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

We present a method for constructing, maintaining and consulting a database of proper nouns. We describe noun phrases composed of a proper noun and/or a description of a human occupation. They are formalized by finite state transducers (FST) and large coverage dictionaries and are applied to a corpus of newspapers. We take into account synonymy and hyperonymy. This first stage of our parsing procedure has a high degree of accuracy. We show how we can handle requests such as: 'Find all newspaper articles in a general corpus mentioning the French prime minister', or 'How is Mr. X referred to in the corpus; what have been his different occupations through out the period over which our corpus extends?' In the first case, non trivial occurrences of noun phrases are located, that is phrases not containing words present in the request, but either synonyms, or proper nouns relevant to request. The results of the search is far better than than those obtained by a key-word based engine. Most answers are correct: except some cases of homonymy (where a human reader would also fail without more context). Also, the treatment of people having several different occupations is not fully resolved. We have built for French, a library of about one thousand such FSTs, and English FSTs are under construction. The same method can be used to locate and propose new proper nouns, simply by replacing given proper names in the same FSTs by variables.

1 Introduction

Information Retrieval in full texts is one of the challenges of the next years. Web engines attempt to select among the millions of existing Web Sites, those corresponding to some input request. Newspaper archives is another example: there are several gigabytes of news on electronic support, and the size is increasing every day. Different approaches have been proposed to retrieve precise information in a large database of natural texts:

1. Key-words algorithms (e.g. Yahoo): co-occurrences of the different words of the request are searched for in one same document. Generally, slight variations of spelling are allowed to take into account grammatical endings and typing errors.
2. Exact pattern algorithms (e.g. OED): sequences containing occurrences described by a regular expression on characters are located.
3. Statistical algorithms (e.g. LiveTopic): they offer to the user documents containing words of the request and also words that are statistically and semantically close with respect of clustering or factorial analysis.

The first method is the simplest one: it generally provides results with an important noise (documents containing homographs of the words of the request, not in relation with the request, or documents containing words that have a form very close to that of the request, but with a different meaning).
The second method yields excellent results, to the extent that the pattern of the request is sufficiently complex, and thus allows specification of synonymous forms. Also, the different grammatical endings can be described precisely. The drawback of such precision is the difficulty to build and handle complex requests.
The third approach can provide good results for a very simple request. But, as any statistical method, it needs documents of a huge size, and thus, cannot take into account words occurring a limited number of times in the database,

which is the case of roughly one word out of two, according Zipf's law[1](Zipf, 1932).
We are particularly interested in finding noun phrases containing or referring to proper nouns, in order to answer the following requests:

1. *Who is John Major?*
2. *Find all document referring to John Major.*
3. *Find all people, who have been French ministers of culture.*

With the key-word method, texts containing the sequence *'John Major'* are found, but also, texts containing *'a UN Protection Force, Major Rob Anninck'*, *'P. Major'*, *'a former Long Islander, John Jacques'* and *'Mr. Major'*.
The statistical approach will probably succeed (supposing the text is large enough) in associating the words *John Major*, with the words *Britain*, *prime* and *minister*. Therefore, it would provide documents containing the sequence *'the prime minister, John Major'*, but also *'the French prime minister'* or *'Timothy Eggar, Britain's energy minister'* which have exactly the same number of correctly associated words. Such answers are an inevitable consequence of any method not grammatically founded.
M. Gross and J. Senellart (1998) have proposed a preprocessing step of the text which groups up to 50 % of the words of the text into compound utterances. By hiding irrelevant meanings of simple words which are part of compounds, they obtain more relevant tokens. In the preceding example, the minimal tokens would be the compound nouns *'prime minister'* or *'energy minister'*, thus, the statistical engine could not have misinterpreted the word *'minister'* in *'energy minister'* and in *'prime minister'*.
We propose here a new method based on a formal and full description of the specific phrases actually used to describe occupations. We also use large coverage dictionaries, and libraries of general purpose finite state transducers. Our algorithm finds answers to questions of types *1*, *2* and *3*, with nearly no errors due to silence, or to noise. The few cases of remaining errors are treated in section *5* and we show, that in order to avoid them by a general method, one must perform a complete syntactic analysis of the sentence.
Our algorithm has three different applications. First, by using dictionaries of proper nouns and local grammars describing occupations, it answers requests. Synonyms and hyponyms are formally treated, as well as the chronological evolution of the corpus. By consulting a preprocessed index of the database, it provides results in real time. The second application of the algorithm consists in replacing proper nouns in FSTs by variables, and use them to locate and propose to the user new proper nouns not listed in dictionaries. In this way, the construction of the library of FSTs and of the dictionaries can be automated at least in part. The third application is automatic translation of such noun phrases, by constructing the equivalent transducers in the different languages.
In section 2, we provide the formal description of the problem, and we show how we can use automaton representations. In section 3, we show how we can handle requests. In section 4, we give some examples. In section 5, we analyze failed answers. In section 6, we show how we use transducers to enrich a dictionary.

