HAL
open science

Unstructured Data Integration through Automata-Driven Information Extraction

Maroun Abi Assaf, Kablan Barbar, Youakim Badr, Mahmoud Rammal

To cite this version:

Maroun Abi Assaf, Kablan Barbar, Youakim Badr, Mahmoud Rammal. Unstructured Data Integration through Automata-Driven Information Extraction. International Journal of Computer Science and Business Informatics, 2013, 2 (1), pp.1-14. hal-01169150

HAL Id: hal-01169150

https://hal.science/hal-01169150v1

Submitted on 6 Oct 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

International Journal of Computer Science and Business Informatics

IJCSBI.ORG

Unstructured Data Integration through Automata-Driven Information Extraction

Maroun Abi Assaf, Kablan Barbar
Dept of applied mathematics
Faculty of Sciences II – Lebanese University
Fanar, Lebanon

Youakim Badr
INSA – LYON, France

Mahmoud Rammal
Legal Law Center – Lebanese University
Sin-Fil, Lebanon

ABSTRACT
Extracting information from plain text and restructuring them into relational databases raise a challenge as how to locate relevant information and update database records accordingly. In this paper, we propose a wrapper to efficiently extract information from unstructured documents, containing plain text expressed with natural-like language. Our extraction approach is based on the automata formalism to describe the wrapping process running from text documents to Databases. As usual, relevant information in the text document are delimited by regular expressions, which define the extracting automaton. Each automaton is enriched by an output function that automatically generates SQL queries synchronized with the extracting process in order to insert extracted data into database records. We validate our extraction approach with automaton-based prototype to extract legal information about Lebanese official journal decrees and automatically insert them into a relational database.

Keywords
Wrappers, Regular Expressions, Automata, SQL Language, XML.

1. INTRODUCTION
Several approaches to generating wrappers have been introduced in the literature to extract information from structured, semi-structured and unstructured documents.

In [2], the authors create wrappers for XML Paragraph-Centric. The extraction patterns are expressed with regular expressions and translated into automata. The automata then carry out the information extraction and produce tuples of values. In [1], the extraction system is based on two automata. The first automaton decomposes grammatically the given text into noun groups, verbs groups and others words groups. On the other hand,

the extraction patterns are encoded into a recognizing patterns automaton that takes as input all the preceding groups and produces the resulting entries. In [4], the system runs in two phases, the learning and the testing phases. The learning phase produces the lexical pattern rules based on a training collection of reports. The testing phase has three components a noun phrasing an automaton and a neutral network. The noun phrasing runs as a syntactical analyzer and produces a list of relevant information. The automaton makes a matching operation and the neutral network predicts the matched phrases' most possible entity type. In [6] an approach to automatically locate data-rich regions and to extract relevant attribute-value pairs of database records from web pages across different sites is presented. This approach relies on the fact that the attribute-value pairs of the records usually occurs next to each other's in well-designed web pages.

We note that all these approaches do not deal with the context of relevant information for plain text documents and do not integrate in a unique formalism both the extracting and the database writing processes.

In contrast to these approaches, we propose an extraction approach, a.k.a. wrapper, that deal with plain text without a predefined structure (i.e., XML and HTML) and without a natural language pre-processing (i.e., lexical, semantic analysis) to identify phase structures and relevant information. Our extraction approach is based on a formal description of the wrapping process from text document to databases by means of automata output functions that generate automatically SQL queries synchronized with the extracting process. Moreover, on the formal aspect, we express with the same formalism both the extracting and the database mapping processes. We particularly rely on the automata and regular expressions formalism to express the structure of relevant data and locate them. Each relevant information is described by a regular expression. The sequence separating two relevant information is also formulated with regular expressions. After recognizing relevant information, a SQL insert query is generated. For each relevant information, we specify four elements; its regular expression, the regular expressions for its preceding and proceeding sequence and the corresponding SQL query. All, the regular expressions forms the extracting automaton and the SQL queries are included in the output function of the automaton. Then, the automata cover both the extracting and the database writing processes. The research is a first step to build automata generic system based on attributed grammars like in [3].

