Using automata to detect and correct errors in the written English of French-speakers [◊]

Natalie Kübler & Etienne Cornu

Abstract

This paper presents a second-language grammar checking approach based on a particular form of finite state automata. Three types of automata are used to track down an error: data extraction automata which build complex noun phrases and identify verbal groups, filter automata which look for specific contexts and detection automata which are used to find specific errors.

1. Introduction

The work we present here is part of the ARCTA Prototype project which consists in the development of a second language grammar checker for French native speakers who write in English[1]. In this article, we describe the formalism we have used to encode and detect errors.

The general subject of error detection has developed rapidly over the last few years. However, even today's best grammar checkers have severe limitations when dealing with texts written in a second language. Several questions must be answered in order to achieve the goal of detecting and correcting the errors people really make: Which errors should be dealt with? How should the linguistic data be organized? Which mechanism can be used? Tschumi & Tschichold (this issue) answer the first question and we will try to address the other two.

Organizing the linguistic data in a grammar checker represents the first important step, especially in the field of second-language error detection. The errors that native speakers of French make in English must be taken into account when structuring this information. The specific

[◊] This research was supported by a grant from the Swiss CERS/KWF (2054.2).

[1] For a general presentation of the project, see the article by C. Tschumi, F. Bodmer, E. Cornu, F. Grosjean, L. Grosjean, N. Kübler & C. Tschichold (this issue).

linguistic structures concerned influence the functionalities implemented in the prototype, and, conversely, the mechanism selected to detect errors has an impact on the organization of the data.

The specific mechanism implemented in our prototype is based on a variation of finite state automata, i.e. the augmented transition networks first described by Woods (1970). They operate on two levels: on the terminal symbols, i.e. syntactic categories such as *N* (noun) or *V*(verb) that cannot be further analysed, and on two types of non-terminal symbols, noun phrases and prepositional phrases.

2. Linguistic Data

Our prototype contains a dictionary, feature lists and various automata that reflect how linguistic information has been organized. Basic data that concern syntactic category and inflectional information are stored in the dictionary. More specific information about words is either represented under the form of Boolean attributes or consists of tables containing different types of attribute-value pairs.

2.1. Error-oriented language description

Our model does not contain production rules such as *S==>NP VP* that are found in grammar checkers based on a syntactic theory such as GPSG, LFG (Cornu, 1992) and ACFG (Vosse, 1992). We have determined (see Cornu, 1992) that with today's parser technology a language description based on the structures present in ill-formed texts is better adapted to the problem.

The errors that we process can be split roughly into three subclasses: morpho-syntactic errors, single word errors, and structural errors. We will present each type below and will then focus on structural errors.

2.1.1. Morpho-syntactic errors

Morpho-syntactic errors occur in situations where grammatical rules govern the morphological aspects of the words; they are therefore based on syntactic categories. Subject-verb agreement or erroneous adjective-noun agreement in the noun phrase belong to this subclass:

1.**He*	*collect*	*money*	*for*
Pron[3rd,sing]	V[plur]	NP	Prep
his association			
NP			

2.**These*	*olds*	*houses*	*are*	*expensive*
Det[plur]	*Adj[*plur]*	*N[plur]*	*V*	*Adj*

In (1), the verb should have the third person singular form *collects*; in (2), since adjectives are invariable in English, the adjective *old* should not take a plural ending.

2.1.2. Single-word errors

Errors that are based on specific lexical items and that cannot be associated to word classes are considered to be single-word errors. They are generated mostly by a lack of knowledge about exceptions or idiomatic structures in English (frozen or semi-frozen structures). Function words can also be used erroneously. The confusion between *each* and *every* represents one of these errors.

2.1.3. Structural errors

These errors concern subclasses of words that show the same syntactic behavior. Basic syntactic categories such as *Verb* are split into subclasses depending on their subcategorization. Noun predicates and adjective predicates can be distributed into subclasses as well. Errors occurring with one of these words can be found with all the other items that belong to the same subclass. Prepositions following a noun, an adjective, or a verb are very often incorrect. Lists containing the correct preposition can be included in the data:

reason for; responsible for; drop in; conscious of; effect on.

