Alexandre Christie

Modèles mathématiques pour l'indexation assistée de documents législatifs communautaires

OA-76-06-410-FR-C

La plupart des algorithmes informatiques d'indexation automatique des documents sélectionnent des mots clés déjà présents dans les documents. L'expérience prouve qu'il est possible d'atteindre de meilleurs résultats en utilisant des mots sélectionnés dans un thésaurus et ne se trouvant pas nécessairement dans le texte.

Dès lors, comment la précision et la flexibilité d'une indexation peuvent-elles être combinées avec la rapidité et l'économie d'un ordinateur?

Alexandre Christie nous fournit la réponse en développant une approche mathématique de l'indexation par thésaurus qu'il programme ensuite en Java et en Visual Basic for Applications.

Cette approche rend l'indexation presque totalement indépendante de la langue utilisée. Procédé particulièrement utile au sein des organisations internationales ainsi que pour les systèmes d'information des parlements nationaux et pour toute organisation devant indexer de larges corpus de documents par thésaurus. L'algorithme de l'auteur est actuellement en phase d'intégration dans le processus d'indexation de la Commission européenne et devrait permettre des économies considérables en remplaçant la majeure partie du système actuel d'indexation manuelle.

Modèles mathématiques pour l'indexation assistée de documents législatifs communautaires s'articule en deux parties. La première, qui ne requiert pas de connaissances approfondies en mathématiques, expose le contexte de l'étude. La deuxième, plus pointue, donne dans le détail les modèles mathématiques et les algorithmes.

Alexandre Christie
Cette publication est fondée sur un mémoire de maîtrise qui a été soumis au département de mathématiques de l'université Henri Poincaré, Nancy I, en France. C'est en considération de ce travail que le jury a décerné à l'auteur le titre d'«Ingénieur-maître en génie mathématique», en février 2006.

Alexandre Christie a prolongé ses études au St John's College d'Oxford, au Royaume-Uni, et travaille actuellement dans le secteur des investissements financiers à New York et à Londres.

ISBN 92-894-4043-0
9 789289 404338

Office des publications
Publications.europa.eu

Alexandre Christie

Modèles mathématiques pour l'indexation assistée de documents législatifs communautaires

Office des publications
Publications.europa.eu

Alexandre Christie

Modèles mathématiques pour l'indexation assistée de documents législatifs communautaires

Office des publications
Publications.europa.eu

Cette publication est fondée sur un mémoire de maîtrise qui a été soumis au département de mathématiques de l'université Henri Poincaré, Nancy I, en France. C'est en considération de ce travail que le jury a décerné à l'auteur le titre national d'«Ingénieur Maître en Génie Mathématique», en février 2006.

Alexandre Christie a prolongé ses études au St John's College d'Oxford, au Royaume-Uni, et travaille actuellement dans le secteur des investissements financiers à New York et à Londres.

De nombreuses autres informations sur l'Union européenne sont disponibles sur l'internet via le serveur Europa (http://europa.eu).

Une fiche bibliographique figure à la fin de l'ouvrage.

Luxembourg: Office des publications officielles des Communautés européennes, 2009

ISBN 92-78-40433-0

Printed in Luxembourg

Table des matières

Préface

Trouver l'information utile sur les sites multilingues de l'Union européenne: une gageure?

Les institutions de l'Union européenne (UE) rédigent et diffusent, aujourd'hui le plus souvent sur l'internet, un volume sans cesse croissant de textes de divers types. Sur le support matériel du papier, ces textes prenaient la forme de livres, magazines, newsletters, brochures, dépliants, journaux, annuaires, etc. Ces quelque 7 000 publications annuelles sont identifiées, indexées et cataloguées par l'Office des publications officielles des Communautés européennes pour pouvoir être retrouvées, vendues, conservées. Les codes à barres qui figurent sur les couvertures en attestent.

Les textes publiés au Journal officiel de l'Union européenne, *eux aussi édités par l'Office des publications, sont traduits dans toutes les langues officielles de l'Union, soit vingt-trois depuis le début de 2007. Les différentes versions linguistiques paraissent simultanément dans les éditions du Journal officiel. Il est en effet normal qu'un texte ayant des effets juridiques sur les citoyens européens soit intelligible par chacun d'eux. C'est un postulat, peut-être coûteux, mais primordial dans un État de droit, pour une organisation démocratique.*

Au-delà de cet aspect linguistique, il convient que chacun de ces textes officiels puisse être retrouvé aisément. C'est une gageure dans un monde textuel et institutionnel, bavard et spécialisé, dont les documents sont parfois rédigés dans un jargon technique et juridique mal maîtrisé par les citoyens, voire inconnu de ces derniers.

Si ce n'est pour des références très précises de documents (un numéro de règlement par exemple), les moteurs de recherche universellement disponibles sur nos écrans conduisent à des résultats inexploitables. Les réponses sont en effet trop nombreuses et ne correspondent pas au sujet de la recherche, qui est souvent formulée de façon trop imprécise. Comment aider le citoyen dans son libellé? C'est l'objet d'une liste hiérarchisée «d'expressions destinées à représenter sans ambiguïté les concepts contenus dans les documents et dans les questions posées» sur l'internet, dénommée thésaurus dans le langage du monde de la recherche d'information. Le Parlement européen et l'Office des publications, en relation étroite avec les parlements nationaux de l'Union, ont développé un tel outil ([1]). Eurovoc a été élaboré sur une base multilingue; tous les concepts ont été énoncés dans les différentes langues européennes afin de permettre de formuler une recherche dans sa propre langue pour retrouver un document dans une autre langue ou pour pouvoir comparer les traductions d'un document.

Mais encore faut-il affecter à chaque texte paru dans les quelque 750 000 pages produites chaque année au Journal officiel les expressions univoques de ce thésaurus, indexer les documents préparatoires à la législation, la jurisprudence de la Cour de justice des Communautés européennes… Cette tâche immense, aujourd'hui confiée à des documentalistes et à des juristes, est à la fois lente (une huitaine de jours) et très coûteuse (1,5 million d'euros par an).

Assister ces «humains» à l'aide d'un automate: l'idée s'est imposée peu à peu. À cet effet, la conjugaison de compétences transversales était nécessaire. Un logiciel répondant à ces exigences a été élaboré puis validé par Alexandre Christie, brillant mathématicien, alors étudiant à l'université Henri Poincaré, Nancy I, en France, aidé par son directeur de stage, le D^r^ Holger Bagola, expert dans le domaine linguistique et en informatique appliquée en linguistique.

Jacques Raybaut
ancien directeur
de la direction «Journal officiel et accès au droit»
de l'Office des publications

([1]) Thésaurus multilingue Eurovoc (http://europa.eu/eurovoc/).

La nature du problème de l'indexation

Il est avéré que les utilisateurs de l'internet ont souvent recours aux moteurs de recherche. Quelle que soit la quête de l'utilisateur, ces moteurs de recherche sont des outils indispensables qui lui permettent de naviguer dans la masse désorganisée d'informations proposées par l'internet. Ainsi, vous reconnaîtrez le scénario suivant.

Vous lancez votre navigateur internet préféré, vous naviguez vers un moteur de recherche, et une zone de texte apparaît. Après avoir tapé quelques termes bien choisis, vous cliquez sur «Recherche», en espérant obtenir une réponse pertinente à votre question. Le plus souvent, le moteur de recherche vous renvoie plusieurs dizaines de pages, chacune ayant une dizaine de liens vers les pages que le système estime être les plus pertinentes pour votre requête.

C'est alors que le problème du filtrage commence. Devant le nombre de résultats, il faut trier, et ce n'est pas toujours facile. Parfois, on peut éliminer certains liens, car le titre ou le texte qui les accompagnent suffisent pour savoir que le lien ne nous intéresse pas. Mais dans d'autres cas, il faut cliquer sur le lien et parcourir le document associé avant de savoir s'il est pertinent.

Cette procédure, qui compare les mots du texte et les mots clés définis par les auteurs des documents avec les termes de la requête, est coûteuse pour l'utilisateur, et ne fournit pas toujours les résultats souhaités.

Un autre problème que l'on rencontre souvent lors d'une recherche sur l'internet est que seuls les documents contenant les termes tels que l'utilisateur les a saisis sont sélectionnés. Les documents contenant le même terme, mais dans une autre langue, ne sont pas sélectionnés, et le chercheur perd ainsi un grand nombre de résultats qui auraient pu être pertinents. Dès lors, comment modifier ces algorithmes pour permettre une recherche multilingue, tout en améliorant l'efficacité et en réduisant le nombre de résultats inutiles?

Les problèmes d'une recherche efficace et multilingue sont particulièrement ressentis au sein de l'Office des publications officielles des Communautés européennes, qui, en tant qu'éditeur officiel de l'Union européenne, doit gérer plusieurs centaines de documents tous les jours dans toutes les langues officielles.

I. Introduction

1. Résumé du projet

Eurovoc est un thésaurus multilingue, c'est-à-dire un ensemble de termes décrivant les différents aspects de la législation communautaire. Dans le but de faciliter la recherche automatique de documents, l'Office emploie des experts analystes pour associer à chaque document législatif une liste de mots clés. Cette liste de mots clés est sélectionnée parmi les termes contenus dans Eurovoc, et le processus d'association de mots clés à des textes s'appelle l'«indexation». L'indexation effectuée par les analystes étant longue et coûteuse, l'Office nous a demandé d'étudier le potentiel des méthodes d'analyse de données, qui pourraient servir de base à un logiciel d'indexation automatique.

Il existe deux façons d'indexer un document: par extraction (où les mots clés sont choisis parmi les termes du texte), ou par association (où les mots clés n'apparaissent pas forcément dans le texte). Une étude préliminaire nous a permis de constater que les méthodes d'extraction ne sont pas viables dans ce cas. La plus grande partie de la recherche actuelle étant basée sur le principe de l'extraction, nous avons construit des modèles d'indexation par association en utilisant des algorithmes de catégorisation, ainsi que leur implémentation Java ([1]) dans l'environnement Weka ([2]).

Nous avons notamment été confronté au problème de la modélisation d'un texte. N'ayant jamais utilisé des algorithmes de catégorisation avec des textes, il a fallu expérimenter plusieurs approches avant de trouver une modélisation efficace.

Au cours de notre étude, nous avons essayé plusieurs algorithmes et paramétrages différents. Nous avons d'abord tenté d'associer chaque texte à une classe représentant l'ensemble des mots clés qui lui ont été associés. Cette approche nous a permis d'obtenir un taux de bonne classification entre 6 et 30 %.

Nous avons alors choisi d'aborder le problème dans le sens inverse: au lieu d'associer chaque texte à une classe, nous avons tenté d'associer chaque terme contenu dans le thésaurus à l'ensemble des textes qui lui ont été associés par les analystes. Cette dernière stratégie nous a permis de construire des modèles beaucoup plus efficaces, ayant un taux de bonne classification variant entre 90 et 97 %.

À l'issue de cette analyse, nous avons créé un programme en Java qui implémente l'algorithme de catégorisation qui fournit les meilleurs résultats. Ce programme permet à la fois d'analyser la qualité de l'indexation automatique en la comparant à une indexation connue et d'indexer automatiquement des documents dont on ne connaît pas l'indexation. Cette dernière fonction permet, entre autres, d'identifier clairement les textes et les termes qui ont été incorrectement associés ainsi que le type d'erreur commise et les manières dont on peut améliorer l'algorithme.

2. Organisation du présent document

Le présent document décrit les étapes de notre projet en détail. Nous y expliquons tout d'abord le rôle et le fonctionnement de l'Office des publications. Nous présentons ensuite les résultats de nos investigations sur les systèmes de recherche actuellement utilisés par les bibliothécaires dans la recherche documentaire. Après, nous explorons les méthodes d'indexation utilisées par l'Office des publications et nous expliquons l'organisation des documents législatifs sur lesquels nous avons travaillé.

Nous expliquons ensuite la construction de notre modèle mathématique d'indexation, et nous précisons les différentes étapes dans l'élaboration des règles d'association que nous utilisons. Nous décrivons les concepts mathématiques qui nous ont permis de mettre au point notre algorithme final et nous expliquons la façon dont nous avons créé un programme d'indexation automatique.

Enfin, nous analysons les résultats et nous explorons différentes façons de rendre notre programme plus efficace.

([1]) Java 2, Sun Microsystems.

([2]) Witten, I. H., et Frank, E., «Data mining: practical machine learning tools and techniques», 2nd edition, Morgan Kaufmann, San Francisco, 2005.

II. Les publications de l'Union européenne

1. L'Office des publications officielles des Communautés européennes ([1])

L'Office des publications officielles des Communautés européennes est l'éditeur officiel des institutions européennes (qui comprennent le Parlement européen, le Conseil, la Commission, la Cour de justice et la Cour des comptes), ainsi que des autres organisations de l'Union européenne (dont les agences décentralisées). Sa mission est de publier et de distribuer les publications officielles de l'Union européenne sur tous les supports, dont les formats papier et multimédia, et dans les 23 langues officielles de l'Union européenne ([2]).

Bien que l'Office des publications n'ait été officiellement établi comme organisme indépendant qu'en 1969, ses origines remontent au service des publications de la Communauté européenne du charbon et de l'acier, qui a publié le *Journal officiel de la Communauté européenne du charbon et de l'acier* depuis 1952. Par conséquent, depuis le 30 décembre 2002, les archives de l'Office des publications contiennent toutes les publications européennes issues d'un demi-siècle d'édition, la première édition du Journal officiel ayant été publiée en quatre langues (allemand, français, italien et néerlandais) le 30 décembre 1952.

Les documents publiés par l'Office aujourd'hui émanent des institutions, agences et organes européens, et une partie d'entre eux paraît chaque jour dans le Journal officiel (JO — voir le point suivant), dans 22 langues officielles de l'Union, ainsi qu'en irlandais pour le droit européen primaire. Ces textes sont à l'usage des institutions elles-mêmes, mais également des citoyens européens et des autres citoyens de par le monde qui ont besoin de la législation et d'informations sur le fonctionnement de l'Union européenne. La plupart des documents sont d'abord rédigés en anglais ou en français et plus rarement en allemand, les trois principales langues de travail au sein des institutions européennes, et sont ensuite traduits dans les autres langues de l'Union.

Une activité d'une telle ampleur est unique dans le monde de l'édition. L'élargissement de l'Union de 2007 a accueilli la Bulgarie et la Roumanie et, partant, deux nouvelles langues. Lors de l'élargissement de 2004, quelque 800 000 pages — en 9 langues — de la législation communautaire alors en vigueur (également appelée «acquis communautaire») ont été publiées. Au 1er janvier 2007, ce sont quelque 90 000 pages de l'acquis en vigueur qui devaient être publiées en bulgare et en roumain.

L'Office contribue ainsi aux efforts des institutions européennes pour rendre plus transparent le processus législatif et rendre plus accessibles les informations relatives à la législation européenne, dans le but de rapprocher l'Europe de ses citoyens.

Bien que ceci sorte du cadre de notre projet, il est intéressant d'ajouter que l'Office publie également des appels d'offres pour des services et biens à caractère public, et plus de 1 000 appels d'offres sont actuellement publiés (toutes langues confondues) tous les jours ouvrables.

2. Le Journal officiel

C'est dans le cadre de ce grand nombre de publications que les techniques mathématiques viennent prêter main-forte aux documentalistes. Afin de pouvoir expliquer au mieux la mission que nous avons entrepris d'accomplir au sein de l'Office des publications, il faut tout d'abord expliquer ce qu'est le Journal officiel, qui contient les documents sur lesquels nous avons travaillé.

Le *Journal officiel de l'Union européenne* (JO) est la publication produite par l'Office des publications qui rassemble les textes de la législation communautaire en vigueur, ainsi que d'autres informations à caractère officiel des institutions. C'est la seule publication au monde qui soit produite tous les jours ouvrables dans 22 langues plus, le cas échéant, en irlandais. Elle contient deux séries, dont la série L (législation) sur laquelle nous avons travaillé et qui comprend:

- les règlements;
- les directives;
- les décisions;
- les recommandations;
- les interprétations officielles des éléments ci-dessus.

Nous avons également travaillé sur le *Répertoire de la législation communautaire en vigueur* (édition 2001). Ce répertoire recense les références des textes législatifs ainsi que des amendements éventuels. Il contient également les références des accords et des conventions qui ont été signés par l'Union européenne dans le contexte de ses relations extérieures, des traités de l'UE, des actes complémentaires tels que ceux issus du Conseil des ministres et des chefs d'État ou de gouvernement, et des autres actes liés. Le Répertoire nous a permis de sélectionner les chapitres sur lesquels nous allions mener notre étude.

([1]) Cette partie est inspirée du site http://publications.europa.eu/index_fr.htm

([2]) Les 23 langues officielles sont l'allemand, l'anglais, le bulgare, le danois, l'espagnol, l'estonien, le finnois, le français, le grec, le hongrois, l'irlandais, l'italien, le letton, le lituanien, le maltais, le néerlandais, le polonais, le portugais, le roumain, le slovaque, le slovène, le suédois et le tchèque.

3. Le problème de l'indexation

Le lecteur se rendra compte à ce stade de l'ampleur de la tâche qui incombe à l'Office des publications. Pour gérer une telle multitude de documents, l'Office se doit d'avoir un système d'archivage organisé et efficace, afin de permettre aux utilisateurs externes de pouvoir trouver tout document recherché dans les meilleurs délais. La question est à présent de déterminer le système d'archivage et de recherche le mieux adapté à cette situation.

III. La recherche de documents

1. Les algorithmes traditionnels

Avant de tenter d'apporter quelques éléments de réponse, nous voudrions tout d'abord dire un mot sur les techniques utilisées par les moteurs de recherche traditionnels. Ce type de recherche utilise une méthode dite *«full text»*, c'est-à-dire que le moteur essaie de trouver les termes recherchés dans les corps des textes. Cet algorithme de recherche est aujourd'hui utilisé dans la plupart des systèmes de recherche d'information électroniques. Entre autres, on trouve ce type d'algorithme de recherche dans les cas suivants:

- lors de la recherche sur l'internet. Les moteurs de recherche sur l'internet comptent parmi les plus gros systèmes de recherche;
- lors de la recherche de documents. La plupart des systèmes d'exploitation aujourd'hui permettent ce type de recherche. On peut ainsi rechercher des fichiers contenant certains termes. Certains systèmes permettent même de rechercher des expressions régulières, ce qui constitue une amélioration de la commande «grep» sur les systèmes Unix;
- certaines bibliothèques, dont la bibliothèque de la faculté des sciences de Nancy, publient leur catalogue sur l'internet et permettent de rechercher un mot parmi le titre ou même le contenu des livres et documents disponibles.

Bien que l'utilité et l'efficacité des algorithmes de recherche sur les textes complets soient reconnues depuis longtemps, dans certains cas, ce type d'algorithme n'est pas du tout efficace, et est même parfois totalement inapproprié. Les cas où ces techniques de recherche sont le moins adaptées sont ceux où le corpus ([1]) de recherche est très diversifié et volumineux, comme le World Wide Web. Les anciennes techniques, qui se basaient uniquement sur la recherche à texte complet, ont été abandonnées peu à peu, et de nouvelles techniques sont apparues, qui permettent de filtrer les résultats retournés par les algorithmes classiques. Ces techniques indexent tous les termes contenus dans un document, afin de pouvoir comparer ces termes indexés avec les termes d'une recherche.

L'inconvénient de ces nouvelles techniques est que les langues naturelles sont complexes et ambiguës, et la qualité des résultats retournés n'est pas toujours optimale. Parfois, un utilisateur recherchant un mot particulier ne trouvera pas les documents qui contiennent son synonyme, ou bien les documents trouvés pourront contenir le terme voulu, mais utilisé dans un autre sens que celui recherché par l'utilisateur. En revanche, ces techniques permettent de résoudre, au moins partiellement, le problème de la recherche sur le web, et elles représentent une solution pratique dans des contextes généraux.

([1]) Un corpus est un ensemble de textes.

Il y a également un autre cas où les techniques à texte complet ne sont pas efficaces: la recherche d'images et de son. La solution actuelle est d'associer à chaque image et/ou son des informations textuelles, qui peuvent être utilisées pour la recherche. De nombreuses techniques ont été adoptées pour automatiser la création d'informations textuelles, dont les méthodes d'intelligence artificielle et de «datamining» pour la reconnaissance d'images, la classification automatique, etc.

2. L'indexation de documents

L'autre façon d'effectuer une recherche est de comparer les mots clés d'un document avec les critères de recherche. Le fait d'associer des mots clés à des documents s'appelle l'*indexation*. Il ne faut pas confondre l'indexation avec la compilation d'un index de tous les termes apparaissant dans un texte.

Il y a deux façons principales d'indexer un document: par *extraction* ou par *association*. L'*extraction* de mots clés dans un document est caractérisée par le fait que les mots clés choisis sont présents dans le texte; il s'agit donc de sélectionner les mots d'un texte qui le caractérisent le mieux. À l'inverse, l'*association* de mots clés est la sélection de termes faisant partie d'une liste de termes préalablement choisis. Ainsi, les termes sélectionnés lors de l'association de mots clés ne font pas forcément partie du texte de départ. Cette liste de termes s'appelle un *thésaurus* ([2]), et les termes qu'il contient s'appellent des *descripteurs*.

Par ailleurs, on distingue les thésaurus conceptuels des thésaurus de langage naturel. Ainsi, dans les thésaurus conceptuels, les descripteurs sont relativement abstraits. Le thésaurus Eurovoc, sur lequel nous avons travaillé et que nous découvrirons dans la section suivante, est un exemple de thésaurus conceptuel: ses 6 000 descripteurs décrivent les concepts principaux dans un grand nombre de domaines de la législation européenne en utilisant des termes tels qu'«ACQUISITION DES CONNAISSANCES, OUTIL AGRICOLE et PRÉVENTION DES RISQUES» ([3]). Contrairement aux thésaurus conceptuels, les thésaurus de langage naturel sont plus concrets, et leurs auteurs essayent d'y inclure une liste exhaustive des termes utilisés dans un domaine. Agrovoc, pour l'agriculture, et WordNet, qui n'est pas spécialisé mais qui contient 95 000 ensembles de synonymes, sont deux exemples.

([2]) Thésaurus: répertoire alphabétique de termes normalisés pour l'analyse de contenu et le classement des documents d'information *(Le Nouveau Petit Robert)*.

([3]) Il est conventionnel de noter les descripteurs Eurovoc en petites capitales.

IV. Le thésaurus Eurovoc ([1])

1. Description

Eurovoc, créé en 1995 par l'Office des publications en collaboration avec les institutions, est un thésaurus conceptuel couvrant des domaines très diversifiés, dont: la politique; la législation; la finance; les questions sociales; les sciences; le transport; l'environnement; la géographie; les institutions; etc. Eurovoc est utilisé par le Parlement européen, l'Office des publications, ainsi qu'au moins quinze autres institutions (principalement parlementaires), afin d'indexer leurs documents multilingues pour la recherche automatique. Ce thésaurus existe dans 21 langues officielles de l'Union ainsi qu'en croate, et une équipe de linguistes s'assure que le thésaurus est constamment tenu à jour.

Eurovoc contient 6 075 descripteurs qui sont organisés en 8 niveaux hiérarchiques, comprenant les relations *«terme large»*, *«terme restreint»* et *«terme lié»*, cette dernière concernant des liens entre termes qui ne sont pas reliés hiérarchiquement. Eurovoc contient également un certain nombre de termes qui sont spécifiques à la langue utilisée et dont le but est d'assister les indexeurs dans leur tâche. Ces termes s'appellent des non-descripteurs, et il s'agit le plus souvent de synonymes des descripteurs.

2. Pourquoi utiliser un thésaurus pour indexer des documents?

La plupart des grandes entreprises ou organismes publics utilisent des thésaurus pour l'indexation, l'archivage et la récupération de leurs documents, qu'ils soient sous format papier ou électronique. En effet, une liste de descripteurs soigneusement choisis donne aux usagers un aperçu rapide du contenu d'un document et leur permet de parcourir le document par thème. L'aspect hiérarchique d'un thésaurus permet également de rechercher un domaine sans avoir à entrer tous les termes qui le caractérisent. Ainsi, on peut rechercher «matériaux radioactifs» sans avoir à rechercher tous les termes «plutonium», «uranium», etc.

Mais le plus gros avantage du thésaurus Eurovoc est qu'on peut tirer parti des traductions de chaque descripteur dans les langues de l'Union. Des termes/des critères de recherche exprimés en une langue peuvent ainsi être recherchés dans un corpus de textes multilingues.

La version originale de la plupart des documents législatifs communautaires est rédigée en anglais ou en français et est ensuite traduite dans les autres langues. Pour indexer un document écrit dans une langue autre que l'anglais ou le français, il suffit d'indexer le document en français ou en anglais, puis de trouver la traduction des mots clés dans les autres langues. Cette traduction peut se faire facilement en utilisant l'identifiant (le numéro) des mots clés dans le thésaurus Eurovoc, qui permet de trouver la traduction de n'importe quel descripteur dans 21 langues officielles.