The remaining of the paper is organized as follows. Section 2 establishes a panorama of related works. Section 3 recalls the formalism of automata with output functions and shows how we translate the relevant information descriptions into extracting automata. In section 4, we present our experimental results to extract legal decrees from the text document of the

Lebanese official journal and insert them into a database. In section 5, we conclude our work and provide some research trends.

2. RELATED WORKS

Several approaches have been proposed to extract data from text document using automata. In [1], the authors describe a finite-state processor for information extraction from real-world text, named FASTUS. FASTUS uses a pre-defined finite-state machine (Recognizing Phrases Automaton) to decompose the sentences of the real-world text into noun groups, verb groups, and several other critical word classes' phrases which will form the candidate list of phrases. Next, the given patterns of interest (extraction pattern) is encoded into a finite state machine (Recognizing Patterns Automaton), and the resulted candidate list of phrases is passed as input to this automaton that identifies phrasal matches with the given patterns of interest and returns the necessary information. In [2], the authors deal with the problem of creating a wrapper for XML paragraph-centric documents named Xtractor. They propose a specification language to write the extraction patterns, based on Regular Expressions, but is more simple and easier to read than Regular Expressions themselves. This specification language includes a set of meta-words that are referenced in a provided dictionary for the domain of interest, and associates lexemes with these meta-words. The extraction pattern, which is a regular language, is translated into an equivalent finite-state automaton in the first phase. Then in the second phase, the finite-state automaton will carry out the information extraction, and the result is returned as data tuples of attributes.

In a similar approach, in [4] authors deals with the problem of extracting meaningful entities, such as person names, addresses, narcotic drugs, or vehicle names, from free texts of police narrative reports through a neural network-based entity extractor. The system has three major components; Noun phrasing, Finite state machine and lexical lookup, and Neural network. The Noun phrasing component is a form of a syntactical analyzer that extracts the noun phrases from the reports and will form the candidate list of relevant information values. The Finite state machine and lexical lookup component is a finite state machine that finds matches between the words of the extracted noun phrases list and the items of a provided lexicon that contains lists of possible values of the entities of interest. The Neural network component, a feedforward/backpropagation neural network, predicts the matched phrases' most possible entity type. The system also has two states; a learning state where it identifies the lexical pattern rules (extraction pattern rules) based on a training collection of reports, where the entities are manually identified by a human, and saves these rules in the neutral network as synaptic weight, and a testing state where the system

extracts the phrases and find item matches from the reports and predicts their entity type based on the obtained extraction pattern rules from the learning state.

In [6], authors describe a web information extraction system that extracts attribute-value pairs, like Product code, Manufacturer, Price, and Description of products from web pages that present product descriptions. The system introduces a notion called "Structural-Semantic Entropy" to identify if a web page contains a data-rich region or not. The data-rich region of a web page is the region that contains the actual relevant information. "Structural-Semantic Entropy" means the measure of randomness of the leaf nodes that contain a semantic role of a given non-leaf node. A leaf node contains a semantic role if it contains a keyword label associated with the attribute-value pairs. This keyword label list is known beforehand and given as input to the wrapper. The more this randomness is higher the more the non-leaf node that is inspected is likely to be a data-rich node. For every non-leaf node in the DOM tree of the web page its "Structural-Semantic Entropy" is calculated, and every node having a "Structural-Semantic Entropy" higher than a given threshold, which is learned by experimentation, is considered to be a data-rich node that contains a record. Finally, for every discovered data-rich node, the contents of the next text-nodes of its leaf nodes that are annotated with a semantic role are usually extracted as the value of the semantic roles (or attributes).

We conclude that most of these representative approaches do not provide formal description of the whole extracting and database writing processes and they do not run on non-structured text documents.

3. METHODOLOGY

Automata are used to support syntactical analyzer in the compiling process of programming languages. They act like a virtual machine to recognize words belonging to rational languages by passing from one state to another according to a symbol of the initial alphabet.

