AUTOMATIC WRAPPER SYSTEM FOR SEMI-STRUCTURED DOCUMENTS BASED ON DATA MINING

Irina RANCEA[1], Valentin SGÂRCIU[2]

Lumea în care evoluăm presupune înțelegerea şi acumularea unei cantități imense de informație împărțită în diferite surse care necesită integrare şi sinteză. A apărut necesitatea unor aplicații inteligente, capabile să proceseze sau să colecteze automat informațiile dorite. Acestea folosesc algoritmi de clusterizare pentru a descoperi grupuri. Totodată, datorită experienței obținute în timp în domeniul aplicațiilor software tendința care se impune este de automatizare a proceselor, economisind astfel timp prețios al dezvoltatorilor, timp care poate fi folosit în proiectarea de noi concepte, arhitecturi. Lucrarea propune o îmbinare între descoperirea de informații în documente şi procesarea acestora în vederea automatizării proceselor software.

Our world involves understanding and storing of huge information from different sources that need integration and synthesis. The necessity of smart applications that can automatically process and collect such information was critic. These applications use clustering analysis in order to find common groups of data. Also due to the knowledges in the software applications area, the new trend for this domain is process automation, saving in this way time for design of new concepts and architectures. Our paper proposes a combination of discovering and processing information stored in documents in order to automate software processes.

Keywords: natural language processing, data mining, cluster analysis

1. Introduction

There are various approaches that work with technologies based on natural language. Even if the complete understanding of natural language is still far away from the current technologies abilities, the methods used by the IE (information extraction) approach are more accurate and can recognize different entities and some relations between them.

The research in the processing natural language area have been inspired from linguistics – the text is parsed using information described by a formal grammar and a lexicon; the results are then semantically interpreted and used to extract information about the topic target.

[1] Eng., Faculty of Automatic Control and Computers, University of POLITEHNICA Bucharest, Romania, e-mail: irina.rancea@gmail.com

[2] Prof., Faculty of Automatic Control and Computers, University of POLITEHNICA Bucharest, Romania, e-mail: vsgarciu@aii.pub.ro

Text Mining is a new approach and uses methods from the information identification and statistics areas. Its target is not to understand the text or even a part of it, but to extract patterns from a huge input documents. The most simple form of Text Mining is Information Extraction. Other forms include automatically classify text, topic extraction. [1]

The topic of this paper is designing an IE system that can extract precise information from a tutorial. A tutorial is part of the semi-structured documents class. Classic IE systems use NLP techniques such as grammars and lexicons, and IE systems for web extractions uses Data Mining techniques such as exploiting syntactic patterns. The technical documents about programming languages references can be easily structured in a pattern. The final target of the results is to create data files that can be automatically processed in order to design software applications for syntactic parsing of a language programming.

Sometimes the information extracted can be a large amount of data that need to be classified in relevant and not relevant data. In order to identify which data are relevant, the paper suggests clustering the information extracted, and then applying a cluster analysis based on a set of defined parameters.

2. Theoretical background

2.1. Information extraction methodology

The most part of the digital information is described in natural language. There are dedicated methods that can be used in order to obtain from a large amount of data just the needed information. Information retrieval allows helps in discovering documents related to the target topic, but doesn't allow to create queries and receive answers. Information extraction identifies pieces of the target information described in natural language and offers a structure to store and process them automatically.

An IE (information extraction) problem is defined by its input and target data. The input can be documents written in natural language or semi-structured documents – on-line or off-line. The objective of an extraction can be *k-tuple* relation (where k is the attribute number from a record) or a complex object from an hierarchical structure of data.

The systems that solves IE problems are called wrappers. A wrapper is system integration component that offers an unified interface for accessing various information sources. A wrapper usually perform a matching procedure for one or multiple patterns.

The research in this area was inspired from MUC – Message Understanding Conferences. The massive contribution of the MUC team has classified IE approaches in two major classes with the following examples of IE systems:

- pre-MUC approach: AutoSolg [2], LIEP [3], PALKA [4], HASTEN [5], CRYSTAL [6]
- post-MUC approach: WHISK [6], RAPIER [7], SRV [8], WIEN [9], SoftMealy [10], STALKER [11]

Chang [12] compares IE systems in term of human interaction – systems that need software developers, systems that need annotations, systems that need no annotation and semi-supervised systems.