2 Formal Description

We deal with noun phrases containing a description of an occupation, a proper noun, or both combined. For example, *'a senior RAF officer'*, *'Peter Lilley, the Shadow Chancellor'*, *'Sir Terence Burns, the Treasury Permanent Secretary'* or *'a former Haitian prime minister, Rosny Smarth'*. For our purpose, we must have a formal way of describing and identifying such sequences.

2.1 Description of occupations

We describe occupations by means of local grammars, which are directly written in the form of FS graphs. These graphs are equivalent to FSTs with inverted representation (FST) (Roche and Schabes, 1997) as in figure 1, where each box represents a transition of the automaton (input of the transducer), and the label under a box is an output of the transducer. The initial state is the left arrow, the final state is the double square. The optional grey boxes, (cf figure 2), represent sub-transducers: in other words, by 'zooming' on all sub-transducers, we view a given FST as a simple graph, with no parts singled out. However, we insist on

[1] This is true whatever the size of the database is.

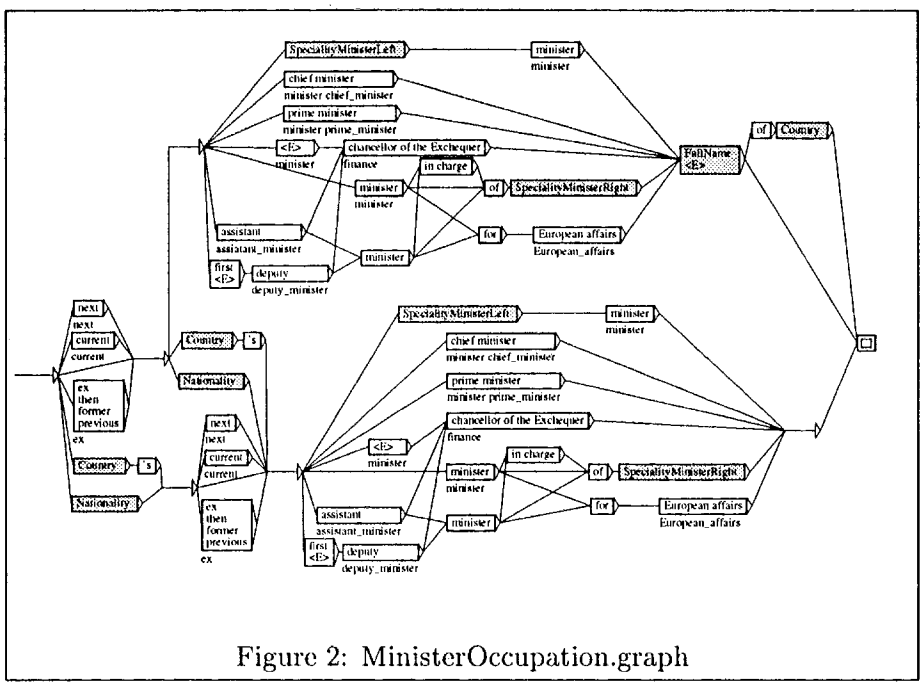

Figure 2: MinisterOccupation.graph

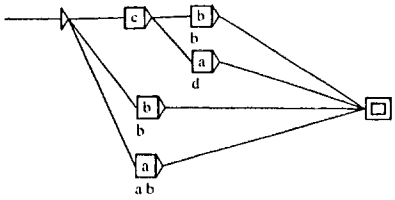

Figure 1: Formal example

keeping sub-FST automata, as they will be computed independently, and as they allow us to keep a simple representation of complex constructions. The output of a grey box, is the output of the sub-transducer. The symbol labeled **<E>** represents the void transition, and the different lines inside are parallel transitions. Such a representation is convenient to formulate linguistic constraints. A graph editor (Silberztein, 1993) is available to directly construct FSTs. In theory, such FSTs are more powerful than traditional FSTs[2].