Formally, an automaton is a 5-tuple $A=< \sum, S, s_0, T, \delta >$ where:

- $\sum$ is the alphabet
- S is the set of states
- s_0 is the initial state and an element of S
- T is the set of terminal states and a subset of S
- δ is the transition function defined by:

$$\delta : S \times \sum \rightarrow S$$

The transition function is represented by a graph and leads the recognizing process. As in [5], we can define a semantic for automata transition. A semantic has the same domain as the transition function but it associates

semantic values with each transition. This technique allows the enrichment of the automaton and the possibility of executing of complementary semantic operations in parallel with the automaton transition function. Then, we formalize an extended automaton by a 7-tuple $A=< \sum, S, s_0, T, \delta, D, \lambda >$where $\sum$, S, s_0, T, δ remains the same as preceding and where D is a semantic domain and λ is the semantic function defined by:

$$\lambda : S \times \sum \rightarrow D$$

As application of the data integration process from text document to database, we consider the data integration process applied to the Lebanese Official Journal (LOJ) in the Center of Documentation and Research of the Lebanese University. The main unit in LOJ is the decree that contains general information like decree number, date,…, many 'based on' clauses, many articles, a 'decrees' clause, a place, a date and many signatures (see Figure 1). Each week, the center receives an electronic copy of LOJ as a text document. The relevant information in the decrees are selected and transferred manually to database by a copy-paste operation from text document to database. This operation takes too much time and produces errors. In the center, they want to develop a program that both selects automatically the relevant information from the decrees of LOJ and insert them in the database. That is why we are going now to use automata to formalize the data integration process for LOJ decrees.

The relational schema of the database associated with the LOJ is described as follows:

decree(decree_id_pk, type, number, title, location, date)
basedon(basedon_id_pk, decree_id_fk, value)
andafter(andafter_id_pk, decree_id_fk, value)
article(article_id_pk, decree_id_fk, value)
signature(signature_id_pk, decree_id_fk, value)

The list of relevant information with their values from the example decree of Figure 1, and their corresponding regular expressions and database operations are given in Table 1.

In addition, we give finite state automata where the transition function is represented by a graph and the output function is formally identified. Each state in the automata is implemented as a string to which we add the symbol recognized at each transition. The domain D contains four operations:

1. insert: to insert a table record
2. update: to update a field of a table record
3. concatenate: to concatenate two strings
4. clear: to clear the string state.

Decree No. 6873
Giving the Ministry of Finance treasury advance to pay the last installment of a series of differences of ranks and salaries of civilian and military

The President of the Republic,
Based on the Constitution,
Based on the law of public accounting, as amended, particularly Article 203 and subsequent of it,
Based on the suggestion of the Minister of Finance,
And after the approval of the Council of Ministers on 15/11/2011,

Decrees the following:

Article 1: To give the Ministry of Finance treasury advance worth / 500.000.000.000 / LP (only five hundred billion pounds).
Article 2: It is not allowed for the party taking the advance, to use it for any other end other than the one that is given for it.
Article 3: The Director of Treasury, director of the disbursements, director of budget and expenditure control in the Ministry of Finance, and the director of Administrative Affairs at the Directorate General of the Ministry of Finance, within their competences, should pursue the repayment of the advance.
Article 4: This Decree shall be published and reported where needed.

Baabda, the 18th of November 2011
Signature: Michel Sleiman

Issued by the President of the Republic
President of the Ministers' Council
Signature: Muhammad NajibMikati

Minister of Finance
Signature: Muhammad Al Safady

Figure 1. A decree of the Lebanese Official Journal

Table 1. Relevant Information

Relevant Information	Value in Example Decree	Preceding Regular Expression	Wanted Entity' Regular Expression	Following Regular Expression	Database Operation
Type	"Decree"	ε	[a-zA-Z]+	No.	Insert a new record in the table decree and update the type field in this record
Number	6873	No.	[0-9]+	[a-zA-Z]	Update the number field of the current record in the decree table
Title	"Giving the Ministry of Finance treasury advance to pay the last installment of a series of	[0-9]*	[a-zA-Z]+	The President of the Republic,	Update the title field of the current record in the decree table

