INTEX and the processing of natural languages

Max Silberztein
silberz@bestweb.net

Contents

1. Introduction p. 2
2. Launching INTEX p. 3
3. Opening a text p. 4
4. Finite State Transducers in INTEX p. 5
5. Preprocessing the text p. 8
   INTEX units of processing p. 10
   Ambiguity p. 11
6. Apply dictionaries and lexical FSTs p. 14
7. Priority levels p. 15
8. INTEX dictionaries p. 16
9. From a DELAS to a DELAF p. 17
10. Multiple entries in the DELAS p. 18
    First entries of a DELAS p. 19
11. Inflectional FSTs p. 20
    'Delete' operator p. 21
    Stack operators p. 22
    Resulting DELAF p. 24
12. Lexical FSTs p. 25
13. Text dictionaries p. 27
14. Highlight compounds in the text p. 28
15. Locate a regular expression p. 29
16. Index a FST p. 30
17. Various Text transformations p. 32
    Enhanced FSTs p. 33
18. Statistical Analyses p. 34
19. Disambiguation with Local Grammars p. 37
20. Conclusion p. 39
1. Introduction

INTEX is a linguistic development environment that allows users to build large-coverage Finite State descriptions of Natural Languages and apply them to large texts (several dozen million words in real time).

Several modules of INTEX have been available since 1992 under NextStep; INTEX has been fully integrated in a graphical interface since 1996 (release 3.0), at which point it began to be distributed to research centers as a linguistic development tool. INTEX has just been ported to Windows 95-NT, as INTEX 4.0.

INTEX uses the work performed at the Laboratoire d’Automatique Documentaire et Linguistique (LADL), founded in 1967 by Prof. Maurice Gross. The goal of the LADL is to build a large-coverage description of Natural Languages by using 3 sets of tools:

- **Electronic dictionaries** and Finite State descriptions of the vocabulary and the morphology of Natural languages;
- **Local grammars** to identify frozen, semi-frozen and phrases in texts;
- Transformational rules described in a Lexicon-Grammar to extract (or generate) elementary sentences from (into) complex sentences.

One important aspect of INTEX is that Texts, Dictionaries and Grammars are all represented by **Finite State Transducers** (FSTs). Therefore, all the operations the user performs via the graphical interface are translated into a small number (about 30) of elementary operations on FSTs. For instance, applying a set of dictionaries to a text is performed by constructing a union of the dictionaries' FSTs, then applying the resulting FST to the text FST; removing lexical ambiguities in the text is performed by computing the intersection between a grammar FST and the text FST, etc.
2. Launching INTEX

The first operation consists of selecting the directory where the linguistic data is stored: alphabet of the language, preprocessing dictionaries and Finite State Transducers (FSTs), dictionaries for simple and compound words, FSTs representing the inflectional and derivational morphology of the language, FSTs used to remove lexical ambiguities, utilities for the maintenance of the dictionaries and the grammars.
3. Opening a text (Text->Open)

INTEX processes three types of texts:

- ASCII files are considered as raw texts; they are considered as sequences of units delimited by the NEWLINE character; they do not contain any linguistic data;
- INTEX files are enriched texts where linguistic units (usually, sentences) have been delimited by the mark \{s\}; these files may contain linguistic tags;
- Concordance files are sequences of lines; each line consists of 3 columns: a left context, a sequence, and a right context. Concordances may have been created by previously indexing a FST in a text.

Below, a raw ASCII text is loaded and then indexed.
4. Finite State Transducers (FSTs) in INTEX

First of all, a few definitions:

- an FST is a device that recognizes some sequences in the input, and associate them with some outputs. Typically, sequences are sequences of characters or sequences of words in the text written in a natural language; outputs are some linguistic information;

- an FST has the form of a graph that starts with an initial state, and ends with a terminal state. Recognized sequences are the ones that can be spelled by a path that goes from the initial state to the terminal state; Outputs of the FST are produced when a sequence has been recognized.

Now, let's explain how FSTs are applied to texts by the INTEX system:

1. *FSTs are applied from left to right*

When the FST has matched one sequence of the text, it is reapplied after the end of the matching sequence. For instance, consider the following text:

```
   z a b c d z
```

if we apply the following FST to this text in REPLACE mode (*FST outputs replace matching sequences*):

```
   a b c PATH#1
   b c d
```

we produce the resulting text:

```
   z PATH#1 d z
```

The sequence `a b c` matched and was replaced by the output of the transducer, i.e. `PATH#1`. If we apply the same FST in MERGE mode (*FST outputs are inserted in the text*), we produce the following result:

```
   z a b PATH#1 c d z
```

The sequence `a b c` matched, and the output `PATH#1` was inserted before the character `c`.

