ADAPTIVE FINITE STATE AUTOMATA APPLIED TO MORPHOLOGICAL ANALYSIS OF ARABIC WORDS

Iazzi Said[1], Iazzi Abderrazak[1], Yousfi Abdellah[2] and Bellafkih Mostafa[3]
[1]Laboratory GSCM-LRIT, Faculty of Science Mohammed V University in Rabat, Morocco
[2]Team ERADIASS, FSJES Souissi, Mohammed V University in Rabat, Morocco
[3]Department Telecommunications Systems, Networks and Services, STRS Laboratory,
National Institute of Posts and Telecommunications, Rabat, Morocco
E-Mail: iazzisaid@yahoo.fr

ABSTRACT

In this paper we propose a new implementation of an algorithm for the morphological analysis system of Arabic words which is based on the finite state automatons. This system combines the Buckwalter approach and the finite state automaton approach. We offer a reimplemented version of the tool which adapts the automatons in the form of a network of literates, from a set of words from the input Arabic language and from the tool which itself performs the morphological analysis, This system is based on very restricted dictionaries and searches for solutions in a global network using the Viterbi algorithm which proposed the optimal solutions to the analyzer. Our approach was tested on a corpus of 10,000 Arabic words. The results obtained are very impressive and show the importance of our new approach.

Keywords: finite state automaton, morphological analysis, correspondence tables, viterbi algorithm, machine learning.

1. INTRODUCTION

Computer morphological analysis is one of the first steps in the automatic processing of natural language texts. Tree representation or the uses of databases are just two ways to do this.In this work we used the representation of Arabic words by finite state automatons which are also relevant. Finite state automata are among the best known techniques for implementing morphological analysis [1]; [2] [15].

A morphological analyzer is a key resource for the storage and retrieval of information, particularly for morphologically rich languages. It is also useful for developing applications of automatic language processing such as documentary research, electronic dictionaries and marking systems etc. In order to promote further development, we have made our scripts, finite state automaton models, lexicons, meta-morphological rules, lists of generated verbs and nouns and freely available test data sets so that others can use and extend them [3] [4] [5] [6] [7] [9] [14].

Several works have been carried out with the goal of developing morphological analyzers of the Arabic language. In this work, we are more interested in the morphological analyzer based on a finite state automaton and in Buckwalter analysis [3].

Buckwalter Analyzer [3] has developed a system of morphological analysis of the Arabic language which determines all the possible segmentations of a word. Then looking for the results in the lists of radicals suffixes and prefixes. And also checking if the morphologies of each solution are compatible with each other by examining three correspondence tables between: prefix-radical, prefix-suffix and radical-suffix.

Among the difficulty of constructing the dictionary of radicals is the need to know different morphological transformations of each root. Moreover this dictionary is rather large because of each root one finds at least five radicals in this dictionary for example the root "كتب", in the dictionary of radicals of Buckwalter one finds: "كتاب" "كتيب" "كتوب" "كاتب" "كتب".

Finite state automata are mathematical constructions now quite classic in computer science and their application goes well beyond the automatic natural language processing.

For morphological analysis finite state automata are used to represent morphological relations in a given language [8] [13]. The purpose of a finite state automaton is to define the different possible combinations between the alphabets of a given lexicon. In our work, we have adapted viterbi algorithm to choose the possible optimal solutions to the analyzed words.

Among the drawbacks of the finite state automaton approach is that they use all the words of the lexicon to form the global network [13].

To remedy these drawbacks we propose a new approach which uses finite state automata and a restricted dictionary of radicals to perform the morphological analysis of a word.

The effectiveness of the proposed framework has been proven experimentally and the results obtained are comparable to the state of the art.

2. FINITE STATE AUTOMATONS APPLIED TO MORPHOLOGICAL ANALYSIS

Finite state automata are computational models made up of a finite number of states and no memory. So that a certain state of such an automaton only depends on the preceding state [8] [13]. In its simplest form, a finite state automaton accepts or rejects strings of a certain language.

A language consists of a set of strings recognized by a finite state automaton which is called a regular language [13]. A language which is accepted by a finite state automaton can be described by regular expressions [8].

A finite state automaton is composed of a finite set of states represented graphically by circles of a transition function describing the action which makes it possible to pass from one state to another these are the arrows, of a or more initial states and one or more end states.

A finite state automaton is therefore a directed graph where the nodes correspond to the states and the arcs to the arrows.

The morphological analyzer then only follows in the automaton a path according to the letters of the word analyzed and if a corresponding path exists, returns consequently all the possible sequences of this path.
The published finite state automaton models are programming language independent resources that can also be used for language processing applications.

