Tools for locating noun phrases with finite state transducers.

Jean Senellart
LADL (Laboratoire d'automatique documentaire et linguistique.)
Université Paris VII
2, place Jussieu
75251 PARIS Cedex 05
email: senella@ladl.jussieu.fr

Abstract

The processes of retrieving and describing information in full texts are very close, at least for the first user (developer). We describe here practical tools that allow construction of an FST database which describes and then to retrieve nominal phrases composed with a sequence denoting an occupation and a proper noun.

1 Introduction

To be able to retrieve information in a text, one must be able to describe in details the forms under which this information is expressed. For example, if we do not have the information that *the chancellor of the Exchequer* is *the British finance minister*, that *the shadow minister for Trade and Industry* is not a *minister*, but refers to *the British opposition* or that *the Christ's ministry* has nothing to do with *politics*, we will be misled in our search. We are thus talking about linguistic "information". We have shown in (1998a), how descriptions of occupations and proper noun phrases could be processed by finite state transducer (FST). In a French daily newspaper, one sentence out of two contains such a noun phrase, thus, one cannot attempt to parse texts without recognizing such sequences. The recognition at this stage, should be accurate: *minister of finance* and *the minister, Mr Jones, for Scottish affairs* are well-formed nominals, but *minister for finance* is not. The syntactic and semantic description relies on the same basis: we must be able to fully enumerate such nominals.

We can find many formalisms proved to be powerful enough to describe such or such natural language phenomenon, but the real problem is the linguistic description. Generally, few examples are given, and it is assumed that formalisms will accommodate a completed database. The reasons of this situation are several: first, the description stage is considered as trivial; however when seriously attempted, many new theoretical problems appear. Second, if we are able to handle without problem a dozen of rules, when their number increase (to several thousand), processes become rapidly difficult to check and to understand, because interactions between rules are not treated.

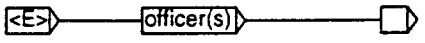

Figure 1: Starting point of the *officer* category

We present here practical methods that allow to create dynamically, to maintain and to debug a large database of finite state transducers. We will develop our example, and show, when one starts from scratch, how to construct a precise and large database. The text we use is one year of the *International Herald Tribune* (about 10 millions words).

2 Starting from scratch

Let us suppose we describe (in order to search) the semantic classes corresponding to the occupation *officer* (corresponding to the less ambiguous French equivalent word : *officier*). We start searching the sentences containing only the word *officer*with automaton of figure 1. In our corpus, 85 occurrences are found; the concordance is then sorted according to right contexts:

senior cabinet	officer	after an embarrassing
Gere in "An	Officer	and a Gentleman";
as a paratroop	officer	and had held
chief operating	officer	and make available
former Marine	officer	and National Security
the executive	officer	asks from topside.
African loan	officer	at one leading
a four star	officer	based in Hawaii
a former police	officer	convicted of murder

We first observe that the meaning *army* is mixed with the meanings *chief executive*, *loan* or *chief operating officer*. These meanings are not relevant at this stage. The left con-

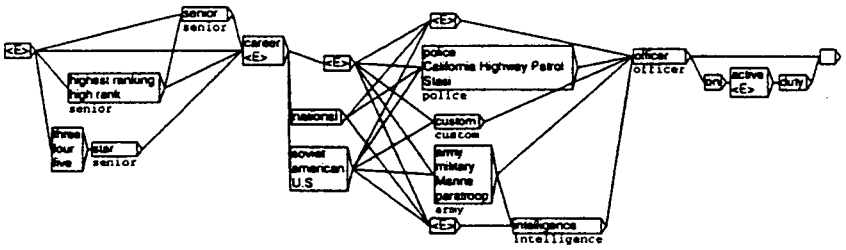

Figure 3: More complex classification

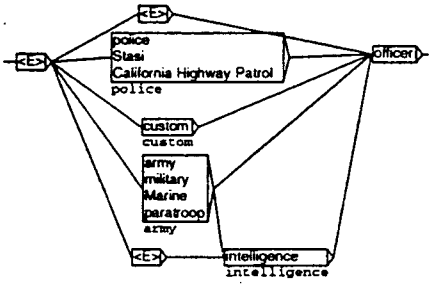

Figure 2: first *Officer* classification

text seems more interesting than the right context for desambiguation. So, we sort the concordance by left contexts, and extract all different adjectives or nouns that make explicit the occupation of *army*, *intelligence*, *police*, *Marine*... We group these modifiers in four semantic categories *intelligence*, *army*, *custom* and *police*. Depending on the need, this grouping could be made more precise. Then we construct transducer incorporating these modifiers, and giving as output of each sequence, the associated class (figure 2). Notice that two categories could be combined: *army* and *intelligence*.