Muslea [13] (has developed RISE – *Repository of On-line Information Sources in Information Extraction Tasks*) classifies extractors in three classes based on the input document type and the structures and constraints of the extraction rule. The first extractors class uses uses extraction patterns based on syntactic/semantic constraints; the second class called WI – *wrapper induction* uses rules based on delimitations and the third class uses both delimitations and syntactic/semantic constraints.

IE systems can be analyzed taking into consideration the following parameters: the problem difficulty, the technologies used, the user effort for the training process and the necessity of porting the system on different domains.

2.2. Clustering methods

The goal of clustering methods is to group elements sharing common information. The purpose of clustering is to gather the elements that are most similar between them, but less similar to all the others. [14] Clustering methods can be divided in two classes: partitioning methods and hierarchical methods. Each of the class consists of a set of different algorithms for identifying clusters. [15] [16]

Hierarchical methods proceed by stages producing a sequence of partitions, each corresponding to a different number of clusters. They can be 'agglomerative', meaning that groups are merged, or 'divisive', in which one or more groups are split at each stage.

Partitioning methods move observations iteratively from one group to another, starting from an initial partition. The number of partitions can be specified in advance and does not change during the iteration. One of the most common partitioning is the K-means algorithm [17], that will be used in our cluster analysis during the paper work.

The basic **k-means algorithm** for finding K clusters have the following stages:

- 1. select K points as the initial centroids
- 2. assign all points to the closest centroid
- 3. recompute the centroid of each cluster
- 4. repeat steps 2 and 3 until the centroids won't change

The cluster population is: [17]

$$P = \{p\}_{j=1,k} \tag{1}$$

where $p_j = \{v / x_v \in Cluster_j\}$ and

$$X = [x_1, x_2, \cdots x_n] \in \mathbb{R}^{m \times n} \tag{2}$$

is the data points organized as a matrix column .

The cluster centroid is the point where the parameter value is the average of all the parameters values: [18]

$$c_j = \frac{1}{n_j} \cdot \sum_{v \in p_j} x_v \tag{3}$$

where n_j is number of elements from p_j .

The distance used by the K-means algorithm is the Euclidean Distance: [19]

$$d(i,j) = \sqrt{(|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + ... + |x_{ip} - x_{jp}|^2)} \tag{4}$$

3. System Architecture

Information extraction applied on input documents is performed by an IE System [20] that consists of a set of original algorithms. Our IE system has the following features: (Fig. 1)

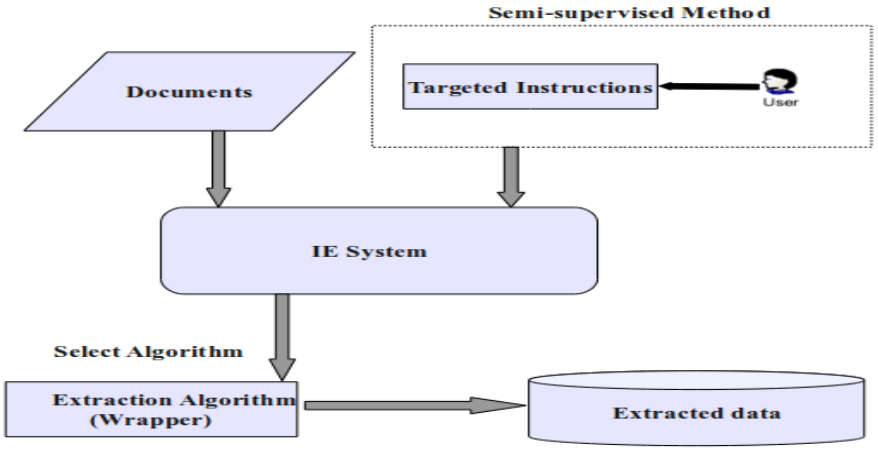

Fig. 1. IE System Architecture

– **IE method**: our system uses a semi-supervised method – receives a general pattern on which builds the regular expression

– **Top-down approach**: it starts from the most general regular expressions applying specific terms on its results

– **Training algorithm**: defined by pattern extraction
– **Pattern**: is the generic name for a language programming instruction ; no sub-patterns are allowed
– **Tokening**: word level
– **Extraction rule type**: described by a regular expression and a text window that covers an area of tokens before the text target and an area of tokens after the text target

Our IE System receives input documents that represent language programming tutorials; these kind of input documents can be classified as semi-structured. The target information are the syntax structures for various lexical and syntactical constructions allowed by the programming language.