---

1. FST inputs are displayed inside boxes, FST outputs are displayed below boxes.
In these two examples, the sequence b c d was not even 'seen' by the system. The sequence a b c has priority over the sequence b c d because it has matched before.

Note that the output of the FST may be the empty string. The same rule applies, even though the text was not modified. For instance, the following FST:

produces the following, unmodified text:

z a b c d z

(a b c matched, then the FST is being applied at the position at d z).

(2) Longest matches have priority over shorter ones.

For instance, consider the following text:

z a b c d z

if we apply the following FST to it:

we get the resulting text:

(in REPLACE mode) z PATH#2 z
(in MERGE mode) z a b c PATH#2 d z

In other words, the matching sequence a b c d has priority over the sequence a b c because it is longer.
(3) INTEX doesn't handle ambiguous FSTs

If one sequence in the text is associated with 2 or more different outputs, INTEX performs an undefined action. For instance, if the following FST is applied to the text \textit{z a b z} in REPLACE mode:

we get one of the two results:

\[ z \text{ PATH#1 } z \]
or
\[ z \text{ PATH#2 } z \]

(4) INTEX doesn't handle FSTs that recognize the empty string

INTEX produces an error message if one attempts to apply an FST that recognizes the empty string.

Attention:

- INTEX \textbf{can} apply ambiguous FSTs to texts that are represented by FSTs;
- Therefore, ambiguous FSTs \textbf{can} also be used to disambiguate texts (because the disambiguated text is internally represented by an FST);
- When applying Finite State Automata\textsuperscript{2} to texts, users \textbf{can} choose to index only shortest matching sequences, only the longest matches or all matches.

\textsuperscript{2} In INTEX, Finite State Automata are FSTs that produce the empty string.
5. Preprocessing the text (Text->Preprocessing)

Let us go back to the parsing of an ASCII file. After having loaded the file, users can preprocess the text, i.e. prepare the text for the linguistic analyses. The preprocessing consists of three operations: identification of sentences, of unambiguous compounds, and of special tokens.

2.1. Identifying sentences

The standard FST Sentence.fst (stored in the current language directory) is applied to the text in MERGE mode, i.e. the output of the FST is inserted in the text. Generally, this FST is used to insert the sentence delimiter \{S\} between consecutive sentences. Further INTEX processing will take this mark into account to process and index every linguistic unit.

Gray nodes refer to embedded FSTs; for instance, LettreMaj is the name of an FST that identifies the 26 capital letters A…Z; MotsComposésAvecMaj is the name of an FST that lists all the French compound words that end with a capital letter (e.g. Vitamine C).
The FST `Sentence.fst` must be read in the following manner:

- if a period is followed by a word in capital letters, INTEX inserts the sentence delimiter between the period and the word (see on the top of the FST);
- if an single uppercase letter is followed by a period, followed by a word in uppercase (e.g. `J. Dupont`), INTEX does not insert any sentence delimiter;
- compound words that end with an uppercase letter (e.g. `Vitamine C`) may occur at the end of a sentence; the uppercase letter followed by a period must not be mistaken for an abbreviated firstname.

Thanks to the `Left to Right` priority seen previously, the last processing gets priority over the second processing, which has priority over the first processing. We then get the correct result:

\[
\begin{align*}
\text{J. Dupont comes.} & \quad \{S\} \quad \text{Paul eats some vitamine C.} & \quad \{S\} \quad \text{Luc also.} \\
(\text{C. Luc} & \quad \text{is not processed as J. Dupont}). \text{Although this FST is not perfect (some systematic errors}\end{align*}
\]
are due to the use of some English abbreviations in French texts), it process usual French novels and journalistic texts with a high rate of success (>99.5%).

2.2. Identifying unambiguous compounds

The second step of the preprocessing will consist of identifying and tagging the unambiguous compound words in the text, this operation corresponds to a look-up of the dictionary `Norm.dic` (stored in the current language directory).

Let us first define INTEX units of processing.
INTEX units of processing

INTEX users define the alphabet of the language in the file Alphabet stored in the current language directory.

All the characters that are listed in this file are letters; the other characters are delimiters.

The alphabet file consists of a sequence of lines ordered alphabetically; each line contains two or three 'equivalent' letters; these equivalence classes are used by the sort and the lookup routines. For instance, here are the first 7 lines of the French alphabet:

- Aa
- AÀà
- Aââ
- Bb
- Cc
- CÇç

Linguistic units are classified in two main types:

- simple words are sequences of letters, e.g. table
- compound words are sequences of simple words, e.g. red tape

Since the apostrophe, the hyphen and the blank are not listed in the alphabet, INTEX treats the following words as compounds (even though their constituents cannot appear alone):

- aujourd'hui, attaché-case, parce que

Generally, simple and compound words are identified by consulting the dictionaries of the system. In certain cases, they may be identified by applying morphological FSTs.
Ambiguity

Ambiguous words are words that correspond to more than one lexical entry in the dictionaries and the FSTs of the system.

