Learning the Morphological Features of a Large Set of Words*

Abolfazl Fatholahzadeh

Supélec - Campus de Metz
2, rue Édouard Belin, 57078 Metz, France.

e-mail: Abolfazl.Fatholahzadeh@supelec.fr

Abstract. Given K - a large set of words - this paper presents a new method for learning the morphological features of K. The method, LMF, has two components : preprocessing and processing. The first component makes use of two separate methods, namely, refinement and time–space optimization. The former is a method that uses the closed world assumption of the default logic for partitioning K into a set of hierarchical languages. The latter is for efficiently learning the morphological features of each language outputted by the former method. Although, the finite-state transducers or the two-trie structure can be used to map a language onto a set of values, but we use our own competitor which has recently been proposed for such a mapping, consisting of associating a finite-state automaton accepting the input language with a decision tree (dt) representing the output values. The advantages of this approach are that it leads to more compact representations than transducers, and that decision trees can easily be synthesized by machine learning techniques.

In the processing phase, given an input string (x), thanks to the hierarchical languages establishing the preferency order for the utilization of the current automaton(g_i) among the multiple ones, if x can be spelled out using g_i, then the output is returned using its counterpart namely dt_i, otherwise, we inspect other alternative until an output or failure be done. LMF has learned good strategies for the large sets of the words which are consuming tasks form space and times point of views *e.g., all* the verbs in French, including all the conjugated forms of each verb.

Keywords: morphological features, automata, decision trees, learning.

1 Introduction

The morphological features (*i.e.,* mode, tense, person and gender) are supposed to be the important ingredients of the lexicons which are widely used in the process of determining for a word (*e.g.,* "livre") its output values (*e.g.,* Verb+IND-PRES-1-SING, Verb+IND-PRES-3-SING, Verb+IMP-PRES-3-SING, Noun+MASC-SING and Noun+FEM-SING).

*This work is partially supported by le Conseil Régional de Lorrain.

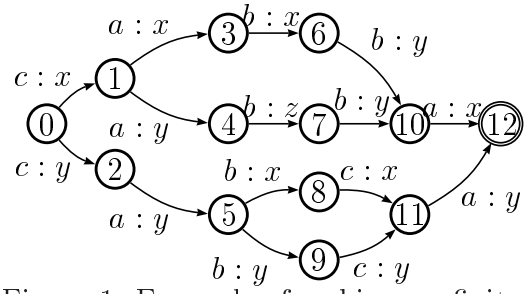

Figure 1: Example of ambiguous finite-state transducer shown by a (13,16) automaton [4, Page 158].

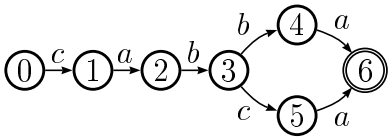

Figure 2: Our alternative - a (7,7) unlabeled automaton along with two decision rules. If b2 = 'b' Then v1 = [xxxxx,xxyyx,xtzyx]. If b2 = 'c' Then v2 = [yzxxy,yzyyy]. $\mathtt{b}_2$ stands for the second character from right to left of the input language.

An obvious solution to such a task is to store all the desired words along with their associated output values in a large-scale dictionary. But in this case two major problems have to be solved: *fast lookup* and *compact representation*. Two modern and efficient methods can achieve fast lookup by determination and compact representation by minimization. The first method is the technique of *two–tries* proposed by Aoe et al [1]. This method has the advantage of being applicable to a dynamic set of keys but unfortunately it has the disadvantage (Please refer to the page 488 of [1]) of containing more than states (hence the transitions) representing the data compared to its competitor, namely, the automata [13].

The second method is the *transducers* (*i.e.,* automata with outputs) [6, 8, 9] which have proved to be a very formal and robust execution framework for linguistic phenomena, but there are still some aspects that should be investigated. In particular, as shown in Figures 1, the transducers assign the unnecessary labels to some arcs of the graph representing the automaton. That is why, in our recent work, we have proposed a method to avoid such unnecessary labels (hence the states and the transitions) as pictured in Figure 2. Our solution for mapping a language onto a set of values is based on associating a finite-state automaton accepting the input language with a decision tree representing the output values. The advantages of this approach are that it leads to more compact representations than transducers, and that decision trees can easily be synthesized by machine learning techniques.

