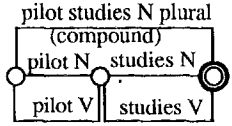
TEXT DISAMBIGUATION BY FINITE STATE AUTOMATA, AN ALGORITHM AND EXPERIMENTS ON CORPORA

Emmanuel Roche*
Insitut Gaspard Monge CERIL-LADL**
Paris France

1. Abstract*

Consulting a dictionary for the words of a given text provides multiple solutions, that is, ambiguities; thus, the sequence of words *pilot studies* could lead for example to:
pilot: *N singular, V infinitive, V (conjugated)*
studies: *N plural , V (conjugated)*
pilot studies: *N plural (compound).*

These informations could be organized in the form of a finite automaton such as:



The exploration of the context should provide clues that eliminate the non-relevant solutions. For this purpose we use local grammar constraints represented by finite automata. We have designed and implemented an algorithm which performs this task by using a large variety of linguistic constraints. Both the texts and the rules (or constraints) are represented in the same formalism, that is finite automata. Performing subtraction operations between text automata and constraint automata reduce the ambiguities. Experiments were performed on French texts with large scale dictionaries (one dictionary of 600.000 simple inflected forms and one dictionary of 150.000 inflected compounds). Syntactic patterns represented by automata, including shapes of compound nouns such as Noun followed by an Adjective (in gender-number agreement) (Cf 5.1), can be matched in texts.
This process is thus an extension of the classic matching procedures because of the on-line dictionary consultation and because of the grammar constraints. It provides a simple and efficient indexing tool.

2. Motivation

Automatic analysis by phrase-structure grammar is time consuming. The need for fast procedures leads to grammar representations that are less powerful but easier to handle than general unification procedures. Pereira and Wright 1991 and Rimon and Herz 1991 proposed such approaches, that is, algorithms that perform the construction of a finite-state automaton approximation of a phrase-structure grammar. These automata are then used as simple checkers of well-formed patterns. However, parsing a sentence and only providing the information that it does (or doesn't) match the automaton description is not sufficient. One should provide (see K. Koskenniemi 1990) the readings of the text that respect exactly the constraints. We propose here an algorithm that provide all these readings. Moreover, the automaton of a given text can be highly ambiguous, and in order to increase its adequacy (e.g. to study given syntactic patterns), we may want to customize it. To achieve such a result, we construct automata that eliminate paths irrelevant to the given study[1]. Once this operation was performed, significant patterns (like *Noun Adjective* in French) can be extracted. Technical terms in many domains take the form of sequences such as *Noun Adjective, Noun de Noun* etc. Their recognition thus leads to an efficient indexation process. This is a complementary approach to statistical treatments like those presented in K.Church, W. Gale, P. Hanks, D. Hindle 1989 or in N. Calzolari and R. Binzi 1991.
Moreover, we use Finite-State Automata (FSA) at all stages of the process: for dictionary consultation, for disambiguation and for the final extraction process. This allowed the experiments to be done on-line starting with untagged corpora.
One of the crucial points is that tagged text should be represented by FSA in order to be disambiguated (disambiguated texts are already in this form in Rimon and Herz 1991 and K. Koskenniemi 1990). FSA representation for ambiguities representation is not a new approach but in our contribution, we

* This work was supported by DRET and Ecole Polytechnique.

** Université Marne la Vallée. Institut Gaspard Monge. 2 Allée Jean Renoir. 93160 Noisy le Grand. France eroche@ladl.jussieu.fr Université Paris 7

[1] Some of these paths may correspond to legitimate solutions.

systematized it for different types of ambiguities, namely:

1. Morphological features ambiguity (gender for instance),
2. Part of speech ambiguity,
3. Phrase ambiguity (compound v.s. sequence of simple words).

3. Presentation of an example

Let us take, for instance, the French sequence (1).. *le passe...*
Both words are ambiguous, *le* can either be an article (*the*) or a pronoun (*it, him* or *her*) and *passe* can either be a noun or a verb. Moreover the noun *passe* is still ambiguous, since it can mean either *a pass key* (and is then masculine) or *a pass* (like in *a forward pass,* it is then feminine). The verb form *passe,* in turn, is ambiguous, it is a conjugated form of the canonical form *passer* (*to pass*) in one of the three tenses: indicative present, subjunctive present or imperative present. For the first two tenses, it can either be in the first or in the third singular person and, for the latter, it has to be in the second person of singular.
The problem is the following: the consultation of the simple form dictionary DELAF[2] (600.000 entries) first provides a sequence tagged as follows:

le (pronoun, article) passe(noun-ms, noun-fs, verb-P3s:S3s:P1s:S1s:Y2s)
(where the abbreviations are m: masculine, s: singular, 3: third person, P: present, S: subjunctive, Y: imperative)

The compound form dictionary DELACF[3] (150.000 entries) is used, it marks sequences like *pomme de terre* (*potato)* as frozen. In a second step, we provide the automaton representation of figure 1, to be read in the following way: The first word is either a pronoun or an article, its spelling is *le*, the second word is either a singular noun (ambiguous: one meaning is masculine (*the pass key)* and the other feminine (*the forward pass*)) or else a verb conjugated at the persons, tenses and numbers specified above.

