Frequencies of occurrence of entries and subcategorization frames in *LGLex* lexicon with IRASubcat

Elsa Tolone[1], **Romina Altamirano**[1]
[1]FAMAF, Universidad Nacional de Córdoba
elsa.tolone@univ-paris-est.fr, romina.altamirano@gmail.com

Abstract: We present a method for enlarge a lexicon (with frequencies information), that is useful for parsing and others NLP applications. We show an example enlarging the verbal *LGLex* lexicon of French [13], using several corpora extracted from the evaluation campaign for French parsers Passage [9]. To do that, we use the results of the FRMG parser [11] with IRASubcat [2], a tool that automatically acquires subcategorization frames from corpus in any language and that also allows to complete an existing lexicon. We obtain the frequencies of occurrence for each input and each subcategorization frame for 14,068 distinct lemmas.

Keywords: Lexicon-Grammar, syntactic lexicon, French lexicon, subcategorization, frequency of occurence.

1. Introduction

The volume of textual information available today makes the manual processing of information impossible, therefore intelligent automatic processing becomes a necessity. In this article we describe how to improve a French lexicon using a tool for automatic acquisition of subcategorization frames from corpora.

The overall objective is the natural language understanding, improved basic tools and resources for automatic analysis of French. There are different applications, from information extraction to support second language learning. The syntactic lexicons are basic resources in most advanced tasks of Natural Language Processing (NLP), since most of the systems that have some ability to understand natural language required syntactic and semantic knowledge for each predicate (verb, noun or adjective).

Lexicon-Grammar tables are currently one of the major sources of syntactic lexical information for the French language [6]. Moreover, several Lexicon- Grammar tables exist for other languages, such as Italian, Brazilian Portuguese, Modern Greek, Korean, Romanian, and others.

We improved the Lexicon-Grammar tables to make them usable in various NLP applications, in particular parsing [13]. So we genrated a French syntactic lexicon for verbs, nouns playing the predicative role frozen expressions including verbal and adjectival idioms, and adverbs from the Lexicon- Grammar tables, called *LGLex* [5].

Then, we converted the verbs and predicative nouns into the Alexina framework, that is the one of the Le*fff* lexicon (*Lexique des Formes Fléchies du Français* – Lexicon of French inflected form) [10], a large-coverage morphological and syntactic lexicon for French.

This enables its integration in the FRMG parser (French MetaGrammar) [11], a large-coverage deep parser for French, based on Tree Adjoining Grammars (TAG), that usually relies on the Le*fff*. We evaluated the FRMG parser with the resulting lexicon on the reference corpus of the evaluation campaign for French parsers EASy (*Évaluation des Analyseurs Syntaxiques du français*) [15] and Passage (*Produire des Annotations Syntaxiques à Grande Échelle*) [16], using a component integrated in the processing chain of FRMG which eliminates the ambiguity to consider only one analysis per sentence.

In this article we present a method for enlarge a lexicon (with frequencies information) [14], that is useful for parsing and others NLP applications. We show an example enlarging the verbal *LGLex* lexicon of French, using several corpora extracted from the evaluation campaign for French parsers Passage [9]. To do that, we use the results of the FRMG parser with the IRASubcat tool [2]. First we describe all lexical resources used in section 2: i.e, the Lexicon-Grammar tables, the *LGLex* lexicon and the FRMG parser. Then, we present the tool IRASubcat in section 3. In section 4, we show how we used this tool with verbal *LGLex* lexicon, explaining the experiment. We finish by explaining the work performed and the next steps in section 5.

2. Lexical resources

First we describe what are the Lexicon-Grammar tables and we converted them into the *LGLex* lexicon. Then, we describe the conversion into *LGLex*- Le*fff* syntactic lexicon to integrate them in the FRMG parser. Finally, we present the format of a corpus processed by the FRMG parser.

