Tutoriel :
Intex et ses applications informatiques

Max Silberztein(1), Thierry Poibeau(2), Antonio Balvet(3)

(1)IBM et LADL
ms1@us.ibm.com
(2)Thales TRT et LIPN
Thierry.Poibeau@thalesgroup.com
(3)Thales TRT et Modyco
Antonio.Balvet@thalesgroup.com

Ce support de cours comporte de larges extraits du manuel Intex (principalement chapitres 1 et 15), par ailleurs disponible sur le site du LADL (`http://ladl.univ-mlv.fr/INTEX/index.html`). On pourra se reporter au manuel pour toute information complémentaire sur Intex. Il est complété par un aperçu de certaines applications développées par Thales Recherche et Technologies.

Support du tutoriel TALN 2001 « Intex et ses applications informatiques »

Conférenciers : Thierry Poibeau et Antonio Balvet

Résumé du tutoriel

Intex est un environnement de développement utilisé pour construire, tester et accumuler rapidement des motifs morpho-syntaxiques qui apparaissent dans des textes écrits en langue naturelle. Un survol du système est présenté dans [Silberztein, 1999] ; le manuel d'instruction est disponible [Silberztein 2000]. Chaque description élémentaire est représentée par une grammaire locale, qui est habituellement entrée en machine grâce à l'éditeur de graphe d'Intex.

Une caractéristique importante d'Intex est que chaque grammaire locale peut être facilement réemployée dans d'autres grammaires locales. Typiquement, les développeurs construisent des graphes élémentaires qui sont équivalents à des transducteurs à états finis, et réemploient ces graphes dans d'autres graphes de plus en plus complexes. Une seconde caractéristique d'Intex est que les objets traités (grammaires, dictionnaires et textes) sont représentés de façon interne par des transducteurs à états finis. En conséquence, toutes les fonctionnalités du système se ramènent à un nombre limité d'opérations sur des transducteurs. Par exemple, appliquer une grammaire à un texte revient à construire l'union des transducteurs élémentaires, la déterminiser, puis à calculer l'intersection du résultat avec le transducteur du texte. Cette architecture permet d'utiliser des algorithmes efficaces (par ex. lorsqu'on applique un transducteur déterministe à un texte préalablement indexé), et donne à Intex la puissance d'une machine de Turing (grâce à la possibilité d'appliquer des transducteurs en cascade).

Dans ce tutoriel, nous montrerons comment utiliser un outil linguistique tel qu'Intex dans des environnements informatiques. Nous nous appuierons sur des applications de filtrage et d'extraction d'information, réalisées notamment au centre de recherche de Thales. Les applications suivantes seront détaillées, tant sur le plan linguistique qu'informatique :

- filtrage d'information à partir d'un flux AFP [Meunier *et al.* 1999]

- extraction de tables d'interaction entre gènes à partir de bases de données textuelles en génomique. [Poibeau 2001]

Le tutoriel montrera comment Intex peut être employé comme moteur de filtrage d'un flux de dépêches de type AFP dans un cadre industriel. Il détaillera également les fonctionnalités de transformations des textes (transduction) permettant de passer rapidement de structures linguistiques variées à des formes normalisées permettant de remplir une base de données. Sur le plan informatique, on détaillera l'appel aux routines Intex, les paramétrages possibles (découpage en phrases, choix des dictionnaires...), et on survolera les nouvelles possibilités d'intégration (Intex API).

Bibliographie

Max Silberztein, 1999. Intex: a FST toolbox. *Theoretical Computer Sciences*, 231:1. Elsevier Science.

Max Silberztein, 2000. Manuel d'Utilisation Intex 4.22. Disponible à la page INTEX hébergée sur le site du LADL: http://ladl.univ-mlv.fr/INTEX/index.html

Frédéric Meunier, Antonio Balvet, Thierry Poibeau, 1999. Projet CORAIL (COmposition de Requêtes par des Agents Intelligents Linguistiques), *Linguesticae Investigationes*, 22, pp.369-381.

Thierry Poibeau, 2001, Extraction d'information dans les bases de données textuelles en génomique au moyen de transducteurs à nombre fini d'états, *TALN'2001*, Tours (*à paraître*).

1 PRÉSENTATION RAPIDE D'INTEX

1.1 Introduction

INTEX est un environnement de développement utilisé pour construire des descriptions formalisées à large couverture des langues naturelles, et les appliquer à des textes de taille importante en temps réel.

Les descriptions des langues naturelles sont formalisées sous la forme de dictionnaires électroniques, de grammaires représentées par des graphes à états finis et de lexiques-grammaires. INTEX fournit des outils pour décrire la morphologie flexionnelle et dérivationnelle, la variation orthographique et terminologique, le vocabulaire (les mots simples, les mots composés et les expressions figées), les phénomènes semi-figés à la limite entre lexique et syntaxe (grammaires locales, description des accords) et la syntaxe (grammaires syntagmatiques).

INTEX est aussi utilisé comme traitement de corpus : il permet de traiter rapidement des textes de plusieurs centaines de méga-octets (typiquement, le texte d'une année d'un quotidien, ce qui est équivalent à 150 livres de poche). Les opérations typiques sur les textes incluent l'indexation de motifs morpho-syntaxiques, d'expressions figées ou semi-figées (par ex. techniques), de concordances lemmatisées, et l'étude statistique des résultats.

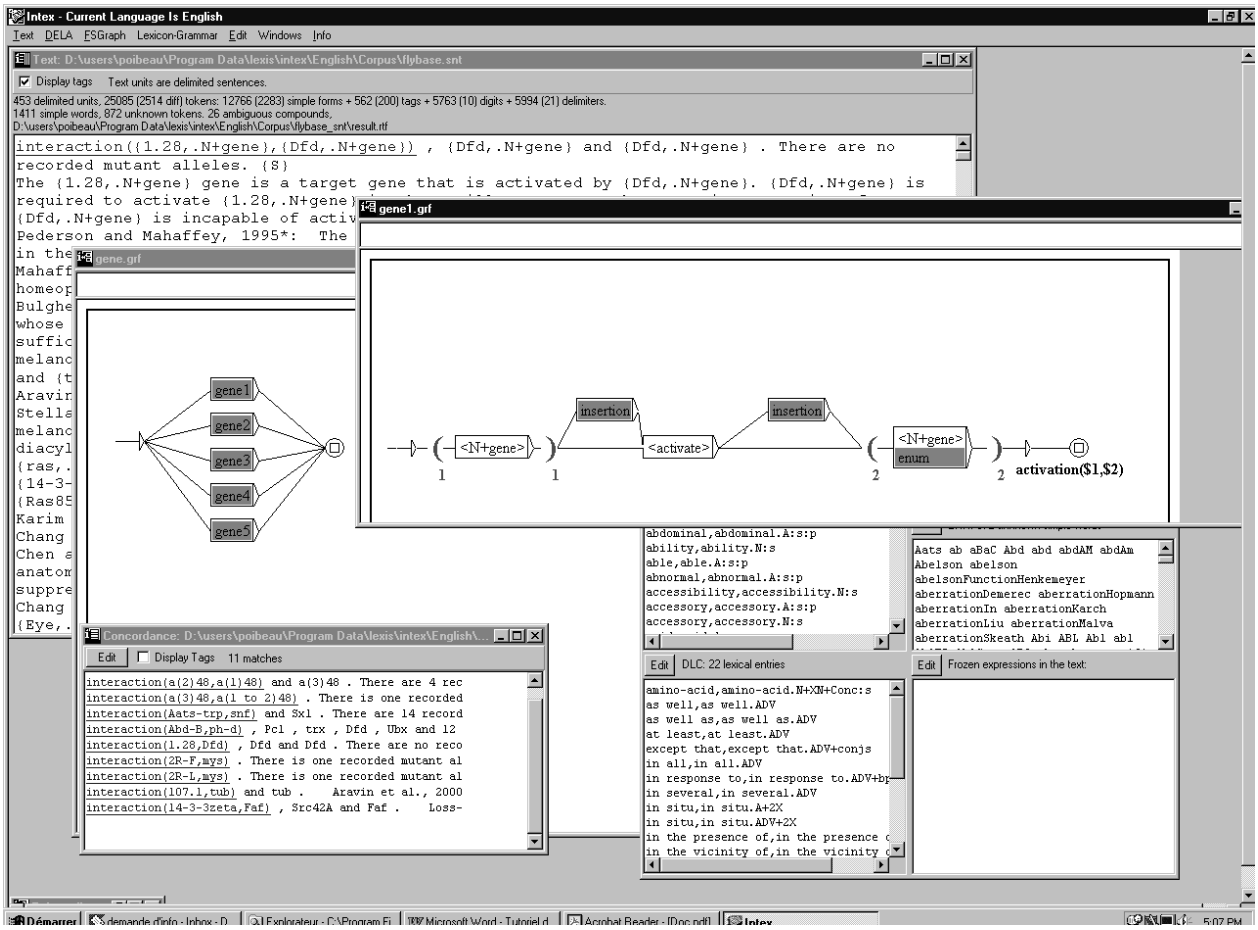

Figure 1 : l'interface du système Intex

1.2 Caractéristiques techniques

INTEX tourne sous Windows 95, Windows 98 et Windows NT 4.0. INTEX étant très exigeant en terme de puissance de calcul et d'occupation mémoire, certaines anciennes versions de ces systèmes qui contiennent des dysfonctionnements (« bugs ») peuvent affecter le bon fonctionnement du système. Nous conseillons vivement de mettre à jour le système d'exploitation, en particulier en téléchargeant le dernier « Service Pack » disponible.