Finite state automatons are defined by the following elements:

- The set of finite states:

$Q = \{q_I, q_1 q_2, \dots, q_n, q_F\}$

- A transition function which has to parameterizes a state and a symbol and which returns a state:

$\delta : Q \times \sum \rightarrow Q$

- One or Several final states $q_F \subseteq$ Q

Several authors have used finite state automata to do Arabic morphological analysis such as: Jaccarini [10], Beesly [1, 2].

In the next section, we are going to describe the format of the input data and illustrating how morphological analysis works. And also present the results of the newly reimplemented tools and then discussing some possible future modifications.

3. THE AUTOMATON-BASED MORPHOLOGICAL ANALYZER

To decrease the size of the dictionary of radicals (stems) in Buckwalter analyzer [3], to reduce the size of the dictionary of stems in Buckwalter analyzer[3].We use automaton to group together all the prefixes, suffixes and infixes of the Arabic language. Then the search for the morphological analysis of a word is done in this global network which is based on automaton [15].

In this case, each root will be presented by a single stem. For example the root ‘كتب’ presented by ’كتب‘.

To implement our idea, we are based on automata in which each word is modeled by an automaton, whose radical lettersare presented by a state which loops on itself, and the affixes are presented by the characters forming these affixes for an automaton is represented by a graph.

Example: in this presentation of Figure-1, a graph which illustrates a simple finite state automaton which models the decomposition of several words which consists of a prefix‘ف’, the infix ‘ا’ is suffix ‘ها’

The words 'فجامعها' ,'فداخلها' ..., are presented by the following automaton graph:

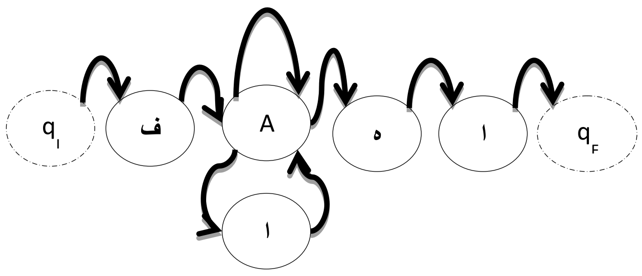


Figure-1. The graphical representation is illustrated of a simple finite state automaton which models the decomposition of several words.

An automaton of the word "فجامعها" consists of the following elements:

- q_IStart State.
- q_FFinal State.
- "ف"one character of the word prefix "فجامعها".
- "ا"word infix "فجامعها".
- "ا","ه" two characters of the word suffix "فجامعها".
- A = "ج", "م", "ع"represent the letters that make up the root of the wordجمع.In this case, we loop three times on the state A.

Based on all prefixes, infixes and suffixes of the Arabic language, we build a global network of states with only one input state = $q_I$andonly one output state. = q_F.
Our global network is defined entirely by:

- The set of all the states. consists of all the characters composing the affixes (suffixes, prefixes and infixes), state A, the initial state $q_I$and the final state q_F :

Q={q_I, q_F, A,"ف","و","ي","ل",...,"ه","م","ت",...}

- The set of all possible transitions connecting the characters of suffixes, prefixes and infixes with states A, q_I et q_F

.

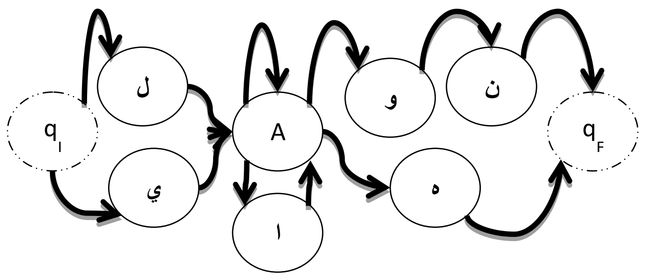


Figure-2. Example of route of the words "يدخلون" and "لضاربه" in the global network.

To analyze a given word w, we look in the global network for the different possible paths associated with w.
The set of these paths is given by:

$$S = \{ \xi \ \epsilon B \ / \ P_r(w / \xi) \neq 0\} \tag{1}$$

B : the set of all possible paths in our network.

ξ : A possible path of the same length as w,in the global network and which may present the word w.

The solutions are all the paths which allow the word w to be emitted with a non-zero probability. To facilitate and reduce the calculation in formula (1), we use the Viterbi algorithm. This algorithm is given by the following formula:

$$\delta_t(c_j) = \underset{c_i \rightarrow c_j}{\mathrm{NL}} \ (\delta_{t-1}(c_i) \cdot a_{ij} \cdot 1_{c_j}(w_t))$$

NL(x) is the function which gives the non-zero values of x.

