1. Introduction

Many words are ambiguous in their part of speech. For example, *show* can be a noun or a verb. However, when a word appears in a text, the ambiguity is usually much reduced: in *Several of the most important studies show that experience is critical*, the word *show* can only be a verb. A lexical tagger is a system that assigns lexical categories to words. Disambiguating of lexical categories consists in using context to reduce the number of lexical categories assigned to words. Disambiguation of lexical categories is one of the major challenges facing those who devise lexical taggers.

The problem of tagging words with lexical categories arises quite often in natural language processing, e.g. in spelling correction, grammar and style checking, phrase recognition, text-to-speech conversion, corpus analysis... In parsing, the correct tagging of words is a side effect of parsing[1], but if lexical categories are previously assigned to words, higher-level analysis is usually facilitated (Milne, 1986; Hindle, 1989; Rimon, Herz, 1991; Cutting et al., 1992). Large disambiguated text corpora are important resources for many applications, e.g. for training probabilistic systems, and manual tagging is slow, expensive and error-prone. Thus, lexical taggers are useful as a front end to many natural language processing systems.

2. Methodology

The tag set

A lexical tagger labels each word-form in a text with one or several tags, i.e. codes which convey lexical information. When this information consists of part of speech only, there are about 10 to 20 lexical categories. If it includes some more grammatical data, the lexical categories are finer and more numerous. These grammatical data may be:

- lemma, e.g. *show* for the word-form *shown*, or information needed to obtain the lemma from the word-form;
- inflectional features (gender, tense...), which are a basic information in Romance languages;
- delimitation of compound words, i.e. frozen sequences of at least two simple words separated by graphic separators (e.g. *of course, text processor*), in which case specific tags are assigned to compounds, e.g. *Adverb* for *of course*;

If lemmata are explicitly given in tags, and if inflectional codes, e.g. conjugation codes, are also included in tags, word-forms can be deduced from their tags. Higher-level information, e.g. syntactic relation to the predicate (Koskenniemi, 1990), is seldom considered in disambiguation of lexical categories.

The tagged Brown Corpus uses a set of 87 simple tags (Garside, Leech, Sampson, 1987, pp. 165-183) which has been used later in other projects. For French, a tag set with part of speech and inflectional features only has approximately the same size. This paper describes experiences in French with lexical categories including lemma in lexical information, both for simple words and for compounds; the word-form can be deduced from the tag. The size of the tag set is thus that of a lexicon of the language. In the following we assume that the word-form can be deduced from the tag.

The form of disambiguated text

The output of most lexical taggers for an input text is a sequence of word/tag pairs: a unique tag is assigned to each word. This choice may a priori be supported by two assertions: (i) that it is possible to build or tune a lexical tagger so as to assign a unique tag to each word without any error; or (ii) that it is not a serious imperfection for a lexical tagger to assign a unique wrong tag to a word in a text. The status of assertion (i) is unclear, unless we consider a parser as a part of or a front end to the lexical tagger instead of the reverse: even with a coarse tag set, the correct lexical tagging of some natural sentences involves recognizing the syntactic structure of the whole sentence or even understanding its meaning. Assertion (ii) is controversial too, especially in the context of parsing: a manual correction of the output of the lexical tagger is sometimes impracticable; it is natural for a parser to rule out analyses, but not to create new analyses with parts of speech different from those in the input. Moreover, if a unique tag is assigned to each word, even actually ambiguous sentences are represented as lexically unambiguous.

In contrast, a number of recent lexical taggers

[1] This is considered as the origin of the verb *parse*.

(Silberztein, 1989; Koskenniemi, 1990; Rimon, Herz, 1991; Roche, 1992) allow for several solutions when the text is lexically ambiguous and when the only right solution cannot be found. These contributions aim at guaranteeing zero silence: the correct tag(s) for a word should never be ruled out. In other words, an analysis can be discarded only if there is no doubt that it is wrong. This objective is not often mentioned, and unrealistic for taggers that assign a unique tag to each word, unless assertion (i) is assumed.

Since tags of words in the same sentence are not independent, the output of such systems for a given sequence of words is a set of one or several sequences of tags. The set of tag sequences for a given input sequence is represented in an appropriate form. Since it is a finite set of sequences which usually have much in common, this form is always that of an acyclic finite-state automaton, a notion also termed as directed acyclic graph (DAG) or directed acyclic word graph (DAWG), finite-state machine (FSM) or finite-state network (Koskenniemi, 1990), sentence graph (Rimon, Herz, 1991), or word lattice (Vosse, 1992). One of the advantages of using acyclic automata in this context is that it systematizes the representation of lexical ambiguity, both before and after disambiguation, and for all types: parts of speech, inflectional features, phrase ambiguities (compound vs. sequence of simple words). Fig. 1 exemplifies part of speech ambiguity for *traverse* "to cross"; "strut", and phrase ambiguity for *chemin de fer* "railway" which contains *chemin* "path" and *fer* "iron".