Ambiguous compound words are sequences that correspond either to more than one lexical compound entry, such as:

- pied noir \((\text{Blackfoot}, \text{Frenchman born in Algeria})\)
- pied noir \((\text{Blackfoot}, \text{American Indian})\)

or to more than one sequence of lexical (simple or compound) entries, e.g.:

- red tape \(\{\text{red tape,.N}\} + \{\text{red,.A}\} \{\text{tape,.N}\}\)

Unambiguous compound words correspond to only one lexical compound entry, e.g.:

- a priori \(\{\text{a priori,.ADV}\}\)

Tagging a text consists of replacing its forms by the corresponding lexical entry, written between curly brackets.

One can only replace unambiguous, or disambiguated forms

It is desirable to identify and tag unambiguous compound words as soon as possible, in order not to treat their constituents (e.g. 'a' and 'priori') as simple words. This operation can be performed during the preprocessing analysis by means of the special dictionary Norm.dic stored in the current language directory, or by FSTs applied in REPLACE mode. For instance, the following FST:

\[
\begin{array}{c}
\text{a priori} \\
\{\text{a priori,.ADV}\} \\
\text{aujourd'hui} \\
\{\text{aujourd'hui,.ADV}\}
\end{array}
\]

would tag the two French unambiguous adverbs.

The French dictionary Norm.dic lists over one thousand entries.
2.3. Identifying special tokens

The last stage of the preprocessing consists of identifying and tagging special tokens, such as elided and contracted words, unambiguous abbreviations, etc. This step is performed by applying the FST `Norm.fst` (stored in the current language directory) in REPLACE mode.

For instance, by applying the following FST to English texts:

![Diagram of FST](image_url)

one replaces all the utterances of the sequences `you've` and `you're` by the more explicit sequences `you have`, `you are`. Such replacements are performed before indexing the text and before consulting the dictionaries.

One could perform a similar substitution to replace the sequence `I've` by the form `I have`, but that would lead to artificially add an ambiguity in the text: the sequence `I've` is not ambiguous (it is the pronoun `I`, followed by the verb `have`), while `I have` is ambiguous, as shown in the sentence:

\[\text{Nelson and Napoleon I have met in …}\]

(`I` is used as a roman numeral). Thus, it is better to use the following FST:

![Diagram of FST](image_url)

\[I,\text{PRO:1s} \ {\text{have,}V:P1s}\]

`I` is the pronoun, first person singular; `have` is the verb conjugated in the present, first person singular.
It is also possible to replace one word by a more complex sequence, such as in the following FST:

```
cannot
{can,.V} not
doesn't
does not
shan't
shall not
won't
{will,.V} not
```

(the forms *can* and *will* are ambiguous in general; it is better to tag them here).

### Unambiguous sequences of simple words

During this early stage of preprocessing, it is also possible to disambiguate a number of grammatical words by replacing them with a tag when they occur in certain contexts. For instance, the following FST:

```
{s',si.CONJS} il
{s',se.PRO+PPV} ils
{s',si.CONJS} ils
```

replaces on the one hand all the utterances of the sequence *s'ìl* by the following sequence:

```
... {s',si.CONJS} il ...
```

On the other hand, a sequence like *s'aide* would be replaced by:

```
... {s',se.PRO+PPV} aide ...
```

Thus, the ambiguous form (*s*' corresponds either to the conjunction *si*, or the pronoun *se*) has been disambiguated.
6. Apply dictionaries and lexical FSTs (Text->Apply dictionaries)

After the completion of the preprocessing stage, the user selects the dictionaries and FSTs used to identify simple words (left column) and compound words (right column) in the text.

The three types of files are:

- **.dic** — an ASCII file that corresponds to a DELAF-type dictionary;
- **.fst** — an FST graph
- **.bin** — a dictionary compacted into a FST represented in a binary file.
7. Priority levels

Dictionaries and FSTs are associated with a 3-level priority system used to hide or impose some lexical data:

- if a (simple or a compound) form in the text matches a high priority dictionary or FST, the consultation stops;
- if not, all 'regular' dictionaries and FSTs are consulted;
- if no lexical entry corresponds to the form, low level priority dictionaries and FSTs are consulted.

