REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR

ET DE LA RECHERCHE SCIENTIFIQUE

CENTRE UNIVERSITAIRE DE KHENCHELA

DIRECTION DE LA POST GRADUATION ET DE LA RECHERCHE SCIENTIFIQUE

Ecole Doctorale de Sciences et Technologies de l'Information

et de la Communication

N° d'ordre :

Série :

MEMOIRE

Présenté en vue de l'obtention du diplôme de :

Magistère en informatique

Option : Système d'information et de connaissances

Thème :

INDEXATION DE BASE DE DONNEES AUDIO
à l'aide de cartes auto-organisatrices (SOM)
basée sur une ontologie

Présenté par :

CHERGUI WAHID

Soutenu publiquement le …/…/2009

Devant le jury :

Président : Dr Youcef CHIBANI , *Maître de conférences,* Université des S.T. Houari Boumediene.

Rapporteur : Dr Halima BAHI, *Maître de conférence - Université de Annaba.*

Examinateur : Dr Nadir FARAH, *Maître de conférences - Université de Annaba.*

Examinateur : Dr Tarek KHADIR, *Maître de conférence - Université de Annaba.*

Merci mon Dieu de m'avoir donné
la force, la patience et la volonté d'arriver au terme de ce travail.

Je dédie ce modeste travail à mes enfants chéris :
Amira malak et Adem.

Remerciements

Je souhaite remercier très vivement mon encadreur Dr Halima Bahi, non parce que c'est l'usage, mais pour m'avoir guidé dans mes travaux, conseillé avec professionnalisme et une très grande expertise, sans jamais compter son temps. Ses apports majeurs me permettent aujourd'hui de vous présenter cette thèse.

Je suis très reconnaissant envers le Docteur CHIBANI de me faire l'honneur de présider le jury de cette soutenance. J'exprime toute ma gratitude au Docteur Farah et au Docteur KHADIR pour accepter d'être mes examinateurs de thèse.

Je tiens ensuite à exprimer ma gratitude à mon employeur, la SONELGAZ, et à travers elle son Directeur de distribution Khenchela ,Mr Kherchouche, ainsi que Monsieur Houha, Mr Bouchareb, Mr. BELMEKKI et leurs collaborateurs pour m'avoir soutenu et permis de réaliser cette thèse. Je remercie l'ensemble de mes collègues de travail, pour leur aide, collaboration et sympathique soutien.

Je remercie mes parents de m'avoir inculqué le goût du savoir et de l'ambition.

Je remercie ma femme pour ses encouragements permanents, son aide précieuse et son soutien sans faille.

Mes amitiés s'adressent à mes collègues de la promotion 2006 de l'Ecole Doctorale, bakhoche, brahim, Issam, hakim, Malek, Ali,…..

Merci à tous mes amis qui m'ont permis de me changer les idées quand j'en avais besoin, Saleh, Kamel, Ali, Azedine, Adel, Chafai.

RESUME

Pour regrouper les documents audio de sorte que les appels similaires appartiennent au même groupe. Une méthode qui peut résoudre le problème d'indexation audio doit comprendre les quatre éléments suivants: la méthode de transcription, la méthode de représentation des documents, l'algorithme de regroupement et la méthode de représentation des résultats.

A travers la méthode de transcription, les documents audio sont transcris par le système ASR (Automatic Speech Recognition) sous forme de documents texte dans différentes langues, et pour les indexer thématiquement il faut qu'on résolve le problème de multilinguisme**, pour cela la méthode de présentation doit présenter les documents transcrits sous forme de vecteur de concepts obtenus par la projection sur l'ontologie de domaine.** C'est la raison pour laquelle la méthode basée sur une ontologie a été choisie**,** où les instances des termes sont de type multi langue. La méthode de regroupement doit être suffisamment puissante pour pouvoir grouper un grand ensemble de documents et permettre à l'utilisateur de visualiser des résultats. Les cartes auto organisatrices « SOM » ont été choisies pour cette raison.

Mots Clés *:* indexation audio, ontologie domaine, mesure de similarité, Les cartes auto organisatrices.

Abstract

In order to bring together audio documents in a way similar calls belong to a same group, the appropriate method for indexing audio data must include the following four elements:

The transcription method
The method for representing documents
The groupement algorithm
The method for results display.

Through the transcription method, audio documents are recorded with ASR (Automatic Speech Recognition) in various languages. To allow their thematic indexation, we have to solve the problem of Multilanguage. The transcript documents must be considered as concept vectors achieved by a projection on a domain ontology. This is the reason why an ontology based method has been chosen, terms being of Multilanguage types. The gathering method must be powerful enough to allow the grouping of a great quantity of documents and give the user the possibility of displaying the results. Self Organizing Maps (SOM) are chosen for this purpose.

Key words : audio indexation, domain ontology, similarity measures, Self Organizing Maps

SOMMAIRE

Liste des Figures :

Liste des Tableaux :

Introduction

De nos jours, tous les secteurs d'activité foisonnent de données de toutes sortes et en particulier les documents audio. Ces données sont souvent mal structurées et difficiles d'accès. Conserver la maîtrise sur ces grands ensembles d'information devient de plus en plus complexe. Et il est devenu nécessaire de mettre au point de nouvelles formes de représentation, de description de contenu pour accéder à ces données.

Le traitement automatique de la parole est, par essence, un domaine de recherche pluridisciplinaire. Il utilise conjointement des notions empruntées au traitement de signal, à la linguistique (phonétique, phonologie, sémantique, pragmatique…), au traitement de l'information ou encore à l'algorithmique. Le module introduit les connaissances nécessaires à la mise en œuvre d'un système de reconnaissance de la parole.

L'indexation de documents audio pose des problèmes spécifiques. Citons les principaux : la longueur des fichiers de parole qui ne sont pas préalablement segmentés en phrases, les fréquents changements de locuteurs, parfois non natifs, la superposition de parole et de musique, les alternances de parole large bande et de parole téléphonique, la présence de différents types de bruits et la parole simultanée et le problème major du la multilinguisme des documents audio.

De ce fait, nous considérons les différentes taches d'indexation comme pouvant être approchées suivant un même paradigme qui consiste à les envisager comme un problème de classification automatique, Ce paradigme permet de résoudre plusieurs taches clés de l'indexation audio, comme l'indexation selon l'un des critères suivants :

- La décomposition du document en segments parole / music ou bruit.
- Classification de la musique (classique, jazz, blues, …)
- Classification des bruits, …
- L'identification de la langue, il s'agit de classer les documents selon la langue employée dans le discours.
- Recherche de locuteurs : il s'agit de définir où débute et où finir l'intervention d'un locuteur.
- Indexation par locuteur : il s'agit d'organiser la base de données selon les locuteurs de la base.
- Détection de mots clefs,
- Topic detection and tracking (TDT)

Pour résoudre le problème, une méthode est présentée pour regrouper les documents audio dans des groupes thématiques cohérents. Dans ce document, nous présentons une méthode qui fait appel aux cartes auto organisatrices SOM (Self Organizing Map) basé sur ontologie de domaine. Ainsi, cette thèse s'organise de la manière suivante :

Le Chapitre 1 : présente les modèles et les approches de la recherche de l'information en vue de proposer des fonctionnalités de recherche d'information pour les documents audio.

Le Chapitre 2 : traite de l'indexation audio et de la problématique de notre travail. Il s'agit d'expliquer la tache de l'indexation en terme général et comment l'appliqué pour les documents audio ainsi que la méthode de transcription et de reconnaissance de parole.

Le Chapitre 3 : présente les outils utilisés pour regrouper les documents audio de sorte que les appels similaires appartiennent au même groupe. Une méthode qui peut résoudre le problème d'indexation audio doit comprendre les quatre éléments suivants: la méthode de transcription, la méthode de représentation des documents, l'algorithme de regroupement et la méthode de représentation des résultats.

Le Chapitre 4 : présente la spécification de notre architecture de l'indexation thématique basée sur une ontologie.

Une fois que la totalité des appels téléphoniques sont transcrits, les résultats obtenus sont des fichiers textuels de différentes langues, et pour les indexer thématiquement il faut résoudre le problème de multilinguisme. **La solution proposée exige la présence d'une ontologie de domaine** où les instances des termes sont de type multi langue et que la **procédure de la projection** des documents arrive à présenter le contenu de chaque fichier transcrit comme **un vecteur de concepts**.

Pour réaliser cette solution on doit présenter les démarches nécessaires pour la création et la mise en œuvre d'une ontologie.

Le chapitre est présenté en trois parties :

- Dans la première partie, nous allons présenter les concepts fondamentaux qui permettent de créer une ontologie de domaine « Sonelgaz » sur laquelle seront projetés les documents transcrits, à savoir, sa définition, ses constituants, son utilité.

- Dans la deuxième partie, nous allons présenter la réalisation de l'ontologie Sonelgaz ainsi que les termes, les concepts et les relations utilisés.

- Dans la troisième partie : La question qui se pose c'est comment peut-on représenter les documents transcrits en utilisant notre ontologie ?

Pour répondre à cette question nous avons réalisé un algorithme qui donne les étapes de la projection en utilisant les résultats du modèle *DocCore*.

Il est possible de réaliser la projection en utilisant **les deux étapes de l'extraction des termes du document** et **la Recherche des termes correspondants à des concepts ou instances de l'ontologie. C**omme chaque terme extrait peut avoir plusieurs sens, donc correspondre à plusieurs concepts ou noeuds dans l'ontologie, il convient de procéder au calcul des mesures de similarité entre les différents sens des termes, en vue de sélectionner, pour chaque terme, le meilleur sens correspondant dans l'ontologie.

Le modèle *DocCore* est l'un des méthodes de mesure de similarité entre deux noeuds représente une valeur condensée résultant de la comparaison de deux sens possibles pour deux termes (donc deux concepts candidats); ce choix de modèle n'utilise pas la distance entre les positions des deux concepts candidats dans l'ontologie **seulement, mais encore les relations sémantiques de l'ontologie**. Et c'est important parce que notre ontologie a comme caractéristique les relations de **Terme vers sa variante**, **Concept vers terme, Concept vers concept** et **Terme vers terme.**

Le Chapitre 5 : présente les démarches de la réalisation de l'algorithme de kohonen -SOM- pour l'exploitation des résultats obtenus par la projection des documents transcrit dans l'ontologie.

Chapitre 1:

La recherche d'information

1. Introduction

L'information" joue inévitablement un rôle vital dans la société d'information d'aujourd'hui. L'invention du média électronique a permis de stocker de vastes quantités d'information dans des supports minuscules. L'ordinateur permet de traiter les quantités énormes d'information ainsi stockées. Ces contributions majeures constituent la base de la société de l'information électronique d'aujourd'hui. La quantité d'information manipulée par diverses organisations dans le monde d'aujourd'hui est élevée et sa gestion sans l'ordinateur n'est plus imaginable. Si on prend l'exemple du web, qui représente incontestablement la plus grande source d'information disponible jusqu'à présent et qui ne cesse de croître, un moteur de recherche populaire rapporte plus de huit (8) milliards de pages dans son index en juillet 2005 alors qu'elles étaient seulement 320 millions en 1997 et 3.3 milliards en septembre 2002. Le nombre d'utilisateurs est quand à lui estimé aujourd'hui à plusieurs centaines de millions. Ces facteurs ont soulevé des défis majeurs pour les tâches de collecte et de gestion de l'information, le stockage efficace de l'information, la transmission efficace de l'information et la recherche efficace de l'information.

Ces défis n'ont jamais été entièrement relevés même avec les derniers développements et inventions des technologies électroniques et de la communication. Des aspects du problème semblent se déplacer petit à petit de l'indisponibilité de l'information (source) à la difficulté d'extraction de l'information voulue à partir de sources disponibles. Les derniers développements des technologies électroniques et de la communication, semblent résoudre, en partie si ce n'est en totalité, le premier problème connu dans le passé comme "le problème de l'indisponibilité de l'information pour l'accès". De nos jours, nous sommes privilégiés dans un monde riche en information, dans lequel la plupart, si ce n'est la totalité, de l'information dont nous avons besoin est au bout de nos doigts et est prête à être exploitée. Toutefois, les conséquences de ce flux d'information ont mené à la sélection de données non pertinentes, en réponse à nos requêtes d'information. L'utilisateur se voit alors dérouté et ne sait par où commencer sa quête d'information, quand la finir, s'il a eu une information correcte et la plus récente. Il ignore aussi si ce qu'il a eu représente la totalité de l'information pertinente disponible ou alors s'il existe plus d'informations pertinentes.

La recherche d'information s'oriente vers le traitement des documents multimédias. L'information est alors contenue dans le texte, l'audio et les images (fixes ou animées) qu'il faut analyser, structurer et indexer afin de pouvoir les exploiter.

2. La Naissance

Le domaine de recherche d'information remonte au début des années 1950, peu après l'invention des ordinateurs. Comme plusieurs autres domaines informatiques, les pionniers de l'époque étaient enthousiastes à utiliser l'ordinateur pour automatiser la recherche des informations, qui dépassaient la capacité humaine : il y avait une explosion d'information après la deuxième guerre mondiale.

Le nom de « recherche d'information » (information retrieval) fut donné par Calvin N. Mooers en 1948 pour la première fois quand il travaillait sur son mémoire de maîtrise [Mooers, 48]. La première conférence dédiée à ce thème – International Conference on Scientific Information - s'est tenue en 1958 à Washington. On y comptait les pionniers du domaine, notamment, Cyril Cleverdon, Brian Campbell Vickery, Peter Luhn, etc.

Les premiers problèmes qui intéressaient les chercheurs portaient sur l'indexation des documents afin de les retrouver. Déjà à la « International Conference on Scientific Information », Luhn avait fait une démonstration de son système d'indexation KWIC (Keyword in Context) qui sélectionnait les index selon la fréquence des mots dans les documents, et filtrait des mots vides de sens en employant des « stoplistes ». C'est à cette période que le domaine de RI est né.

3. Processus de base de recherche d'information (RI)

Le processus général de Recherche d'Information (RI) est bien décrit par R.K. Belew dans son livre "FOA- Finding Out About" [Belew, 00]. Il résume le processus entier en trois (3) processus élémentaires (Figure 1-1) : "Poser une Question" (requête), "Construire une Réponse" (liste des documents pertinents) et "Evaluer la Réponse" (jugement des documents restitués). Le premier est lié au facteur cognitif humain selon lequel l'utilisateur définit son besoin en information pour combler une lacune dans ses connaissances. Cette lacune peut être partielle (confirmer ou compléter sa connaissance préalable) ou totale (acquérir une nouvelle connaissance).

Formuler clairement une question correspondant à un besoin en information est connu pour être la plus difficile partie de sa "Réponse". Un utilisateur peut ou non être capable de définir complètement les caractéristiques de la "Réponse" (besoin en information). Souvent il regarde s'il sait de quelle information il a besoin. Cet état cognitif mal défini est transformé par la suite en une expression externe dans un langage et est nommé la "Requête".

La tâche de "Construire une Réponse" est de la responsabilité du "Répondeur" (Answerer) qui est dans le cas de l'ordinateur le "Système de Recherche d'Information" (SRI). Les problèmes inhérents aux machines font que cette tâche devient plus difficile pour un SRI que pour l'être humain. Quelques uns de ces problèmes incluent le manque d'intelligence pour comprendre le problème, pour palier les difficultés dues à l'ambiguïté du langage naturel et chercher des solutions ainsi que le manque de connaissances de fond pour répondre de façon compréhensible et avec les détails adéquats à l'utilisateur.

La dernière phase "Evaluer la Réponse" implique que l'utilisateur évalue

"mentalement" les réponses qu'on lui a proposées pour décider jusqu'à quel point elles sont pertinentes. Le processus peut idéalement s'arrêter à ce niveau si l'utilisateur est complètement satisfait, ou alors, l'évaluation des réponses peut mener l'utilisateur lui même à repenser et redéfinir le besoin en information. Ce processus se produit hors du système de RI. Dans le SRI, les jugements des réponses par l'utilisateur, peuvent servir pour reformuler la requête et/ou initier un processus d'apprentissage (learning process). Le chercheur en RI a peu de contrôle sur ces facteurs liés à l'utilisateur et externes au SRI. Cependant, il est essentiel qu'il puisse les considérer, étant donné qu'il est responsable et qu'il est attendu de lui, de concevoir et de créer des systèmes flexibles et capables de gérer ces facteurs externes.

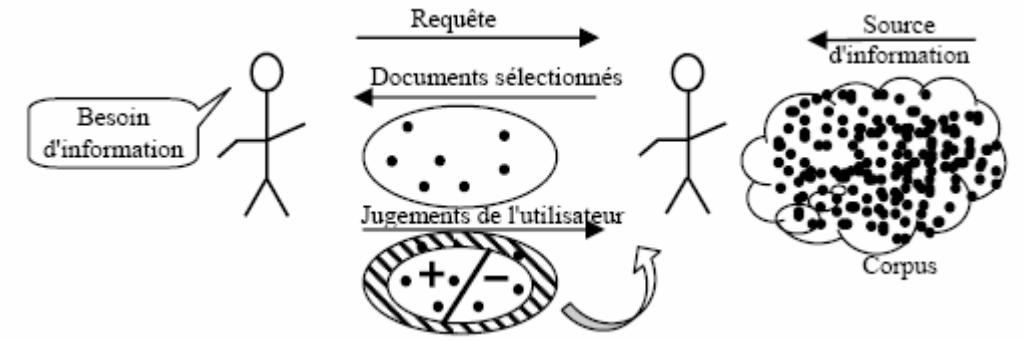


Figure 1-1: Recherche d'Information en réponse à une requête
(Schéma inspiré de [Belew, 00]).

4. Concepts de base de la RI

La Recherche d'Information (RI) [Rijsbergen, 79] [Grossman et al., 98] [Salton, 71] [Baeza-Yates et al., 99] est traditionnellement définie comme l'ensemble des techniques permettant de sélectionner à partir d'une collection de documents, ceux qui sont susceptibles de répondre aux besoins de l'utilisateur. Gérer des textes, implique stocker, rechercher et explorer des documents pertinents.
Plusieurs concepts clés s'articulent autour de cette définition :

4.1 Collection de documents :

La collection de documents (ou fond documentaire, corpus) constitue l'ensemble des informations exploitables et accessibles. Elle est constituée d'un ensemble de documents. Dans le cas général et pour un souci d'optimalité, la base constitue des représentations simplifiées mais suffisantes pour ces documents. Ces représentations sont étudiées de telle sorte que la gestion (ajout suppression d'un document) ou l'interrogation (recherche) de la base se font dans les meilleurs conditions de coût.

4.2 Document :

Le document constitue l'information élémentaire d'une collection de documents. L'information élémentaire, appelée aussi granule de document, peut représenter tout ou une partie d'un document.

4.3 Les satisfactions de besoin d'information :

La satisfaction dont parle Alan Smeaton [Smeaton et Agosti, 96] est celle de l'utilisateur qui a besoin d'une information et qui confronte les résultats d'un système de recherche d'information à ses attentes. La prise en compte de l'utilisateur comme partie intégrante du domaine de la recherche d'information n'est pas unique en informatique, le domaine de l'interaction homme machine [Preece et al., 94] par exemple, considère également cette caractéristique comme un domaine d'étude. Cette spécificité, finalement assez peu commune dans les disciplines informatiques, place explicitement l'Homme, et plus particulièrement la communication entre l'Homme et la l'Ordinateur, au centre des préoccupations. Cette satisfaction de l'utilisateur explique par ailleurs la forme des processus qui historiquement furent mis en oeuvre pour évaluer, et donc comparer, les systèmes de recherche d'information. Ces évaluations se basent sur des comparaisons entre les documents renvoyés par un système et les documents évalués par des experts comme pertinents pour une ou plusieurs requêtes. Cette satisfaction de l'utilisateur d'un système de recherche d'information est polymorphe et nécessite d'être clairement posée pour bien comprendre ces implications sur la RI.

4.3.1. Cycle d'interaction :

Pour situer les multiples aspects de cette insatisfaction, nous nous appuyons sur les travaux de Norman dans [Norman, 86] qui a caractérisé un cycle d'interaction entre un utilisateur et un système informatique par sept étapes. Ce cycle étant général à toute interaction entre un utilisateur et un système informatique, il a été appliqué à une interaction avec un système de recherche d'information dans [Mulhem et Nigay, 96] en Figure 1-2. Comme nous le voyons dans la Figure 1-2, le point d'origine de l'interaction est l'utilisateur : il a une tâche à accomplir pour retrouver des informations pertinentes pour lui. Pour la réaliser, et ensuite formuler un ou des buts à accomplir pour cette résolution. L'utilisateur commence par formuler mentalement son besoin d'information (point 1). Ensuite, il spécifie la manière de formuler son besoin sous forme de requête (point 2). Il exécute enfin les actions afférentes (point 3) grâce à l'interface fournie par le système. Le Système de Recherche d'Information (SRI) évalue la requête et renvoie une réponse. L'utilisateur prend conscience de cette réponse (point 4) par une modification de l'interface (l'écran de l'ordinateur qui présente la liste des documents trouvés comme pertinents, généralement). L'utilisateur interprète cette modification (point 5) en mettant du sens sur cette modification (compréhension de l'affichage du système). Enfin, dans le point 6 il est en mesure d'évaluer cette réponse par rapport à ses intentions initiales. Il est alors dans un nouvel état cognitif et peut soit reformuler (en cas d'insuccès par exemple) soit terminer son processus de recherche.

En se basant sur les éléments de la Figure 1-2, nous décrivons plus précisément comment un système de recherche d'information et un utilisateur interagissent à chacun des points 2 à 6, et nous considérons également les éléments liés au fonctionnement interne du système.