We look for the states c$_i$ which give non-zero values of

$$\delta_{t-1}(c_i) \cdot a_{ij} \cdot 1_{c_j}(w_t).$$

$\delta_T(q_F)$: Is the maximum probability of sending the word w from a given path. By a recursive calculation we recover all the possible paths which give these non-zero values of NL (T the length of the word w).
With:

c_j : the j$^{\text{ème}}$ state ($c_j \in Q$).

a_{ij} : The probability of state transition c_i to the state c_j .

$$a_{ij} = \begin{cases} 1 \text{ if the transitions possible} \\ 0 \text{ else} \end{cases}$$

w_t : t$^{\text{ème}}$ character of the word w.

$$1_{c_j}(w_t) = \begin{cases} 1 & \text{if } c_j = w_t \\ 0 & \text{else} \end{cases}$$

we take:
$1_A(w_t) = 1$

Initialisation:

$$\delta_0(c_i) = \begin{cases} 1 & if \ c_i = q_I \\ 0 & else \end{cases}$$

Example:

Either the word w= "فنقول" the analysis is done according to the following steps:

$$\delta_0(q_i) = \begin{cases} 1 & if \ q_i = q_I \\ 0 & else \end{cases}$$

$$\delta_1(\text{ف}) = \max_{q_i \rightarrow \text{ف}} (\delta_0(q_i) \cdot a_{q_i \text{ف}} \cdot 1_{\text{ف}}(w_1)) = 1$$

Indeed: $\delta_0(q_I) = 1 \, \text{et} \, \delta_0(q_i) = 0 \quad q_i \neq q_I$

$a_{q_I \text{ف}} = 1$ all states are linked to the initial state.

$1_{\text{ف}}(w_1) = 1$ because w_1 ="ف"

$$\delta_1(A) = (\delta_0(q_I) \cdot a_{q_I A} \cdot 1_A(w_1)) = 1$$

We calculate in the same way $\delta_t(c_i)$ for t=2,….,5 and for all states.

At the end, we find the following paths in Table-1:

Table-1. Possible paths associated with the word "فنقول" in the global network.

suffix	prefix	proposed paths
0	0	q_I A A A A A q_F
0	ف	q_Iف A A A A q_F
0	فن	q_Iفن A A A q_F
0	فن	q_Iفن A و A q_F
0	ف	q_Iف A A و A q_F

4. TESTS END RÉSULTATS

a. Tests

To evaluate our new approach, we first created the different dictionaries of suffixes, prefixes and radicals. Then from the list of prefixes, suffixes and infixes. We generated a global network of states, as it was pointed out previously without using lexical dictionaries (Figure-3). This is the great advantage of our analyzer over the Buckwalter analyzer and our analyzer is based on a finite state automaton.

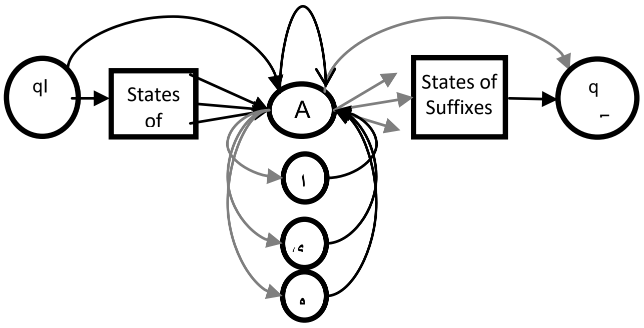


Figure-3. Diagram of our global network.

For the construction of the dictionary of radicals, we make the difference between two types of roots: healthy roots and defective roots (معتلة),the presentation of healthy roots in our dictionary is done by the roots themselves, but for defective verbs we keep the Buckwalter radicals [3].

For example the root "قال" is presented by 5 radicals in our dictionary: قال ، قل ،قول،قيل، قائل.But for the healthy root "كتب" they are presented only by "كتب".

With this presentation, we reduced the size of the dictionary of radicals by 62.5% compared to that of Buckwalter [3].

Likewise, we built an interdiction table, between prefix-suffix and this to eliminate impossible solutions (Table-2).

Table-2. Extract from the table of interdiction pref-suff.

Prefixes	suffixes
في	تهم
ي	تما
ي	نا
ن	تم

b. Implementation of the approach:

To implement our approach, we have developed a program in Java made up of three main classes:

- **Const Network class:** it is used to build the global network from the lists of prefixes, suffixes and infixes.
- **Analy Morpholo class:** it uses the Viterbi algorithm to find all possible paths which associated with a given word, in the global network which has already been built by the previous class.
- **Verf Compatib class:** it checks the validity of the solutions proposed by the class *Analy Morpholo*, by checking the existence of the root in the dictionary of radicals, and the compatibility between suffixes and prefixes.