In figure 1, the transducer recognizes the sequences **a**, **b**, **ca**, **cb**. To each of these *input* sequences, it associates an *output* noted $val(input)$. Here, $val(a) = \{ab\}$, $val(b) = \{b\}$, $val(c)$ is not defined as c is not recognized by the automaton. $val(ca) = \{d\}$, and $val(cb) = \{b\}$.

We define an ordering relation on the set of recognized sequences by a transducer $\mathcal{T}$, that is: $x \leq_{\mathcal{T}} y \Leftrightarrow \forall e \in val(x),\ e \in val(y)$. In our example, $b \leq_{\mathcal{T}} a$ and $b =_{\mathcal{T}} cb$ with derived equality relation.

We construct our transducer describing occupations in such a way that with this ordering[3] relation:

- Two sequences x, y are synonyms if and only if $x =_{\mathcal{T}} y$
- The sequence y is an hyponym of x (i.e. y is a x) if and only if $x \leq_{\mathcal{T}} y$.

The transducer in figure 2 describes[4] different sequences referring to the word *minister*. Sub-parts of the transducers `Country` and `Nationality` are given in figure 3 and 4.

By construction, all the sequences recognized are grammatically correct. For example, the variant of *minister of European affairs*: *minister for European affairs* is recognized, but not

[2] If a sub-automaton refers to a parent automaton, we will be able to express context dependent words such as $a^n b^n$.

[3] The equality relation $=_{\mathcal{T}}$ and the strict comparison are directly deduced from $\leq_{\mathcal{T}}$ definition.

[4] For the sake of clarity, it is not complete, for example it doesn't take into account regional ministries as in USA or in India. It doesn't represent either the sequence *deputy prime minister*. Moreover, a large part could be factorized in a sub-automaton.

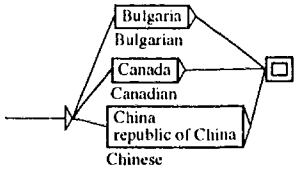

Figure 3: Country.graph

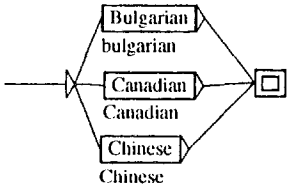

Figure 4: Nationality.graph

French minister for agriculture. The output of the transducer is compatible with our definition of order:

- val(France's culture minister)
$=_{\mathcal{T}}$ $\{French, minister, Culture\}$
$=_{\mathcal{T}}$ val(culture minister of France)
$\geq_{\mathcal{T}}$ val(French minister)
- *'chancellor of the Exchequer'*$=_{\mathcal{T}}$*'finance minister'*
- *'prime minister'*$\geq_{\mathcal{T}}$*'minister'* i.e. a prime minister is a minister but *'deputy minister'*$\not\geq_{\mathcal{T}}$*'minister'* i.e. a deputy minister is not a minister.

Reciprocally, given an output, it easy to find all paths corresponding to this output (by inverting the inputs and the outputs in the transducer). This will be very useful to formulate answers to requests, or to translate noun phrases: the "natural language" sequences corresponding to the set $\{minister, French\}$ are : "French minister" or "minister of France". We will note $val^{-1}(\{minister, French\})$ = {'french minister', 'minister of France'}.

2.2 Full Name description

The full name description is based on the same methodology (cf. figure 5), except that the boxes containing `<PN:FirstName>` and `<PN:SurName>` represent words of the proper nouns dictionaries. The output of this transducer is computed in a different way: the output is the surname, the firstname if available, and the gender implied either by the firstname, or by the short title: Mr., Sir, princess, etc....

3 Handling requests: a dynamic dictionary

In order to instantly obtain answers for all requests, we build an incremental index of all matches described by the FST library. At this stage, the program proposes new possible proper nouns not yet listed, they complete the dictionary. Our index has the following property: when an FST is modified, it is not the whole library which is recompiled, but only the FSTs affected by the modification. We now describe this stage and show how the program consults the index and the FST library to construct the answer.

3.1 Constructing the database

In (Senellart, 1998), a fast algorithm that parses regular expressions on full inverted text is presented. We use such an algorithm for locating occurrences of the FSTs in the text. For each FST, and for each of its occurrences in the text, we compute the position, the length, and the FST associated output of the occurrence.