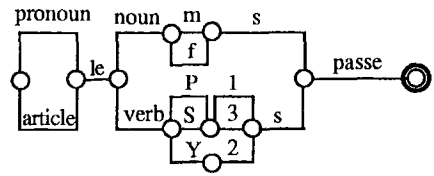


Figure 1

On the other hand grammar rules provide constraints which can be described as forbidden sequences. In our example, since the clitic sequence is highly constrained (M. Gross 1968), the pronoun *le* can be followed either by another pronoun or by a verb. The article *le* cannot be followed by a verb or by a feminine noun (except for parts of compounds). This set of forbidden sequences is described by the automaton of figure 2.

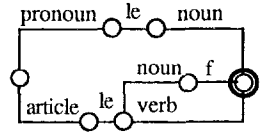


Figure 2.

Thus the FSA representing the text according to the rules should be the FSA of figure 3.

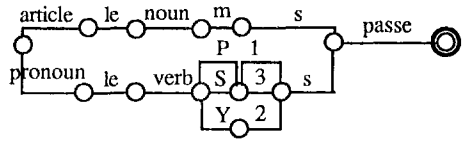


Figure 3.

The problem consists in constructing the automaton of figure 3 given those of figures 1 The reader probably noticed that the rules were described as a set of forbidden sequences, which is unusual. The formal operation and the algorithm are easier to describe with negatively defined rules, it is the reason why we use this device here. However, given the grammar corresponding to the automaton representation, the procedure is equivalent to a set of rules expressed in a positive, and hence more usual way.

4. The algorithm

4.1 Formal description of the problem.

The problem, informally described, can easily be specified in the following way:

[2]DELAF: LADL's inflected forms dictionary for simple words: B. Courtois 1984,1989.

[3]DELACF: LADL's inflected forms dictionary for compound words: M. Silberztein 1989.

Given a text, its FSA representation (e.g. figure 1) A_1 is defined by the 5-tuple $(Alph,Q_1,i_1,F_1,d_1)$ which respectively denotes its alphabet, its state set, its starting state, its final state set and its transition function[4] which maps (Q_1*Alph) into Q_1. Moreover, A_1 has the property of being acyclic (it is a Directed Acyclic Graph (DAG)). The constraints are represented by the FSA A_2, defined in the same way by $(Alph,Q_2,i_2,f_2,d_2)$. These automata define respectively the regular languages $L_1=L(A_1)$ (i.e. the language accepted by A_1) and $L_2=L(A_2)$ (i.e. the language accepted by A_2) . Since L_2 describes the set of sequences (or factors) forbidden in any word of L_1, if A describes the text after the filtering, this means that L=L(A) follows the condition $L = L_1 \setminus Alph^* \; L2 \; Alph^*$
This operation on languages will be called **factor subtraction** and will be noted $L=L_1$ **f-** L_2. At this point, we can define the related operation on automata: if $L_1=L(A_1)$ and $L_2=L(A_2)$ we say that A is the factor subtraction of A_1 and A_2 and note it $A= A_1$ f- A_2 if $L=L_1$ f- A_2 and $L=L(A)$.

4.2 Informal description of the algorithm

We will first apply the algorithm on a small example. Suppose that A_1 is the automaton represented in figure 4, that A_2 is the automaton represented in figure 5 and that we want to compute A_1 f- A_2.