For the sake of clarity, we consider only the verbs in a given language and will show how our alternate approach can be combined with the closed world assumptions of the default reasoning. We show that the representation developed here provides a richer language for dealing with a set of strings where each of which is associated with one or more set of strings while keeping in the core of our system the two mentioned desiderata: compact representation and fast lookup. After presenting the default reasoning and its applicability to the morphology, we illustrate in Section 3 combining the automata and the decision tree. In Section 4 the refinement is described. The main algorithm of LMF along with examples in four languages closes: Azeri, English, French and Persian are described in Section 5. Finally, the concluding remarks close the paper.

2 Using Default Logic in Morphology

Default reasoning is a special but very important form of *non–monotonic* reasoning [5]. The term "default reasoning" is used to denote the process of arriving at conclusions

based upon patterns of inferences of the form "In the absence of any information to the contrary assume ..." (*e.g.,* if all elephants we have seen had a trunk, we might think that all elephants have a trunk). Of course, the possible circumstances in which any "presumed" correct line of reasoning can be defeated astound, and we are doomed to make mistakes when our experiences does not support the current situation. If we assume that the morphology world of the natural languages is closed one then there is a great chance that the rate of the classification noise be lower, even zero.

Example 1: *w.r.t.* the world of the verbs in French, even if there is no indications about the verb "zaper" in our system, LMF is able to learn 95 morphological features associated with the conjugated forms (*e.g.,* "zapons") of that verb.

Remark 1: The number 95 came from the fact that LMF is designed to learn the morphological features of all modes, namely indicative (IND), subjunctive (SUB), conditional (COND), imperative (IMP), infinitive(INF) and participate (PART). IND mode has 48 forms in eight tenses: present, imperfect, past, future, *etc.* Each of which allows to generate six forms according to: (1) gender (singular and plural); and (2) the person (1, 2, and 3). SUB mode has 24 forms in four tenses. COND mode has 24 forms in two tenses. IMP, INF modes has two and three forms, respectively. PART mode has usually three forms, two for some irregular verbs.

2.1 The Closed World Assumption

It seems not generally recognized that the reasoning components of many natural language understanding systems have default assumption built into them. The representation of knowledge upon which the reasoner computes does not explicitly indicate certain default assumptions. Rather, these default are realized as part of the code of the reasoner's process structure containing the hierarchies.

The starting point of the default reasoning is a set of *inference rules*(axioms) possibly along with some facts of the domain at hand collected in database which we call axiomal database (noted by G_{ax}). Given G_{ax}, the task based on the "specificity" and "inheritance" is to draw a plausible inference for the input. These can be illustrated by the classical Tweety example as follows: Consider the database containing four defaults: "penguins are birds", "penguins do not fly", "birds fly" and "birds have wings". "Specificity" tell us Tweety is a penguin, then Tweety doesn't fly because *penguin* is a more specific classification of Tweety than *bird*. "Inheritance" on the other hand, does equip Tweety with wings, by virtue of being a bird, albeit an exceptional bird *w.r.t.* flying ability.

From efficient implementation of the reasoner's process structure point of view, if the class "Specificity" lies "*above*" the generic class *i.e.,* there is some pointer leading from penguin's to node bird in G_{ax}, then given a particular penguin we can conclude that it doesn't fly. Notice that the reasoner's process structure of G_{ax} can be either a network - the graph of the taxonomy - or a set of first order formulae. The second option has been chosen to form G_{ax} of the morphology world in our work. In that option for fast inference purpose, G_{ax} is organized according to *priorities* which are given as ordering of predicates formulae, or default rules: in conflicting situations preference is given to item with high priority. That is to say, the data are added in G_{ax} in the following orders: (1) the facts of the exceptional data; (2) the facts

associated with generic axioms; (3) the exceptional axioms describing the specificity; and finally (4) the generic axioms.