Les caractéristiques standard d'un micro-ordinateur pouvant faire tourner INTEX sur des textes de la taille d'un roman (moins de 1 Mo) ne sont pas très élevées : PC de type Pentium 2, 32 Mo de mémoire vive, 500 Mo de disque dur disponibles, écran de 17 pouces, résolution de 1024x768 en 16 bits avec un taux de rafraîchissement minimal de 75 Hz.

Si INTEX est utilisé pour faire de la recherche documentaire sur des textes importants (50 Mo et plus), ou si INTEX est utilisé pour compiler des dictionnaires de mots composés (10 000 entrées et plus) ou des tables de lexique-grammaire, nous conseillons la configuration suivante : PC de type Pentium 3, 128 Mo de mémoire, 2 Go de disque dur disponibles.

Si INTEX est utilisé comme outil de développement pour construire des graphes, un bon écran est nécessaire : moniteur de 19 pouces minimum, capable d'afficher 1280x1024 points en 16 bits avec un taux de rafraîchissement de 80 Hz ou plus.

1.3 Outils

1.3.1 Transducteurs finis

Une caractéristique essentielle d'INTEX est que tous les objets traités (textes, dictionnaires, grammaires) sont à un moment ou à un autre représentés par des **transducteurs à états finis**.

Un transducteur à état fini est un graphe qui représente un ensemble de séquences en entrée, et leur associe des séquences produites en sortie. Typiquement, une grammaire représentera des séquences de mots (lues dans le texte), et produira des informations linguistiques (par exemple des informations sur la structure syntaxique) ; un dictionnaire représentera des séquences de lettres (qui épellent chaque entrée lexicale), et produira des informations lexicales (partie du discours, codes flexionnels, etc.) ; le transducteur d'un texte représentera les séquences de mots (qui forment chaque phrase) et leur associe des informations lexicales et/ou syntaxiques (les marques linguistiques produites par les différentes analyses).

Représenter ces trois types d'objets de la même façon présente des avantages considérables, notamment en terme de rapidité d'exécution. Toutes les opérations effectuées par INTEX se ramènent à un ensemble limité d'opérations sur des transducteurs. Par exemple, appliquer des dictionnaires à un texte consistera à construire l'union des transducteurs de chaque dictionnaire (le résultat est un transducteur), et à projeter ce transducteur sur le transducteur du texte. Par ailleurs, INTEX permet des opérations inédites, comme par exemple l'application de grammaires à des dictionnaires (par exemple pour vérifier leur format).

1.3.2 Automates finis, expressions rationnelles

En ce qui concerne INTEX, les **automates finis** sont un cas particulier de transducteurs à états finis : ce sont des transducteurs qui ne produisent aucune information, autre que l'information binaire « séquence reconnue », ou « séquence non reconnue ». Typiquement, nous utiliserons des automates finis pour étudier (rechercher, indexer, extraire, compter, etc.) des séquences définies dans des textes.

Les **expressions rationnelles** constituent un moyen rapide d'entrer des automates finis simples en machine, sans avoir à dessiner des graphes. Lorsque la séquence à rechercher consistera en un, deux ou trois mots, il est bien plus rapide d'entrer ces mots directement dans une expression rationnelle ; en revanche, dès que la structure à rechercher se complique, il est préférable de construire des graphes.

1.4 Commandes INTEX

Les fonctionnalités disponibles grâce à l'interface graphique d'INTEX sont aussi disponibles sous la forme de commandes Windows 95/NT. On peut donc construire des applications qui

ont la puissance d'INTEX simplement en appelant ces commandes, soit directement dans un programme "BATCH" (en alignant les commandes) ou dans un programme plus sophistiqué écrit en PERL, C++, JAVA, etc. Cette possibilité permet aussi d'ouvrir INTEX en modifiant son comportement (on peut ainsi remplacer certaines de ses fonctionnalités).

1.5 La communauté INTEX

A ce jour, plus de 100 utilisateurs utilisent INTEX comme outil de recherche ou d'enseignement, dans une dizaine de pays. Certains sont intéressés par les fonctionnalités de traitement de corpus (analyse de textes littéraires, recherche d'information dans des journaux ou des documents techniques, etc.), d'autres par les fonctionnalités de description linguistique (description de la morphologie, du lexique et des expressions d'une langue), d'autres enfin par l'aspect linguistique informatique (analyse automatique de textes).

Ces utilisateurs constituent une véritable communauté, et nous encourageons vivement les nouveaux utilisateurs à rejoindre, à travers les journées INTEX (les quatrièmes journées INTEX ont eu lieu en juin 2001 à Bordeaux), le colloque annuel sur les lexiques-grammaires comparés. Le site INTERNET ladl.univ-mlv.fr contient aussi des informations sur les travaux sur les dictionnaires électroniques DELA, les lexiques-grammaires et INTEX.

2 UTILISER INTEX EN LIGNE DE COMMANDE

2.1 Les variables d'environnement

Pour pouvoir utiliser INTEX en ligne de commande, il faut (bien entendu) qu'INTEX ait été installé sur la machine, en général à partir du programme d'installation automatique « **Setup.exe** ». Ensuite, les programmes d'INTEX doivent avoir accès aux cinq variables d'environnement suivantes :

INTEX : variable contenant le chemin du dossier dans lequel l'application INTEX a été installée ; par défaut : « **c:\Program files\Intex** » ;

INTEXPRV : variable contenant le chemin du dossier INTEX personnel ; par défaut : « **c:\Intex** » ;

INTEXAPP : variable contenant le chemin du dossier de l'application INTEX dans lequel les commandes ont été installés ; par défaut : « **c:\Program files\Intex\App** » ;

INTEXLNG : variable contenant le chemin du dossier de l'application INTEX de la langue courante ; par exemple : « **c:\Program files\Intex\French** » ;

INTEXLNG0 : variable contenant le chemin du dossier INTEX personnel de la langue courante ; par exemple « **c:\Intex\French** ».

En général, on voudra aussi ajouter **INTEXAPP** au contenu de la variable système **PATH** pour pouvoir utiliser les commandes INTEX à partir de n'importe quel dossier.

Voici un extrait de fichier « profile.bat » utilisé pour lancer les commandes INTEX à partir d'une fenêtre de commande DOS :

```
set INTEX= C:\Program Files\Intex

set INTEXPRV= C:\MonIntex

set INTEXAPP= %INTEX%\App

set INTEXLNG= %INTEX%\French

set INTEXLNG0= %INTEXPRV%\French

set PATH= %INTEXAPP%;%PATH%
```

Attention à bien comprendre la différence entre le dossier de l'application et le dossier personnel :

- Le dossier de l'application est utilisé lors de l'installation d'INTEX ; il contient l'ensemble des programmes INTEX, ainsi que les données originelles (dictionnaires et grammaires entre autres) fournies avec INTEX. En général, ce dossier ainsi que tous les dossiers imbriqués doivent être protégés en écriture : aucun utilisateur ne doit pouvoir modifier les fichiers contenus dans ce dossier ;

– Chaque utilisateur INTEX a son propre dossier personnel ; c'est dans ce dossier que sont rangées les données appartenant à chaque utilisateur : résultats des traitements (concordances, index), textes, dictionnaires et grammaires personnels, etc. Par exemple, chaque utilisateur peut éditer des dictionnaires ou grammaires fournis avec INTEX ; en fait une copie de ces fichiers sera rangée dans le dossier personnel. Pour tous les traitement ultérieurs, si un dictionnaire ou une grammaire apparaît à la fois (avec le même nom) dans le dossier personnel et dans le dossier de l'application, la version rangée dans le dossier personnel sera utilisée. De cette façon, plusieurs utilisateurs INTEX peuvent partager le même ordinateur. L'utilisateur peut indiquer où son dossier personnel se trouve grâce au menu « **Info > Preferences** ».