2.1. The Lexicon-Grammar tables

Lexicon-Grammar tables are currently one of the major sources of lexical and syntactic information for the French language. Their development was initiated as early as the 1970s by M. Gross[1], at the LADL (*Laboratoire d'Automatique Documentaire et Linguistique*) [6, 7], and then the LIGM (*Laboratoire d'Informatique de Gaspard-Monge*) at University Paris-Est in France [3, 8].

Lexical information is represented in the form of *tables*. Each table puts together elements of a given category (for a given language) that share a certain number of *defining features*, which usually concern subcategorization. These elements form a *class*.

Tables are represented as matrices: each row corresponds to a lexical item of the corresponding class; each column lists a feature that may be valid or not for the different members of the class; at the intersection of a row and a column, the symbol + (resp. –) indicates that the feature corresponding to the column is valid (resp. not valid) for the lexical entry corresponding to the row.

The resources described in this paper correspond to the Lexicon-Grammar tables of simple verbs, in which previously implicit features have been made explicit[2] for more convenient use in NLP. All tables are fully available[3] under a free license (LGPL-LR).

2.2. The *LGLex* syntactic lexicon

The current version of French Lexicon-Grammar tables enables the use of their lexical data in NLP tools [12]. To this end, we converted the tables into an exchange format, based on the same linguistic concepts as those handled in the tables. This conversion is based on *LGExtract*: a generic tool for generating a syntactic lexicon for NLP from the Lexicon-Grammar tables [5]. It relies, first off, on a global table of classes in which we added the missing features and, second, on a single extraction script including all operations related to each feature to be performed for all tables.

Thanks to *LGExtract*, we generated a French lexicon for NLP from all Lexicon-Grammar tables and for most lexical-grammatical categories: verbs, predicative nouns, idioms and adverbs. This syntactic lexicon is named *LGLex* [5, 13]. It is manually evaluated and freely available4 under the LGPL-LR license in both plain text and XML format.

LGLex is currently composed of 13,895 verbal entries including 5,738 distinct entries (from 67 tables)[4].

2.3. The *LGLex*-Le*fff* syntactic lexicon

The Le*fff* is a freely available and large-coverage morphological and syntactic lexicon for French [10][5]. It relies on the Alexina framework for the acquisition and modeling of morphological and syntactic lexicons. To represent lexical information, an Alexina lexicon relies on a two-level architecture:

– the *intensional lexicon* associates (among others) an inflection table and a canonical subcategorization frame with each entry and lists all possible redistributions from this frame;

– the *compilation* of the intensional lexicon into an *extensional lexicon* builds different entries for each inflected form of the lemma and every possible redistribution.

We converted the verbs and predicative nouns of *LGLex* lexicon into the Alexina framework [15]. This enables its integration in the FRMG parser a large-coverage deep parser for French, based on TAG, that usually relies on the Le*fff*.

The Alexina format lexicon extracted from *LGLex* is called *LGLex*-Le*fff*, to distinguish it from the Le*fff*. The resulting verbal lexicon contains 22,060 entries for 5,736 distinct verb lemmas (on average, 3.85 entries per lemma). As a comparison, the Le*fff* only contains 7,072 verbal entries for 6,818 distinct verb lemmas (on average, 1.04 entries per lemma). The resulting lexicon extracted from *LGLex*, despite the fact that it describes fewer verbal lemmas, has a larger coverage in terms of syntactic constructions and therefore is much more ambiguous. At the extensional level, the Le*fff* has 361,268 entries whereas the *LGLex*-Le*fff* has 1,130,960 entries.

2.4. Format of processed corpus with the FRMG parser

This work allows the use of the linguistic data coded in Lexique-Grammaire tables for French to be used as a lexical database for a French parser, in particular the FRMG parser [11][6], which is relied on a syntactic lexicon in the Alexina format [15].