This type of index is compressed in the same way entries of the full inverted text are. This choice of structure has the following features:

1. There is no difference of parsing between a 'grey (autonomous) box' and a 'normal one'. Once sub-transducers have been compiled, they behave like normal words. Thus, the parsing algorithms are exactly the same.
2. A makefile including dependencies between the different graphs is built, and modifications of one graph triggers the re-compilation of the graphs directly or indirectly dependent.
3. This structure is incremental: adding new texts to the database is easy, we only need to index them and to merge the previous index with the new one by a trivial pointer operation.

A description of a whole noun phrase is given made by the graph of figure 6.

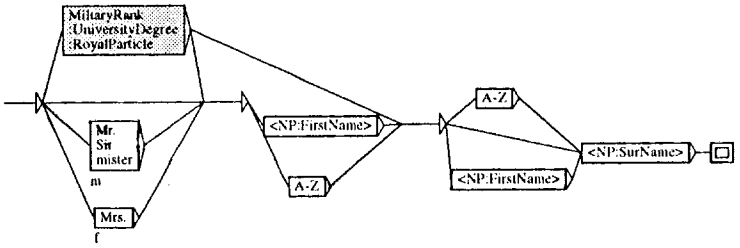

Figure 5: FullName.graph

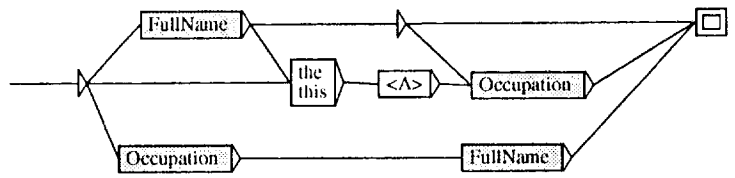

Figure 6: NounPhrases.graph, the <A> label stands for any adjective. (Information of the general purpose dictionary)

We use a second structure: a *dynamic proper noun dictionary* $\mathcal{D}$ that relies on the indexes of **Occupation.graph** and **FullName.graph**. $\mathcal{D}$ is called 'dynamic' dictionary, because the information associated to the entries depend on the locations in the text we are looking for. The algorithm that constructs $\mathcal{D}$ is the following:

1. For each recognized occurrence we associate O_1 which is the output of **FullName.graph** and the output O_2 of the **Occupation.graph** (see section 4 for examples).
2. If O_1 is not empty, find O_1 in $\mathcal{D}$: that is, find the last e in $\mathcal{D}$ such that $O_1 \leq_\tau e$. - If there is none, create one : i.e. associate this FullName with the occupation O_2 and with the current location in the text.
   - If there exists one, and its occupation is compatible with O_2 then add the current location to this entry. Or else, create a new entry for O_1 (eventually completed by the information from e) with its new occupation O_2, and pointing to the current location in the text.
3. If O_1 is empty: the noun phrase is limited to the occupation part. Find the last entry in $\mathcal{D}$ compatible with O_2, and then add the current location to the entry.

A detailed run of this algorithm is given in section 4.

3.2 Consulting the database

Given a request of type 1: *Who is P*. We first apply the **NounPhrases.graph** to P. If P is not recognized, the research fails. It it is recognized, we obtain two outputs O_1 and O_2 as previously mentioned. For this type of request O_1 cannot be empty. So we look in $\mathcal{D}$ for the entries that match O_1 (there can be several, often when the first name is not given, or given by its initial). Then, we print the different occupations associated to these entries.
Given a request of type 2: the result is just an extension of the previous case: once we have found the entries in $\mathcal{D}$, we print all positions associated in the text.
Given a request of type 3, the method is different: we begin by applying the **NounPhrases.graph** to P. In this case, O_1 is empty. Then we look up the entries of $\mathcal{D}$, and check if at some location of the text, its occupation is compatible with the occupation of the request.

4 Examples of use

Consider the following chronological extract of French newspaper :

1- M. Jack Lang, ministre de l'éducation nationale et de la culture,
2- Chargé le 7 avril 1992 par M. Lang de réfléchir aux conditions de
3- M. Jack Lang a lancé dimanche soir à la télévision l'idée d'impliquer
4- Commentant l'action du ministre de la culture, le premier adjoint
5- En définitive l'idée de M. Lang apparaît comme un rêve !
6- Le directeur de l'American Ballet Theater, Kevin McKenzie :
7- M. Lang présente son projet de réforme des lycées prévoyant
8- Tous, soutenez la loi Lang, par distraction, de temps en temps, ici
9- M. Jack Lang, maire de Blois, a officiellement déposé sa
10- Sortants : Michel Fromet, suppléant de Jack Lang, se représente
11- De son coté, Carl Lang, secrétaire général du Front national, a
12- et Jack Lang, ancien ministre de l'éducation nationale et de la culture,
13- l'ancien ministre, Jack Lang, et son successeur, Jacques Toubon,
14- Jack Lang, maire de Blois et ancien ministre,
15- ..., le nouveau ministre de l'éducation nationale, Jacques Toubon,