2.2 Dossiers et fichiers utilisés par INTEX

Les formats des fichiers utilisés sont soit standard ou « évidents » (lorsqu'il s'agit de fichiers au format texte par exemple). Toutes les commandes appelées par l'interface graphique sont affichées dans la console d'INTEX **(Info > Consol**e) avec leurs paramètres (touche F2). En général, on peut effectuer un simple copier/coller entre la console INTEX et une fenêtre de commande pour lancer directement ces commandes.

Le contenu de la console est aussi disponible dans le fichier « **log.txt** » rangé dans le dossier personnel. Ce dossier est remis à zéro à chaque lancement d'INTEX.

2.2.1 Dossier du texte

Lorsqu'on utilise les commandes INTEX directement (sans passer par l'interface graphique), on peut ranger les fichiers résultats des traitements où l'on veut. Dans la suite, nous préciserons cependant où les résultats des traitements sont rangés lorsque les commandes sont appelées par INTEX (lorsqu'on utilise l'interface graphique).

Quand on utilise l'interface graphique d'INTEX pour traiter un texte donné, INTEX associe au texte un dossier dans lequel toutes les informations relatives à ce texte sont rangées : l'index du texte, son vocabulaire, les concordances construites, etc. Le **dossier du texte** est rangé à côté du fichier texte ; son nom est obtenu à partir du nom du fichier en remplaçant le point d'extension par un caractère souligné « _ ». Par exemple, si le texte traité est :

```
c:\Mon \Intex\French\Corpus\Toto.txt
```

alors le dossier du texte sera :

```
c:\Mon \Intex\French\Corpus\Toto_txt
```

2.2.2 Fichiers et dossiers spéciaux

La plupart des commandes INTEX utilisent par défaut ou obligatoirement des fichiers dont le nom et le dossier sont spécifiques :

Alphabet : le fichier qui recense les caractères traités comme des lettres ; ce fichier doit obligatoirement se trouver dans le dossier INTEX de l'utilisateur ou, sinon dans le dossier de l'application INTEX.

idx, **ida** : l'index du texte courant ; ces deux fichiers doivent être rangés dans le dossier du texte. Noter que les programmes `dicos`, `reconind` et `recorind` accèdent à ces fichiers, même si ceux-ci n'apparaissent pas en ligne de commande.

Attention : à chaque fois qu'INTEX ouvre un texte, il vérifie que la date de dernière modification du fichier texte est antérieure à celle du fichier idx. Si ce fichier n'existe pas, ou s'il a été créé avant la dernière modification du texte, INTEX relance l'indexation du texte, et détruit tous les fichiers du dossier du texte.

licence : fichier d'inscription du logiciel. Ce fichier est utilisé pour vérifier que la clé d'installation correspond au numéro de série du disque dur « **C:** », et aussi pour décrypter les dictionnaires « **.bin** » rangés dans les dossiers de l'application. Ce fichier doit absolument être rangé dans le dossier « **App** » du dossier INTEX de l'application, et ne doit pas être modifié.

Delaf, **Delac**f, **Delae** : INTEX s'attend à ce que ces dossiers existent dans le dossier de la langue courante, dans le dossier de l'application INTEX et aussi dans le dossier privé de l'utilisateur. Ces dossiers sont utilisés par exemple par les programmes `dicos`, `dicoc` et `dicoe` en conjonction avec le caractère « ~ » pour localiser des dictionnaires.

2.2.3 Résultats des traitements

La plupart des commandes INTEX produisent des commentaires, par exemple, « **Loading grammar DET** » ou « **Cannot find grammar AdvTime** », et/ou une description des résultats produits, par exemple « **235 matching sequences** ». En général, lorsqu'on utilise l'interface graphique d'INTEX, ces commentaires sont écrits dans le fichier « **res.txt** » du dossier personnel **(INTEXPRV**).

2.3 Les commandes

Les commandes suivantes correspondent à des fichiers « .exe » rangés dans le dossier **INTEXAPP**.

`tokenslist Index ResultingList`

Sous l'interface graphique d'INTEX, après avoir ouvert et indexé un fichier-texte **(Text > Ope**n) INTEX fournit la liste des 100 lexèmes *(token*s) les plus fréquents dans la fenêtre « **Tokens list** ». Rappelons qu'INTEX distingue quatre types de lexèmes :

- *simple forms* : séquences de lettres délimitées par des séparateurs ;
- *tags* : séquences de caractères entre les deux caractères spéciaux « { » et « } » ;
- *digits* : les chiffres (de « 0 » à « 9 ») ;
- *delimiters* : tout séparateur, à part les chiffres et les blancs (espace, caractères de tabulation, « NEWLINE » et « CARRIAGE RETURN »).

La commande `tokenslist` prend comme argument l'index du texte, et construit la liste de **tous** les lexèmes du texte, associés à leur fréquence et triée alphabétiquement.

Le fichier index du texte construit par INTEX (si l'on utilise l'interface graphique) a pour nom « `idx` » et est rangé dans le dossier du texte.

`concord LLength RLength Tag? Text Index Concordance`

Ce programme construit une concordance (`Concordance`) à partir d'un texte (`Text`), de l'index des séquences reconnues (`Index`), et de la donnée du nombre de caractères à gauche des séquences reconnues (`LLength`) et à droite (`RLength`).

Attention : le paramètre `RLength` compte le nombre de caractères en incluant ceux contenus dans les séquences reconnues. Si ce paramètre est inférieur à la longueur des séquences reconnues, celles-ci seront coupées.

Cette fonctionnalité a été choisie parce que les concordances sont souvent construites pour être imprimées, chaque contexte étant présenté sur une ligne. En sélectionnant des longueurs de contextes adéquates et une police de caractères à largeur fixe (comme « Courier »), on peut obtenir de belles concordances.

Si le paramètre `RLength` est 0 (zéro), les séquences reconnues sont toujours affichées en entier, mais aucun contexte droit n'est fourni. On peut donc obtenir la simple liste des séquences reconnues en donnant 0 comme longueurs gauche et droite.

Le paramètre `Tag?` a pour valeur « yes » si la concordance doit afficher les étiquettes, « no » sinon.

```
dic2fst AsciiDelaf FstDelaf
```

construit un transducteur fini déterministe minimal à partir d'un dictionnaire au format DELAF/DELACF. Le transducteur est représenté par deux fichiers : `FstDelaf.bin` contient la partie « automate », c'est-à-dire la partie reconnaissance du transducteur ; le fichier `FstDelaf.inf` contient la partie « production » du transducteur, c'est-à-dire les informations associées à chaque mot reconnu.

```
dicoc [cdl] Text ResDir Stats ResAmbDic ResNonAmbDic ResIndex
C-Dic...
```

Programme de consultation de dictionnaires de mots composés. Le premier paramètre est un caractère qui décrit le type de fichier texte dans lequel on recherche les mots composés :

« `c` » : le texte est une concordance ; en conséquence, on ne traite que le texte en deuxième colonne (les colonnes sont délimitées par un caractère de tabulation) ;

« `d` » : les unités de texte (en général, des phrases) sont délimitées par la marque « `{S}` » ;

« `l` » : les unités de texte sont les lignes (ou paragraphes), délimitées soit par le caractère « NEW LINE », soit par la séquence de deux caractères « CARRIAGE RETURN » - « NEW LINE ».

Le second paramètre est le nom du fichier-texte dans lequel on recherche les mots composés ; le troisième paramètre est le nom du dossier dans lequel les quatre fichiers-résultats vont être rangés :

- `Stats` : nom du fichier (rangé dans `ResDir`) qui contient le nombre de mots composés trouvés ;
- `ResAmbDic` : nom du fichier (rangé dans `ResDir`) qui contient la liste des mots composés ambigus trouvés ;
- `ResNonAmbDic` : nom du fichier (rangé dans `ResDir`) qui contient la liste des mots composés non-ambigus trouvés ;
- `ResIndex` : nom du fichier (rangé dans `ResDir`) qui contient l'index des mots composés trouvés.

Note : les deux paramètres ResAmbDic et ResNonAmbDic sont gardés seulement pour des raisons de compatibilité. Lorsqu'on utilise l'interface d'INTEX, ils sont toujours égaux : la distinction entre mots composés ambigus et non-ambigus n'est plus pertinente dans le module de consultation des dictionnaires ; c'est maintenant une fonctionnalité du module de désambiguïsation.