Relevant Information	Value in Example Decree	Preceding Regular Expression	Wanted Entity' Regular Expression	Following Regular Expression	Database Operation
	differences of ranks and salaries of civilian and military"				
BasedOn	"the Constitution"… "the suggestion of the Minister of Finance"	Based on	[a-zA-Z,.0-9]+	Based on | And after	Insert a new record in the basedOn table and update the value field in this record
AndAfter	"the approval of the Council of Ministers on 15/11/2011"	And after	[a-zA-Z,.0-9]+	And after | Decrees the following:	Insert a new record in the andAfter table and update the value field in this record
Article	"To give the Ministry of Finance treasury advance worth / 500.000.000.000 / LP (only five hundred billion pounds)." …	Article [0-9]*:	[a-zA-Z,.0-9]+	Article [0-9]*: | [a-zA-Z]*, the [0-9]{2}th of (January|…|December) [0-9]{4}	Insert a new record in the article table and update the value field in this record
Location	"Baabda"	Article [0-9]*: This Decree shall be published and reported where needed.	[a-zA-Z]+	, the [0-9]{2}th of (January|…|December) [0-9]{4}	Update the location field in current record of the decree table
Date	"the 18th of November 2011"	[a-zA-Z]+,	the [0-9]{2}th of (January|…|December) [0-9]{4}	Signature:	Update the date field in current record of the decree table
Signature	"Michel Sleiman", "Muhammad NajibMikati", "Muhammad Al Safady"	Signature:	[a-zA-Z]+	Issued by | Minister	Insert a new record in the signature table and update the value field in this record

The automata associated with the decree relevant information are:

- **FSA(Type):**

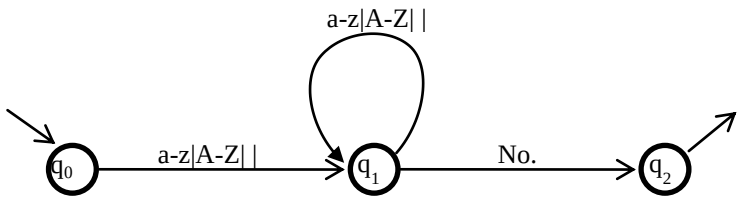

Figure 2. Type Finite State Automaton

D = {Insert(qi, 'decree', 'type'), Concatenate(qi, c) }
$\lambda : Q \times \sum \rightarrow D$
$\lambda(q_0, c)$ = Concatenate(q_0, c) where c $\in$a-z|A-Z| |
$\lambda(q_1, c)$ = Concatenate(q_1, c) where c $\in$a-z|A-Z| |
$\lambda(q_1$, No.) = Insert(q_0+q_1, 'decree', 'type')

This automaton will extract all the alphabetical characters including a space character between the start of the decree and a "No." keyword and insert them as the value of the 'type' attribute of a new record in the 'decree' table.

- **FSA(Number):**

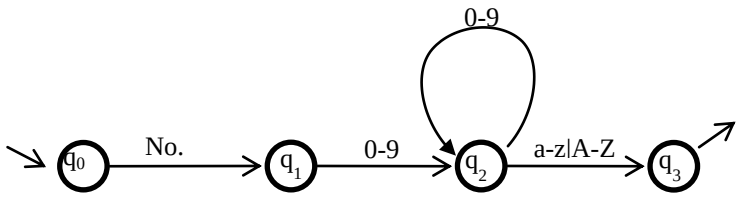

Figure 3. Number Finite State Automaton

D= { Update(q_i, 'decree', 'number'), Concatenate(q_i, c) }
$\lambda : Q \times \sum \rightarrow D$
$\lambda(q_0$, No.) = ε
$\lambda(q_1, c)$ = Concatenate(q_1, c) where c $\in$ 0-9
$\lambda(q_2, c)$ = Concatenate(q_2, c) where c $\in$ 0-9
$\lambda(q_2$, a-z|A-Z) = Update(q_1+q_2, 'decree', 'number')

This automaton will extract all the numerical characters between the "No." keyword and any alphabetical character and update the 'number' attribute of the last inserted record of the 'decree' table with this value.