([1]) Le thésaurus Eurovoc est consultable sur http://europa.eu/eurovoc/

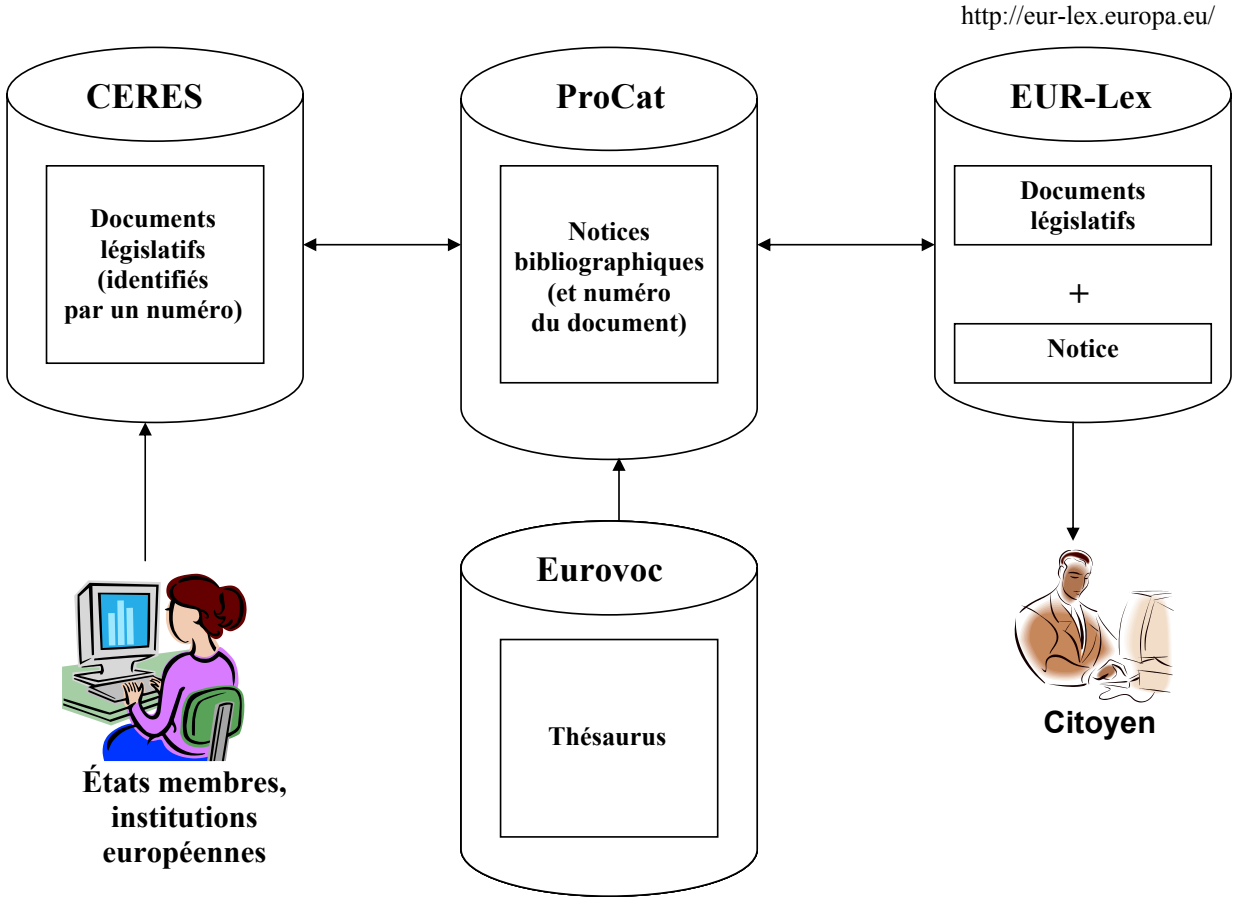
V. L'indexation par les analystes documentaires

Afin de tirer parti des avantages du thésaurus Eurovoc, il faut associer à chaque document législatif un certain nombre de mots clés.

Actuellement, l'Office des publications sous-traite ce travail à une société privée d'analystes, qui lisent chaque nouveau document législatif et sélectionnent les descripteurs Eurovoc qui le caractérisent le mieux. Pour y parvenir, quatre bases de données sont utilisées, comme le montre la figure ci-dessous.

Tous les documents législatifs publiés au Journal officiel sont stockés dans la base de données CERES en format XML. Pour chaque document contenu dans cette base, les indexeurs créent une fiche bibliographique, stockée dans ProCat (Production Catalogue). Ces fiches bibliographiques contiennent des liens entre les documents dans CERES et les descripteurs du thésaurus Eurovoc qui sont choisis comme mots clés. Chaque document, avec sa fiche bibliographique, est consultable par le citoyen sur le site EUR-Lex (http://eur-lex.europa.eu/).

Ce procédé d'indexation «humaine» prend beaucoup de temps — les linguistes indexent en moyenne une dizaine de documents par jour. L'emploi de ces experts représente également un budget considérable pour l'Office des publications, et les résultats sont subjectifs. C'est pourquoi l'Office nous a demandé de rechercher des modèles mathématiques pouvant servir de base à des solutions automatisées, ou du moins semi-automatisées.



VI. La recherche actuelle ([1])

Durant les premières semaines de notre stage, nous nous sommes documenté sur certaines études qui ont été faites dans le domaine de l'indexation assistée par ordinateur. La plupart de ces études concernent l'indexation avec un thésaurus de langue naturelle (voir le chapitre III, point 2), et utilisent des méthodes d'extraction ([2]). Haller a travaillé sur l'économie en 2001, et Montejo Ráez sur la physique des particules et Pouliquen dans un corpus médical, en 2002. Ces études nous ont permis de mieux comprendre le contexte de l'indexation automatisée, mais les méthodes employées, qui se fondent sur des thésaurus de langue naturelle, ne conviennent pas à Eurovoc.

D'autres études effectuées par Marjorie Hlava en 1996 ont été effectuées sur Eurovoc. Pour créer un système d'indexation, elle a défini plus de 40 000 règles linguistiques, faisant appel à des listes de synonymes et à des caractéristiques linguistiques. Un tel système n'est pas utilisable dans notre cas, puisqu'il dépend trop fortement de la langue.

Bien que moins courantes, des études plus mathématiques ont cependant été réalisées. Ferber, en 1997, a créé un système permettant de récupérer des documents en anglais en utilisant des critères de recherche dans une autre langue. Son corpus d'apprentissage consistait en une collection de titres, et le thésaurus utilisé, OECD, est plus petit qu'Eurovoc et n'existe qu'en quatre langues. Le taux de bonne classification obtenu était de 64 %.

Des études théoriques ont également été effectuées, telles que celle effectuée par Adam Kilgarriff de l'université de Brighton, dans laquelle il compare différents critères statistiques pour l'identification de mots clés. S'appuyant en partie sur cette étude, Pouliquen, Steinberger et Ignat ont créé, en 2003, un programme d'indexation assistée avec les descripteurs Eurovoc. Si notre approche est différente de celle de Pouliquen, Steinberger et Ignat, nous nous sommes néanmoins inspiré de leur travail pour la mise en situation du problème et pour la description du thésaurus Eurovoc.

Malheureusement, la plupart de ces études ne peuvent pas être appliquées au cas de l'Office des publications. En effet, comme nous allons le montrer, le procédé d'extraction ne donne pas de bons résultats dans notre recherche. Par ailleurs, le nombre de descripteurs à associer à chaque texte est variable, et on veut générer des listes de mots clés relativement courtes afin d'en conserver la pertinence.

Par conséquent, nous avons dû travailler avec peu de soutien documentaire, et les modèles que nous allons décrire sont le résultat de notre propre recherche, nécessairement trop limitée dans le temps.

([1]) Pour les références exactes, voir la bibliographie en annexe.
([2]) Recherches référencées par Ignat, Pouliquen et Steinberger.

VII. Les données

L'ensemble des documents du Journal officiel étant vaste, nous avons, dans un premier temps, concentré notre étude sur le chapitre de la pêche, les documents de ce chapitre étant suffisamment nombreux (138 textes) et homogènes pour mener une première étude mathématique.

Comme cela a été expliqué dans la section V, les textes se trouvent dans la base CERES alors que les indexations (notices) se trouvent dans ProCat. Il a donc d'abord fallu extraire les textes et leurs fiches bibliographiques pour les regrouper dans un seul fichier ([1]). Ces données étant stockées en format XML, elles ont été séparées par chapitre et converties en format CSV (Comma Separated Value) pour qu'elles puissent être importées dans Excel ([2]).

Les données pour les modèles sur le chapitre de la pêche se présentent comme suit:

La colonne A de chaque document Excel contient les textes, et les lignes des autres colonnes correspondant à chaque texte contiennent les mots clés qui leur ont été associés.

Selon nos données, les analystes ont associé une moyenne de 5,6 descripteurs par texte.

Microsoft Excel - data2.xls

Fichier Edition Affichage Insertion Format Outils Données Fenêtre ?

Tapez une question

Arial 10

B27 politique commune de la pêche

	A	B	C	D	E	
1	TEXTS	KEYWORDS				
2	Règlement no du Conseil du février	Madagascar	accord de pêche	accord CE	contrepartie d'accord	ratifi
3	Accord sous forme échange de le	Madagascar	accord de pêche	accord CE	contrepartie d'accord	acc
4	Protocole fixant les possibilités de	accord de pêche	accord CE	Madagascar	contrepartie d'accord	prot
5	Règlement no de la Commission (	Estonie	Lettonie	Lituanie	Pologne	droit
6	Décision de la Commission du fév	Royaume-Uni	gestion des pêches	ressource halieutique	Manche	droit
7	Décision du Conseil du janvier rel	accord intérimaire CE	accord de pêche	France	Espagne	Port
8	Accord sous forme échange de le	accord intérimaire CE	accord de pêche	Côte d'Ivoire	contrepartie d'accord	Frar
9	Règlement no du Conseil du mars	Indonésie	désastre naturel	aide communautaire	bateau de pêche	aide
10	Règlement no du Conseil du janvie	reconduction d'accord	accord de pêche	contrepartie d'accord	Comores	polit
11	Décision du Conseil du novembre	accord de pêche	reconduction d'accord	accord intérimaire CE	Comores	cont
12	Accord sous forme de lettres relat	accord de pêche	Comores	contrepartie d'accord		
13	Décision du Conseil du avril relativ	convention ONU	gestion des pêches	ressource halieutique	océan Pacifique	adhe
14	Bruxelles le COM final Propositior	flotte de pêche	politique d'aide	IFOP	bateau de pêche	aide
15	Décision du Conseil du octobre re	convention internationale	États-Unis	Costa Rica	droit de la mer	cons
16	Décision de la Commission du dé	Pays-Bas	droit de pêche	ressource halieutique	eau communautaire	zone
17	Règlement no de la Commission (	groupement de producteurs	poisson de mer	transformation alimentaire	aide communautaire	inde
18	Décision du Conseil du décembre	mer Baltique	gestion des pêches	accord de pêche	Russie	Unic
19	Règlement no de la Commission (	produit de la pêche	prix de retrait	prix de vente		
20	Règlement no de la Commission (	produit de la pêche	prix de retrait	prix de vente		
21	Règlement no de la Commission (	produit de la pêche	prix de référence			
22	Règlement no de la Commission (	produit de la pêche	retrait du marché	aide communautaire	prime de stockage	tran
23	Règlement no de la Commission (	produit de la pêche	prime de stockage	aide communautaire		
24	Règlement no de la Commission (	produit de la pêche	aide communautaire	transformation alimentaire	alimentation animale	péré
25	Règlement no du Conseil du déce	adhésion à l'Union européenne	réglementation de la pêche	gestion des pêches	conservation des pêches	capt

([1]) Nous remercions M^me^ Martignon pour avoir effectué cette extraction.

([2]) Nous remercions M. Chapelant pour avoir effectué ces opérations.

VIII. L'hypothèse d'extraction

Il existe deux façons d'indexer un document: par extraction ou par association (voir le chapitre III, point 2). Nous avons commencé par écrire un programme afin de tester le potentiel de la méthode d'extraction dans le cas étudié.

L'Office des publications utilisant peu les langages de programmation, et sa version de la machine virtuelle Java étant incomplète, nous avons décidé d'écrire ce programme en VBA (Visual Basic for Applications), ce langage étant disponible pour tous les utilisateurs de Microsoft Office et facile d'emploi ([1]).

Ce programme, dont le code source peut être consulté en annexe (1. Percentages1), calcule d'abord le pourcentage de textes qui contiennent explicitement les descripteurs qui leur ont été associés. Il calcule ensuite le pourcentage de textes qui contenaient un descripteur, mais auxquels ce descripteur n'a pas été associé. Enfin, le programme crée une nouvelle fiche dans laquelle il associe à chaque texte tous les descripteurs qu'il contient, et compare cette indexation avec celle produite par les analystes documentaires.

En exécutant notre programme sur le corpus des textes du chapitre «Pêche», nous avons découvert que seulement 10,40 % des textes contiennent explicitement les descripteurs qui leur ont été associés et que 72,80 % des textes contiennent au moins un descripteur qui ne leur a pas été assigné. Si ces pourcentages avaient été inversés, l'extraction aurait pu être viable, mais les résultats de notre programme nous ont indiqué que l'extraction ne donnerait pas de bons résultats.

Pour étayer cette conclusion, nous avons calculé le pourcentage de textes pour lesquels l'indexation créée par extraction ne correspondait pas à l'indexation des spécialistes. Le résultat fut désastreux: le taux d'erreur variait entre 94 et 100 %. De plus, les listes de mots clés produites par document sont bien plus longues que celles produites par les analystes, avec une moyenne de 22,45 mots clés par document, à comparer avec la moyenne de 5,4 obtenue par les analystes.

Ces résultats, très mauvais, prouvent que l'indexation par extraction n'est pas viable pour le thésaurus Eurovoc.

Microsoft Excel - extract_4_results

File Edit View Insert Format Tools Data Window Help

F125 politique commune de la pêche

	A	B	C	D	E	F	G	H	I	J	K
1	TEXTS	KEYWORDS		10,40%	72,80%						
5	'Règlemen	Estonie	Lettonie	Lituanie	Pologne	droit de pê	crustacé	réglementation de la pêche			
6	'Décision c	Royaume-I	gestion de	ressource	Manche	droit de pê	filet de pêc	capture ac	pavillon de	France	Belgique
7	'Décision c	accord inté	accord de	France	Espagne	Portugal	licence de	ratification	contreparti	Côte d'Ivoir	protocole d'acc
8	Accord sor	accord inté	accord de	Côte d'Ivoir	contreparti	France	Espagne	Portugal	protocole c	ratification	licence de pêch
9	'Règlemen	Indonésie	désastre n	aide comm	bateau de	aide secto	IFOP				
10	'Règlement no du Conseil du janvier relatif à la conclusion de l accord sous forme d échange de lettres relatif à la prorogation p										
11	'Décision c	reconducti	accord de	contreparti	Comores	politique commune de la pêche					
12	'Accord so	accord de	reconducti	accord inté	Comores	contreparti	politique commune de la pêche				
13	'Décision c	accord de	Comores	contrepartie	d'accord						
14	'Bruxelles	convention	gestion de	ressource	océan Pac	adhésion à	Union européenne				
15	'Décision c	flotte de pê	politique d'	IFOP	bateau de	aide aux s	Asie du Su	industrie d	bateau de	dérogation au droit commu	
16	'Décision c	convention	États-Unis	Costa Rica	droit de la	conservatio	poisson de	gestion de	adhésion à	décision C	Conseil de l'Uni
17	'Règlemen	Pays-Bas	droit de pê	ressource	eau comm	zone de pê	capture au	quota de p	réglementa	filet de pêche	
18	'Décision c	groupemer	poisson de	transforma	aide comm	indemnisat	prix de ven	prix à l'imp	politique c	ommune des prix	
19	'Règlemen	mer Baltiq	gestion de	accord de	Russie	Union euro	adhésion à	pays balte	Pologne		
20	'Règlemen	produit de	prix de retr	prix de vente							
21	'Règlemen	produit de	prix de retr	prix de vente							
22	'Règlemen	produit de	prix de référence								
23	'Règlemen	produit de	retrait du n	aide comm	prime de s	transformation alimentaire					
24	'Règlemen	produit de	prime de s	aide communautaire							
25	Règlement	produit de	aide comm	transforma	alimentatic	péréquation financière					

NB: Les termes surlignés en orange sont ceux qui ont été correctement indexés par extraction. On observe qu'il s'agit surtout de noms de pays ou de groupes de pays.

([1]) Nous avons par la suite fait installer la version 2.4 de Java, ce qui nous a permis d'écrire notre programme d'indexation en Java (voir le chapitre XI, point 2).

IX. Association de descripteurs

Comme l'extraction n'est pas utilisable pour l'identification de mots clés dans ce contexte, nous avons adapté des algorithmes d'analyse numérique pour la classification. Pour ce faire, il a d'abord fallu prétraiter les données.

1. Le prétraitement des données

Les textes stockés dans CERES sont les textes tels qu'ils peuvent être consultés sur l'internet. Ces textes, laissés tels quels, contiennent de nombreux termes et symboles qui peuvent nuire à l'analyse.

Une première étape a par conséquent été de prétraiter les textes, c'est-à-dire de les préparer à l'analyse. Pour ce faire, nous avons écrit un programme en VBA qui élimine systématiquement toute ponctuation, tout symbole inutile ({}[]|-@#$%^&*()_-+=<>/\?!~) et tous les espacements de tabulation. Son code source est consultable en annexe (2. DeleteWords).

De plus, pour toutes les langues européennes (à l'exception du finnois), il existe des termes qui sont utilisés trop souvent pour que leur utilisation soit pertinente pour une analyse comme celle-ci; on appelle ces termes des «mots stop». Afin d'éviter d'en tenir compte dans l'analyse, nous avons appliqué l'algorithme de filtrage des mots stop de l'environnement d'analyse de connaissance Weka ([1]), en modifiant son code source Java afin d'y inclure la liste des mots stop qui nous a été communiquée par le Dr Bagola (cette liste est consultable en annexe).

Une fois ces opérations terminées, nous avons transformé chaque texte en un vecteur de termes. Pour y parvenir, tous les termes ont été convertis en minuscules, et chaque terme a ensuite été transformé en un nombre qui dépend de sa fréquence. Si on calcule la fréquence d'un terme dans un document sans tenir compte du nombre de documents contenant ce terme, le résultat obtenu n'est pas très pertinent ([2]). Nous avons donc poursuivi l'expérience en introduisant à chaque fois de nouvelles transformations, et nous avons finalement adopté la transformée IDF ([3]), qui est définie comme suit:

$$\text{fréquence IDF d'un terme} = f_{i,j} \cdot \log\left(\frac{\text{nombre total de documents}}{\text{nombre de documents contenant le terme } i}\right)$$

où $f_{i,j}$ est la fréquence du terme i dans le document j.

Ces opérations ont été effectuées en faisant passer les données textuelles de notre programme en VBA par le filtre StringToWordVector de Weka, et le résultat se présentait sous la forme d'une liste de nombres représentant les fréquences des termes présents pour chaque texte.

Pour donner une idée de la transformation subie par les documents, voici le début du règlement (CE) n° 555/2005 du Conseil du 17 février 2005 relatif à la pêche thonière au large de Madagascar.

Règlement (CE) n° 555/2005 du Conseil
du 17 février 2005

relatif à la conclusion du protocole fixant les possibilités de pêche thonière et la contrepartie financière prévues dans l'accord entre la Communauté économique européenne et la République démocratique de Madagascar concernant la pêche au large de Madagascar, pour la période allant du 1er janvier 2004 au 31 décembre 2006

LE CONSEIL DE L'UNION EUROPÉENNE,

vu le traité instituant la Communauté européenne, et notamment son article 37, en liaison avec l'article 300, paragraphe 2, et l'article 300, paragraphe 3, premier alinéa,

vu la proposition de la Commission,

vu l'avis du Parlement européen [1],

considérant ce qui suit:

(1) Conformément à l'accord entre la Communauté économique européenne et la République démocratique de Madagascar concernant la pêche au large de Madagascar [2], les deux parties ont négocié pour déterminer les modifications ou compléments à introduire dans cet accord à la fin de la période d'application du protocole annexé à celui-ci.

(2) À la suite de ces négociations, un nouveau protocole fixant les possibilités de pêche et la contrepartie financière prévues dans l'accord précité pour la période du 1er janvier 2004 au 31 décembre 2006, a été paraphé le 8 septembre 2003.

(3) Il est dans l'intérêt de la Communauté d'approuver ledit protocole.

([1]) Witten, I. H., et Frank, E., «Data mining: practical machine learning tools and techniques», 2nd edition, Morgan Kaufmann, San Francisco, 2005.

([2]) En effet, des termes comme «commission», «décision» et «amendement» apparaissent très fréquemment dans les textes législatifs communautaires.

([3]) IDF signifie «Inverse Document Frequency» et est une mesure de l'importance relative d'un terme dans un document d'un corpus. Pour plus d'informations, voir Robertson, S., «Understanding inverse document frequency: on theoretical arguments for IDF», *Journal of Documentation,* n° 60, 2004, p. 503-520.

Ce début de texte devient un fichier contenant les données numériques pour chaque terme contenu dans ce document, calculé par la formule de la fréquence IDF reproduite ci-dessus.

Voici un extrait d'un tel fichier:


```
@attribute abrogeant numeric
@attribute accessoires numeric
@attribute accord numeric
@attribute accords numeric
@attribute accordées numeric
@attribute acte numeric
@attribute action numeric
@attribute actions numeric
@attribute activité numeric
@attribute activités numeric
@attribute actuellement numeric
@attribute adm
[…]

3   0.893929, 21  1.156972, 22  1.184157, 27  1.272765,
35  0.163517, 42  1.272765, 46  0.150914, 48  0.036083,
61  0.991718, 66  0.991718, 87  0.144698, 92  0.460647,
103 0.422657, 105 1.184157, 114 0.189433, 129 1.30501,
134 0.030812, 137 0.360807, 138 0.010119, 155 1.035053,
164 0.450953, 166 0.020388, 169 0.063057, 170 0.971025,
183 1.184157, 184 0.5546,   206 0.480453, 216 1.212453,
227 0.746593, 240 0.169906, 268 0.120374, 280 1.374382,
296 1.013047, 301 0.377923, 317 0.030812, 319 0.422657,
320 1.411859, 330 0.875925, 331 0.015235, 339 0.036083,
349 1.105604, 352 0.893929, 355 0.625151, 361 0.875925,
375 1.08128,  378 1.130812, 398 1.451478, 409 1.272765,
420 0.052141, 439 0.612882, 440 0.06858,  442 1.30501,
443 0.169906, 444 0.26614,  455 0.824557, 470 0.991718,
473 0.327794, 477 0.689978, 498 1.130812, 499 1.241953,
529 0.108525, 533 1.637425, 537 2.299735, 544 2.653812,
553 0.875925, 560 0.085418, 561 0.480453, 562 0.84126,
577 0.746593, 580 0.991718, 609 0.244364, 615 1.057782…
```

Pour chaque élément de cette liste, le premier nombre fait référence au numéro du terme dans la liste des attributs, et le deuxième nombre est sa fréquence IDF.

Afin de modéliser les descripteurs qui ont été associés à chaque texte par les linguistes, nous avons utilisé le filtre StringToNominal de Weka, qui transforme une chaîne de caractères en instance nominale.

2. La modélisation

Alors que nous essayions des algorithmes de catégorisation, nous avons été confronté à un problème de conception. Nous avons mis plusieurs semaines avant de trouver la façon optimale d'organiser les données prétraitées, et nous avons essayé de nombreuses stratégies, dont la plupart se sont révélées vaines. Il serait long et fastidieux d'établir une liste de tous nos essais infructueux, mais nous allons néanmoins en décrire quelques-uns afin de montrer pourquoi nous avons adopté notre analyse finale.

i) Des textes vers les descripteurs

Nous avons d'abord tenté de créer une indexation en associant chaque texte à une catégorie pour créer des modèles associant chaque texte à une entité. Comme le nombre de descripteurs associés à chaque texte est variable, notre problème était de trouver une catégorie pouvant représenter le plus d'informations possibles sur l'indexation.

Entre autres méthodes, nous avons tenté d'associer chaque texte au premier descripteur qui lui a été associé par les analystes documentaires. L'application d'algorithmes de classification a donné un taux de bonne classification d'au plus 30 %. Autrement dit, 70 % des textes ont été associés par le modèle à un descripteur qui n'était pas celui choisi par les analystes. Ce résultat reflète l'indépendance de portée de chaque descripteur associé. En effet, les descripteurs sélectionnés comme mots clés appartiennent à des hiérarchies Eurovoc différentes ([1]), et le premier descripteur associé à un texte a la même importance que tout autre descripteur associé à ce texte.