Les paramètres suivants `C-Dic` sont les noms des dictionnaires de mots composés (d'extension « `.bin` », « `.dic` » ou « `.fst` ») que l'on veut appliquer.

- Si le premier caractère du nom d'un dictionnaire est « ~ », alors le dictionnaire est recherché dans le dossier **Delacf** de la langue courante, d'abord dans le dossier INTEX de l'utilisateur, puis dans le dossier INTEX de l'application. Par exemple, « **~Noms.bin** » est une référence à un dictionnaire « **Noms.bin** » rangé dans le dossier « **c:\My Intex\French\Delacf** », ou alors dans le dossier « **c:\Program files\Intex\Delacf** ».

- Si le caractère qui précède le point (l'extension du nom de fichier) est un caractère « - », alors le dictionnaire est appliqué en premier ; si le caractère est un caractère « + », le dictionnaire est appliqué en dernier.

`dicoe [cdl] Text ResDir Stats ResDic ResIndex TextDLF TextDLC FSTs…`

Programme de recherche des expressions figées. Le premier paramètre est un caractère qui décrit le type de fichier texte dans lequel on recherche les mots composés :

- « `c` » : le texte est une concordance ; en conséquence, on ne traite que le texte en deuxième colonne (les colonnes sont délimitées par un caractère de tabulation) ;

- « `d` » : les unités de texte (en général, des phrases) sont délimitées par la marque « `{S}` » ;

- « `l` » : les unités de texte sont les lignes (ou paragraphes), délimitées soit par le caractère « NEW LINE », soit par la séquence de deux caractères « CARRIAGE RETURN » - « NEW LINE ».

Le second paramètre est le nom du fichier-texte dans lequel on recherche les expressions figées ; le troisième paramètre est le nom du dossier dans lequel les trois fichiers-résultats vont être rangés :

- `Stats` : nom du fichier (rangé dans `ResDir`) qui contient le nombre d'expressions figées trouvées ;

- `ResDic` : nom du fichier (rangé dans `ResDir`) qui contient la liste des expressions figées trouvées ;

– `ResIndex` : nom du fichier (rangé dans `ResDir`) qui contient l'index des expressions figées trouvées.

Les deux paramètres suivants `TextDLF` et `TextDLC` sont les noms des dictionnaires du texte (que l'on a préalablement construit grâce à `dicos` et `dicoc`) que l'on veut utiliser pour reconnaître les expressions figées. Pour ne pas utiliser ces fichiers, on peut entrer le caractère « – ».

Les paramètres suivants sont les noms des grammaires d'expressions figées (extension « `.fst` ») que l'on veut appliquer.

Si le premier caractère du nom d'une grammaire est « ~ », alors la grammaire est recherchée dans le dossier **Delae** de la langue courante, d'abord dans le dossier INTEX de l'utilisateur, puis dans le dossier INTEX de l'application. Par exemple, « **~C1d.fst** » est une référence à une grammaire « **C1d.fst** » rangée dans le dossier « **c:\My Intex\French\Delae** », ou alors dans le dossier « **c:\Program files\Intex\Delae** ».

Ces grammaires sont :

-- des « vrais transducteurs », c'est-à-dire auto-suffisants, en général compilés à partir de graphes ou de séries de graphes (grâce à la commande **FSGraph > Compile FST**) ;

-- ou alors des transducteurs qui contiennent des références non résolues à des grammaires rangées dans le dossier « **Graphs\Lib** », comme par exemple Ins. ou Date. Ces grammaires ont été construites à partir d'un graphe-patron et d'une table de lexique-grammaire. Les références sont résolues par le programme `dicoe`.

`dicos [cdl] Text ResDir ResDic ResErr Stats S-Dic...`

Programme de consultation de dictionnaires de mots simples.

`[cdl]` : type du texte (« c » : concordance, « d » : texte délimité, « l » : ligne par ligne) ;

Le second paramètre `Text` est le nom du fichier-texte dans lequel on recherche les mots simples ; le troisième paramètre est le nom du dossier dans lequel les trois fichiers-résultats vont être rangés :

– `ResDic` : nom du fichier (rangé dans `ResDir`) qui contient la liste des mots simples trouvés ;

– `ResErr` : nom du fichier (rangé dans `ResDir`) qui contient la liste des formes simples non trouvées ;

– `Stats` : nom du fichier (rangé dans `ResDir`) qui contient le nombre de mots simples trouvés.

Les paramètres suivants `S-Dic` sont les noms des dictionnaires de mots simples (d'extension « `.bin` », « `.dic` » ou « `.fst` ») que l'on veut appliquer.

– Si le premier caractère du nom d'un dictionnaire est « ~ », alors le dictionnaire est recherché dans le dossier **Delaf** de la langue courante, d'abord dans le dossier INTEX de l'utilisateur, puis dans le dossier INTEX de l'application. Par exemple, « **~Prenoms.bin** » est une référence à un dictionnaire « **Prenoms.bin** » rangé dans le dossier « **c:\My Intex\French\Delaf** », ou alors dans le dossier « **c:\Program files\Intex\Delaf** ».

– Si le caractère qui précède le point (l'extension du nom de fichier) est un caractère « - », alors le dictionnaire est appliqué en premier ; si le caractère est un caractère « + », le dictionnaire est appliqué en dernier.

`enrich Text Index Result`

Ce programme prend comme paramètre le nom d'un fichier-texte `Text`, et le nom d'un fichier qui contient un index de séquences du texte à modifier `Index`, et construit le texte résultat `Result`. L'index a été typiquement construit par un des programmes de reconnaissance (`recon`, `reconind`, `recor` ou `recorind`).

Lorsqu'on utilise l'interface d'INTEX, les modifications à apporter au texte (pour construire le texte résultat) sont de deux types : soit on remplace les séquences reconnues par les séquences produites par le transducteur ; soit on insère les séquences produites par le transducteur dans le texte.

`etiqa S-Dic C-Dic E-Dic Text FST-Text Stats`

`etiqa` est utilisé pour construire le transducteur du texte. `S-Dic` est le dictionnaire qui contient tous les mots simples du texte (Sous l'interface INTEX, il a été construit avec le programme dicos) ; `C-Dic` est le dictionnaire qui contient tous les mots composés du texte (Sous INTEX, il a été construit avec `dicoc`) ; `E-Dic` est le dictionnaire qui contient toutes les expressions figées du texte (Sous INTEX, il a été construit avec `dicoe`) ; `Text` est le texte (qui doit être au format « `.snt` ») ; le résultat est le `FST-text` ; `Stats` contient quelques mesures des résultats (nombre d'états et de transitions).

```
etiqc [cdl] Text TaggedText C-Dictionaries...
```

`etiqc` est utilisé pour reconnaître et étiqueter les mots composés d'un texte.

`[cdl]` : type du texte (« c » : concordance, « d » : texte délimité, « l » : ligne par ligne) ;

Le second paramètre est le texte ; le nom du fichier texte étiqueté est donné en troisième paramètre.

Les paramètres suivants sont les noms des dictionnaires de mots composés (fichiers d'extension « `.bin` », « `.dic` » ou « `.fst` ») :

- Si le premier caractère du nom d'un dictionnaire est « ~ », alors le dictionnaire est recherché dans le dossier **Delacf** de la langue courante, d'abord dans le dossier INTEX de l'utilisateur, puis dans le dossier INTEX de l'application. Par exemple, « **~Noms.bin** » est une référence à un dictionnaire « **Noms.bin** » rangé dans le dossier « **c:\My Intex\French\Delacf** », ou alors dans le dossier « **c:\Program files\Intex\Delacf** ».

- Si le caractère qui précède le point (l'extension du nom de fichier) est un caractère « - », alors le dictionnaire est appliqué en premier ; si le caractère est un caractère « + », le dictionnaire est appliqué en dernier.

```
etiqg [Ddl] [01234567] Text FST S-Dic C-Dic NAC-Dic Result
Stats
```

Programme d'étiquetage de textes avec levée d'ambiguïté par grammaire locale.

L'étiquetage consiste à remplacer les formes simples non ambiguës, i.e. associées à une seule information lexicale, par l'information lexicale elle-même écrite entre accolades ; même chose pour les séquences de formes simples qui correspondent à des mots composés non ambigus.

Le premier paramètre « Ddl » correspond au type du texte :

- « D » : on étiquette un dictionnaire de mots composés ;

- « d » : on étiquette un texte au format « `.snt` » ;

- « l » : on étiquette un texte ligne par ligne (ou paragraphe par paragraphe).