The IE system contains a set of four wrapper algorithms, one for each class of structure identified in a programming language. Our extraction algorithms covers the following classes of lexical and syntactic structures :
– Lexical structures: keywords, operators and their precedence, comments
– Syntactic structures: based on the language programming lexicon

As case studies for the IE system described above we will present in this paper only two algorithms – the one for keywords extraction and the one for syntactic structures extraction. Our algorithms have the following phases.

IE Algorithm for keywords extractions
Step 1: Load input document
Step 2: Identify area for the target pattern
- identify looking pattern
- store all the occurrences of the pattern and then identify the most relevant occurrence
- define a window text based on a proposed estimation

Step 3: Extract keywords
- extract the most relevant result using clustering methods
- make cluster analysis based on a set of proposed parameters

Algorithm for syntactic structures
Step 1: Load input document
Step 2: Identify area for the target pattern
- identify looking pattern
- store all the occurrences of the pattern and then identify the most relevant occurrence
- define a window text based on a proposed estimation

Step 3: Extract syntax
- look for special characters that can indicate that the piece of text is containing the syntax of the given structured. These special characters include:
  - [] - marks in general the optionals

- {} - marks as usual blocks delimits
- ; - marks in general instruction's ending

- format the results
  - code the special characters to a predefined set
  - identify reserved words and mark them as special ones
  - identify all the identifiers and mark them uniquely

4. Experimental results

4.1. Keywords extraction

The proposed algorithm has been applied on a set of four input documents, listed below:

1) *Draft Standard for the Functional Verification Language e* [21]
2) *C++ Language Tutorial* [22]
3) *The JavaTM Language Specification Third Edition* [23]
4) *SystemVerilog 3.1a Language Reference Manual* [24]

The algorithm extracts a number of results, as described by Table 1.

Table 1

Extracted results for language programming keywords

Document	***Number of extracted results***
Draft Standard for the Functional Verification Language e	39
C++ Language Tutorial	43
The JavaTM Language Specification Third Edition	83
SystemVerilog 3.1a Language Reference Manual	119

In order to identify the most relevant section extracted we have applied a clustering algorithm on the results. The clustering algorithm was K-means [16] with the K parameter chosen as in the following table:

Table 2

K-parameter values

Document	***K parameter***
Draft Standard for the Functional Verification Language e	6
C++ Language Tutorial	6
The JavaTM Language Specification Third Edition	11
SystemVerilog 3.1a Language Reference Manual	9

We have chosen the K-means algorithm because of its simplicity and good time and space complexity. The space complexity is $O(mn)$ where m is the number of points and n is the number of attributes. The time requirements are

$O(I*K*m*n)$ where I is the number of iterations requires for convergence. I is typically a small value (5-10) and can be easily bounded as most changes occur in the first few iterations. The algorithm is linear in *m*, the number of points, and is efficient and simple, as long as the number of clusters is significantly less than *m*.

Cluster analysis is performed on a set of qualities indexes. [25] [26] For the purpose of our case studies we proposed the following set of such parameters:

a) Cluster density - this parameter has the following interpretation: the larger the gap the smaller the similarity between members.

b) Cluster relevance based on centroids positions versus the observations averages - the higher the difference between the two parameters, the higher the cluster probability to contain irrelevant points. Fig.2 shows the results of computed centroids versus observations averages for each input document.