However, more complex information about the syntactic structures and the positions of the predicate's arguments is needed. Report verbs and dative verbs generate errors in syntactic structures, as in:

1a. **You will ask questions to his friends*
1b. **They showed to each other...*
2a. **I explained him that I was poor myself*

2b. **They will not allow to them to come*
3a. **That permitted to become rich*
3b. **Marian gave to him a message*
4a. **They remind us (about+E) those in Japan* (*E* = empty category)
4b. **He attribute her wonderful qualities*
5a. **He told the poverty of his order*
5b. **We bought the suit to one of your competitors*

A formal description of these structures can be given in part in tables of the following form:

orientation	engl verb	N0 V N1	N1 =: question answer	N1 =: fact event	N0 V that S	N0 V that S (mod/sub)	N0 V that N0 V modal	N0 V Nhum that S	N0 V prep Nhum that S	Préposition	-ing Form	N1 -ing Form	N0 V to inf	N0 V to inf0	N0 V Nhum to inf	N0 V prep Nhum to inf	N0 V wh- to inf	Quote	whether/if S	wh-question	it is Vpp that S	N0 is Vpp to inf
thou	accept	+	+	+	+	-	-	?	?	-			+		-	?	-	-	-	-	+	-
thou	accept	+	+	+	?	-	-	?	?	-			+		-	?	-	-	-	-	+	-
n	acknowledge		+	+	+	-		-	+	to											+	-
	adjure							+	-	E												
DOR	admit	-	-	+	+	-	-	-	+	to	+	-	-	+	-	-	-	+	-	-	+	-
DOR	advise		-	-	-	-	+	-	-	E			-	-	+	-	-	+	-	-	-	-
thou	agree				+	+		-	+	with	-	-	-	+	-	+		+			+	+
n	allege				+	-		-	+	to											+	+
DOR	announce			+	+	-	-	-	+	to			-	-	-	-	-	+	-	+	+	-
	answer		+	-	+	-	-	+	-	E			-	-	+	-	-	+	-	-	-	-
DOR	appear				it	-		-	+	to	-	-	+	+	-	+					-	-
ROD	ask		+	-	-	+	?	+	-	E	-	-	-	+	+	-	+	+	+	+	-	-
DOR	assure	-	-	of +	-	-	-	+	-	E			-	-	?	-	-	+	-	-	-	-
DOR	bark		+	-	+	-	-	-	+	at			?	-	?	?+	?	+			-	-
ROD	beg		+?	-	-	-	-	?	+	from			-	-	+	-	-	+	-	-	-	-

In our system, the syntactic information contained in these tables is represented by lists and by the automata that use these lists. One of the lists dealing with report verbs has the following format:

admit	*Prep=Y*	*Prep_Gov=to*
announce	*Prep=Y*	*Prep_Gov=to*
tell	*Prep=N*	*Prep_Gov=∅*

3. Error detection

Before we describe error detection as such, we have to understand the steps that take place before the automata are executed.

3.1 Preprocessing

The basic text is tagged and disambiguated, which means that each lexical unit receives one and only one syntactic category[2]. Basic noun phrases are located by an independent mechanism. Here are some examples of basic *NP*s that must be found before the automata can be executed:

[the planet Mars]
NP

[The man] [whom] [I] saw gave [me] [those]
NP NP NP NP NP

[a house] *of* *[crystal pillars]*
NP Prep NP

Information about these groups of words can be accessed at two levels: the *NP* level and the terminal units level (syntactic categories such as *ART*, *ADJ* and *N*). For some compounds, detailed information cannot be accessed since the system uses a list of basic and very common sequences of words belonging to different syntactic categories: pronouns, prepositions, adjectives, adverbs and nouns. Here are some examples of such "frozen" expressions:

[2] See the article by F. Bodmer (this issue).

a lot of	PRONOUN==> tagged as a single syntactic category, without the detailed analysis *ART N PREP*
well-worn	ADJECTIVE
according to	PREPOSITION
carte blanche	NOUN
in order to	CONJUNCTION

Our automata operate on three levels. The first level is a preprocessing one and is concerned only with extracting new data from existing structures (Data Extraction Automata, DEA). The second level is optional and contains filter automata (FA). These identify sentence patterns in order to determine when to activate the third-level which contains detection automata (DA). The following diagram clarifies the organization of our automata:

Level 1	Level 2	Level 3
DEA	FA ==>	DA1
DEA		DA2

3.2. Data extraction automata

One of the tasks of first level automata consists in building complex noun phrases on the basis of simple ones that have been identified during the preprocessing stage. For the complex *NP*: *a house of crystal pillars*, two basic *NP*s have already been identified:

[a house]	*of*	*[crystal pillars]*
NP	Prep	NP

The automaton used to build the complex *NP* in such a case has the following structure:

```
AUTOMATON NP_OF_NP

(+NEWNP, ENV_CL::=NP_COMPL,NBR_NP::=$N)
($P/='of')
[CAT=V, +PREPOS_V, PREP_GOV=>$P]*1
<NP (NP_POS::=NP_1,NBR_NP=>$N)>
@[IF='of']
<NP (NP_POS::=NP_2)>
```

The anchoring - i.e. the starting point of the automaton - is set on the inflected form *of*. The system first looks to the right for an *NP*, then to the left of the preposition for another *NP* and to the left of that *NP* for a potential verb which could belong to one of the subclasses of prepositional verbs. If it finds one, the automaton then checks if the verb governs the preposition *of*. If it does, the automaton fails and nothing happens. If it does not, a complex *NP* is formed (*+NEWNP*) that inherits the *NUMBER* feature of the first basic *NP*. If the latter has other features, these are also passed on to the complex *NP* of a higher level. In this case, the first basic *NP* is taken as the head of the complex *NP*.

A slight inconvenience of this approach is that some complex *NP*s will never be detected. If the verb can either be followed by *of* or function as a transitive verb, it might be possible that the preposition really belongs to a complex *NP* and not to the verb. In that case, the system misses a complex *NP*. However, it is usually the case that if a verb that allows *of* is followed by this preposition, the preposition is attached to the verb.

The other function performed by data extraction automata consists in identifying verbal groups. These types of automata extract features such as the tense, aspect, and mode of each main verb found in the text. During this process, the automata must also determine whether verbs such as *to have* function as auxiliaries or as main verbs. The following examples illustrate this potential ambiguity:

a. *She has a lot of money in the bank* => *has* is the main verb

b. *She has deposited a lot of money in the bank* => *has* is the auxiliary of *deposited*

3.3. Filter automata

The purpose of filter automata is to check the context of a sentence. Depending on the result of a filter automaton, one or several groups of third-level detection automata may be triggered.

Filter automata function like a large net looking for specific contexts. Each filter automaton is associated with a group of detection automata that are concerned with one specific problem.

For example:

```
AUTOMATON COMPL_V_FILTER_1

@[CAT=V,+COMPL_V,+PPREP]
^[IF='that'|(CAT=PRON,+WH_PRON)] (R1=>$R)
```

The success of this automaton means that a verb belonging to the list *COMPL_V* has been recognized and that either the inflected form *that* or a string from the *WH_PRON* syntactic category has been found somewhere further on in the sentence. In this case, success allows a series of error detection automata to run, all of which are related to the mistakes occurring with this type of verb.

3.4. Detection automata

There are two types of detection automata: some are triggered by specific filter automata, and the others are not. Except for this distinction, they have the same characteristics as the other automata.

As mentioned earlier, some detection automata are associated with lists containing linguistic information. These lists contain words that are already in the dictionary; this allows the system to access directly the information needed when a specific automaton concerning a lexical item included in one of these lists is running. Lists contain words that share similar semantic or syntactic features (e.g. *+HUMAN*) and may include specific syntactic properties for each item. For instance, the following list concerns dative verbs:

word1	feature1=value1	feature2=value2
& DAT_V	PREP_GOV	SHIFT
# *give*	*to*	+
# *allow*	*to*	+
# *show*	*to*	+
# *address*	*to*	-
# *attribute*	*to*	-