1 M. Gross takes as his starting point the study of simple French sentences. He thus takes the view that the minimum unit of meaning is the sentence. The principle adopted is to identify simple sentences and study the transformations that they can support. Studied features for each of these sentences are mainly formal features of syntax rather than semantics, which ensures reproducibility of tests [6]. However, some semantic features were taken into account when they could be tested clearly.

2 In order to make previous implicit features explicit, we created a table of classes [12, 13]. Its role is to assign features when their value is constant over a class, e.g. class defining features. Each row stands for a class and each column stands for a feature. Each cell corresponds to the validity of a feature in a class. In particular, the table of French verbs classes is composed of 67 different classes and 556 features.

3 `http://infolingu.univ-mlv.fr/english` > Language Resources > Lexicon- Grammar > Download.

4 If a verb has several meanings, it is divided in several lexical items. For example, *se rendre* has two meanings, so two lexical items:
Jean s'est rendu à mon opinion (John finally accepted my opinion).
Vercingetorix s'est rendu à Cesar (Vercingetorix surrendered to Ceasar).

5 On-line distribution under the LGPL-LR license at `http://gforge.inria.fr/ projects/alexina/`.

6 FRMG is free software, like Le*fff*, available under the INRIA GForge: `http://gforge. inria.fr/projects/mgkit/`. It is also possible to play with the chain of processing and visualizing the grammar FRMG on `http://alpage.inria.fr/frmgdemo`.

The integration of *LGLex*-Le*fff* in the FRMG parser is straightforward. The result is a variant of the FRMG parser, that we shall call FRMG*LGLex*, to distinguish it from the standard FRMGLe*fff*.

To use the results of the parsing in NLP applications of high-level, *Forest utils*[7]represents the forest of dependencies in format XMLDep [11]. Basically, we represent in XMLDep format a graph of dependencies with nodes (lemmas), grouped in clusters (forms), with arcs describing the syntactic dependencies between nodes.

3. IRASUBCAT

IRASUBCAT is a tool that acquires subcategorization information about the behaviour of any tag class (e.g., part of speech, syntactic function, etc.) or combination of them, from corpora [1, 2]. We are interested in using it to acquire information about verbs. It is aimed to address a variety of situations and needs, ranging from rich annotated corpora to virtually raw text (because the tags to study can be selected in the configuration file). The characterization of linguistic patterns associated to verbs will be correspondingly rich. The tool allows to customize most of the aspects of its functioning, to adapt to different requirements of the users. Moreover, IRASUBCAT is platform-independent and open source[8].

IRASUBCAT takes as input a corpus in XML format. This corpus is expected to have some kind of annotation associated to its elements, which will enrich the description of the patterns associated to verbs. The minimal required annotation is that verbs are marked. If no other information is available, the form of words will be used to build the patterns. If the corpus has rich annotation for its elements, the system can build the patterns with the value of attributes or with a combination of them, even with lexical items. The only requirements are that verbs are marked, and that all linguistic units to be considered to build the patterns are siblings in the XML tree.

The output of IRASUBCAT is a lexicon, also in XML format, where each of the verbs under inspection is associated to a set of subcategorization patterns. A given pattern is associated to a given verb if the evidence found in the corpus passes certain tests. Thresholds for these tests are defined by the user, so that precision can be prioritized over recall or the other way round. In all cases, information about the evidence found and the result of each test is provided, so that it can be easily assessed whether the threshold for each test has the expected effects, and it can be modified accordingly.
The lexicon also provides information about frequencies of occurrence for verbs, patterns, and their co-occurrences in corpus.

Moreover, IRASUBCAT allows to integrate the output lexicon with a pre-existing one, merging information about verbs and patterns with information that had been previously extracted, possibly from a different corpus or even from a hand-built lexicon. The only requirement is that the lexicon is in the same format as IRASUBCAT output lexicon.