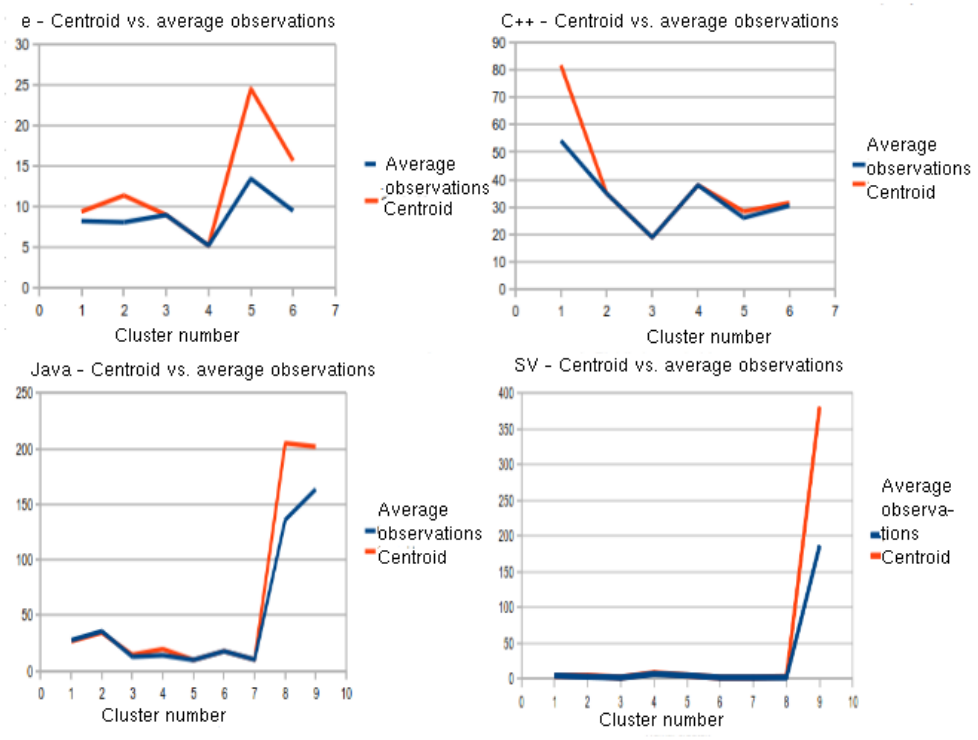

Fig. 2. Centroids values vs. Observations averages

c) Cluster size – the closer to a threshold value, the higher the similarity.

We propose the following size weights regarding the keywords of a language programming (Table 3):

Table 3

Keywords – cluster size weights

Cluster size	*Size weights - keywords*
[1 .. 2]	0.10
[3 .. 7]	0.40
[7 .. 10]	0.80
[10 .. 15]	0.50
[15 .. 30]	0.30
[30 .. 50]	0.10

d) Cluster relevance – we propose the following distance for computing the cluster relevance and we will call it *REL*:

$$REL_{c_j} = \frac{1}{k} \cdot \sum_{i=1}^{m} x_i * P_i, j = 1..N \quad (4)$$

where:

N = number of clusters
k = number of points inside a cluster
x = observations, defined as the number of lines from each cluster point
P = cluster size weights, as defined in Table 3.

The computed relevance parameter are listed in Table 4.

Table 4

Cluster relevance parameter for keywords

Document		
	Cluster number	*Cluster relevance REL*
Draft Standard for the Functional Verification Language e	1	1.37
	2	3.34
	3	3.30
	4	1.05
	5	8.96
	6	5.93
C++ Language Tutorial	1	22.28
	2	14
	3	9.5
	4	15.2
	5	10.72
	6	24.21

Document		
	Cluster number	*Cluster relevance REL*
The JavaTM Language Specification Third Edition	1	19.56
	2	24.60
	3	6.44
	4	8.17
	5	5
	6	12.92
	7	3.9
	8	47.64
	9	57.49
SystemVerilog 3.1a Language Reference Manual	1	0.40
	2	0.31
	3	0.18
	4	1.47
	5	0.50
	6	0.20
	7	0.20
	8	0.20
	9	55.59

The following cluster analysis has been done based on the results of the above defined parameters :

a) ***Draft Standard for the Functional Verification Language e***

- very small cluster size: $dimC_3 = 2$
- higher scattering degree: C_5 and C_6 - more than 6000 units
- extreme values for the relevance parameter: C_1 and C_4 (very small values), C_5 (very high value)
- extreme value for centroids vs. observations average (high value) for cluster C_5
- the most relevant cluster becomes C_2

b) ***C++ Language Tutorial***

- very small cluster size: $dimC_3 = dimC_4 = 1$
- higher scattering degree: C_5 and C_6 - more than 1000 units