Le second paramètre correspond au type d'étiquetage :

- « 0 » : les informations lexicales sont présentées entre parenthèses, après les formes étiquetées (utilisé pour étiqueter un dictionnaire de mots composés) ;
- « 1 » : les étiquettes contiennent toute l'information lexicale ;
- « 2 » : idem ; on étiquette les mots composés ambigus ;
- « 3 » : les étiquettes contiennent le lemme seul (i.e. on lemmatise le texte) ;
- « 4 » : idem ; on étiquette aussi les mots composés ambigus ;
- « 5 » : les étiquettes contiennent le lemme et la catégorie ;
- « 6 » : idem ; on étiquette aussi les mots composés ambigus ;
- « 7 » : le résultat est une expression rationnelle (i.e. on étiquette aussi les formes ambiguës).

La grammaire locale de levée d'ambiguïté est un `FST` (quatrième paramètre). Sous l'interface INTEX, c'est en général l'union des grammaires locales sélectionnées. Les deux paramètres suivants sont les dictionnaires du texte : des mots simples `S-Dic` et des mots composés `C-Dic` (en général, construits avec `dicos` et `dicoc`). Le paramètre suivant `NAC-Dic` est le nom d'un dictionnaire utilisé pour traiter en priorité certaines séquences et formes non-ambiguës ; le texte étiqueté est donné ensuite `Result` ; quelques comptages sont écrits dans le fichier `Stats` (nombre de séquences désambiguïsées ; nombre d'incohérences).

```
flexion FST-directory DELAS DELAF errors
```

Programme de flexion automatique d'un dictionnaire DELAS. Le premier paramètre est le nom du dossier dans lequel tous les transducteurs de flexion (fichiers à extension « .fst ») sont rangés ; le résultat consiste en deux fichiers : un fichier de type DELAF et un fichier qui contient toutes les entrées du dictionnaire DELAS qui n'ont pas pu être fléchies.

```
fst2txt [cdl] [gp] [fsr] FST Text S-Dic NAC-Dic AC-Dic ResText
```

Application d'un transducteur à un texte

`[cdl]` : type du texte (« c » : concordance, « d » : texte délimité, « l » : ligne par ligne) ;

`[gp]` : on donne priorité aux séquences reconnues les plus longues (« g ») ou les plus petites (« p ») ;

`[fsr]` : on insère les séquences produites par le transducteurs dans le texte (« f ») ; on remplace les séquences reconnues par les séquences produites (« s ») ; on ne tient pas compte des séquences produites (« r ») ;

`FST` : le transducteur à appliquer au texte ;

`Text` : le nom du fichier-texte (fichier d'extension « .txt » ou « .snt ») ;

`S-Dic` : le dictionnaire des mots simples du texte ;

`NAC-Dic` : le dictionnaire des mots composés non ambigus du texte ;

`AC-Dic` : le dictionnaire des mots composés ambigus du texte ;

`ResText` : le texte résultat.

`genere FST Limit Format Language`

On explore un transducteur en générant les chemins possibles.

`FST` : le premier argument est soit un graphe (extension « .grf »), soit un transducteur (extension « .fst ») ;

`Limite` : si ce paramètre est 0, on cherche à générer tous les chemins possibles. Si ce paramètre est positif, on arrête l'exploration du transducteur après avoir généré un nombre donné de séquences ; si ce paramètre est négatif, on arrête l'exploration du transducteur après un nombre de secondes donné ;

`Language` : le fichier qui contient toutes les séquences reconnues par le transducteur.

`Format` : le format du résultat. On peut choisir de représenter les séquences reconnues & produites de quatre façons :

1 : reconnues => produites ; par exemple : `tables => table.N:fs`

2 : dans le format DELAF ; par exemple : `tables,table.N :fs`

3 : dans le format DELACF ; par ex. : `tables rondes,table ronde.N:fp`

4 : séquences reconnues synchronisées avec séquences produites ; par ex. : `la_<PRO> vole_<V>.` (si la séquence reconnue est « *la vole* » et la séquence produite est « <*PRO*> <*V*> »)

`gr2fst GraphDir Graph Fst Determ?`

Programme de compilation d'un transducteur fini (fichier d'extension « .fst ») à partir d'un graphe (et de tous les graphes imbriqués).

`GraphDir` : dossier dans lequel le graphe est rangé (c'est aussi dans ce dossier qu'INTEX commence sa recherche des graphes imbriqués)

`Graph` : nom du fichier graphe

`FST` : le transducteur résultat ;

`Determ?` : « yes » si l'on veut déterminiser le transducteur ; « no » sinon.

`index [cdl] Text Keys Addr Stats TokensList CharsList`

Programme d'indexation d'un texte `Text`. Les clés de l'index sont les lexèmes *(token*s) qui appartiennent à quatre classes :

- les séquences de lettres (formes simples) ;
- les chiffres (de « 0 » à « 9 ») ;
- les étiquettes (toute séquence de caractères entre deux accolades « { » et « } »)
- les séparateurs (i.e. tous les caractères qui ne sont ni des lettres, ni des chiffres, ni des blancs).

`[cdl]` : type du texte (« c » : concordance, « d » : texte délimité, « l » : ligne par ligne) ;

L'index est construit dans deux fichiers : `Keys` contient tous les lexèmes, et `Addr` contient la liste des adresses de toutes les occurrences correspondantes dans le texte.

`TokensList` contient la liste des 100 lexèmes les plus fréquents ;

`CharsList` contient la liste de tous les caractères trouvés dans le fichier texte.

```
inter Grammar FST-Text ResFST-Text Stats
```

Programme de désambiguïsation d'un transducteur de texte.

inter prend une grammaire `Grammar` représentée par un transducteur de levée d'ambiguïté et un transducteur de texte `FST-Text`, et construit l'intersection de ces deux transducteurs, i.e. le nouveau transducteur du texte `ResFST-Text` compatible avec la grammaire.

La grammaire est un transducteur au format « `.fst` » ; en général, calculé à partir de l'union de tous les transducteurs sélectionnés (eux-mêmes compilés à partir de graphes).

Le `FST-Text` est lui-même construit avec le programme `etiqa`. Il contient précisément un transducteur par phrase du texte.

```
next2iso next-file iso-file
```

Programme de conversion de textes au format NextStep OpenStep vers le format Windows ANSI.

```
parse [WPS] [SLA] [limit] Grammar FST-Text ResText Stats
```

Analyseur syntaxique d'INTEX. Construit l'arbre qui représente la structure de la phrase (mode « normal ») ou l'arbre de dérivation de l'analyseur (mode « debug »).

`Grammar` est un graphe qui contient en général d'autres graphes imbriqués ; la grammaire décrite peut être une grammaire CF (Context Free). La production de la grammaire est utilisée pour représenter la structure des séquences reconnues : typiquement, une grammaire reconnaît des séquences et produit des parenthèses étiquetées autour des constituants caractéristiques (syntagmes, groupes nominaux, modifieurs).

Le texte `FST-TEXT` est lui-même représenté par un transducteur. Il est construit en général par application du programme `etiqa`, et éventuellement du programme `interg`.

`[WPS]` : la grammaire est appliquée à toute la phrase (« W ») ; au début de chaque phrase (« P ») ou la fin de chaque phrase (« S »).

Attention au paramètre « W » : ne pas oublier d'inclure dans la grammaire l'éventuel signe de ponctuation de fin de phrase, sinon aucune phrase ne sera reconnue.

`[SLA]` : on indexe les séquences reconnues les plus courtes (« S »), les plus longues (« L ») ou toutes les séquences (« A »).

Le fichier résultat `ResText` contient la liste des séquences reconnues dans lesquelles ont été insérées les marques produites par la grammaire (les parenthèses étiquetées). Chacune des séquences structurées peut être ensuite visualisée sous forme d'arbre par INTEX.

```
re2fst RegExp FST.
```

Construit un automate déterministe minimal `FST` à partir d'une expression rationnelle `RegExp`.

```
recondic FST Dictionary IncorrectEntries
```

Applique une grammaire à un dictionnaire. La grammaire est automate représenté par un fichier « .fst » (les productions éventuelles sont ignorées).

Typiquement, la grammaire décrit le format valide des entrées du dictionnaire. Toutes les entrées du dictionnaire qui ne sont pas reconnues sont écrites dans le fichier résultat `IncorrectEntries`.