- **FSA(Title):**

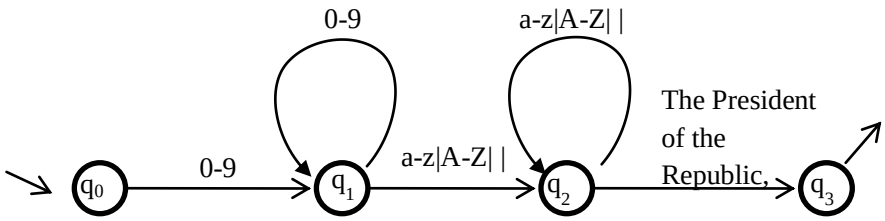

Figure 4. Title Finite State Automaton

D = { Update(q_i, 'decree', 'title'), Concatenate(q_i, c) }
λ : Q x ∑ → D
λ(q_0, 0-9) = ε
λ(q_1, 0-9) = ε
λ(q_1, c) = Concatenate(q_1, c) where c ∈a-z|A-Z| |
λ(q_2, c) = Concatenate(q_2, c) where c ∈a-z|A-Z| |
λ(q_2, The President of the Republic,) = Update(q_1+q_2, 'decree', 'title')

This automaton will extract all the alphabetical characters including a space character between the delimiter formed by any numerical characters and the keywords "The President of the Republic,", and updatethe 'title' attribute of the last inserted record of the 'decree' table with this value.

- **FSA(BasedOn):**

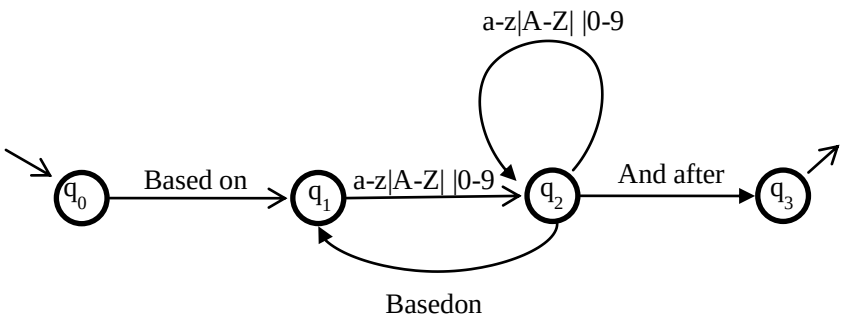

Figure 5. BasedOn Finite State Automaton

D = {Insert(q_i, 'basedon', 'value'), Concatenate(q_i, c),Clear(q_i)}

λ : Q x ∑ → D
λ(q_0, Based on) = ε
λ(q_1, c) = Concatenate(q_1, c) where c ∈a-z|A-Z| |0-9
λ(q_2, c) = Concatenate(q_2, c) where c ∈a-z|A-Z| |0-9
λ(q_2, Based on) = Insert (q_1+q_2, 'basedon', 'value'); Clear(q_1); Clear(q_2)
λ(q_2, And after) = Insert(q_1+q_2, 'basedon', 'value')

This automaton will extract all the alphanumeric characters including a space character between every "Based on" keyword and another "Based on" or "And after" keywords, and insert them as the value of the 'value' attribute of a new record in the 'basedon' table.