We designed IRASUBCAT to be adaptable in a variety of settings. The user can set the conditions for many aspects of the tool, in order to extract different kinds of information for different representational purposes or from corpora with different kinds of annotation. For example, the system accepts a wide range of levels of annotation in the input corpus, and it is language independent. To guarantee that any language can be dealt with, the corpus needs to be codified in UTF-8 format, in which virtually any existing natural language can be codified.

4. Experiment with IRASUBCAT and the *LGLex* lexicon of French

We want to use the results of FRMG parser on a big corpus with IRASUBCAT in order to improve the *LGLex* lexicon of French, adding the frequencies of occurrence for each entry and each subcategorization frame. To do this, we must:
– choose a corpus with millons of words, also we just only need a small part of this corpus for the experiment;
– parse the corpus with the FRMG parser, with and without the *LGLex* lexicon (i.e. only with the Le*fff* lexicon) – results with FRMG*LGLex* and with FRMGLe*fff*;
– convert both the processed corpus and the *LGLex* lexicon into XML format, required by IRASUBCAT;
– use IRASUBCAT in order to add the frequencies of occurrence extracted from the big corpus into the *LGLex* lexicon.

4.1. The corpus

The processed corpus with FRMG*LGLex* (cf. 2.4 to see how we use the FRMG parser with the *LGLex* lexicon) used for the experiment is the CPJ (Corpus Passage Jouet) with 100K sentences of AFP, Europarl, Wikipedia and Wikisources, extracted from the corpus of the evaluation campaign (in 2009) for French parsers Passage [9].

4.2. Conversion into XML format

We created 2 programs in Python: one to convert the verbal *LGLex* lexicon in the same format as IRASUBCAT output lexicon, another to convert the processed corpus CPJ with the FRMG parser in a format directly readable
by IRASUBCAT.

Conversion of the verbal *LGLex* lexicon:

7 *Forest utils* is a set of Perl scripts to convert between various formats for shared derivation forest produced by parsers for TAG: `https://gforge.inria.fr/ projects/lingwb/`.
8 IRASUBCAT is available for download at `http://www.cs.famaf.unc.edu.ar/ ~romina/irasubcat/`.

The input is the verbal *LGLex* lexicon, or more precisely, the *extensional lexicon* of *LGLex*- Le*fff* lexicon, which contains each inflected form of the lemma and every possible redistribution (cf. 2.3).

In the output lexicon converted into XML format as IRASUBCAT output lexicon (named **lglex-lefff-IRASubcat.xml**), each lemma is associated to a set of subcategorization patterns. For example:

```
<pattern id="['Suj:cln|sn', 'Obj:sn']"></pattern>
<pattern id="['Suj:(cln|sn)', 'Obl:de-sinf']"></pattern>
```

The first pattern represents a subject which can be nominative clitic or noun phrase, and a direct object which is a noun phrase. The second represents an optional subject (between parenthesis) with the same distribution as the first, and an oblique (non-cliticizable) argument which is an infinitive clause introduced by a preposition *de*.

In fact, we simplify by omitting the realizations. So, we have only the syntactic functions (with the first letter in lower case) because it's more easy to find them in the processed corpus. We also ordered syntactic functions in alphabetical order to allow the research of all the order in the processed corpus (see the option ORDER OF TAGS = NO in 4.3).

For each lemma represented by his identifier (for example, *verb="achever ___V_1_1"*, which corresponds to the 1st entry in the verb class 1), a count of occurrences of this lemma is initialized to 0 (*count_oc_verb="0"*). We extracted the set of subcategorization patterns from all his inflected forms and all his redistributions and the number of different pattern is indicated (for example, *different_patterns="6"*). For each pattern (*['obj', 'suj'], ['obl', 'suj'], ['obl2', 'suj'] and ['obl', 'obl2']*), a count of occurences of this pattern for this lemma and a count of occurences of this pattern for all verbs are both initialized to 0 (*count_w_verb="0" total_count="0"*).