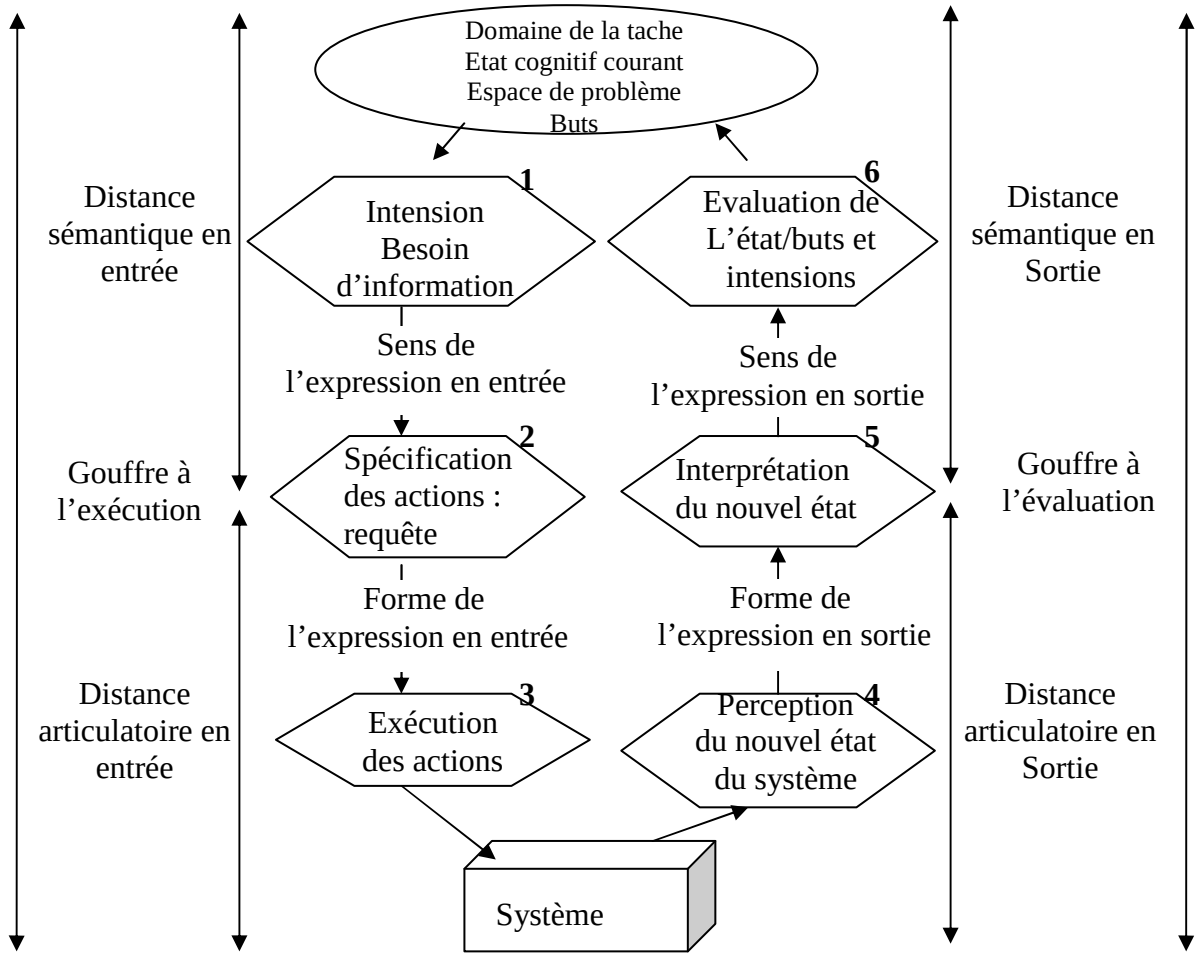


Figure 1-2 : Les sept étapes des activités d'un utilisateur d'un système de RI.

4.3.2. Expression du besoin d'information :

La satisfaction de l'utilisateur est liée à la facilité et à la fidélité avec laquelle il peut exprimer son besoin d'information. En effet, même si une approche fournit de bons résultats, elle ne sera pas satisfaisante si l'utilisateur éprouve les pires difficultés pour déterminer sa requête.

Cet élément porte sur le point 2 de la Figure 1-2 : la difficulté de formulation de requêtes pour l'utilisateur peut se révéler rédhibitoire à l'utilisation d'un SRI. Une interface de requête est formée de différents champs non explicites entre lesquels l'utilisateur doit indiquer un opérateur booléen (ET/OU/NON) peut être déroutante, dans la mesure où l'utilisateur doit penser à des opérateurs booléens qui ne lui viennent pas à l'esprit lors de sa réflexion sur son besoin d'information.

Le point 3 de la Figure 1-2 est également lié à la manière dont l'utilisateur interagit physiquement avec le système : si l'étape d'action de l'utilisateur est compliquée, il se peut qu'il modifie sa requête par rapport à son idée initiale et cette modification peut provoquer des dégradations nuisibles à une réponse adéquate du système. Si une requête doit être posée en tapant au clavier des termes et en sélectionnant des opérateurs booléens (comme dans certains moteurs recherche du Web), cette opération peut désorienter un utilisateur.

4.3.3. Présentation des réponses par le système :

Cette satisfaction de l'utilisateur est également liée à la présentation des réponses (points 4 et 5 de la Figure 1-2), c'est à dire à la partie liée au retour d'information provenant du système.

S'il est admis que le système de recherche d'information est plus satisfaisant quand il fournit des résultats ordonnés par ordre de pertinence décroissante [Maron et Kuhns, 60], il est aussi nécessaire d'en permettre une vision claire et fidèle.

Par rapport au point 4 de la Figure 1-2, l'utilisateur doit se rendre facilement compte du fait que le système a terminé de traiter sa requête et présente ses réponses. Avec les moteurs de recherche du Web, ceci devient difficile à déterminer à cause de la quantité d'images animées contenant des publicités qui perturbent l'acquisition de cette information.

L'utilisateur doit être en mesure également de déterminer les informations pertinentes dans la présentation de la réponse (point 5 de la Figure 1-2). Certaines informations, comme la mesure de pertinence du système pour les documents, peuvent aider à déterminer si le système est adéquat pour le besoin d'information de l'utilisateur; cependant, des informations comme les dates de création et de dernière modification des documents retrouvés ne sont pas toujours bienvenues.

De plus, si l'organisation de la réponse est difficilement compréhensible à un utilisateur, il ne sera pas (où sera mal) en mesure de comprendre fidèlement la réponse fournie par le système, ce qui amènera certainement son insatisfaction. Le domaine des réalités virtuelles 3D, attrayante a priori car il facilite la présentation d'informations nombreuses suivant une vue cohérente, peut paraître attractif pour traiter ce problème, mais la question n'est pas tranchée actuellement (cf. [Sebrechts et al., 99]).

Pour confronter les résultats au besoin d'information (point 6) les approches hypertextes ont permis un grand pas en avant : l'utilisateur peut aisément vérifier qu'un document est pertinent, alors que par le passé (ou dans certains systèmes utilisés dans des bibliothèques), le système ne renvoyait qu'un titre et une référence et qu'avec ces informations on ne peut pas toujours statuer sur la réussite de la recherche effectuée.

4.3.4 Le système-noyau :

Nous avons décrit ici les aspects d'interaction entre un utilisateur et un système de recherche d'information. Comme nous l'avons indiqué précédemment, la recherche d'information place l'utilisateur dans une position centrale, mais le "noyau" du système est aussi un point crucial de ce domaine de recherche. Ce noyau est lié au problème de l'analyse du contenu des documents en vue de permettre une recherche rapide et qualitativement bonne; et il est dénoté par le cube "système" au bas de la Figure 1-2.

La satisfaction de l'utilisateur intègre des aspects extrêmement pragmatiques, mais incontournables pour un système réel, comme le temps de traitement d'une requête par exemple [Baeza Yates et Ribeiro Neto, 00 ; page 390]. Ce critère a une influence sur les modèles de recherche d'information : un modèle qui ne permet pas de fournir une réponse rapide n'est pas satisfaisant, même si qualitativement il est réputé fournir de meilleurs résultats. Bien que les ordinateurs deviennent de plus en plus puissants, la quantité de données présentent dans notre société "de l'information" croit plus vite que la capacité des processeurs. La loi de Moore énoncée en 1965 stipule que le nombre de transistors des processeurs double tous les 18 mois [Moore, 65]. Par ailleurs, des estimations faites à 14 mois d'intervalle [Lawrence et Lee Giles, 98 ; 99] évalue le nombre de pages Web à respectivement 320 à 800 millions, ce qui représente une augmentation de rapport 2,5 en 14 mois. Si l'on considère que la capacité de traitement des ordinateurs est proportionnelle au nombre de transistors des processeurs, la question de l'efficacité des systèmes de recherche d'information est invariablement d'actualité.

La satisfaction d'un utilisateur revêt enfin un aspect lié à la qualité des réponses fournies par le système, que nous appelons qualité du système. Si le système n'est pas en mesure de fournir des informations pertinentes du point de vue de l'utilisateur, il ne le satisfait pas. Un système peut échouer sur cet aspect s'il n'est pas en mesure de représenter de manière adéquate le contenu du document lors de l'étape d'extraction de son contenu. Cette adéquation est en rapport avec ce qu'il est nécessaire et suffisant de prendre en compte dans un document pour fournir des résultats satisfaisants. La définition de modèles de recherche d'informations aptes à supporter la représentation du contenu sémantique des documents ainsi que les opérations de recherche est également liée au noyau du système. Certains de ces modèles sont qualifiés d'opérationnels, c'est à dire qu'ils sont utilisables directement sur des documents. Les modèles opérationnels les plus connus sont le modèle Vectoriel [Salton, 71] et différents modèles probabilistes comme celui d'INQUERY [Turtle et Croft, 90] ou le modèle de langage pour la RI [Song et Croft, 99]. D'autre part, un méta modèle comme le modèle logique de recherche d'information [van Rijsbergen, 89] définit un cadre théorique dans lequel des instances opérationnelles peuvent être définies, comme les travaux sur les graphes conceptuels [Ounis et Pasça, 98].

Le processus d'extraction des éléments importants (nécessaires et suffisants) d'un document en vue de sa recherche est appelé indexation. Pour les documents textuels par exemple, le fait d'utiliser durant la phase d'indexation tous les mots d'un document ou seulement ceux qui apparaissent dans un ensemble défini a priori produira des résultats de recherches différents si un utilisateur demande des documents traitants de termes absents de l'ensemble prédéfini. Si la cause peut être étudiée et traitée par une approche scientifique (par exemple en utilisant un plus grand ensemble de termes dans le vocabulaire d'indexation), une raison d'un autre ordre peut contrecarrer la satisfaction de l'utilisateur par d'un système de recherche d'information : l'ensemble des documents accessibles par le SRI ne contient pas ceux susceptibles d'être pertinents pour l'utilisateur car le système ne "sait" pas les indexer. Cette cause n'est ni anodine, ni triviale. Elle n'est pas anodine car l'emploi de modèles évolués adaptés à ces documents peut déboucher sur de nouvelles applications, et

dès lors provoquer de nouvelles problématiques et solutions qui profitent au domaine de la RI en général. Elle n'est pas triviale, car les approches développées en RI textuelle "classique" ne se transposent pas directement à d'autres médias et donc des travaux interdisciplinaires sont nécessaires.

Chacun des éléments ci-dessus doit être intégré pour réellement proposer des systèmes satisfaisants. Nos travaux rapportés ici portent sur les points liés à l'indexation et à la recherche de documents structurés et/ou non textuels, en gardant à l'esprit que l'utilisateur et le système doivent "se comprendre". Ceci nécessite pour le système d'interpréter les documents qu'il manipule par l'intermédiaire de symboles ayant un sens pour l'utilisateur.

5. Système de Recherche d'Information

Pour répondre aux besoins en information de l'utilisateur, un SRI met en oeuvre un certain nombre de processus pour réaliser la mise en correspondance des informations contenues dans un fond documentaire d'une part, et des besoins en information des utilisateurs d'autres part. Ces processus supposent que la collection de documents est unique. Pour des raisons d'optimisation du coût de recherche notamment en temps de réponse, plusieurs travaux se sont intéressés à la recherche parallèle sur plusieurs collections ayant des caractéristiques plus au moins proches [MacFarlane et al., 00] [Mozer, 84] [Wyle et al., 89] et à la fusion des résultats de recherche [Voorhees et al., 95] [Leclavé et al., 00] [Beigbeder et al., 05].

Un système de recherche d'information intègre trois fonctions principales représentées schématiquement par le processus en U de recherche d'information [Belkin et al., 92]. La Figure 1-3 illustre l'architecture générale d'un système de recherche d'information. D'un coté, on a l'information accessible dans le système. Elle est en général le résultat de collecte de documents ou de sous collections de documents traitant d'un même domaine ou de domaines proches. D'un autre coté, on a le besoin en information exprimé par l'utilisateur, en général sous forme de requête, une fois stabilisé. Ensuite, l'information aussi bien que le besoin en information passent par des étapes de traitement pour être exploitables. Ces processus s'appuient sur un certain nombre de modèles permettant de sélectionner des informations pertinentes en réponses à une requête utilisateur. Il s'agit principalement du processus de représentation et du processus de recherche :

5.1 Processus de représentation :

Un processus de représentation a pour rôle d'extraire d'un document ou d'une requête, une représentation paramétrée qui couvre au mieux son contenu sémantique. Ce processus de conversion est appelé indexation. Le résultat de l'indexation constitue le descripteur du document ou de la requête, qui est une liste de termes significatifs pour l'unité textuelle correspondante, auxquels sont associés généralement des poids pour différencier leur degré de représentativité. L'ensemble des termes reconnus par le SRI est rangé dans une structure appelée dictionnaire constituant le langage d'indexation.

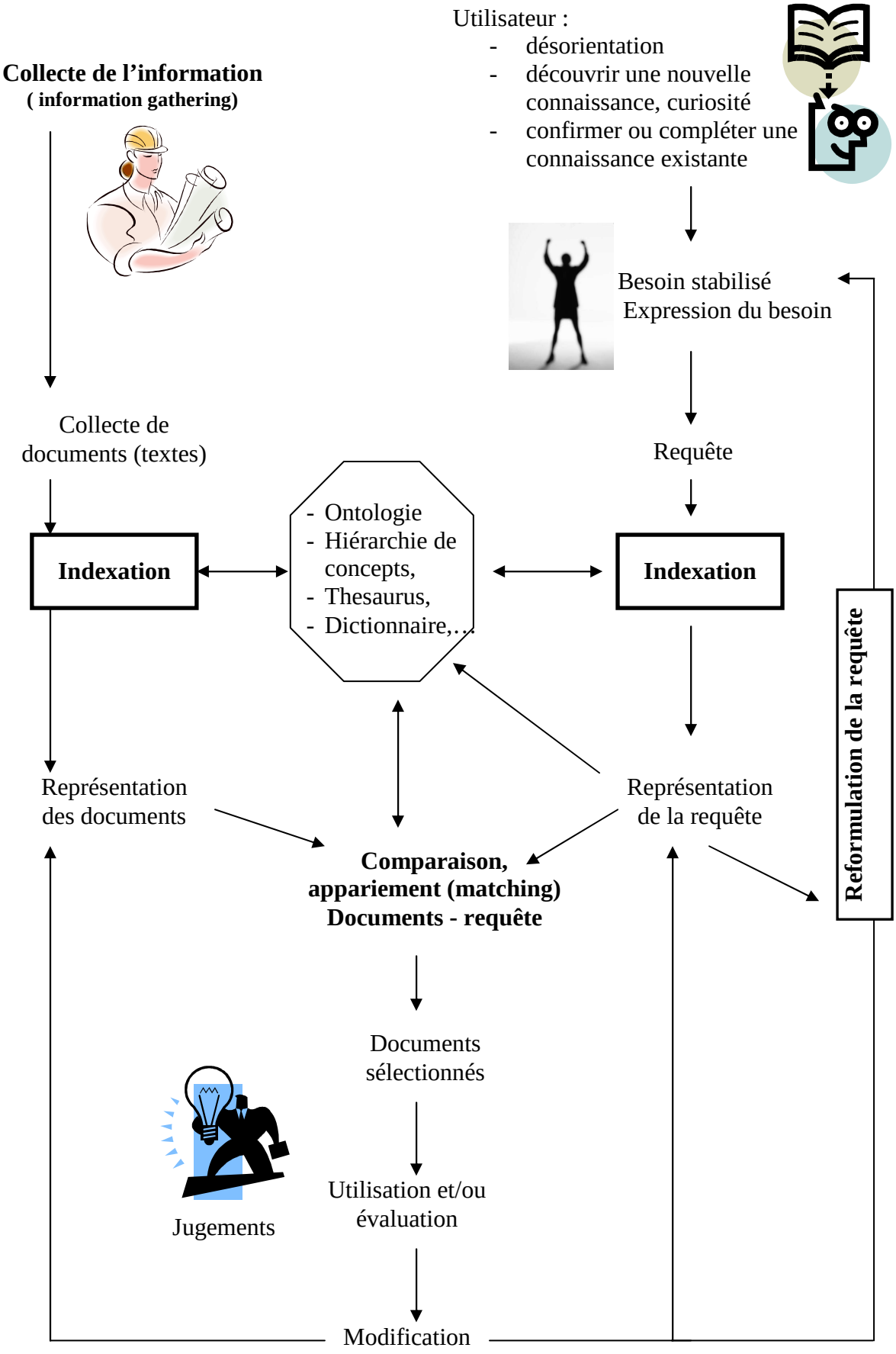


Figure 1-3: Processus général de recherche d'information.

5.2 Processus de recherche :

Il représente le processus du noyau d'un SRI. Il comprend la fonction de décision fondamentale qui permet d'associer à une requête, l'ensemble des documents pertinents à restituer. Il est utilisé pour la recherche d'informations proprement dite et est étroitement lié au modèle de représentation des documents et des requêtes. Ces modèles de recherche représentent ce qui diffère le plus entre les SRI. Ils sont inspirés de concepts mathématiques afin de pouvoir évaluer certaines relations, notamment la relation d'appariement entre la requête et les documents. La problématique majeure des SRI est de retrouver les quelques dizaines ou milliers de documents pertinents parmi des millions de documents. Cet écart de cardinalité rend cette tâche encore plus difficile.

En plus des étapes de représentation et de recherche, quelques systèmes peuvent supporter une étape supplémentaire de ***reformulation automatique de requêtes***. Cette étape a pour objectif d'améliorer les performances du SRI, donc la précision dans les réponses du système. Une des méthodes utilisées pour la reformulation de requêtes est la réinjection de pertinence, qui ajoute des termes issus des premiers documents restitués (supposés pertinents) pour améliorer la requête de l'utilisateur.

5.3 Le Processus d'Indexation

Selon [Wiki, 06] nos moyens limités d'entendement nous obligent à effectuer des classifications des objets que nous devons traiter en catégories. Celles-ci ont été considérées par la philosophie :

- Au départ comme préexistantes à l'observation : c'est la démarche dite platonicienne où l'on considère que les catégories en question préexistent à l'entendement humain, qui ne fait que les découvrir plus ou moins imparfaitement. Cette démarche subsista approximativement jusqu'à la fin du Moyen Âge, où elle était curieusement désignée sous le nom de réalisme.

- Par la suite estimées comme des regroupements ad hoc et ne visant qu'à la commodité d'usage : il n'existerait pas en soi de « champignons comestibles » et de « champignons vénéneux », mais l'effet observé des champignons nous aurait conduits à les classer fonctionnellement en comestibles et en vénéneux. Cette démarche opposée au réalisme du Moyen Âge fut nommée nominalisme. [Russell, 00] fait remarquer que si l'on devait les nommer aujourd'hui, on permuterait les deux appellations. La classification automatique vise à créer ces catégories à partir de procédés ne faisant intervenir que les données, sans la subjectivité de l'expérimentateur. Bien que les premières bases de l'approche algorithmique de la classification automatique soient relativement anciennes, ce n'est qu'avec le développement de l'informatique qu'elles sont devenues possibles à mettre en oeuvre sur de grands échantillons de données.

5.3.1 Etymologie et histoire des définitions

Les termes 'classification' et 'catégorisation' ont des histoires et des origines très différentes. A ce jour, il semble persister une confusion entre ces termes [Rehel, 05]. Aucune définition scientifique n'a pu être trouvée, hormis le Webster qui donne deux sens au mot classification : celui de 'taxonomie' (ou 'taxinomie') et celui de … catégorie.

La première définition de la classification date de 1787 [Féraud, 1787]. Le terme apparaît pour la première fois dans la cinquième édition du dictionnaire de l'Académie Française en 1798 [ACA, 1798] sous la définition : « distribution en classes et suivant un certain ordre ». Le terme 'catégorisation' n'existe pas dans le dictionnaire de l'Académie Française, contrairement au mot 'catégorie'. On pourrait néanmoins le définir comme étant l'action de créer des catégories ou le résultat de cette action. Ce mot vient du grec katêgoria (ou katègoriai : « qualité attribuée à un objet ») et du bas latin. Aristote définit les catégories comme étant les « espèces les plus générales de ce qui est signifié par un mot simple ». Il regroupe dans un même ensemble des éléments proches et dénombre dix catégories. D'un point de vue philosophique, elles ne sont pas sans analogie avec les 'Principes', opposés deux à deux, dont certains Pythagoriciens voulaient que tout fût formé : fini et infini, pair et impair, repos et mouvement, etc… D'autres mouvements philosophiques (les stoïciens, les logiciens de Port Royal, ou Kant par exemple) discuteront du nombre ou du bien fondé des catégories.

5.3.2 Classification, catégorisation et segmentation

On trouve de nombreuses définitions, souvent complémentaires, parfois contradictoires. [Jalam, 03] définit la catégorisation de textes comme étant la recherche d'une relation bijective qui consiste à "chercher une liaison fonctionnelle entre un ensemble de textes et un ensemble de catégories (étiquettes, classes)". [Sebastiani, 04] ajoute la notion de classes cibles prédéfinies et voit la catégorisation de textes comme étant une tâche de tri. Enfin, [Turenne, 01] définit la classification en ajoutant la notion de hiérarchie de classes à travers certaines propriétés communes. Mais finalement, [Rehel, 05] rappelle que dans la littérature scientifique, les termes classification et catégorisation sont indifféremment utilisés. Pourtant, il apparaît important de définir ces termes car l'impact est important pour l'évaluation [Nakache, 06].

Compte tenu de l'historique de ces deux termes et de leur acception actuelle, nous proposons de définir la classification comme étant l'action d'organiser un ensemble en structures ordonnées ou hiérarchisées. Nous définissons la catégorisation comme étant l'action d'affecter des éléments, qui possèdent des caractéristiques communes, à des catégories pré-établies, sans relation d'ordre. Cette définition se rapproche de la pensée aristotélicienne de par l'absence d'ordre ou hiérarchie et l'analogie avec les Pythagoriciens est conservée à travers le mode de pensée booléen (un élément appartient ou non à une catégorie). Par contre, dans une classification, il sera possible de mesurer l'écart entre la proposition et la valeur attendue. Enfin, nous définissons la segmentation ou le clustering comme étant une catégorisation dont les classes ne sont pas obligatoirement connues à priori.

	Classification	Catégorisation	Segmentation
Résultat	Structure avec relation d'ordre (arbre ou graphe)	Structure préexistante au traitement, sans relation d'ordre	Groupes avec des propriétés communes
Structure résultante avec relation d'ordre	Oui	Non	Non
Structure résultante préexistante	Oui	Oui	Non
Ajout de nouvelles classes	Rare	Rare	Fréquent, en fonction du besoin
Exemple	Classification des emplois	Catégories socio professionnelles, Spam ou non, …	Segmentation de clientèle
Evaluation : mesure en cas d'erreur entre classe prédite et classe réelle	[0;1] car la structure hiérarchique permet de valoriser l'écart, par exemple selon le niveau du plus grand groupe commun	{0;1} car un individu appartient ou non à une classe (raisonnement booléen)	{0;1} car un individu appartient ou non à une classe (raisonnement booléen)
Exemples d'algorithmes	CAH (classification ascendante hiérarchique) et algorithmes par apprentissage	Algorithmes par apprentissage	Knn, K means, cartes auto organisatrices de Kohonen

Tableau 1-1 : Tableau comparatif entre classification catégorisation et segmentation (Tableau extrait de [Didier Nakache, 07]).