Giving a high priority to a dictionary or a FST for simple words is generally useful in order to remove 'unpleasant' ambiguities from the general-purpose dictionaries: if one knows that a certain use of a word never occurs in the current text, one can insert the word in a high priority dictionary, without the irrelevant analysis. For instance, the word *la* in French is three-time ambiguous:

\[
\{la,le.DET:fs\} + \{la,le.PRO+PPV\} + \{la,.N:ms\}
\]

*la* can be either a determiner, a pronoun or the masculine noun (musical note). These three analyses are represented in the general-purpose Delaf.bin dictionary. By entering the only first two entries in a dictionary that has priority over the Delaf, one gets rid of the noun (which is not frequent).

Giving a high priority to a dictionary or a FST for compounds is useful to get rid of systematic structural ambiguities when compound words can be shortened. For instance, the following regular expression represents two synonymous variants of the same adverb:

\[
\text{Dans l'intimité (la plus stricte + <E>)}
\]

(*in the intimacy*). If the longest variant occurs in a text, INTEX must not parse it as ambiguous, as seen in the following two tagged results:

- either the adverb: \{\text{dans l'intimité la plus stricte,ADV}\}
- or the sequence: \{\text{dans l'intimité,ADV}\ \text{la plus stricte}\}

Obviously, the longest variant has to have priority over the shortest one. Giving priority to dictionaries and FSTs for compounds gives priority to the longest matches.
8. INTEX Dictionaries (Dictionary->Open)

Dictionaries applied by INTEX to identify simple forms in texts must be in the format of a **DELAF**; dictionaries used to identify compounds must be in the format of a **DELACF**. These dictionaries associate text utterances with one lemma and some linguistic information. Here is for instance one entry of the French DELAF:

\[ \text{amuserions, amuser.V+t+4:C1p} \]

the form \textit{amuserions} is associated with the lemma \textit{amuser} which is a transitive (+t) verb (V) of the syntactic class #4 (+4); the form is conjugated in the Conditional, first person plural (C1p).

The format of user-defined dictionaries must be checked via the command: **Check format**.

---

CheckSemInfo describes the syntactic and semantic features associated with the entry (e.g. +t for transitive); CheckFlexInfo describes the inflectional codes (e.g. :ms for masculine singular).

---

3. Of course, users may edit the FSTs that describe INTEX dictionaries' format in order to add their own linguistic information.
9. From a DELAS to a DELAF (Dictionary->Inflect)

Some dictionaries can be entered by the users directly in the form of a DELAF; typically, dictionaries of first names, abbreviations, etc.

However, in most languages, the dictionary of all the inflected forms would be too big to be entered 'manually'. It is then better to manually build a DELAS-type dictionary, and automatically generate the corresponding DELAF.

In a DELAS dictionary, lexical entries are lemmas; each entry is associated with one FST that represents all the corresponding inflected forms. For instance, here are five entries of the French DELAS:

- `amuser, V3+t+4`
- `cousin, N1+Anim`
- `cousin, N32+Hum`
- `de, PREP`

`V3, A31, N1, N32` and `PREP` are names of inflectional FST files.

- All the words in a language that have the same set of suffixes, associated with the same inflectional information are associated with the same inflectional FST. For instance, in French, all the verbs that conjugate like `amuser` (`aider, voler, etc.`) are associated with the FST `V3`; all the nouns that take an 'e' in the feminine and an 's' in the plural are associated with the FST `N32`, etc.;

- if a word has no inflection, it must be associated with an FST that has no input and produces no output. For instance, in French, the FST `PREP` is the following (prepositions do not inflect):

```plaintext
  └─┘
```

10. Multiple entries in the DELAS

If one word is associated with more than one inflectional class, it must be represented by more than one DELAS entry. For instance:

\[
\text{cousin, N3}2+\text{Hum} \\
\text{cousin, N}1+\text{Anim}
\]

The first entry corresponds to a person (= cousin); the noun gets the feminine form *cousine*; the second entry corresponds to an animal (= mosquito); this noun is masculine only. In the same manner, the French DELAS includes the two following entries:

\[
\text{voile, N1}+\text{Conc} \\
\text{voile, N}21+\text{Conc}
\]

the first entry corresponds to a masculine noun (= veil); the second entry corresponds to a feminine noun (= sail). Although both entries are associated with the same distributional class (+Conc stands for concrete nouns), and to the same inflection (they both take an 's' in the plural) they must be associated with two different FSTs: one generates the code \(m\) (masculine) whereas the other generates the code \(f\) (feminine).

