in viewing the table itself. The most frequent patterns generated are P1 = (C138, C22), P4 = (C22, C127) and P5 = (C137, C22), and the most frequent patterns with optional elements between are O1(P6) = (C127,[C138] ,C22), O4(P223) = (C22, [C127], C22) and O6(P29) = (C22, [C127], C138).

The composite patterns for the first substitution level are 14,651 and the most frequent are P5059 = (C127, P1), P5064 = (P4, P1) and P5068 = (C22, P1). The composite patterns for the second substitution level are 4,982 and the most frequent are P19711= (P1, P5059), P19722= (C137, P5064) and P19758 = (P4, P5059). The patterns that contain grammatical categories 13, 29, 50 and 54 were ignored from the most frequent results as they contain grammatical categories that correspond to punctuation marks, and they are considered irrelevant for the interpretation of the results.

Table 1. Most frequent generated patterns from brown corpus.

#	ID	Element 1	Element 2	Count	Level
1	1	C138	C22	31061	0
2	2	C22	C50	20972	0
3	3	C22	C54	19520	0
4	4	C22	C127	18198	0
5	5	C137	C22	11108	0
6	6	C127	C22	10958	0
7	7	C127	C138	9788	0
8	8	C22	C162	8669	0
9	5059	C127	P1	8252	1
10	9	C50	C22	7843	0
32	5064	P4	P1	3538	1
46	5068	C22	P1	2590	1
430	19711	P1	P5059	294	1

6 Conclusions

The most significant pattern found corresponds to P1 = (C138, C22) = (DEFINITE

ARTICLE, NOUN) repeated 31061 times, an intuitive and common case presented in the English language. The following patterns P4 = (C22, C127) = (noun, preposition) repeated 18198 times, and P5 = (C137, C22) = (indefinite article, noun) repeated 11108 times, which likewise correspond to common structures of English grammar.
For patterns generated in higher levels of substitution, the most frequently correspond to P5059 = (C127, P1) = (PREPOSITION, DEFINITE ARTICLE, NOUN) repeated 8252 times, P5064 = (P4, P1) = (NOUN, PREPOSITION, DEFINITE ARTICLE, NOUN) repeated 3538 times and P5068 = (C22, P1) = (NOUN, INDEFINITE ARTICLE, NOUN) repeated 2590 times. If we arrange the patterns by frequency, it is found that P5064 is the most common pattern # 32, and P5068 is the most common pattern # 46.

It is considered as relevant patterns the patterns that contain semantics, the most frequent are P25 = (C22, C11) = (NOUN, VERB) 3953 times repeated, P40 = (C11, C127) = (VERB, PREPOSITION) 2352 times repeated, P54 = (C11, C22) = (NOUN, PREPOSITION) 1815 times repeated, P62 = (C117, C11) = (PERSONAL PRONOUN, VERB) 1510 times repeated, P5090 = (C11, P1) = (VERB, DEF. ARTICLE, NOUN) 1261 times repeated, P5095 = (P1, C11) = (DEF. ARTICLE, NOUN, VERB) 1117 times repeated, P5444 = (P11, C11) = (AND LINKING, NOUN, VERB) 122 times repeated.

The interpretation of the results may vary depending on the purpose of use of the tool. In this case it has been interpreted to ignore patterns containing grammatical categories of punctuation signs (C50 and C52) as it does not present relevant information about the semantics of the texts.

In summary, it was implemented a system that contains modules and algorithms that provide a pattern generation from text of a corpus. This system supports the RSHP model and expands itself through the copy of the generated patterns to the database of RSHP and by exporting the pattern content to other hierarchical formats. It was generated a final list of patterns that contain basic patterns at most.

There are several tools in the area of research that allow various analyzes and techniques within the framework of natural language processing like morphological, syntactic and semantic analysis [18]. However, tools for generating patterns based on these techniques are relatively new developments that are still implementing and testing different approaches of text analysis [19]. The implementation of these techniques was a priority for the project with the consideration of a necessary integration of results to other projects and models.