We find the same 85 occurrences, but about one third make more precise the word *officer*:

the highest ranking	army officer	killed in more
the French	Army officer	who commands
engineer and	army officer	who served as
a senior	intelligence officer	who defected
a national	intelligence officer	, George Kolt,
by an	intelligence officer	. Mr. Inman's
the former	Marine officer	and National
As a Soviet	military intelligence officer	in London
ranking American	military officer	on active duty
highest ranking U.S	military officer	to visit Vietnam
a former	military officer	with the vision

In the same way, looking at the alphabetically sorted left and right context distinguishes new modifiers: to the right *on active duty*, and to the left *Soviet*, *American*, *U.S*, ... , *former* and *highest ranking*. Adding these possibilities, plus equivalent forms that come instantly[1] to mind, we obtain the transducer of figure 3. The number of different path of this automaton is now 3,348: whereas the number of lines of concordances examined is roughly 20. We could go on, but the size of the automaton becoming larger, it is more and more difficult to handle it on one screen. Moreover, semantically, we clearly see that some categories (e.g. nationality) will become far too important to be represented as one graph. It must be clustered.

3 Partitioning (manual clustering.)

Nationality is a homogeneous semantic category, composed of simple words, or compounds (*South Korean*), it can easily be stored in a list. We begin to construct it, starting with the three nationalities found: *American*, *Soviet*, and *French*. To be compatible with the search engine described in (1998b), an output is also associated with these words: for example the nationality itself, as in:

`American/American, French/French, Soviet/Soviet`

We must also construct a list of countries as we find productive forms such as *U.S military officer*, *France's officer*... The list of countries is thus initiated from the few countries we find instantly, or we can think of. To use lists, we define special nodes in the automaton (gray nodes) containing the name of the dictionary. We show in the automaton of figure 4 an example of such use.

In fact, we can localize other homogeneous sub-category as in automaton of figure 3: *senior*, *high ranking*, *four star*... indicate the rank of the officer. Here, it is more difficult to gather them into a single list, as their combinations are not trivial, we thus construct a

[1] At this stage, we do not pretend to be complete, we are only building the main structure.

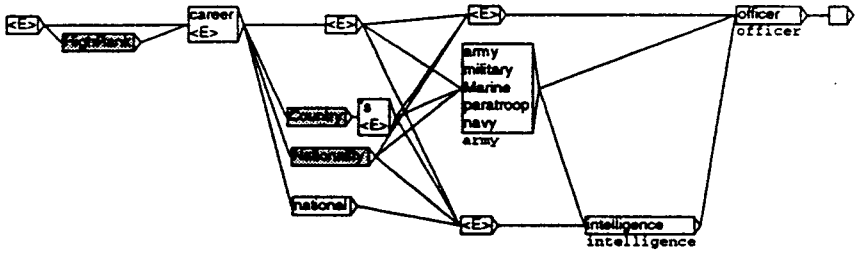

Figure 6: Automaton with different level

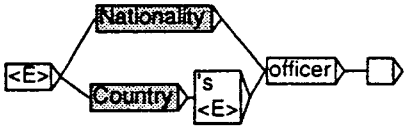

Figure 4: Example of dictionary use

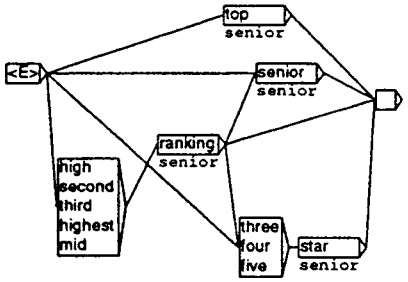

Figure 5: High rank sub-automaton

sub-automaton named *HighRank* (cf automaton 5). All the paths of this automaton combined with *officer* correspond approximatively to the *senior officer* rank. It will be sufficient for the search engine. Finally, we specialize our initial automaton to only military officers, as we clearly see that the rank and others modifiers associated with the *police* category will probably not be the same that those associated with the *military* category. We then create the *military officer* automaton (figure 6). Two main ideas should be kept in mind about clustering methodology: - homogeneous semantic sub-categories in an automaton should be put in a sub-automata, and - an automaton containing two semantic notions that we can separate, should be transformed into a more simple automaton containing only one semantic category. This clustering has many advantages: first, it allows a better readibility of automata, second, it factorizes homogeneous semantic units that we could use for other purposes. And third, it is very useful for processing, as we will show later.