Example:

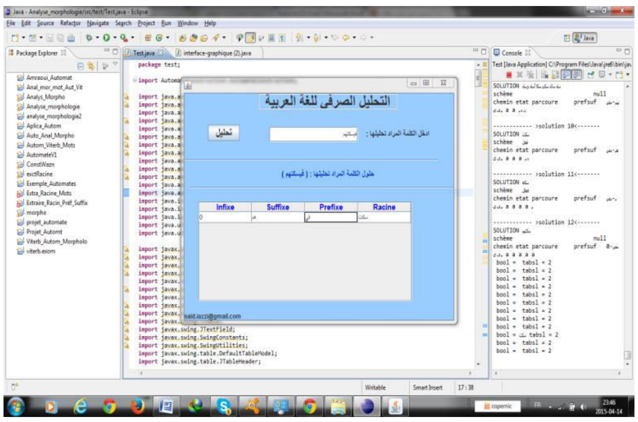
For the analysis of the word "فيسكتهم", our system follows the following steps (Figure-4):

- The search for different paths in our global network in Table-3:

Table-3. The possible paths of the word "فيسكتهم" in the global network with their proposed roots.

Suffixes	Prefixes	Possible paths	Proposed roots	N
تهم	0	AAAAتهم	فيسك	1
هم	0	AAAAAهم	فيسكت	2
0	0	AAAAAAA	فيسكتهم	3
تهم	ف	فAAAتهم	يسك	4
هم	ف	فAAAAهم	يسكت	5
0	ف	فAAAAAA	يسكتهم	6
تهم	في	فيAAتهم	سك	7
هم	في	فيAAAهم	سكت	8
0	في	فيAAAAA	سكتهم	9

- The extraction of the roots associated with these paths, by looking for the positions of states "A" in the word "فيسكتهم".
- Verification of the existence of roots in the dictionary of radicals, this step only keeps the following solutions:
- سك في - تهم
- سكت في - هم
- Verification of the compatibility between the suffixes and the prefixes of the remaining solutions, it only keeps
- سكت في - هم


Java - Analyse_morphologie/src/test/Test.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Test.java
interface-graphique (2).java
Console
التحليل الصرفى للغة العربية
تحليل
ادخل الكلمة المراد تحليلها :
حلول الكلمة المراد تحليلها : (فيسكتهم)
Infixe
Suffixe
Prefixe
Racine
said.iazzi@gmail.com
Writable
Smart Insert
17 : 38

```
Amraoui_Automat
Anal_mor_mot_Aut_Vit
Analys_Morpho
Analyse_morphologie
analyse_morphologie2
Aplica_Autom
Auto_Anal_Morpho
Autom_Viterb_Mots
AutomateV1
ConstWazn
exctRacine
Exemple_Automates
Extra_Racine_Mots
Extraire_Racin_Pref_Suffix
morpho
projet_automate
Projet_Automt
Viterb_Autom_Morpholo
viterb.exom
```

```
package test;

import javax.swing.JTextField;
import javax.swing.SwingConstants;
import javax.swing.SwingUtilities;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.JTableHeader;
```

Infixe	Suffixe	Prefixe	Racine
0	هم	في	سكت

```
------------ >solution 10<-------
SOLUTION
schème
chemin etat parcoure      prefsuf

------------ >solution 11<-------
SOLUTION
schème
chemin etat parcoure      prefsuf

------------ >solution 12<-------
SOLUTION
schème                          null
chemin etat parcoure      prefsuf
 bool =  tabsl = 2
 bool =  tabsl = 2
 bool =  tabsl = 2
 bool =  tabsl = 2
 bool =  tabsl = 2
 bool =  tabsl = 2
 bool =  tabsl = 2
 bool =  tabsl = 2
 bool =  tabsl = 2
 bool = سكت tabsl = 2
 bool =  tabsl = 2
 bool =  tabsl = 2
```

Figure-4. Result of the analysis the word 'فيسكتهم'.

c. Results:

Morphological analysis using finite state automata and the Buckalter-based approach are simply a list of all possible solutions for automatic morphological analysis of Arabic words. The type of machine learning used is the viterbi algorithm (Figure-5).