Pour chacune de ces tâches, nous avons en entrée des individus statistiques (pour notre cas, des documents) $\{d_1, d_2,\ldots d_i\}$, décrits par un ensemble d'attributs $\{a_1, a_2,\ldots a_j\}$ que nous souhaitons organiser en classes $\{c_1, c_2,\ldots c_k\}$. L'exemple des emplois est caractéristique de notre proposition de définitions. Pour une classification des emplois, la notion d'ordre et de hiérarchie est présente alors que la catégorie socio professionnelle représente l'appartenance ou non à un groupe, sans relation d'ordre.

5.3.3 Application de l'indexation

Afin d'assurer la recherche dans des conditions acceptables de coût et d'efficacité, une étape primordiale doit s'effectuer avant l'étape de recherche effective de l'information. Cette étape consiste à analyser le document lors de l'organisation du fond documentaire afin de produire un ensemble de mots clés, appelés aussi descripteurs, que le système pourra gérer aisément puis utiliser dans le processus de recherche ultérieur. Cette opération est appelée indexation [Salton, 71][SparkJones, 79][Rijsbergen, 79][Deerwester et al., 90][Soule Dupuy, 90]. Cet ensemble de mots clés peut être regroupé dans un thésaurus [Crouch et al., 89] [Crouch et al., 92] [Frakes et al., 92], mais en pratique, un thesaurus représente une notion plus large qu'une liste de mots clés. Il regroupe plusieurs relations de types linguistique (équivalence, association, hiérarchie) et statistique (pondération).

L'indexation peut se faire selon trois modes différents :

- Manuel : chaque document est analysé par un spécialiste du domaine ou alors par un documentaliste,
- Automatique : à l'aide d'un processus entièrement informatisé,
- Semi-automatique : ici un premier processus automatique permet d'extraire les termes du document. Cependant le choix final reste au spécialiste du domaine ou au documentaliste pour établir les relations entre les mots clés et choisir les termes significatifs.

Le résultat de l'indexation est un ensemble de termes définissant ce qui est appelé le langage d'indexation. On peut distinguer deux types de langage d'indexation :

- Langage contrôlé : il s'agit d'un lexique figé de descripteurs. L'indexation est alors le plus souvent manuelle, parfois semi-automatique, un professionnel choisit un ou plusieurs descripteurs pour représenter le document.
- Langage libre : Les descripteurs sont extraits automatiquement des documents, ou de la requête de l'utilisateur. Ici, un document est le plus souvent indexé par la liste des mots qui le composent.

L'indexation manuelle a l'avantage d'assurer une meilleure correspondance entre les documents et les termes choisis par les indexeurs pour les représenter (termes d'indexation). Ceci a pour conséquence une meilleure précision dans les documents que le système de RI retourne en réponses aux requêtes des utilisateurs [Nie et al., 99]. L'inconvénient majeur de cette méthode d'indexation est l'effort intellectuel qu'elle exige (en temps et en nombres de personnes). De plus, un degré de subjectivité lié au facteur humain fait que pour un même document, des termes différents peuvent être sélectionnés par des indexeurs différents. Il peut même arriver qu'une personne, à des moments différents, indexe différemment le même document.

Dans le cas de l'indexation semi-automatique [Jacquemin et al., 02], appelée aussi indexation supervisée, les indexeurs utilisent un vocabulaire contrôlé sous forme de thesaurus ou de base terminologique. C'est le cas notamment lorsqu'il s'agit d'indexer des articles du domaine médical à l'aide du thesaurus MeSH. Les termes dans les bases terminologiques, sont préordonnés. Ils peuvent être liés par de simples relations hiérarchiques tels que *Is*-a (pour former des taxonomies ou arbres) ou par un ensemble plus riche de relations lexicaux sémantiques (dans ce cas on parle de réseau sémantique).

L'indexation automatique est sans doute celle qui a été le plus étudiée en recherche d'information, étant donnée sa faculté d'automatisation du processus d'indexation. Elle comprend un ensemble de traitements sur les documents. On y distingue : l'extraction automatique des descripteurs, l'utilisation d'un antidictionnaire pour éliminer les mots outils, la lemmatisation, le repérage de groupes de mots, la pondération des mots avant de créer l'index.

6. Les modèles de RI

La première fonction d'un système de recherche d'information est de mesurer la pertinence d'un document vis-à-vis d'une requête. Un modèle de RI a pour rôle de fournir une formalisation du processus de recherche d'information. Il doit accomplir plusieurs rôles dont le plus important est de fournir un cadre théorique pour la modélisation de cette mesure de pertinence.

De façon générale, les modèles de RI peuvent être classés en trois principales classes ou modèles qui sont :

6.1. Les modèles booléens

Le modèle booléen s'appuie sur la logique de Boole en permettant à l'utilisateur d'exprimer son besoin par une formule logique sur les mots. Les documents satisfaisant cette formule sont considérés comme pertinents et renvoyés à l'utilisateur. Lorsque cette satisfaction est binaire, il n'y a pas de notion de degré de pertinence, donc le modèle n'est utile que dans les cas où le nombre de résultats retrouvés correspond aux attentes de l'utilisateur. Afin d'y remédier, le modèle booléen a été étendu de nombreuses manières. Par exemple, [Salton et al., 83] ajoutent un degré de satisfaction de la requête logique fonction des propriétés statistiques des mots et une pondération des opérateurs logiques «ET» et «OU». Ces approches s'apparentent autant au modèle booléen qu'au modèle vectoriel. Parallèlement aux modèles fondés sur la théorie des ensembles « stricts », il existe d'autres modèles basés sur la théorie des ensembles flous qui prennent en compte la corrélation entre les mots pour définir l'appartenance de chaque document aux ensembles des mots qui les composent. Les performances de cette classe de modèles ont rarement été comparées à celles des modèles les plus répandus.

6.2. Les modèles vectoriels

Les modèles algébriques utilisent une projection des documents et des requêtes dans un espace vectoriel dans lequel ces vecteurs peuvent être comparés. La pertinence estimée d'un document est directement proportionnelle à une mesure de sa similarité à la requête. Le modèle vectoriel (Vector Space Model, VSM, [Salton et al., 75]) est le modèle le plus populaire en recherche d'information car il permet d'obtenir des performances intéressantes en nécessitant peu de ressources. L'idée de cette modélisation est d'exprimer chaque documents ($\overrightarrow{d}$) et la requête ($\overrightarrow{q}$) comme des vecteurs dans l'espace formé par le vocabulaire (équations 6.2.1 et 6.2.2). Chaque dimension de cet espace représente un mot du vocabulaire (u_i) ; la composante du vecteur document (ou requête) pour ce mot, $w_{i,d}$ est donnée dans l'équation 6.2.3, en fonction de la fréquence du mot dans le document, $f_i(d)$, par rapport à sa fréquence dans le corpus (N est le nombre de documents, et n_i le nombre de documents où apparaît le mot u_i). Cette normalisation par rapport à la fréquence dans le corpus est appelée Inverse Document Frequency (IDF) et permet de spécifier que les événements peu fréquents sont plus susceptibles d'intéresser l'utilisateur. Cette

notion est présente dans la recherche d'information depuis ses débuts [Bar-Hillel,58], mais a fait l'objet de nombreuses formulations. A partir de la représentation vectorielle d'un document, sa pertinence à la requête est estimée en calculant sa similarité avec le vecteur requête.

La similarité la plus répandue est la similarité cosine, cosinus de l'angle entre les deux vecteurs ($\overrightarrow{q}$) et ($\overrightarrow{d}$), comme le détaille l'équation 6.2.4. [Savoy et Berger ,05] comparent les formulations de pondération pour t *f* et id *f* et les différentes similarités sur la campagne CLEF 2005. Certaines par exemple prennent en compte le biais de la longueur des documents et de la fréquence de mots intra document pour permettre des améliorations significatives au sens de l'évaluation.

$$\vec{d} = (w_{0,d}..., w_{|W|,d})^T \quad 6.2.1$$

$$\vec{q} = (w_{0,q}..., w_{|W|,q})^T \quad 6.2.2$$

$$w_{i,d} = tf_{i,d} \times idf_i = log(1 + f_i(d)) \times \log \frac{N}{n_i} \quad 6.2.3$$

$$cosine(\vec{q}, \vec{d}) = \frac{\vec{d} \cdot \vec{q}}{|\vec{d}| \times |\vec{q}|} = \frac{\sum_{i=0}^{|W|} w_{i,d} w_{i,q}}{\sqrt{\sum_{i=0}^{|W|} w_{i,d}^2} \sqrt{\sum_{i=0}^{|W|} w_{i,q}^2}} \quad 6.2.4$$

Malgré la faible complexité du calcul de la similarité entre deux documents qui rend celui-ci adapté aux grands corpus, il est souvent reproché à ce modèle de ne pas prendre en compte l'ordre des mots (modèles à sac de mots), ni même la relation entre les mots « maison » et « blanche » sont des mots très répandus, donc faiblement pondérés alors que « maison blanche » devrait avoir un impact beaucoup plus fort sur la similarité).

[Wong et al., 85] ont proposé le modèle vectoriel généralisé (Generalized Vector Space Model, GVSM) pour prendre en compte les corrélations inter mots. Le modèle vectoriel impose une base orthonormale de l'espace des documents. Cette base implique que chaque vecteur représentant un mot est orthogonal à tous les autres vecteurs représentant des mots. Dans le modèle vectoriel généralisé, le vecteur représentant un mot est défini selon sa corrélation avec les autres mots du lexique. Ainsi, comme l'illustre la Figure 1-4, un vecteur document (somme des vecteurs représentant les mots qu'il contient) prendra en compte les affinités des mots à apparaître ensemble. La complexité de ce modèle est beaucoup plus grande que celle du modèle classique, pour un gain de performance pas toujours convaincant (augmentation du rappel, diminution de la précision).

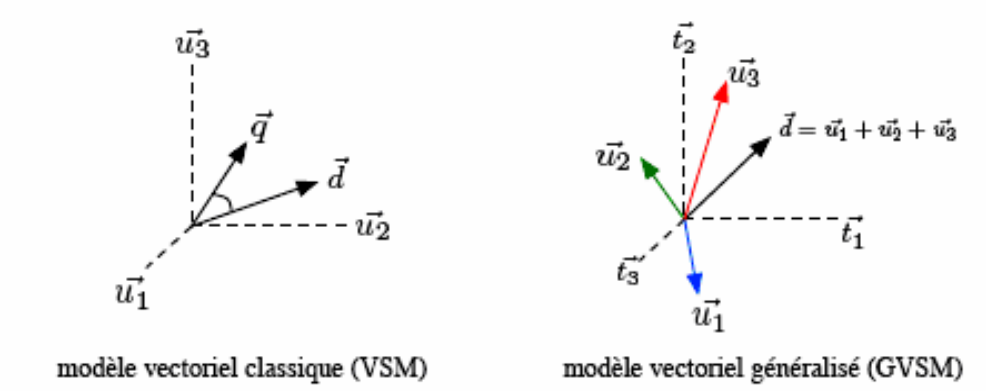


Figure 1-4: Illustration de la différence entre le modèle vectoriel classique et le modèle vectoriel généralisé et autres modèles impliquant des vecteurs mots non orthogonaux.

($\overrightarrow{q}$) , ($\overrightarrow{d}$), $\overrightarrow{U_i}$ et $\overrightarrow{t_j}$ représentent respectivement la requête, un document, un mot (unité informative) et une dimension sous jacente aux mots (thème).

Latent Semantic Analysis (LSA), également référencé sous le nom de Latent Semantic Indexing (LSI), peut être vu comme une extension de GVSM offrant une réduction de la taille de l'espace de comparaison et donc de la complexité [Deerwester et al., 90]. La méthode repose sur une réduction de la matrice documents mots X à ses dimensions principales en utilisant une décomposition en valeurs singulières (SVD). La matrice est décomposée en une multiplication de la matrice de vecteurs singuliers gauche U, de la matrice diagonale de valeurs singulières S et de la matrice de vecteurs singuliers droite VT (équation 6.2.5). Lorsque les valeurs singulières sont ordonnées de la plus grande à la plus petite, réduire le rang k de la matrice S correspond à approximer la matrice X en minimisant l'erreur au sens de la norme L2 entre les mots (équation 6.2.6, dans laquelle X* est l'approximation de X et Sk la matrice de valeurs propres réduite au rang k). De plus, les dimensions de l'espace réduit font apparaître des « thématiques » selon lesquelles sont exprimés les documents. Une projection de la requête dans cet espace permet de calculer une similarité prenant en compte les caractéristiques thématiques de la requête.

$$\mathbf{X = USV^T} \qquad 6.2.5$$

$$\mathbf{X^* = US_kV^T} \qquad 6.2.6$$

LSA offre une réduction de dimension de bonne qualité, mais nécessite d'évaluer la matrice d'occurrences entre mots et documents et de décomposer cette matrice. Afin d'éviter cette réduction coûteuse en ressources, [Kanerva et al.,00] proposent une technique nommée *Random Indexing*. Cette approche consiste à associer aux mots des vecteurs aléatoires quasi orthogonaux (contenant un grand nombre de 0 et un petit nombre de -1 et +1), de dimension fixe et de construire un équivalent de la matrice de concurrences grâce à des accumulateurs. La réduction est d'aussi bonne qualité que pour LSA, avec l'avantage de supporter le passage à l'échelle et de pouvoir être mise à jour de façon incrémentale. De nombreux autres modèles algébriques sont décrits dans la littérature tels que la recherche documentaire à base de réseaux de neurones [Wilkinson et Hingston, 91], utilisant les mots de la requête en entrée et générant

les documents en sortie ; l'emploi de LSA pour créer les vecteurs de mots du modèle vectoriel généralisé et les modèles vectoriels thématiques (TVSM, [Becker et Kuropka, 03]) associés à une classification en thèmes des mots et/ou des documents.

6.3. Les Modèles Probabilistes :

Le cadre probabiliste est très attirant pour la recherche documentaire et a été appliqué de nombreuses façons. Trois événements entrent en jeu dans ce cadre : la requête Q, un document D et la pertinence binaire R. Les modèles proposés diffèrent par leur estimation de la probabilité qu'un document soit pertinent pour une requête donnée P(R=1ID,Q). La Figure 1-5 illustre les principales approches, à savoir : le modèle classique, les modèles de langage et les modèles de pertinence. Le modèle classique de [Robertson et Spärck-Jones, 88] est fondé sur le ratio de vraisemblance entre P(R=1ID,Q) et P(R=0ID,Q), les modèles de langage de [Ponte et Croft, 98] estiment la probabilité que la requête soit issue de la même distribution qu'un document P(Q I D) et les modèles de pertinence [Lavrenko, 02] utilisent la divergence entre les distributions P(QIR) et P(DIR). Les probabilités incluant la pertinence sont difficiles à estimer dans le cas où cette variable n'est pas observée (schéma de recherche d'information classique). Par contre, lorsqu'un a priori sur la pertinence est fourni (par exemple grâce à des interactions utilisateur), ces modèles ont un avantage certain.

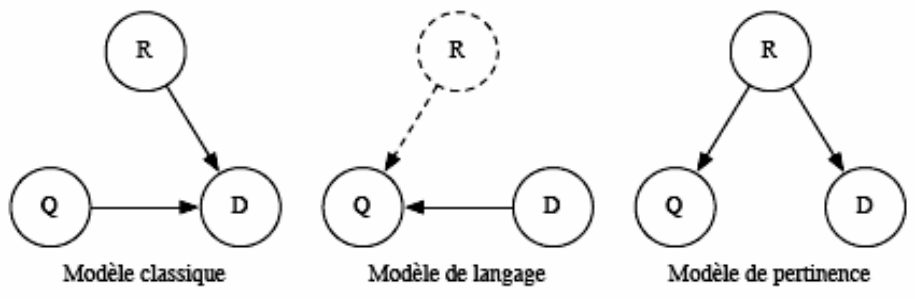


Figure 1-5: Différentes modélisations probabilistes pour la recherche documentaire selon le formalisme des modèles graphiques.

Ces modèles estiment la pertinence d'un document pour la requête en fonction des événements Q (la requête), D (le document) et R (la pertinence). Dans la Figure 1-5, une absence de flèche correspond à une hypothèse d'indépendance entre deux événements. Les éléments en pointillés sont implicites au modèle et n'apparaissent pas dans sa formulation.

Le modèle classique, nommé Binary Independence Retrieval (BIR), fait l'hypothèse que les mots sont indépendants deux à deux. Ses dernières extensions sont détaillées dans [Spärck-Jones et al., 00]. La probabilité qu'un document soit pertinent est exprimée comme le produit des probabilités de pertinence de chaque mot qui le compose. Cette probabilité est modélisée par une distribution de poisson prenant en compte l'« élitisme » d'un mot pour un document (sur pondération des mots très représentatifs du thème) en fonction de son nombre d'occurrences [Robertson et Walker, 94].

Les modèles de langage [Ponte et Croft, 98] tentent d'introduire une meilleure estimation de la distribution des mots dans les documents la modélisant par une loi multinomiale. Toutefois, [Church et Gale, 95] prouvent que ce type de distribution ne reflète pas complètement la réalité. En général, le cadre du maximum de vraisemblance est utilisé pour estimer la probabilité pour un modèle de document de générer la requête. Les idées du modèle algébrique LSI/LSA ont été reprises par pLSI/pLSA, une approche probabiliste de l'analyse en sémantique latente [Hofmann, 00]. Cette approche définit un modèle génératif de la matrice de comptes selon lequel les mots proviennent de k thèmes mis en relation par des variables latentes de mélange. pLSI a le désavantage de nécessiter l'estimation de beaucoup de paramètres et d'aboutir à du sur apprentissage. Pour lever ces problèmes, [Blei et al., 03] proposent Latent Dirichlet Allocation (LDA), un modèle où les mots des documents sont toujours générés par une distribution multinomiale. Néanmoins, les thèmes latents sont considérés comme des variables régissant le comportement des paramètres de la multinomiale à travers une distribution de Dirichlet. Ce modèle implique une estimation coûteuse de ses paramètres (généralement approximée) sans toutefois apporter les gains en performance escomptés sur une tâche de recherche documentaire [Wei et Croft, 06].

Il existe d'autres modèles probabilistes fondés sur une topologie de réseaux bayesiens, comme par exemple [Callan et al., 92], à la base du système Inquery. Ce modèle cherche à estimer l'inférence d ! u ! q. Cette inférence est aussi estimée dans le cadre plus général des réseaux de croyances [Ribeiro-Neto et Muntz, 96]. Une autre méthode, Retrieval by Logical Imaging [Crestani et van Rijsbergen, 95] apporte une vision intéressante en utilisant une logique modale pour étendre à un cadre probabiliste les idées du modèle booléen (Retrieval by Logical Imaging, RbI).

7. Evaluation des Systèmes de RI

L'évaluation des systèmes de recherche d'information constitue une étape importante dans l'élaboration d'un modèle de recherche d'information. En effet, elle permet de caractériser le modèle et de fournir des éléments de comparaison entre modèles.

Les suggestions de mesures et les techniques d'évaluation des systèmes de recherche d'information, se sont multipliées depuis une vingtaine d'années, dans la lignée des projets de la DARPA 4 dont le plus connu est sans doute le programme TREC.

D'une façon générale, un système de recherche d'information idéal a deux objectifs :

- Retrouver tous les documents pertinents,
- Rejeter tous les documents non pertinents.

8. Projet de la Recherche d'information

Voici quelques grands projets d'expérimentations dans l'histoire de la RI.

• **Projet Cranfield (dirigé par Cyril Cleverdon, 1957-1967)** [Cleverdon, 67] :

Dans la première phase de ce projet, on visait à tester l'efficacité de différentes façons d'indexer et de rechercher des documents. Ces tests sont vigoureusement contrôlés. Une collection de test est constituée d'un ensemble d'articles (18 000 dans Cranfield I) et un ensemble (1 200) de requêtes. Ces requêtes sont évaluées par des experts afin de déterminer les réponses souhaitées les articles pertinents. Les résultats d'une recherche automatique sont comparés avec les réponses souhaitées pour mesurer la performance en terme de précision et rappel.

Le projet Cranfield a une influence marquante sur toute l'histoire de la RI. On utilise encore aujourd'hui les mêmes principes d'évaluation pour les systèmes de RI.

• **Projet MEDLARS – MEDical Literature Analysis and Retrieval System (F. Wilfrid Lancaster, complete en 1968)** [Lancaster, 68]**:**

Comme l'indique son nom, les documents dans la collection sont dans le domaine biomédical. Ces documents sont indexés manuellement, avec un vocabulaire contrôlé. Les résultats sont évalués en terme de précision et de rappel. Les résultats de ce projet montrent qu'en utilisant une approche automatique, il est possible d'atteindre la même performance avec une indexation manuelle et un vocabulaire contrôlé. Une analyse des résultats a aussi montré que l'utilisation d'un vocabulaire contrôlé et de l'indexation manuelle étaient largement responsable des cas d'échecs dans la recherche de documents pertinents, qui peuvent être évités par l'approche automatique.

• **SMART (Gerard Salton, 1ère version 1961-1965)** [Salton, 71]:

Dans ce projet, une série d'expérimentations a été menée, portant sur divers sujets comme :

- La comparaison entre l'indexation manuelle et l'indexation automatique ;
- Le problème de recherche d'information interactive et la rétroaction de pertinence (relevance feedback);
- L'architecture de système de RI ;
- L'utilisation du modèle vectoriel ;
- Le regroupement de documents (ou clustering) .

Le système SMART fut réécrit dans les années 1970 et 1980 par E. Fox et C. Buckley. Ce système a été, et est encore, utilisé par de nombreux chercheurs pour des expérimentations en RI. Le système SMART est sans doute le système qui a eu le plus grand impact sur l'histoire de la RI.

• **Projet STAIRS - STorage And Information Retrieval System** [Blair et al., 85] **:**

Les documents sont dans le domaine de droit. L'indexation automatique utilise la troncation de suffixes, et la recherche exploite une liste de synonymes. Contrairement aux expérimentations antérieures qui utilisaient de petites collections, les tests de Blair et Maron portent sur une collection de taille réaliste – 40 000 documents totalisant 350 000 pages. Le résultat montre que la performance de moins de 20% de rappel est insuffisante

dans le domaine de droit pour lequel le rappel est très important.

• **TREC - Text REtrieval Conference, (D. Harman, 1992 -)** [Harman, 92] **:**

Cette série de conférences a pour objectif de tester des méthodes et des systèmes de RI avec des collections de plus grandes tailles. Elle est organisée annuellement. Les tâches (tracks) changent d'une année sur l'autre, mais elles reflètent bien les intérêts des chercheurs et les besoins réels. Au fil des années, il y a eu la RI ad hoc (la tâche classique de RI – soumettre des requêtes sur une collection statique), le filtrage de l'information, la RI non anglais (en espagnol, français, chinois) et translinguistique (retrouver des documents dans une langue différente de celle de la requête), la question réponse, la RI multimédia (vidéo et parole), etc. Ces conférences attirent chaque année des chercheurs universitaires et industriels. Les conférences TREC ont grandement contribué au développement récent de la RI, en fournissant des collections de tests réalistes, et en offrant une nouvelle méthodologie d'évaluation. Elles ont grandement stimulé le domaine de RI.