Example 2: *w.r.t.* Tweety the orders of G_{ax} is as follows: (1) Penguin(tweety); (2) Bird(tweety); (3) $(\forall x) Penguin(x) \rightarrow \neg Flies(x)$; (4) $(\forall x) Bird(x) \rightarrow Flies(x)$.

(3) can be paraphrased as "penguins usually cannot fly". If a particular penguin (say Foo) can fly, this is obviously a counter exceptional data (or insensitivity to specificity) *w.r.t.* to (3). Although, how the representation of the insensitivity to specificity can be done in the open world (*i.e.,* the data related to the exceptions and in particular those of the counter exceptions are not known in advance), but this is not a limitation for our work because the databases of LMF is composed only using three predicates : regular, exceptional and counter-exceptional. The selection of the counter exceptional data is based on the fast inference purpose.

The LMF policy for such above purpose is to take into account both the high priority of usage in the text of a given language (*e.g.,* the auxiliary verbs of a given language such as "avoir" - to have - or "être" - to be -) and the seldom of data *w.r.t.* exceptional data (*e.g.,* "aller" -to go - the only member of the class 22 of the irregular verbs) or its specificity *w.r.t.* the general data (*e.g.,* "Haïr" meaning to hate, which is also a unique member of the 20th class of the regular verb).

3 Combing the Automata and the Decision Trees

In what follows, we summarize our recent work [3] concerning the combination of the automata and the decision trees. We assume the reader to be familiar with both the theory of finite automaton and the decision tree learning as presented in standard books *e.g.,* [13] and [7], respectively. We refer to a *key* and a *value* denoted by k and kv, respectively, as a sequence of characters surrounded by empty spaces which may have one or more internal spaces. We may use key and word (including verbs), interchangeably, as well as, the value, key–value and the morphological features.

The input of our algorithm for such above combination is the following customary form: $f = \{(k_i, v_i) | i = 1, \ldots, n\}$ for representation and fast lookup. The point of our idea is as follows: If an input string(x) can be recognized using the *unlabeled* finite-state-automaton (g) associated with the keys (of f) - hence having less states and transitions compared to the transducer as shown in Figures 1 and 2 - then use the learn decision tree (dt) for outputting the value associated with x. Table 1 shows a simple decision tree (dt) of $f1 = \{(Iran, Tehran), (Iraq, Baghdad), (Ireland, Dublin)\}$. Note that the `dt` *w.r.t.* $\mathtt{f_2} = \{$(Iran,Asia),(Iraq,Asia)$\}$ has a unique solution-path i.e. ($kvAsia$) - no condition (*i.e.,* question) is required to discriminate the key-value.

3.1 Acyclic Finite-state Automaton

Recall that an acyclic finite-state automaton is a graph of the form $\mathtt{g} = (Q, \Sigma, \delta, q_0, F)$ where `Q` is a finite set of states, Σ is the alphabet, $\mathtt{q_0}$ is the start state, $F \subseteq Q$ is the accepting states. δ is a partial mapping $\delta : Q \times \Sigma \longrightarrow Q$ denoting *transition*. If $a \in \Sigma$, the notation $\delta(q, a) = \perp$ is used to mean that $\delta(\mathtt{q}, \mathtt{a})$ is undefined. Let $\Sigma^\star$ denotes the set containing all strings over Σ including zero-length string, called the

Table 1: Backward attribute-based Data and Decision Tree.

b_7	b_6	b_5	b_4	b_3	b_2	b_1	KV	Solution-Path	Question	KV
$\star$	$\star$	$\star$	I	r	a	n	Tehran	(b_1 n kv Tehran)	b_1 = n?	Tehran
$\star$	$\star$	$\star$	I	r	a	q	Baghdad	(b_1 q kv Baghdad)	b_1 = q?	Baghdad
I	r	e	l	a	n	d	Dublin	(b_1 d kv Dublin)	b_1 = d?	Dublin

Table 2: Ten keys of the same lengths along with associated values.