Initial tagging and lexical disambiguation

Most lexical taggers divide the task into two steps: first (initial tagging), word forms are considered out of context to make out the list of all tags for each word; second (lexical disambiguation), context is taken into account in order to select a subset of the initial tagged sequences.

In other lexical taggers (Klein, Simmons, 1963; Dermatas, Kokkinakis, 1989; Pelillo, Refice, 1991; Brill, 1992; Federici, Pirrelli, 1992), both subtasks are carried out at the same time and a disambiguated output is built directly, generally in order to avoid the construction of a large dictionary.

The modular approach seems to us a reasonable one. The two subtasks are clearly defined. Once a tag set has been chosen, the subtasks are independent: one can expect that so could be the methods to achieve them with the best results. Improving initial tagging is a matter of morphological description of words, whereas refining upon lexical disambiguation is a matter of grammatical description of word sequences.

This approach is coherent with the use of acyclic automata for the representation of lexical ambiguity. After the initial tagging, the set of sequences recognized by the automaton is the set of a priori possible tag sequences for the input sequence. During lexical disambiguation, the automaton is modified. The number of sequences recognized by the automaton decreases, but the number of states and transitions in the automaton may increase or decrease.

The modular approach is of course of a particular interest when one uses a reliable morphological dictionary that gives for each word form, either simple or compound, the list of possible tags. In this case, initial tagging is simply performed by dictionary lookup. Such a framework for French was developed at the LADL and the CERIL with the lexicons DELAF (Courtois, 1990) and DELACF (Silberztein, 1990), and with the compression and lookup algorithms implemented by Revuz (1991) and Roche (1992) to improve upon tries (Knuth, 1973). It is now integrated into the system INTEX (Silberztein, 1993). The compressed size of DELAF is less than 900 Ko for 700.000 forms.

Handcrafted vs. statistical data

The information used by lexical taggers in order to

Fig. 1.

to disambiguate words may be hand-made grammatical knowledge or statistical data learnt automatically in large text corpora. The tagger in Hindle (1989) uses a mixture of both. Examples of lexical taggers with hand-made grammatical knowledge are Klein, Simmons (1963), Hindle (1983), Silberztein (1989), Paulussen, Martin (1992), Roche (1992). In Rimon, Herz (1991), the data are automatically produced from hand-made context-free grammars. In all of these contributions, the grammatical data are represented in finite-state automata, or could easily be. Examples of lexical taggers with statistical data are Greene, Rubin (1971), Hindle (1989), Brill (1992), Federici, Pirrelli (1992), where the data are rules produced through statistical processes; Marshall (1983), Jelinek (1985), DeRose (1988), Cutting et al. (1992), etc., where the data consist of tables of statistics, e.g. parameters of a Markov model. All of these contributions assign a unique tag to each word.

The arguments in favour of any of these two types of approaches often refer to their ability to contend with the diversity of unrestricted text. Some doubt that hand-made linguistic knowledge can take into account all that can happen in unrestricted text. Others think that information learnt automatically in a text corpus, even in a large one with carefully distributed samples of text, is not accurate enough for unrestricted text. It seems to us that the use of hand-written linguistic knowledge is better adapted if one aims at zero silence, i.e. at guaranteeing that an analysis is discarded only if there is no doubt that it is wrong. The writer of the linguistic data can use a text corpus as an aid, but must be able to create counter-examples that are not in the corpus, so that the resulting linguistic data may be independent of it.

We will focus on lexical tagging with the objective of zero silence. A method adapted to this objective combines the possibility of several solutions (the output for an input word sequence is an acyclic automaton), the modular approach (initial tagging and lexical disambiguation are considered independent once a tag set is chosen), and the use of hand-made linguistic knowledge for initial tagging (large morphological dictionaries) and for lexical disambiguation (grammatical data). The only contributions in this framework are those of Silberztein (1989, 1993) and Roche (1993). Both consist of implemented algorithms and use the same dictionaries. In the following, we make formal descriptions of the two systems and a formal and practical comparison. Mathematical terminology is that of Berstel (1979) with the exception that the empty word is denoted by ε.