9. Extension à la parole

Les premières approches de la recherche d'information dans un contenu parlé ont d'abord utilisé des techniques similaires à celles développées pour les documents textuels, appliquées à la transcription automatique du flux de parole. La recherche documentaire audio (Spoken Document Retrieval, SDR) est la première formalisation de la tâche au travers de la campagne TREC 7. Cette tâche est associée à la recherche d'information dans des documents papier numérisés par Optical Caracter Recognition (OCR) car, dans les deux cas, les erreurs introduites peuvent être assimilées à un bruitage du contenu linguistique originel. La tâche SDR de TREC [Garofolo et al., 99] consiste à indexer 500 heures d'émissions radio en anglais, en utilisant les transcriptions automatiques (à différents taux d'erreur de mots, Word Error Rate, WER) des documents issus de la campagne Hub 4 (Przybocki et al., 98) organisée par NIST. L'information recherchée dans les documents audio est exprimée sous la forme d'une requête textuelle semblable à celles exploitées dans TREC ad-hoc.. Il faut remarquer que la plupart des systèmes de recherche documentaire fonctionnent soit sur du texte soit sur l'audio, mais ne mélangent pas les deux modalités. [Sanderson et Shou, 02]; [Favre ,03] soulignent qu'en général ce mélange défavorise l'audio et qu'aucune technique permettant de réduire cet écart n'a été proposée à ce jour.

Les évaluations TREC montrent que le taux d'erreur de mots est linéairement corrélé aux performances en recherche documentaire et qu'un taux d'erreur inférieur à 40% permet d'obtenir des résultats acceptables par l'utilisateur [Garofolo et al., 99]. Cette bonne réussite s'explique d'abord par la longueur des requêtes TREC et la quantité d'informations qu'elles contiennent (environ 10 mots porteurs de sens, à comparer à des requêtes WEB de moins de 2 mots en moyenne). L'impact du taux d'erreur peut être limité à 10% des performances sur la transcription manuelle en utilisant des techniques d'expansion de requête et de document. [Johnson et al., 00] notent que le gain des différentes techniques n'est pas cumulatif et que l'utilisation de corpus externes propres (et thématiquement proches des données traitées) pour l'expansion est bénéfique. Toutefois, [Hansen et al., 04] font face à des conditions moins favorables sur les données de la National Gallery of the Spoken Word (NGSW) avec des taux d'erreur de mots de 40% et observent qu'un bon choix des paramètres utilisés lors de l'expansion permet

d'obtenir un cumul des gains (20% relatif au total).

Des données audio sont utilisées dans le cadre d'une autre tâche intéressante lors de la campagne Topic Detection and Tracking (TDT), pour laquelle il faut faire du suivi de thème et détecter les nouveautés dans le flux d'informations [Allan, 02]. Cette tâche a impliqué la mise au point de nouvelles méthodes de détection de coupures thématiques en utilisant aussi bien le contenu liguistique que le contenu audio. Les techniques de recherche d'information audio tentent maintenant d'aller plus loin que la parole des flux radio, en se focalisant sur la parole spontanée et sur les applications temps réel. [Brown et al., 01], par exemple, s'attachent à annoter des flux télévisuels et des conférences avec des informations susceptibles d'intéresser le spectateur. Pour ce qui est de la parole spontanée, [Byrne et al.,04] ont annoté un corpus de 10000 heures d'interview puisées dans les enregistrements de la Shoah Visual History Foundation. Ce corpus est à ce jour le plus grand corpus de parole spontanée réunissant de nombreux locuteurs sur le même thème; ce corpus permettra certainement de mieux tester les approches de recherche d'information et de segmentation que les corpus téléphoniques de Switchboard [Godfrey et al., 92].

Le taux d'erreur de mots n'est pas le seul problème lié à la transcription automatique du contenu parlé, les systèmes de transcription ont en effet un vocabulaire limité aux mots les plus fréquents (dans le but de minimiser le taux d'erreur de mots, tout en limitant les ressources nécessaires). Les mots les moins fréquents sont considérés comme des mots hors vocabulaire (Out ofVocabulary, OOV) et ignorés lors du décodage du signal de parole. Ils ne pourront être retrouvés et paradoxalement, ce sont justement les événements peu fréquents et inattendus qui sont le plus susceptibles de sélectionner les documents pertinents. En effet, le moteur de recherche SpeechBot [Thong et al., 00] a offert pendant plusieurs années l'accès à du contenu parlé transcrit automatiquement sur le web et il a été observé que plus de 12% des mots utilisés dans les requêtes étaient hors vocabulaire. Le problème est aussi lié aux modèles de langages nécessairement mal estimés pour les langues à ressources minoritaires comme les langues africaines [Abdillahi et al., 06]. Des techniques basées sur l'utilisation de sous parties des mots comme les phonèmes ou les radicaux sont apparues pour essayer de remédier au problème des mots hors vocabulaire [Wechsler et al., 98]. Ces approches demandent une phonétisation de la requête, puis la comparaison de cette séquence de phonèmes avec les hypothèses de transcription phonétique du système de transcription. Une mesure de confiance basée sur l'adéquation entre la modélisation phonétique et le contenu acoustique est utilisée afin de ne rapporter que des séquences proches de la meilleure hypothèse (probabilité a posteriori du sous graphe d'hypothèses passant par le chemin étudié). L'utilisation du treillis/ de phonèmes apporte un gain intéressant en rappel au détriment de la précision car de nombreux passages ont une transcription phonétique similaire à la requête sans pour autant impliquer la présence des mêmes mots. [Yu et Seide, 04] intègrent la recherche dans le treillis de phonèmes avec une recherche dans le treillis de mots afin de profiter de l'augmentation à la fois du rappel et de la précision. Face à un taux d'erreur de mots de l'ordre de 43% à 60% selon les conditions, ils observent un gain de 10% en performance sur la détection de mots (word spotting) par rapport à l'utilisation d'une des deux méthodes isolément. Les mots hors vocabulaire ont des effets de bord sur la qualité de la transcription, car ils sont remplacés par une séquence de mots acoustiquement proches, mais qui diverge du contenu réel et provoque des erreurs autour du mot inconnu. [Bazzi et Glass, 00] proposent par exemple d'introduire un mot «INCONNU» dans le vocabulaire et d'utiliser un modèle phonétique

complet pour représenter son acoustique. Cette approche par cas particulier n'entre pas dans les cadres mathématiques utilisés en transcription et demande un contrôle fin de son activation.

Nous retiendrons que la recherche d'information audio s'est surtout concentrée sur l'interaction transcription/recherche. Pour preuve, le standard MPEG 7 [Manjunath et al., 02] adopte, entre autre, la représentation par treillis d'hypothèses phonétiques pour le stockage des transcriptions automatiques de flux structurés afin d'autoriser la remise en cause du lexique de reconnaissance.

10. Conclusion

Notre démarche vise à étudier des modèles et des approches en vue de proposer des fonctionnalités de recherche d'information pour les documents audio.

L'extraction de ce contenu et la modélisation du processus de recherche. Les éléments décrits dans ce rapport mettent davantage en relief l'étape d'indexation des documents, mais présentent des aspects liés à la recherche de ces documents.

Nos travaux sont donc axés sur ces nouvelles catégories de documents (nouvelles par rapport au texte du point de vue de l'histoire de la recherche d'information), et nous définissons des modèles et systèmes à même de satisfaire le besoin d'information de l'utilisateur.

Nous traitons les documents dont le contenu est peu abordé par la communauté de recherche d'information. La raison de cette "désaffection" n'est pas que ces données sont peu nombreuses (au contraire), mais que les modèles habituels de recherche d'information s'appliquent difficilement et nous devons définir les bases des approches susceptibles d'apporter des solutions à cette problématique.

Nous estimons que la recherche d'information doit maintenant prendre en compte les nouveaux types et/ou média de documents. Cette prise en compte nécessite la définition de nouveaux modèles de recherche d'information adaptés et la réalisation des outils adéquats.

Chapitre 2:

Indexation audio

1. Introduction

La quantité d'information rendue disponible par les réseaux croit fortement chaque jour. Cette information représente une grande richesse dès lors qu'elle est structurée et accessible. L'indexation et la recherche d'information sont devenues des tâches primordiales pour réaliser ces objectifs.

Avec l'apparition de nombreux documents multimédias, associé à la généralisation de leur utilisation pour de nombreuses applications, représente de nouveaux challenges pour la société de l'information. Le volume continuellement croissant de ces données augmente la difficulté de leur accès. Ainsi, une conséquence directe de l'accroissement rapide de ces données numériques est un fort besoin pour des nouvelles méthodes efficaces d'indexation, de classification et d'accès par le contenu.

La recherche documentaire sur des documents parlés a été rendue possible en utilisant la reconnaissance automatique de la parole pour indexer les transcriptions, sans toutefois aborder l'ensemble de la problématique de l'indexation sonore, cette activité poursuit plusieurs axes de recherches.

2. Définition de l'indexation du point de vue de la documentation

L'indexation consiste, en toute généralité, à substituer au document originel une représentation de ce document sous la forme d'une description abrégée, le plus souvent textuelle. Sa finalité est de permettre de repérer rapidement au sein d'un ensemble (ou d'un document), les documents (ou les extraits) pertinents en fonction d'une requête donnée. Cette définition, très générale, est conforme à celle donnée par la norme en vigueur [Afnor, 96].

Pour certains, le terme indexation désigne à la fois l'acte de générer cette description abrégée, et les techniques de recherche qui y sont associées. Ceci est particulièrement le cas pour les systèmes d'indexation automatique qui englobent les deux actes en un seul. Bien entendu ces deux phases doivent être pensées en adéquation. Mais dans le monde documentaire, l'usage est de nettement dissocier les deux, indexation et recherche.

3. Le rôle de l’indexation

L'indexation a pour rôle de représenter de façon homogène le contenu sémantique des documents du corpus. L'homogénéité de l'indexation réfère à la conformité, de cette représentation, à un langage d'indexation définissant (en extension ou en intention) les termes d'indexation utilisables. La notion de termes d'indexation est à prendre dans notre contexte au sens large, et nous entendons par terme d'indexation toute forme produite par l'indexation d'un document, quelle que soit sa complexité.

Le terme "indexation" encapsule donc deux problèmes bien distincts :

- La définition d'un langage d'indexation, permettant la représentation des concepts des documents du corpus ;
- La mise en place d'un processus d'indexation permettant l'extraction, à partir des documents du corpus, de termes d'indexation, c’est à dire de leur représentation conforme au langage d'indexation.

Même si elle est très vague, cette première définition est cependant consensuelle. Par exemple Borko et Bernier [Borko, 78] disent "indexing is the process of analysing the informational content of records of knowledge and expressing the informational content in the language of the indexing system", alors que Wellish [Wellish, 91] donne une définition plus technocrate se référant à la norme ISO 5127/1: "an operation intended to represent the results of the analysis of a document by means of a controlled or natural indexing language". De même Rowley [Rowley, 88] écrit "the indexing process creates a description of a document or information, usually in some recognized and accepted style or format". Salton [Salton, 83] complète cette définition, en ajoutant trois objectifs à l'indexation :

“- to allow the location of documents dealing with topics of interest to the user ;
- to relate documents to each other, and thus relate the topic areas, by identifying distinct documents dealing with similar, or related, topic areas ;
- to predict the relevance of individual documents to specific information requirements through the use of index terms with well-defined scope and meaning."

Ces différentes définitions nous montrent la dualité de l'indexation : représenter le contenu des documents afin de permettre aux utilisateurs de les retrouver. Ces deux objectifs sont difficiles à réunir, et la recherche en indexation le montre bien. En effet dans la plupart des travaux, l'indexation est soit orientée document soit orientée requête.

L'indexation orientée document a pour objectif de résumer ou de représenter le contenu de chaque document, c’est à dire son signifiant et son signifié. L'indexation orientée requête doit, pour chaque document, refléter les requêtes pour lesquelles il est pertinent : l'indexation d'un document doit alors représenter les raisons pour lesquelles un utilisateur consulte ce document.

3.1 Indexation orientée document

L'indexation orientée document consiste à définir, à partir du document seulement, son contenu, que l'on qualifie dans ce contexte, d' "à-propos". Lancaster [Lancaster, 91] décrit cette indexation comme: "a conceptual analysis, which, first and foremost, involves deciding what a document is about - that is, what it covers". Indexer un document revient à définir le processus qui permet de passer de la forme ou du signal d'un document à son fond, en d'autres termes de son signifiant à son signifié. Déterminer le signifié d'un document est une démarche délicate et subjective, car beaucoup de paramètres interviennent dans cette identification : la qualité du signifiant, l'indexeur, la base de connaissances, …

3.2. Indexation orientée requête

La façon la plus classique de procéder à une indexation orientée requête est d'anticiper les requêtes et donc de confronter chaque document de la base à une liste de requêtes prédéfinies. La liste des requêtes forment alors le langage d'indexation.

Certains utilisent la méthode suivante pour déterminer un langage d'indexation : pour tout document du corpus, un groupe de documentalistes répond à la question "pourquoi un de nos utilisateurs serait-il intéressé par ce document ?" En répondant à cette question par une liste de termes, les documentalistes génèrent ainsi un langage d'indexation. Ensuite, le processus d'indexation procède à l'indexation grâce à un

filtrage : l'indexeur vérifie chaque terme associé a priori à un document et se demande : "est-ce que l'un de nos utilisateurs intéressé par ce document utiliserait ce terme pour formuler sa requête ?".

Les problèmes majeurs posés par une indexation orientée requête résident dans son évolution face à de nouvelles requêtes, et surtout dans la difficulté à l'automatiser.

4. Rôle d'un terme d'indexation

Un document indexé est constitué d'un ensemble de termes d'indexation, dont le rôle est de refléter le contenu sémantique du document. Un bon terme d'indexation doit pour cela jouer une double fonction.

D'une part, il doit refléter tout ou partie du contenu du document, de façon à ce que le document soit retrouvé quand il est recherché. Cette propriété est à mettre en relation directe avec la mesure de rappel du système.

D'autre part, un bon terme d'indexation doit permettre de bien partitionner le corpus entre les documents qu'il indexe et les documents qu'il n'indexe pas. En ce sens, un bon terme d'indexation n'indexe pas tout le corpus. Cette propriété est à mettre en relation directe avec la mesure de précision du système.

5. Indexation automatique de document audio

Les recherches en indexation automatique de document audio ont été guidées par des projet comme Informedia [1], THISL [Abberley et al., 99], ou OLIVE [De Jong et al., 99] et surtout par les campagnes d'évaluation américaines organisées dans le cadre des conférences TREC par le NIST. TREC [2] (Texte REtrieval Conference) est autant une conférence qu'une campagne d'évaluation dont la tâche principale est l'indexation et la recherche d'information. La conférence permet l'évaluation de systèmes qui à partir d'une question ou d'une requête, sélectionnent un ensemble de documents. Le cadre correspond bien aux problématiques posées par l'indexation manuelle, puisque la collection est connue mais pas les requêtes. La question de la généricité et éventuellement de la pérennité d'une indexation est formellement posée. Les évaluations organisées dans le cadre de TREC se focalisent sur l'accès aux données textuelles dans un contexte multilingue, mais elles s'étendent également à d'autres types de données comme c'est le cas avec SDR [3] (Spoken Document Retrieval) puis TRECVID [4] (Vidéo).

1 http ://www.informedia.cs.cmu.edu
2 http ://trec.nist.gov/
3 http ://www.nist.gov/speech/tests/sdr/2000/
4 http://www-nlpir.nist.gov/projects/trecvid/

La motivation des évaluations SDR est de permettre le développement de systèmes d'accès à une information non textuelle telle que l'audio en réunissant les communautés de la reconnaissance automatique de la parole et de la recherche d'information. Les technologies de reconnaissance de la parole permettent de transcrire automatiquement des documents audio, créant ainsi une interface textuelle avec le signal. La transcription contribue à la constitution d'un document au format texte qui se substitue à l'audio et sur lequel il sera possible d'appliquer des techniques éprouvées d'indexation automatique et de recherche d'information. Le lien avec l'indexation manuelle est ici évident, puisque la démarche préconisée passe par un document de substitution qui peut s'assimiler d'un point de vue fonctionnel à la notice documentaire.

De 1997 à 2000, le cadre d'évaluation a beaucoup évolué cherchant toujours à se rapprocher d'une application réaliste. Les informations données par la suite sont relatives aux deux dernières campagnes SDR (TREC-8 en 1999 et TREC-9 en 2000). En pratique, un système SDR est la combinaison d'un système de reconnaissance automatique de la parole et d'un système de recherche d'information, comme le montre la Figure 2-1. La sortie du système de reconnaissance peut être un treillis de mots, une liste des meilleures hypothèses, une transcription phonétique, ou la meilleure hypothèse, qui est le cas le plus répandu. L'indexation transforme d'abord le texte en une forme normalisée avec comme étapes, le filtrage à l'aide d'une stoplist (liste de termes à retirer, la plupart des termes sont des mots outils ou très fréquents), la lemmatisation, et la normalisation. Puis, pour chaque document un score est calculé. Les scores les plus utilisés sont soit de type Okapi (TF*IDF) [Robertson et al., 1994], soit basé sur les probabilités unigrammes. La requête est très souvent un texte court qui ne contient pas nécessairement les index qui se trouvent dans les documents pertinents. L'expansion de requête permet d'augmenter le nombre de termes en effectuant une première requête soit sur le même corpus (Blind Relevance Feedback), soit sur un corpus parallèle (ParaUel Blind Relevance Feedback). Cette première requête permet de sélectionner un premier ensemble de documents. Les index les plus fréquents de ces documents sont sélectionnés pour enrichir la requête.

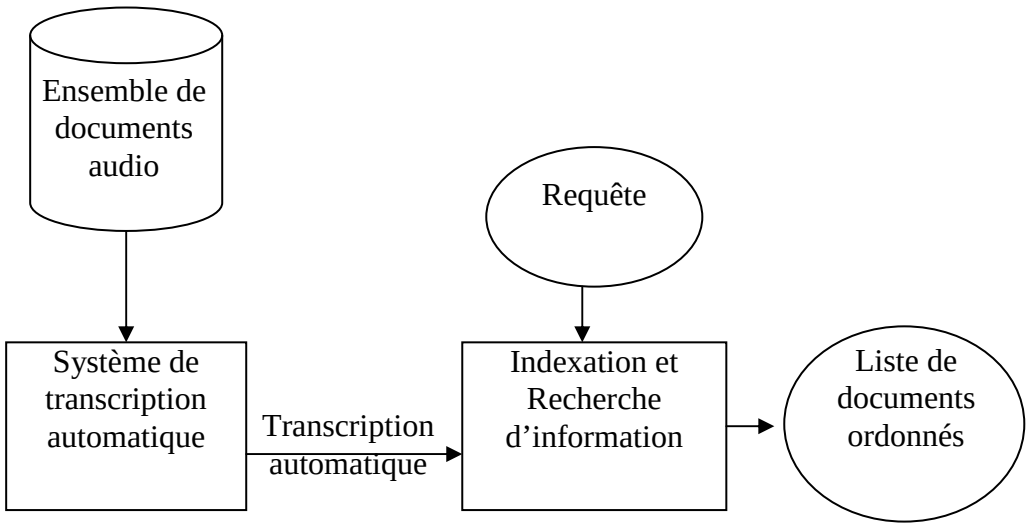


Figure 2-1 : Architecture type d'un système SDR

6. Spécificité de la parole transcrite

La sortie d'un système de transcription automatique se distingue des textes « écrits» que les systèmes d'indexation ont l'habitude de traiter. En premier lieu, une transcription contient des erreurs de reconnaissance qui modifient le contenu syntaxique et sémantique du discours, la parole présente certaines spécificités qui sont dues à la spontanéité du discours, telles que les hésitations, les répétitions, ou les modifications et les corrections du propos.

À cela, s'ajoute le fait que la grammaire n'est pas toujours respectée. Enfin, une transcription automatique n'est qu'une suite de mots localisés dans le temps. Cette suite de mots n'est pas segmentée en phrase, ne contient aucune ponctuation et pour certaine langue comme l'anglais la casse est unique ; souvent, le choix d'une casse unique relève de l'habitude et de l'inertie plus que du bon sens. Ces dernières spécificités sont particulièrement dommageables car la ponctuation, la phrase et le paragraphe permettent dans un texte de délimiter les unités thématiques du discours. Et beaucoup de techniques d'extraction d'information s'appuient en partie sur la casse et la ponctuation.

7. Reconnaissance automatique de la parole

Le système de transcription automatique est un des éléments fondamentaux des systèmes d'indexation automatique de documents audio. Il s'appuie sur une modélisation statistique du processus de génération de la parole.

7.1 Reconnaissance statistique de la parole

La plupart des systèmes reposent sur une modélisation statistique du processus de génération de la parole. Mettre des mots sur un signal équivaut à trouver la séquence de mots W = w1.w2…..wn la plus probable étant donné le signal observé. Chaque mot de cette chaîne appartient à l'ensemble fixe et fini que constitue le vocabulaire V. Et en fait de signal, c'est une suite de vecteurs d'observations acoustiques X. Ceci se traduit par l'équation suivante :

$$\hat{W} = \arg\max_{W} P(W / X) \qquad 7.1.1$$

où P(W|X) est la probabilité de la séquence de mots W étant donné la suite d'observations acoustiques X. L'application de la formule de Bayes permet d'inverser les dépendances et la modélisation «Source/Canal» permet de décomposer le processus de production de la parole comme le montre la Figure 2-2. Le message W est généré par un modèle linguistique P(W). Ce message est transformé par le modèle de prononciation P(H/W) en une séquence de phones H.

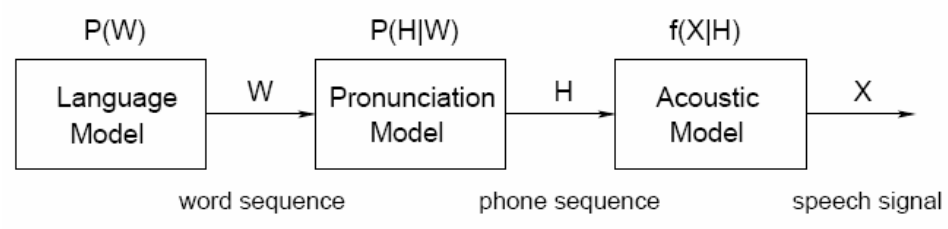


Figure 2-2: Modèle source/canal du processus de production de la parole (Figure extraite de [Gauvain et Lamel,03]

Enfin le canal acoustique f (X/H) encode H dans le signal X. La reconnaissance de la parole consiste alors à trouver la séquence de mots W* qui maximise la probabilité a posteriori de W décomposée de la manière suivante :

$$\hat{W} = \arg\max_{W} \sum_{H} P(W)P(H/W)f(X/H) \qquad 7.1.2$$

L'équation 7.1.2 met en évidence les différentes composantes et l'architecture d'un système de reconnaissance automatique de la parole telle qu'elle est représentée à la Figure 2-2. Le décodeur cherche la séquence de mots la plus probable en utilisant les modèles acoustiques pour estimer f (X/H), un modèle de prononciation pour P(H/W), et un modèle de langage pour la probabilité a priori P(W). Donc pour qu'un mot existe du point de vue du décodeur, il est indispensable qu'il fasse partie du vocabulaire, que sa transcription phonétique soit disponible, et que le modèle de langage soit apte à lui attribuer une probabilité.