Sometimes, different sets of syntactic or semantic properties may lead to different inflections. For instance, the verb *voler* in the meaning of 'to steal' accepts a passive construction; therefore, the four past participle forms *volé, volée, volés, volées* must be represented, as we see in:

\[
\text{Ces fleurs ont été volées (= these flowers have been stolen)}
\]

The verb *voler* in the meaning of 'to fly' is intransitive; therefore, only the invariable form *volé* must be represented. This leads to at least two entries in the DELAS:

\[
\text{voler, V3}U+i \\
\text{voler, V3}+t
\]

\(V3U\) represents only one invariable participle form whereas \(V3\) represents the four forms.
First entries of the French DELAS dictionary, 'basic' vocabulary

a, N2
a, XI+PMCO
à, PREP
ab, XI+PMCO
abaisé, A32
abaisser, V3+t+11
abaisser, V3+t+32RA
abaisser, V3+t+38L
abaisser, V3E+pi
abandon, N1
abandonné, A32
abandonné, N32+Hum
abandonner, V3+t+32H
abandonner, V3+t+36DT
abandonner, V3+t+38LI
abandonner, V3+t+38LR
abandonner, V3+t+6
abandonner, V3+t+9
abandonner, V3E+pi+7
abandonner, V3U+i+31H
abandonner, V3U+i+35R
abasourdi, A32
abasourdissant, A32
abatage, N1
abats, N2P+Conc
abattage, N1
abatement, N1
abattoir, N1+Conc
abattre, V68+t+32H
abattre, V68+t+32R3
abattre, V68+t+37E
abattre, V68+t+38LD
abattre, V68+t+4
abattre, V68E+pi+35R
abattu, A32
abattu, N1
abbaye, N21+Conc
abc, N2
abcès, N2+Conc
abdication, N21
abdiquer, V3+t+32R3
abdomen, N1+Conc
abdominal, A76
abdominaux, N2P
abeille, N21+An1
aberrant, A32
aberration, N21
11. Inflectional FSTs

Inflectional FSTs associate sets of suffixes to the corresponding inflectional information. For instance, the FST $N32$ (used to describe the inflection of the noun cousin) is displayed below (this file is stored in the Inflection subdirectory of the current language directory):

The FST associates each suffix of the DELAS entry cousin to some inflectional codes:

- if nothing is concatenated to the DELAS entry, one gets cousin; this form is associated with the inflectional codes ms (masculine singular);
- if 'e' is concatenated to the DELAS entry, one gets cousin; this form is associated with the inflectional codes fs (feminine singular);
- if 's' is concatenated to the DELAS entry, one gets cousins; this form is associated with the inflectional codes mp (masculine plural);
- if 'es' is concatenated to the DELAS entry, one gets cousins; this form is associated with the inflectional codes fp (feminine plural).

A simple process of concatenating the FST N32 to the lexical entry, and then exploring the FST to generate all the paths will produce the resulting DELAF entries:

```
cousin, cousin.N32+Hum:ms
cousine, cousin.N32+Hum:fs
cousins, cousin.N32+Hum:mp
cousines, cousin.N32+Hum:fp
```
'Delete' operator

Lemmas are not always proper prefixes of their inflected forms. For instance, the verb 'have' gets the following forms:

\[ \text{have, having, has, had} \]

In order to produce the last three forms, one needs to be able to 'delete' characters from the lemma. This operation is performed by adding the 'delete' character \( \text{L} \) (Go Left) to the alphabet of the DELAS dictionary. The FST used to produce the four inflected forms is then:

From the DELAS entry: \text{have,V1}, the process of constructing a DELAF dictionary becomes the following: (1) first we get the four following lines by exploring the FST V1:

\[
\begin{align*}
\text{have, have.V1:W} \\
\text{haveLing, have.V1:G} \\
\text{haveLLs, have.V1:P3s} \\
\text{haveLLd, have.V1:K}
\end{align*}
\]

(2) then we spell each inflected form, interpreting the operator \( \text{L} \) as a 'delete last character' command. The final result is then:

\[
\begin{align*}
\text{have, have.V1:W} \\
\text{having, have.V1:G} \\
\text{has, have.V1:P3s} \\
\text{had, have.V1:K}
\end{align*}
\]
Stack operators

In theory, the two operators *insertion, deletion* are sufficient to represent all kinds of inflection, even highly irregular ones:

\[
\text{avoir} \Rightarrow \text{ont}
\]

Basically, one can always connect any form to any lemma, by deleting all the letters, and then spelling the form.

The problem I discuss now is a practical one. Consider for instance the following German nouns:

\begin{itemize}
  \item Anrand, Balg, Blatt, Dach, Daus, Fach, Gehalt, Gemach, Geschmack, Gewand, Haus, Inland, Kaff, Kalb, Kraut, Lamm, Land, Mahd, Mann, Mark, Maul, Pfand, Rand, Salband, Wald, Wams...
\end{itemize}

These nouns have a similar inflected form:

\begin{itemize}
  \item Anränder, Bälger, Blätter, Dächer, Däuser, Fächer, Gehälter, Gemächer, Geschmäcker, Gewänder, Häuser, Inländer, Käffer, Kälber, Kräuter, Lämmer, Länder, Mähder, Männer, Märker, Mäuler, Pfänder, Ränder, Salbänder, Wälder, Wämser...
\end{itemize}

Basically, the inflection of all these forms is the same: add an ¨ to the a two letters before the end, and add the suffix *er*.