Les quatre programmes suivants : `recon`, `reconind`, `recor` et `recorind` implémentent des variantes de la même fonctionnalité : il s'agit d'appliquer une grammaire à un texte. `recon` et `reconind` appliquent un transducteur fini « .fst » au texte ; `recor` et `recorind` appliquent un graphe « .grf » (qui contient des imbrications éventuellement) au texte. `recon` et `recor` appliquent la grammaire au texte lui-même, tandis que `reconind` et `recorind` utilisent l'index du texte pour accélérer la recherche.

```
recon [cdl] [gpt] [fsr] Limit FST Text S-Dic NAC-Dic AC-Dic
ResIndex Stats
```

Applique une grammaire `FST` à un texte `Text`. La grammaire `FST` est un transducteur représenté par un fichier « .fst ».

`[cdl]` : type du fichier texte (« c » : concordance au format « .con »; « d » : texte délimité au format « .snt » ; « l » : texte Windows ANSI au format « .txt » (traité ligne par ligne) ;

`[gpt]` : on indexe les séquences reconnues les plus longues (« g »), les séquences reconnues les plus courtes (« p »), ou toutes les séquences reconnues (« t ») ;

`[fsr]` : on insère les productions du transducteur dans le texte (« f ») ; on remplace les séquences reconnues par les séquences produites par le transducteur (« s ») ; on ne tient pas compte des productions du transducteur (« r ») ;

`Limit` : on interrompt la recherche après avoir trouvé un certain nombre de séquences. Si `Limit = 0`, alors on recherche toutes les séquences du texte ;

Lors de l'application du transducteur au texte, on utilise éventuellement des ressources lexicales qui sont rangées dans les trois dictionnaires du texte :

`S-Dic` : dictionnaire des mots simples du texte ;

`NAC-Dic` : dictionnaire des mots composés du texte non ambigus ;

`AC-Dic` : dictionnaire des mots composés du texte ambigus.

Le résultat du traitement est l'index des séquences reconnues `ResIndex`.

```
reconind [cdl] [gpt] [fsr] Limit FST Text S-Dic NAC-Dic AC-Dic
ResIndex Stats
```

Applique une grammaire `FST` à un texte indexé `Text`. L'index du texte est rangé dans deux fichiers de nom « **idx** » (la liste des lexèmes du texte) et « **ida** » (les adresses des occurrences des lexèmes dans le texte) trouvés dans le dossier du texte (on suit la convention INTEX). Par exemple, si le texte est rangé dans le fichier :

```
c:\Intex\French\Corpus\toto.snt
```

alors les deux fichiers suivants doivent exister :

```
c:\Intex\French\Corpus\toto_snt\idx
```

```
c:\Intex\French\Corpus\toto_snt\ida
```

La grammaire `FST` est un transducteur représenté par un fichier « .fst » qui a été compilé soit à partir d'une expression rationnelle, soit à partir d'un graphe.

`[cdl]` : type du fichier texte (« c » : concordance au format « .con »; « d » : texte délimité au format « .snt » ; « l » : texte Windows ANSI au format « .txt » (traité ligne par ligne) ;

`[gpt]` : on indexe les séquences reconnues les plus longues (« g »), les séquences reconnues les plus courtes (« p »), ou toutes les séquences reconnues (« t ») ;

`[fsr]` : on insère les productions du transducteur dans le texte (« f ») ; on remplace les séquences reconnues par les séquences produites par le transducteur (« s ») ; on ne tient pas compte des productions du transducteur (« r ») ;

`Limit` : on interrompt la recherche après avoir trouvé un certain nombre de séquences. Si `Limit = 0`, alors on recherche toutes les séquences du texte ;

Lors de l'application du transducteur au texte, on utilise éventuellement des ressources lexicales qui sont rangées dans les trois dictionnaires du texte :

`S-Dic` : dictionnaire des mots simples du texte ;

`NAC-Dic` : dictionnaire des mots composés du texte non ambigus ;

`AC-Dic` : dictionnaire des mots composés du texte ambigus.

Le résultat du traitement est l'index des séquences reconnues `ResIndex`

```
recor [cdl] [gpt] [fsr] Limit GRF Text S-Dictionary NAC-
Dictionary AC-Dictionary Index Stats
```

La grammaire `GRF` est un transducteur représenté par un fichier « .grf » qui peut éventuellement inclure des graphes imbriqués.

`[cdl]` : type du fichier texte (« c » : concordance au format « .con »; « d » : texte délimité au format « .snt » ; « l » : texte Windows ANSI au format « .txt » (traité ligne par ligne) ;

`[gpt]` : on indexe les séquences reconnues les plus longues (« g »), les séquences reconnues les plus courtes (« p »), ou toutes les séquences reconnues (« t ») ;

`[fsr]` : on insère les productions du transducteur dans le texte (« f ») ; on remplace les séquences reconnues par les séquences produites par le transducteur (« s ») ; on ne tient pas compte des productions du transducteur (« r ») ;

`Limit` : on interrompt la recherche après avoir trouvé un certain nombre de séquences. Si `Limit = 0`, alors on recherche toutes les séquences du texte ;

Lors de l'application du transducteur au texte, on utilise éventuellement des ressources lexicales qui sont rangées dans les trois dictionnaires du texte :

`S-Dic` : dictionnaire des mots simples du texte ;

`NAC-Dic` : dictionnaire des mots composés du texte non ambigus ;

`AC-Dic` : dictionnaire des mots composés du texte ambigus.

Le résultat du traitement est l'index des séquences reconnues `ResIndex`.

`recorind [cdl] [gpt] [fsr] Limit GRF Text S-Dictionary NAC-Dictionary AC-Dictionary Index Stats`

Applique une grammaire `GRF` à un texte indexé `Text`. L'index du texte doit être rangé dans deux fichiers de nom « **idx** » (la liste des lexèmes du texte) et « **ida** » (les adresses des occurrences des lexèmes dans le texte) trouvés dans le dossier du texte (on suit la convention INTEX).

La grammaire `GRF` est un transducteur représenté par un fichier « .grf » qui peut éventuellement inclure des graphes imbriqués.

`[cdl]` : type du fichier texte (« c » : concordance au format « .con »; « d » : texte délimité au format « .snt » ; « l » : texte Windows ANSI au format « .txt » (traité ligne par ligne) ;

`[gpt]` : on indexe les séquences reconnues les plus longues (« g »), les séquences reconnues les plus courtes (« p »), ou toutes les séquences reconnues (« t ») ;

`[fsr]` : on insère les productions du transducteur dans le texte (« f ») ; on remplace les séquences reconnues par les séquences produites par le transducteur (« s ») ; on ne tient pas compte des productions du transducteur (« r ») ;

`Limit` : on interrompt la recherche après avoir trouvé un certain nombre de séquences. Si `Limit = 0`, alors on recherche toutes les séquences du texte ;

Lors de l'application du transducteur au texte, on utilise éventuellement des ressources lexicales qui sont rangées dans les trois dictionnaires du texte :

`S-Dic` : dictionnaire des mots simples du texte ;

`NAC-Dic` : dictionnaire des mots composés du texte non ambigus ;

`AC-Dic` : dictionnaire des mots composés du texte ambigus.

Le résultat du traitement est l'index des séquences reconnues `ResIndex`.

```
table2fst MetaGraph Table ResFST
```

On construit le transducteur `ResFST` (grammaire au format « .fst ») équivalent à la table de lexique-grammaire `Table`, en utilisant un graphe patron `MetaGraph`.

```
tri [cdltr] Text ResText
```

Programme de tri. Noter que ce programme prend en compte la langue et l'alphabet courants (grâce à la variable INTEXLNG0) ; en particulier, le tri alphabétique dépend de l'ordre dans lequel les lettres ont été décrites dans le fichier **Alphabet** du dossier de la langue courante.

`[cdlt]` : le programme tri un dictionnaire au format DELACF (« c »), au format DELAF (« d »), une séquences de lignes (« l ») ou un texte ligne par ligne (« t »). La différence entre les deux options « l » et « t » : avec l'option « l » les caractères de ponctuation sont pris en compte ; avec l'option « t », on ne tient pas compte des ponctuations. Typiquement, l'option « l » est utilisée pour trier des listes de mots ou d'expresions, tandis que l'option « t » est utilisée pour trier des textes.

Le fichier résultat `ResText` ne contient jamais de doublons.

```
tri r column-number concordance result
```

La variante du programme de tri spécifiquement utilisée pour trier des concordances est appelée si le premier paramètre est « r ».

`column-number` : ce paramètre a six valeurs possibles : 123, 132, 213, 231, 312, 321. Ces valeurs correspondent à l'ordre de priorité dans le tri. Par exemple, 132 signifie qu'on trie les lignes de la concordance selon la première colonne ; si des lignes ont la même première colonne, alors on les trie en prenant en compte la troisième colonne ; si des lignes ont les mêmes première et troisième colonnes, alors on trie selon la seconde colonne.

```
verifg [Ddl] mode FST S-Dic C-Dic NAC-Dic Text ResIndex Stats
```

Ce programme est une sorte d'hybride entre le programme `recon` d'application d'un FST à un texte et le programme d'étiquetage `etiqg`.