3. The tag sets

Roche (1992)

In this system, a tag is a sequence of several symbols that belong to a finite alphabet A_1, which is defined as follows. Let

$Cat = N \mid V \mid A \mid ADV \mid DET \mid DETP \mid DETQ \mid PRO \mid PREP \mid CONJC \mid CONJS \mid INTJ \mid XINC$

be the set of parts of speech[2],

$Mf = N1 \mid N2 \mid N3 \mid ... \mid V1 \mid V2 \mid V3 \mid ... \mid A1 \mid A2 \mid A3 \mid ... \mid A79 \mid A80$

be the set of inflectional codes, e.g. conjugation codes,

Voc = *a* | *à* | *aa* | *aabam* | *aalénien* | *aalénienne* | *aaléniennes* | *aaléniens* | ... | *zyzomys* | *zzz*

be the set of all possible inflected forms in the language, including compounds like *chemin de fer*,

$Val = v\text{-}t \mid P \mid F \mid C \mid I \mid J \mid S \mid T \mid Y \mid G \mid K \mid W \mid 1 \mid 2 \mid 3 \mid m \mid f \mid s \mid p$

be the set of inflectional feature values, and *Unit* be a symbol which appears in front of every token. Now $A_1 = Unit \mid Cat \mid Mf \mid Voc \mid Val$; the tag set is

$C \subset Unit\ Cat\ (Mf\ Voc^+\ Val^+ \mid \varepsilon) \quad Voc^+ \subset A_1{}^+$

C is a code because the first symbol of every element of C is the symbol *Unit*.

Two of the initial taggings of

(1) *Il traverse un cours d'eau* "He crosses a river"

take the following forms:

(2) *Unit PRO il m s 3 Il Unit V 3 traverser P 3 s traverse Unit DET un m s un Unit N cours/d'eau m s cours/d'eau*

(3) *Unit PRO il m s 3 Il Unit N 21 traverse f s traverse Unit DET un m s un Unit V 31 courir P 2 s cours Unit PREP d Unit N 23 eau f s eau*

Note that (2) is the only correct tagging of the sentence. In (3), *traverse* is mistaken for a noun and *cours* is mistaken for a verb.

[2] In the following, most sets of symbols and sets of sequences are represented as rational expressions, and the bar | stands for set union.

We define a rational word function

$\varphi_1 : A_1^* \dashrightarrow Voc^*$

which maps any sequence of tags into the corresponding sequence of word-forms by deleting all other information; dom$\varphi_1 = C^*$. Examples:

φ_1(*Unit DET le Unit N N*1 *passe m s passe*) = *le passe*

$\varphi_1^{-1}(\{le\ passe\})$ = *Unit* (*DET* | *PRO*) *le Unit* (*N* (*N*1 *passe m* | *N*21 *passe f*) | *V V*3 *passer v-t* ((*P* | *S*)(1 | 3) | *Y* 2)) *s passe*

φ_1(*Unit N;NA coup fumant m s coup fumant*) = *coup fumant*

The result of dictionary lookup for an input sequence *t* in *Voc** is $\varphi_1^{-1}(\{t\})$. This set of tags is represented in the form of an acyclic automaton on the alphabet A_1.

Silberztein (1989, 1993)

It uses another alphabet, A_3, which contains all possible complete word tags in the lexicon, e.g.:

<il,PRO:ms3>
<traverser,V3:P3s>
<cours/d'eau,NDN:ms>

The tag gives the lemma, the part of speech, the inflectional code if any, and after a semicolon the sequence of inflectional features if any. The a priori sentence tags (2) and (3) above now take other forms:

(4) *<il,PRO:ms3> <traverser,V3:P3s> <un,DET:ms> <cours/d'eau,NDN:ms>*

(5) *<il,PRO:ms3> <traverse,N21:fs> <un,DET:ms> <courir,V31:P2s> <de,PREP> <eau,N23:fs>*

We define a continuous morphism

$\varphi_2 : A_3^* \dashrightarrow Voc^*$

which maps any sequence of tags into the corresponding sequence of word-forms by deducing the word-form and deleting all other information. Examples:

φ_2(*<le DET:ms> <passe N*1*:ms>*) = *le passe*

$\varphi_2^{-1}(\{le\ passe\})$ = (*<le DET:ms>* | *<le PRO:ms*3*s>*)(*<passer V*3*:P*1*s>* | *<passer V*3*:P*3*s>* | ... | *<passe N*21*:fs>*)

φ_2(*<coup/fumant N;NA:ms>*) = *coup fumant*

The result of dictionary lookup for an input sequence *t* in *Voc** is $\varphi_2^{-1}(\{t\})$. This set of tags is represented in the form of an acyclic automaton on the alphabet A_2.