Key	onC	myC	mnH	onH	nnH	nnC	mnC	nyC	myH	oyC
Value	down	down	up	down	up	up	up	up	down	down

empty string ε. The extension of the partial δ mapping with $x \in \Sigma^\star$ is a function $\delta^\star : Q \times \Sigma^\star \longrightarrow Q$ and defined as follows:

$$\delta^\star(\mathtt{q}, \varepsilon) = \mathtt{q}$$

$$\delta^\star(\mathtt{q}, \mathtt{ax}) = \begin{cases} \delta^\star(\delta(q,a),x) & \text{if } \delta(q,a) \neq \perp \\ \perp & \text{otherwise.} \end{cases}$$

A finite automaton is said to be (n,m)–automaton if $|\mathtt{Q}| = n$ and $|\mathtt{E}| = m$ where E denotes the set of the edges (transitions) of g. The property $\delta^\star$ allows fast retrieval for variable-length strings and quick unsuccessful search determination. The pessimistic time complexity of $\delta^\star$ is $\mathcal{O}(n)$ *w.r.t.* a string of length n.

3.2 Decision Tree Learning

Decision tree learning is a method for approximating discrete–valued target functions, in which the learned function is represented by a decision tree (dt). Learned decision trees can also be re-represented as a set of if–then rules to improve human readability.

Example 3: Below we list the if–then rules representing the decision tree associated with data of Table 2.

If f_1 = 'o' **Then** KV = 'down';
If f_1 = '$m' \wedge f_2 =' y'$ **Then** KV = 'down';
If f_1 = '$m' \wedge f_2 =' n'$ **Then** KV = 'up';
If f_1 = 'n' **Then** KV = 'up';

where f_1 and f_2 denote first character and sceond character (of the key from left to right), respectively. Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instances. Each node in the tree specifies a test of some *attribute* (*e.g.,* $b1$ of Table 1) instance, and each branch descending from that node corresponds to one of the possible values for this attribute. An instance is classified by starting at the root of the tree, testing the attribute value by this node, then moving down the tree branch corresponding to the value of the attribute in the given example. This process is then repeated for the subtree rooted at the new node. Notice that the implementation of the decision tree is based on `m-array` tree rather than the binary one. The former allows to save the decision tree in a less space compared to the latter. Figure 4 shows such a learned tree representing the values of the keys of Table 2.

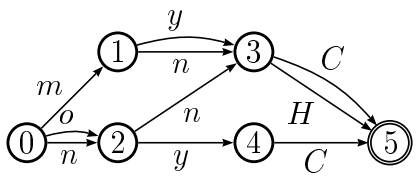

Figure 3: A (6,10) unlabeled automaton for recognizing the keys of Table 2.

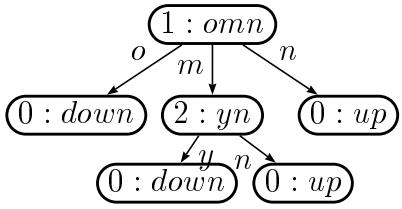

Figure 4: Learned decision tree for determining the value of any recognized key of Table 2.

Table 3: Distribution of French regular verbs according to the class and the frequency noted by C and F, respectively.

C	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
F	3875	156	165	342	69	114	19	12	9	254	26	49	2	302	1

4 Refinement

The refinement process has the following tasks to perform:

1. Transform the input of LMF, namely our input, namely $f = \{(k_i, v_i)|i = 1, \ldots, n\}$ into axiomal database D_{ax}, as described in Section 2.1.

2. Partition D_{ax} into the counter-exceptional, exceptional and general axioms.

The transformation is based on the closed world assumption of the morphology assuming that the set of the words of (f) noted by K can be divided into two subsets of so-called regular and irregular words. The regular forms follows the fact that their derivate/inflectional forms (each noted by d_k) can be generated using those axioms specified by the linguists which are usually further refined in a set of finer regular axioms (*axiom*). Using a root (of the word) each axiom allows to generate all d_ks of the word. The root is obtained by removing a particular substring of used *axiom*.

Example 4: The regular forms of the verbs in French is divided into the first group containing 13 classes (ranged from 6 to 18) and the second group which is composed of two classes (ranged from 19 to 20), where each number stands for an *axiom*. Below the repartition of 5189 infinitives (of the regular verbs) used in our experiment is shown in Table 3.