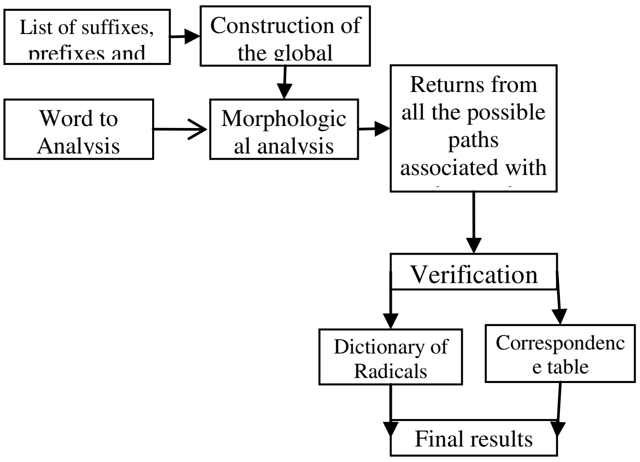

List of suffixes, prefixes and
Construction of the global
Word to Analysis
Morphological analysis
Returns from all the possible paths associated with
Verification
Dictionary of Radicals
Correspondence table
Final results

Figure-5. Scheme of the process of morphological analysis based on an automaton.

The test was carried out on 10,000 words which represent different grammatical categories (verbs, nouns, etc.), 96% of these words were correctly analyzed, and our analyzer returned the different possible analyzes of these words. While the rest of the words (4%), our analyzer failed in their analyzes, 95% of these errors are due to not taking into account the accounting relations between the prefixes_roots and the suffixes_roots in our analyzer.

5. CONCLUSION AND PERSPECTIVES

In this article, we have proposed a method of morphological analysis based on an automaton. This method combines the approach based on dictionaries and the approach based on finite state automata.

Our approach reduced the size of the dictionary of radicals. Likewise for the construction of the globalnet work. We do not need the dictionary of radicals; this network is built only from the list of affixes.

The results obtained using our approach, are very important and show the importance and usefulness of this approach. Following this work, we will integrate in this analyzer the two other correspondence tables: prefixes_roots and suffixes_roots.

We have helped to improve this situation by developing a set of resources, including lexiconsof verbs; nounsand rules for generated words, starting from the Arabic language. Which are all available. So that others can use and extend them. [8] [13].

REFERENCES

[1] Beesly K. R. 1998. Arabic Morphology Using Only Finate-State Operations, Proceedings of the Workshop on Computational Approaches to Semetic languages. Montreal, Quebec. pp. 50-57.

[2] Beesley K. R. 1996. Arabic Finite-State Morphological Analysis and Generation. Proceedings of the 16th conference on Computational linguistics, Vol 1. Copenhagen, Denmark: Association for Computational Linguistics. pp. 89-94.

[3] Buckwalter. T. 2002. Buckwalter Arabic Morphological Analyzer. Version 1.0. Linguistic Data Consrtium, catalog. Number LDC2002L49 and ISBN 1-58563-257-0.

[4] Darwish. K. 2002. Building a shallow Arabic morphologi-cal analyser in one day. In Proceedings of the ACLWorkshop on Computational Approaches to Semitic Lan-guages, Philadelphia, PA.

[5] El-Sadany T. A. and Hashish M. A. 1989. An Arabic Morphological System. IBM Systems Journal. 28(4): 600-612.

[6] Goldsmith and John. A. 2001. Unsupervised learning of the morphology of a natural language. Computational Linguistics. 27(2): 153-198.

[7] Hegazi N. and El Sharkawi A. 1986. Natural Arabic Language Processing, Proceedings of the 9th National Computer Conference and Exhibition, Riyadh, Saudi Arabia. 1-17.

[8] Hopcroft and J. Ullman. 1979. Introduction to Automata Theory, Languages, and Computation. Addison-Wesley. 55, 163.

[9] Iazzi S. All. 2021: The Use of the Relational Concept in the Arabic Morphological Analysis. (IJACSA) International Journal of Advanced Computer Science and Applications. 12(11).

[10] Jaccarini A. 1997. Grammaires modulaires de l'arabe. These de doctorat. Universite de Paris-Sorbonne.

[11] Khoja. S and Garside. R. 1999. Stemming Arabic text. Computer Science Departement, Lancaster University, Lancaster, UK.

[12] Koskenniemi and Kimmo. 1983. Two Level Morpology. A General Computational Model for Word-form Recognition and Production. Publication No. 11, Dep. of General Linguistics, University of Helsinki, Helsinki.

[13] Sipser Introduction to the Theory of Computation, Second Edition. Course Technology, February 2005. ISBN 0534950973. 55.

[14] Yousfi A. 2010. The morphological analysis of Arabic verbs by using the surface patterns. IJCSI International Journal of Computer Science Issues. 7(3, 11).

[15] Wehrli E. 1997. L'analyse syntaxique des langues naturelles : problèmes et méthodes, Paris, Masson.