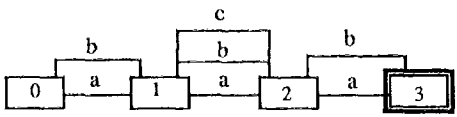


Figure 4

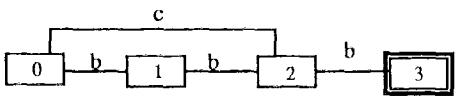


Figure 5
Each state of the automaton $A=A_1$ f- A_2 will be labelled with a state of A_1 and a set of states of A_2 (i.e a member of the power set of Q_2). More concretely the automaton $A=A_1$ f- A_2 of figure 6 is built in the following way:
The initial state is labelled $(0,\{0\})$, the first 0 refers to the state 0 of A_1 (0_1 for short). The letter *a* leads, from 0_1 to the state 1_1 of A_1 but to nothing in A_2, we construct the state $(1,\{0\})$ which means that, for *a*, 0 leads to 1 in A_1 but that $\{0\}$ leads to nothing (the empty set) in A_2 to which we systematically add the initial state. On the other hand, $d_2(\{0\},b)=\{1\}$ to which we add, as for *a*, the state 0; thus, in A, $d((0,\{0\}),b) = (d_1(0,b), \{0,d_2(0,b)\})= (1,\{0,1\})$. For each state being constructed, we list the states it could refer to in A_2 and, for each of these states, their image by the letter being considered. A specific configuration is when the state of A being considered has one of his label that leads to the final state of A_2, it means that a complete sequence of A_2 has been recognized and should then be deleted. This is the case if we look at state $(2,\{0,1,2\})$ in A: $d_2(\{0,1,2\},b)=\{1,2,3\}$ where 3 is final, thus it has no transition for *b*, which leads to delete the path *bbb* forbidden by A_2.

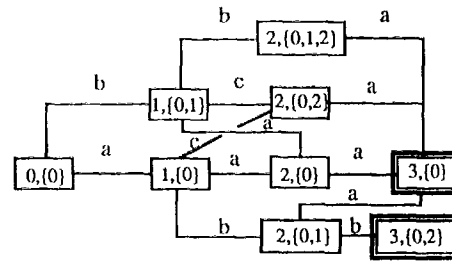


Figure 6
The following algorithm computes A1 f- A2

```
1.f[0]=(i1,{i2})
2.q=0;
3.n=1;
4.F=∅;
5.do
6.      (x1,X2)=f[q];
7.      G={i2};
8.      for each s ∈ Alph so that d1(x1,s)≠∅
9.          y1=d1(x1,s);
10.         for each x'∈ X2
11.             if d2(x',s)=f2
12.                 G=∅; goto 8;
13.             else
14.                 G=G U {d2(x',s)};
15.         endfor
16.         if ∃q'<=(n-1) so that f[q']=(y1,G)
17.             d(q,s)=q'
18.         else
19.             f[n]=(y1,G);d(q,s)=n; n+=1;
20.             if y1∈F1 then F=F U {n};
21.     endfor
22.     q+=1;
23.while (q<n)
```

[4]The automata are assumed to be deterministic, which is not an additional constraint since one can determinize them (see Aho, Hopcropft, Huffman 1974 for instance).

5. Experimental results

Given a syntactic pattern (first Noun followed by an Adjective) and a text, we can detect its occurrences. We can search the text without applying constraints, this provides output 1 (figures 7 and 8).We can also search it after having applied constraints (output 2). We shall compare both outputs. For instance, for the sentence:

L'individu n'y est pas perçu comme une valeur abstraite et universelle, mais comme un être concret, comme le membre d'un ensemble particulier, localisé et qui n'existe que dans son rapport à cet ensemble.

the program provides the following matchings:

(1)	(2)
y est	valeur abstraite
valeur abstraite	être concret
être concret	ensemble particulier
d un	
ensemble particulier	

Figure 7

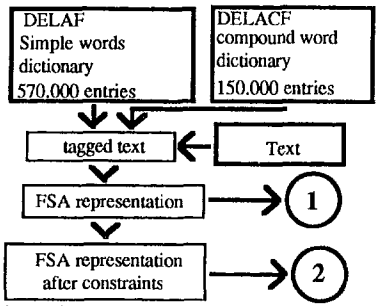


Figure 8

The program runs in three steps (figure 8). It first takes a text and tags it according to the two dictionaries. The text is then transformed into its FSA representation on which the constraints are applied.

Given a pattern (Noun followed by an Adjective), we compare its number of ocuurences in both outputs 1 and 2. This will give us a measure of the power of the filtering.

It is worthwhile to point out that the experiments were realized on untagged corpora, namely that the duration of the tagging process is included in the figures given in the tables. These experiments were done on personal computers[5] and it can be seen that the time spent is low enough to permit on-line use (for compound word enrichment for instance).

5.1 Searching Noun-Adjective patterns

First, let us consider the pattern Noun Adjective (which is approximately equivalent to the English sequence adjective-noun). We first tried to search each contiguous pair of words whose first element was labelled as a noun and whose second element was an adjective. This provides the result of the first line of figure 9. The first filtering uses the fact that, in French, the word and the adjective have to agree on their gender and on their number. This gives, for the same texts, the results of the second line. Third we applied the algorithm described above as a second filter, this leads to the results of the third line.