– At the beginning $\mathcal{D}$ is empty.
– We read the sentence 1:
$O_1 = \{m, Jack, Lang\}$,
$O_2 = \{minister, education, culture\}$. There is no entry in $\mathcal{D}$ corresponding to O_1, thus we create in $\mathcal{D}$ the following entry :

```
SurName=Lang,FirstName=Jack,Gender=m,
(Line 1 Occupation=minister,education,culture)
```

– We read the sentence 2: $O_1 = \{m, Lang\}$. O_1 matches the only entry in $\mathcal{D}$, and moreover as O_2 is empty: it also matches the entry. Thus we add the line 2, as a reference to this first entry.

```
SurName=Lang,FirstName=Jack,Gender=m,
(Line 1,2 Occupation=minister,education,culture)
```

...
– At the end of the processing, $\mathcal{D}$ equals to:

```
SurName=Lang,FirstName=Jack,Gender=m,
(Line 1,2,3,4,5,7 Occupation=minister,education,culture)
(Line 9,12,13,14 Occupation=mayor,Blois)
SurName=Fromet,FirstName=Michel,Gender=m,
(Line 10 Occupation=minister_deputy,education,culture)
SurName=Lang,FirstName=Carl,Gender=m,
(Line 11 Occupation=head_party,FN)
SurName=Toubon,FirstName=Jacques,Gender=m,
(Line 13,15 Occupation=minister,education)
```

Now if we search all parts of the text mentioning the *minister of culture*, we apply **NounPhrases.graph** to this request and we find $O_1 = \{\}$, $O_2 = \{minister, culture\}$. The only entries in $\mathcal{D}$ matching O_2 correspond to the lines 1,2,3,4,5,7,13,15. This was expected, lines referring to the homonym of Jack Lang have not be considered, nor line referring Jack Lang designated as the mayor of Blois.

5 Remaining errors

Some cases are difficult to solve, as we can see in the sentence: *In China, the first minister has ....* The first phrase of the sentence: *In China* is an adverbial, and could be located everywhere in the sentence. It could even be implicit, that is, implied by the rest of the text. In such a case, a human reader will need the context, to identify the person designated. We are not able, to extract the information we need, thus the result is not false, but imprecise.
Another situation leads to wrong results: when one same person has several occupations, and is designated sometimes by one, sometimes by another. To resolve such a case, we must represent the set of occupations that are compatible. This is a rather large project on the 'semantics' of occupation.
Finally, as we can see if figure 6, a determiner and an adjective can be found between the Fullname part, and the Occupation part. In most case, it is something 'this', or 'the', or 'the well-known', or 'our great', and can be easily described by a FST. But in very exceptional case, we can find also complex sequences between the Fullname part, and the Occupation part. For example: 'M. X, who is since 1976, the prime minister of ...'. In this case, it is not possible, in the current state of the developpment of out FST library, to provide a complete description.

6 Building the dictionaries and the database

The results of our approach is in proportion the size of the database we use. We show that using variables in FSTs, and the bootstrapping method, this constraint is not as huge it seems. One can start with a minimal database and improve the database, when testing it on a new corpus. Suppose for example, that the database is empty (we only have general purpose dictionaries). We ask the system to find all occurrences of the word 'minister', the result has the following form of concordance:

```
The Israeli foreign minister, Shimon Peres, said the intern
the Russian foreign minister, Andrei V. Kozyrev, was likely
Berlusconi as prime minister, but ty issue ought to be the
couri, as the Greek minister of culture, thought up the ide
```

```
first deputy prime minister, Oleg Soskovets; Moscow has pl
```

On this small sample, we see that it is interesting to search the different occurrences of "(<A>+<N>) <minister>" and we obtain the list: `prime, foreign, Greek, finance, trade, interior, Cambodian, ...`