The elements of A_3 are completely specified tags. This system also uses another alphabet, A_2, which contains incomplete grammatical specifications, namely:

- all possible simple word-forms, i.e.
  Voc = *a* | *à* | *aa* | *aabam* | *aalénien* | *aalénienne* | *aaléniennes* | *aaléniens* | ... | *zyzomys* | *zzz*
- incomplete grammatical specifications on part of speech, inflectional feature values, lemma, or a combination of these, e.g.:
  <ADV> for any adverb
  <V> for any verb form
  <V:Ps> for any verb form in the present singular
  <traverser> for any word-form whose lemma is *traverser*
  <passeur, N:s> for any singular form of the noun passeur

<prendre:P>	*<V:P>*
*<prendre:P*3*p>*	*<V:P*3*p>*
<prendre:p>	*<V:p>*

- the symbol *<MOT>* which matches any simple word.

If and only if an element *a* of A_3 may satisfy the constraints expressed in an element *b* of A_2, we write $a \in \sigma(b)$, thus defining a substitution σ from A_2^* into the power-set of A_3^*:

σ(*<prendre:P*3*p>*) = *<prendre V*66*:P*3*p>*

σ(*<prendre:P>*) = *<prendre V*66*:P*1*s>* | *<prendre V*66*:P*2*s>* | ... | *<prendre V*66*:P*3*p>*

σ(*passe*) = *<passer V*3*:P*1*s>* | *<passer V*3*:P*3*s>* | ... | *<passe N*21*:fs>*

σ(*<MOT>*) = σ(*Voc*)

Relations between A_1 and A_3

The tag sets *C* and A_3 are equivalent. We define an injective morphism

$\alpha : A_3^* \dashrightarrow A_1^*$

and we extend it to an isomorphism from A_3^* onto C^*. Example:

α(*<le DET:ms> <passe N*1*:ms>*) = *Unit DET le Unit N N*1 *passe m s passe*

We have $\varphi_1 \circ \alpha = \varphi_2$.

4. Specifying a formal language defines a lexical disambiguation

Let $L \subset C^*$ be a set of tag sequences: for each text $t \in Voc^*$, the set $\varphi_1^{-1}(\{t\}) \cap L$ is the set of taggings of *t* that are in *L*. Let *Corr* be the set of correct taggings of all texts: $\varphi_1^{-1}(\{t\}) \cap Corr$ is the set of correct taggings of t. If we specify L and compute $\varphi_1^{-1}(\{t\}) \cap L$, the noise in the tagging is thus

$$\varphi_1^{-1}(\{t\}) \cap L \setminus Corr$$

and the silence is

$$\varphi_1^{-1}(\{t\}) \cap Corr \setminus L$$

To ensure that silence rate is zero, we have to specify L so that each correct text tagging be in L. We call the specification of L a local grammar, since it deals with grammatical notions and it can be robust for local constraints but not, up to now, for global constraints in sentences. Usually, L is a rational language.

The same local grammar can be used to tag texts and to detect errors. Whenever $\varphi_1^{-1}(\{t\}) \cap L$ is empty, the text t has no correct tagging and contains an error.

Roche (1992)

Let $F \subset A_1{}^*$ be a set of sequences that can never appear in a correct text tagging, e.g.

$$Unit\ DET\ un\ m\ s\ un\ Unit\ V\ A_1\ A_1\ P$$

where A_1 stands for any element of A_1. This interdiction means that the determiner *un* cannot be followed by a verb in the present. The set

$$R(F) = C^* \setminus A_1{}^* F A_1{}^*$$

is the set of taggings that have no factor in the set F of forbidden sequences: every correct text tagging should be in $R(F)$. A lexical disambiguation is thus defined by a specification of $R(F)$. This specification takes the form of a finite finite-state automaton which recognizes F and is called a local grammar. A lexical disambiguation is performed by selecting, for each text $t \in Voc^*$ and each rational set F, the set $\varphi_1^{-1}(\{t\}) \cap R(F)$ of taggings of t with no forbidden sequences. The efficient algorithm of Roche (1992) constructs an automaton that recognizes $\varphi_1^{-1}(\{t\}) \cap R(F)$ directly from an automaton that recognizes the set $\varphi_1^{-1}(\{t\})$ and an automaton that recognizes F.

This type of local grammars are cumulative: for each rational sets $F_1, F_2 \subset A_1{}^*$,

$$R(F_1 \cup F_2) = R(F_1) \cap R(F_2)$$

For each rational language F, there is a rational language F_0, minimal for set inclusion, such that:

$$R(F_0) = R(F)$$

and the elements of F_0 are minimal in F and in F_0 for factor ordering.

Silberztein (1989, 1993)

This algorithm is exemplified and informally described in Silberztein (1993). We give a more formal specification of the lexical disambiguation performed by this algorithm.