Une autre stratégie expérimentée a été l'association de tous les mots clés d'un document en répétant le texte du document autant de fois qu'il y avait de mots clés. De cette façon, chaque copie de texte était considérée comme étant une nouvelle entité, et cela permettait de prendre tous les mots clés de chaque document dans le modèle. Pour y parvenir, nous avons écrit un programme en VBA dont le code source est disponible en annexe (7. Separate.bas). Le meilleur taux de succès, obtenu par le classifieur bayésien multinomial, était de 6 %. Ce taux démontre que la duplication d'attributs ne fait qu'accroître le «bruit» d'un modèle.

ii) Des descripteurs vers les textes

À la suite de nombreux essais, nous avons découvert que la création de modèles pour chaque descripteur Eurovoc est beaucoup plus efficace que la création d'un modèle par texte.

Cependant, le thésaurus Eurovoc contenant plus de 6 000 descripteurs en langue française, une analyse portant sur tous les descripteurs serait lourde en termes de temps et d'espace mémoire. Nous avons donc décidé, pour nos premiers essais, d'éliminer tous les descripteurs qui ont été assignés à moins de 5 textes et tous ceux qui ont été assignés à des textes contenant moins de 2 000 caractères. Ce sont des conditions que nous avons choisi de modifier par la suite pour améliorer la précision de l'indexation automatique (voir ci-dessous). Nous avons réalisé cette élimination en écrivant un programme en VBA, dont le code source peut être consulté en annexe (3. DeleteDescriptors).

Nous avons ensuite créé deux programmes en VBA permettant ensemble de construire une nouvelle feuille Excel pour chaque descripteur restant, avec en colonne A tous les textes et en colonne B le nombre 1 ou 0 selon que le descripteur a été associé au texte correspondant ou non.

([1]) Les hiérarchies du thésaurus Eurovoc représentent des niveaux d'abstraction différents.

Le code source de ces programmes est consultable en annexe (4. AssignedTexts et 5. AssignedDescriptors).

Cette façon d'organiser les données permet de créer un modèle de catégorisation pour chaque descripteur Eurovoc. Ainsi, pour chaque descripteur, nous associons les textes à l'une des deux classes: la classe 1 si le descripteur a été associé au texte et la classe 0 dans le cas contraire.

À ce stade, il y avait lieu de déterminer des règles permettant de décider si un descripteur donné devait être associé à un texte donné.

X. Algorithmes de catégorisation

Au cours de nos expériences, nous avons construit et testé plusieurs modèles différents avec des paramètres différents. Nous allons décrire les plus significatifs pour l'objectif recherché, ainsi que certains autres qui sont intéressants de par le type d'erreur commise.

Nous décrivons d'abord quelques critères de signification que nous avons testés, et qui permettent d'associer une mesure à chaque terme d'un texte. Cette mesure peut ensuite être comparée aux descripteurs qui lui sont associés; de cette manière, on peut identifier les termes les plus susceptibles d'apparaître dans un texte qui a été indexé avec un descripteur donné.

Nous décrivons ensuite quelques algorithmes de catégorisation que nous avons testés dans l'environnement Weka et nous en analysons les résultats.

1. Critères de signification (1)

Dans le but de créer un modèle qui associe chaque descripteur aux textes qui lui correspondent, nous établissons un ensemble de termes qui sont mathématiquement liés aux descripteurs. Cet ensemble de termes «significatifs» est ensuite incorporé dans l'algorithme de catégorisation.

Pour ce faire, nous avons essayé plusieurs critères afin de déterminer si un terme est significatif ou non.

i) Un modèle simple

Comme dans le cas des documents CERES nous ne tenons pas compte de la structure interne des textes, on peut poser le problème plus simplement afin de ne pas alourdir inutilement les formules. Prenons le cas de deux textes, *X* et *Y*, et soit *w* un terme contenu dans les deux textes. On a alors la situation suivante:

	X	*Y*	
w	*a*	*b*	$a+b$
pas *w*	*c*	*d*	$c+d$
	$a+c$	$b+d$	$a+b+c+d=N$

Ce tableau signifie qu'il y a *a* occurrences du terme *w* dans le texte *X* (qui contient $a + c$ termes) et *b* dans le texte *Y* (qui contient $b + d$ termes).

(1) Ces critères sont expliqués dans Kilgarriff, A., «Which words are particularly characteristic of a text? A survey of statistical approaches», Sussex, 1996.

ii) Le test du χ^2

Une façon de sélectionner un terme dans un texte est d'effectuer un test du χ^2 avec pour hypothèse nulle que les deux textes *X* et *Y* contiennent des termes tirés au hasard et suivant une loi uniforme parmi un ensemble de termes.

Si cette hypothèse est vérifiée, pour un tableau de contingence de taille $m \times n$, la statistique

$$\sum \frac{(O - E)^2}{E}$$

(où *O* est la valeur observée, *E* est la valeur attendue calculée sur la base de l'ensemble des textes et où la somme porte sur tous les éléments du tableau) suivra une loi du χ^2 avec $(m - 1)(n - 1)$ degré de liberté (2).

Le test du χ^2 peut être utilisé avec des tableaux de contingence de toutes tailles et peut donc permettre la comparaison de plusieurs termes. Dans tous les essais, l'hypothèse nulle a été largement rejetée.

Le but recherché ici est d'identifier les termes les mieux reliés aux descripteurs Eurovoc choisis comme mots clés par les experts. On pourrait penser que plus la valeur du χ^2 est élevée (où l'hypothèse nulle est la plus fortement rejetée), plus le terme en question est distinctif. Ceci est vrai lorsque la fréquence du terme est constante dans tous les textes, mais de fortes valeurs du χ^2 sont plus souvent associées à la quantité d'information disponible sur le terme, et non aux caractéristiques du terme lui-même.

iii) Information mutuelle

Une autre façon d'identifier les termes les plus significatifs présents dans un texte qui a été indexé par un descripteur particulier est l'information mutuelle (IM) (3). Il s'agit ici de prendre le logarithme du rapport entre la fréquence relative dans un corpus et sa fréquence relative dans l'ensemble des corpus.

Si on exprime l'information mutuelle par rapport au tableau de contingence représenté au chapitre X, point 1.i), on obtient:

$$IM_{W,X} = \log_2\left(\frac{a}{a+c} \times \frac{N}{a+b}\right)$$

(2) À condition que toutes les valeurs attendues soient supérieures ou égales à 5.

(3) Church et Hanks, 1990.

Contrairement au test précédent, ceci est une mesure théorique d'information et non un test d'hypothèse statistique. Cette mesure quantifie l'information que le terme *w* fournit sur le corpus *X* et vice versa. Cette mesure fut introduite dans l'analyse du langage pour mesurer la cooccurrence et pour spécifier l'information fournie par un terme au sujet d'un autre.

Church et Hanks indiquent que cette mesure n'est pas valable pour des fréquences basses et proposent une fréquence minimale de 5. Contrairement au χ^2, l'information mutuelle n'accumule pas l'information procurée par chaque terme d'un corpus; elle ne nous éclaire que sur la relation existant entre un corpus et un terme. Si on applique l'information mutuelle sur les termes d'un même corpus, les valeurs les plus hautes seront obtenues pour les termes les plus rares. En revanche, le test du χ^2 retourne des valeurs fortes pour les termes les plus fréquents.

En essayant d'appliquer l'information mutuelle sur les textes d'EUR-Lex et en comparant les résultats avec les termes associés aux descripteurs choisis par les analystes, il apparaît que l'information mutuelle retourne trop souvent les termes rares et que le test du χ^2 retourne trop souvent les termes fréquents.

iv) La vraisemblance logarithmique (G^2)

Une autre étude permet d'éviter les problèmes posés par l'information mutuelle et le test du χ^2. Il s'agit de la vraisemblance logarithmique ([1]), notée G^2, qui permet d'identifier les termes les plus significatifs d'un texte, même s'ils n'apparaissent qu'une fois.

Pour le tableau de contingence du chapitre X, point 1.i):

$$\begin{aligned} G^2 = 2[&(a\log(a) + b\log(b) + c\log(c) + d\log(d) \\ &- (a+b)\log(a+b) - (a+c)\log(a+c) \\ &- (b+d)\log(b+d) - (c+d)\log(c+d) \\ &+ (a+b+c+d)\log(a+b+c+d)] \end{aligned}$$

Dans le but de déterminer les termes les plus significatifs pour un descripteur, sans avoir à tenir compte de la structure interne d'un texte, G^2 est une mesure mathématiquement bien fondée, et après avoir essayé cette mesure sur notre corpus, on s'aperçoit qu'elle correspond assez bien aux estimations des experts.

v) Conclusion des expériences sur les critères de signification

D'après les études théoriques de ces différents tests et mesures et d'après nos tests sur les données d'EUR-Lex, il apparaît que la vraisemblance logarithmique fournit les informations les plus utiles. Cette conclusion semble confirmée par le choix de Steinberger, Pouliquen et Ignat, qui ont adopté cette mesure pour leur modèle.

([1]) Dunning, 1993.

2. Arbres décisionnels de catégorisation ([2])

La mesure de signification adoptée dans le point précédent doit à présent être incorporée dans un algorithme décisionnel, qui doit permettre de déterminer, pour chaque descripteur Eurovoc, quels sont les textes qui lui correspondent le mieux ([3]).

i) Intérêts d'un arbre de décision

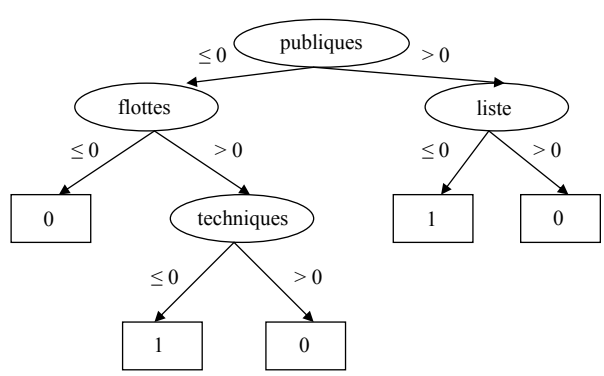
Les arbres de décision, qui sont surtout utilisés pour la classification, nous ont permis d'établir des règles d'association entre les descripteurs Eurovoc et les textes du corpus d'apprentissage. Leur construction se fonde sur une suite de tests sur les variables explicatives de chaque descripteur.

Nous avons construit ce modèle à partir d'un ensemble de textes dont l'indexation est connue, et les règles résultantes se présentent sous la forme d'arbres.

Les intérêts des arbres de décision sont nombreux. Dans le cas présent:

- ils peuvent être utilisés sur tout corpus de texte ayant été indexé avec Eurovoc;
- ils peuvent traiter des attributs de nature variée (dans ce cas-ci, plusieurs mesures de la fréquence);
- ils permettent à l'utilisateur de comprendre le chemin suivi pour parvenir à une décision (un chemin dans un arbre de décision représente une formule logique propositionnelle);
- ils effectuent automatiquement une sélection de variables pertinente;
- ils sont non paramétriques (bien que dans notre modèle, nous ayons introduit des paramètres dont l'ajustement influence les résultats de manière significative).

Avant d'expliquer l'algorithme de construction, nous donnons un exemple d'arbre de décision pour le descripteur «flotte de pêche»:



([2]) Certaines parties de cette démonstration sont inspirées du cours de Laurent Bougrain en fouilles de données, université Henri Poincaré, 2004.

([3]) D'après l'indexation des analystes.

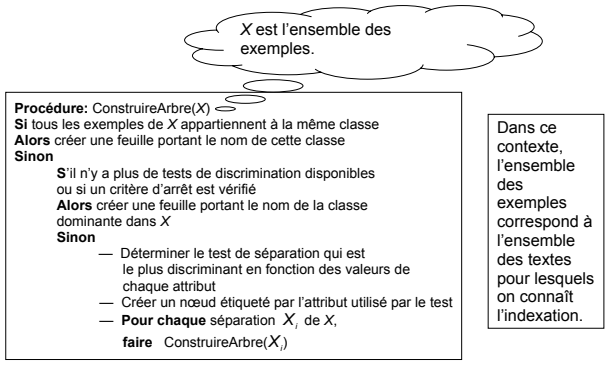
Cet arbre de décision permet de déterminer si le mot clé «flotte de pêche» doit être associé à un texte donné. Les termes encerclés sont les termes qui sont mathématiquement liés au descripteur «accord de pêche» (selon la vraisemblance logarithmique). Si la valeur (1) associée au terme «publiques» est strictement supérieure à 0, alors on regarde la valeur associée au terme «liste». Si elle est strictement supérieure à 0, on n'associe pas «flotte de pêche» à ce texte. Sinon, on l'associe. Et ainsi de suite.

Ainsi, l'automatisation d'un tel modèle de décision n'est pas une boîte noire: on peut facilement comprendre les critères de décision dans chaque cas.

ii) Principes de construction

Construire tous les arbres de décision possibles afin d'identifier le meilleur n'est pas envisageable, puisqu'une telle approche multiplierait les analyses sur tous les termes de tous les textes et prendrait trop de temps et de mémoire. On cherche donc à construire l'arbre par induction descendante.

L'algorithme de base peut se résumer ainsi:



L'arbre CART [voir le chapitre X, point 2.iv)] est une implémentation de cet algorithme.

Ici, les classes sont 1 ou 0: 1 si le descripteur analysé est associé au texte, 0 dans le cas contraire. Comme les attributs correspondent à l'expression IDF de la fréquence, le test de discrimination que nous utilisons est la comparaison de cette fréquence avec 0 (une fréquence ne pouvant jamais être négative). Enfin, le critère de séparation que nous avons choisi d'utiliser est la valeur de la vraisemblance logarithmique.

Afin de mieux comprendre la construction de l'arbre de décision pour l'indexation Eurovoc, prenons l'exemple cité précédemment du descripteur «flotte de pêche».

Après calcul, la mesure de la vraisemblance logarithmique indique que le test sur la valeur du terme «publiques» est le plus discriminant. Selon que la fréquence IDF du terme «publiques» est positive ou non, on mesure la vraisemblance logarithmique sur tous les autres termes d'un texte et on sélectionne celui pour lequel la vraisemblance logarithmique indique le mélange le plus faible, et ainsi de suite.

iii) Élagage

On pourrait continuer la construction de l'arbre jusqu'à ce que tous les termes d'un texte soient utilisés. Ceci équivaudrait à apprendre tous les cas «par cœur» et aurait l'avantage de réduire le taux d'erreur jusqu'à 0 %. Cependant, une telle stratégie n'aurait aucun intérêt.

En effet, un tel arbre décisionnel reposerait entièrement sur un échantillon particulier: le corpus d'apprentissage, dont les éléments peuvent être bruités.

Il est plus utile d'avoir un modèle moins complexe, plus facilement applicable et qui généralise les exemples du corpus d'apprentissage de manière à pouvoir appliquer l'arbre décisionnel à des textes nouveaux, dont on ne connaît pas l'indexation.

Dans le but d'éviter la construction d'un arbre trop complexe, on élague l'arbre complet en remontant des feuilles vers la racine. En voici le principe (2):

Soit $\{T_{max}, T_1, \ldots, T_n\}$ la séquence d'arbres, où T_{max} est l'arbre complet et T_n est l'arbre constitué uniquement de la racine.

Pour passer de T_k à T_{k+1}, il faut transformer un nœud de T_k en feuille. Le bénéfice de cet élagage est mesuré par un critère qui représente un compromis entre la complexité de l'arbre et l'erreur obtenue sur un corpus de test. Ce critère permet de décider quand il y a lieu d'arrêter l'élagage. Ainsi, si le coût en termes de complexité de T_{k+1} dépasse le coût de T_k, on cesse l'élagage et l'arbre est terminé.

Parmi les différents critères, on cherche à minimiser

$$c(T_k, s) = \frac{MC_{élag}(T_k, s) - MC(T_k)}{n(T_k)(nt(T_k, s) - 1)}$$

où $MC_{élag}(T_k, s)$ est le nombre d'exemples de l'ensemble d'apprentissage qui sont mal classés par le nœud s de T_k si on transforme s en feuille,

$MC(T_k)$ est le nombre d'exemples de l'ensemble d'apprentissage qui sont mal classés par le nœud s de T_k,

$n(T_k)$ est le nombre de feuilles de T_k,

$nt(T_k, s)$ est le nombre de feuilles du sous-arbre situé sous le nœud s de T_k.

(1) Voir le chapitre IX, point 1.

(2) Extrait du cours de Laurent Bougrain en fouille de données, université Henri Poincaré, 2005.

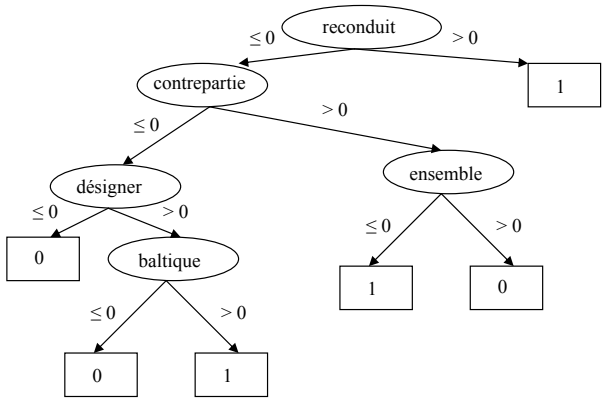
iv) Évaluation des résultats sur un exemple

Prenons l'exemple du descripteur «accord de pêche». Le filtre StringToWordVector de Weka, qui transforme les textes du chapitre de la pêche de la manière décrite au chapitre IX, point 1, montre que 25 textes sur les 138 de ce chapitre ont été indexés avec le descripteur «accord de pêche».

Cette représentation signifie que les experts linguistes ont associé le descripteur «accord de pêche» à 25 textes sur les 138 textes du chapitre de la pêche.

La mise en œuvre de l'implémentation Weka de l'arbre décisionnel CART donne lieu à l'arbre ci-après ([1]).

Cet arbre a été construit en tenant compte des 138 textes (instances) et des 1 570 termes (attributs).



Dans le but de quantifier la qualité de cet arbre et de pouvoir le comparer à d'autres algorithmes de classification, nous avons d'abord séparé notre corpus de textes en deux parties: les deux premiers tiers du corpus devaient servir à l'apprentissage, c'est-à-dire à la construction de l'arbre, et le tiers restant servait à comparer les textes associés à «accord de pêche» par l'arbre avec les textes indexés avec «accord de pêche» par les analystes.

Cette méthode de vérification indique que, dans le corpus de test, 97,87 % des textes ont été correctement classifiés.

Afin d'obtenir une estimation plus réaliste, nous avons effectué une comparaison en validation croisée: le corpus total a été divisé en 10 parties contenant un nombre de textes égal; le premier dixième a été utilisé pour le test et le reste pour la construction de l'arbre. Une fois le résultat noté, le deuxième dixième a été utilisé pour le test et le reste pour la construction de l'arbre. Après avoir à nouveau noté le résultat, nous avons procédé de même avec les 10 dixièmes du corpus total; enfin, nous avons calculé la moyenne des taux de bonne classification.

Cette façon de tester l'arbre est plus réaliste, puisqu'elle se base sur plusieurs corpus de test différents. Cette fois-ci, une moyenne de 131 textes ont été correctement associés au descripteur «accord de pêche», soit 94,93 %. Nous avons essayé de nombreux algorithmes de classification, et le taux obtenu par l'arbre CART est le meilleur dans la plupart des cas (nous verrons, dans le point suivant, que les classifieurs bayésiens fonctionnent mieux dans de rares cas).

Outre le taux de bonne classification, un critère important dans l'évaluation d'un modèle est le type d'erreur commise. C'est pourquoi il est important de prendre en compte la matrice de confusion, qui indique le nombre de textes qui n'ont pas été associés au descripteur alors qu'ils auraient dû l'être, ainsi que le nombre de textes qui ont été associés au descripteur mais qui n'auraient pas dû l'être.

Dans le cas du descripteur «accord de pêche», la matrice de confusion se présente ainsi:

ANALYSTES	MODÈLE	
	Pas associé	Associé
Pas associé	111	2
Associé	5	20

Ainsi, le descripteur «accord de pêche» a été correctement dissocié de 111 textes et a été correctement associé à 20 textes. Dans 2 cas, ce descripteur n'aurait pas dû être associé mais il l'a été (erreur de type 1), et dans 5 cas ce descripteur aurait dû être associé mais ne l'a pas été (erreur de type 2).

On dira que le taux de *précision* est de 2/138, soit 0,14 %, et que le taux de *rappel* est de 5/138, soit 0,38 %.

Cette matrice de confusion joue un rôle primordial dans le choix du meilleur algorithme de catégorisation. Le D[r] Bagola nous ayant informé que, pour l'indexation, les erreurs de type 1 sont moins gênantes que les erreurs de type 2, l'algorithme de l'arbre CART s'est à nouveau révélé le plus approprié.

Un relevé des différentes mesures d'erreur pour «accord de pêche» est disponible en annexe.

3. Modèles bayésiens

Les arbres de décision décrits dans la section précédente nous ont permis d'obtenir d'excellents taux de classification, mais dans de rares cas un modèle bayésien a révélé un résultat légèrement meilleur.

([1]) Il arrive que, lorsqu'un descripteur est trop peu utilisé par les analystes, l'arbre décisionnel optimal soit celui qui associe systématiquement ce descripteur à la classe 0. C'est le cas du descripteur «restriction à l'importation», qui n'est associé qu'à un seul texte parmi les 138 du chapitre de la pêche.

La formule de probabilité conditionnelle de Bayes nous dit que

$$P(C_i \mid x) = \frac{P(x \mid C_i) \times P(C_i)}{P(x)} = \frac{P(x \mid C_i) \times P(C_i)}{\sum_j P(x \mid C_j) \times P(C_j)}$$

Dans le cas que nous étudions:

x représente la valeur de la fréquence IDF d'un terme donné,

C_i est la classe i (valant 1 si le descripteur analysé est associé à un texte contenant un terme de fréquence IDF valant x, sinon 0),

$P(C_i \mid x)$ signifie «probabilité de C_i sachant x» et représente la probabilité a posteriori qu'un texte appartienne à la classe i sachant qu'il contient un terme de fréquence IDF valant x,

$P(C_i)$ est la probabilité a priori qu'un texte appartient à la classe i,

$P(x \mid C_i)$ représente la probabilité qu'un texte contient un terme de fréquence IDF valant x sachant qu'il appartient à la classe i (cette quantité s'appelle la *vraisemblance*),

$P(x)$ représente la probabilité qu'un texte contient un terme dont la fréquence IDF vaut x.

Pour affecter un individu à une classe, on minimise le risque d'erreur en prenant la décision d'affecter l'individu à la classe dont la probabilité a posteriori est la plus grande.

Ainsi, si les densités de probabilité a priori sont les mêmes pour toutes les classes, la règle de Bayes revient alors à affecter un individu à la classe dont la densité de probabilité est la plus forte.

En effectuant des essais, il s'avère que la distribution bayésienne multinomiale décrit assez bien le comportement de certaines indexations. Ainsi, dans l'ensemble des 635 descripteurs utilisés dans le chapitre «pêche», seuls 5 ont été classés plus efficacement par la distribution bayésienne multinomiale que par l'algorithme CART (le taux d'erreur descendant entre 1 et 4 points de pourcentage).

XI. Création d'un algorithme d'indexation assistée

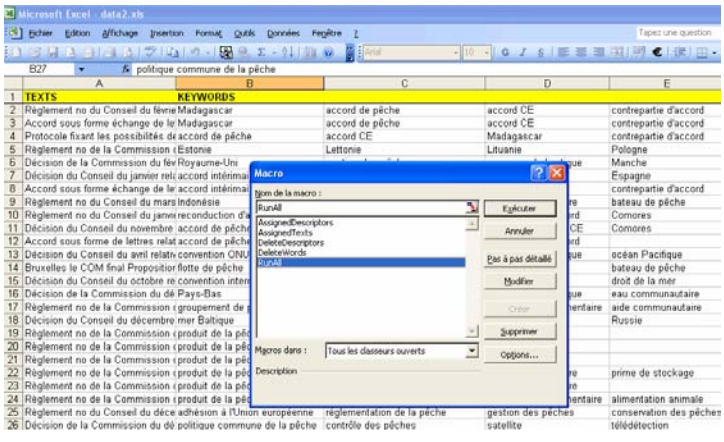
Le but de notre projet à l'Office des publications était l'élaboration de modèles mathématiques pouvant servir de base à une indexation automatisée. Après de nombreux essais, nous avons conclu que l'implémentation Weka de l'arbre décisionnel CART procurait les meilleurs résultats, tant du point de vue du taux de bonne classification que de celui du type d'erreur commise, pour la vaste majorité des descripteurs Eurovoc.