Now we have established the main structure, we must complete the different automata and dictionaries. We show in the next section, a general method that permits to quickly enrich[2] the database.

4 Use of variables

The automaton of figure 6 represents the largest structure we found around the word *officer*. The initial goal was to construct nominals of the *army officer* semantic category. Such categories certainly contain sequences not containing at all the base word *officer* (Synonyms, hyponyms, or any semantically related nouns). To find such words, we have nevertheless now a important clue: their right and left contexts are close to the context of the word *officer*. We replace the word *officer* by a variable in the automaton of figure 6. A variable is a word or a sequence of words. We find on the same corpus 1,239 occurrences. The concordance we obtain is sorted alphabetically on that variable. This allows us to locate very quickly (in less than one minute) to locate new terms that should be put in parallel with the word *officer*. We give here an extract:

the East German	intelligence	chief at the time.
Mr Park an	army	general who seized
A retired	army	general, Mr. Chung
Communist	army	general, sent his
Croatian	Army	helicopters directed fighter
the former	intelligence	official as his
former top	intelligence	official named to
Korea a Japanese	intelligence	official said.
which	military	officials and federal
be determined	military	officials said.
the UN	military	officials said.
for them",	military	officials said.
police and	military	officials.

[2] And more and more quickly when the size of the text increases.

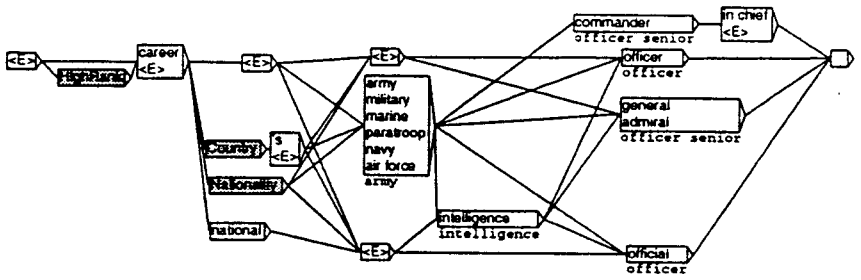

Figure 7: Completion of the *officer* column

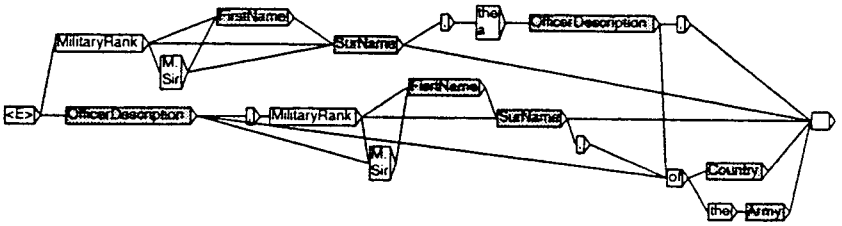

Figure 8: Full nominal with both occupation and proper noun part

We also observe new right and left contexts relevant to our description (but we do not take them into account here, in order to keep our steps in sequence). With this new words, we complete the automaton and obtain the automaton of figure 7. To maintain clarity in the automaton, we always put in a same column, equivalent units or term. For that reason, *officer* is vertically aligned with *general*, *official*... *Country* is vertically aligned with *national* and *Nationality*, etc. When we add new boxes, we add the necessary transitions we can think of. Sometimes, choices are difficult: for example, can we find *career general* ? In those cases, we permit it by default. The final automaton may recognize sequences that are not stylistically or semantically perfect, but that are always well-formed, and their presence will not generate any noise.

We skip a few stages to arrive to the proper noun description. Contrary to the French case, where the full name is always put in an external apposition of the occupation noun phrase, in English both can be mixed, we find in our corpus for example:

General Jean Cot of the French Army said the idea of...

... the French commander of troops there. General Jean Cot,

General Jean Cot, the French Army officer who commands the

And as *General John Shalikashvili* has said, the Partnership...