7.1.1 Codage du signal

Le signal de parole est échantillonné avec une fréquence comprise entre 8 et 16 kHz. Une transformée de Fourier à court terme (algorithme FFT) est appliquée sur une fenêtre d'observation de 30 ms et ce, toutes les 10 ms. Un filtrage est effectué pour mettre le spectre à l'échelle MEL. C'est une échelle perceptive qui modélise à l'aide d'un banc de filtres, la réponse en fréquence du système auditif humain. Deux types de coefficients peuvent alors être calculés : les coefficients MFCC [Davis et Mermelstein, 80] sont issus d'une transformation de Fourier inverse appliquée au logarithme du spectre de puissance, alors que les coefficients PLP sont obtenus par transformation de Fourier inverse de la racine cubique du spectre de puissance suivie d'une analyse par prédiction linéaire [Hermansky, 90]. Généralement, les douze premiers coefficients sont retenus, auxquels s'ajoute le logarithme de l'énergie normalisée. Treize coefficients sont ainsi obtenus qui représentent un intervalle de signal de 10 ms. La dimension du vecteur acoustique est finalement augmentée à 39 composantes en ajoutant l'approximation de la dérivée première et seconde des treize coefficients (les coefficients delta et delta-delta).

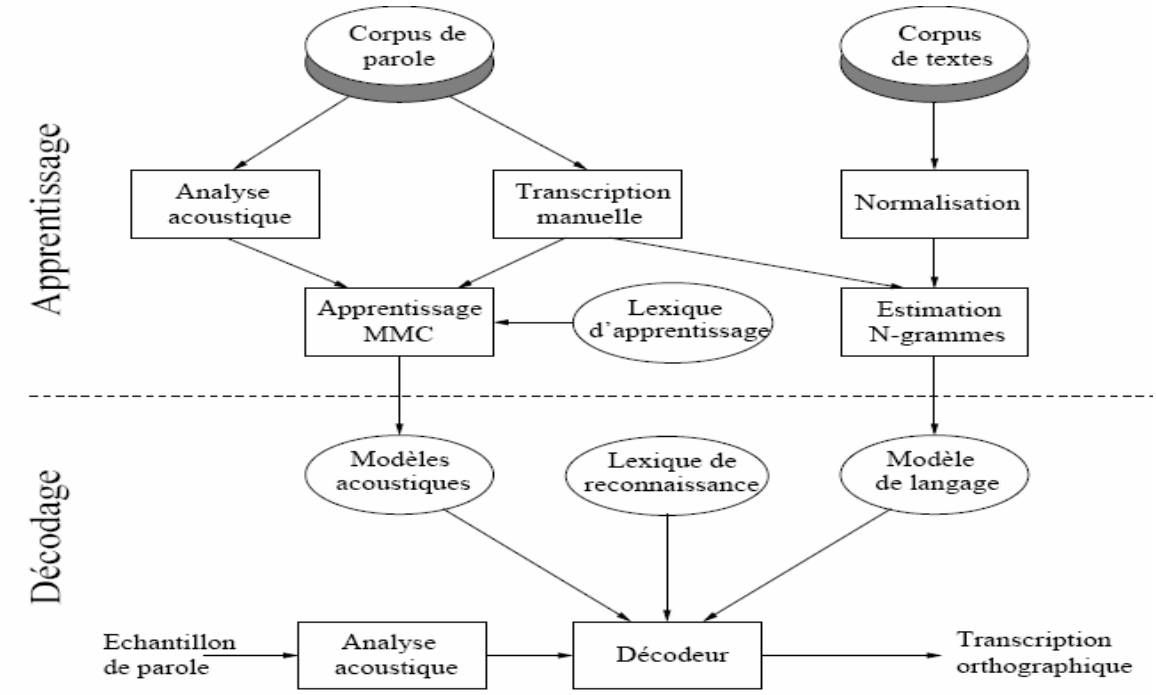


Figure 2-3 : Architecture d'un système de reconnaissance automatique de la parole (Figure extraite de [Gauvain et Lamel,02]

7.1.2 Modélisation acoustique

Les modèles de Markov Cachés (MMC) sont utilisés dans la plupart des systèmes de reconnaissance automatique de la parole pour modéliser les séquences de vecteurs acoustiques à partir d'unités acoustiques. L'unité acoustique élémentaire est l'allophone : un phonème en contexte modélisé par un MMC à 3 états tel qu'il est représenté à la Figure 2-3. À chaque état est associée une densité de probabilité sur les vecteurs acoustiques. Le processus stochastique de génération d'une suite de T vecteurs acoustiques X = (x_1,…. x_T) par la séquence d'états s = (s_0,……, s_T) est alors décrit par la densité de probabilité conjointe f(X, s) :

$$f(X,s) = \pi_{s0} \prod_{t=1}^{T} a_{st-1\ st}\, f(Xt \mid St) \qquad 7.1.2.1$$

où π_s est la probabilité initiale de l'état s, a_{ij} est la probabilité de transition de l'état i à l'état j, et f(.Is) est la densité d'émission associée à l'état s. L'entraînement des modèles acoustiques consiste à estimer pour chaque MMC les probabilités de transition et les densités d'émission associées à chacun de ses états. Pour ces dernières, le choix se porte généralement sur une estimation paramétrique qui fait l'approximation d'un mélange de gaussiennes. La densité d'émission s'écrit alors :

$$f(X_t \mid s) = \sum_{K=1}^{K} \omega_k N(X_t \mid m_{sk}, \sum_{sk}) \qquad 7.1.2.2$$

où ω_k est le poids associé à la gaussienne k avec un vecteur moyen m_k et une matrice de covariance Σ_k (souvent supposée diagonale). La méthode d'entraînement est le plus souvent fondée sur le critère du maximum de vraisemblance qui consiste à estimer les valeurs des paramètres qui maximisent la vraisemblance des données d'apprentissage. L'algorithme EM (Expectation Maximization) [Dempster et al., 1977] est couramment utilisé pour effectuer cette estimation. Les états du modèle sont alignés sur les données d'apprentissage en affectant chaque vecteur à un état grâce à l'algorithme de Viterbi. Et les valeurs des paramètres sont obtenues via les formules de réestimation de Baum-Welsh [Baum et al., 1970].

Le critère du maximum de vraisemblance conduit à l'estimation de modèles représentatifs des données d'apprentissage et des locuteurs présents dans ces données. Cependant il existe des choix et des méthodes d'adaptation qui permettent aux modèles de s'affranchir en partie des données d'apprentissage [Gauvain et Lee, 1994]; [Leggetter et Woodland, 1995]. En particulier, en ce qui concerne l'indépendance vis-à-vis du locuteur, les corpus d'entraînement contiennent les énoncés d'au moins une centaine de locuteurs différents. De plus de meilleurs résultats sont obtenus en modélisant séparément les locutrices et les locuteurs.

7.1.3 Lexique de reconnaissance

Le lexique de reconnaissance (également appelé dictionnaire phonétique, dictionnaire de prononciation, lexique phonétisé ou simplement lexique) est le lien entre la modélisation linguistique et la modélisation acoustique. Il regroupe tous les mots du vocabulaire et leurs transcriptions phonétiques. Il suppose donc le choix préalable des entrées lexicales et du jeu de phonèmes utilisé pour les décrire. Le choix du jeu de phonèmes est lié à la modélisation acoustique et à la langue (par exemple nous utilisons environ 45 phonèmes pour l'anglais, 49 pour l'allemand, 33 pour le français, et seulement 26 pour l'espagnol). Le choix du vocabulaire est lié à la modélisation linguistique. Les dictionnaires de prononciation sont généralement construits manuellement par des experts. La Figure 2-4 représente la chaîne de modélisation d'un mot du vocabulaire : un mot est transcrit phonétiquement, puis chaque phonème est converti en allophone et chaque allophone correspond à un MMC. En pratique, chaque entrée lexicale est décrite à l'aide d'une séquence d'éléments choisis parmi 33 phonèmes, auxquels s'ajoutent 3 symboles spécifiques pour représenter les hésitations, les respirations et les silences.

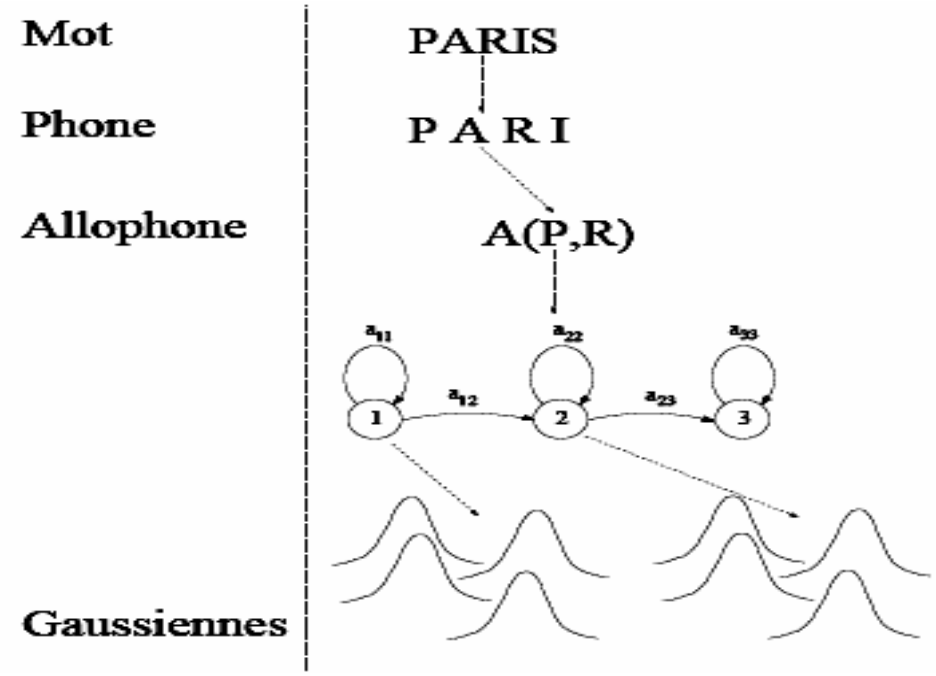


Figure 2-4 : Modélisation d'un mot : un mot est transcrit phonétiquement, puis chaque phonème est converti en allophone et à chaque allophone correspond à un MMC (Figure extraite de [Gauvain et Lamel,02]

7.1.4 Modélisation linguistique

Un modèle de langage tente de capturer les contraintes sémantiques et syntaxiques du langage via l'étude de ses régularités. La finalité est d'estimer la probabilité a priori P(W) de toute séquence de mots extraites du vocabulaire. La plupart des systèmes utilisent des modèles n-gram de mots qui prédisent un mot en fonction des n-1 précédents. En général n varie de 2 à 4, donnant un modèle nommé respectivement bigramme, trigramme et quadrigramme.

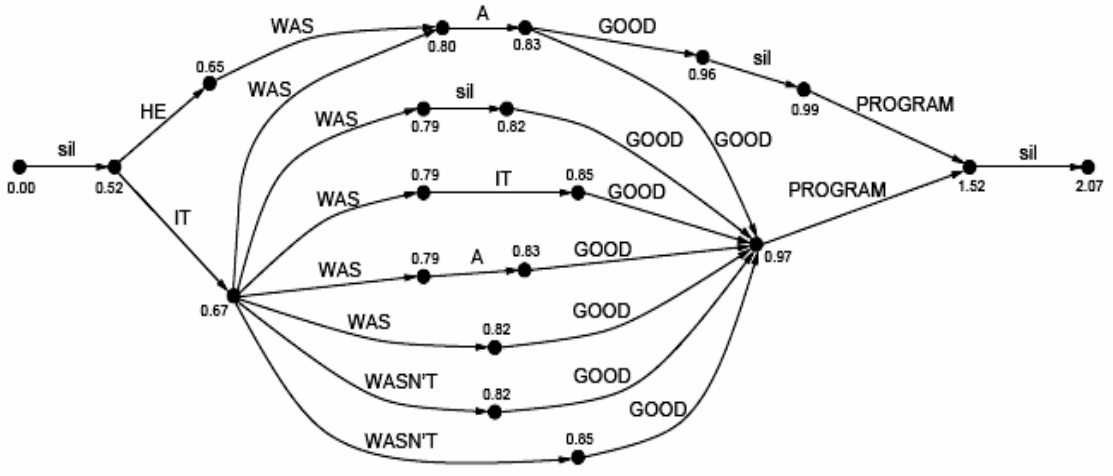


Figure 2-5 : Exemple de treillis de mots. Les scores linguistiques et acoustiques ne sont pas représentés (Figure extraite de [Gauvain et Lamel,02]

7.1.5 Décodeur

Le décodeur a la lourde tâche de déterminer la suite de mots ayant la probabilité la plus élevée à partir d'une séquence de vecteurs acoustiques, étant donné le lexique, les modèles acoustiques et linguistiques. Pour cela, il envisage un certain nombre d'hypothèses parmi lesquelles, il choisit la plus probable. Dans le cadre d'un système de reconnaissance automatique de la parole à grand vocabulaire (typiquement 65 000 mots), il est évidemment impossible d'envisager toutes les hypothèses. La plupart des décodeurs fonctionnent en plusieurs passes ce qui permet de réduire le temps de calcul et les besoins en mémoire en introduisant les connaissances progressivement.

D'une passe à l'autre, l'espace de recherche se transmet via un treillis de mots. Un treillis est un graphe acyclique dirigé (Direct Acyclic Graph) où chaque noeud correspond à un instant donné, et chaque arc est une hypothèse de mot pondérée par un score linguistique et un score acoustique. Ces scores sont estimés à partir des modèles linguistiques et acoustiques. Chaque segment de parole à décoder est donc représenté selon un treillis de mots comme le montre Figure 2-4. Un treillis de mots est une représentation compacte des hypothèses les plus probables qui permet un décodage rapide en plusieurs passes, l'estimation de mesures de confiance et l'application d'autres sources de connaissances. Ainsi il devient très simple de modifier par exemple le modèle de langage en remplaçant simplement les scores linguistiques.

Le décodeur utilisé par la suite procède en trois passes. Tout d'abord un premier treillis de mots est généré avec un modèle de langage trigramme. Ce treillis est utilisé pour adapter les modèles acoustiques. La deuxième passe redéveloppe un nouveau treillis avec les modèles acoustiques adaptés et un modèle de langage bigramme. Enfin, en guise de troisième passe, le treillis est recalculé avec un modèle de langage quadrigramme. Pour les trois passes, trois modèles de langages sont utilisés. Il s'agit sauf spécification contraire, du même modèle de langage quadrigramme, mais restreint selon les besoins aux modèles trigramme et bigramme.

7.2 Système de transcription automatique

La section précédente décrit succinctement comment un système de reconnaissance automatique de la parole décode un segment de parole. Or une émission téléradio diffusée est un flux de données qui n'est pas segmenté et qui ne contient pas uniquement de la parole. Le premier élément d'un système de transcription automatique de documents téléradio diffusés est le « partitionneur » dont le rôle est justement de préparer le flux de données pour l'étape de reconnaissance de la parole. Cette section décrit dans un premier temps le partitionneur et dans un second temps, le principal critère d'évaluation d'un système de transcription automatique.

7.2.1 Le partitionneur

Le partitionneur effectue la segmentation du flux audio en détectant les changements de conditions acoustiques. Chaque segment est classé selon trois catégories : parole, musique, et bruit. Les segments de parole sont ensuite étiquetés en locuteur, bande passante et genre du locuteur. L'étiquetage en locuteur utilise une procédure itérative qui regroupe les segments par locuteur. Le résultat est un ensemble de segments regroupés suivant leurs attributs acoustiques, ce qui permettra par la suite au système de reconnaissance automatique de la parole de choisir les modèles acoustiques les plus appropriés pour décoder chaque segment et d'adapter les modèles entre deux passes.

7.2.2 Taux d'erreur sur les mots

L'évaluation d'un système de reconnaissance automatique de la parole consiste en la comparaison entre la transcription de référence et celle fournie par le système. Trois types d'erreur sont pris en compte. Pour cela, les deux transcriptions sont alignées entre elles et chaque mot de l'une a son correspondant dans l'autre. Lorsque deux correspondants ne sont pas les mêmes, il y a une erreur dite de substitution. Si dans la transcription de référence se trouve un mot n'ayant pas de correspondant dans la transcription automatique, alors cela signifie que le système a oublié ce mot. C'est une élision. Enfin l'erreur symétrique est une insertion.

Le taux d'erreur sur les mots est la mesure la plus employée pour évaluer un système de reconnaissance automatique de la parole que ce soit dans le cadre de la dictée vocale ou de la transcription automatique de document audio. Ce taux se calcule à partir du compte des différentes erreurs comme il suit :

$$\text{Taux d'erreur sur les mots} = \frac{substitution + insertion + élision}{mots.dans.la.référence} \qquad 7.2.2.1$$

Le taux d'erreur sur les mots est très couramment abrégé WER (Word Error Rate). Cette métrique présente certains inconvénients. Le premier est son coût puisqu'il faut pour calculer ce taux, transcrire manuellement et finement l'émission. De plus, les comptes utilisés sont indépendants des mots sur lesquels portent les erreurs. Cette mesure considère donc que les erreurs sont toutes préjudiciables de la même manière, et cela indépendamment du cadre d'application.

7.2.3 Effet des erreurs de reconnaissance

Un sous ensemble de 10 heures du corpus audio a été transcrit manuellement afin d'évaluer les systèmes de reconnaissance. La mesure communément utilisée est le WER (Word Error Rate ou taux d'erreur sur les mots). Le taux d'erreur du système présenté par le LIMSI à TREC-9 est de 20% [Gauvain et al., 00].

Lors des évaluations TREC-8 et TREC-9, la corrélation entre le WER et le MAP (Mean Average Precision) a été étudiée. Dans l'article [Garofolo et al., 98], les auteurs observent une certaine corrélation entre ces deux quantités tout en concluant que « le WER est insuffisant pour prédire les performances en recherche d'information » . Intuitivement cela s'explique puisque le WER accorde une importance égale à tous les mots. Or en recherche d'information, un article n'a pas la même importance qu'un nom propre surtout s'il est filtré par une stoplist. Les auteurs poursuivent donc en

proposant d'autres taux d'erreurs qui prennent en compte l'importance relative des mots. Une erreur de transcription n'a pas le même impact suivant qu'elle porte sur un article indéfini ou sur un nom propre. Ces taux d'erreurs peuvent par exemple être appliqués après une étape de lemmatisation ou après un filtrage de certains mots par une stoplist. Mais la mesure ayant le meilleur coefficient de corrélation avec la précision moyenne est le taux d'erreur sur les entités nommées [Garofolo et al., 98].

8. Conclusion

L'indexation automatique du signal audio a vocation à extraire d'un enregistrement sonore une représentation symbolique. Cette représentation est organisée par catégories de caractères suivant une structuration qui peut être générale ou détaillée. Dans le cas de la musique, par exemple, sont visés des concepts tels que le rythme, la mélodie, ou encore l'instrumentation, ceux ci peuvent prendre une forme hautement structurée : la partition musicale.

Les applications de l'indexation automatique ne se limitent pas à l'extraction automatique de partitions. On retrouve parmi les plus populaires, des applications s'articulant autour de la recherche, la navigation et l'organisation des bases de données sonores : on parle de recherche par le contenu. En effet, l'obtention à partir des signaux, de représentations pertinentes permet d'envisager de retrouver, dans de grandes bases de données, les sons "ressemblant" `a un exemple de référence- c'est la recherche par similarité- et plus généralement les sons répondant aux critères définis par l'utilisateur.

Derrière cet objectif d'indexation se profile un processus fondamental : celui qui organise les événements sonores en catégories.

De ce fait, nous considérons les différentes taches d'indexation comme pouvant être approchées suivant un même paradigme qui consiste à les envisager comme un problème de classification automatique, La Figure 2-6 présente un exemple d'une telle réalisation.

Ce paradigme permet de résoudre plusieurs taches clés de l'indexation audio, comme l'indexation selon l'un des critères suivant :

- La décomposition du document en segment parole / music ou bruit.
- Classification de la musique (classique, jazz, blues, …)
- Classification des bruits, …
- L'identification de la langue, il s'agit de classer les documents selon la langue employé dans le discours.
- Recherche de locuteurs, il s'agit de définir où débute et où se termine l'intervention d'un locuteur.
- Indexation par locuteur, il s'agit d'organiser la base de données selon les locuteurs de la base.
- Détection de mots clefs,
- Topic detection and tracking (TDT)

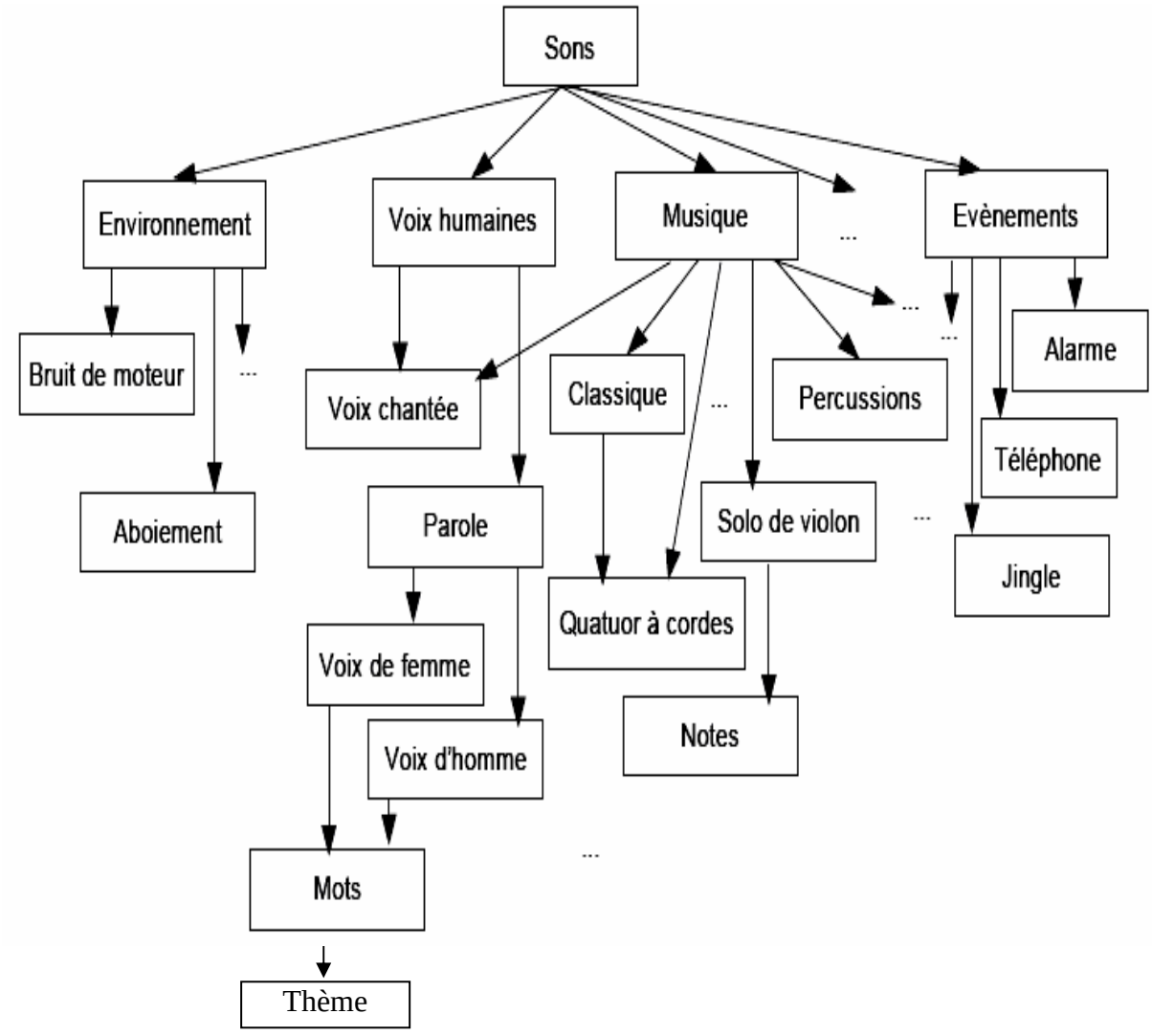


Figure 2-6 : Exemple de schéma de classification audio général.