Unfortunately, with the only two operators of insertion and deletion, each of these forms would have to be associated with a different FST:

\[
\text{Anrand} \Rightarrow \text{Anränder}
\]
\[
\text{Balg} \Rightarrow \text{Bälger}
\]
\[
\text{Blatt} \Rightarrow \text{Blätter}
\]

\[\Rightarrow\] Over 40 almost identical inflectional FSTs would have to be added to the system!

I added two operators to the inflection process: \texttt{c} (Copy) and \texttt{r} (Go Right); all the previous inflections are then performed with the same command:

\[
\text{LLLäRCCer}
\]

\text{go left three times, insert an 'ä', go right one time, copy the two letters, insert 'er'}
The string produced by the FST is processed by four stack operators that take a constant time:

- \( c \) insert character 'c' at the end of the form
- \( L \) delete last character; push it onto the stack
- \( R \) pop the stack
- \( C \) copy the character at the top of the stack to the end of the form; pop the stack

Here is how the string **BalgLLLälger** is processed:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Resulting form</th>
<th>Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Insert B</td>
<td>B^</td>
</tr>
<tr>
<td>a</td>
<td>Insert a</td>
<td>Ba^</td>
</tr>
<tr>
<td>l</td>
<td>Insert l</td>
<td>Bal^</td>
</tr>
<tr>
<td>g</td>
<td>Insert g</td>
<td>Balg^</td>
</tr>
<tr>
<td>L</td>
<td>Push g</td>
<td>Balg</td>
</tr>
<tr>
<td>L</td>
<td>Push l</td>
<td>Bal</td>
</tr>
<tr>
<td>L</td>
<td>Push a</td>
<td>Bal</td>
</tr>
<tr>
<td>ä</td>
<td>Insert ä</td>
<td>Bal</td>
</tr>
<tr>
<td>R</td>
<td>Pop</td>
<td>Bal</td>
</tr>
<tr>
<td>C</td>
<td>Pop &amp; Insert</td>
<td>Bäl</td>
</tr>
<tr>
<td>C</td>
<td>Pop &amp; Insert</td>
<td>Bälg</td>
</tr>
<tr>
<td>e</td>
<td>Insert e</td>
<td>Bälge</td>
</tr>
<tr>
<td>r</td>
<td>Insert r</td>
<td>Bälger</td>
</tr>
</tbody>
</table>

The result is **Bälger**. In the same manner, the following French verbs are associated with a unique FST V6:

- **acheter**, **amener**, **beder**, **écarteler**, **élever**, **peser**, **semer**

The conjugated forms: achète, amène, bêde, écartèle, élève, pèse, and sème are produced by the command: **LLLLèCC**.

- 200 FSTs are required for the description of English,
- 250 FSTs for Italian,
- 300 for French and Spanish,
- 450 for Bulgarian,
- 670 for German.

⇒ The construction of DELAF dictionaries takes a time proportional to their length (linear time).
Resulting DELAF dictionary

a,.N:ms:mp
a,.XI+PMCO
à,.PREP
ab,.XI+PMCO
abaissé,.A:ms
abaissée,abaissé.A:fs
abaissés,abaissé.A:mp
abaissées,abaissé.A:fp
abaissen,.V+t+11:W
abaissant,abaisser.V+t+11:G
abaissé,abaisser.V+t+11:Kms
abaissée,abaisser.V+t+11:Kfs
abaissés,abaisser.V+t+11:Kmp
abaissées,abaisser.V+t+11:Kfp
abaisse,abaisser.V+t+11:F1s:F3s:S1s:S3s:Y2s
abaisses,abaisser.V+t+11:F2s:S2s
abaissons,abaisser.V+t+11:F1p:Y1p
abaissiez,abaisser.V+t+11:F2p:Y2p
abaissent,abaisser.V+t+11:F3p:S3p
abaissais,abaisser.V+t+11:I1s:I2s
abaissait,abaisser.V+t+11:I3s
abaissions,abaisser.V+t+11:I1p:S1p
abaissiez,abaisser.V+t+11:I2p:S2p
abaissai,abaisser.V+t+11:I3p
abaissas,abaisser.V+t+11:I2s
abaissas,abaisser.V+t+11:I3s
abaissâmes,abaisser.V+t+11:J1p
abaissâtes,abaisser.V+t+11:J2p
abaissèrent,abaisser.V+t+11:J3p
abaisserai,abaisser.V+t+11:F1s
abaisseras,abaisser.V+t+11:F2s
abaissera,abaisser.V+t+11:F3s
abaisseron,abaisser.V+t+11:F1p
abaisserez,abaisser.V+t+11:F2p
abaisseront,abaisser.V+t+11:F3p
abaissasse,abaisser.V+t+11:T1s
abaissasses,abaisser.V+t+11:T2s
abaissât,abaisser.V+t+11:T3s
abaissassions,abaisser.V+t+11:T1p
abaissassiez,abaisser.V+t+11:T2p
abaissassent,abaisser.V+t+11:T3p
...