- **FSA(AndAfter):**

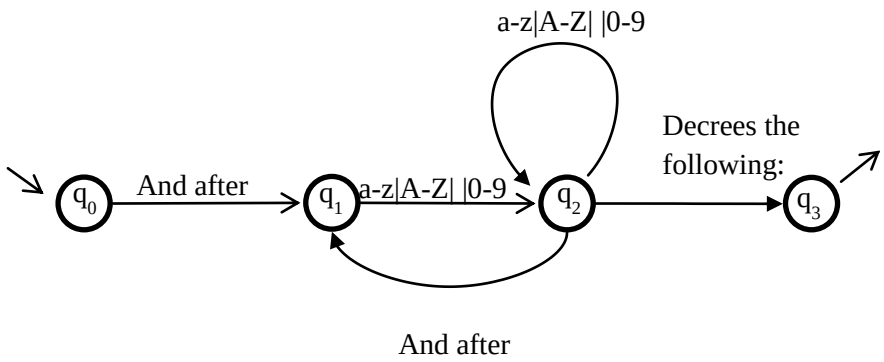

Figure 6. AndAfter Finite State Automaton

D = {Insert(q_i, 'andafter', 'value'),Concatenate(q_i, c), Clear(q_i)}
$\lambda : Q \times \sum \rightarrow D$
$\lambda(q_0$, And after) = ε
$\lambda(q_1$, c) = Concatenate(q_1, c) where c ∈a-z|A-Z| |
$\lambda(q_2$, c) = Concatenate(q_2, c) where c ∈a-z|A-Z| |
$\lambda(q_2$, And after) = Insert (q_1+q_2, 'andafter', 'value'); Clear(q_1); Clear(q_2)
$\lambda(q_2$, Decrees the following:) = Insert(q_1+q_2,'andafter', 'value'); Clear(q_1); Clear(q_2)

This automaton will extract all the alphanumeric characters including a space character between every "And after" keyword and another "And after" or "Decrees the following:" keywords, and insert them as the value of the 'value' attribute of a new record in the 'andafter' table.

- **FSA(Article):**

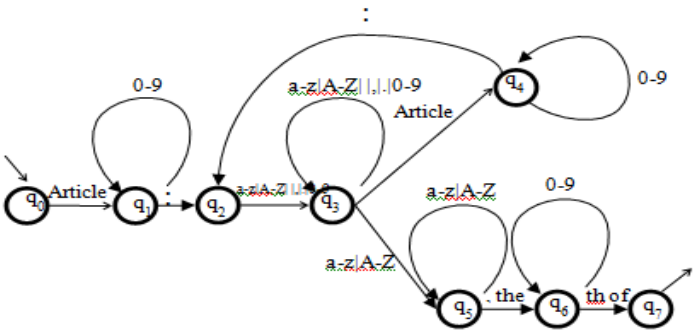

Figure 7. Article Finite State Automaton

D = {Insert(q_i, 'article', 'value'), Concatenate(q_i, c), Clear(q_i)}
λ : Q x $\sum \rightarrow$ D
$\lambda(q_0$, Article) = ε
$\lambda(q_1$, 0-9) = ε
$\lambda(q_1$, :) = ε
$\lambda(q_2$, c) = Concatenate(q_2, c) where c $\in$a-z|A-Z| |,|.
$\lambda(q_3$, c) = Concatenate(q_3, c) where c $\in$a-z|A-Z| |,|.
$\lambda(q_3$, Article) = ε
$\lambda(q_4$, 0-9) = ε
$\lambda(q_4$, :) = Insert (q_2+q_3, 'article', 'value'); Clear(q_2); Clear(q_3)
$\lambda(q_3$, a-z|A-Z) = ε
$\lambda(q_5$, a-z|A-Z) = ε
$\lambda(q_5$, , the) = ε
$\lambda(q_6$, 0-9) = ε
$\lambda(q_6$, th of) = Insert(q_2+q_3, 'article', 'value');

This automaton will extract all the alphanumeric characters including a space, comma, and dot characters between every "Article n:" keyword (where n is a number) and another "Article n:" keyword or any word followed by a date delimiter, and insert them as the value of the 'value' attribute of a new record in the 'article' table.