Notons que les frontières entre classes ne sont pas toujours définies de façon univoque. Dans la Figure 2-6, on peut voir, par exemple, que la classe “musique” et la classe “voix humaines” sont recouvrantes puisque la classe “voix chantée” appartient à ces deux catégories. La définition de classes disjointes peut donc s’avérer délicat dans des contextes d’application particuliers.

Problématique

La sonelgaz société algérienne de l'électricité et du gaz, société étatique qui s'occupe de la production, transport et la distribution de l'électricité et gaz naturel

Il est apportant de noter que la distribution de l'électricité et de gaz est assurée par des directions régionales (47 directions régionales) sur le territoire national, dont le traitement de différentes réclamation se fait via la 'T.I.A' (traitement informatique des appels)

Le TIA est assuré au niveau des agences commerciales relevant dans les directions régionales, et ce par un agent qui assure la tache de l'écoute via la ligne téléphonique de la clientèle et la saisie sur micros des différentes réclamations et doléances, chose qui est considérée coûteuse sur le plan financier, et nécessitant la présence de l'agent opérateur le long des heurs de réception, et vu le nombre très important reçu quotidiennement au niveau de la cellule T.I.A fait apparaître des difficulté dans le traitement de ces appels tel que le problème de l'identification et la classification des appels reçus , considéré comme le problème majeur de cette application

L'analyse des différentes réclamations et doléances saisie fait ressortir les thèmes d'interventions suivants : branchement électricité, ou gaz, facture non reçu, duplicata, contestation facture.

Après l'identification du thème de l'appel par l'agent opérateur, il transfère les messages ver les services concernés pour traitement.

Les appels téléphoniques ne sont pas seulement nombreux mais aussi parlés en plusieurs langues. Divers appels téléphonique de même thème sont parlés en divers langues ; Cela peut produire des ensembles immenses des appels téléphonique parlés dans des langues différentes qui doivent être exploités. Afin de faciliter la recherche et l'accès, il est nécessaire de grouper les appels. De plus, les groupes doivent être étiquetés pour guider les utilisateurs dans le choix de ceux qui conviennent le plus à ses besoins.

Pour résoudre le problème, une méthode est présentée pour regrouper les signaux audio dans des groupes thématiques cohérents. Dans ce document, nous présentons une méthode qui fait appel aux cartes auto organisatrices SOM (Self Organizing Map).

Avant d'appliquer l'algorithme SOM, Quatre aspects importants peuvent affecter l'efficacité de la méthode:

1. La méthode de transcription.
2. La méthode de représentation des documents.
3. L'algorithme de regroupement.
4. La méthode de représentation des résultats.

La méthode de transcription, Nous profitons de la taille de la base de données, les documents audio sont décodés par la reconnaissance automatique de la parole on choisissons des meilleurs termes d'indexation pour chaque document. La méthode de regroupement doit être suffisamment puissante pour pouvoir grouper un grand ensemble de documents et permettre à l'utilisateur de visualiser des résultats. Les cartes auto organisatrices - SOM - ont été choisies pour cette raison. La méthode de représentation de documents doit produire des vecteurs de documents de faible dimension et posséder la capacité d'indexer des documents multilingues. C'est la raison pour laquelle la méthode basée sur une ontologie a été choisie. La méthode qui combine la SOM et une ontologie peut aider à réduire les dimensions et produire des résultats encourageants avec des documents audio multilingues.

Chapitre 3:

Outils utilisés

1. Introduction

Soit une base de donné audio qui englobe un ensembles des appels téléphoniques, que l'on souhaite regrouper de sorte que les appels similaires appartiennent au même groupe. Une méthode qui peut résoudre ce problème doit comprendre les quatre éléments suivants :

1. La méthode de transcription.
2. La méthode de représentation des documents.
3. L'algorithme de regroupement.
4. La méthode de représentation des résultats.

Il y a plusieurs différentes propositions qui donnent la solution au problème. Cependant, dans ce chapitre, on va présente l'état de l'art des méthodes qui résolvent le problème.

2. Méthodes de transcription

La base du système d'indexation par thème est la reconnaissance de la parole, les documents créés pour l'indexation sont formés seulement du texte « OUTPUTED » par le système de reconnaissance de la parole , les documents audio sont transcris par le système ASR (Automatic Speech Recognition) sous forme des documents texte.

La façon la plus triviale de résoudre le problème de l'indexation audio est d'utiliser un reconnaisseur de parole puissant pour reconnaître le texte dit et d'ensuite analyser celui ci à l'aide d'un éditeur de texte. Dans le cadre du projet THISL (Thematic Indexing of Spoken Language), ils ont basé sur l'intégration du système Large Vocabulary Continuous Speech Recognition (LVCSR) avec les technologies de la recherche de l'information (Information Retrieval (IR)) et le traitement des langues naturelles (Natural Language Processing NLP).

Plusieurs discours de reconnaissance ont été créés dans le cadre de la THISL projet portant sur l'anglais britannique [Robinson et al., 99], l'anglais américain [Renals et al., 98, Abberley et al., 99b], et le français [Andersen, 98]. Pour minimiser le taux (WER) de mots incorrectement reconnus par rapport à un texte de référence, l'évaluation du TREC SDR (Spoken Document Retrieval) a permis de développer une méthode à double fenêtre temporelle glissante pour l'indexation des transcriptions issues du système de reconnaissance. La segmentation et l'identification en thèmes sont deux problèmes indissociables; on peut donc améliorer significativement le processus d'indexation en évitant les erreurs dues à un décodage en deux étapes: segmentation puis identification. Le nouveau moteur de recherche a été évalué en utilisant les données de l'évaluation SDR de TREC-9 (600h heures d'audio) et les résultats ont été comparés à l'approche en deux étapes. Les résultats en terme de précision moyenne montrent clairement l'intérêt de cette technique d'indexation par fenêtre glissante : 52.3% contre 33.3%, à comparer avec 59.6% pour une segmentation manuelle.

Les différentes évaluations TREC ont montré que le taux d'erreur dans les transcriptions n'affectait pas trop les performances en recherche documentaire (une chute de 10% de la précision pour un WER de 25% et une chute de 15% pour un WER de 50% [Allan 02], sachant que les performances aux récentes évaluations Rich Transcription 2003 (RT'03) sont aux alentours d'un WER de 10,5%).

L'Abbot LVCSR c'est l'une des applications du TREC SDR qui [Robinson et al. ,96] fait appel à un système hybride de réseaux de neurones artificiels / modèle de Markov caché (ANN / HMM) acoustique et un modèle soutenu hors modèle trigramme langue. Au temps de la reconnaissance, la reconnaissance fait appel à un vocabulaire d'environ 65000 mots. Sur un PC moderne il tourne dans deux ou trois fois en temps réel, la production d'une meilleure transcription, un mot graphique (contenant d'autres hypothèses possibles) et mot de téléphone et niveau des mesures de confiance.

Le système THISL IR (indexation thématique de la langue parlée) IR (Information Retrieval) [Robinson et al. ,96,98,99] utilisé comme base de référence dans les expériences LSI, ou tous les mots décodés dans un document sont utilisés comme index de termes. Le plus souvent, les mots outils sont filtrés loin, d'autres sont à éviter comme les formes fléchies.

3. Méthodes de représentation de documents

Les documents ne peuvent pas être interprétés directement par une méthode de regroupement. À cause de cela, une procédure d'indexation qui fait correspondre au texte une représentation compacte de son contenu doit être appliquée. Il y a plusieurs choix pour représenter des documents audio transcris. La solution la plus souvent utilisée est de se baser sur des vecteurs de caractéristiques des documents [Fabrizio, 02]. En général, les composants des vecteurs décrivent le contenu des documents. Le modèle de l'espace vectoriel [Salton, 75] sert de base à cette représentation. Les différentes méthodes décrivent un document en se basant sur différentes caractéristiques et critères de sorte que l'objectif final de décrire au mieux le sens du document tout en assurant toujours une dimensionnalité acceptable de son vecteur de représentation soit respecté. Un des plus grands défis dans la fouille de données en général et dans le domaine du regroupement de documents numériques par les SOM en particulier est la grande dimension des vecteurs lors de traitement d'un ensemble de documents.

Un document est caractérisé par un ensemble d'éléments sémantiques permettant de le représenter sous forme de vecteur. Le nombre de dimensions du vecteur d'un document est différent de celui du vecteur d'un autre document. Cependant, pour grouper un ensemble de documents, il est nécessaire que les documents aient des représentations comparables. Or, un élément sémantique peut apparaître dans un document mais pas dans un autre. Pour résoudre ce problème, un dictionnaire doit être construit. Ce dictionnaire contient tous les éléments sémantiques trouvés dans l'ensemble des documents de sorte qu'aucun d'eux n'apparaisse plus d'une fois dans le dictionnaire. Un vecteur basé sur ce dictionnaire sera alors produit pour chaque document. Le vecteur produit a la même dimension que le dictionnaire et les éléments sémantiques y Figurent dans le même ordre, un élément sémantique peut être un terme, un mot, une catégorie de mots ou un concept. Figure 3-1.

L'importance d'un élément sémantique dans un document est mesurée en se basant sur son nombre d'occurrences dans ce document. Si un élément n'est pas présent dans le document, il aura un poids égal à 0 (cela signifie que cet élément n'a aucune importance du tout dans le document). La même procédure est appliquée à tous les éléments sémantiques du document. Après que tous les documents aient été indexés, nous avons un ensemble de vecteurs représentant les documents. Il peut y avoir d'autres tâches de prétraitement comme l'application de la mesure de EFIDF, normalisation ... à accomplir avant que l'ensemble de vecteurs puisse être employé comme entrées pour l'algorithme de regroupement.

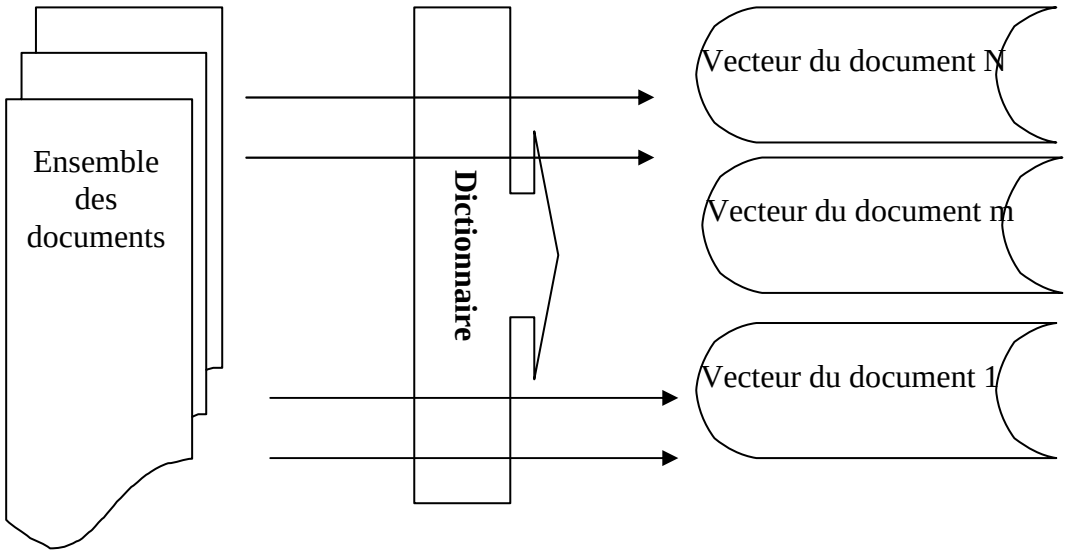


Figure 3-1: Représentation de chaque document sous forme de vecteur d'éléments sémantiques.

Pour calculer l'importance d'un élément sémantique, il y a plusieurs méthodes. Les méthodes les plus couramment utilisées sont les suivantes (voir): EF et EFIDF:

• EF (element frequency): Fréquence de l'élément sémantique, Le nombre d'occurrences d'un élément sémantique dans un document sera employé pour représenter son importance. Plus la valeur ef est grande, plus l'élément est important. Il y a des cas où un élément a une valeur ef élevée pour tous documents de la collection. Il est clair que cet élément ne peut permettre de distinguer différents groupes de documents. C'est un inconvénient de la méthode.

• efidf: Element Frequency-Inverse Document Frequency. Cette méthode a été proposée pour surmonter l'inconvénient de la méthode ef: un élément sémantique qui apparaît dans tous les documents doit avoir un poids petit.

$$efidf(i,j) = ef(i,j) * \log\left(\frac{N}{df(j)}\right) \qquad 3.1$$

Où i est l'index du document courant, j est l'index de l'élément sémantique courant, ef(i,j) est la valeur ef de l'élément j dans le document i, df(j) est le nombre de documents où l'élément j apparaît. Si un élément sémantique est présent dans plusieurs documents, sa valeur df(j) sera grande. En conséquence, sa valeur efidf sera petite.

3.1 Modèle de l'espace vectoriel

Le modèle de l'espace vectoriel (Vector Space Model - VSM) sert de base à la représentation des données textuelles par des vecteurs dans l'espace euclidien. Selon [Salton, 75], un document est représenté par un vecteur des termes. Soit on donne un poids au terme, soit on l'enregistre simplement comme « présent »/« non présent» dans le document courant en assignant la valeur 1 s'il est présent et 0 autrement. Un terme peut être un mot simple ou un mot composé. L'extraction des termes s'effectue après élimination des mots outils (les mots outils incluent des articles, des conjonctions de coordination, etc ..). Les autres mots sont alors indexés. Nous appelons ceci la pleine représentation des documents. Dans cette méthode, les éléments sémantiques sont des termes qui apparaissent dans les documents. Cette approche est très courante.

Comment un terme peut il être extrait à partir d'un document donné? Une solution commune pour ceci est d'employer un dictionnaire prédéfini des termes et un autre dictionnaire prédéfini des mots outils. Les mots outils se trouvant dans les documents ne sont pas pris en compte pour construire la représentation. Un nouveau dictionnaire est construit à partir de l'intersection des termes se trouvant dans les documents et appartenant au dictionnaire de termes. Ce dictionnaire a une taille égale ou inférieure à celle du dictionnaire prédéfini de termes. Figure 3-2.

Sachant qu'un document peut contenir beaucoup de termes, l'ensemble de tous les documents peut produire un dictionnaire de grande dimension. Pour un grand ensemble de textes, il est impossible d'utiliser la pleine représentation. D'ailleurs, elle ne prend pas en compte le contexte de chaque mot dans les documents et pour cette raison, la polysémie et la synonymie ne sont pas traitées.

Donc le choix des termes d'indexation c'est important et l'indexation se typifie sur la base de deux critères [Lewis, 92b]

- Choix des termes: par un humain ou par l'ordinateur
- Nombre de termes: Indéfini (ils peuvent toujours s'additionner) ou déterminé, par avance ou en cours d'indexation

Statistiquement, on obtient une efficacité optimale avec un nombre de termes limité; des fréquences faibles; pas de redondances; et peu de bruit distorsions ou inconsistances dans les poids.

La polysémie, la synonymie et le champs de validité des termes (termes techniques par exemple) compliquent la définition des algorithmes en aval de la représentation [Lewis, 92b]. L'approche naïve revient à des termes et des mots (avec des poids binaires ou non). Cette approche simple est très efficace: tant en IR [Salton et Buckley, 87] qu'en catégorisation, il a été montré que les représentation les plus sophistiquées ne donnent pas nécessairement de meilleurs résultats [Dumais et al., 98]; [Cohen, 95]; [Siolas et D'alche Buc, 00].

On a essayé d'utiliser des parties de texte comme termes d'indexation. La notion de « partie de texte » peut être définie soit syntaxiquement [Lewis, 92a], soit statistiquement (ensemble des mots dont le groupement est statistiquement significatif, plutôt que partie de texte formant une cohérence syntaxique [Caropreso et al., 01]). Les résultats, en 2002, n'étaient pas concluants [Sebastiani, 02], mais les recherches

dans cette direction se poursuivent [Metzler et Croft, 05]; [Metzler et Croft, 06]; [Alvarez et al., 04].

Dans le même ordre d'idées, la reconnaissance d'entités textuelles peut être employée pour raffiner l'indexation.

Sachant que l'utilisation des vecteurs pour représenter des documents sert également de base à d'autres méthodes.

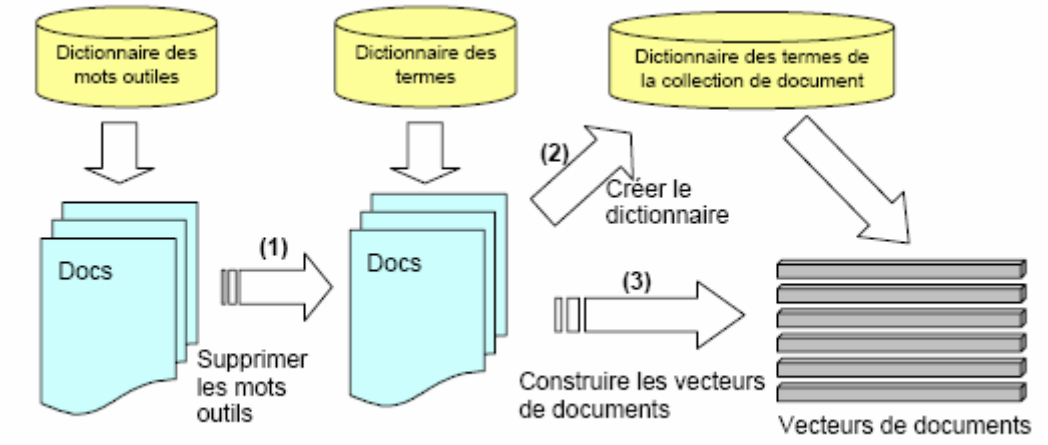


Figure 3-2 : Modèle de l'espace vectoriel

L'algorithme de représentation des documents selon ce modèle se compose des trois étapes suivantes :

- Première étape : On supprime les mots outils dans la collection de documents à l'aide d'un dictionnaire de mots outils. Ce dictionnaire est prédéfini. Les mots outils sont les mots vides qui ne contiennent aucun contenu (par exemple : le, la, les, à, de,...) ou les mots de liaison (par exemple : ce, cette, enfin, cependant, ...), etc.

- Deuxième étape : Supposons qu'on a un dictionnaire des termes. À parti de ce dictionnaire et l'ensemble de documents obtenu après l'étape au dessus, on construit un nouveau dictionnaire des termes qui contient tous les termes apparus dans ces données textuelles. La taille de ce nouveau dictionnaire est inférieure ou égale la taille du dictionnaire précédent.

- Dernière étape : En utilisant le nouveau dictionnaire de terme, on calcule le poids de chaque terme dans les documents et construit les vecteurs qui les représentent. La dimension du vecteur représentant des documents est égale au nombre de termes dans le dictionnaire utilisé.

On appelle cette approche est la pleine représentation. Ce modèle est très courant. Il y a beaucoup de méthodes est développées à base de ce modèle. Pourtant, il a quelques inconvénients. Premièrement, chaque document peut contenir beaucoup de termes. Le dictionnaire de termes qui est produit à partir de l'ensemble des documents peut avoir une taille très grande. Le vecteur représentant des documents a donc une dimension très grande. Il est difficile de traiter un ensemble des grands vecteurs quand le nombre de vecteurs est important et on perd beaucoup de temps pour faire ça. En outre, cette approche ne s'intéresse pas au contexte et à la sémantique de chaque mot dans le document.

3.2 L'analyse en Composantes Principales

C'est la méthode d'analyse factorielle la plus utilisée [Bouroche et Saporta, 80][Jolliffe, 86][Lebart et al.,00]. L'ACP consiste à calculer un nombre réduit de nouvelles dimensions, qui sont des combinaisons linéaires des dimensions originelles des données (c'est à dire des traits descriptifs). Ces nouvelles dimensions sont non corrélées et expriment le maximum de variance des données (en partant de données centrées sur la moyenne). Les nouveaux axes sont les vecteurs propres, ordonnés par valeurs propres décroissantes, de la matrice de covariance des données.

Autrement dit ce sont les principaux axes de dispersion du nuage de données, en ordre d'importance décroissante ; les valeurs propres correspondantes indiquent la part de variance exprimée par chaque axe. Les premiers axes rendent donc généralement compte de la plus grande partie de la variance. Les composantes principales sont les nouvelles valeurs des données sur chaque axe ainsi obtenu.

Cette méthode peut jouer un double rôle de compression des données et d'outil d'exploration dans des domaines fortement multidimensionnels. En effet les axes principaux ainsi calculés permettent à la fois une réduction des données et une interprétation plus facile du domaine traité, car les nouvelles dimensions sont souvent très significatives. Cela peut notamment se révéler intéressant pour l'analyse de données en langage naturel [Lebart et Salem, 94].

Le problème est que l'ACP demande le calcul préalable de la matrice carrée de covariance des données, qui est de taille n2 pour des vecteurs de dimension n. Cette matrice est déjà coûteuse à calculer, et sa taille et son traitement deviennent prohibitifs en grande dimension. Ainsi des données de dimension 1000 donneraient lieu à une matrice de un million d'éléments !