We have in total 14 068 distinct lemma. Here is a complete example of **lglex-lefff-IRASubcat.xml** (see the option DICTIONARY EXISTING = lglex-lefff-IRASubcat.xml in 4.3)[9]:

```
<dictionary>
  <entry verb="achever___V_1_1" count_oc_verb="0">
   <tag name="fs" different_patterns="6">
     <pattern id="['obj', 'suj']" count_w_verb="0" total_count="0" rejected_patterns_freq_test="NO">
     </pattern>
     <pattern id="['obl', 'suj']" count_w_verb="0" total_count="0" rejected_patterns_freq_test="NO">
     </pattern>
     <pattern id="['obl2', 'suj']" count_w_verb="0" total_count="0" rejected_patterns_freq_test="NO">
     </pattern>
     <pattern id="['obl', 'obl2']" count_w_verb="0" total_count="0" rejected_patterns_freq_test="NO">
     </pattern>
   </tag>
  </entry>
</dictionary>
```

Conversion of the processed corpus with the FRMG parser:

The input is the processed corpus CPJ with the FRMG parser, more precisely, with FRMG$_{LGLex}$, i.e. the FRMG parser with the *LGLex*-Le*fff* lexicon. In the processed corpus CPJ, we represented a graph of dependencies with nodes (lemmas), grouped in clusters (forms), with arcs describing the syntactic dependencies between nodes (cf. 2.4). So, we want to extract only the useful information in a format directly readable by IRASUBCAT.

In the output in XML format (named **CPJ-IRASubcat.xml**), for each sentence of the corpus (for example, *<sentence ID*[10]*="12" corpus="frwikipedia_012" s="12">*), we extracted the verbs (*cat="v"*) with their identifiers (for example, *lemmaid="achever___V_1_1"*). For each verb, we extracted the syntactic functions and we indicated the number of arguments (*nb_fs="2"*) and then, each syntactic function (*fs*) one by one (for example, *fs="suj"* for subject, and *fs="obl2"* for oblique).

Here is a complete example of **CPJ-IRASubcat.xml**:

```
<sentence ID="12" corpus="frwikipedia_012" s="12">
  <word lexica="achevée" lemma="achever" lemmaid="achever___V_1_1" cat="v" nb_fs="2">achevée</word>
  <word fs="suj"></word>
  <word fs="obl2"></word>
</sentence>
```

In this example, we can see that we decided to list all the syntactic functions after the verb. So, then we wanted to use IRASUBCAT reading only the arguments after the verb, for example 3 arguments (in the practice, we have never found

9 We have only 4 different patterns if we consider only the syntactic functions, without the realizations.

10 We lowered *id* because if you have *ID* attribute at *sentence* level, the execution of IRASUBCAT produced a file with the ID's of sentences that give origin of the patterns in the result dictionary. We calculated the *ID* as the number of the sentence considering all corpus, whereas s is the number of the sentence in the current corpus (here, Wikipedia Fr).

4 arguments). But the option LENGTH OF SIDE OF THE VERB FOR THE PATTERN = 3 (see in 4.3) allowed to read the 3 arguments before and after the verb. So, we changed the code of IRASUBCAT to read only after the verb. The best solution would be to add an other option, one to specify the number of arguments to read before the verb (LENGTH OF RIGHT SIDE OF THE VERB FOR THE PATTERN = 0) and another one to specify the number of arguments to read after the verb (LENGTH OF LEFT SIDE OF THE VERB FOR THE PATTERN = 3).