We describe these four different structures (left and right apposition, proper nouns and occupations mixed, and only proper noun) in a simplified way[3] (to keep small automata for the presentation) by the automaton of figure 8. This automaton contain *FirstName* and *SurName* boxes. These two dictionaries are initially empty, they need to be filled, and we easily imagine that their size will quickly become very large. To that aim, we replace them by variable. In that case, we can make more precise the morphological form of these variables. (As a first approximation, let us say that they are both single words with a capital first letter). We apply this automaton to the corpus, and obtain a list of 582 occurrences. These occurrences are sorted according to the *FirstName* and *SurName* variable, and we only need to confirm these terms to be actual *First* and *SurName*. A first draft for the both dictionary is thus constructed: 15 entries of *FirstName* and 67 of *SurName*. The whole construction is dynamic: once we have some proper nouns, we seek their contexts, and find new structures for the *officer* automaton. These new structures allows us to obtain new proper nouns, new countries, new nationalities... This is the bootstrap effect.

[3] In particular, this automaton recognizes *the British General of France* !

We have built in one hour, a large coverage automaton (more than 200,000 different paths, without counting proper noun combinations) for this mere semantic category. The final number of recognized items is 863 (to compare with the 85 that we started from, and that were in majority used in another context (economic)). The number of the potential recognized sequences, as compared to the effectively recognized sequences is a guarantee that, on a totally new corpus, we will obtain instantly a good coverage and new potential *SurName* and *FirstName* (and this, with a rate proportional to the size of existing dictionaries).

5 Software

The needed tools for such a methodology are:

1. **Graph editor.** The first tool (included in INTEX (1994)) is a graphic graph editor. This editor allows to create, copy and align parts of them. It handles input and output of transducers. It allows, by simple clicking, to open a sub-automaton.
2. **Index-based parsing algorithm.** A key feature of the text-based automaton construction, is in the possibility to have immediate concordances. It is hardly thinkable parse sequentially a 10 million word corpus. Moreover, with many levels of automata (sometimes more than 20), the size of the developed main automaton becomes quickly huge, that we cannot re-compute for each concordance. Thus, we have chosen to index each sub-automaton independently, with a dependency graph (cf below) like *makefile*, we only re-compute, the modified graph, and those depending on it. The index parsing algorithm is described in (Senellart, 1998a). It allows us, to obtain concordances on the whole corpus, in a mean time less than 1s on a average personal computer.
3. **Concordance manager.** We have shown during all along this paper, that the way the concordance are sorted has a great importance. Under INTEX, we can sort concordances according to the recognized sequence, to the right, or left context, and any combination of the three parameters. Moreover, when we put variables in the automaton, we must are able to validate recognized sequence linked to the variable in a click.
4. **Debugging tools.** Maintaining a large number of automata is not simple: some automata depend on others of which, the exact name and the exact function must be recalled. Exactly as in compiling programs with a large number of source files, we can compute and represent (graphically) the dependencies between the different automaton. For example the graph of figure 9, represents the dependent sub-automaton used in the *Officer* automaton, and the same for each of the sub-automaton. The depth in this graph is four.

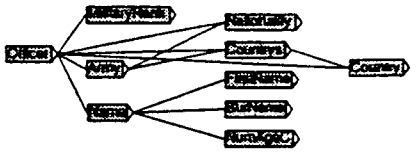

Figure 9: Dependency graph of *Officer* automaton

6 Conclusion

The simplicity with which we created the *Officer* automaton is not based on new theoretical algorithms or formalism. It is totally based on useful tools elaborated at LADL. I have built for French a database of large coverage for proper nouns and occupations. This database includes more than 200 kinds of different main automata, proper noun dictionaries of towns, surnames, each with several thousand entries. This is a practical result, and I think that this is the case of almost all linguistic phenomena. We can describe them theoretically, but in practice the number of the different cases, and the links between them make impossible to list them without appropriate tools. The bootstrap method we present is a very general methodology, that make use of the text in an efficient way, to construct a local grammar.

References

J. Senellart. 1998a. Fast pattern matching in indexed texts. Being published in TCS.

J. Senellart. 1998b. Locating noun phrases with finite state transducers. In *Proc. of ACL-COLING'98.*

M. Silbertzein. 1994. Intex: a corpus processing system. in coling proceedings. In *Proc. of COLING'94.*