We separate automatically in this list, words with uppercase first letter and lowercase words. This provide a first draft for a Nationality dictionary (on a 1Mo corpus, we obtain 234 entries (only with this simple rule). The list is then manually checked to extract noise as 'Trade minister of ...'. We then sort the lowercase adjective and begin to construct the minister graph. We find directly 23 words in the subgraph "SpecialityMinisterLeft", plus the special compounds "prime minister" and "chief minister". We then apply this graph to the corpus and attempt to extend occurrences to the left and to the right. We notice that we can find a name of country with an "'s"just to the left of the occupation, and thus we catch potential names of country with the following request : "[A-Z][a-z]*'s :MinisterOccupation", where `[A-Z] [a-z]*` is any word beginning with an uppercase letter. This is an example of variable in the automaton. Pursuing this text-based method and starting from scratch, in roughly 10 minutes, we build a first version of the dictionaries: Country (87 entries) and Nationality (255 entries), Firstname (50 entries), Surname (47 entries), plus a first version of the MinisterOccupation and the FullName FSTs... The graphical tools and the real-time parsing algorithms we use are crucial in this construction.

Remark that the strict domain of proper noun cannot be bounded: when we describe occupations in companies, we must catch the company names. When we describe the medical occupation, we are lead to catch the hospital names... Very quickly the coverage of the database enlarges, and dictionaries of names of companies, of towns must be constructed. Concerning the French, in a newspaper corpus, one word out of twenty is included in a occupation sequence: i.e. one sentence out of two in our corpus contained such noun phrase.

7 Conclusion

In conclusion, we have developed this system first for the French language, with very good results. It partially solves the problem of Information Retrieval for this precise domain. In fact the "occupation" domain is not closed: is a "thief" an occupation ? To avoid such difficulties, and in order to reach a good coverage of the domain, we have described essentially institutional occupations. We know full well that if we want to be precise, a very deep semantic description should be done: for example, it is not sure that we can say a "prime minister" of France is comparable with a "prime minister" of UK ? One of strength of the described system is that it enables us to gather information present in different locations of the corpus, which improves punctual descriptions. Another interest of having such representations for different languages is a possibly automatic translation for such noun phrases. The output of the source language will be used in the target language of FSTs to identify paths having the same output, hence the same meaning. We are working to adapt the representation to other languages, such as English and the challenge is not only to repeat the same work on another language, but to keep the same output for two synonyms in French and English, which is not easy, because some occupations are totally specific to a language.

Our method is totally text-based, and the appropriate tools allow us to enrich the database progressively. We strongly believe that the complete description of such noun phrases is needed (for all needs: IR, translation, syntactic analysis...), and our interactive method which is quite efficient to this aim.

References

M. Gross and J. Senellart. 1998. Nouvelles bases pour une approche statistique. In *JADT98*, Nice, France.

E. Roche and Y. Schabes, eds. 1997. *Finite state language processing.* MIT Press.

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

M. Silberztein. 1993. *Dictionnaires électroniques et analyse automatique de textes.* Masson.

Zipf. 1932. *Selected Studies of the Principle of Relative Frequencies in Language.* Cambridge.

Résumé

Nous présentons une méthode permettant de construire et de maintenir semi-automatiquement (avec vérification manuelle) une base de donnée de noms propres associés à des professions. Nous décrivons exactement les groupes nominaux composés d'un nom propre et/ou d'une séquence décrivant une profession. La description est faite à l'aide de transducteurs finis et de dictionnaires d'usage courant à large couverture. Nous montrons ensuite comment nous pouvons traiter des requêtes du type: 'Quels sont les articles dans le corpus mentionnant le premier ministre français ?', ou 'Comment Mr. X est décrit, quelles ont été ses différentes professions au cours de la période couverte par notre corpus ?' Dans le premier cas, des occurrences non triviales sont trouvées: par exemple, celles ne comportant pas de mots de la requête, mais des constructions synonymes ou même le nom propre associé à cette profession par des occurrences précédentes. Le résultat d'une telle recherche est donc largement supérieur à ce qu'on obtient par mots-clefs, ou par association statistique. Mis à part quelques cas d'homonymies, toutes les réponses sont exactes, certaines peuvent être imprécises. Nous avons construit pour le français, une telle bibliothèque de transducteurs finis, et un travail analogue est en cours pour l'anglais. D'une manière aussi importante que le formalisme utilisé, nous montrons comment l'utilisation d'une interface conviviale de construction de graphe rend possible une telle démarche. Nous montrons comment utiliser ces mêmes transducteurs pour compléter les dictionnaires de noms propres, et donc d'avoir de meilleurs résultats. Nous montrons enfin comment de tels transducteurs peuvent être utilisés pour traduire les termes décrivant des professions.