As above, let $L \subset A_3{}^*$ be a set of tag sequences: for each text $t \in Voc^*$, the set $\varphi_2^{-1}(\{t\}) \cap L$ is the set of taggings of t that are in L. If each correct text tagging is in L, computing $\varphi_2^{-1}(\{t\}) \cap L$ performs a lexical disambiguation.

The local grammar is a strictly alphabetic finite transducer, i.e. a finite automaton each transition of which is labelled by one element in A_2 as input label and one element in A_2 as output label. It realizes

Fig. 2.

realizes a rational transduction $\tau \in \mathrm{Rat}(A_2{}^*\times A_2{}^*)$, i.e. a relation between input sequences in $A_2{}^*$ and output sequences in A_2. The transduction must have the property that $(\varepsilon, \varepsilon) \notin \tau$ where ε is the empty word. Fig. 2 is an example of this type of local grammar. The set domτ is the set of possible input sequences of τ and will recognize grammatical sequences. The set imτ is the set of possible output sequences of τ and will impose grammatical constraints on the sequences recognized.

Let ρ be the equivalence relation over $A_3{}^*$ defined by:

$$\forall\ u, v \in A_3{}^* \quad (u\, \rho\, v \ \Leftrightarrow$$
$$(|u| = |v| \ \text{ and } \ \forall\ i \in [1, |u|]\ \ \varphi_2(u_i) = \varphi_2(v_i)))$$

e.g. <*superbe N*21*:fs*> <*gaulliste A*31*:fs*> and <*superbe A*31*:fs*> <*gaulliste N*31*:fs*> are in relation by ρ but <*pomme/de/terre N;NDN:fs*> <*cuire V*91*:Kfs*> and <*pomme N*21*:fs*> <*de PREP*> <*terre/cuite N;NA:fs*> are not.

We define a transduction $g(\tau) \in \mathrm{Rat}(A_3{}^*\times A_3{}^*)$ by:

$$\forall\ u, v \in A_3{}^* \quad ((u, v) \in g(\tau) \ \Leftrightarrow$$
$$(\exists\ (u', v') \in \tau\ \ (u \in \sigma(u') \text{ and } v \in \sigma(v'))$$
$$\text{and } u\, \rho\, v))$$

For each text $t \in Voc^*$, we define a transduction $f(t, \tau) \in \mathrm{Rat}(A_3{}^*\times A_3{}^*)$ by:

$$\forall\ u, v \in A_3{}^* \quad ((u, v) \in f(t, \tau) \ \Leftrightarrow$$
$$(u, v \in \mathrm{Pref}(\varphi_2{}^{-1}(\{t\})) \text{ and } (u, v) \in g(\tau))$$

For each text t, we define a set $F_\tau(t) \subset A_3{}^*$ by:

$$\text{if } \sigma(\mathrm{dom}\tau) \cap \mathrm{Pref}\ (\varphi_2{}^{-1}(\{t\})) = \emptyset,$$
$$F_\tau(t) = A_3 \cap \mathrm{Pref}\ (\varphi_2{}^{-1}(\{t\}));$$
$$\text{else} \qquad F_\tau(t) = \mathrm{im}f(t, \tau).$$

Now we define a set $S(\tau) \subset A_3{}^*$ by defining the set $S(\tau) \cap \varphi_2{}^{-1}(\{t\})$ for each t by induction on $|t|$:

$$S(\tau) \cap \varepsilon = \varepsilon;$$
$$\text{if } |t| > 0, S(\tau) \cap \varphi_2{}^{-1}(\{t\})$$
$$= \bigcup_{u \in F_\tau(t)} u.(S(\tau) \cap \varphi_2{}^{-1}(\{\varphi_2(u)^{-1}t\}))$$
$$= \{uv \in A_3{}^* \mid \varphi_2(u)\varphi_2(v) = t \ \text{ and } \ u \in F_\tau(t) \ \text{ and } \ v \in S(\tau)\}$$

An equivalent definition of $S(\tau)$ uses a new symbol $@ \notin A_2 \mid A_3$ and the projection

$$\varphi_@ : (A_3 \mid @)^* \dashrightarrow A_3{}^*$$

such that $\varphi_@(@) = \varepsilon$:

$$S(\tau) = \varphi_@((@A_3 \mid \mathrm{im}g(\tau))^* \setminus (@ \mid A_3)^* @ \varphi_@{}^{-1}(\varphi_2{}^{-1}(\varphi_2(\sigma(\mathrm{dom}\tau))Voc^*))$$

The local grammar is written so that each correct text tagging be in $S(\tau)$. A function of INTEX (Silberztein, 1993) constructs an automaton that recognizes $\varphi_2{}^{-1}(\{t\}) \cap S(\tau)$ directly from an automaton that recognizes the set $\varphi_2{}^{-1}(\{t\})$ and a transducer that realizes τ. For example, the local grammar in Fig. 2 applied to the text *la veine porte* produces the automaton in Fig. 3.