The automata used to check the structures of these verbs consist of one filter automaton that checks if the context contains one of these verbs, and then a series of detection automata that detect potential errors. The two filter automata below check if there is a verb that belongs to the dative verb list and whether that verb allows the "shift" transformation or not:

```
AUTOMATON DAT_V_FILTER_1        AUTOMATON DAT_V_FILTER_2

@[CAT=V,+DAT_V,+SHIFT]          @[CAT=V,+DAT_V,-SHIFT]
(R1=>$R)                        (R2=>$R)
```

If the automaton *DAT_V_FILTER_1* succeeds, then a series of detection automata (*SUFLU_PREP_DATV_1* and *SUFLU_PREP_DATV_2*) are launched. In the example below, the automaton checks whether a verb allowing a shift is built with a shift structure and with a superfluous preposition (e.g. **he gave to the secretary something*). If that is the case, an error message is produced.

```
AUTOMATON SUFLU_PREP_DATV_1

@[CAT=V, +DAT_V, CF=>$E, PREP_GOV=>$P]
[CAT=PREP,IF=$P]
<SN @[ +HUM]>
{
SCHEMA 3
MESSAGE   "Avec cette structure, le verbe \u\$E\v se construit sans préposition;
           nous vous suggérons de supprimer la préposition \u\$P\v."
}
```

The automaton matches the ill-formed sentence shown below:

**He gives to his friend a wonderful present*

The assignment of a value to the register *$P* means that the system looks for the value of the preposition for the verb in question and puts it in the register. This value is then used during the correction process.

If the automaton fails, the sentence is either correct or it contains another type of error. In that case, the next automaton, *SUFLU_PREP_DATV_2*, is activated. If it fails, then the system goes on to another series of automata.

If the first filter automaton fails but the second (*DAT_V_FILTER_2*) succeeds, the following detection automaton is activated. It checks whether a verb that doesn't allow the dative shift is shifted.

```
AUTOMATON STRUCT_DAT_V

@[CAT=V, +DAT_V,Prep_Gov=>$P, CF=>$E]
<NP @[+HUM] (FIRSTPOS=>$L,LASTPOS=>$R)>
<NP (FIRSTPOS=>$X,LASTPOS=>$Y>
```

The success of the automaton means that the following mistake has been detected:

**He attributes her friend wonderful qualities*

Another type of automaton can be used to check for the correct preposition. In the following example, the automaton checks if the preposition that is used is correct.

```
AUTOMATON CHECK_PREP

@[CAT=V, +DAT_V,Prep_Gov=>$P] <NP> [CAT=PREP,IF=>$R]
<NP> ($R/=$P)
```

Two registers are used here; with *$P*, the attribute *Prep_Gov* value is read from the verb list, and *$R* receives as a value the character string corresponding to the inflected form of the preposition. After the automaton has recognized a specific series of words, a condition is tested on one of the values. In a correct sentence, the preposition must be the same as the one that comes from the verb file. Thus, if they do not correspond (*$R* is different from *$P*), an error is detected, as in:

**He attributes wonderful qualities at his friends*

If the automaton fails, the sentence is considered correct and another automaton is tested. In this example, the series of detection automata triggered by the filter automata is quite small. In other cases (report verbs, for example), the series can be larger in order to look for every possible mistake.

4. Conclusion

It is clear that our approach is less efficient in detecting errors linked with complex grammatical structures. However, an important advantage is that the error detection rules contain enough relevant information within a single framework, such as basic syntactic structures and features specific to errors. In addition, the three execution levels allow us to organize the data in a way that the error detection automata can operate with a higher degree of confidence. This, in turn, greatly reduces the probability of false alarms.

5. Bibliography

CORNU, E. (1992): *The Importance of Linguistic Theories in Grammar Checking*, Workshop on natural language processing: first and second language correction of written texts, SGAICO, Neuchâtel.

GROSS M. (1975): *Méthodes en syntaxe*, Paris, Hermann.

KÜBLER N. (1992): "Verbes de transfert en français et en anglais", *Linguisticae Investigationes*, 16(1), 61-97.

VOSSE, T. (1992): "Detecting and correcting morpho-syntactic errors in real texts", *Proceedings of the Third Conference on Applied Natural Language Processing*, ACL, Trento.

WOODS, W.A. (1970): "Transition network grammars for natural language analysis", *CACM*, 13(10), 591-606.