Si `mode = 1`, (cela correspond à l'option « **display all matching sequences** » du module de levée d'ambiguïtés)le programme fonctionne comme `recon` : on indexe toutes les séquences reconnues par la grammaire `FST` dans le text `Text`.

Si `mode = 0` (option « **Display inconsistencies between LGs and Text** »), alors on indexe seulement les séquences reconnues par le transducteur `FST` qui ne sont pas compatibles avec les séquences de contraintes produites par le transducteur.

Le résultat du traitement est l'index `ResIndex` de toutes les séquences reconnues (mode = 1) ou incohérentes (mode = 0). Cet index a le même format que l'index produit par la commande `recon`.

Les autres paramètres sont identiques à ceux du programme `etiqg`.

3 EXEMPLES D'APPLICATIONS INFORMATIQUES DÉVELOPPÉES A PARTIR D'INTEX

Cette section vise à montrer comment INTEX peut être intégré dans des applications diverses traitant de textes en langage naturel. Thales Recherches et Technologies a développé des applications orientées vers des besoins utilisateur (filtrage de documents, extraction d'information, …). Dans ce cadre, INTEX est masqué et intégré de façon transparente. Les interfaces privilégient les traitements et les fonctionnalités possibles pour l'utilisateur.

3.1 Filtrage et routage de document : le projet CORAIL

Dans le cadre du projet CORAIL (qui regroupait, outre Thomson-CSF/LCR, Informatique CDC/DTA et Université Paris X/CRIS), le laboratoire de recherches de Thales a développé une plate-forme de filtrage d'information, intégrant INTEX. CORAIL permet de définir des filtres sous la forme de grammaires locales élaborées à l'aide de transducteurs à nombre fini d'états (on parle aussi de profil). L'application fonctionne sous Windows et permet à un utilisateur donné de recevoir dans sa boîte aux lettres les textes qui correspondent à son ou ses profils. Un démonstrateur a été réalisé pour l'analyse en temps réel du fil d'agence de presse AFP. La copie d'écran ci-dessous montre la boîte aux lettre d'un utilisateur abonné à différents profils (élections municipales, épizootie, marchés publics…). On voit souligné un passage pertinent identifié grâce à INTEX.

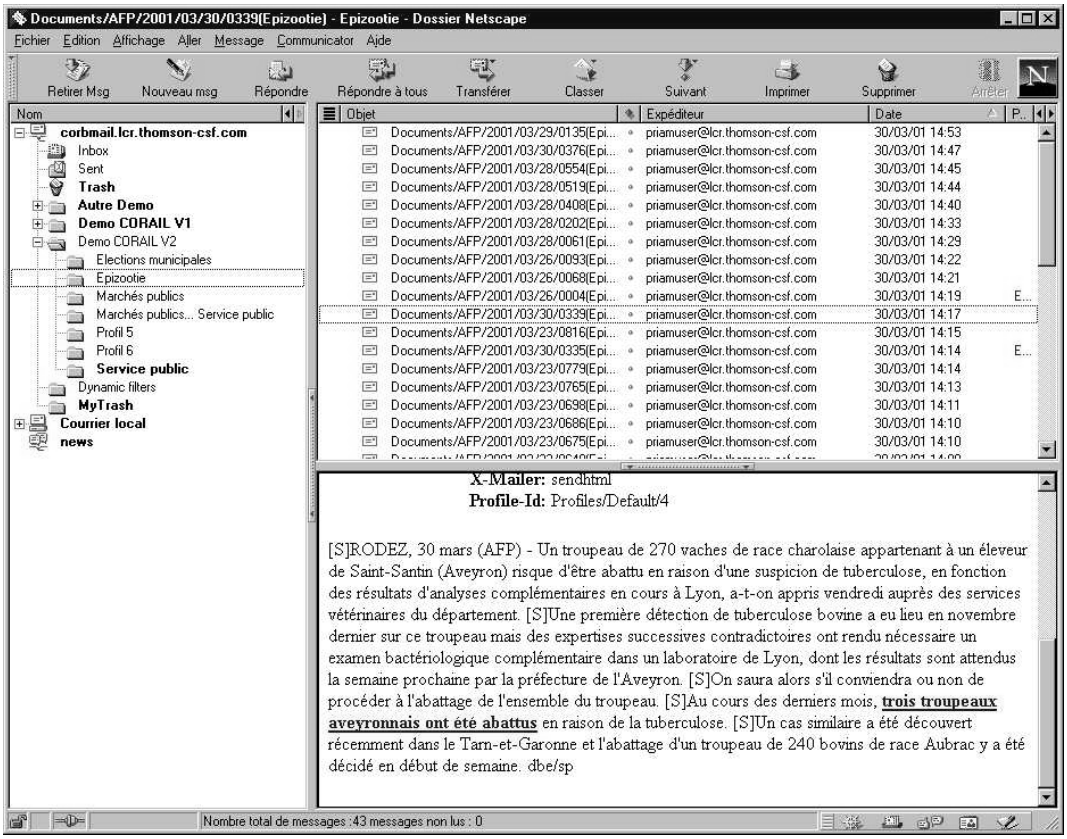

Figure 2 : Réception dans Netscape de dépêches AFP filtrées par le système CORAIL

3.1.1 Interface d'édition de filtres

Thales Recherches et Technologie a par ailleurs développé une version simplifiée de l'interface d'édition et de manipulation d'automates. Des études d'ergonomie menées dans le cadre du projet Corail ont montré qu'il était ainsi en partie possible à des utilisateurs non linguistes de développer leurs propres filtres, sans aide extérieure.

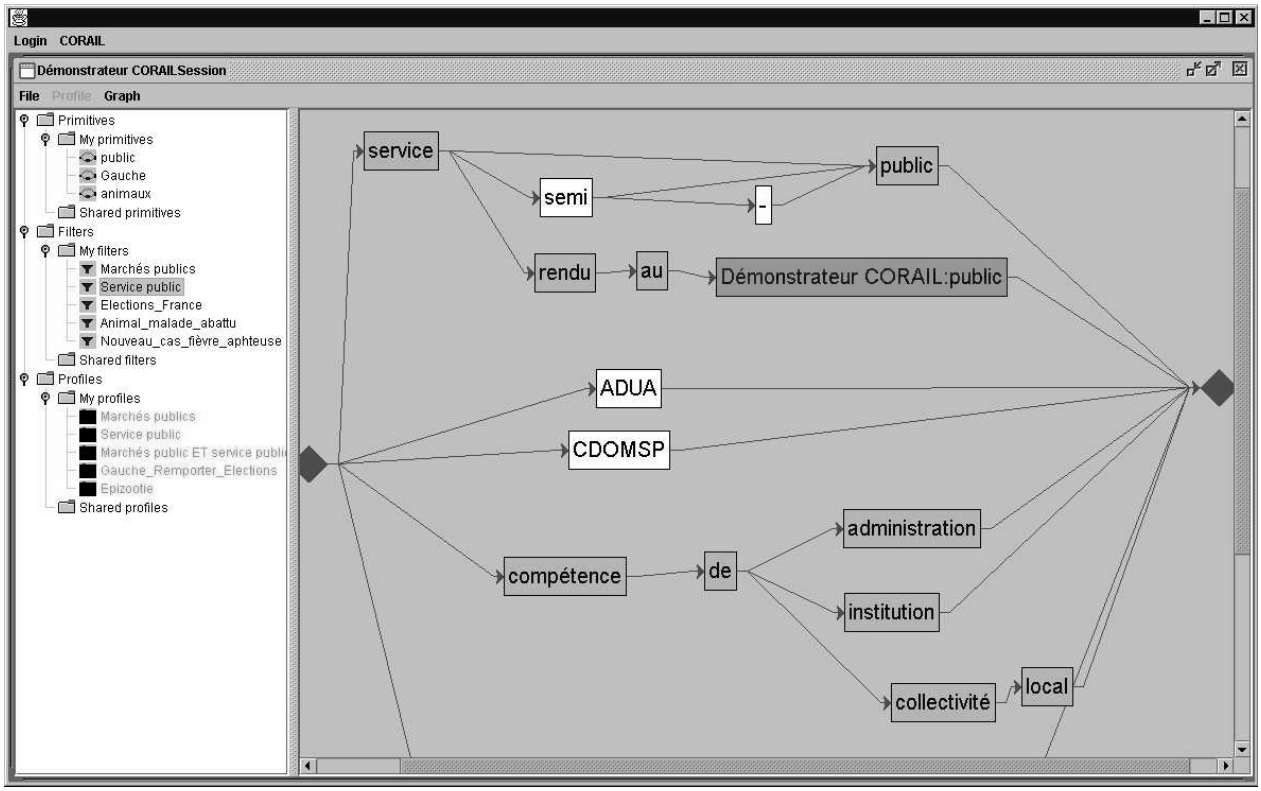

Figure 3 : Interface simplifiée d'édition de filtre

Bibliographie

Frédéric Meunier, Antonio Balvet, Thierry Poibeau, 1999. Projet CORAIL (COmposition de Requêtes par des Agents Intelligents Linguistiques), *Linguisticae Investigationes*, 22, pp.369-381.