5. A formal comparison

From a formal point of view, both methods provide possibilities of defining rational languages, but these theoretical possibilities are not exactly the same for the two methods, i.e. there exist some $S(\tau)$ that cannot be expressed in the form $\alpha^{-1}(R(F))$, and some $R(F)$ that cannot be expressed in the form $\alpha(S(\tau))$.

An $S(\tau)$ which is not an $\alpha^{-1}(R(F))$

For each rational language F, the following holds:

$$\forall\ u_1, u_2 \in A_1{}^*\$$$
$$(u_1 u_2 \in R(F) \Rightarrow u_1 \in R(F) \text{ and } u_2 \in R(F))$$

Now define τ with the graph of Fig. 4. Assume that there is a rational language $F \subset A_1{}^*$ such that $S(\tau) = \alpha^{-1}(R(F))$. Then

Fig. 3.

Fig. 4.

$$\forall\, u_1, u_2 \in A_3^*$$

$$(u_1 u_2 \in S(\tau) \Rightarrow u_1 \in S(\tau) \text{ and } u_2 \in S(\tau))$$

In fact, with $\varphi_2(u_1)$ = *sent* and $\varphi_2(u_2)$ = *il la touche*:

u_1u_2 = *<sentir V28:P3s> <il PRO:ms3s> <le DET:fs> <touche N21:fs>* $\in S(\tau)$

u_1 = *<sentir V28:P3s>* $\in S(\tau)$

but

u_2 = *<il PRO:ms3s> <le DET:fs> <touche N21:fs>* $\notin S(\tau)$

An $R(F)$ which is not an $\alpha(S(\tau))$

Some $R(F)$ cannot be expressed in the form $\alpha(S(\tau))$. It is the case whenever all elements of F are minimal for factor ordering, f and g are elements of $\alpha^{-1}(F) \setminus \varepsilon$, and f has a prefix $h \neq f$ such that $\varphi_2(h) = \varphi_2(g)$. Then no transduction τ in $\mathrm{Rat}(A_2^* \times A_2^*)$ satisfies $R(F) = \alpha(S(\tau))$.

We prove it for a particular language:

F = Unit PRO il Unit DET le | *Unit PRO il Unit PRO le Unit V V66 prendre v-t P 2 s prends*

$\alpha^{-1}(F)$ = *<il PRO:ms3s> <le DET:ms>* | *<il PRO:ms3s> <le PRO:ms3s> <prendre V66:P2s>*

Assume there exists $\tau \in \mathrm{Rat}(A_2^* \times A_2^*)$ such that $R(F) = \alpha(S(\tau))$.

Assume that $\sigma(\mathrm{dom}\tau) \cap \mathrm{Pref}\,(\varphi_2^{-1}(\{\textit{il le}\})) = \emptyset$. Then

F_τ(*il le*) = *<il PRO:ms3s>*

$\varphi_2^{-1}(\{\textit{il le}\}) \cap S(\tau)$ =

<il PRO:ms3s>$(\varphi_2^{-1}(\{\textit{le}\}) \cap S(\tau))$

but

Unit DET le $\in R(F)$

Consequently,

<le DET:ms> $\in S(\tau)$

and

<il PRO:ms3s> <le DET:ms> $\in S(\tau)$

whereas

Unit PRO il Unit DET le $\notin R(F)$:

contradiction.

Thus $\sigma(\mathrm{dom}\tau) \cap \mathrm{Pref}\,(\varphi_2^{-1}(\{\textit{il le}\})) \neq \emptyset$ and

F_τ(*il le*) = imf(*il le*, τ).

We have $\varphi_2^{-1}(\{\textit{il le}\}) \subset \mathrm{Pref}\,(\varphi_2^{-1}(\{\textit{il le prends}\}))$, hence

$$\sigma(\mathrm{dom}\tau) \cap \mathrm{Pref}\,(\varphi_2^{-1}(\{\textit{il le}\})) \subset$$

$$\sigma(\mathrm{dom}\tau) \cap \mathrm{Pref}\,(\varphi_2^{-1}(\{\textit{il le prends}\})) \neq \emptyset$$

and

F_τ(*il le prends*) = imf(*il le prends*, τ).