- **FSA(Location):**

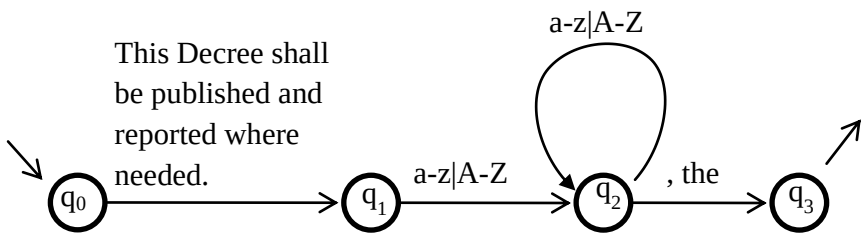

Figure 8. Location Finite State Automaton

D = { Update(q_i, 'decree', 'location'), Concatenate(q_i, c) }
λ : Q x $\sum \rightarrow$ D
$\lambda(q_0$, This Decree shall be published and reported where needed.) = ε
$\lambda(q_1$, c) = Concatenate(q_1, c) where c $\in$a-z|A-Z
$\lambda(q_2$, c) = Concatenate(q_2, c) where c $\in$a-z|A-Z
$\lambda(q_2$, , the) = Update(q_1+q_2, 'decree', 'location')

This automaton will extract the word between the two sequences "This Decree shall be published and reported where needed." and ", the" keywords and update the 'location' attribute of the last inserted record of the 'decree' table with this value.

- **FSA(Date):**

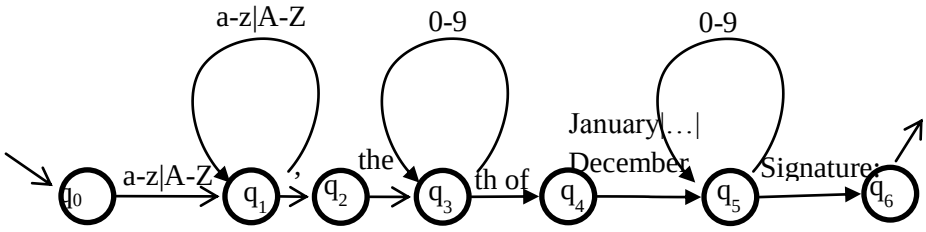

Figure 9. Date Finite State Automaton

D = { Update(q_i, 'decree', 'date'), Concatenate(q_i, c) }
λ : Q x ∑ → D
λ(q_0, a-z|A-Z) = ε
λ(q_1, a-z|A-Z) = ε
λ(q_1, ,) = ε
λ(q_2, the) = Concatenate(q_2, the)
λ(q_3, c) = Concatenate(q_3, c) where c ∈ 0-9
λ(q_3, th of) = Concatenate(q_3, th of)
λ(q_4, w) = Concatenate(q_4, w) where w ∈ January|…| December
λ(q_5, c) = Concatenate(q_5, c) where c ∈ 0-9
λ(q_5, Signature:) = Update(q_2+q_3+q_4+q_5, 'decree', 'date')

This automaton will extract the date according to the given format and that is located between any word and a "Signature:" keyword, and update the 'date' attribute of the last inserted record of the 'decree' table with this value.

- **FSA(Signature):**

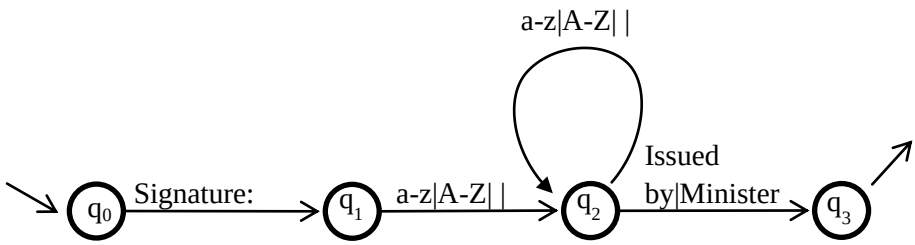

Figure 10. Signature Finite State Automaton

D = { Insert(q_i, 'signature', 'value'), Concatenate(q_i, c) }
λ : Q x ∑ → D
λ(q_0, Signature:) = ε
λ(q_1, c) = Concatenate(q_1, c) where c ∈a-z|A-Z| |
λ(q_2, c) = Concatenate(q_2, c) where c ∈a-z|A-Z| |
λ(q_2, Issued by|Minister) = Insert(q_1+q_2, 'signature', 'value')