DELAF dictionaries are stored in minimal deterministic Finite State Transducers.
12. Lexical FSTs

Generally, FSTs must be used to represent infinite sets of lexical entries (e.g. numerical determiners); they are used as well to put together members of a 'natural' family, such as derived forms of a lemma, orthographic variants, synonymous expressions, etc. For instance, here is a FST used to tag the orthographic variants of the noun *tsar* in French:

Here is a FST that represents all the derived forms of the noun *France*:
The following FST $\texttt{Dnum}$ identifies and tag all the numerical determiners from 2 to 999999 written in French:

Since it is possible to embed FSTs in others, one can construct a large library of FSTs that will be re-used in other bigger projects. More than 3,000 FSTs have already been built for various descriptions of French:

- technical expressions (e.g. stock market, weather reports, medical statements, etc.),
- semi-frozen expressions (e.g. expressions of feelings),
- complements of dates, duration, addresses,
- etc.
13. Text dictionaries

The result of the application of the system dictionaries and FSTs is displayed below:

- the list of all the simple forms associated with their lemma and some linguistic data
- the list of all the simple forms that have not been found in the selected dictionaries
- the list of all ‘ambiguous’ compounds, such as ‘red tape’ (either a person, or a tape)
- the list of all ‘unambiguous’ compounds, such as ‘grand parent’

If a form is ambiguous, it appears on more than one line. For instance, abandonné is ambiguous: either the adjective masculine singular (A:ms), or the noun (N:ms), or the participle of the verb abandonner (V:Kms). The sequence à ce sujet (= speaking of this) can be an adverb; it is considered as an ‘ambiguous’ compound because the sequence is not obligatorily the adverb, as for instance in:

_ Je pense à ce sujet bien défini_  
 (= I think of this well defined topic)

These four files can be edited, so that the dictionaries used by further processings are closely adapted to the text (one can easily get rid of ambiguities that do not occur in the specific text).
14. Highlight Compounds in the text (Text->Locate)

All the compounds that have been identified by the consultation of the selected dictionaries are indexed; it is then possible to highlight them in the text (via the Display indexed sequences panel).

Generally, the indexation of compound nouns produces a very precious corpus to be used by information retrieval systems, because compound nouns are almost never ambiguous (compared to simple nouns).
15. Locate a regular expression in the text (Text->Locate)

One can index all the sequences of the text that match a particular regular expression. For instance:

<be> going to + (will + shall ) <V:W>

<be> stands for any inflected form associated with the lemma be (that is, any conjugated form of the verb to be); <V:W> stands for any form associated with the category V and the inflectional code W (that is, any verb in the infinitive). Any code present in any of the dictionaries is instantaneously useable, for instance:

<N+Hum> Noun, associated with the code Hum (Human)
<N-Hum> Noun, not Human
<N:fp> feminine plural Noun
<!PREP> any word that is not a Preposition

The resulting index can be displayed in the form of a concordance:
16. Index a FST in the text (Text->Locate)

One can index all the sequences of the text that match a particular FST.

The user can choose to index:

- shortest matches;
- longest matches;
- all matches.

FST outputs can:

- be ignored;
- be merged into the text;
- replace recognized sequences.

With the two last options, the FST can be reapplied several times to the text, for instance until no modification has been performed.
One of the grammars used to locate semi-frozen noun phrases that represent companies; when applied to newspapers, the resulting index can be used by information retrieval systems:
17. Various Text Transformations

FSTs can also be applied to texts in order to modify the text itself.

For instance, the following FST, if used in REPLACE mode, could be used to remove adverbs from the text (useful for information retrieval systems):

```
<ADV> →
```

In the same manner, one could design FSTs that would delete expressions like:

```
I think that, In the same manner, as far as I am concerned,
officials said that, he argued that, it seems doubtful that, etc.
```

The following FST can be used in MERGE mode to insert parentheses around some noun phrases:
Enhanced FSTs

In INTEX FSTs, outputs and inputs are synchronized, that is, the input sequences that match are indexed in the same time as the corresponding output. That feature is essential to the application in MERGE mode (see for instance how parentheses are inserted by the previous FST), or by disambiguation FSTs, where lexical constraints have to be applied to the correct matching forms.