Antonio Balvet, Frédéric Meunier, Thierry Poibeau, Didier Viard, Frantz Vichot et Francis Wolinski, 2001. « Filtrage de documents et grammaires locales : le projet CORAIL », *à paraître dans les actes de la conférence ISKO*, Nanterre, 5-6 juillet 2001

3.2 Extraction d'information

Dans de nombreux services de veille, les analystes sont confrontés à de large masse de textes qu'ils doivent analyser. La mise en surbrillance de certains éléments pertinents facilite l'analyse en guidant la lecture. La figure suivante présente l'analyse des entités nommées (noms de personnes et d'entreprise, dates, etc.) d'un texte. L'information est reprise par type d'entité dans la partie gauche. L'analyse des séquences pertinentes et leur mise en surbrillance grâce à l'insertion de balises HTML est effectuée en partie par INTEX.

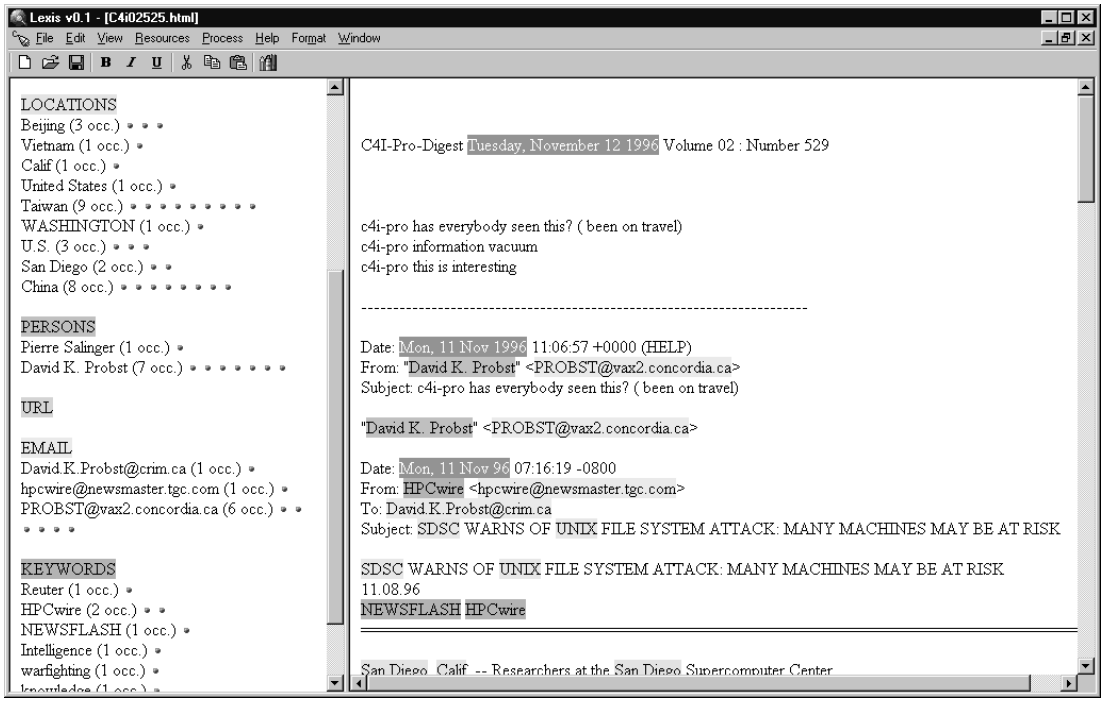

Figure 4 : Marquage des entités nommées

Il est aussi possible de remplir des bases de données avec de l'information structurée extraite de textes, comme sur la figure suivante.

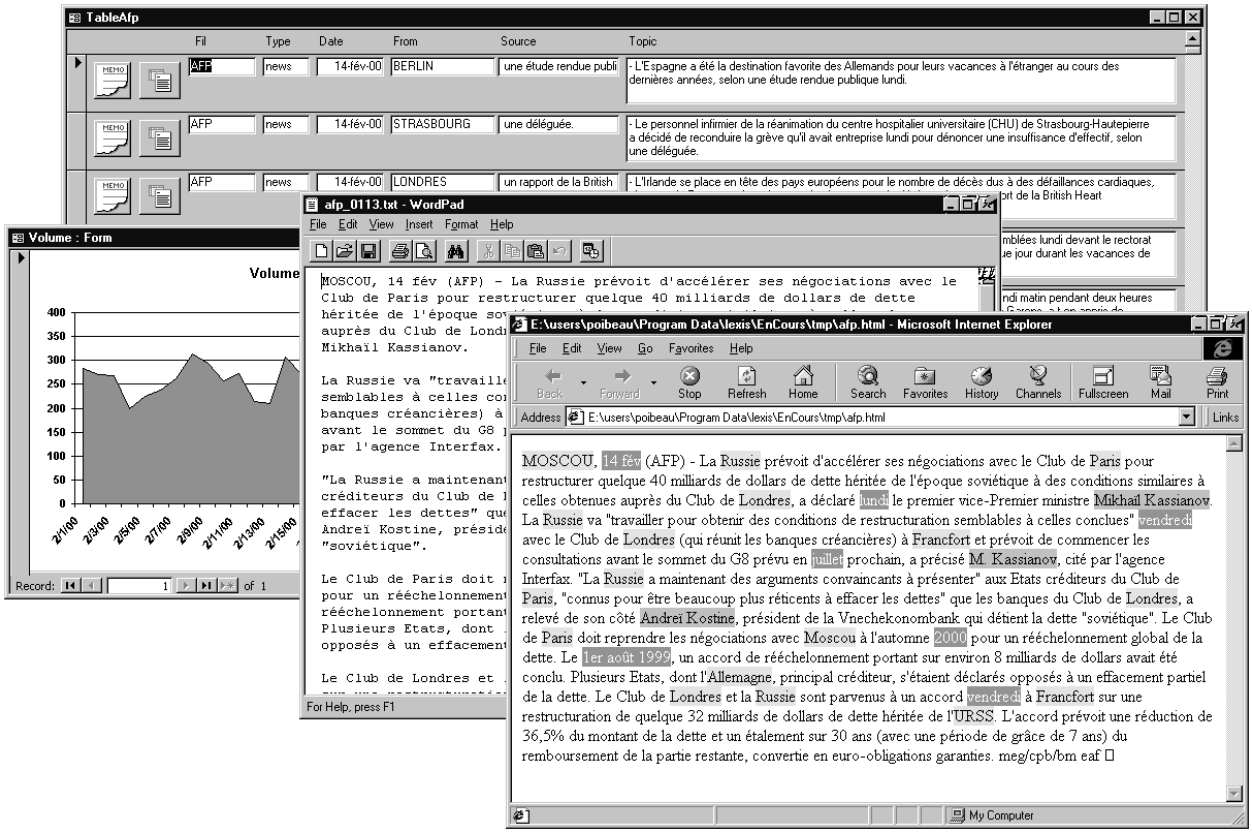

Figure 5 : Intégration de différents modules d'analyse de textes. Le texte source est analysé pour en extraire les entités nommées et remplir une base de données avec de l'information structurée. A partir de la base de données, il est possible d'accéder à plusieurs fonctions de suivi sur les champs structurés. Une telle analyse est bien sûr impossible sur du texte brut.

Bibliographie

Thierry Poibeau, 1999. « Repérage des entités nommées : un enjeu pour la veille technologique », *Terminologies Nouvelles* (actes de la conférence Terminologie et Intelligence Artificielle (TIA), Nantes, mai 1999), n°19, pp. 43-51

Thierry Poibeau, 2000. « Corpus-based learning for Information Extraction », *Actes du workshop Machine Learning for Information Extraction* (ML4IE), ECAI'2000, Berlin.