7 Future Work

As future work, it is considered the development of a text search tool that uses semantic patterns to find content in different dominions. Also, the system can be improved in several ways: as the performance of the algorithms, as the inclusion of new techniques for corpus analysis, interface improvements, and inclusion of new reports, adding new parameters for implementation, and integration with other tools for knowledge representation.

References

1. Larman, C. UML y Patrones: Una Introducción al Análisis y Diseño Orientado a Objetos y al Proceso Unificado, Segunda Edición, Prentice-Hall, 2002. Chapter 23.
2. Gómez-Pérez, Asunción. Fernando-López, Mariano. Corcho, Oscar. Ontological Engineering. London: Springer, 2004.
3. Thomason, Richmond H. What is Semantics? Version 2. March 27, 2012. Available in: http://web.eecs.umich.edu/~rthomaso/documents/general/what-is-semantics.html
4. Amsler, R.A. A Taxonomy for English Nouns and Verbs. Proceedings of the 19th Annual Meeting of the Association for Computational Linguistic. Stanford, California, 1981. pp. 133-138.
5. Llorens, Juan. Definición de una Metodología y una Estructura de Repositorio Orientadas a la Reutilización: El Tesauro de Software. Universidad Carlos III. 1996.
6. Moreno, Valentín. Representación del Conocimiento de Proyectos de Software Mediante Técnicas Automatizadas. Anteproyecto de Tesis Doctoral. Universidad Carlos III de Madrid. Marzo 2009.
7. Cowie, Jim. Wilks, Yorick. Information Extraction. En Dale, r. (ed). Handbook of Natural Language Processing. New York: Marcel Dekker, 2000. pp.241-260.
8. Dale, R. Symbolic Approaches to Natural Language Processing. En Dale, R (ed). Handbook of Natural Language Processing. New York: Marcel Dekker, 2000.
9. Riley, M. D. Some Applications of Tree-based Modeling to Speech and Language Indexing. Proceedings of the Darpa Speech and Natural Language Workshop. California: Morgan Kaufmann, 1989. pp. 339-352.
10. Hopcroft, J. E. Ullman, J. D. introduction to automata theory, languages and computations. addison-wesley, reading, ma, united states. 1979.
11. Triviño, J. L. Morales Bueno, R. A Spanish Pos Tagger with Variable Memory. in Proceedings of the Sixth International Workshop On Parsing Technologies (iwpt-2000). ACL/SIGPARSE, Trento, Italia, 2000. pp. 254-265.
12. Martí, M. A. Llisterri, J. Tratamiento del Lenguaje Natural. Barcelona: Universitat de Barcelona, 2002. p. 207.
13. Abney, Steven. Part-of-speech Tagging and Partial Parsing, S. Young and G. Bloothooft (eds.) Corpus-based Methods in Language and Speech Processing. An Elsnet Book. Bluwey Academic Publishers, Dordrecht. 1997.
14. Carreras, xavier. Márquez, luis. phrase recognition by filtering and ranking with perceptrons. en proceedings of the 4th ranlp conference, borovets, bulgaria, september 2003.
15. Weischedel, R. Metter, M. Schwartz, r. Ramshaw, L. Palmucci, J. coping with ambiguity and unknown through probabilistic models. computational linguistics, vol. 19, pp. 359-382.
16. Poesio, M. semantic analysis. en dale, r. (ed). handbook of natural language processing. new york: marcel dekker, 2000.
17. Llorens, J., Morato, J., Genova, G. RSHP: an information representation model based on relationships. in ernesto damiani, lakhmi c. jain, mauro madravio (eds.), soft computing in software engineering (studies in fuzziness and soft computing series, vol. 159), springer 2004, pp. 221-253.
18. Alonso, Laura. Herramientas Libres para Procesamiento del Lenguaje Natural. Facultad de Matemática, Astronomía y Física. UNC, Córdoba, Argentina. 5tas Jornadas Regionales de Software Libre. 20 de Noviembre de 2005. available in: http://www.cs.famaf.unc.edu.ar/~laura/freenlp
19. Rehberg, C. P. Automatic Pattern Generation in Natural Language Processing. United States Patent. US 8,180,629 b2. May 15, 2012. January, 2010.