	Editorial 4185 bytes (1 page)	Article 13010b. (4p)	Novel 369292b (100p).	Corpus 1738115
N Adj	**66** 15"	**227** 40"	**3334** 19'	**31532** 1h25
N Adj agreement	**56** 15"	**198** 40"	**2651** 19'	**28234** 1h25
N Adj agreem.+ constr.	**13** 20"	**40** 50"	**1277** 41'	**11125** 3h05
N Adj real number[6]	**10**	**34**	**1150**	not counted

Figure 9[7]

The texts are in the form of ASCII files. The first one was a magazine editorial of about 1 page[8], the second one is an article of about 4 pages[8]. The third one is a novel of the French 19th century writer Jules Verne: *Les aventures du docteur Ox*. The fourth one is a compilation of texts with a large amount of law texts. We gave, in the last line, the number of patterns that should have been detected if the filtering had been perfect; this was done by hand.

6. Conclusion

[5] Experiments were done with an IBM PS2 386 25Mhz with an OS/2 V1.3 and 8Mb ram. The program is in C.

[6] This number is of course obtained by hand, which explains why we didn't do it on the fourth text.

[7] The simple word morphological dictionary of 570.000 factorized entries was compressed into 1Mb and the 150.000 compound forms DELACF was compressed into 2Mb.

[8] From Société Magazine 1989

These experiments will be expanded on a larger amount of patterns and on various types of corpora, but we already think that those we presented here show that the method can actually be used as a practical tool for easing the construction of terminological lists.

7. Bibliography

Aho, Alfred V. , John E. Hopcroft, Jeffrey D. Ullman, 1974. The Design and Analysis of Computer Algorithms. Addison Wesley, 467p.

Calzolari, Nicoletta, Remo Bindi, 1990. Acquisition of Lexical Information from a Large Textual Italian Corpus. Coling 90, Proceedings of the Conference. Helsinki.

Church, Kenneth, William Gale, Patrick Hanks, Donald Hindle, 1989. Parsing, Word Associations and Typical Predicate-Argument Relations. Internal report. Bell Laboratories, Murray Hill.

Courtois, Blandine, 1984, 1989. DELAS: Dictionnaire Electronique du LADL pour les mots simples du français, Paris: Rapport technique du LADL, Université Paris 7.

Gross, Maurice. 1968 Grammaire transformationnelle du français, 1-Syntaxe du verbe. Cantilène, Paris, 183p.

Koskenniemi, Kimmo, 1990. Finite-State Parsing and Disambiguation. Coling-90. Proceedings of the conference. Helsinki.

Peireira, Fernando C.N. , Rebecca N. Wright. 1991. Finite state approximation of phrase structure grammars, 29th Meeting of the A.C.L, Proceedings of the conference. University of California, Berkeley.

Revuz Dominique, 1991. Dictionnaires et lexiques, méthodes et algorithmes. PhD dissertation. Université Paris 7, Paris, 130p.

Rimon, Mori, Jacky Herz, 1991. The recognition capacity of local syntactic constraints. Fifth Conference of European Chapter of Association for Computational Linguistics. Proceedings of the Conference, Berlin.

Silberztein Max, 1989. Dictionnaires électroniques et reconnaissance lexicale automatique. PhD dissertation, Université Paris VII, Paris, 175p.

8. Annexe

Some constraints

We present here a sample of constraints as they are actually implemented. The following set of sequences represents paths that have to be deleted from the FSA representing the text. This set is to be compiled into a FSA before being used by the program. The ?? word means that it can be any transition of the state it is compared to. For instance the comparison of ?? and the character a gives a as result. It leaves empty the parameters that are specific to the words being matched (i.e. the word itself or its canonical form).

/il/unit/lex/n/
/il/unit/lex/v/??/??/??/v-t/??/1/??/
/il/unit/lex/v/??/??/??/v-t/??/2/??/
/il/unit/lex/v/??/??/??/v-t/??/3/P/
/il/unit/lex/pre/ne/unit/lex/v/??/??/??/v-t/??/1/??/
/il/unit/lex/pre/ne/unit/lex/v/??/??/??/v-t/??/2/??/
/il/unit/lex/pre/ne/unit/lex/v/??/??/??/v-t/??/3/P/
/il/unit/lex/pro/le/unit/lex/v/??/??/??/v-t/??/1/??/
/il/unit/lex/pro/le/unit/lex/v/??/??/??/v-t/??/2/??/
/il/unit/lex/pro/le/unit/lex/v/??/??/??/v-t/??/3/P/
/il/unit/lex/v/??/??/??/v-t/??/1/??/
/il/unit/lex/v/??/??/??/v-t/??/2/??/
/il/unit/lex/v/??/??/??/v-t/??/3/P/
/il/unit/lex/det/
/il/unit/lex/adj/