Now

Unit PRO il Unit PRO le $\in R(F)$

therefore

<il PRO:ms3s> <le PRO:ms3s> $\in S(\tau)$

Thus there are $u, v \in A_3^*$ such that:

$u\,v$ = *<il PRO:ms3s> <le PRO:ms3s>*

$u \in F_\tau$(*il le*)

$v \in S(\tau)$

but

$u \in F_\tau$(*il le*) = imf(*il le*, τ) $\subset$ imf(*il le prends*, τ)

= F_τ(*il le prends*)

and, since $u \neq \varepsilon$, $\alpha(v) \neq$ *Unit PRO il Unit PRO le*, hence

$\alpha(v)$ *Unit V V66 prendre v-t P 2 s prends* $\in R(F)$

therefore

v *<prendre V66:P2s>* $\in S(\tau)$

then

$u\,v$ *<prendre V66:P3s>* $\in$

F_τ(*il le prends*)$(\varphi_2^{-1}(\{\varphi_2(v)\ \textit{prends}\}) \cap S(\tau))$

so

<il PRO:ms3s> <le PRO:ms3s> <prendre V66:P2s> $\in S(\tau)$:

contradiction.

Similar methods with boundary symbols

The counterexamples above show that none of the methods has a larger expressive power than the other. The differences are related to the recognition of the left and right limits of sentences[3]. It is possible to modify the methods

[3] D. Perrin called my attention to that.

above in order to introduce boundary symbols, so that the resulting methods share exactly the same theoretical expressive power (Laporte, to appear).

6. A practical comparison

From a formal point of view, the methods described above are ways of defining a rational language $L \subset C^*$ (resp. $L \subset A_3^*$) by specifying a finite-state local grammar. Their interest does not come from that, since any rational language can be specified in a such a way (Kleene's theorem). It is rather a practical interest. Local-grammars are hand-made linguistic data: they are useful if they are easy to write and easy to check, i.e. if with a limited size and a good level of readability, they define a superset of *Corr*, a "large" language. Considerations on the ease of designing and checking local grammars are subjective, they depend on the grammatical intuitions of the writer of the grammars. The following is based on our experiments in writing local grammars and testing them with both tools on text corpora.

Designing local grammars

Look at a portion of non-disambiguated tagged text, e.g. in Fig. 5, produced by INTEX: that will probably immediately give you ideas of disambiguation rules. Such spontaneous ideas usually sound like "The determiner *du* cannot be followed by a tensed verb", i.e. some sequence is forbidden, or "If the determiner *du* is followed by a verb, the verb can only be in the present participle", which contains a recognition part and a grammatical constraint. Designing a first version of a local grammar is an easy task, e.g. using INTEX. Depending on the writer of the grammar, the fact that Roche's local grammars contain negative information may be felt as an obstacle; for our own part we did not feel so.

Checking local grammars

Grammatical intuitions may be wrong, e.g. "A noun noun is not followed by a noun". Our objective of zero silence requires testing carefully the grammars.

Roche's grammars of forbidden sequences can be checked manually, using one's intutition to find counterexamples to them and refine them (this is a faculty that can be trained). Corpora and a tool to analyse and visualise them can be helpful: a forbidden grammatical sequence should not be found in corpora. INTEX can locate in corpora the occurrences of a given grammatical pattern, but even for an actually forbidden sequence, many occurrences (with incorrect tags) may be found. The writer of the grammar can use the occurrences located to find counterexamples. Even with a tagged corpus, this aid cannot easily be turned to an automatic test: the fact that a grammatical sequence is not in a given tagged corpus does not mean it is forbidden. It is to be expected that human intuition can be trained to be more efficient than automatic browsing, for this task as for many others.

The formal definition of the languages specified by Silberztein's local grammars show how the rules of interpretation of these grammars are complex. This formal definition is necessary to know in detail what a grammar will do when applied to texts. However, the way those grammars work is more intuitive than what the formal definition would suggest. Here again, it is possible to train one's intuition to find counterexamples to a grammar and refine it. The complexity of the method comes up e.g. if you have two correct local grammars and want to build a grammar that solves the same ambiguities: in general, their union is not correct and you have to study them in detail.

Fig. 5.

Conclusion

The disambiguating tools described above have much in common: in both cases, finite-state local grammars are used to define rational languages, and the grammar is written in such a way that every correct tagged sequence should belong to the rational language. The way to write the grammar depends on the rules which define the rational language as a function of the grammar. Other disambiguating tools could be designed following the same general principle.

We mentioned that such tools can also be used to detect non-lexical errors. Another application is to be foreseen if we follow the same general principle but use tags with syntactic information in addition to grammatical information, including boundary tags. The result of the tagging would then take into account syntactic structure and more lexical ambiguities, and therefore would be a parsing of input text.

References

Berstel, Jean. 1979. *Transductions and Context-Free Languages*, Stuttgart: Teubner, 278 p.

Brill, Eric. 1992. "A Simple Rule-Based Part of Speech Tagger", *3rd Applied ACL*, Trento (Italy), pp. 152-155.