INTEX users are allowed to purposely modify the conditions of this synchronization, by using internal variables to store parts of the matching input sequence. This feature is the same as in the UNIX tool sed. Here is one example. The following FST:

is used to extract the adverbs that may be inserted between the auxiliary verb avoir and the following past participle. For instance, when the FST is applied to the following text:

Luc m'avait souvent amusé

the sequence avait is stored in memory slot $1; the sequence souvent is stored in memory slot $2, and the sequence amusé is stored in memory slot $3. Performing the replacement produces the following result:

Luc m'avait amusé souvent

Look at the FST that identifies some French negations in the Norm directory (on the form ne <V> Neg).
18. Statistical Analyses

Any index (the index of the compounds, of the sequences that match a regular expression, or a FST) can be studied with a number of statistical measurements. Here for instance, the frequencies of the forms matching the regular expression <être> (all the conjugated forms of the verb être) are represented in a pie:
Below, we study the evolution of the number of Nouns (<N>) and Determiners (<DET>). The proportion of determiners grows much faster than the one of the nouns (which is almost linear).
Below we study the density of two regular expressions in the *Herald Tribune*, January 1994:

Peaks correspond to an important local density of the forms that match the indexed regular expression.

Various other statistical measures are available.
19. Disambiguation with Local Grammars

Local grammars are FSTs that recognize text sequences (e.g. *il le la donne*), then applies corresponding lexical constraints (e.g. `<PRO> <PRO> <PRO> <V:3s>`) to remove lexical hypotheses.

The result of the application of one or more disambiguating FSTs can be displayed in several ways. INTEX can:

- index all matching sequences (i.e. to study the coverage of the local grammars)
- index all inconsistencies between selected local grammars and the text (to locate errors in the grammars, or agreement errors in the text)
- tag the text, i.e. replace all disambiguated forms by the corresponding lexical entry;
- lemmatize the text, i.e. replace all disambiguated forms by the corresponding lemma;
- build a regular expression that represents all the ambiguities of the text;
- build and display the text in the form of a FST.
Internally, each text sentence is represented by a FST. INTEX can display it. For instance, the text:

\[
\text{il donne la pomme de terre cuite}
\]

is represented below with all the ambiguities that remain after having applied selected local grammars. Notice how the two ambiguous compound nouns \textit{pomme de terre} (= potato) and \textit{terre cuite} (= clay) are represented in the FST.
Conclusion

Research Tool

Over 30 research centers are presently using INTEX as a research tool in various domains: computational linguistics, corpus-based linguistics, information retrieval, terminology, literature studies, teaching of a second-language. At the First INTEX users' Workshop (March 1993, Université Paris 7), 60 participants came from 10 countries; the 15 papers demonstrated the various applications of the system.

I would like to emphasize the fact that thanks to INTEX, about 60 researchers from all over Europe have started to work within a shared framework, using the same tools and the same methodology to construct large coverage descriptions of a dozen natural languages. I believe this situation is unique in the field of Linguistics.

Large-coverage INTEX descriptions have already been built for Bulgarian (contact Prof. Elena Paskaleva, University of Sofia), English, French and Spanish (Contact Prof. Maurice Gross, Université Paris 7), German (Prof. Franz Günthner, University Maximilian, Münich), Greek (Prof. Ana Symeonides, University of Salonique), Italian (Prof. Annibale Elia, University of Salerne), Polish (Prof. Zygmunt Vetulani), Portuguese (Prof. Elisabete Ranchodd, University of Lisbon), Russian (Alex Kolesov, Academy of Sciences, Moscow). INTEX Dictionaries are being constructed for Korean (Jeesun Nam, University of Seoul) and Old French (Prof. Hava Bat-Zeev, University of Tel Aviv).

Teaching Tool for Linguistics, Corpus Linguistics and Computational Linguistics

INTEX is a great tool to teach Corpus Linguistics and Computational Linguistics.

Describing the linguistic data included in INTEX (Dictionaries for Simple Words, Description of the Morphology, Dictionaries for Compound words, Dictionaries for frozen expressions, Local grammars for disambiguating texts, etc.) as well as the technology used to handle this data (Finite State Automata and Transducers) corresponds to a full year course (50 Hours) for Masters students. There are a plethora of projects that remain to be proposed to students, as the description of Natural Languages is not nearly complete!

Teaching Tool for Second Languages

INTEX is also used to teach French as a second-language; it allows teachers to ask students to locate morpho-syntactic patterns in wide texts (such as 5 years of the newspaper *Le Monde*), to build local grammars for semi-frozen expressions (e.g. how to express a date in French) and to ‘play’ (edit, correct, apply and test) with some linguistic descriptions.