4.3. Using IRASUBCAT with *LGLex*

We changed the information in the configuration file to execute IRASUBCAT with our lexicon **lglex-lefff-IRASubcat.xml** and our corpus **CPJ-IRASubcat.xml** (in UTF-8):

```
TO CONSIDER VERB LIST = NO
#Option NO is going to consider every verbs, put path of file if you want to consider only verbs of the list in the file.
DICTIONARY EXISTING = lglex-lefff-IRASubcat.xml
#Option NO is going to create a new dictionary, put path of dictionary exist if you want to actualize it.
LENGTH OF SIDE OF THE VERB FOR THE PATTERN = 3
#ALL is going to consider every scope, and 3 is going to consider 3 patterns ONLY at rigth of verb.
COMPLETE WITH WORD = NO
#Here NO, is not complete with anything, other word, is going to complete with this word.
ORDER OF TAGS = NO
#Here NO, is going not to consider the order of tags, put YES if you want to consider the order.
TARGET TAGS = fs
#We consider only the tag fs, which contains syntactic categories, put tag1,tag2,... if you want to consider all this tags, put NO if you
#only want to consider lexical items.
USE LEXICAL ITEMS = NO
#Put YES if you want to consider the lexical items instead of lemmas or syntactic categories specified in the target of tags.
INTRODUCE VERBAL MARK = NO
#Put YES if you want the system put the symbol "|" in the position of the verb.
COLAPSE PATTERNS = NO
#Put YES if you want the system can collapse pattern, identifying optional constituents.
MAX ITERATION FOR FIND COLAPSE PATTERNS = FALSE
#FALSE means that the system is going to collapse every patterns that it can, put a number if you want the system stops after n
#iterations.
MINIMAL ABSOLUTE VERBAL FREQUENCY = 0
#Here 0, is going to consider all verbs even if they occur few times, other number, is going to consider the verb only if it found n
#times.
MINIMAL RELATIVE FREQUENCY TO CONSIDER PATTERN = 0
#Here 0, is going to consider all co-occurrences of verbs with a pattern even if they occur few times, other number, is going to
#consider the pattern only if it found n times with the verb.
USE LIKELIHOOD RATIO TEST = NO
#Put YES to use Likelihood Ratio to filter co-occurrences.
```

The execution (with the command line: **python IRASubcat.py CPJ-IRASubcat.xml**[11] **cat=”v” sentence lemmaid**[12] **config_CPJ.cfg**[13]) create the file **OutputDictionaryOrd.xml** with the lexicon, the file info file with the statistics of execution, and the file **IdsSentencesOrigenDictionary.xml** with the ID’s of sentences that give origin of the patterns in **OutputDictionaryOrd.xml**.

Here is the previous example of **lglex-lefff-IRASubcat.xml** as it appears in **OutputDictionaryOrd.xml**:

```
<dictionary>
  <entry verb="achever___V_1_1" count_oc_verb="1">
    <tag name="fs" different_patterns="4">
      <pattern id="['obj', 'suj']" count_w_verb="0" total_count="1001" rejected_patterns_freq_test="NO">
      </pattern>
      <pattern id="['obl', 'suj']" count_w_verb="0" total_count="214" rejected_patterns_freq_test="NO">
      </pattern>
      <pattern id="['obl2', 'suj']" count_w_verb="1" total_count="325" rejected_patterns_freq_test="NO">
      </pattern>
```

11 The first argument (**CPJ-IRASubcat.xml**, see an example in 4.2) is the path of our corpus. Remember that this corpus need to be in UTF-8 in XML format, the corpus can be in any language, IRASUBCAT needs is that the corpus have the verbs marked, like a characteristic in XML with a particular value, but IRASUBCAT has the capability of take as input a rich corpus, with a lot of information about its items.

12 The arguments **cat=”v” sentence lemmaid** indicate how identify in the corpus which one is the characteristic (*cat*) and value (*v*) to find verbs, the level father (*sentence*) of the level that have characteristics to study (that is the same level that have the characteristic for mark verbs), and the key of the dictionary of output (the value of *lemmaid*, as for example *achever___V_1_1* in 4.2). Note that *lemmaid* and *cat=”v”* need to be at level *word* (which have father level *sentence*).