Lors de notre première présentation de ces résultats à l'Office des publications, la possibilité de sous-traiter à une société d'informatique la création d'un logiciel d'indexation automatique basé sur notre étude a été mentionnée.

Par la suite, ayant voulu pousser nos découvertes plus loin, nous avons décidé de créer un programme d'indexation automatique. À cet effet, nous avons incorporé dans notre programme notre adaptation de l'arbre décisionnel CART de Weka.

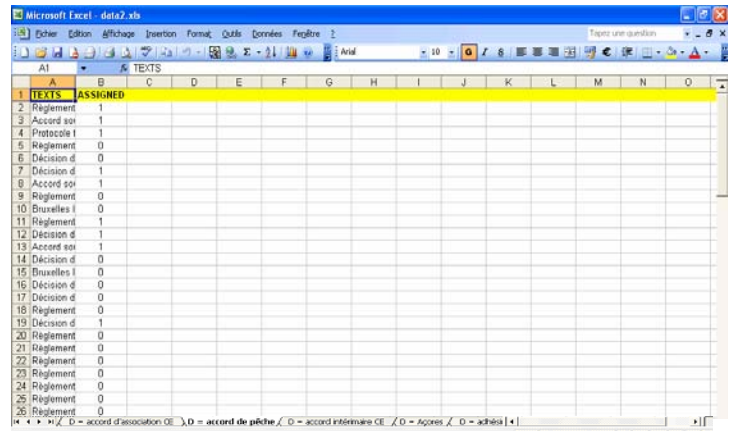
1. Première partie du programme: le prétraitement

Notre programme se compose de deux parties. La première partie est écrite en VBA (RunAll — voir le code source en annexe au point 6) et exécute les programmes DeleteWords, DeleteDescriptors, AssignedTexts et AssignedDescriptors. Cette routine VBA permet ainsi de préparer les données à l'analyse, en effectuant une partie du prétraitement des textes et en créant une nouvelle fiche Excel pour chaque descripteur Eurovoc qui a été utilisé au moins une fois (1) par les analystes.



La mise en route de cette phase préliminaire est très simple. Il suffit d'importer les textes dont on connaît l'indexation dans un document Excel et de lancer l'exécution en passant par l'interface des macros.

Pendant l'exécution, l'utilisateur peut observer le prétraitement des textes et la création des nouvelles feuilles Excel pour chaque descripteur, dont la colonne A contient les textes prétraités et la colonne B contient le nombre 1 ou 0 selon que le descripteur a été associé au texte correspondant ou non. Voici par exemple le résultat pour le descripteur «accord de pêche».



La première phase de prétraitement étant terminée, la deuxième partie de notre programme d'indexation, qui est écrite en Java, entre en jeu.

2. Deuxième partie du programme: la modélisation

Cette partie du programme prend pour base le résultat de l'étape précédente du programme, c'est-à-dire, pour chaque descripteur restant, une feuille qui indique les textes auxquels il a été associé.

Cette partie du programme effectue les opérations suivantes:

1. pour chacune de ces feuilles (donc pour chaque descripteur restant), le programme construit un arbre de décision de la façon décrite dans le chapitre X;
2. pour chacun des nouveaux textes (dont on ne connaît pas l'indexation), il applique chaque arbre de la façon suivante. Pour chaque arbre:
   a) il calcule la fréquence IDF de chaque mot sélectionné par l'arbre,
   b) il applique les règles de l'arbre à ces mots pour déterminer si le descripteur de l'arbre doit être associé au texte. C'est ainsi qu'il produit une liste de descripteurs pour chaque texte;

(1) Au départ, nous avions décidé de limiter le nombre de descripteurs à ceux qui ont été utilisés au moins 5 fois. C'est en faisant des essais que nous avons pu obtenir les meilleurs résultats en ajustant ce paramètre à 1.

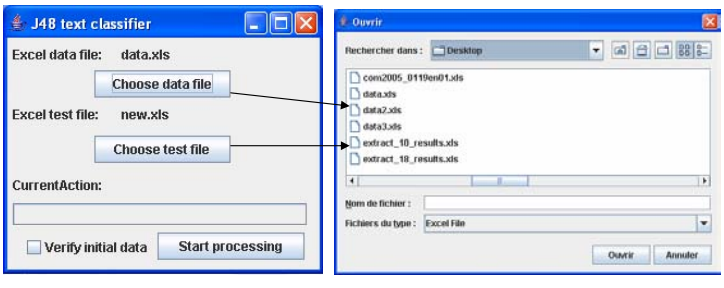
3. si les descripteurs choisis par les analystes pour les textes ainsi indexés sont connus, le programme effectue une comparaison entre ces descripteurs et ceux retournés par le programme.

Cette partie du programme fait appel à la bibliothèque JExcel API, qui permet à un programme Java d'interagir avec Microsoft Excel. Cette bibliothèque permet ainsi de consulter des documents Excel et également de créer et de modifier de tels documents.

Une autre bibliothèque utilisée est celle de Weka. Notre programme fait appel au filtre StringToWordVector ainsi qu'aux classes implémentant l'arbre décisionnel CART.

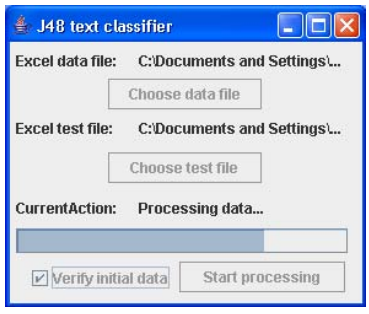
Nous avons créé un fichier exécutable .jar (Java Archive) qui contient toutes les classes compilées nécessaires à l'exécution du programme, ainsi qu'un fichier MS-DOS batch (.bat) qui permet de lancer l'exécution du fichier .jar en augmentant la pile utilisable par la Machine Virtuelle Java. En effet, la taille de la pile par défaut n'est pas suffisante lorsque notre programme est utilisé sur des ensembles de textes particulièrement volumineux et notre fichier batch fixe la pile à 1 024 MB (cette valeur pouvant être modifiée en changeant le paramètre du fichier batch).

L'interface graphique de notre programme d'indexation est très simple, sa raison d'être étant de vérifier la qualité des modèles mathématiques utilisés plutôt que de servir à une indexation à long terme.



Le premier bouton permet de parcourir l'arborescence du disque local pour sélectionner le document Excel issu de l'exécution de la routine VBA RunAll. Le deuxième bouton permet de sélectionner un document Excel contenant des textes dont l'indexation est inconnue, ou qui contient des textes déjà indexés dont on veut vérifier la qualité de l'indexation automatique et en analyser les erreurs.

Dans ce dernier cas, il suffit d'activer la *check box* située à côté du bouton de validation. L'étiquette «CurrentAction» permet de savoir quelle opération est effectuée par le programme à un moment donné, et la barre de progression permet d'estimer le temps d'exécution restant.



Grâce aux fonctionnalités de la bibliothèque JExcel API, le programme crée automatiquement un nouveau fichier .arff (qui est le format d'entrée standard pour Weka — une description du format .arff est disponible en annexe au point 10) correspondant à chaque feuille Excel créée par la routine VBA RunAll. Grâce aux instructions en ligne de commande de Weka, notre programme passe chaque fichier .arff par le filtre StringToWordVector en spécifiant les options de prétraitement (voir le chapitre IX, point 1). Chaque fichier obtenu est ensuite soumis au classifieur CART.

Le programme crée ensuite un nouveau document Excel, nommé «Results» et placé dans le même répertoire que celui du programme. Si la *check box* n'a pas été activée, le fichier Results contient l'indexation proposée pour les documents dont on ne connaissait pas l'indexation. Si la *check box* a été activée, le fichier Results contient une analyse de la comparaison entre l'indexation automatique et l'indexation des analystes.

Dans une première feuille contenant l'indexation des analystes, les descripteurs «manquants» dans l'analyse automatique sont mis en évidence par un fond rouge, les autres étant en vert. Dans une deuxième feuille contenant l'indexation automatique, les descripteurs «en trop» sont en rouge, les autres en vert.

Dans les deux cas, le pourcentage de classification incorrecte est calculé pour chaque texte et indiqué dans la colonne B des deux feuilles, pour chaque type d'erreur (descripteur «manquant» et descripteur «en trop»). Il est ainsi facile d'identifier les erreurs et de quantifier la qualité de l'algorithme.

XII. Analyse des résultats de l'indexation automatique

Ayant créé un programme qui indexe automatiquement des documents, nous avons testé ce programme sur les chapitres suivants ([1]):

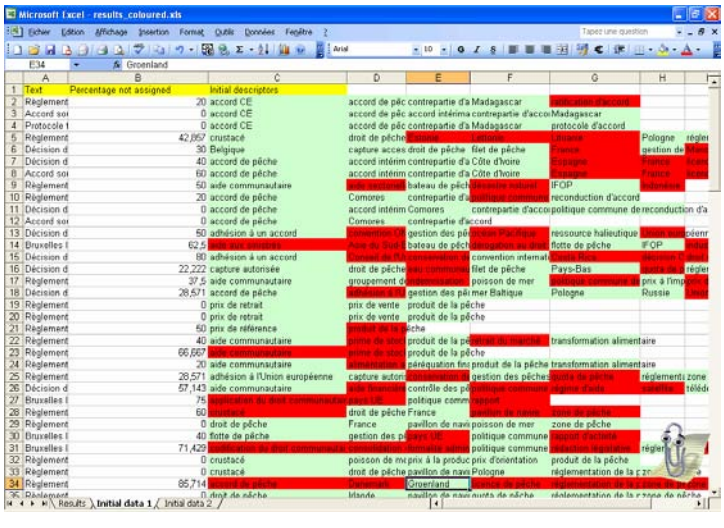
Chapitre 4:	Pêche (138 textes)
Chapitre 8:	Politique de la concurrence (202 textes)
Chapitre 10:	Politique économique et monétaire et libre circulation des capitaux (144 textes)
Chapitre 18:	Politique étrangère et de sécurité commune (154 textes)

L'ordinateur que nous avons utilisé prend cinq heures pour le traitement de chacun de ces chapitres. Ce temps peut être réduit en utilisant un processeur plus puissant.

1. Résultats pour le chapitre de la pêche

Le nombre de descripteurs proposé par notre système ne diffère pas beaucoup de celui proposé par les analystes. Par exemple, pour le chapitre de la pêche, notre système propose une moyenne de 4,5 descripteurs par texte, comparativement aux 5,6 descripteurs proposés par les analystes. Ainsi, la plupart des textes sont indexés avec un nombre de descripteurs entre 4 et 6.

Les autres résultats pour le chapitre de la pêche se présentent comme suit:



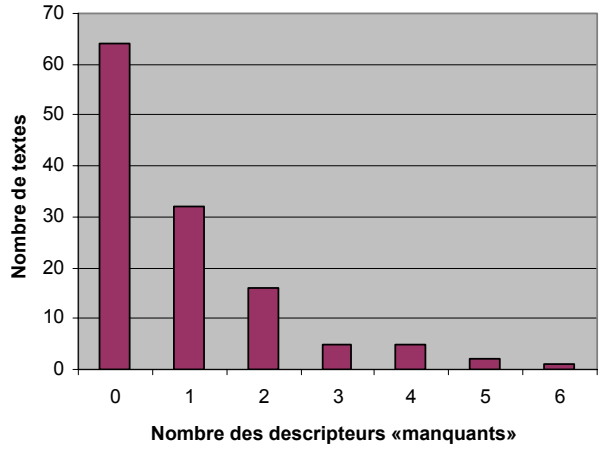
([1]) Nous remercions M. Chapelant pour avoir extrait ces documents et pour les avoir convertis en format CSV.

Ce tableau représente les descripteurs «manquants» par rapport à l'indexation des analystes. À la colonne B, on peut constater que le taux d'erreur est variable et que sa moyenne est de 20,27 %. Autrement dit, pour chaque texte, parmi les descripteurs sélectionnés par les analystes, 8 descripteurs sur 10 sont automatiquement associés.

Cette représentation des erreurs permet également d'identifier les descripteurs sur lesquels les erreurs sont commises. Dans le cas présent, on peut constater que les erreurs portent surtout sur les noms de lieux (pays, groupes de pays, etc.).

Cette observation permet d'améliorer l'algorithme en ajoutant une règle associant systématiquement tout nom de pays contenu dans un texte. En adoptant cette stratégie, on réduit le taux d'erreur moyen à 16,04 %.

Le nombre de descripteurs «manquants» peut également être interprété de la façon suivante:



Sur ce graphique, on peut constater que: 64 textes ont été indexés par (au moins) tous les descripteurs choisis par les analystes; 32 textes ont été indexés par tous les descripteurs choisis par les analystes sauf 1; et ainsi de suite.

Le deuxième type d'erreur est celui des descripteurs «en trop». En d'autres termes, il s'agit du pourcentage par texte des descripteurs qui ont été assignés par le programme mais pas par les analystes. Pour le chapitre de la pêche, le taux moyen de descripteurs «en trop» est de 6,56 %.

2. Résultats pour les autres chapitres

Les taux de descripteurs «manquants» et de descripteurs «en trop» pour les quatre chapitres peuvent se résumer de la manière suivante:

	Descripteurs «manquants»	Descripteurs «en trop»
Pêche	16,04 %	6,56 %
Politique de la concurrence	20,14 %	2,19 %
Politique économique et monétaire et libre circulation des capitaux	9,3 %	2,97 %
Politique étrangère et de sécurité commune	21,17 %	3,50 %

Les taux indiqués ci-dessus varient entre 10 et 20 % pour les descripteurs «manquants», et entre 2 et 6,5 % pour les descripteurs «en trop». Une hypothèse qui pourrait expliquer ces variations serait que les différents chapitres de la législation européenne n'ont pas tous le même niveau d'abstraction linguistique ni la même homogénéité. Un autre facteur est le nombre de textes par chapitre, qui varie entre 138 (Pêche) et 202 (Politique de la concurrence).

3. Indexation de l'ensemble des documents législatifs

Nous avons également lancé notre programme sur l'ensemble des documents législatifs indexés dans EUR-Lex. Ce corpus de textes est beaucoup plus volumineux (on nous a fourni 730 documents), mais aussi beaucoup plus diversifié, puisque tous les chapitres y sont représentés.

Après un temps d'exécution de 32 heures, le taux moyen de descripteurs «manquants» était de 26 %, ce qui confirme l'hypothèse que les meilleurs résultats sont obtenus en construisant le modèle sur un ensemble de textes plus homogène.

XIII. Conclusion

Voici nos suggestions pour améliorer ce projet en vue d'applications futures.

Notre programme d'indexation pourra être amélioré par des moyens très simples, mais coûteux en termes de temps. Ces améliorations sont principalement d'ordre linguistique et sortent donc du cadre de ce projet. Nous proposons néanmoins quelques pistes.

- Comme le fichier de sortie de notre programme met en valeur les termes incorrectement indexés (en les mettant sur fond rouge) et comme il sépare les erreurs en deux feuilles — l'une pour les descripteurs «manquants», l'autre pour les descripteurs «en trop» —, il est facile d'identifier les types d'erreurs commises et de proposer des solutions visant une amélioration de l'indexation.

  Ainsi, l'analyse des résultats sur le chapitre de la pêche nous a permis de définir une règle supplémentaire (celle qui associe les noms de lieux) qui a réduit le pourcentage d'erreurs de 22,27 % à 16,04 %.

- Le programme actuel ne reconnaît pas les différentes déclinaisons des verbes. Ainsi, un verbe à l'infinitif et son participe passé sont considérés comme étant deux termes différents.

  Les performances pourront être améliorées si on intègre au programme une liste de tous les verbes et noms susceptibles d'être utilisés avec toutes leurs déclinaisons et formes possibles, de façon que les termes faisant partie d'un même champ grammatical soient considérés comme un seul terme.

- Les textes sur lesquels notre modèle a été construit sont en français. On peut supposer que le programme sera également efficace dans d'autres langues qui sont semblables au français du point de vue structurel, comme l'anglais (contrairement aux langues telles que l'allemand et le finnois). Il faudra effectuer des essais pour vérifier cette supposition.

Bien que notre système ait été construit avec le thésaurus Eurovoc et les textes d'EUR-Lex, il est probable qu'il pourra également être utilisé avec d'autres thésaurus et d'autres textes. CORDIS semble être un environnement dans lequel on peut effectuer de telles expériences.

Sous un angle plus personnel, notre projet à l'Office des publications nous a permis de découvrir le rôle que peuvent jouer les mathématiques dans un monde habituellement réservé aux linguistes et aux juristes. Nous avons toujours pensé qu'un texte pouvait être lu ou écrit, mais nous n'avions jamais réellement imaginé modéliser des textes avec l'aide des mathématiques.

À l'issue de nos quatre années d'étude de la théorie des mathématiques, ce projet nous a permis de mettre en pratique les concepts appris sur les thèmes des probabilités, des statistiques, de l'algorithmique, sur Java, VBA et sur les analyses de données. Ayant mis au point le système que nous voulions tester, nous nous sommes servi de connaissances en informatique pour mettre en œuvre les modèles mathématiques. C'est grâce à cette combinaison des mathématiques et de l'informatique que ce projet a pu être réalisé.

Nous sommes très heureux que notre système d'indexation ait suscité de l'intérêt au sein de l'unité «Journal officiel et accès au droit», où la direction considère la possibilité de faire appel à une entreprise externe dans le but de créer une application pouvant s'intégrer dans la structure d'information de l'Office et utilisant notre système pour l'indexation de documents.

Nous sommes également très heureux que notre système ait intéressé l'équipe CORDIS, qui gère une base de données sur la recherche et le développement pour les États membres. Si le thésaurus et l'ensemble des textes étaient adaptés à ce nouveau contexte (Eurovoc n'étant pas adapté pour l'instant aux documents portant sur la recherche et le développement), notre système pourrait aussi être essayé au sein de CORDIS.

Enfin, notre projet à l'Office des publications nous a aussi permis de vivre — dans une ambiance multiculturelle et enrichissante — la vie d'un stagiaire de la Commission, en nous engageant dans diverses activités, dont le *Journal des stagiaires de la Commission*, pour lequel nous avons écrit et édité plusieurs articles. Notre expérience à l'Office restera inoubliable et nous espérons pouvoir participer, de loin ou de près, au développement de notre système ainsi qu'à son intégration dans de nouveaux domaines.

Annexes

1. Code source du programme «Percentages1»

```
Option Explicit

Sub Percentages1()
    Dim i, j, k, n, Counter, Rows(1 To 3), bFound As Boolean
    Dim Percent(1 To 3), Descriptors() As String, StopWords() As String
    Dim Text() As String, Keywords() As String, Columns, Counter2, Counter3
    Columns = 0
    For i = 1 To 3
        Worksheets(i).Activate
        Worksheets(i).Cells(1, 1).Select
        Selection.End(xlDown).Select
        Rows(i) = Selection.Row
    Next
    ReDim Text(2 To Rows(1)): ReDim Descriptors(2 To Rows(2)): ReDim StopWords(2 To Rows(3))
    Worksheets(1).Activate
    For i = 2 To Rows(1)
        Text(i) = UCase(Worksheets(1).Cells(i, 1))
    Next
    For i = 2 To Rows(2)
        Descriptors(i) = UCase(Worksheets(2).Cells(i, 1))
    Next
    For i = 2 To Rows(3)
        StopWords(i) = UCase(Worksheets(3).Cells(i, 1))
    Next
    RemoveSW Text, StopWords
    Worksheets.Add , Worksheets(3)
    For i = 2 To Rows(1)
        Worksheets(1).Activate
        Do While Worksheets(1).Cells(i, Columns + 1).Value <> Empty
            Columns = Columns + 1
        Loop
        ReDim Keywords(Columns)
        For j = 2 To Columns
            Keywords(j) = UCase(Worksheets(1).Cells(i, j))
        Next
        RemoveSW Keywords, StopWords
        bFound = True
        For j = 2 To Columns
            If InStr(1, Text(i), " " & Keywords(j) & " ") = 0 Then
                bFound = False
                Exit For
            End If
            DoEvents
        Next
        If bFound = True Then Counter = Counter + 1
        Percent(1) = Counter / (Rows(1) - 1)
        Worksheets(1).Cells(1, 4) = Percent(1)
        Columns = 0
    Next
    Columns = 0
    Worksheets(1).Cells(1, 1).Select
    Worksheets(4).Name = "New indexation"
    Counter = 0
    For i = 2 To Rows(1)
        bFound = False: n = 2
        Worksheets(1).Activate
        Do While Worksheets(1).Cells(i, Columns + 1).Value <> Empty
            Columns = Columns + 1
        Loop
        ReDim Keywords(Columns)
```

```
        For j = 2 To Columns
            Keywords(j) = UCase(Worksheets(1).Cells(i, j))
        Next
        RemoveSW Keywords, StopWords
        Worksheets(4).Cells(i, 1).Value = Worksheets(1).Cells(i, 1).Value
        For j = 2 To Rows(2)
            If InStr(1, Text(i), " " & Descriptors(j) & " ") <> 0 Then
                Worksheets(4).Cells(i, n).Value = Worksheets(2).Cells(j, 1)
                Counter3 = Counter3 + 1
                n = n + 1
                For k = 2 To Columns
                    If Keywords(k) = Descriptors(j) And Keywords(k) <> Empty Then
                        Counter2 = Counter2 + 1
                        Worksheets(1).Cells(i, k).Interior.ColorIndex = 40
                        bFound = True
                    End If
                Next
            End If
            DoEvents
         Next
         If bFound = False Then Counter = Counter + 1
         Worksheets(1).Cells(i, 30) = Columns - 1 - Counter2
         If Columns <> 1 Then Worksheets(1).Cells(i, 31) = Format((Columns - 1 - Counter2) /
(Columns - 1), "Percent")
         Worksheets(4).Cells(i, 50) = Counter3 - Counter2
         If Counter3 <> 0 Then Worksheets(4).Cells(i, 51) = Format((Counter3 - Counter2) /
Counter3, "Percent")
         Counter2 = 0
         Percent(2) = Counter / (Rows(1) - 1)
         Worksheets(1).Cells(1, 5) = Percent(2)
         'Worksheets(4).Cells(1, 4) = i: Worksheets(4).Cells(1, 5) = Counter
        Columns = 0
    Next
End Sub
```

2. Code source du programme «DeleteWords»

```
Option Explicit

Public Sub DeleteWords()

    Dim arrWords() As String
    Dim i As Integer
    Dim j As Integer
    Dim k As Integer
    Dim strNoneAlpha As String
    Dim strWord As String
    Dim logNoneAlphaFound As Boolean
    Dim strText As String
    Dim strOldText As String

    Application.Cursor = xlWait

    strNoneAlpha = "{}[]|-@#$%^&*()_-+=<>/\?!~" & Chr(34) & Chr(160) & Chr(171) & Chr(176) &
Chr(180) & _
                                                Chr(186) & Chr(187) & Chr(215) & Chr(216) &
Chr(248)

    i = 2
    While Trim(ThisWorkbook.Sheets("data").Cells(i, 1).Value) <> ""

        strText = ""
        strOldText = ThisWorkbook.Sheets("data").Cells(i, 1).Value
        strOldText = Replace(strOldText, Chr(34), " ") ' remove double quotes (")
        strOldText = Replace(strOldText, Chr(39), " ") ' remove single quotes (')
        arrWords = Split(strOldText, " ")
        For j = 0 To UBound(arrWords)
```

```
            DoEvents
            strWord = Trim(arrWords(j))
            If Len(strWord) > 1 Then
                logNoneAlphaFound = False
                For k = 1 To Len(strNoneAlpha)
                    If InStr(1, strWord, Mid(strNoneAlpha, k, 1)) <> 0 Then
                        logNoneAlphaFound = True
                        Exit For
                    End If
                Next k
                If Not logNoneAlphaFound Then
                    strText = strText & " " & strWord
                End If
            End If
        Next j
        ThisWorkbook.Sheets("data").Cells(i, 1).Value = "'" & Trim(strText) & "'"
        i = i + 1
    Wend
    Application.Cursor = xlDefault
End Sub
```