Courtois, Blandine. 1990. "Un système de dictionnaires électroniques pour les mots simples du francais", in *Langue française* no. 87, *Dictionnaires électroniques du francais*, Paris: Larousse, pp. 11-22.

Cutting, Doug, JulianKupiec, Jan Pedersen,Penelope Sibun. 1992. "A practical part-of-speech tagger", *3rd Applied ACL*, Trento (Italy), pp. 133-140.

Federici, Stefano, Vito Pirrelli. 1992. "A Bootstrapping Strategy for Lemmatization: Learning Through Examples", *Papers in Computational Lexicography. COMPLEX 92*, F. Kiefer, G. Kiss, J. Pajzs, eds., Linguistics Institute of the Hungarian Academy of Sciences, Budapest, pp. 123-135.

Garside, Roger, GeoffreyLeech, Geoffrey Sampson. 1987. *The Computational Analysis of English*, Longman.

Hindle, Donald. 1983. "Deterministic parsing of syntactic non-fluencies", *21st Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference.*

Hindle, Donald. 1989. "Acquiring disambiguation rules from text", *27th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference,* pp. 118-125.

Koskenniemi, Kimmo. 1990. "Finite-state parsing and disambiguation", *Proceedings of COLING 90*, H. Karlgren, ed., Helsinki University, pp. 229-232.

Milne, Robert. 1986. "Resolving Lexical Ambiguity in a Deterministic Parser", *Computational Linguistics*, vol. 12, no. 1, pp. 1-12.

Paulussen, Hans, Willy Martin. 1992. "DILEMMA-2: a Lemmatizer-Tagger for Medical Abstracts", *3rd Applied ACL*, Trento (Italy), pp. 141-146.

Pelillo, Marcello, Mario Refice. 1991. "Syntactic disambiguation through relaxation processes", *Eurospeech 91*, vol. 2, pp. 757-760.

Revuz, Dominique. 1991. *Dictionnaires et lexiques, methodes et algorithmes*, PhD dissertation, Publication no. 91-44 of LITP, University Paris 7, 105 p.

Rimon, Mori, Jacky Herz. 1991. "The recognition capacity of local syntactic constraints", *5th Conference of the European Chapter of the ACL. Proceedings of the Conference*, Berlin, pp. 155-160.

Roche, Emmanuel. 1992. "Text disambiguation by finite-state automata, an algorithm and experiments on corpora", in *COLING-92, Proceedings of the Conference*, Nantes.

Silberztein, Max. 1989. *Dictionnaires électroniques et reconnaissance lexicale automatique*, PhD dissertation, LADL, University Paris 7, 176 p.

Silberztein, Max. 1990. "Le dictionnaire électronique des mots composés", in *Langue française* no. 87, *Dictionnaires électroniques du francais*, Paris: Larousse, pp. 71-83.

Silberztein, Max. 1993. *Dictionnaires électroniques et analyse automatique de textes. Le système INTEX*, Paris: Masson, 233 p.

Vosse, Theo. 1992. "Detecting and Correcting Morpho-syntactic Errors in Real Texts", *3rd Applied ACL*, Trento (Italy), pp. 111-118.

7. Evaluation of results

Grammatical disambiguation algorithms should ideally decide upon one of the several possible tags for a given token. Since this is not always possible, they reduce the number of possible tags. Accuracy in the results of the tagging is required, since they serve as input to further processing without any possibility of human post-edition. In particular, discarding the correct tag for a word in the text to be processed may cause the failure of the processing of a whole sentence. Thus, two figures are of interest for evaluating the accuracy of a grammatical disambiguation system: the average proportion of tags selected among a priori possible tags per word (reduction rate), but also the proportion of words whose correct tag is discarded by the system (error rate).

The size of the tag set is of course a crucial element in the evaluation of a lexical tagger. The larger this set, the finer the information conveyed by each tag. Moreover, noise rates and silence rates in systems with different tag sets may not be compared, since a coarser tag set automatically reduces both noise rate and silence rate.

1) stochastic and probabilistic methods (e.g. D. Hindle, 1989; M. Pelillo, M. Refice, 1991), which are based upon statistics on a large text corpus; they generally reach the optimal reduction rate (even ambiguous sentences are represented as unambiguous) and the most recent yield error rates of a few percent;

2) grammatical methods (e.g. D. Hindle, 1983, M. Silberztein, 1989, 1993, E. Roche, 1993), designed to be independent of any corpus and thus likelier to contend with the diversity of unrestricted text; their interest comes from the fact that they can aim at zero error rate, if one accepts a reduction rate not optimal. handcrafted knowledge