13 The fifth argument (**config_CPJ.cfg**) is the configuration file customized to accept our corpus and our kind of execution (cf. 4.3 for the details of the configuration file).

```
    <pattern id="['obl', 'obl2']" count_w_verb="0" total_count="0" rejected_patterns_freq_test="NO">
    </pattern>
  </tag>
 </entry>
</dictionary>
```

We can see that the number of occurrences of the verb *achever___V_1_1* in the corpus is 1 and the pattern is *['obl2', 'suj']*. For this pattern, we have in total 325 occurences in the corpus for all verbs.

Here is an example of **IdsSentencesOrigenDictionary.xml**:

```
<ids_from>
 <entry verb="achever___V_1_1" total_count="1">
  <tag name="fs">
   <pattern id="['obj', 'suj']">
    <s_list>
    []
    </s_list>
   </pattern>
   <pattern id="['obl', 'suj']">
    <s_list>
    []
    </s_list>
   </pattern>
   <pattern id="['obl2', 'suj']">
    <s_list>
    ['12']
    </s_list>
   </pattern>
   <pattern id="['obl', 'obl2']">
    <s_list>
    []
    </s_list>
   </pattern>
  </tag>
 </entry>
</ids_from>
```

We can see that the occurence of *verb="achever___V_1_1"* with the pattern *['obl2', 'suj']* is in the sentence *['12']* as we have seen in 4.2.

Here is the information into info file for an extract of the CPJ:

```
Time: 271.12977194786072 seconds or 4.5188295324643457 minits
Total count of sentence: 1219
Total count verbs: 2020
Total different verbs: 14125
Total different patterns: 3
Total patterns rejected by frequence test: 0
Total patterns rejected by Likelihood Ratio test: 0
Total patterns 'NO_DECIDE' Likelihood Ratio test: 0
Total patterns accepted by Likelihood Ratio test: 0
```

The frequencies indicated in **OutputDictionaryOrd.xml** allow us to know the total number of occurences of each pattern in the corpus (table 1). We don't indicate the patterns which never appear.

The frequencies indicated in **IdsSentencesOrigenDictionary.xml** allow us to calculate the number of verbs associated with each total number of occurences of this verbs (table 2). We indicate the verb when there is only one verb.

pattern	***total_count***
['obj', 'suj']	1001
['obl2', 'suj']	325
['obl', 'suj']	214
['att', 'suj']	142
['loc', 'suj']	92
['obj', 'suj']	91
['suj']	62
['objde', 'suj']	55
['obj']	26
['dloc', 'suj']	11
others	0

Table 1. Number of occurrences of patterns

verb or number of verbs	***total_count***
être_____2	63
pouvoir___V_1_88	60
devoir___V_1_38	37
faire_____2	22
dire___V_9_130	19
vouloir___V_15_82	17
2	16
avoir___V_37E_10	13
2	12
3	10
4	9
3	8
8	7
12	6
14	5
30	4
63	3
192	2
740	1
13 043	0

Table 2. Number of occurrences of verbs

5. Conclusion

Using IRASubcat with the converted lexicon and the relevant information extracted of the processed corpus we can complete the lexicon with the frequencies of occurrence for each verb and each syntactic function. The processed corpus is the results of the frmg parser with *LGLex* lexicon, so it could find wrong sense.

The next step is to consider the information on realizations, that we must extract from processed corpus, but it is not a straightforward task. Then we have to use the frmg parser with Le*fff* lexicon only, without the *LGLex* lexicon influences the results. We could also use IRASubcat with another parser which is statistical, such as MaltParser, MSTParser, or Berkeley Parser [4]. And we could do a comparison using the original lexicon and the enlarged lexicon with that different parsers to verify that the accuracy is better using more information.