3. Code source du programme «DeleteDescriptors»

```
Public Sub DeleteDescriptors()

    Dim i As Integer
    Dim j As Integer
    Dim k As Integer
    Dim iNumTexts As Integer
    Dim iNumAssigned As Integer
    Dim strDescriptor As String
    Dim strKeywords As String
    Dim arrKeywords() As String

    Application.Cursor = xlWait

    i = 2
    iNumTexts = 0
    While Trim(ThisWorkbook.Sheets("data").Cells(i, 1).Value) <> ""
        If Len(ThisWorkbook.Sheets("data").Cells(i, 1).Value) >= 2000 Then
            j = 2
            strKeywords = ""
            While Trim(ThisWorkbook.Sheets("data").Cells(i, j).Value) <> ""
                strKeywords = strKeywords & "|" & UCase(Trim(ThisWorkbook.Sheets("data").
Cells(i, j).Value))
                j = j + 1
            Wend
            If strKeywords <> "" Then
                iNumTexts = iNumTexts + 1
                ReDim Preserve arrKeywords(iNumTexts)
                arrKeywords(iNumTexts - 1) = strKeywords & "|"
            End If
        End If
        i = i + 1
    Wend
    i = 2
    strDescriptor = UCase(Trim(ThisWorkbook.Sheets("descriptors").Cells(i, 1).Value))
    While strDescriptor <> ""
        DoEvents
        iNumAssigned = 0
        For j = 0 To iNumTexts - 1
            If InStr(1, arrKeywords(j), "|" & strDescriptor & "|") <> 0 Then iNumAssigned =
iNumAssigned + 1
        Next j
        If iNumAssigned < 5 Then
            ThisWorkbook.Sheets("descriptors").Cells(i, 1).Value = ""
        End If
```

```
        i = i + 1
        strDescriptor = UCase(Trim(ThisWorkbook.Sheets("descriptors").Cells(i, 1).Value))
    Wend
    With ThisWorkbook.Sheets("descriptors")
        .Range("A2:A" & i).Sort Key1:=.Range("A2"), Order1:=xlAscending, Header:= _
        xlGuess, OrderCustom:=1, MatchCase:=False, Orientation:=xlTopToBottom, _
        DataOption1:=xlSortNormal
    End With
    Application.Cursor = xlDefault
End Sub
```

4. Code source du programme «AssignedTexts»

```
Public Sub AssignedTexts()

    Dim i As Integer
    Dim j As Integer
    Dim k As Integer
    Dim l As Integer
    Dim strKeywords As String
    Dim strDescriptor As String

    Application.Cursor = xlWait

    Application.DisplayAlerts = False
    i = 1
    While i <= ThisWorkbook.Sheets.Count
        If ThisWorkbook.Sheets(i).Name = "new" Then
            ThisWorkbook.Sheets(i).Delete
        Else
            i = i + 1
        End If
    Wend
    Application.DisplayAlerts = True

    ThisWorkbook.Sheets.Add after:=ThisWorkbook.Sheets(ThisWorkbook.Sheets.Count)
    With ThisWorkbook.Sheets(ThisWorkbook.Sheets.Count)
        .Name = "new"
        .Cells(1, 1).FormulaR1C1 = "TEXTS"
        .Cells(1, 2).FormulaR1C1 = "ASSIGNED"
        .Rows("1:1").Font.Bold = True
        .Rows("1:1").Interior.ColorIndex = 6
        .Rows("1:1").Interior.Pattern = xlSolid
        .Columns("B:B").HorizontalAlignment = xlCenter
        .Columns("B:B").EntireColumn.AutoFit
    End With
    i = 2
    While Trim(ThisWorkbook.Sheets("data").Cells(i, 1).Value) <> ""
        DoEvents
        ThisWorkbook.Sheets("new").Cells(i, 1).Value = "'" & ThisWorkbook.Sheets("data").
Cells(i, 1).Value
        ThisWorkbook.Sheets("new").Cells(i, 2).Value = 0
        j = 2
        strKeywords = ""
        While Trim(ThisWorkbook.Sheets("data").Cells(i, j).Value) <> ""
strKeywords = strKeywords & "|" & UCase(Trim(ThisWorkbook.Sheets("data").Cells(i, j).Value))
            j = j + 1
        Wend
        strKeywords = strKeywords & "|"
        k = 2
        Do While Trim(ThisWorkbook.Sheets("descriptors").Cells(k, 1).Value) <> ""
            strDescriptor = "|" & UCase(Trim(ThisWorkbook.Sheets("descriptors").Cells(k, 1).
Value)) & "|"
            If InStr(1, strKeywords, strDescriptor) <> 0 Then
                ThisWorkbook.Sheets("new").Cells(i, 2).Value = 1
                Exit Do
            End If
```

```
            k = k + 1
        Loop
        i = i + 1
    Wend
    Application.Cursor = xlDefault
End Sub
```

5. Code source du programme «AssignedDescriptors»

```
Public Sub AssignedDescriptors()

    Dim i As Integer
    Dim j As Integer
    Dim k As Integer
    Dim l As Integer
    Dim strKeywords As String
    Dim strDescriptor As String
    Dim strDescriptor2 As String
    Dim arrKeywords() As String
    Dim arrTexts() As String
    Dim iNumTexts As Integer

    Application.Cursor = xlWait

    Application.DisplayAlerts = False
    i = 1
    While i <= ThisWorkbook.Sheets.Count
        If Left(ThisWorkbook.Sheets(i).Name, 4) = "D = " Then
            ThisWorkbook.Sheets(i).Delete
        Else
            i = i + 1
        End If
    Wend
    Application.DisplayAlerts = True
    i = 2
    iNumTexts = 0
    While Trim(ThisWorkbook.Sheets("data").Cells(i, 1).Value) <> ""
        j = 2
        strKeywords = ""
        While Trim(ThisWorkbook.Sheets("data").Cells(i, j).Value) <> ""
strKeywords = strKeywords & "|" & UCase(Trim(ThisWorkbook.Sheets("data").Cells(i, j).Value))
            j = j + 1
        Wend
        iNumTexts = iNumTexts + 1
        ReDim Preserve arrKeywords(iNumTexts)
        ReDim Preserve arrTexts(iNumTexts)
        arrKeywords(iNumTexts - 1) = strKeywords & "|"
        arrTexts(iNumTexts - 1) = Trim(ThisWorkbook.Sheets("data").Cells(i, 1).Value)
        i = i + 1
    Wend
    i = 2
    strDescriptor = Trim(ThisWorkbook.Sheets("descriptors").Cells(i, 1).Value)
    While strDescriptor <> ""
        DoEvents
        strDescriptor2 = "D = " & Left(strDescriptor, 27)
        ThisWorkbook.Sheets.Add after:=ThisWorkbook.Sheets(ThisWorkbook.Sheets.Count)
        With ThisWorkbook.Sheets(ThisWorkbook.Sheets.Count)
            .Name = strDescriptor2
            .Cells(1, 1).FormulaR1C1 = "TEXTS"
            .Cells(1, 2).FormulaR1C1 = "ASSIGNED"
            .Rows("1:1").Font.Bold = True
            .Rows("1:1").Interior.ColorIndex = 6
            .Rows("1:1").Interior.Pattern = xlSolid
            .Columns("B:B").HorizontalAlignment = xlCenter
            .Columns("B:B").EntireColumn.AutoFit
            For j = 0 To iNumTexts - 1
                .Cells(j + 2, 1).Value = "'" & arrTexts(j)
```

```
                If InStr(1, arrKeywords(j), "|" & UCase(strDescriptor) & "|") <> 0 Then
                    .Cells(j + 2, 2).Value = 1
                Else
                    .Cells(j + 2, 2).Value = 0
                End If
            Next j
        End With
        i = i + 1
        strDescriptor = Trim(ThisWorkbook.Sheets("descriptors").Cells(i, 1).Value)
    Wend
    Application.Cursor = xlDefault
End Sub
```

6. Code source du programme «RunAll»

```
Public Sub RunAll()
    DeleteWords
    DeleteDescriptors
    AssignedTexts
    AssignedDescriptors
End Sub
```

7. Code source du programme «Separate.bas»

```
Sub Separate()
    Dim Rows, Columns, Text, i, j, k
    k = 1
    Cells(1, 1).Select
    Selection.End(xlDown).Select
    Rows = Selection.Row
    For i = 1 To Rows
        ActiveSheet.Range("A" & CStr(i)).End(xlToRight).Select
        Columns = Selection.Column
        Cells(i, 1).Select
        Text = Selection.Value
        For j = 1 To Columns - 1
            Worksheets(ActiveSheet.Index + 1).Cells(k, 1).Value = Text
            Worksheets(ActiveSheet.Index + 1).Cells(k, 2).Value = ActiveSheet.Cells(i, j + 1).
Value
            k = k + 1
        Next
    Next
End Sub
```

8. Code source du programme d'indexation en Java

i) Classe de lancement

```
package j48textclassifier;
public class Main {
    public static void main(String[] args) {
        MainFrame frame = new MainFrame();
        frame.setVisible(true);
    }
}
```

ii) Classe principale

```
package j48textclassifier;
import java.awt.HeadlessException;
import java.awt.event.ActionEvent;
```

```
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.File;
import java.io.IOException;
import java.text.NumberFormat;

import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JProgressBar;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;
import javax.swing.filechooser.FileFilter;

import jxl.Sheet;
import jxl.Workbook;
import jxl.format.Colour;
import jxl.read.biff.BiffException;
import jxl.write.Label;
import jxl.write.WritableCell;
import jxl.write.WritableCellFeatures;
import jxl.write.WritableCellFormat;
import jxl.write.WritableSheet;
import jxl.write.WritableWorkbook;
import weka.classifiers.trees.J48;
import weka.core.Attribute;
import weka.core.FastVector;
import weka.core.Instance;
import weka.core.Instances;
import weka.filters.unsupervised.attribute.StringToWordVector;
import javax.swing.JCheckBox;

public class MainFrame extends JFrame implements Runnable {
    private static final long serialVersionUID = 3256718476985774903L;
    public String dataFileName;
    public String testFileName;
    private JPanel jPanel = null;
    public JButton jButton = null;
    private JLabel jLabel = null;
    public JLabel dataFileNameLabel = null;
    private JLabel jLabel1 = null;
    public JLabel testFileNameLabel = null;
    public JButton jButton1 = null;
    private JProgressBar jProgressBar = null;
    public JButton jButton2 = null;

    private String[] dataTexts;
    private String[] testTexts;
    private String[] descriptors;
    private String[][] dataResults;
    private String[][] newDataResults;
    private String[][] testResults;
    private JLabel jLabel2 = null;
    private JLabel currentActionLabel = null;
    private JButton jButton3 = null;
    public Thread thread = null;
    public boolean checkInitialData = false;
    public JCheckBox jCheckBox = null;
    /**
     * This method initializes
     *
     */
    public MainFrame() {
        super();
        initialize();
    }
```

```
/**
 * This method initializes this
 *
 */
private void initialize() {
    this.setSize(295, 247);
    this.setTitle("J48 text classifier");
    this.setContentPane(getJPanel());
    this.addWindowListener(new MyWindowAdapter(this));
    dataFileName = "data.xls";
    testFileName = "new.xls";
}
/**
 * This method initializes jPanel
 *
 * @return javax.swing.JPanel
 */
private JPanel getJPanel() {
    if (jPanel == null) {
        currentActionLabel = new JLabel();
        currentActionLabel.setBounds(105, 125, 170, 20);
        currentActionLabel.setText("");
        jLabel2 = new JLabel();
        jLabel2.setBounds(5, 125, 100, 20);
        jLabel2.setText("CurrentAction:");
        dataFileNameLabel = new JLabel();
        dataFileNameLabel.setBounds(105, 5, 170, 20);
        dataFileNameLabel.setText("data.xls");
        testFileNameLabel = new JLabel();
        testFileNameLabel.setBounds(105, 60, 170, 20);
        testFileNameLabel.setText("new.xls");
        jLabel1 = new JLabel();
        jLabel1.setBounds(5, 60, 100, 20);
        jLabel1.setText("Excel test file:");
        jLabel = new JLabel();
        jLabel.setBounds(5, 5, 100, 20);
        jLabel.setText("Excel data file:");
        jPanel = new JPanel();
        jPanel.setLayout(null);
        jPanel.add(getJCheckBox(), null);
        jPanel.add(currentActionLabel, null);
        jPanel.add(jLabel2, null);
        jPanel.add(getJButton2(), null);
        jPanel.add(getJProgressBar(), null);
        jPanel.add(getJButton1(), null);
        jPanel.add(testFileNameLabel, null);
        jPanel.add(jLabel1, null);
        jPanel.add(dataFileNameLabel, null);
        jPanel.add(jLabel, null);
        jPanel.add(getJButton(), null);
    }
    return jPanel;
}
/**
 * This method initializes jButton
 *
 * @return javax.swing.JButton
 */
private JButton getJButton() {
    if (jButton == null) {
        jButton = new JButton();
        jButton.setSize(125, 25);
        jButton.setLocation(80, 30);
        jButton.setText("Choose data file");
        jButton.addActionListener(new MyActionListener(this, "data"));
    }
    return jButton;
}
```

```
    /**
     * This method initializes jButton1
     *
     * @return javax.swing.JButton
     */
    private JButton getJButton1() {
        if (jButton1 == null) {
            jButton1 = new JButton();
            jButton1.setSize(125, 25);
            jButton1.setLocation(80, 90);
            jButton1.setText("Choose test file");
            jButton1.addActionListener(new MyActionListener(this, "test"));
        }
        return jButton1;
    }
    /**
     * This method initializes jProgressBar
     *
     * @return javax.swing.JProgressBar
     */
    private JProgressBar getJProgressBar() {
        if (jProgressBar == null) {
            jProgressBar = new JProgressBar();
            jProgressBar.setBounds(5, 152, 270, 20);
        }
        return jProgressBar;
    }
    /**
     * This method initializes jButton2
     *
     * @return javax.swing.JButton
     */
    private JButton getJButton2() {
        if (jButton2 == null) {
            jButton2 = new JButton();
            jButton2.setSize(135, 25);
            jButton2.setLocation(138, 179);
            jButton2.setText("Start processing");
            jButton2.addActionListener(new MyActionListener(this, "process"));
        }
        return jButton2;
    }
    private void enableButtons() {
        jButton.setEnabled(true);
        jButton1.setEnabled(true);
        jButton2.setEnabled(true);
        jProgressBar.setValue(0);
        jCheckBox.setEnabled(true);
    }
    public void run() {
        try {
            try {
                currentActionLabel.setText("Loading training data...");
                Workbook dataWorkbook = Workbook.getWorkbook(new File(dataFileName));
                Sheet dataSheet = dataWorkbook.getSheet("data");
                if (dataSheet == null)
                    throw new InvalidFormatException("There is no sheet named data.");
                if (dataSheet.getColumns() < 1)
                    throw new InvalidFormatException("Data sheet should contain at least one
column.");
                int actualRows;
                for(actualRows=1;actualRows<dataSheet.getRows();actualRows++)
                    if (dataSheet.getCell(0, actualRows).getContents().length() == 0)
                        break;
                dataTexts = new String[actualRows - 1];
                for (int i = 0; i < dataTexts.length; i++)
                    dataTexts[i] = dataSheet.getCell(0, i + 1).getContents();
                Sheet descriptorSheet = dataWorkbook.getSheet("descriptors");
                if (descriptorSheet == null)
                    throw new InvalidFormatException("There is no sheet named descriptors.");
```

```
                if (descriptorSheet.getColumns() != 1)
                    throw new InvalidFormatException("Descriptor sheet should contain exactly
one column.");
                for(actualRows=1;actualRows<descriptorSheet.getRows();actualRows++)
                    if (descriptorSheet.getCell(0, actualRows).getContents().length() == 0)
                        break;
                descriptors = new String[actualRows - 1];
                dataResults = new String[descriptors.length][dataTexts.length];
                newDataResults = new String[descriptors.length][dataTexts.length];
                jProgressBar.setMaximum(descriptors.length);
                for (int i = 0; i < descriptors.length; i++) {
                    jProgressBar.setValue(i);
                    descriptors[i] = descriptorSheet.getCell(0, i + 1).getContents();
                    Sheet resultSheet;
                    if (descriptors[i].length() > 27)
                        resultSheet = dataWorkbook.getSheet("D = " + descriptors[i].
substring(0, 27));
                    else
                        resultSheet = dataWorkbook.getSheet("D = " + descriptors[i]);
                    if (resultSheet == null)
                        throw new InvalidFormatException("There is no sheet named D = " +
descriptors[i] + ".");
                    if (resultSheet.getColumns() != 2)
                        throw new InvalidFormatException("There should be 2 columns in sheet D
= " + descriptors[i] + ".");
                    for (int j = 0; j < dataTexts.length; j++) {
                        if (!resultSheet.getCell(0, j + 1).getContents().equals(dataTexts[j]))
                            throw new InvalidFormatException("Cell A" + (j + 1) + " in sheet D
= " + descriptors[i]
                                    + " should contain the same text as cell A" + j + " in
sheet data.");
                        dataResults[i][j] = resultSheet.getCell(1, j + 1).getContents();
                    }
                }
            } catch (IOException ex) {
                JOptionPane.showMessageDialog(this, "Error opening data file!", "Error",
JOptionPane.ERROR_MESSAGE);
                enableButtons();
                return;
            } catch (BiffException ex) {
                JOptionPane.showMessageDialog(this, "Error reading data file! Maybe not xls
format.", "Error", JOptionPane.ERROR_MESSAGE);
                enableButtons();
                return;
            } catch (InvalidFormatException ex) {
                JOptionPane.showMessageDialog(this, "Error parsing data file! " + ex.
getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
                enableButtons();
                return;
            }
            try {
                currentActionLabel.setText("Loading test data...");
                Workbook testWorkbook = Workbook.getWorkbook(new File(testFileName));
                Sheet dataSheet = testWorkbook.getSheet("data");
                if (dataSheet == null)
                    throw new InvalidFormatException("There is no sheet named data.");
                if (dataSheet.getColumns() < 1)
                    throw new InvalidFormatException("Data sheet should contain at least one
column.");
                testTexts = new String[dataSheet.getRows() - 1];
                jProgressBar.setMaximum(testTexts.length);
                testResults = new String[descriptors.length][testTexts.length];
                for (int i = 0; i < testTexts.length; i++) {
                    testTexts[i] = dataSheet.getCell(0, i + 1).getContents();
                }
            } catch (IOException ex) {
                JOptionPane.showMessageDialog(this, "Error opening test file!", "Error",
JOptionPane.ERROR_MESSAGE);
                enableButtons();
                return;
```

```
            } catch (BiffException ex) {
                JOptionPane.showMessageDialog(this, "Error reading test file! Maybe not xls
format.", "Error", JOptionPane.ERROR_MESSAGE);
                enableButtons();
                return;
            } catch (InvalidFormatException ex) {
                JOptionPane.showMessageDialog(this, "Error parsing test file! " + ex.
getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
                enableButtons();
                return;
            }
            try {
                WritableCellFormat redFormat = new WritableCellFormat();
                WritableCellFormat greenFormat = new WritableCellFormat();
                WritableCellFormat yellowFormat = new WritableCellFormat();
                redFormat.setBackground(Colour.RED);
                greenFormat.setBackground(Colour.LIGHT_GREEN);
                yellowFormat.setBackground(Colour.YELLOW);
                currentActionLabel.setText("Processing data...");
                jProgressBar.setMaximum(descriptors.length);
                WritableWorkbook outputs = Workbook.createWorkbook(new File("results.xls"));
                for (int i = 0; i < descriptors.length; i++) {
                    jProgressBar.setValue(i);
                    FastVector attInfo = new FastVector();
                    Attribute a1 = new Attribute("Text", (FastVector) null);
                    FastVector outClasses = new FastVector();
                    outClasses.addElement(new String("0"));
                    outClasses.addElement(new String("1"));
                    Attribute a2 = new Attribute("Result", outClasses);
                    attInfo.addElement(a1);
                    attInfo.addElement(a2);
                    Instances initialData = new Instances("training", attInfo, dataTexts.
length);
                    initialData.setClass(a2);
                    Instance inst;
                    StringToWordVector filter = new StringToWordVector();
                    filter.setDelimiters(" \t.,:'\"()?!");
                    filter.setIDFTransform(true);
                    filter.setLowerCaseTokens(true);
                    filter.setTFTransform(true);
                    filter.setUseStoplist(true);
                    filter.setWordsToKeep(1000);
                    filter.setInputFormat(initialData);
                    for (int j = 0; j < dataTexts.length; j++) {
                        inst = new Instance(2);
                        inst.setDataset(initialData);
                        inst.setValue(a1, dataTexts[j]);
                        inst.setValue(a2, dataResults[i][j]);
                        filter.input(inst);
                    }
                    for (int j = 0; j < testTexts.length; j++) {
                        inst = new Instance(2);
                        inst.setDataset(initialData);
                        inst.setValue(a1, testTexts[j]);
                        inst.setMissing(a2);
                        filter.input(inst);
                    }
                    if (checkInitialData)
                    for (int j = 0; j < dataTexts.length; j++) {
                        inst = new Instance(2);
                        inst.setDataset(initialData);
                        inst.setValue(a1, dataTexts[j]);
                        inst.setMissing(a2);
                        filter.input(inst);
                    }
                    filter.batchFinished();
                    Instances trainData = filter.outputFormat();
                    Instance processed;
                    for (int j = 0; j < dataTexts.length; j++) {
                        processed = filter.output();
```

```
                        trainData.add(processed);
                    }
                    J48 classifier = new J48();
                    classifier.buildClassifier(trainData);
                    try {
                        for (int j = 0; j < testTexts.length; j++) {
                            inst = filter.output();
                            double[] result = classifier.distributionForInstance(inst);
                            if (result[0] > result[1])
                                testResults[i][j] = "0";
                            else
                                testResults[i][j] = "1";
                        }
                        if (checkInitialData)
                        for (int j = 0; j < dataTexts.length; j++) {
                            inst = filter.output();
                            double[] result = classifier.distributionForInstance(inst);
                            if (result[0] > result[1])
                                newDataResults[i][j] = "0";
                            else
                                newDataResults[i][j] = "1";
                        }
                    } catch (Exception ex) {
                        JOptionPane.showMessageDialog(this, "Error while processing data: " +
ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
                        ex.printStackTrace();
                        enableButtons();
                        return;
                    }
                }
                int currentColumn = 0;
                WritableSheet sheet = outputs.createSheet("Results", 0);
                sheet.addCell(new Label(0, currentColumn, "New texts", yellowFormat));
                sheet.addCell(new Label(1, currentColumn++, "Descriptors", yellowFormat));
                int currentIndex;
                for (int i = 0; i < testTexts.length; i++) {
                    sheet.addCell(new Label(0, currentColumn, testTexts[i]));
                    currentIndex = 1;
                    for (int j = 0; j < descriptors.length; j++)
                        if (testResults[j][i].equals("1"))
                            sheet.addCell(new Label(currentIndex++, currentColumn,
descriptors[j]));
                    currentColumn++;
                }
                if (checkInitialData) {
                    sheet = outputs.createSheet("Initial data 1", 1);
                    currentColumn = 0;
                    sheet.addCell(new Label(0, currentColumn, "Text", yellowFormat));
                    sheet.addCell(new Label(1, currentColumn, "Percentage not assigned",
yellowFormat));
                    sheet.addCell(new Label(2, currentColumn++, "Initial descriptors",
yellowFormat));
                    int notAssigned;
                    int totalAssigned;
                    for (int i = 0; i < dataTexts.length; i++) {
                        sheet.addCell(new Label(0, currentColumn, dataTexts[i]));
                        currentIndex = 2;
                        notAssigned = 0;
                        totalAssigned = 0;
                        for (int j = 0; j < descriptors.length; j++)
                            if (dataResults[j][i].equals("1") && newDataResults[j][i].
equals("0")) {
                                sheet.addCell(new Label(currentIndex++, currentColumn,
descriptors[j], redFormat));
                                notAssigned++;
                                totalAssigned++;
                            } else if (dataResults[j][i].equals("1")) {
                                sheet.addCell(new Label(currentIndex++, currentColumn,
descriptors[j], greenFormat));
                                totalAssigned++;
```

```
                        }
                    if (totalAssigned > 0)
                        sheet.addCell(new Label(1, currentColumn, NumberFormat.
getInstance().format((1.0 * notAssigned / totalAssigned) * 100)));
                    currentColumn++;
                }
                sheet = outputs.createSheet("Initial data 2", 2);
                currentColumn = 0;
                sheet.addCell(new Label(0, currentColumn, "Text", yellowFormat));
                sheet.addCell(new Label(1, currentColumn, "Percentage badly assigned",
yellowFormat));
                sheet.addCell(new Label(2, currentColumn++, "New descriptors",
yellowFormat));
                int badlyAssigned;
                for (int i = 0; i < dataTexts.length; i++) {
                    sheet.addCell(new Label(0, currentColumn, dataTexts[i]));
                    currentIndex = 2;
                    badlyAssigned = 0;
                    totalAssigned = 0;
                    for (int j = 0; j < descriptors.length; j++)
                        if (dataResults[j][i].equals("0") && newDataResults[j][i].
equals("1")) {
                            sheet.addCell(new Label(currentIndex++, currentColumn,
descriptors[j], redFormat));
                            badlyAssigned++;
                            totalAssigned++;
                        } else if (newDataResults[j][i].equals("1")) {
                            sheet.addCell(new Label(currentIndex++, currentColumn,
descriptors[j], greenFormat));
                            totalAssigned++;
                        }
                    if (totalAssigned > 0)
                        sheet.addCell(new Label(1, currentColumn, NumberFormat.
getInstance().format((1.0 * badlyAssigned / totalAssigned) * 100)));
                    currentColumn++;
                }
            }
            outputs.write();
            outputs.close();
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(this, "Error while processing data: " + ex.
getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
            ex.printStackTrace();
            enableButtons();
            return;
        }
        JOptionPane.showMessageDialog(this, "Done! Results saved in results.xls", "Done",
JOptionPane.INFORMATION_MESSAGE);
        enableButtons();
        System.gc();
    } catch (HeadlessException e) {
        e.printStackTrace();
    }
}
public void dispose() {
    if (thread != null)
        thread.stop();
    super.dispose();
}
/**
 * This method initializes jCheckBox
 *
 * @return javax.swing.JCheckBox
 */
private JCheckBox getJCheckBox() {
    if (jCheckBox == null) {
        jCheckBox = new JCheckBox();
        jCheckBox.setBounds(14, 180, 122, 25);
        jCheckBox.setText("Verify initial data");
        jCheckBox.addActionListener(new MyCheckBoxActionListener(this));
```

```
        }
        return jCheckBox;
    }
    public static void main(String[] args) {
        MainFrame frame = new MainFrame();
        frame.setVisible(true);
    }
} // @jve:decl-index=0:visual-constraint="139,6"

class MyCheckBoxActionListener implements ActionListener{
    MainFrame parent;

    public MyCheckBoxActionListener(MainFrame parent){
        this.parent = parent;
    }
    public void actionPerformed(ActionEvent arg0) {
        parent.checkInitialData = !parent.checkInitialData;
    }
}
class MyActionListener implements ActionListener {
    MainFrame parentFrame = null;
    String target = null;
    public MyActionListener(MainFrame parentFrame, String target) {
        this.parentFrame = parentFrame;
        this.target = target;
    }
    public void actionPerformed(ActionEvent e) {
        if (target.equals("process")) {
            Thread thread = new Thread(parentFrame);
            thread.start();
            parentFrame.thread = thread;
            parentFrame.jButton.setEnabled(false);
            parentFrame.jButton1.setEnabled(false);
            parentFrame.jButton2.setEnabled(false);
            parentFrame.jCheckBox.setEnabled(false);
            return;
        }
        JFileChooser chooser = new JFileChooser();
        FileFilter filter = new FileFilter() {
            public String getDescription() {
                return "Excel File";
            }
            public boolean accept(File file) {
                if (file.isDirectory())
                    return true;
                String fileName = file.getName();
                String extension = fileName.substring(fileName.lastIndexOf('.'));
                if (extension.equals(".xls"))
                    return true;
                else
                    return false;
            }
        };
        chooser.setFileFilter(filter);
        chooser.setAcceptAllFileFilterUsed(false);
        int returnVal = chooser.showOpenDialog(parentFrame);
        if (returnVal == JFileChooser.APPROVE_OPTION) {
            if (target.equals("data")) {
                parentFrame.dataFileName = chooser.getCurrentDirectory().getAbsolutePath() +
"\\" + chooser.getSelectedFile().getName();
                parentFrame.dataFileNameLabel.setText(parentFrame.dataFileName);
            } else if (target.equals("test")) {
                parentFrame.testFileName = chooser.getCurrentDirectory().getAbsolutePath() +
"\\" + chooser.getSelectedFile().getName();
                parentFrame.testFileNameLabel.setText(parentFrame.testFileName);
            }
        }
    }
}
```

```
class InvalidFormatException extends RuntimeException {
    private static final long serialVersionUID = 4050767087247374384L;
    String message = null;
    public InvalidFormatException(String message) {
        this.message = message;
    }
    public String getMessage() {
        return message;
    }
}
class MyWindowAdapter extends WindowAdapter {
    MainFrame parent;
    public MyWindowAdapter(MainFrame parent) {
        this.parent = parent;
    }
    public void windowClosing(WindowEvent arg0) {
        if (parent.thread != null)
            parent.thread.stop();
        parent.dispose();
        super.windowClosing(arg0);
    }
}
```