- extreme values for the relevance parameter: C_3 (the smallest value)
- the most relevant clusters became C_1 and C_2 - we also take into consideration cluster C_1 because it has a good density even if it represents an extreme value for the parameter centroids vs. observations average

c) ***The JavaTM Language Specification Third Edition***

- very small cluster size: C_5, C_6, C_9
- higher scattering degree: C_2 and C_8 - more than 5000 units
- extreme values for the relevance parameter: C_3, C_4, C_5, C_7 (the smallest values)
- extreme value for centroids vs. observations average for cluster C_8
- the most relevant cluster becomes C_1

d) ***SystemVerilog 3.1a Language Reference Manual***

- very small cluster size: $dim C_1, C_4, C_5, C_6, C_7 < 10$, the rest having an average size of 20
- the centroids values vs. observations average are very small (up to 10), the only cluster with a high value being C_9
- the most relevant cluster becomes C_9

4.2. Syntactic structures extraction

The proposed algorithm has been applied on a *Draft Standard for the Functional Verification Language e* [21] document. The algorithm works on a predefined concepts lexicon of the programming language. This lexicon contains data structures and actions that can be implemented in a source code. The concepts defined in the input document have the following hierarchy. (Fig. 3) The algorithm has been applied on a set from each concept.

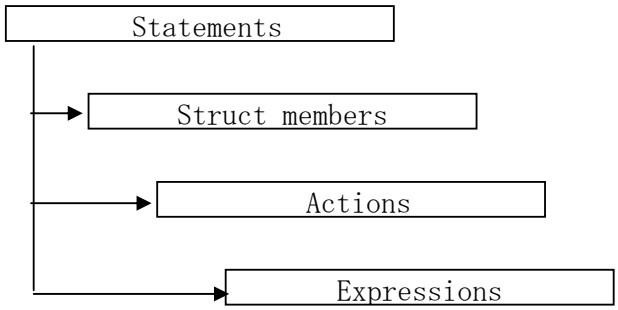

Fig. 3. Key concepts of the 'e' language programming

In Table 5 he have the algorithm results only for a subset of the statements concept:

Table 5

Extracted syntax for syntactic structures of the input document

Structure syntax as written in the input document	Structure syntax after being identified and processed
struct *struct-type* [**like** *base-struct-type*] { [*struct-member*; ...]}	'struct' ID LBRACKET 'like' ID RBRACKET LBRACE LBRACKET ID DOT ID
extend [*struct-subtype*] *base-struct-type* { [*struct-member*; ...]}	'extend' LBRACKET ID RBRACKET ID LBRACE LBRACKET ID DOT ID
unit *unit-type* [**like** *base-unit-type*] { [*unit-member*; ...]}	'unit' ID LBRACKET 'like' ID RBRACKET LBRACE LBRACKET ID DOT ID

6. Conclusions

The IE system algorithms produced good results for all the lexical and syntactic structures of the language programming. Our system proved that even if we have very different input documents it can decide which section is about lexical structures without any prior information. As an overview of the quality indexes parameters proposed in this paper we may conclude that the the distance proposed – REL – is a very good indicator of clusters relevance for the lexical structure analyzed.

As future developments we propose the extension of clustering analysis with other clustering algorithms and new distances in order to perform a comparative analysis.

Acknowledgements

The work has been funded by the Sectoral Operational Program Human resources Development 2007-2013 of the Romanian Ministry of Labor, Family and Social Protection Through the Financial Agreement POSDRU/6/1.5/S/16.