Remark 2: As appear from Table 3, 20th class has only one member, namely "Haïr". However, as we mentioned earlier, it is not considered is a a regular data. Indeed, *w.r.t.* to the inference process, it is wise to consider it as a counter-exceptional data. The reason is to speed up the inference processing by mentioning explicitly the data and axioms is the following order: counter-exceptional, exceptional and general. This process constitutes the well known practical trick of the default logic. So, 5188 (*i.e.,* 5189 -1) roots along with 19 classes will be used as the reservoir for learning the extended database of 492860 (*i.e.,* 5188×95) d_ks of the lexicographers expressed in a raw database.

An *axiom* can be described using a two dimensional vector of size r, where r stands for the number of morphological features in use. The first row of such a vector

Table 4: Information on size of 13943 verbs of the third group in French and morphological information along with the forest of the decision trees obtained by the partitive learning mode. Ent. refers to number of call to the entropy function.

Data		**Decision Tree**			**Gain**	
Len.	Freq.	Inodes	Leaves	Ent.	K%	V%
2	11	9	3	15	66%	19%
4	183	133	40	371	81%	23%
5	412	225	66	904	88%	44%
6	943	460	131	2149	91%	47%
7	1480	578	202	3388	93%	57%
8	2160	727	240	5065	94%	62%
9	2317	692	342	6664	95%	67%
10	2115	582	252	6531	96%	70%
11	1729	445	207	6361	96%	72%
12	1168	318	125	4980	97%	70%
13	733	164	69	3472	97%	75%
14	389	106	50	2620	97%	70%
15	183	59	22	1624	97%	68%
16	72	36	18	1063	95%	50%
17	25	9	4	288	97%	64%
18	7	3	2	83	96%	58%

is composed of r the values. The second row contain different substrings related to d_ks. Usually, the lexicographers are used to add the word in explicit database in which each entry is composed one d_k and a value. Since it may happen that for a d_k different values be associated with it (*e.g.,* aime IND-PRES-1-SING, IMP-PRES-3-SING, *etc.*) therefore, the learning process should assure to collect them into a set of morphological features representing a set of unique ambiguity class. In summary, the entire lexicon can viewed as follows. First on can form the the four following reservoir f_g, s_g, f_e and f_c representing: (1) f_g: Database related to the general axioms; (2) s_g: Database of suffixes of the regular (general) words; (3) f_e: Database of derivate forms expressed as the exceptional data; (4) f_c: Database of derivate forms based on the high priority relating the counter exceptional data. Notice that f_g along with s_g will be used to recognize the derivate forms of the words governed by the general axioms.

4.1 More Refinement: Learning by Partitive Mode

As we mentioned earlier, the input of decision tree learning is a fixed attributes the size of this table is $\ell + 1 \times n$, where ℓ denotes the length of the longest keys of f and n is the number of *keys*. Usually, we have to use the dummy characters (noted by $\star$ see Table 1). Using the dummy characters augment the size of the input table. Because of the very recursive nature of the learning process, including the characterization of the decision tree may be a time consuming task for the large data. An alternative to the a unique table is to employ multiple tables as follows. First f is divided into q

user-inputs (f_i) such that the length of the keys of each f_i be identical, then form the corresponding decision trees. So, in the partitive mode, we have to learn a *forest of the decision tress* : composed a vector of r positive integers. ith number is pointed to the ith decision tree.

Searching a value for an input string (x of length y) works as follows. If y belongs to the vector of above mentioned numbers, first we spell out x this time using the automaton associated with entire keys of K. If x spelled out correctly, then we use the y^{th} decision tree to output the value.

Example 5: The value of $x = abababad$ can not be learned *w.r.t.* current $f = \{(abc, 1), (ababbac, 2)(abababc, 3)\}$. We have length(x) = 8 which is not member of {3,5,7}. In the contrary, for $x = abc$ the value is 1 *i.e.,* (1) $length(x) \in \{3, 5, 7\}$, (2) x is recognized using the automaton associated with $K = \{abc, ababc, abababc\}$ and (3) no question is required for f_3 the value is 1. Table 4 shows the Information on size of 13943 verbs of the third group in French and morphological information along with the forest of the decision trees obtained by the partitive learning mode.