9. Les différentes mesures d'erreur pour le descripteur «accord de pêche» ([1])

```
J48 pruned tree
------------------

reconduit <= 0
|   contrepartie <= 0
|   |   désigner <= 0: 0 (103.0/1.0)
|   |   désigner > 0
|   |   |   baltique <= 0: 0 (4.0)
|   |   |   baltique > 0: 1 (2.0)
|   contrepartie > 0
|   |   ensemble <= 0: 1 (14.0/1.0)
|   |   ensemble > 0: 0 (6.0)
reconduit > 0: 1 (9.0)

Number of Leaves :     6

Size of the tree:     11

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         131               94.9275 %
Incorrectly Classified Instances         7                5.0725 %
Kappa statistic                          0.8206
Mean absolute error                      0.0633
Root mean squared error                  0.2258
Relative absolute error                 21.1026 %
Root relative squared error             58.5611 %
Total Number of Instances              138

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   Class
  0.982     0.2        0.957     0.982     0.969    0
  0.8       0.018      0.909     0.8       0.851    1

=== Confusion Matrix ===

   a   b   <-- classified as
 111   2 |   a = 0
   5  20 |   b = 1
```

([1]) Calculées dans l'environnement Weka.

10. Description du format .arff ([1])

Attribute-Relation File Format (ARFF)

April 1st, 2002

An ARFF (Attribute-Relation File Format) file is an ASCII text file that describes a list of instances sharing a set of attributes. ARFF files were developed by the Machine Learning Project at the Department of Computer Science of The University of Waikato for use with the Weka machine learning software. This document describes the version of ARFF used with Weka versions 3.2 to 3.3; this is an extension of the ARFF format as described in the data mining book written by Ian H. Witten and Eibe Frank (the new additions are string attributes, date attributes, and sparse instances).

This explanation was cobbled together by Gordon Paynter (gordon.paynter@ucr.edu) from the Weka 2.1 ARFF description, email from Len Trigg (lenbok@myrealbox.com) and Eibe Frank (eibe@cs.waikato.ac.nz), and some datasets. It has been edited by Richard Kirkby (rkirkby@cs.waikato.ac.nz). Contact Len if you're interested in seeing the ARFF 3 proposal.

Overview

ARFF files have two distinct sections. The first section is the **Header** information, which is followed the **Data** information.

The **Header** of the ARFF file contains the name of the relation, a list of the attributes (the columns in the data), and their types. An example header on the standard IRIS dataset looks like this:

```
% 1. Title: Iris Plants Database
%
% 2. Sources:
%      (a) Creator: R.A. Fisher
%      (b) Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
%      (c) Date: July, 1988
%
@RELATION iris

@ATTRIBUTE sepallength  NUMERIC
@ATTRIBUTE sepalwidth   NUMERIC
@ATTRIBUTE petallength  NUMERIC
@ATTRIBUTE petalwidth   NUMERIC
@ATTRIBUTE class        {Iris-setosa,Iris-versicolor,Iris-virginica}
```

The **Data** of the ARFF file looks like the following:

```
@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
```

Lines that begin with a % are comments. The **@RELATION**, **@ATTRIBUTE** and **@DATA** declarations are case insensitive.

Examples

Several well-known machine learning datasets are distributed with Weka in the $WEKAHOME/data directory as ARFF files.

([1]) Cette description est issue de l'université de Waikato en Nouvelle-Zélande.

The ARFF Header Section

The ARFF Header section of the file contains the relation declaration and attribute declarations.

The @relation Declaration

The relation name is defined as the first line in the ARFF file. The format is:

```
@relation <relation-name>
```

where <relation-name> is a string. The string must be quoted if the name includes spaces.

The @attribute Declarations

Attribute declarations take the form of an orderd sequence of **@attribute** statements. Each attribute in the data set has its own **@attribute** statement which uniquely defines the name of that attribute and it's data type. The order the attributes are declared indicates the column position in the data section of the file. For example, if an attribute is the third one declared then Weka expects that all that attributes values will be found in the third comma delimited column.

The format for the **@attribute** statement is:

```
@attribute <attribute-name> <datatype>
```

where the <attribute-name> must start with an alphabetic character. If spaces are to be included in the name then the entire name must be quoted.

The <datatype> can be any of the four types currently (version 3.2.1) supported by Weka:

numeric
<nominal-specification>
string
date [<date-format>]

where <nominal-specification> and <date-format> are defined below. The keywords **numeric**, **string** and **date** are case insensitive.

Numeric attributes

Numeric attributes can be real or integer numbers.

Nominal attributes

Nominal values are defined by providing an <nominal-specification> listing the possible values: {<nominal-name1>, <nominal-name2>, <nominal-name3>, ...}

For example, the class value of the Iris dataset can be defined as follows:

```
@ATTRIBUTE class        {Iris-setosa,Iris-versicolor,Iris-virginica}
```

Values that contain spaces must be quoted.

String attributes

String attributes allow us to create attributes containing arbitrary textual values. This is very useful in text-mining applications, as we can create datasets with string attributes, then write Weka Filters to manipulate strings (like StringToWordVectorFilter). String attributes are declared as follows:

```
@ATTRIBUTE LCC    string
```

Date attributes

Date attribute declarations take the form:

```
@attribute <name> date [<date-format>]
```

where <name> is the name for the attribute and <date-format> is an optional string specifying how date values should be parsed and printed (this is the same format used by SimpleDateFormat). The default format string accepts the ISO-8601 combined date and time format: "yyyy-MM-dd'T'HH:mm:ss".

Dates must be specified in the data section as the corresponding string representations of the date/time (see example below).

ARFF Data Section

The ARFF Data section of the file contains the data declaration line and the actual instance lines.

The @data Declaration

The **@data** declaration is a single line denoting the start of the data segment in the file. The format is:

```
@data
```

The instance data

Each instance is represented on a single line, with carriage returns denoting the end of the instance.

Attribute values for each instance are delimited by commas. They must appear in the order that they were declared in the header section (i.e. the data corresponding to the nth **@attribute** declaration is always the nth field of the attribute).

Missing values are represented by a single question mark, as in:

```
@data
4.4,?,1.5,?,Iris-setosa
```

Values of string and nominal attributes are case sensitive, and any that contain space must be quoted, as follows:

```
@relation LCCvsLCSH

@attribute LCC string
@attribute LCSH string

@data
AG5,   'Encyclopedias and dictionaries.;Twentieth century.'
AS262, 'Science -- Soviet Union -- History.'
AE5,   'Encyclopedias and dictionaries.'
AS281, 'Astronomy, Assyro-Babylonian.;Moon -- Phases.'
AS281, 'Astronomy, Assyro-Babylonian.;Moon -- Tables.'
```

Dates must be specified in the data section using the string representation specified in the attribute declaration. For example:

```
@RELATION Timestamps

@ATTRIBUTE timestamp DATE "yyyy-MM-dd HH:mm:ss"

@DATA
"2001-04-03 12:12:12"
"2001-05-03 12:59:55"
```

Sparse ARFF files

Sparse ARFF files are very similar to ARFF files, but data with value 0 are not be explicitly represented.

Sparse ARFF files have the same header (i.e **@relation** and **@attribute** tags) but the data section is different. Instead of representing each value in order, like this:

```
@data
0, X, 0, Y, "class A"
0, 0, W, 0, "class B"
```

the non-zero attributes are explicitly identified by attribute number and their value stated, like this:

```
@data
{1 X, 3 Y, 4 "class A"}
{2 W, 4 "class B"}
```

Each instance is surrounded by curly braces, and the format for each entry is: <index> <space> <value> where index is the attribute index (starting from 0).

Note that the omitted values in a sparse instance are **0**, they are not "missing" values! If a value is unknown, you must explicitly represent it with a question mark (?).

Warning: There is a known problem saving SparseInstance objects from datasets that have string attributes. In Weka, string and nominal data values are stored as numbers; these numbers act as indexes into an array of possible attribute values (this is very efficient). However, the first string value is assigned index 0: this means that, internally, this value is stored as a 0. When a SparseInstance is written, string instances with internal value 0 are not output, so their string value is lost (and when the arff file is read again, the default value 0 is the index of a different string value, so the attribute value appears to change). To get around this problem, add a dummy string value at index 0 that is never used whenever you declare string attributes that are likely to be used in SparseInstance objects and saved as Sparse ARFF files.

11. Liste des mots stop ([1])

(Référence: Eurodicautom, CELEX)

DA

EURODICAUTOM: af, at, de, den, det, en, et, for, fra, hos, med, og, om, på, til

CELEX: AF, AL, ALLE, ALT, ANDEN, ANDET, ANDRE, AT, BLEV, BLEVET, BLIVE, BLIVER, BOER, DA, DE, DEM, DEN, DENNE, DENNES, DENS, DER, DERES, DERFOR, DERTIL, DERVED, DET, DETS, DETTE, DETTES, DIG, DIN, DINE, DISSE, DISSES, DIT, DOG, DU, EFTER, ELLER, EN, END, ENHVER, ENTEN, ER, ET, ETHVERT, FLERE, FLEST, FOR, FORDI, FRA, GENNEM, HAFT, HAM, HAN, HANS, HAR, HAVDE, HELE, HELT, HENDE, HENDES, HERAF, HEROM, HERTIL, HUN, HVAD, HVEM, HVER, HVERKEN, HVERT, HVILKE, HVILKEN, HVILKET, HVIS, HVOR, HVORIMOD, HVORPAA, HVORTIL, HVORVED, IDET, IKKE, IMIDLERTID, INDEN, INDTIL, INGEN, INTET, JEG, JER, JERE, JERES, JERT, KAN, KUN, KUNNE, KUNNET, LIDT, MAA, MAATTE, MAATTET, MAN, MED, MELLEM, MEN, MIG, MIN, MIT, NAAR, NOGEN, NOGET, NOGLE, OG, OGSAA, OM, OP, OVER, PAA, SAA, SAADAN, SAADANNE, SAADANT, SAAFREMT, SAALEDES, SAMME, SAMMEN, SAMT, SIG, SIN, SINE, SIT, SKAL, SKULLET, SOM, TIL, UDEN, UNDER, VAERE, VAERET, VAR, VED, VEDROERENDE, VI, VIL, VILLE, VILLET, VISSE

DE

EURODICAUTOM: am, an, auf, aus, bei, beim, das, dem, den, der, des, die, ein, eine, einem, einen, einer, eines, für,im, in, mit, und, vom, von, vor, zu, zum, zur

CELEX: AB, ABER, ALLE, ALLEN, ALS, AM, AN, ANDERE, ANDEREM, ANDEREN, ANDERES, AUCH, AUF, AUFGRUND, AUS, AUSSER, BEI, BEIM, BESTIMMTE, BESTIMMTEN, BETREFFEND, BETREFFENDE, BETREFFENDEM, BETREFFENDEN, BETREFFENDER, BETREFFENDES, BIS, DA, DABEI, DADURCH, DAFUER, DAGEGEN, DAHER, DAHIN, DAHINTER, DAMIT, DANACH, DANEBEN, DANN, DAR, DARAN, DARAUF, DARAUS, DARIN, DARUEBER, DARUM, DARUNTER, DAS, DASS, DAVON, DAZU, DAZWISCHEN, DEM, DEN, DENEN, DER, DEREN, DERER, DES, DESHALB, DESSEN, DIE, DIESE, DIESEM, DIESEN, DIESER, DIESES, DRITTEN, DURCH, EIN, EINE, EINEM, EINEN, EINER, EINES, EINIGE, EINIGEM, EINIGEN, EINIGER, EINIGES, ENTSPRECHEN, ER, ERFORDERLICHEN , ES, FESTGELEGT, FESTGESETZ, FOLGENDE, FOLGENDEN, FOLGT, FUER, GEGEN, GELTEN, GEMAESS, GEMEINSAME, GEMEINSAMEN, GENANNTEN, GERICHTET, GESCHEHEN, GESTUETZT, GILT, HABEN, HAT, HERRN, IHM, IHN, IHNEN, IHR, IHRE, IHREN, IHRER, IHRES, IM, IN, INNERHALB, INS, INSBESONDERE, IST, JE, JEDE, JEDEM, JEDEN, JEDER, JEDES, JEDOCH, JEGLICHE, JEGLICHEM, JEGLICHEN, JEGLICHER, JEGLICHES, JENE, JENEM, JENEN, JENER, JENES, KANN, KEIN, KEINE, KEINEM, KEINEN, KEINER, KEINES, KOENNE, KOENNEN, KOENNTE, KOENNTEN, MAG, MEHRERE, MEHRERER, MEHRERER, MEIN, MEINE, MEINEM, MEINEN, MEINER, MEINES, MIT, MITHIN, MOEGEN, MUESSEN, MUESSTEN, MUSS, NACH, NACHDEM, NACHHER, NACHSTEHENDER, NICHT, NOCH, NR, NUR, OB, OBSCHON, ODER, OHNE, SEI, SEIEN, SEIN, SEINEM, SEINEN, SEINER, SEINES, SEIT, SEITDEM, SEITHER, SICH, SIE, SIND, SO, SOLLEN, SOLLTE, SOLLTEN, SONSTIGE,

([1]) Cette liste a été compilée par le Dr Bagola.

SONSTIGEM, SONSTIGEN, SONSTIGER, SONSTIGES, SOWIE, TEILEN, TRITT, UEBER, UEBERALL, UM, UND, UNTER, VOM, VON, VOR, VORGESEHENEN, WAEHREND, WAEHRENDDESSEN, VANN, WAR, WEGEN, WENN, WERDEN, WIDER, WIE, WIRD, WO, WOBEI, WODURCH, WOFUER, WOGEGEN, WOHER, WOHIN, WOHINTER, WOMIT, WONACH, WONEBEN, WORAN, WORAUF, WORAUS, WORDEN, WORIN, WORUEBER, WORUM, WORUNTER, WOVON, WOZU, WOZWISCHEN, WURDE, WURDEN, ZU, ZULETZ, ZUM, ZUR, ZWISCHEN

EL

EURODICAUTOM: απ, απο, γι, για, δι, δια, εισ, εκ, ενα, εναν, ενασ, ενοσ, εξ, επ, επι, καθ, και, κατ, κατα, με, μεσα, μια, μιαν, μιασ, περι, σε, στα, στη, στην, στισ, στο, στον, στουσ, τα, τη, την, τησ, τισ, το, τον, του, τουσ, των, υπο