L'ACP est donc difficile ou impossible à utiliser sur des vecteurs de documents textuels, dont on a vu qu'ils pouvaient comporter des milliers de traits. Diverses approches ont alors été proposées pour réduire la dimension des représentations vectorielles de textes.

On peut citer notamment la méthode appelée Latent Semantic Indexing (LSI) ou Latent Semantic Analysis (LSA), qui n'utilise pas la matrice de covariance, mais extrait de nouveaux axes directement de la matrice document terme (Deerwester et al. 90). L'approche consiste à effectuer une décomposition en valeurs singulières de la matrice des données pour trouver les nouveaux axes [Lebart et al., 00]. Cette technique peut être considérée comme une généralisation de l'extraction des vecteurs propres caractéristiques de l'ACP, généralisation permettant de traiter des matrices rectangulaires et des données non centrées.

Comme le nombre de documents est souvent plus petit que le nombre de termes, cette matrice rectangulaire est moins encombrante et moins coûteuse que la matrice de covariance. La méthode LSI semble donner de bons résultats en pratique pour l'indexation et la recherche, mais contrairement à l'ACP, la signification théorique des nouveaux axes est loin d'être claire et ceux ci sont peu intuitifs. D'autre part les coûts de calcul demeurent importants, car on manipule encore des matrices de grande taille.

3.3 L'indexation sémantique latente (ou Latent Semantic Indexing -LSI)

Cette méthode est développée à base du modèle de l'espace vectoriel. L'objectif de la méthode LSI [Deerwester et al., 90] est de réduire la dimension des représentations de documents obtenu par la transcription des fichiers audio. Premièrement, chaque fichier audio transcrit est également entièrement représenté par un vecteur. Lorsque tous les documents sont caractérisés par des vecteurs, on obtient une matrice dont les colonnes sont des vecteurs de document et où chaque rangée correspond à un terme. Souvent, un terme n'apparaît que dans quelques documents et pour cette raison il y a beaucoup de composants ayant la valeur 0 dans la matrice de sorte que la matrice est très clairsemée (creuse). Figure 3-3.

Ensuite, la décomposition en valeurs singulières (SVD - Singular Value Decomposition) est appliquée pour réduire la taille des vecteurs colonnes de sorte qu'enfin, les vecteurs résultants aient une dimension faible.

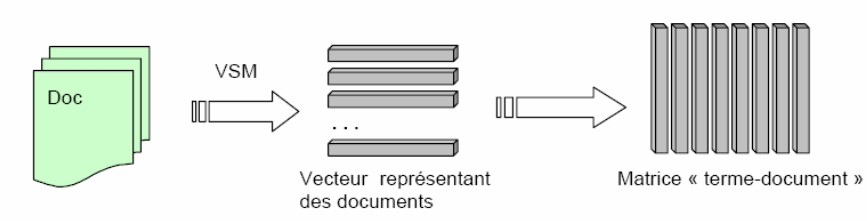


Figure3-3 : Génération la matrice « terme-document »

Au début, nous avons K documents et N termes pour une collection de documents. Les documents sont représentés (en utilisant le modèle de l'espace vectoriel) par des vecteurs de K dimensions de sorte que nous obtenons une matrice de taille KxN appelée la matrice «termes-documents». Supposons que l'on doit ramener le nombre de dimensions de ces vecteurs à K' dimensions (où K'< K). L'algorithme SVD est appliqué de sorte qu'à partir de la matrice des termes-documents que l'on appelle la matrice A, on obtienne le produit de trois autres matrices U, S, et V.

$$A_{K.N} = U_{KK'} * S_{K'K'} * V^{T}_{NK'} \text{ (avec K'<K).} \qquad 3.3.1$$

Avec K'< K, cette décomposition revient à grouper plusieurs termes dans un concept et à représenter les documents par des vecteurs de concepts au lieu de vecteurs de termes. Parmi des matrices résultat, U et V sont des matrices avec des colonnes ortho normales. S est une matrice diagonale des valeurs singulières triées par ordre décroissant. Pour construire la SVD, U est une matrice des vecteurs propres dérivés de K*KT tandis que V est une matrice des vecteurs propres dérivés de KT*K. La matrice V contenant N colonnes et K' rangées peut être vue comme une matrice concepts documents. Chaque colonne de cette matrice est un vecteur des poids des «concepts» et peut être considérée comme un vecteur de document. En conséquence, une réduction de dimensions a été effectuée. Figure 3-4

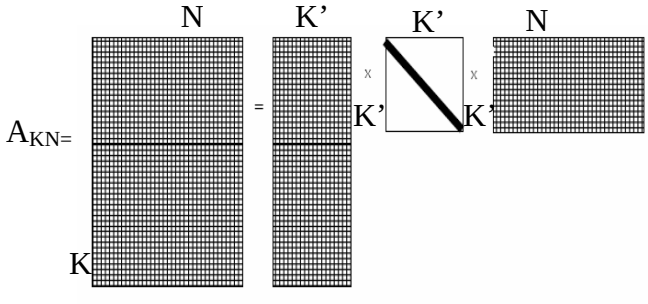


Figure 3-4 : Une décomposition en valeurs singulières

Cependant, il faut noter que cette méthode nécessite beaucoup de calculs et cela peut prendre énormément de temps pour une grande collection de documents. D'une part, les «concepts» que nous avons mentionnés ne sont pas des concepts sémantiques. Les matrices U, S, et V sont justes des matrices de nombres et aucune signification propre n'est attachée au concept auquel un terme appartient.

3.4 Projection aléatoire (ou Random Mapping) dans LSI

Pour une grande collection de documents, la dimension des vecteurs de document joue un rôle très important et une petite réduction de cette dimension peut induire une grande réduction de la durée de traitement dans la méthode LSI, Pour cette raison, S. Kaski dans [Kaski, 99] a proposé la projection aléatoire de sorte que la réduction de la précision soit acceptable lorsque le nombre de dimensions diminue considérablement. Au début, un document est représenté dans un vecteur des occurrences de termes. Ce vecteur est alors multiplié avec une matrice dont les composants sont aléatoirement produits de sorte que le vecteur résultat ait une plus petite dimension.

Si di est le vecteur de document à M dimensions, R est une matrice de taille MxP dont les composants sont des valeurs aléatoires et dont la longueur des vecteurs est l'unité après normalisation et P<<M. di est alors remplacé par d', à P dimensions:

$$d_i^{'} = R.d_i \qquad 3.4.1$$

Naturellement, plus P est petit, plus la durée de traitement est réduite. Mais on ne peut pas choisir une valeur de P=10, par exemple, alors que M=5000. Ainsi quelle valeur de P est acceptable ? Il est possible d'utiliser les similitudes mutuelles des vecteurs de données. Pour évaluer cela, il faut mesurer la similitude de deux vecteurs de documents originaux di et dj et celle de deux vecteurs réduits par correspondance aléatoire $d_i^{'}$ et $d_j^{'}$. Supposons que ces vecteurs sont déjà normalisés dans des vecteurs unité de sorte que le cosinus puisse être calculé par le produit intérieur de deux vecteurs de chaque couple.

$$d'^{T}_{i}.d'_{j} = d_i^T.R^T.R.d_j \qquad 3.4.2$$

La matrice RT.R est alors récrite: RT.R = I + ε

La taille de RT est PxM, de sorte que RT.R est une matrice de taille PxP. I et ε sont deux matrices ayant la forme suivante:

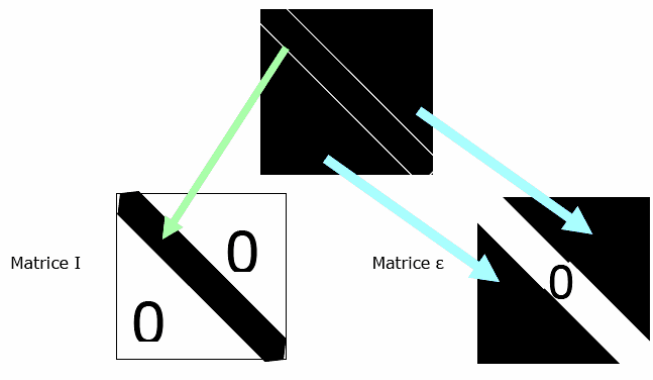


Figure 3-5 : Projection aléatoire

Formes d'I et de ε : I a des valeurs 0 pour les composants qui ne sont pas sur sa diagonale. Les composants sur la diagonale d'I correspondent à ceux de la diagonale de RT.R. (Figure 3-5), ε a des valeurs 0 pour les composantes situées sur sa diagonale. Ses composants qui ne sont pas sur la diagonale sont ceux qui ne sont pas sur la diagonale de RT.R

Si tous les composants de matrice ε ont la valeur 0, la similitude est préservée par correspondance aléatoire. Cependant, avec P<M, ils ne sont jamais égaux à zéro mais ont de petites valeurs.

Dans [Kaski, 99], on nous montre que la correspondance aléatoire peut donner un bon résultat par rapport à la pleine représentation des documents si la dimension des vecteurs est d'environ cent ou plus tandis que le coût de calcul et la durée de traitement sont sensiblement réduits.

Frieze, Karman et Vempala [Frieze et al., 98] proposent une autre façon d'accélérer LSI. Ils donnent des algorithmes rapides pour trouver des approximations de faible dimension pour une matrice A (m X n) . Ils calculer une approximation de décomposition en valeurs singulières à partir d'une projection aléatoire de la matrice A. Pour tout k,δ,Є, leur algorithme de Monte Carlo trouve la description d'une matrice D de rang au plus k de sorte que

$$\|A - D\|_F \leq \|A - A_k\|_F + \varepsilon \|A\|_F \qquad 3.4.3$$

Détient avec une probabilité d'au moins 1 - δ. ($\|A\|_F^2$ est la somme des carrés de toutes les entrées de la matrice, et Ak est la matrice de dimension K rapprochement avec cette mesure.

4. Méthodes de regroupement

Le «regroupement» est une division de données en groupes d'objets similaires [Pavel, 02]. Chaque groupe comprend des objets qui sont similaires entre eux et différents des objets appartenant aux autres groupes. Le regroupement de textes envisage les documents comme des objets. Les documents sont regroupés dans des groupes de telle sorte que les documents qui appartiennent au même groupe sont très similaires les uns aux autres et très différents des autres documents.

De point de vue d'apprentissage automatique, un regroupement est une méthode d'apprentissage non supervisé, c.à.d, les groupes d'objets résultant ne sont pas prédéfinis [Yong, 05]. Les relations cachées entre les documents sont détectées pendant le processus d'apprentissage.

Plusieurs méthodes peuvent être utilisées pour regrouper un ensemble de documents. Selon [Pavel, 02] et [Yong, 05], les méthodes peuvent être classées en diverses catégories:

4.1 Regroupement hiérarchique:

Cette méthode génère un arbre hiérarchique de groupes appelés dendrogramme [Lebart et al., 82] . Il y a deux manières de construire l'arbre: à partir des éléments ou à partir de l'ensemble de tous les éléments. Si on se base sur les éléments, chaque document est au début mis dans un groupe et un groupe ne contient qu'un document. Puis, les deux groupes les plus similaires sont fusionnés pour former un nouveau groupe. Ce processus se répète itérativement jusqu'à ce qu'une certaine condition d'arrêt soit satisfaite. Une méthode qui fonctionne de cette manière est appelée «regroupement agglomératif ». Par contre, si on se base sur l'ensemble de documents, la méthode est appelée « regroupement par division ». Au début du processus de regroupement par division, il n'y a qu'un groupe de tous les documents. Le groupe est divisé en deux sous groupes lors de l'itération suivante. Le processus continue jusqu'à ce que la condition d'arrêt soit satisfaite. La similarité entre deux documents se base sur la distance entre ces documents.

4.2 Regroupement basé sur une partition:

Un document est mis dans un groupe parmi un nombre fixe de groupes. La méthode des K-moyens est un exemple de ce type. Le nombre de groupes est prédéfini. Un document est mis dans un groupe si la distance entre le vecteur de document et le centre du groupe est la plus petite en comparaison avec les distances entre le vecteur et les centres des autres groupes. Il y a deux types de méthodes de ce genre: le «crisp c1ustering» et le «fuzzy c1ustering».

4.3 Méthodes basées sur la densité ou une grille:

Si une méthode se base sur la densité, l'espace euclidien est divisé en un ensemble des composantes jointes. Trois concepts sont importants pour implémenter cette idée: densité, connectivité et frontière. Un groupe est une composante dense jointe. Pour les méthodes basées sur grille, quatre pas sont nécessaires. L'espace de données est d'abord divisé en cellules. Puis la procédure de regroupement est appliquée aux cellules au lieu des données. Les groupes finaux se basent sur l'appartenance de chaque donnée à chaque cellule et les résultats du processus de regroupement de cellules.

4.4 Méthodes basées sur un modèle:

Selon ces méthodes, l'appartenance d'un document à un groupe suit une distribution de probabilités. La carte auto organisatrice est un représentant de ce type de méthode.

Les cartes auto organisatrices de Kohonen (« Kohonen Self-Organising Maps », SOMs) sont des réseaux de neurones non supervisés qui permettent d'ordonner et de classifier des échantillons en fonction de leur similarité. La méthode offre une alternative à la classification hiérarchique unidimensionnelle et aux méthodes d'ordination en réduisant les relations multidimensionnelles à deux dimensions (axes), ce qui facilite la classification et l'interprétation.

Pour choisir une des méthodes présentées précédemment il faut respecter les points suivants :

- La méthode doit être assez puissante pour traiter un grand ensemble de documents.

- Il doit être possible de visualiser les résultats du regroupement pour que l'utilisateur puisse les exploiter.

Les cartes auto organisatrices SOM - est une des méthodes basées sur un modèle. La SOM [Kohonen, 82] a été utilisée dans le projet WEBSOM pour regrouper un très grand ensemble de documents. Ce travail est décrit dans [Kohonen et al., 00]. Les résultats obtenus par la méthode WEBSOM sont très encourageants. De plus, concernant la visualisation des résultats, la SOM peut représenter les groupes sur une carte. Plusieurs outils de visualisation et d'étiquetage des résultats ont été proposés pour aider l'utilisateur à facilement exploiter l'ensemble des documents.

4.5 Les cartes auto organisatrices SOM :

Jusqu'à maintenant, beaucoup de recherches ont été réalisées sur l'application de la SOM pour le regroupement de documents. Le modèle classique de SOM, qui a été à l'origine proposé par Tuevo Kohonen, a été employé dans beaucoup d'expériences récentes. En outre, il y a également quelques autres tendances de modification de structure de la carte pour réduire certaines faiblesses de la SOM classique ou pour renforcer certains aspects positifs. Nous revoyons dans cette section quelques représentants de ces tendances.

4.5.1 SOM classique

Nous pouvons constater qu'il y a beaucoup de recherches récentes qui utilisent la SOM classique comme méthode de regroupement de textes. Tuevo Kohonen et ses collègues ont également développé une méthode de regroupement appelée WEBSOM, qui emploie la SOM comme noyau. Cette méthode a été construite depuis 1995 mais elle a été en permanence améliorée, de sorte qu'en 2000 elle était citée sur leur site Web http://websom.hut.fi/websom/. Avec «la plus grande collection de documents organisée sur une carte simple composée d'environ 7 millions de résumés». D'autres informations sur cette recherche très intéressante peuvent être vues en [Kohonen et al., 00].

Dans beaucoup de recherches d'autres groupes, la SOM classique est employée pour démontrer un certain aspect des recherches. Par exemple, en [Ahmad et al., 01], une nouvelle méthode pour produire des ensembles de caractéristiques de textes qui seront groupés est appliquée avant l'utilisation de la SOM. Dans [Rauber et Merkl, 01], l'étiquetage automatique des groupes de SOM est discuté. [Ultsch, 03] se concentre sur la visualisation de données après le regroupement de documents. L'abondance des recherches et des applications qui emploient la SOM classique montre que cet algorithme est très puissant.

4.5.2. SOM de taille flottante

Avec la SOM, nous devons décider les dimensions de la carte à l'avance et sa taille ne sera pas changée pendant le processus d'apprentissage. Il est très difficile de décider à l'avance de la meilleure taille d'une carte convenant à un ensemble d'entrées. Même lorsqu'on choisit une taille fixe, il est difficile d'expliquer la raison de ce choix. Pour cette raison, des modifications ont été proposées pour ajuster la taille de la carte pendant le processus d'apprentissage.

Beaucoup de recherches ont été faites pour faire la SOM classique devenir la «SOM augmentée», la «SOM adaptative», la «SOM dynamique», la «SOM croissante» ou encore différents types de cartes plates. Dans [Fritzke, 91], les "Growing Cell Structures" (GCS) sont d'abord construites en commençant par une carte de petite taille, de 4 cellules par exemple. Pendant le processus d'apprentissage, de nouvelles cellules sont ajoutées à la carte. Cet ajout est basé sur la distribution de probabilité des vecteurs d'entrée dans l'espace de données. La suppression d'une cellule est également considérée quand la cellule n'est pas choisie pendant longtemps comme gagnante. Cela signifie que cette cellule se trouve à une position dans l'espace de données où la densité de probabilité est nulle. Comme indiqué en [Fritzke, 91], la carte résultant de ce processus a deux propriétés: la topologie est préservée, comme dans la SOM classique, et de plus, la distribution est préservée. On dit que les structures obtenues reflètent les problèmes originaux, càd, «la structure de cellules est dépendante du problème». Dans [Merkl, 97], les cartes de grille croissantes par accroissement sont proposées pour résoudre le problème de carte de taille fixe de la SOM classique, et le problème de visualisation de données qui a été mal résolu dans les précédentes recherches comme les "Growing Cell Structures". Normalement, sur les cartes, il y a des noeuds entourés par quatre autres noeuds voisins (cela signifie que la carte est de topologie rectangulaire, pour d'autres types de topologie, il peut y avoir plus de noeuds voisins). Quand un noeud a moins de 4 voisins, c'est un noeud de frontière. Chaque noeud de frontière a une erreur cumulée dont la grande valeur signifie que beaucoup d'entrées ont été concentrées sur lui, de sorte que son vecteur de poids ne représente pas bien certaines entrées dans la région. Pour cette raison, de nouveaux noeuds sont ajoutés pour devenir des voisins de ce noeud de frontière. Dans ce type de processus, aucun noeud n'est supprimé de la carte parce que, comme indiqué dans [Merkl, 97], son addition est vraiment nécessaire. Cependant, les liens entre les noeuds devraient être contrôlés, de sorte qu'un lien puisse être ajouté ou supprimé. Un noeud qui est ajusté à nouveau est d'abord relié au noeud d'erreur. Si le nouveau noeud a son vecteur de poids semblable à celui de ses voisins, c à d, la distance dans l'espace de données est faible, des liens de lui à ses voisins sont établis.

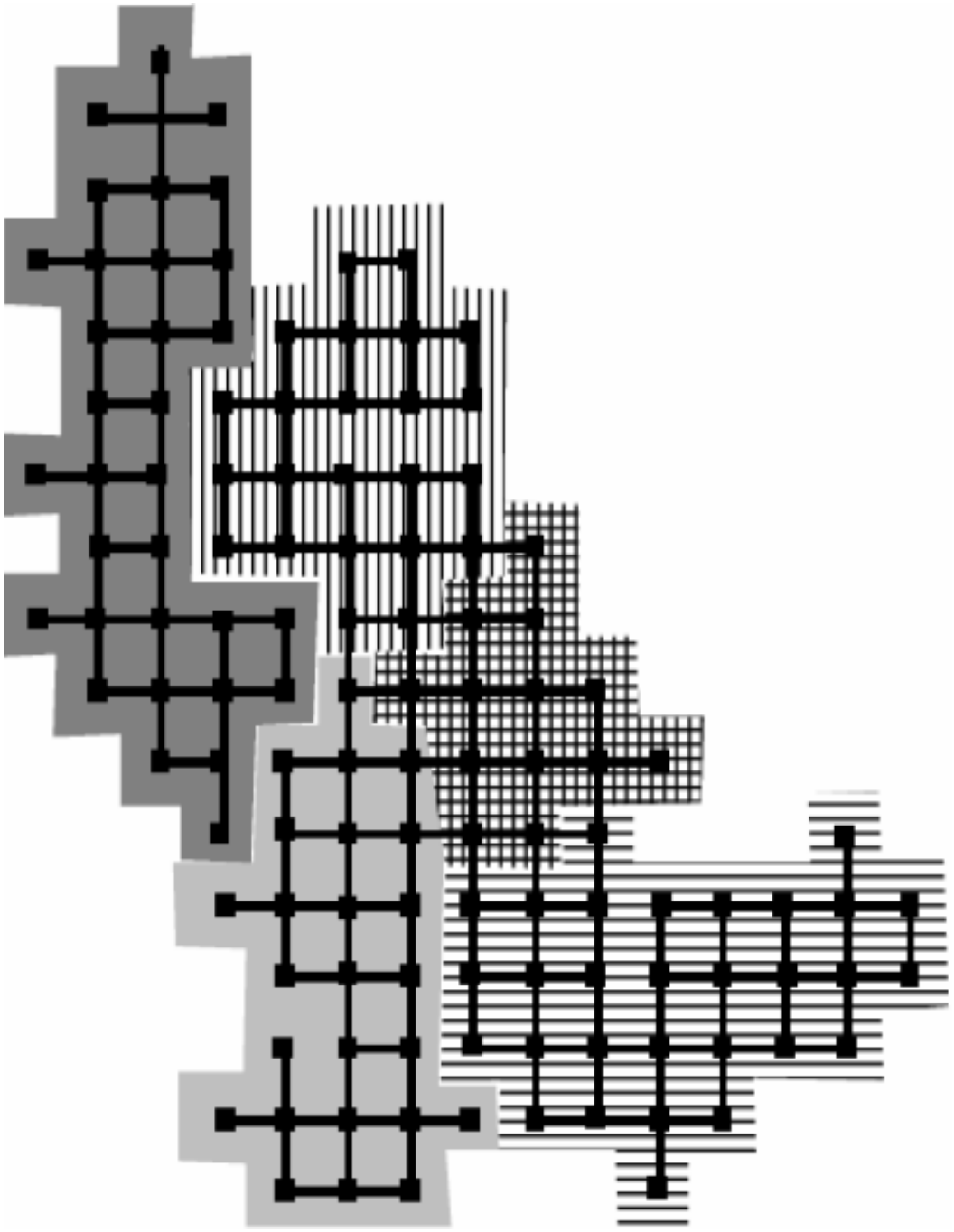


Figure 3-6: La carte finale d'un processus d'apprentissage (extraite de [Merkl, 97]).

Si les poids ne sont pas semblables, les liens correspondants sont supprimés. Au début, Figure 3-6 il y a seulement quatre noeuds sur la carte initiale. Enfin, une carte des régions bien séparées est obtenue, de sorte qu'elle reflète la structure de données et fourinit une vue de l'espace haut dimensionnel d'entrée.