5 Main Algorithm

Below the algorithm for learning morphological features is given which is composed of two components: preprocessing and processing. In the first component four automata and two decision trees along with a forest decision trees containing r decision trees are formed, where r stands for the number of partitions of the exceptional data according to the same key-length criterion. In the second component, if an user-input (x) can be recognized by one of the four automata (see below for the order in use) then the corresponding decision tree will be inspected to output the value. The argument of main function are:

1. $f_g = \{(root_i, axiom_i)|i = 1 \ldots, n_1\}$ *i.e.,* Database related to the general axioms;
2. $s_g = \{(suf_i, mf_i|i = 1 \ldots, m_1\}$ *i.e.,* Database of suffixes of the regular (general) words; mf stands for a morphological features or a set of alternate morphological features;
3. $f_e = \{(d_i, mf_i)|i = 1 \ldots, n_2\}$ *i.e.,* Database of derivate forms expressed as the exceptional data; d_i refers to a derivate form of a base word (*e.g.,* infinitive);
4. $f_c = \{(d_i, mf_i)|i = 1 \ldots n_3\}$ *i.e.,* Database of derivate forms based on the high priority relating the counter exceptional data.

---

func LearningMorphologicalFeatures(f_g, s_g, f_e, f_c)
$K_g \leftarrow CollectKeys(f_g)$. $K_c \leftarrow CollectKeys(f_c)$.
$g_{kg} \leftarrow FormAutomaton(K_g)$; $g_{kc} \leftarrow FormAutomaton(K_c)$.
ApplyPreprocessingPartitiveMode(f_e).
$g_{Ke} \leftarrow FormAutomaton(K_e)$.
$table_c \leftarrow FormInputForLearning(f_c)$.
$t_c \leftarrow LearnDecisionTree(table_c)$.
$t_s \leftarrow LearnDecesionTreeOfSuffixes(s_g)$.

ApplySearch(x).{Processing component, x is an input string.}
cnuf

The function $FormAutomaton()$ follows the elegant algorithms described in [2] for the incremental construction of minimal acyclic finite state automata and transducers from both sorted and unsorted data We adapted the former one such that the length of the longest key be calculated for being used later in the construction of suitable input for learning the `dt` of the counter exceptional data. Please refers to [3] for the description of the function $FormInputForLearning()$ and $LearnDecisionTree()$. The construction of the forest of the decision trees works as follows.

---

func ApplyPreprocessingPartitionMode(f_e)
  $\bigcup_{i=\ell_1}^{\ell_x} f_{ei} \leftarrow Partition(f_e)$
  for $i \in (\ell_1, \ldots \ell_x)$ **do**
    $K_{ei} \leftarrow CollectKeys(f_{ei})$; $g_{kei} \leftarrow FormAtuomaton(f_{ei})$.
    $Table_{ei} \leftarrow FormInputForLearning(f_{ei})$
    $t_{ei} \leftarrow LearnDecisionTree(Table_{ei})$.
  end for
cnuf

Since the search order is based on looking at the following order : (1) counter exceptional, (2) exceptional and general data, then processing component is as follows:

---

func ApplySearch(x)
  return(SearchValue(x, g_{kc}, t_c) OR SearchValueUsingPartitionMode(x, g_{ke}, $forest$) OR SearchByMismatch(x, g_{kg}, s_g, t_s)).
cnuf

For knowing how SearchValue() works, again consider Figure 4 where zero used in a node indicates that node is a leaf one. A positive integer number used in a node has its own meaning indicating the test to be done taking into account the content of the current node under inspection *e.g.,* "1:omn" means that if the first character of x is 'm' then gets the value by descending in the sub-tree of first child. Since the sub-tree has only one node - a leaf - then value is 'down'. If the first character of x is 'm' this time the value has to be selected using the sub-tree of the second child. Depending on the second character ("2:yn") of x the output value is either "down" or "up".