CELEX: ΑΔΙΑΚΟΠΑ, ΑΙ, ΑΚΟΜΑ, ΑΚΟΜΗ, ΑΚΡΙΒΩΣ, ΑΛΗΘΕΙΑ, ΑΛΗΘΙΝΑ, ΑΛΛΑ, ΑΛΛΑΧΟΥ, ΑΛΛΕΣ, ΑΛΛΗ, ΑΛΛΗΝ, ΑΛΛΗΣ, ΑΛΛΙΩΣ, ΑΛΛΙΩΤΙΚΑ, ΑΛΛΟ, ΑΛΛΟΙ, ΑΛΛΟΙΩΣ, ΑΛΛΟΙΩΤΙΚΑ, ΑΛΛΟΝ, ΑΛΛΟΣ, ΑΛΛΟΤΕ, ΑΛΛΟΥ, ΑΛΛΟΥΣ, ΑΛΛΩΝ, ΑΜΑ, ΑΜΕΣΑ, ΑΜΕΣΩΣ, ΑΝ, ΑΝΑ, ΑΝΑΜΕΣΑ, ΑΝΑΜΕΤΑΞΥ, ΑΝΕΥ, ΑΝΤΙ, ΑΝΤΙΠΕΡΑ, ΑΝΤΙΣ, ΑΝΩ, ΑΝΩΤΕΡΩ, ΑΞΑΦΝΑ, ΑΠ, ΑΠΕΝΑΝΤΙ, ΑΠΟ, ΑΠΟΨΕ, ΑΡΑ, ΑΡΑΓΕ, ΑΡΓΑ, ΑΡΓΟΤΕΡΟ, ΑΡΙΣΤΕΡΑ, ΑΡΚΕΤΑ, ΑΡΧΙΚΑ, ΑΣ, ΑΥΡΙΟ, ΑΥΤΑ, ΑΥΤΕΣ, ΑΥΤΗ, ΑΥΤΗΝ, ΑΥΤΗΣ, ΑΥΤΟ, ΑΥΤΟΙ, ΑΥΤΟΝ, ΑΥΤΟΣ, ΑΥΤΟΥ, ΑΥΤΟΥΣ, ΑΥΤΩΝ, ΑΦΟΤΟΥ, ΑΦΟΥ, ΒΕΒΑΙΑ, ΒΕΒΑΙΟΤΑΤΑ, ΓΙ, ΓΙΑ, ΓΡΗΓΟΡΑ, ΓΥΡΩ, ΔΑ, ΔΕ, ΔΕΙΝΑ, ΔΕΝ, ΔΕΞΙΑ, ΔΗΘΕΝ, ΔΗΛΑΔΗ, ΔΙ, ΔΙΑ, ΔΙΑΡΚΩΣ, ΔΙΚΑ, ΔΙΚΟ, ΔΙΚΟΙ, ΔΙΚΟΣ, ΔΙΚΟΥ, ΔΙΚΟΥΣ, ΔΙΟΛΟΥ, ΔΙΠΛΑ, ΔΙΧΩΣ, ΕΑΝ, ΕΑΥΤΟ, ΕΑΥΤΟΝ, ΕΑΥΤΟΥ, ΕΑΥΤΟΥΣ, ΕΑΥΤΩΝ, ΕΓΚΑΙΡΑ, ΕΓΚΑΙΡΩΣ, ΕΓΩ, ΕΔΩ, ΕΙΔΕΜΗ, ΕΙΘΕ, ΕΙΜΑΙ, ΕΙΜΑΣΤΕ, ΕΙΝΑΙ, ΕΙΣ, ΕΙΣΑΙ, ΕΙΣΑΣΤΕ, ΕΙΣΤΕ, ΕΙΤΕ, ΕΙΧΑ, ΕΙΧΑΜΕ, ΕΙΧΑΝ, ΕΙΧΑΤΕ, ΕΙΧΕ, ΕΙΧΕΣ, ΕΚΑΣΤΑ, ΕΚΑΣΤΕΣ, ΕΚΑΣΤΗ, ΕΚΑΣΤΗΝ, ΕΚΑΣΤΗΣ, ΕΚΑΣΤΟ, ΕΚΑΣΤΟΙ, ΕΚΑΣΤΟΝ, ΕΚΑΣΤΟΣ, ΕΚΑΣΤΟΥ, ΕΚΑΣΤΟΥΣ, ΕΚΑΣΤΩΝ, ΕΚΕΙ, ΕΚΕΙΝΑ, ΕΚΕΙΝΕΣ, ΕΚΕΙΝΗ, ΕΚΕΙΝΗΝ, ΕΚΕΙΝΗΣ, ΕΚΕΙΝΟ, ΕΚΕΙΝΟΙ, ΕΚΕΙΝΟΝ, ΕΚΕΙΝΟΣ, ΕΚΕΙΝΟΥ, ΕΚΕΙΝΟΥΣ, ΕΚΕΙΝΩΝ, ΕΚΤΟΣ, ΕΜΑΣ, ΕΜΕΙΣ, ΕΜΕΝΑ, ΕΜΠΡΟΣ, ΕΝ, ΕΝΑ, ΕΝΑΝ, ΕΝΑΣ, ΕΝΟΣ, ΕΝΤΕΛΩΣ, ΕΝΤΟΣ, ΕΝΤΩΜΕΤΑΞΥ, ΕΝΩ, ΕΞ, ΕΞΑΦΝΑ, ΕΞΗΣ, ΕΞΙΣΟΥ, ΕΞΩ, ΕΠΑΝΩ, ΕΠΕΙΔΗ, ΕΠΕΙΤΑ, ΕΠΙ, ΕΠΙΣΗΣ, ΕΠΟΜΕΝΩΣ, ΕΣΑΣ, ΕΣΕΙΣ, ΕΣΕΝΑ, ΕΣΤΩ, ΕΣΥ, ΕΤΕΡΑ, ΕΤΕΡΑΙ, ΕΤΕΡΑΣ, ΕΤΕΡΕΣ, ΕΤΕΡΗ, ΕΤΕΡΗΣ, ΕΤΕΡΟ, ΕΤΕΡΟΙ, ΕΤΕΡΟΝ, ΕΤΕΡΟΣ, ΕΤΕΡΟΥ, ΕΤΕΡΟΥΣ, ΕΤΕΡΩΝ, ΕΤΟΥΤΑ, ΕΤΟΥΤΕΣ, ΕΤΟΥΤΗ, ΕΤΟΥΤΗΝ, ΕΤΟΥΤΗΣ, ΕΤΟΥΤΟ, ΕΤΟΥΤΟΙ, ΕΤΟΥΤΟΝ, ΕΤΟΥΤΟΣ, ΕΤΟΥΤΟΥ, ΕΤΟΥΤΟΥΣ, ΕΤΟΥΤΩΝ, ΕΤΣΙ, ΕΥΓΕ, ΕΥΘΥΣ, ΕΥΤΥΧΩΣ, ΕΦΕΞΗΣ, ΕΧΕΙ, ΕΧΕΙΣ, ΕΧΕΤΕ, ΕΧΘΕΣ, ΕΧΟΜΕ, ΕΧΟΥΜΕ, ΕΧΟΥΝ, ΕΧΤΕΣ, ΕΧΩ, ΕΩΣ, Η, ΗΔΗ, ΗΜΑΣΤΑΝ, ΗΜΑΣΤΕ, ΗΜΟΥΝ, ΗΣΑΣΤΑΝ, ΗΣΑΣΤΕ, ΗΣΟΥΝ, ΗΤΑΝ, ΗΤΑΝΕ, ΗΤΟΙ, ΗΤΤΟΝ, ΘΑ, Ι, ΙΔΙΑ, ΙΔΙΑΝ, ΙΔΙΑΣ, ΙΔΙΕΣ, ΙΔΙΟ, ΙΔΙΟΙ, ΙΔΙΟΝ, ΙΔΙΟΣ, ΙΔΙΟΥ, ΙΔΙΟΥΣ, ΙΔΙΩΝ, ΙΔΙΩΣ, ΙΙ, ΙΙΙ, ΙΣΑΜΕ, ΙΣΙΑ, ΙΣΩΣ, ΚΑΘΕ, ΚΑΘΕΜΙΑ, ΚΑΘΕΜΙΑΣ, ΚΑΘΕΝΑ, ΚΑΘΕΝΑΣ, ΚΑΘΕΝΟΣ, ΚΑΘΕΤΙ, ΚΑΘΟΛΟΥ, ΚΑΘΩΣ, ΚΑΙ, ΚΑΚΑ, ΚΑΚΩΣ, ΚΑΛΑ, ΚΑΛΩΣ, ΚΑΜΙΑ, ΚΑΜΙΑΝ, ΚΑΜΙΑΣ, ΚΑΜΠΟΣΑ, ΚΑΜΠΟΣΕΣ, ΚΑΜΠΟΣΗ, ΚΑΜΠΟΣΗΝ, ΚΑΜΠΟΣΗΣ, ΚΑΜΠΟΣΟ, ΚΑΜΠΟΣΟΙ, ΚΑΜΠΟΣΟΝ, ΚΑΜΠΟΣΟΣ, ΚΑΜΠΟΣΟΥ, ΚΑΜΠΟΣΟΥΣ, ΚΑΜΠΟΣΩΝ, ΚΑΝΕΙΣ, ΚΑΝΕΝ, ΚΑΝΕΝΑ, ΚΑΝΕΝΑΝ, ΚΑΝΕΝΑΣ, ΚΑΝΕΝΟΣ, ΚΑΠΟΙΑ, ΚΑΠΟΙΑΝ, ΚΑΠΟΙΑΣ, ΚΑΠΟΙΕΣ, ΚΑΠΟΙΟ, ΚΑΠΟΙΟΙ, ΚΑΠΟΙΟΝ, ΚΑΠΟΙΟΣ, ΚΑΠΟΙΟΥ, ΚΑΠΟΙΟΥΣ, ΚΑΠΟΙΩΝ, ΚΑΠΟΤΕ, ΚΑΠΟΥ, ΚΑΠΩΣ, ΚΑΤ, ΚΑΤΑ, ΚΑΤΙ, ΚΑΤΙΤΙ, ΚΑΤΟΠΙΝ, ΚΑΤΩ, ΚΙΟΛΑΣ, ΚΛΠ, ΚΟΝΤΑ, ΚΤΛ, ΚΥΡΙΩΣ, ΛΙΓΑΚΙ, ΛΙΓΟ, ΛΙΓΩΤΕΡΟ, ΛΟΓΩ, ΛΟΙΠΑ, ΛΟΙΠΟΝ, ΜΑ, ΜΑΖΙ, ΜΑΚΑΡΙ, ΜΑΚΡΥΑ, ΜΑΛΙΣΤΑ, ΜΑΛΛΟΝ, ΜΑΣ, ΜΕ, ΜΕΘΑΥΡΙΟ, ΜΕΙΟΝ, ΜΕΛΕΙ, ΜΕΛΛΕΤΑΙ, ΜΕΜΙΑΣ, ΜΕΝ, ΜΕΡΙΚΑ, ΜΕΡΙΚΕΣ, ΜΕΡΙΚΟΙ, ΜΕΡΙΚΟΥΣ, ΜΕΡΙΚΩΝ, ΜΕΣΑ, ΜΕΤ, ΜΕΤΑ, ΜΕΤΑΞΥ, ΜΕΧΡΙ, ΜΗ, ΜΗΔΕ, ΜΗΝ, ΜΗΠΩΣ, ΜΗΤΕ, ΜΙΑ, ΜΙΑΝ, ΜΙΑΣ, ΜΟΛΙΣ, ΜΟΛΟΝΟΤΙ, ΜΟΝΑΧΑ, ΜΟΝΕΣ, ΜΟΝΗ, ΜΟΝΗΝ, ΜΟΝΗΣ, ΜΟΝΟ, ΜΟΝΟΙ, ΜΟΝΟΜΙΑΣ, ΜΟΝΟΣ, ΜΟΝΟΥ, ΜΟΝΟΥΣ, ΜΟΝΩΝ, ΜΟΥ, ΜΠΟΡΕΙ, ΜΠΟΡΟΥΝ, ΜΠΡΑΒΟ, ΜΠΡΟΣ, ΝΑ, ΝΑΙ, ΝΩΡΙΣ, ΞΑΝΑ, ΞΑΦΝΙΚΑ, Ο, ΟΙ, ΟΛΑ, ΟΛΕΣ, ΟΛΗ, ΟΛΗΝ, ΟΛΗΣ, ΟΛΟ, ΟΛΟΓΥΡΑ, ΟΛΟΙ, ΟΛΟΝ, ΟΛΟΝΕΝ, ΟΛΟΣ, ΟΛΟΤΕΛΑ, ΟΛΟΥ, ΟΛΟΥΣ, ΟΛΩΝ, ΟΛΩΣ, ΟΛΩΣΔΙΟΛΟΥ, ΟΜΩΣ, ΟΠΟΙΑ, ΟΠΟΙΑΔΗΠΟΤΕ, ΟΠΟΙΑΝ, ΟΠΟΙΑΝΔΗΠΟΤΕ, ΟΠΟΙΑΣ, ΟΠΟΙΑΣΔΗΠΟΤΕ, ΟΠΟΙΔΗΠΟΤΕ, ΟΠΟΙΕΣ, ΟΠΟΙΕΣΔΗΠΟΤΕ, ΟΠΟΙΟ, ΟΠΟΙΟΔΗΠΟΤΕ, ΟΠΟΙΟΙ, ΟΠΟΙΟΝ, ΟΠΟΙΟΝΔΗΠΟΤΕ, ΟΠΟΙΟΣ, ΟΠΟΙΟΣΔΗΠΟΤΕ, ΟΠΟΙΟΥ, ΟΠΟΙΟΥΔΗΠΟΤΕ, ΟΠΟΙΟΥΣ, ΟΠΟΙΟΥΣΔΗΠΟΤΕ, ΟΠΟΙΩΝ, ΟΠΟΙΩΝΔΗΠΟΤΕ, ΟΠΟΤΕ, ΟΠΟΤΕΔΗΠΟΤΕ, ΟΠΟΥ, ΟΠΟΥΔΗΠΟΤΕ, ΟΠΩΣ, ΟΡΙΣΜΕΝΑ, ΟΡΙΣΜΕΝΕΣ, ΟΡΙΣΜΕΝΩΝ, ΟΡΙΣΜΕΝΩΣ, ΟΣΑ, ΟΣΑΔΗΠΟΤΕ, ΟΣΕΣ, ΟΣΕΣΔΗΠΟΤΕ, ΟΣΗ, ΟΣΗΔΗΠΟΤΕ, ΟΣΗΝ, ΟΣΗΝΔΗΠΟΤΕ, ΟΣΗΣ, ΟΣΗΣΔΗΠΟΤΕ, ΟΣΟ, ΟΣΟΔΗΠΟΤΕ, ΟΣΟΙ, ΟΣΟΙΔΗΠΟΤΕ, ΟΣΟΝ, ΟΣΟΝΔΗΠΟΤΕ, ΟΣΟΣ, ΟΣΟΣΔΗΠΟΤΕ, ΟΣΟΥ, ΟΣΟΥΔΗΠΟΤΕ, ΟΣΟΥΣ, ΟΣΟΥΣΔΗΠΟΤΕ, ΟΣΩΝ, ΟΣΩΝΔΗΠΟΤΕ, ΟΤΑΝ, ΟΤΙ, ΟΤΙΔΗΠΟΤΕ, ΟΤΟΥ, ΟΥ, ΟΥΔΕ, ΟΥΤΕ, ΟΧΙ, ΠΑΛΙ, ΠΑΝΤΟΤΕ, ΠΑΝΤΟΥ, ΠΑΝΤΩΣ, ΠΑΡΑ, ΠΕΡΑ, ΠΕΡΙ, ΠΕΡΙΠΟΥ, ΠΕΡΙΣΣΟΤΕΡΟ, ΠΕΡΣΙ, ΠΕΡΥΣΙ, ΠΙΑ, ΠΙΘΑΝΟΝ, ΠΙΟ, ΠΙΣΩ, ΠΛΑΙ, ΠΛΕΟΝ, ΠΛΗΝ, ΠΟΙΑ, ΠΟΙΑΝ, ΠΟΙΑΣ, ΠΟΙΕΣ, ΠΟΙΟ, ΠΟΙΟΙ, ΠΟΙΟΝ, ΠΟΙΟΣ, ΠΟΙΟΥ, ΠΟΙΟΥΣ, ΠΟΙΩΝ, ΠΟΛΥ, ΠΟΣΕΣ, ΠΟΣΗ, ΠΟΣΗΝ, ΠΟΣΗΣ, ΠΟΣΟΙ, ΠΟΣΟΣ, ΠΟΣΟΥΣ, ΠΟΤΕ, ΠΟΥ, ΠΟΥΘΕ, ΠΟΥΘΕΝΑ, ΠΡΕΠΕΙ, ΠΡΙΝ, ΠΡΟ, ΠΡΟΚΕΙΜΕΝΟΥ, ΠΡΟΚΕΙΤΑΙ, ΠΡΟΠΕΡΣΙ, ΠΡΟΣ, ΠΡΟΤΟΥ, ΠΡΟΧΘΕΣ, ΠΡΟΧΤΕΣ, ΠΡΩΤΥΤΕΡΑ, ΠΩΣ, ΣΑΝ, ΣΑΣ, ΣΕ, ΣΕΙΣ, ΣΗΜΕΡΑ, ΣΙΓΑ, ΣΟΥ, ΣΤΑ, ΣΤΗ, ΣΤΗΝ, ΣΤΗΣ, ΣΤΙΣ, ΣΤΟ, ΣΤΟΝ, ΣΤΟΥ, ΣΤΟΥΣ, ΣΤΩΝ, ΣΥΓΧΡΟΝΩΣ, ΣΥΝ, ΣΥΝΑΜΑ, ΣΥΝΕΠΩΣ, ΣΥΝΗΘΩΣ, ΣΥΧΝΑ, ΣΥΧΝΑΣ, ΣΥΧΝΕΣ, ΣΥΧΝΗ, ΣΥΧΝΗΝ, ΣΥΧΝΗΣ, ΣΥΧΝΟ, ΣΥΧΝΟΙ, ΣΥΧΝΟΝ, ΣΥΧΝΟΣ, ΣΥΧΝΟΥ, ΣΥΧΝΟΥ, ΣΥΧΝΟΥΣ, ΣΥΧΝΩΝ, ΣΥΧΝΩΣ, ΣΧΕΔΟΝ, ΣΩΣΤΑ, ΤΑ, ΤΑΔΕ, ΤΑΥΤΑ, ΤΑΥΤΕΣ, ΤΑΥΤΗ, ΤΑΥΤΗΝ, ΤΑΥΤΗΣ, ΤΑΥΤΟ, ΤΑΥΤΟΝ, ΤΑΥΤΟΣ, ΤΑΥΤΟΥ, ΤΑΥΤΩΝ, ΤΑΧΑ, ΤΑΧΑΤΕ, ΤΕΛΙΚΑ, ΤΕΛΙΚΩΣ, ΤΕΣ, ΤΕΤΟΙΑ, ΤΕΤΟΙΑΝ, ΤΕΤΟΙΑΣ, ΤΕΤΟΙΕΣ, ΤΕΤΟΙΟ, ΤΕΤΟΙΟΙ, ΤΕΤΟΙΟΝ, ΤΕΤΟΙΟΣ, ΤΕΤΟΙΟΥ, ΤΕΤΟΙΟΥΣ, ΤΕΤΟΙΩΝ, ΤΗ, ΤΗΝ, ΤΗΣ, ΤΙ, ΤΙΠΟΤΑ, ΤΙΠΟΤΕ, ΤΙΣ, ΤΟ, ΤΟΙ, ΤΟΝ, ΤΟΣ, ΤΟΣΑ, ΤΟΣΕΣ, ΤΟΣΗ, ΤΟΣΗΝ, ΤΟΣΗΣ, ΤΟΣΟ, ΤΟΣΟΙ, ΤΟΣΟΝ, ΤΟΣΟΣ, ΤΟΣΟΥ, ΤΟΣΟΥΣ, ΤΟΣΩΝ, ΤΟΤΕ, ΤΟΥ, ΤΟΥΛΑΧΙΣΤΟ, ΤΟΥΛΑΧΙΣΤΟΝ, ΤΟΥΣ, ΤΟΥΤΑ, ΤΟΥΤΕΣ, ΤΟΥΤΗ, ΤΟΥΤΗΝ, ΤΟΥΤΗΣ, ΤΟΥΤΟ, ΤΟΥΤΟΙ, ΤΟΥΤΟΙΣ, ΤΟΥΤΟΝ, ΤΟΥΤΟΣ, ΤΟΥΤΟΥ, ΤΟΥΤΟΥΣ, ΤΟΥΤΩΝ, ΤΥΧΟΝ, ΤΩΝ, ΤΩΡΑ, ΥΠ, ΥΠΕΡ, ΥΠΟ, ΥΠΟΨΗ, ΥΠΟΨΙΝ, ΥΣΤΕΡΑ, ΦΕΤΟΣ, ΧΑΜΗΛΑ, ΧΘΕΣ, ΧΤΕΣ, ΧΩΡΙΣ, ΧΩΡΙΣΤΑ, ΨΗΛΑ, Ω, ΩΡΑΙΑ, ΩΣ, ΩΣΑΝ, ΩΣΟΤΟΥ, ΩΣΠΟΥ, ΩΣΤΕ, ΩΣΤΟΣΟ, ΩΧ

EN

EURODICAUTOM: a, about, an, and, at, by, for, from, in, of, on, the, to, with

CELEX: ABOUT, ABOVE, ABOVEMENTIONED, ACCORDANCE, ACCORDINGLY, ACTUALLY, AFOREMENTIONED, AFORESAID, AFTER, AFTERWARDS, AGAIN, AGAINST, AGO, AHEAD, ALBEIT, ALIKE, ALL, ALMOST, ALONG, ALONGSIDE, ALREADY, ALSO, ALTHOUGH, ALTOGETHER, ALWAYS, AM, AMONG, AN, AND, AND/OR, ANOTHER, ANY, ANYBODY, ANYMORE, ANYONE, ANYTHING, ANYWAY, ANYWHERE, APART, ARE, AROUND, AS, ASIDE, AT, AWAY, BE, BECAUSE, BEEN, BEFORE, BEFOREHAND, BEHALF, BELOW, BESIDE, BESIDES, BETWEEN, BEYOND, BOTH, BUT, BY, CANNOT, CERTAINLY, CONCERNED, CONCERNING, CONT, COULD, DEPENDING, DID, DO, DOES, DOING, DONE, DULY, DURING, EACH, EITHER, ELSE, ELSEWHERE, ENOUGH, ENSURE, ESPECIALLY, EVER, EVERY, EVERYONE, EVERYTHING, EVERYWHERE, EXCEPT, FAR, FEW, FOR, FROM, GET, GOT, HAD, HARDLY, HAS, HAVE, HAVING, HE, HENCE, HENCEFORTH, HENCEFORWARD, HER, HERE, HEREAFTER, HEREBY, HEREIN, HEREINAFTER, HEREINBEFORE, HEREOF, HEREON, HERETO, HEREUNDER, HEREWITH, HERS, HERSELF, HIM, HIMSELF, HIS, HITHERTO, HOW, HOWEVER, HOWSOEVER, IF, IN, INASMUCH, INDEED, INSIDE, INSOFAR, INTO, IS, IT, ITS, ITSELF, LASTLY, LATTER, LESS, MANY, MAYBE, ME, MORE, MOREOVER, MOST, MR, MY, MYSELF, NAMELY, NEITHER, NEVER, NEVERTHELESS, NONE, NONETHELESS, NOR, NOT, NOW, OF, OFF, OFTEN, ON, ONES, ONLY, ONTO, OR, OTHER, OUR, OURS, OVER, PROVIDED, PROVIDES, RELATING, SAID, SAME, SEEM, SEEN, SHALL, SHE, SHOULD, SO, SOME, SOMEHOW, SOMEONE, SOMETHING, SOMETIMES, SOMEWHAT, SOON, SUCH, THAN, THAT, THE, THEIR, THEIRS, THEM, THEMSELVES, THEN, THERE, THEREAFTER, THEREAGAINST, THEREBY, THEREFOR, THEREFORE, THEREFROM, THEREIN, THEREOF, THEREON, THERETO, THEREUNDER, THEREWITH, THESE, THEY, THIS, THOSE, THUS, TO, TOO, UNDER, UNTIL, UNTO, UP, UPON, VERY, WAS, WE, WERE, WHAT, WHEN, WHENEVER, WHERE, WHEREABOUTS, WHEREAS, WHEREBY, WHEREIN, WHEREOF, WHEREUPON, WHEREVER, WHETHER, WHICH, WHICHEVER, WHILE, WHILST, WHOEVER, WHOM, WHOMSOEVER, WHOSE, WHY, WITH, WITHIN, WITHOUT, WOULD, YET, YOU, YOUR, YOURS

ES

EURODICAUTOM: al, con, de, del, el, en, la, las, los, para, por, sobre, un, una

CELEX: ACASO, ACERCA, ADEMAS, AHI, AHORA, AL, ALGO, ALGUIEN, ALGUN, ALGUNA, ALGUNAS, ALGUNO, ALGUNOS, ALLA, ALLI, ANTES, APENAS, AQUEL, AQUELLA, AQUELLAS, AQUELLOS, AQUI, ASI, ASIMISMO, ATRAS, AUN, AUNQUE, BASTANTE, BASTANTES, CADA, CIERTA, CIERTAS, CIERTO, CIERTOS, COMO, CON, CONCERNIENTE, CONSIGO, CONTRA, CUAL, CUALES, CUALESQUIER, CUALESQUIERA, CUALQUIER, CUALQUIERA, CUANDO, CUANTA, CUANTAS, CUANTO, CUANTOS, CUYA, CUYAS, CUYO, CUYOS, DE, DEBAJO, DELANTE, DEMAS, DEMASIADA, DEMASIADAS, DEMASIADO, DEMASIADOS, DENTRO, DESDE, DESPUES, DETERMINADOS, DETRAS, DICHA, DICHO, DIVERSA, DIVERSAS, DIVERSO, DIVERSOS, DONDE, DURANTE, EL, ELLA, ELLAS, ELLO, ELLOS, EN, ENTONCES, ENTRE, ERA, ERAN, ES, ESA, ESAS, ESE, ESOS, ESTA, ESTABA, ESTABAN, ESTAN, ESTANDO, ESTANDOLA, ESTANDOLE, ESTANDOLES, ESTANDOLO, ESTAR, ESTARA, ESTARAN, ESTARIA, ESTARIAN, ESTARSE, ESTAS, ESTE, ESTEN, ESTO, ESTOS, ESTUVIERA, ESTUVIERAN, ESTUVIERE, ESTUVIEREN, ESTUVIERON, ESTUVIESE, ESTUVIESEN, ETC, ETCETERA, EXCELENTISIMO, EXCELENTISIMOS, EXCEPTO, EXCMO, EXCMOS, EXISTE, FUE, FUERA, FUERAN, FUERE, FUEREN, FUERON, FUESE, FUESEN, HA, HABER, HABERLA, HABERLAS, HABERLE, HABERLO, HABERLOS, HABERSE, HABERSELES, HABIA, HABIAN, HABIDA, HABIDO, HABIENDO, HABIENDOSE, HABIENDOSELES, HABRA, HABRAN, HABRIA, HABRIAN, HACE, HACEN, HACER, HACIA, HAGA, HAGAN, HAN, HARA, HARAN, HASTA, HAY, HAYA, HAYAN, HE, HECHA, HECHO, HEMOS, HICIERA, HICIERAN, HICIEREN, HICIERON, HUBIERA, HUBIERAN, HUBIERE, HUBIEREN, HUBIESE, HUBIESEN, HUBO, ILMA, ILMAS, ILMO, ILMOS, ILUSTRISIMA, ILUSTRISIMAS, ILUSTRISIMO, ILUSTRISIMOS, JAMAS, LA, LAS, LE, LES, LO, LOS, LUEGO, MAS, MAYOR, ME, MEDIANTE, MENOR, MENOS, MI, MIA, MIAS, MIENTRAS, MIO, MIOS, MIS, MISMA, MISMAS, MISMO, MISMOS, MUCHA, MUCHAS, MUCHO, MUCHOS, MUY, NADA, NADIE, NI, NINGUN, NINGUNA, NINGUNAS, NINGUNO, NINGUNOS, NO, NOS, NOSOTRAS, NOSOTROS, NUESTRA, NUESTRAS, NUESTRO, NUESTROS, NUNCA, OBSTANTE, ORA, OS, OTRA, OTRAS, OTRO, OTROS, PARA, PERO, POCA, POCAS, POCO, POCOS, PODRA, PODRAN, PODRIA, PODRIAN, POR, PORQUE, PUDIENDO, PUDIENDOSE, PUDIERA, PUDIERAN, PUDIERE, PUDIEREN, PUDIERON, PUDIESE, PUDIESEN, PUDO, PUEDA, PUEDAN, PUEDE, PUEDEN, PUES, PUESTO, QUE, QUEDA, QUEDAN, QUEDAR, QUEDARA, QUEDARAN, QUEDARIA, QUEDARIAN, QUEDARON, QUEDASE, QUEDASEN, QUEDESE, QUEDESEN, QUEDO, QUIEN, QUIENES, QUIZA, QUIZAS, RELATIVA, RELATIVAS, RELATIVO, RELATIVOS, RESPECTO, SE, SEA, SEAN, SEGUN, SEMEJANTE, SEMEJANTES, SER, SERA, SERAN, SERIAN, SERLE, SERLES, SERLO, SI, SIENDO, SIENDOLES, SIENDOLO, SIGUIENTE, SIGUIENTES, SIN, SINO, SIQUIERA, SOBRE, SOLA, SOLAMENTE, SOLAS, SOLO, SOLOS, SON, SR, SU, SUS, SUYA, SUYAS, SUYO, SUYOS, TAL, TALES, TAMBIEN, TAMPOCO, TAN, TANTA, TANTAS, TANTO, TANTOS, TENDRIA, TENDRIAN, TENEMOS, TENER, TENERLA, TENERLAS, TENERLO, TENERLOS, TENERSE, TENGA, TENGO, TENIA, TENIAN, TENIDO, TENIENDO, TENIENDOSE, TIENE, TIENEN, TODA, TODAS, TODO, TODOS, TRAS, TRAVES, TU, TUVIERA, TUVIERAN, TUVIERE, TUVIEREN, TUVIERON, TUVIESE, TUVIESEN, TUYA, TUYAS, TUYO, TUYOS, U, UN, UNA, UNO, UNOS, VARIAS, VARIOS, VOSOTRAS, VOSOTROS, VUESTRA, VUESTRAS, VUESTRO, VUESTROS, YA, YO

FI

EURODICAUTOM: ja, kanssa

CELEX: -

FR

EURODICAUTOM: a, au, aux, avec, chez, dans, de, des, du, en, et, la, le, les, ou, par, pour, sur, un