This automaton will extract all the alphabetical characters including a space character between every "Signature:" and an "Issued by" or "Minister"

keywords, and insert them as the value of the 'value' attribute of a new record in the 'signature' table.

4. RESULTS

In order to validate our extraction approach, we have developed a .NET-based prototype, including two modules, the extracting module and the database writing module. The extracting module is the implementation of the regular expressions of all relevant information with their preceding and following sequences and also includes the output function for the SQL operations. It extracts relevant information values from decrees of the text document of the Lebanese official journal, and inserts these values in the SQL queries that are saved in an XML document. The database writing module takes as input the XML document for the SQL operations, selects a SQL query, makes a connection to the database and finally executes this query.

We have tested our prototype application on a sum of 100 decrees from several LOJs. Then we manually checked the database entries with the initial 100 decrees' texts for consistency and errors. We found that out of the 100 decrees, 98 decrees where discovered and extracted, and out of the 98 extracted decrees, 5 errors were found in some of the attributes. To measure the correctness of our prototype, we calculate two statistical classifications used in pattern recognition and information retrieval, the Precision (also called positive predictive value) which is the fraction of retrieved instances that are relevant, and the Recall (also known as sensitivity) which is the fraction of relevant instances that are retrieved. Based on the previously found numbers, the number of relevant instances is equal to the 98 extracted decrees minus the number of extracted decrees with errors which is 5, giving us a 93 for the number of relevant instances. This will result in a Precision that is equal to 93/98 which is approximately 95%, and a Recall that is equal to 93/100 which is 93%.

5. CONCLUSIONS

In order to deal with information extraction from plain text without a predefined structure (i.e., XML and HTML) and without a natural language pre-processing, we have presented an automata-based model to describe a wrapper that extracts meta-data from a text document containing the decrees of the Lebanese Official Journals, generates the corresponding SQL writing queries, and executes them in order to insert these extracted data into a relational database. In this model, we specify with the automata formalism both the extracting and the database writing processes. At present time, we have developed a prototype extractor by directly coding the Finite State

Automaton with the Output Function. The prototype extractor showed prominent results in extracting correct meta-data about the articles and successfully inserting them in their correct place in the relational database. In our future works, we would like to target the extracting process by considering conflicts when relevant information occurs in a wrong place. Moreover, we are investigating a new specification language for automata with output function in order to build a generic system that produces automatically the automata program or the extraction and writing programs.

6. REFERENCES

[1] D. E. Appelt, J. R. Hobbs, J. Bear, D Israel and M. Tyson, FASTUS: A Finite-state Processor for Information Extraction from Real-world Text. IJCAI 1993: Chambéry, 1993.

[2] Y. BADR, Xtractor: A Light Wrapper for XML Paragraph-Centric Documents. Proceedings of the 1st International Conference on Signal-Image Technology and Internet-Based Systems, SITIS 2005, 2005.

[3] K. Barbar, Automatic generation of XML-based editors for autonomic systems, International Journal of Autonomic Computing, v.1 n.3, p.246-262, May 2010

[4] M. Chau, J. J. Xu, H. Chen, Extracting Meaningful Entities from Police Narrative Reports. National Conference on Digital Government Research 2002, 2002.

[5] Knuth, D., "Semantics of context-free languages", Math. Systems Theory 5, 127-145, 1968.

[6] X. Zheng, Y. Gu, Y. Li, Data Extraction from Web Pages Based on Structural-Semantic Entropy. Proceedings of the 21st international conference companion on World Wide Web. Pages 93-102. ACM New York, NY, USA, 2012