---

func SearchValue(x, g, dt)
  if $\delta^\star(q_0, x) = q$ such that $q \in F(ofg)$ **then**
    $kv \leftarrow$ GetValue(x,dt).
  else
    $kv \leftarrow nil$; {x is unknown w.r.t. the current g}
  end if
cnuf

The function SearchByMismatch() uses the automaton associated with the general data to know if the root of (the base) word can be recognized by that automaton. If the input string can be spelled out using a given position then there is a chance that the suffix of the input string be recognized using the automaton of the available suffixes (s_g), if so, then GetValue will be activated to output the output value.

---

func SearchByMismatch(x, g_{kg}, t_s)

$pos \leftarrow MisMatchPosition(x, g_{kg})$; $s \leftarrow substr(x, pos)$. {s stands for the suffix}

return(GetValue(s, t_s)).

cnuf

5.1 Examples

Below we illustrate the traces of LMF applied to the verbs in English and French, Azeri and Persian.

Example 6 (French): Let us consider the following phrase: "Il livre un livre." *i.e.,* He is providing a book. Suppose that we are interested in learning the morphological features of the word "livre". The current word cannot be spelled out neither using the automaton associated with the counter exceptional automaton nor with the exceptional automaton. Therefore, the automaton associated with f_g (database of regular roots in French corresponding to the first group) will be called to partially spell out the word "livre". Using function `SearchByMismatch` tell us to stop at the fourth character (from left to right). The remaining part of the current word - "e" - will then be used as the entry of the decision tree associated with the suffixes of f_g outputting the desired result: Verb+IND-PRES-1-SING, Verb+IND-PRES-3-SING, Verb+IMP-PRES-3-SING, Noun+MASC-SING and Noun+FEM-SING.

Remark 3: The reason for which it is preferable to divide the set of words (of a language) into several files, each of which containing the same syntactic category could better be illustrated using our previous example. Indeed, one could use the rules of local grammar *e.g.,* (1) pronoun+verb as in "il livre" and (2) determinant+noun, as in "un livre", for the efficient tagging purpose while learning the morphological and right features of used word in a text.

Example 7 (French): In the the following phrase: "Bush hait Saddam et vice-versa. *i.e.,* Bush hates Saddam and vice-versa." Learning the morphological features of the word "hait" is immediate because this word belongs to the exceptional data containing the verbs of 20th class.

Example 8 (English): The morphological features of the word "stood" in the following phrase: `"He stood the child"`, can also be learned immediately, because it belongs to the exceptional data *w.r.t.* the verbs in English.

Example 9 (Azeri): Like in Turkish, the order of constituents may change rather freely without affecting the grammaticality of a sentence. Due to various syntactic and pragmatic constraints, different orderings are not just stylistic variants of the canonical order. For instance, a constituent that is to be emphasized is generally placed immediately before the verb. This affects the places of all the constituents in

a sentence except that of the verb:

Man	**oşhaxlara**	**ketabi**	**verdim.**	I gave the book to
I	children+DAT	book+ACC	give+P1S	the children.
Oşhaxlara	**man**	**ketabi**	**verdim.**	It was me who gave
children+DAT	I	book+ACC	give+P1S	the children the book.
Man	**ketabi**	**oşhaxlara**	**verdim.**	It was the children to
I	book+ACC	children+DAT	give+P1S	them I gave the book.

The first above sentence is an example of the canonical word order whereas in the second one the subject, `man`, is emphasized. Similarly, in the last one the direct object, `oşhaxlara`, is emphasized.

Remark 4: Although, Azeri has some similarity with old Turkish, but their structures differ in several aspects, notably *w.r.t.* new Turkish. This is particularly true for the the vocabularies and the morphology. All together, this makes the processing of Azeri different from Turkish, including our learning process.