CELEX: AFIN, AINSI, APRES, AU, AURAIT, AUTRE, AUTRES, AUX, AVANT, AVEC, CE, CELLE, CELLES, CELUI, CET, CETTE, CEUX, CHAQUE, COMME, CONCERNANT, CONCERNE, DANS, DE, DEJA, DES, DEVANT, DOIT, DOIVENT, DONC, DONT, DU, DUDIT, EGARD, ELLE, ELLES, EN, ET, ETAIT, ETRE, IL, ILS, JUSQU, LA, LE, LES, LEUR, LEURS, LORS, LORSQUE, LUI, MEME, MME, NE, NON, NOTAMMENT, ONT, OU, OUI, PAR, PAS, PEUT, PEUVENT, POUR, QU, QUE, QUI, SA, SANS, SAUF, SAURAIT, SE, SELON, SERAIT, SES, SI, SOIT, SONT, SOUS, SUR, TANT, TEL, TELLE, TOUS, TOUT, TOUTE, TOUTEFOIS, TOUTES, UN, UNE, VU

IT

EURODICAUTOM: ad, ai, al, all, alla, alle, allo, cogli, coi, col, con, da, dagli, dai, dal, dall, dalla, dalle, dallo, degli, dei, del, dell, della, delle, dello, di, ed, gli, il, in, la, le, lo, negli, nei, nel, nell, nella, nelle, nello, per, su, sugli, sui, sul, sulla, sulle, sullo, un, una, uno

CELEX: ABBIA, ABBIANO, AD, ADDIETRO, AFFINCHE, AGLI, AI, AL, ALCUN, ALCUNA, ALCUNE, ALCUNI, ALCUNO, ALL, ALLA, ALLE, ALLO, ALLORA, ALMENO, ALQUANTO, ALTRA, ALTRE, ALTRESI, ALTRETTANTO, ALTRI, ALTRIMENTI, ALTRO, ALTRONDE, ALTROVE, ANALOGA, ANALOGHE, ANALOGHI, ANALOGO, ANCHE, ANCORA, ANCORCHE, ANZI, ANZICHE, ANZIDETTA, ANZIDETTE, ANZIDETTI, ANZIDETTO, ANZITUTTO, APPENA, APPLICABILE, APPLICABILI, APPRESSO, ATTRAVERSO, ATTUA, ATTUANO, ATTUARE, AVENDO, AVERE, AVESSE, AVESSERO, AVEVA, AVEVANO, AVRA, AVRANNO, AVREBBE, AVREBBERO, AVUTA, AVUTE, AVUTI, AVUTO, BEN, BENCHE, BENE, BENINTESO, CHE, CHI, CHISSA, CHIUNQUE, CI, CIASCUN, CIASCUNA, CIASCUNO, CIOE, CIRCA, CITATA, CITATE, CITATI, CITATO, COI, COL, COLEI, COLORO, COLUI, COME, COMUNQUE, CON, CONCERNENTE, CONCERNENTI, CONSIDERANDO, CONTRA, CONTRO, COSI, COSIDDETTA, COSIDDETTE, COSIDDETTI, COSIDDETTO, COSIDETTA, COSIDETTE, COSIDETTI, COSIDETTO, CUI, DA, DAGLI, DAI, DAL, DALL, DALLA, DALLE, DALLO, DAPPERTUTTO, DAVANTI, DAVVERO, DEBBONO, DEGLI, DEI, DELL, DELLA, DELLE, DELLO, DENOMINATA, DENOMINATE, DENOMINATI, DENOMINATO, DERIVANTE, DERIVANTI, DETTA, DETTE, DETTI, DETTO, DEVE, DEVONO, DI, DIETRO, DINANZI, DINNANZI, DINTORNO, DONDE, DOPO, DOVE, DOVENDO, DOVERE, DOVERONO, DOVESSE, DOVESSERO, DOVETTE, DOVETTERO, DOVEVA, DOVEVANO, DOVRA, DOVRANNO, DOVREBBE, DOVREBBERO, DOVUNQUE, DUNQUE, DURANTE, EBBE, EBBENE, EBBERO, ECC, ECCETERA, ED, EGLI, ELLA, ELLE, ENTRO, ERANO, ESSA, ESSE, ESSENDO, ESSERE, ESSI, ETC, ETCETERA, FA, FACCIA, FACCIANO, FACENDO, FACESSE, FACESSERO, FACEVA, FACEVANO, FANNO, FARA, FARANNO, FARE, FAREBBE, FAREBBERO, FATTA, FATTE, FECE, FECERO, FIN, FINCHE, FINO, FINORA, FINTANTO, FINTANTOCHE, FORSE, FOSSE, FOSSERO, FRA, FRATTANTO, FRATTEMPO, FU, FURONO, GIA, GIU, GLI, GU, HA, HANNO, IERI, IL, IN, INANZI, INDICATA, INDICATE, INDICATI, INDICATO, INDIETRO, INFATTI, INFRA, INGIU, INNANZI, INNANZITUTTO, INOLTRE, INSOMMA, INSU, INTANTO, INTORNO, INVANO, INVECE, IO, IVI, LA, LADDOVE, LAGGIU, LASSU, LE, LEI, LI, LO, LORO, LUI, MA, MAI, MAL, MALE, MANIERA, MANIERE, ME, MEDESIMA, MEDESIME, MEDESIMI, MEDESIMO, MEGLIO, MEMBRI, MEMBRO, MENO, MENZIONATA, MENZIONATE, MENZIONATI, MENZIONATO, MI, MIA, MIE, MIEI, MIO, MODI, MODO, MOLTA, MOLTE, MOLTI, MOLTO, MOMENTI, MOMENTO, NE, NEANCHE, NEGLI, NEI, NEL, NELL, NELLA, NELLE, NELLO, NEMMENO, NEPPURE, NESSUN, NESSUNA, NESSUNO, NIENTE, NOI, NON, NONCHE, NONDIMENO, NONOSTANTE, NOSTRA, NOSTRE, NOSTRI, NOSTRO, OCCORRE, OD, OGGI, OGNI, OGNIQUALVOLTA, OLTRE, OLTRETUTTO, ON, OPPURE, ORAMAI, ORMAI, OSSIA, OVE, OVVERO, OVVEROSIA, OVVEROSSIA, PAG, PAGG, PARTICOLARE, PEGGIO, PEI, PEL, PER, PERCHE, PERCIO, PERDIPIU, PERLOMENO, PERLOPIU, PERO, PERTANTO, PIU, PIUTTOSTO, POCO, POI, POICHE, POSSA, POSSANO, POSSONO, POTESSE, POTESSERO, POTEVA, POTEVANO, POTRA, POTRANNO, POTREBBE, POTREBBERO, POTUTA, POTUTE, POTUTI, POTUTO, PRECISATA, PRECISATE, PRECISATI, PRECISATO, PRECITATA, PRECITATE, PRECITATI, PRECITATO, PREDETTA, PREDETTE, PREDETTI, PREDETTO, PRESENTE, PRESSAPPOCO, PRESSO, PRESSOCHE, PRESTO, PREVISTA, PREVISTE, PREVISTI, PREVISTO, PRIMA, PRO, PUO, PURCHE, QUA, QUAL, QUALCHEDUNA, QUALCHEDUNO, QUALCUNA, QUALCUNO, QUALE, QUALI, QUALORA, QUALSIASI, QUALSIVOGLIA, QUALUNQUE, QUALVOLTA, QUANDO, QUANTO, QUANTOMENO, QUANTUNQUE, QUEGLI, QUEL, QUELL, QUELLA, QUELLE, QUELLI, QUELLO, QUESTA, QUESTE, QUESTI, QUESTO, QUI, QUINDI, RECANTE, RECANTI, RELATIVA, RELATIVE, RELATIVI, RELATIVO, RICHIAMATA, RICHIAMATE, RICHIAMATI, RICHIAMATO, RICORDATA, RICORDATE, RICORDATI, RICORDATO, RIFERITA, RIFERITE, RIFERITI, RIFERITO, RIGUARDANTE, RIGUARDANTI, SARA, SARANNO, SAREBBE, SAREBBERO, SE, SEBBENE, SEGUENTE, SEGUENTI, SEMMAI, SEMPRE, SENZA, SEPPURE, SI, SIA, SIANO, SICCHE, SINCHE, SINORA, SOLO, SOLTANTO, SONO, SOPRA, SOPRACITATA, SOPRACITATE, SOPRACITATI, SOPRACITATO, SOPRATTUTTO, SOTTO, SOTTOCITATA, SOTTOCITATE, SOTTOCITATI, SOTTOCITATO, SOTTOELENCATA, SOTTOELENCATE, SOTTOELENCATI, SOTTOELENCATO, SPECIFICATA, SPECIFICATE, SPECIFICATI, SPECIFICATO, STATA, STATE, STESSA, STESSE, STESSI, STESSO, SU, SUA, SUBITO, SUCCESSIVA, SUCCESSIVE, SUCCESSIVI, SUCCESSIVO, SUCCITATA, SUCCITATE, SUCCITATI, SUCCITATO, SUDDETTA, SUDDETTE, SUDDETTI, SUDDETTO, SUE, SUGLI, SUI, SUINDICATA, SUINDICATE, SUINDICATI, SUINDICATO,

SUL, SULL, SULLA, SULLE, SULLO, SUMMENZIONATA, SUMMENZIONATE, SUMMENZIONATI, SUMMENZIONATO, SUO, SUOI, SUPRA, SURRICORDATA, SURRICORDATE, SURRICORDATI, SURRICORDATO, SURRIFERITA, SURRIFERITE, SURRIFERITI, SURRIFERITO, SUSSEGUENTE, SUSSEGUENTI, TAL, TALCHE, TALE, TALI, TALUN, TALUNA, TALUNE, TALUNI, TALUNO, TANTO, TARDI, TI, TRA, TRANNE, TUA, TUE, TUO, TUOI, TUTT, TUTTALPIU, TUTTAVIA, TUTTO, TUTTORA, UN, UNA, UNO, VA, VANNO, VE, VENENDO, VENGA, VENGANO, VENGONO, VENIRE, VENISSE, VENISSERO, VENIVA, VENIVANO, VENNE, VENNERO, VERRA, VERRANNO, VI, VIENE, VIGORE, VISTA, VISTE, VISTI, VISTO, VOI, VOSTRA, VOSTRE, VOSTRI

NL

EURODICAUTOM: aan, bij, de, door, een, en, het, in, met, om, op, te, tot, uit, van, voor

CELEX: AAN, ACHTER, AF, AL, ALDAAR, ALHOEWEL, ALLE, ALLEN, ALOM, ALREEDS, ALS, ALSDAN, ALSMEDE, ALSNOG, ALSOF, ALSOOK, ALTHANS, ALTIJD, ALVORENS, ALWAAR, ANDER, ANDERE, ANDEREN, ANDERS, ANDERZIJDS, BEHALVE, BEHOUDENS, BIJ, BIJGEVOLG, BIJNA, DAAR, DAARAAN, DAARACHTER, DAARBIJ, DAARBOVEN, DAARBUITEN, DAARDOOR, DAARENBOVEN, DAARENTEGEN, DAARGELATEN, DAARIN, DAARMEDE, DAARMEE, DAARNA, DAARNAAR, DAARNAAST, DAAROM, DAARONDER, DAAROP, DAAROPVOLGEND, DAAROPVOLGENDE, DAAROVER, DAARTEGEN, DAARTEGENOVER, DAARTOE, DAARTUSSEN, DAARUIT, DAARVAN, DAARVANDAAN, DAARVOOR, DAN, DANKZIJ, DAT, DATGEEN, DATGENE, DE, DEGENE, DEGENEN, DER, DERHALVE, DESGEVRAAGD, DESGEWENST, DEWELKE, DEZE, DEZEN, DIE, DIEGENE, DIEGENEN, DIENS, DIKWIJLS, DIT, DOCH, DOEN, DOET, DOOR, DUS, ECHTER, EEN, EENS, ELK, ELKE, ELKEEN, EN, ENE, ENKELE, ENKELEN, ER, ERAAN, ERBIJ, ERIN, EROP, ERTOE, ERUIT, ERVAN, ERVOOR, EVENALS, EVENEENS, EVENMIN, EVENWEL, EVENZEER, EVENZO, GEEN, GEENSZINS, GEHAD, GEWEEST, GEWORDEN, HAAR, HAD, HADDEN, HEB, HEBBEN, HEBT, HEEFT, HEM, HET, HETGEEN, HETWELK, HETWELKE, HETZELFDE, HETZIJ, HIER, HIERBIJ, HIERBOVEN, HIERDOOR, HIERIN, HIERMEDE, HIERMEE, HIERNA, HIERNAAR, HIERNAVOLGENDE, HIEROM, HIEROMTRENT, HIERONDER, HIEROP, HIEROVER, HIERTEGEN, HIERTOE, HIERUIT, HIERVAN, HIERVOOR, HIERZIJ, HIJ, HOE, HOEDANIG, HOEDANIGE, HOELANG, HOEVERRE, HOEWEL, HUN, IEDER, IEDERE, IEDEREEN, IEMAND, IETS, IETWAT, IK, IMMER, IMMERS, IN, INDERDAAD, INDIEN, INTUSSEN, INZAKE, JA, JAWEL, JE, JIJ, JOU, JOUW, JULLIE, KON, KONDEN, MAG, MEDE, MEE, MEEST, MEESTE, MEESTEN, MEN, MENIG, MENIGE, MET, MIJ, MITS, MITSDIEN, MOCHT, MOCHTEN, MOEST, MOESTEN, MOGEN, NAAR, NAARGELANG, NAARMATE, NAAST, NABIJ, NADAT, NADER, NADIEN, NAMELIJK, NAUWELIJKS, NB, NEDER, NEE, NEEN, NEER, NERGENS, NIEMAND, NIET, NIETS, NIMMER, NOCH, NOCHTANS, NOG, NOGAL, NOGMAALS, NOOIT, NR, NU, OF, OFSCHOON, OFTEWEL, OFWEL, OM, OMDAT, OMSTREEKS, OMTRENT, ONDANKS, ONDER, ONDERAAN, ONLANGS, ONZE, OOIT, OOK, OP, OPDAT, OPNIEUW, OVER, PER, REEDS, SAMEN, SEDERT, SINDS, SINDSDIEN, SLECHTS, SOMMIGE, SOMS, SPOEDIG, STEEDS, TE, TEGEN, TELKENS, TEN, TENGEVOLGE, TENSLOTTE, TENZIJ, TER, TERWIJL, TEVENS, TEZAMEN, THANS, TOCH, TOE, TOEN, TOT, TOTDAT, TROUWENS, TUSSEN, UIT, UW, VAN, VANAF, VANUIT, VANWAAR, VANWAARUIT, VANWEGE, VEEL, VELE, VELEN, VELERLEI, VER, VERDEROP, VOLGEND, VOLGENDE, VOLGENS, VOOR, VOORDAT, VOOROP, VOORTS, VORIG, VORIGE, WAAR, WAARAAN, WAARAF, WAARBENEDEN, WAARBIJ, WAARBINNEN, WAARDOOR, WAARHEEN, WAARIN, WAARLANGS, WAARMEDE, WAARMEE, WAARNA, WAARNAAR, WAAROM, WAAROMTRENT, WAARONDER, WAAROP, WAAROVER, WAARTEGEN, WAARTOE, WAARTUSSEN, WAARUIT, WAARVAN, WAARVOOR, WANNEER, WAT, WEDEROM, WEGENS, WEL, WELDRA, WELK, WELKE, WELKEN, WELNU, WERD, WERDEN, WIE, WIENS, WIJ, WILDE, WILLEN, WORD, WORDEN, WORDT, ZAL, ZE, ZEER, ZELF, ZELFDE, ZELFS, ZELVE, ZICH, ZIJ, ZIJN, ZODAT, ZODRA, ZOEVEN, ZOJUIST, ZONDER, ZOU, ZOUDEN, ZOUDT, ZOVEEL, ZOVER, ZOVERRE, ZOWAT, ZOWEL, ZOZEER, ZULK, ZULKE, ZULKS, ZULLEN

PT

EURODICAUTOM: ao, aos, as, com, da, das, de, do, dos, dum, dumr, em, na, nas, no, nos, num, numa, os, para, por, sobre, um, uma

CELEX: ALGO, ALGUEM, ALGUM, ALGUMA, ALGUMAS, ALGUNS, ANTE, AO, AOS, APOS, AQUELA, AQUELAS, AQUELE, AQUELES, AQUILO, AS, ATE, CADA, CERTA, CERTAS, CERTO, CERTOS, COM, COMIGO, COMO, CONNOSCO, CONTIGO, CONVOSCO, CUJA, CUJAS, CUJOS, DA, DADA, DADAS, DADO, DAS, DE, DELA, DELAS, DELE, DELES, DESDE, DESTA, DESTAS, DESTE, DESTES, DEVE, DEVEM, DEVERA, DIRECTAMENTE, DO, DOS, DUM, DUMA, DUNS, DURANTE, ELA, ELAS, ELE, ELES, EM, EMBORA, ENQUANTO, ENTRA, ENTRE, ESSA, ESSAS, ESSE, ESSES, ESTA, ESTAO, ESTAS, ESTE, ESTES, EU, EXISTE, FAZEM, FOI, ISTO, JA, LHA, LHAS, LHE, LHES, LHO, LHOS, MAS, ME, MESMA, MESMO, MEU, MEUS, MIM, MINHA, MINHAS, MUITA, MUITAS, MUITO, MUITOS, NA, NADA, NAS, NELA, NELAS, NELE, NELES, NEM, NENHUM, NENHUMA, NENHUMAS, NENHUNS, NINGUEM, NO, NOS, NOSSA, NOSSAS, NOSSO, NOSSOS, NUM, NUMA, NUMAS, NUNS, ONDE, OS, OU, OUTRA, OUTRAS, OUTRO, OUTROS, PARA, PELA, PELO, PERANTE, PODE, PODEM, POIS, POR, POUCA, POUCAS, POUCO, POUCOS, QUAISQUER, QUALQUER, QUANDO, QUANTO, QUANTOS, QUE, QUEM, SE, SEGUINTE, SEGUINTES, SEM, SERA, SEU, SEUS, SOB, SOBRE, SUA, SUAS, SUFICIENTEMENTE, TA, TAIS, TAL, TANTA, TANTAS, TANTO, TANTOS, TAS, TE, TEM, TENDO, TER, TERA, TEU, TEUS, TI, TO, TODA, TODAS, TODO, TODOS, TOS, TRAS, TU, TUAS, TUDO, UM, UMA, UMAS, UNS, VARIO, VARIOS, VOS, VOSSA, VOSSAS, VOSSO, VOSSOS

SV

EURODICAUTOM: att, av, de, den, det, en, ett, för, från, hos, med, och, om, på, till

CELEX: DOCK, DU, DÄRFÖR, DÄRMED, DÄROM, DÄRTILL, DÄRVID, ELLER, EMELLERTID, HÄRMED, HÄROM, HÄRTILL, HÄRVID, INTE, JAG, OCH, OCKSÅ, SKULLE, SÅLEDES, SÅVIDA

Bibliographie

Agrovoc, Multilingual agricultural thesaurus, World Agricultural Information Center, 1998.

Church, K. W., et Hanks, P., «Word association norms, mutual information and lexicography», *Computational Linguistics*, vol. 16, n° 1, mars 1990.

Croft, W., et Harper, D., «Using probabilistic models of information retrieval without relevance information», *Journal of Documentation,* n° 35, 1979, p. 285-295.

Dunning, T., «Accurate methods for the statistics of surprise and coincidence», *Computational Linguistics*, vol. 19, n° 1, 1993.

Ferber, R., «Automated Indexing with Thesaurus Descriptors: A Co-occurrence Based Approach to Multilingual Retrieval», dans Peters, C., et Thanos, C. (ed.), *Research and Advanced Technology for Digital Libraries*, 1st European Conf. (ECDL'97), Lecture Notes in Computer sciences, Springer, Berlin, 1997, p. 232-255.

Haller, J., Ripplinger, B., Maas, D., et Gastmeyer, M., «Automatische Indexierung von wirtschaftswissenschaftlichen Texten — Ein Experiment», Hamburgisches Welt-Wirtschafts-Archiv, Saarbrücken, 2001.

Hlava, N. M. K., et Hainebach, R., «Multilingual Machine Indexing», NIT'1996.

Jacquemin, C., Daille, B., Royaute, J., et Polanco, X., «In vitro evaluation of a program for machine-aided indexing», *Information Processing and Management*, vol. 38, Elsevier Science BV, Amsterdam, 2002, p. 765-792.

Jones, K. S., «A statistical interpretation of term specificity and its application in retrieval», *Journal of Documentation*, vol. 1, n° 28, 1972, p. 11-21, et n° 60, 2004, p. 493-502.

Jones, K. S., «IDF term weighting and IR research lessons», *Journal of Documentation,* n° 60, 2004, p. 521-523.

Kilgarriff, A., «Which words are particularly characteristic of a text? A survey of statistical approaches», Proceedings of the AISB Workshop on Language Engineering for Document Analysis and Recognition, Sussex, avril 1996, p. 33-40.

Montejo Ráez, A., «Towards conceptual indexing using automatic assignment of descriptors», Proceedings of the Workshop on Personalization Techniques in Electronic Publishing on the Web, 2nd International Conference on Adaptive Hypermedia and Adaptive Web, S. Mizaro & C. Tasso, Malaga, 2002.

NIST Special Publication SP 500-261, The Thirteenth Text Retrieval Conference Proceedings (TREC 2004).

Pouliquen, B., Delamarre, D., et Le Beux, P., «Indexation de textes médicaux par extraction de concepts, et ses utilisations», 6es journées internationales d'analyse statistique des données textuelles (JADT'2002), Saint-Malo, 2002, p. 617-628.

Pouliquen, B., Steinberger, R., et Ignat, C.,«Automatic Annotation of Multilingual Text Collections with a Conceptual Thesaurus», Ontologies and Information Extraction, Workshop at Eurolan'2003: The Semantic Web and Language Technology — Its Potential and Practicalities, Bucarest, 28 juillet-8 août 2003.

Robertson, S., «Understanding inverse document frequency: on theoretical arguments for IDF», *Journal of Documentation,* n° 60, 2004, p. 503-520.

Witten, I. H., et Frank, E., «Data mining: practical machine learning tools and techniques», 2nd edition, Morgan Kaufmann, San Francisco, 2005.

Remerciements

Nous remercions tout particulièrement M. Raybaut et M^me^ Bock, qui ont apporté de nombreuses améliorations au présent document et dont les conseils nous ont beaucoup aidé.

Nous tenons à remercier le D^r^ Bagola, qui nous a suggéré de lier les mathématiques et l'indexation. Il nous a apporté son soutien tout au long de ce projet et nous lui en sommes extrêmement reconnaissant.

M. Chapelant nous a apporté une aide essentielle en convertissant les données d'EUR-Lex en format CSV, et nous l'en remercions, ainsi que M^me^ Martignon, qui a effectué les extractions de la base de données.

Nous remercions également M^me^ Laaboudi, qui nous a envoyé les listes de descripteurs Eurovoc et qui nous a également permis de prendre connaissance des travaux du Joint Research Committee dans le domaine de l'indexation.

La configuration de la Machine Virtuelle Java nous empêchait d'exécuter notre programme d'indexation. Nous sommes très reconnaissant à M. Ramahatra de nous avoir permis de surmonter ce problème.

Nous remercions également M. Misselin, qui s'est occupé des aspects administratifs de ce projet, ainsi que M. Bresch, qui nous a fourni plusieurs publications utiles.

En dehors de l'Office des publications, M. Eibe Frank, de l'université de Waikato en Nouvelle-Zélande, nous a été d'un grand secours en nous expliquant comment convertir les données d'EUR-Lex en format .arff. M. Frank nous a également conseillé sur l'utilisation des algorithmes de Weka en ligne de commande, et nous lui en sommes particulièrement reconnaissant.

Nous remercions également M. Jean-Charles Lamirel et M. Laurent Bougrain, du Laboratoire lorrain de recherche en informatique et ses applications (France) pour leurs conseils sur la façon de convertir un texte en vecteur de termes.

M^me^ Cadot, du Laboratoire lorrain de recherche en informatique et ses applications (France) a accepté de lire ce document et nous la remercions pour ses commentaires très utiles, en particulier sur les aspects mathématiques.

Enfin, nous remercions M^me^ Rachel Toum, de la Banque européenne d'investissement, qui nous a envoyé de nombreuses références d'études sur l'indexation, et qui nous a permis de nous familiariser avec la recherche actuelle dans ce domaine.

Commission européenne

Modèles mathématiques pour l'indexation assistée de documents législatifs communautaires

Luxembourg: Office des publications officielles des Communautés européennes

2009 — 57 p. — 21 x 29,7 cm

ISBN 92-78-40433-0