References

1. Ivana Romina Altamirano. IRASubcat: Un sistema para adquisición automática de marcos de subcategorización a partir de corpus. Master's thesis, FaMAF, National University of Córdoba, Argentina, 2009. (71 pp.).
2. Ivana Romina Altamirano and Laura Alonso Alemany. IRASubcat, a highly parametrizable, language independent tool for the acquisition of verbal subcategorization information from corpus. In *Proceedings of the NAACL HLT 2010 Young Investigators Workshop on Computational Approaches to Languages of the Americas*, pages 84–91, Los Angeles, California, 2010.
3. Jean-Pierre Boons, Alain Guillet, and Christian Leclère. *La structure des phrases simples en français : Constructions intransitives*. Droz, Geneva, Switzerland, 1976.
4. Marie Candito, Joakim Nivre, Pascal Denis, and Enrique Henestroza Anguiano. Benchmarking of statistical dependency parsers for french. In *Proceedings of COLING'2010 (poster session)*, Beijing, China, 2010.
5. Matthieu Constant and Elsa Tolone. A generic tool to generate a lexicon for NLP from Lexicon-Grammar tables. In Michele De Gioia, editor, *Actes du 27e Colloque international sur le lexique et la grammaire (L'Aquila, 10-13 septembre 2008), Seconde partie*, volume 1 of *Lingue d'Europa e del Mediterraneo, Grammatica comparata*, pages 79–193. Aracne, Rome, Italy, 2010. ISBN 978-88-548-3166-7.
6. Maurice Gross. *Méthodes en syntaxe : Régimes des constructions complétives.* Hermann, Paris, France, 1975.
7. Maurice Gross. *Constructing Lexicon-Grammars*. Oxford University Press, Oxford, England, 1994.
8. Alain Guillet and Christian Leclère. *La structure des phrases simples en français : Les constructions transitives locatives*. Droz, Geneva, Switzerland, 1992.
9. Olivier Hamon, Djamel Mostefa, Christelle Ayache, Patrick Paroubek, Anne Vilnat, and Eric de La Clergerie. Passage: from French parser evaluation to large sized treebank. In *Proceedings of the 6th Language Resource and Evaluation Conference (LREC'08)*, Marrakech, Morocco, 2008.
10. Benoît Sagot. The Le*fff*, a freely available and large-coverage morphological and syntactic lexicon for French. In *Proceedings of the 7th Language Resources and Evaluation Conference (LREC'10)*, Valletta, Malta, 2010.
11. François Thomasset and Éric de La Clergerie. Comment obtenir plus des méta-grammaires. In *Actes de la Conférence sur le Traitement Automatique des Langues Naturelles (TALN'05)*, Dourdan, France, 2005.
12. Elsa Tolone. Les tables du Lexique-Grammaire au format TAL. In *Actes de MajecSTIC 2009*, Avignon, France, 2009. (8 pp.).
13. Elsa Tolone. *Analyse syntaxique à l'aide des tables du Lexique-Grammaire français*. Éditions Universitaires Européenes, Saarbrücken, Germany, July 2012. ISBN 978-3-8381-8194-3 (352 pp.).
14. Elsa Tolone and Romina Altamirano. Adding frequencies to the *LGLex* lexicon with IRASubcat. In *Proceedings of ASAII'2013 (poster session)*, Córdoba, Argentina, 2013.
15. Elsa Tolone and Benoît Sagot. Using Lexicon-Grammar tables for French verbs in a large-coverage parser. In Zygmunt Vetulani, editor, *Human Language Technology. Challenges for Computer Science and Linguistics. 4th Language and Technology Conference, LTC 2009, Poznań, Poland, November 6-8, 2009, Revised Selected Papers*, volume 6562 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 183–191. Springer Verlag, 2011. ISBN 978-3-642-20094-6.
16. Elsa Tolone, Benoît Sagot, and Éric de La Clergerie. Evaluating and improving syntactic lexica by plugging them within a parser. In *Proceedings of the 8th Language Resources and Evaluation Conference (LREC'12)*, Istanbul, Turkey, 2012. (8 pp.).