Example 10 (Persian): If we concern ourselves with the unmarked order of constituents, like in Turkish and Azeri, Persian can be characterized as a subject-object-verb language: (a) "Man be baçeha ketab ra dadam." (*i.e.,* I gave the book to the children.) and (b) "Lazat bordand." (*i.e.,* (They) enjoyed). In (a) the morphological features of the verb "dadam" is determined by what we call the counter exceptional data whereas in (b) the segment "Lazat (adjective) bordan (verb)" have to be considered as a compound verb. So, the combination of the morphological features of two words would determine the morphological feature of the mentioned segment.

6 Concluding Remarks

LMF is written in C and applied for learning of the large set of the verbs in French and very limited ones in Persian and Azeri. The experiments show that combing the closed world assumption, the automata and the decision trees is a good approach since our tests provide the right results for more than half million verbs - including the conjugated form - in French. Note that the transducers [8], as the the best available method, have been used in the morphology world. However, the advantages of combing the automata with the decision trees are that it leads to compact representations than transducers, and the decision trees can easily synthesize by machine learning techniques. This is emphasized in this work by Figure 2.

It must be stressed that using automata is appropriate when there is `no need` for frequent updates of one or more databases. This is due to the fact that it is difficult to update quickly the automaton. However, *w.r.t.* our present work, this is not necessarily a limitation because we are dealing with static keys originated from the morphology world. From update viewpoint, using the two-trie structure of Aoe et al. [1] instead of the automata is preferred where there is the need for frequent updates. But in this case, the cost of space (number of states and transitions) is (slightly) expensive compared to the automaton.

An interesting extension is the question of addressing how to learn the regular and irregular data from pure Stringology viewpoint *i.e.,* without attaching a domain to the values of the keys. That is to say, we have to discover the axioms along with possible exceptional and/or counter exceptional ones.

Acknowledgments

I thank the anonymous referees for their constructive comments.

References

[1] Aoe, J–I., Morimoto, K., Shishibori, M., and Park, K. A trie compaction algorithm for a large set of keys. *IEEE Transaction on Knowledge and Data Engineering 8*, 3 (1996), 476–491.

[2] Daciuk, J., Mihov, S., Watson, B. W., and Watson, R. E. Incremental construction of finite-state automata. *Association for Computational Linguistics 26*, 1 (2000), 3–16.

[3] Fatholahzadeh, A. Implementation of dictionaries via automata and decision trees. Champarnaud J. M. and Maurel D. (eds.): Seventh International Conference on Implementation of Automata (CIAA02). In LCNS *Lecture notes on Computer Science*, vol. 2608. Springer, Berlin Heidelberg, (2003), 95–105.

[4] Kempe, A. Factorizations of ambiguous finite-state transducers. In *International Conference on Implementation and Application of Automata* (2000), Daley M., Eramian M., and Yu S. pre-proceeding (eds.), 157–164.

[5] McCarthy J., and Hayes, P.J. Some Philosophic problems from the standpoint of Artificial Intelligence. In *Machine Intelligence* (1969), vol. 4, Meltzer B. and Michie D. (eds), Edinburgh University Press, 463–502.

[6] Mihov, S., and Maurel, D. Direct construction of minimal acyclic sub-sequential transducers. In *International Conference on Implementation and Application of Automata* (2000), Daley M., Eramian E., and S.Yu pre-proceeding (eds.), 150–156.

[7] Mitchell, T. M. *Machine Learning.* Mc Graw-Hill, 1997.

[8] Mohri, M. On some application of finite-state automata theory to natural language. *Natural Language Engineering 2*, 1 (1996), 1–20.

[9] Mohri, M. Finite-state transducers in language and speech processing. *Computational Linguistics 23*, 2 (1997), 269–311.

[10] Mohri, M. Generic $\epsilon-$removal algorithm for weighted automata. In *International Conference on Implementation and Application of Automata* (2000), Daley M., Eramian E., and Yu S. pre-proceeding (eds.) 26–35.

[11] Quinlan, R. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[12] Reiter R. On reasoning by default. In *Reading in Knowledge Representation* (1985), Brachmann R.J. and Levesque H.J. (eds), Morghan Kaufmann, 402–410.

[13] Rozenberg G. and Salomaa A. (eds.) *Handbook of Formal Language.* Springer–Verlag, Berlin Heidelberg, 1997.