R E F E R E N C E S

[1] *R. Baeza-Yates, B. Ribeiro-Neto*, "Modern Information Retrieval", ACM Press, New York, 1999

[2] *E. Riloff*, "Automatically constructing a dictionary for information extraction tasks", Proceedings of the 11th National Conference of Artificial Intelligence (AAAI-93), 00.811-816, AAAI Press / The MIT Press, 1993

[3] *S. Huffman*, "Learning information extraction patterns from examples. Connections, statistical, and symbolic Approaches to Learning for Natural Language Processing", Spingler-Verlag, 1996

[4] *J. Kim, D. Moldovan,* "Acquisition of linguistic patterns for knowledge-based information extraction", IEEE Transactions on Knowledge and Data Engineering **7(5)**: 713-724, 1995
[5] *G. Krupka*, "Description of the SRA system as used for MUC6", Proceedings of the 6th Message Understanding Conference (MUC6), pp. 221-235, 1995
[6] *K. Slonnenger, B. Kurtz, "*Formal Syntax and Semantics of Programming Language", Addison-Wesley Publishing Company, ISBN 0-201-65697-3, 1995
[7] *S. Soderland*, "Learning information extraction rules for semi-structured and free text", Journal of Machine Learning, **34(1-3)**, pp. 233-272, 1999
[8] *M. Califf, R. Mooney,* "Relational learning of pattern-match rules for information extraction", Proceedings of AAAI Spring Symposium on Applying Machine Learning to Discourse Processing Standford, California, 1998
[9] *D. Freitag*, "Information extraction from HTML: Application of a general learning approach", Proceedings of the 15th International Conference on Artificial Intelligence (IJCAI), 1998
[10] *N. Kushmerick, D. Weld, R. Doorenbos*, "Wrapper induction for information extraction", Proceedings of the 15th International Conference on Artificial Intelligence (IJCAI), 1998
[11] *C.N., Hsu, M. Dung*, "Generating finite-state transducers for semi-structured data extraction from the web", Journal on Information Systems **23(8)**, pp. 521-538, 1998
[12] *C.H. Chang, C.N. Hsu, S.C. Lui,* "Automatic information extraction from semi-structured Web pages by pattern discovery", Decision Support Systems Journal, **35(1)**, pp. 129-147, 2003
[13] *I. Muslea, S. Minton, C. Knoblock, "*A hierarchical approach to wrapper induction", Proceedings of the 3rd International Conference on Autonomous Agents (AA-99), 1999
[14] *M. Murty, A. Jain, P. Flyn*, "Data clustering: A review", ACM Computing Surveys **31(3)**, 1999
[15] *J.A. Hartigan,* "Clustering Algorithms", Wiley, New York, 1975
[16] *L. Kaufman, P.J. Rousseeuw,* "Finding Groups of Data", Wiley, New York, 1990
[17] *J.B. MacQueen,* "Some Methods for Classification and Analysis of Multivariate Observations", Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, pp. 281-297, 1967
[18] *J. Ye*, "Numerical Linear Algebra for Data Exploration – Clustering", CSE 494 CSE/CBS 598. 2007
[19] ****Wolfram Mathematica Documentation Center*, "Distance and Similarity Measures", 2011 http://reference.wolfram.com/mathematica/guide/DistanceAndSimilarityMeasures.html
[20] *J. Hobbs*, "The Generic Information Extraction System", Proceedings of the 5th Message Understanding Conference (MUC-5), Morgan Kaufman Publishers, San Mateo, California, 1994
[21] **** *Design Automation Standard Committee of the IEEE Computer Science*, "IEEE P1647™/D9 Draft Standard for the Functional Verification Language e". www.ieee1647.org/downloads/P1647_Draft_6_071214.pdf
[22] *J. Soulie*, "C++ Language Tutorial", 2007 http://www.cplusplus.com/doc/tutorial/
[23] *J. Gosling, B. Joy, G. Steele, G. Bracha*, The Java™ Language Specification Third Edition, ISSN 0-321-24678-0, Santa Clara, California, 2005
[24] **** Accellera Organization, "SystemVerilog 3.1a Language Reference Manual" *www.eda.org/sv/SystemVerilog_3.1a.pdf*
[25] *B. Raskuti, C. Leckie*, "An Evaluation of Criteria for Measuring the Quality of Clusters", Proceedings of the 16th International Joint Conference on Artificial Intelligence, pp. 905-910, ISBN 1-55860-613-0, 1999
[26] *R.C. Dubes, A.K. Jain*, "Validity Studies in Clustering Methodologies", Pattern Recognition, **11**, pp. 235-254, 1979.