4.5.3. SOM hiérarchique

On a proposé beaucoup de cartes multicouches. Dans [Merkl, 97], les cartes de premier niveau sont produites pour représenter les « grands» groupes, c à d. chaque noeud sur les cartes regroupe beaucoup d'entrées, et le nombre de noeuds sur ces cartes est faible. En raison de l'emploi d'un nombre restreint de noeuds, le temps de formation est considérablement réduit. Ensuite, des entrées appartenant à chaque groupe sur des cartes de premier niveau continueront à être groupés sur les cartes de deuxième niveau, de sorte que nous puissions obtenir des groupes secondaires du groupe sur les cartes de premier niveau. Ces entrées, qui ont été groupées dans un groupe sur les cartes de premier niveau, peuvent avoir plusieurs caractéristiques en commun (c'est la raison pour laquelle elles ont été placées dans un groupe). Pour cette raison, lors du processus d'apprentissage des cartes de deuxième niveau, les caractéristiques communes sont exclues des entrées, afin que les vecteurs caractéristiques résultants soient de dimensionnalité plus faible. En conséquence, le temps d'apprentissage pour les cartes de deuxième niveau est réduit. Les cartes du deuxième niveau sont elles mêmes également de petite taille, de sorte qu'une grande réduction du temps d'apprentissage est possible.

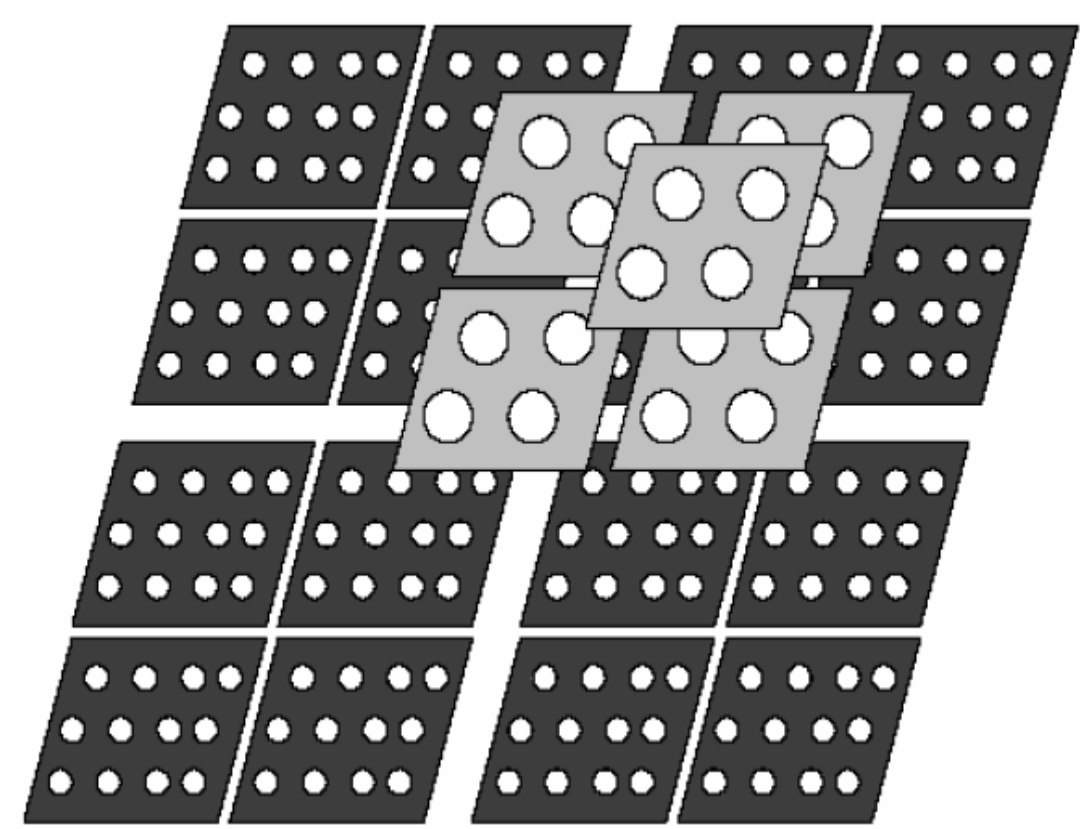


Figure 3-7: Une hiérarchie des cartes auto organisatrices (Extraite de [Merkl, 97]).

Comme indiqué en [Merkl, 97], cette modification peut aider à réduire le temps d'apprentissage pour un ensemble d'entrées. Tandis que le SOM classique prend beaucoup de temps pour déterminer si une entrée sur la frontière d'un groupe appartient à ce groupe ou à d'autres dans le processus d'apprentissage, la SOM hiérarchique l'a simplement mise dans un grand groupe de sorte qu'elle puisse être

organisée plus tard par les cartes du prochain niveau. Cela signifie que pour une entrée donnée, il faut plus longtemps pour trouver le gagnant que sur la SOM classique. Cependant, il reste des questions en suspens dans [Merkl, 97] au sujet de la taille appropriée des cartes de différents niveaux et au sujet du nombre de niveaux hiérarchiques. Figure 3-7

Dans [Dittenbach et al., 01], on propose la carte auto organisatrice hiérarchique croissante pour combiner deux propriétés «croissance» et «hiérarchie». Son processus d'apprentissage commence avec les cartes de premier niveau de taille 2x2. Après plusieurs itérations, il y a deux possibilités: la taille de la carte résultante est augmentée ou des cartes de niveau supérieur sont ajoutées. Cela signifie que la taille du premier niveau peut changer et il n'est pas systématique que pour chaque noeud sur les premières cartes, une SOM de niveau supérieur soit ajoutée. Cette méthode utilise des seuils qui doivent être fixés au préalable. La déviation entre les entrées d'un noeud avec son vecteur de poids nous indique si ce noeud est un noeud d'erreur, disons le noeud A. Un noeud est un noeud d'erreur s'il a la plus grande déviation. Puis, parmi les voisins du noeud d'erreur A, le plus différent par rapport à A est choisi, soit le noeud B. Entre ces deux noeuds A et B, une rangée (si A et B sont sur une rangée) ou une colonne (si A et B sont sur une colonne) de noeuds est insérée. La carte continue à se développer jusqu'à ce que la déviation moyenne de tous les noeuds sur la carte soit plus petite que Tl fois la déviation du noeud correspondant sur la couche précédente. Puis, chaque noeud sur la carte est considéré si des entrées lui appartenant sont organisées sur une carte indépendante de la prochaine couche. Un tel noeud est celui qui a la déviation plus grande que le seuil T2. Dans cet algorithme, Tl et T2 sont deux seuils prédéfinis. En même temps que le nombre d'itérations, ce sont deux paramètres sensibles pour cet algorithme, et un petit changement peut causer de grands changements de performance [Dittenbach et al., 01]. Figure 3-8

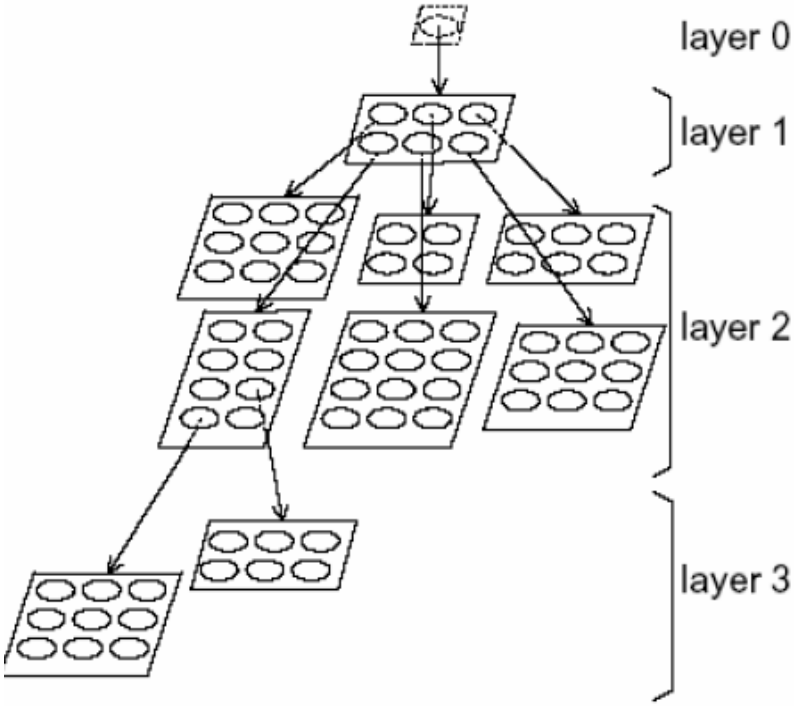


Figure 3-8: Une SOM hiérarchique croissante. (Extraite à partir de [Dittenbach et al., 01]).

Il faut noter que jusque récemment, il y a eu beaucoup de recherche se concentrant sur la mise à niveau des cartes auto organisatrices avec des extensions qui combinent la flexibilité des cartes et la hiérarchie multicouches des entrées. La SOM hiérarchique croissante peut être vue comme la première recherche dans cette direction. Les recherches plus récentes continuent dans ce sens avec encore plus d'ajustements, et les algorithmes récents deviennent plus complexes.

4.5.4 Choix de l'algorithme

Comme nous pouvons voir, l'algorithme de la SOM classique a beaucoup de points faibles qui doivent être considérés, comme le montre l'explosion des recherches qui ont été proposées pour surmonter ses limitations. Cependant, les points suivants peuvent nous donner confiance que l'utilisation de la SOM classique n'est pas du tout insignifiante dans le contexte de ces recherches:

- La SOM classique est plutôt simple mais très puissante. Sa force a été montrée dans le projet de WEBSOM. Toute modification la rendra plus complexe. Par exemple, il n'est pas facile de comprendre « la carte auto organisatrice hiérarchique croissante» de [Dittenbach et al., 01] même pour une personne qui maîtrise la SOM classique.

- N'importe quelle modification à la SOM classique pour la rendre plus puissante sur certains aspects peut la rendre moins puissante sur d'autres aspects. Pour les structures de cellules croissantes dans [Fritzke, 91] par exemple, on dit que la carte résultante peut préserver la topologie de données et préserver la distribution de données (c à d. « La structure de cellules dépendante de problème»), mais l'algorithme résout mal le problème de visualisation de données.

- Avec la SOM classique, on a également tout un ensemble de recherches concernant des problèmes comme la visualisation des données, le choix de paramètres de l'algorithme, l'initialisation des poids, etc. Il y a eu également beaucoup d'expériences effectuées avec la SOM classique pour vérifier son utilisation dans différentes applications. Pour les SOM modifiées, ce support n'est pas disponible, ou s'il l'est, il reste insatisfaisant.

- Les algorithmes améliorés de la SOM classique emploient beaucoup de paramètres sensibles qui affectent leur performance. Pour beaucoup d'algorithmes, ces paramètres doivent être manuellement prédéfinis et il est difficile de trouver les valeurs optimisées.

Pour ces raisons, nous avons décidé d'employer l'algorithme de la SOM classique.

5. La méthode de représentation des résultats

Une carte bien étiquetée peut permettre à l'utilisateur de choisir le groupe qui est le plus approprié à sa demande. Cet objectif devrait être atteint par une bonne méthode d'étiquetage automatique. Après avoir été formée, la SOM devrait être étiquetée et les étiquettes devraient représenter aussi exactement que possible la teneur des groupes qu'ils représentent.

5.1 Etiquetage des groupes générés par la SOM

Étiqueter des groupes des documents sur la SOM a été considéré comme une étape auxiliaire visant à nommer les groupes sur la carte. Cependant, peu de travaux s'intéressent à l'étiquetage des SOMs. Dans cette section, nous décrivons ces travaux.

5.1.1 Étiquetage manuel

Dans ce cas, on assigne manuellement une étiquette à chaque groupe de documents. Dans [Lagus, 97], l'étiquetage d'un groupe est effectué juste après que l'auteur ait lu tous les textes de ce groupe et l'étiquette est simplement un mot utilisé pour faire la différence entre les groupes. L'étiquetage est alors une étape auxiliaire. Cependant, l'étiquetage manuel peut devenir vraiment utile pour rechercher l'information si après lecture de tous les documents, nous faisons un sommaire pour chaque groupe sur la carte. Ce sommaire peut guider l'utilisateur dans son processus de recherche. Figure 3-9

Cette méthode semble convenir à la vérification de la SOM plutôt qu'à l'aide à la recherche. En fait, manuellement, nous pouvons produire des étiquettes aussi exactes et utiles que nous souhaitons. Cependant, cette procédure n'est pas automatique et donc coûteuse en temps. Pour une petite collection de documents, il est possible de faire cela. Mais pour un énorme ensemble de documents, l'étiquetage manuel est impossible. Ces inconvénients peuvent empêcher cette méthode d'être employée pour l'aide à la recherche l'information. En outre, cette approche peut donner des étiquettes variables selon la personne qui a effectué l'étiquetage et le moment où cela a été réalisé.

Figure 3-9 : Un exemple d'étiquetage manuel extrait à partir de [Lagus, 97].Le but de ceci est de déterminer si l'organisation des documents sur la carte est bien faite ou pas. Ce n'est pas utile du tout dans le but de la recherche d'information

5.1.2 Étiquetage basé sur des groupes prédéfinis

La SOM est une méthode de classification non supervisée, c'est à dire elle n'utilise pas de classes données a priori. Cependant, la SOM peut être employée de manière manuellement supervisée de sorte que des étiquettes sur la SOM puissent être apprises à partir des groupes originaux. Les documents utilisés pour la SOM sont des ensembles de documents qui ont été pré classifiés. Par exemple, dans [Timo et al., 97], les articles du groupe de discussion Usenet "comp_ai_neural_nets" ont été employés. Un document, qui est un article, est représenté de telle manière que l'information de son groupe prédéfini soit incluse dans le vecteur de document. Le terme "supervision partielle" est alors employé pour impliquer cette inclusion. Si d est un vecteur de document, alors les composants de d sont: $d = d_1 \dots d_m d_{m+1} \dots d_M$ où $d_1 \dots d_m$ sont des composants représentant le contenu de document et $d_{m+1} \dots d_M$ sont des composants encodant le groupe prédéfini du document. Figure 3-10

Le vecteur d=

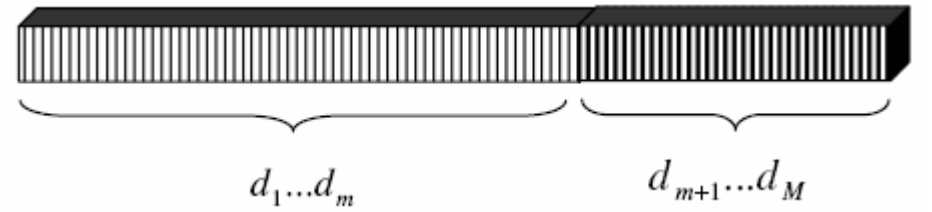


Figure 3-10: Structure des vecteurs en entrée

où m est le nombre de dimensions du vecteur qui représente le contenu du document, (M-m) est le nombre de dimensions du vecteur qui représente le groupe prédéfini du document. En plus de jouer un rôle très important en séparant les groupes sur la SOM les uns des autres, les composants $d_{m+1} \dots d_M$ peuvent être employés pour impliquer une étiquette pour le groupe sur la carte. Figure 3-11

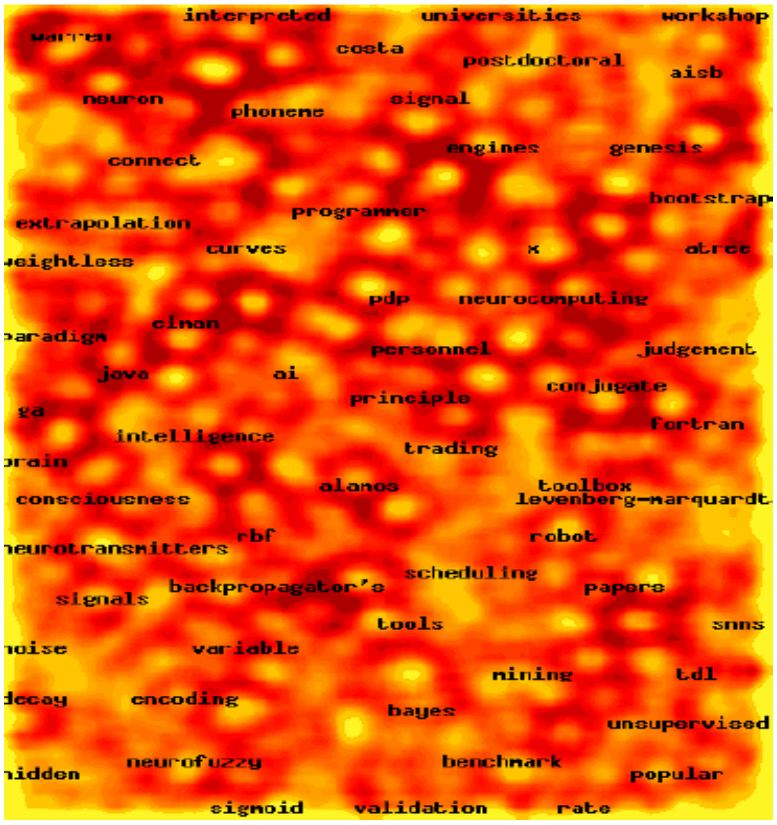


Figure 3-11: Un exemple de l'étiquetage basé sur les groupes prédéfinis extrait du site Web du projet de WEBSOM :

http://websom.hut.fi/websom/comp.ai.neural-netsnew/html/root.html

Après plusieurs itérations, presque tous les documents d'un groupe prédéfini tombent dans un secteur sur la carte (c'est également le but de la SOM partiellement supervisée). La tâche d'étiquetage correspond alors simplement à copier les étiquettes des groupes prédéfinis sur ces secteurs. Le nombre de groupes prédéfinis est plus petit que le nombre de groupes sur la carte. C'est pourquoi en employant cette méthode, plusieurs groupes peuvent avoir la même étiquette de groupe prédéfini. Dans [Timo et al., 97], 20 groupes de nouvelles ont été employés tandis que la taille de la carte était 24x32.

Avec le facteur de supervision, des secteurs dans les cartes peuvent être clairement distingués. Cependant, la méthode utilisée est triviale. Dans le but de la recherche d'information, l'utilisateur peut ne pas obtenir l'aide nécessaire pour trouver les bons documents et en fait, les groupes prédéfinis peuvent ne pas être disponibles. Il semble que cette méthode d'étiquetage est juste employée pour aider à faire la différence entre les secteurs sur la carte.

5.1.3 Etiquetage par les étiquettes des entrées

Cette approche est un peu similaire à celle basée sur les groupes prédéfinis. Une de ces applications est la carte de catégories de mots [Honkela et al., 96], qui est employée pour grouper des mots en se basant sur leur contexte d'occurrence dans la collection de documents. Le principe de cette application est qu'un mot a une signification particulière dans un contexte particulier et dans un document, il y a peut être beaucoup de mots ayant la même signification (phénomène de synonymie). Une catégorie des mots ayant la même signification peut alors être employée au lieu des mots eux mêmes pour représenter le document. Premièrement, tous les mots apparaissant dans la collection courante de documents sont insérés dans un dictionnaire. Un vecteur P dont les éléments sont produits aléatoirement code chaque mot. Le mot considéré appelé le mot clé doit être placé dans son contexte, qui est une expression (ou une fenêtre) contenant W mots consécutifs (il peut y avoir W=2, 3, 4,5 ... mots dans une expression) où le mot clef est au milieu de l'expression. Pour représenter tout le contexte, un vecteur de P.W éléments appelé le vecteur de contexte, est nécessaire. En fait, les vecteurs d'entrée sont les vecteurs des contextes moyens. Figure 3-12

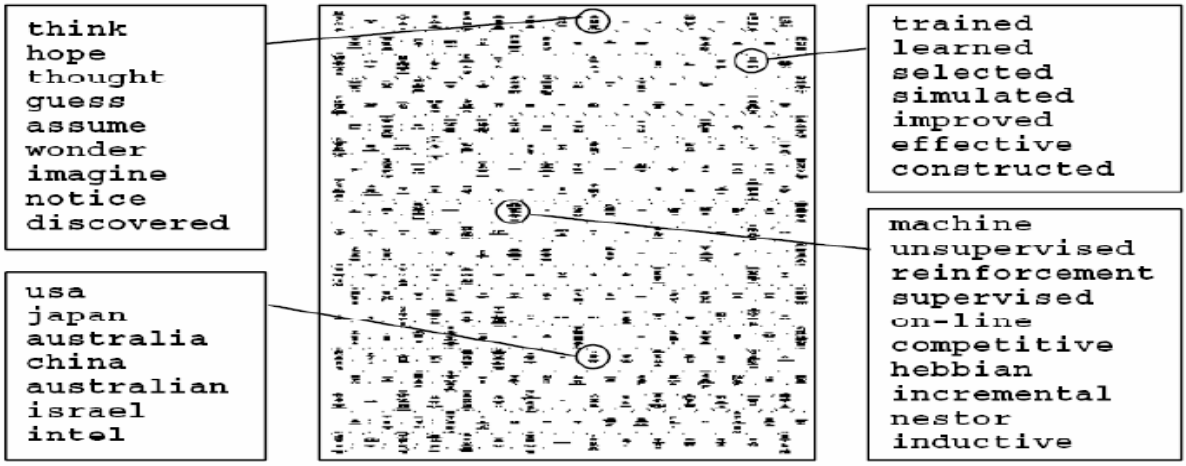


Figure 3-12:Un exemple des cartes de catégories de mots.
Cette Figure est extraite de [Honkela et al., 96].

L'étiquetage est effectué en employant l'étiquette du mot clef du vecteur d'entrée. Les vecteurs qui représentent un contexte semblable tombent dans un groupe sur la carte. L'étiquette du mot clef appartenant à un de ces vecteurs sera employée pour étiqueter le groupe. Pour cette raison, l'étiquette est tout à fait simple. Bien que cette méthode ne fasse pas l'étiquetage d'une collection de document et en conséquence, ne soit pas employée pour aider à la recherche l'information, elle peut nous donner une idée de ce qui peut être fait pour l'étiquetage sur la SOM. C'est pour cette raison que nous la considérons toujours dans ce rapport.

5.1.4 Méthode LabelSOM

C'est l'une des recherches pionnières sur l'étiquetage des groupes sur la SOM. Le but de cette méthode est d'extraire l'étiquette la plus descriptive pour le groupe de documents en se basant sur les caractéristiques communes des documents appartenant à un groupe. Pour cela, la pleine représentation des documents employant le modèle de l'espace vectoriel est utilisée. Pour calculer l'importance d'un mot, la méthode *tfidf* est choisie.

Pour chaque groupe, un vecteur d'erreur de quantification est calculé. Ce vecteur a un nombre de dimensions égal à celui des vecteurs d'entrée et son j^e élément est la somme de déviations produites par différence entre les j^e éléments des vecteurs d'entrée et le j^e élément du vecteur modèle représentant le groupe sur la SOM. Plus un élément dans le vecteur d'erreur de quantification est petit, le mieux les éléments situés à la même position dans les vecteurs d'entrée s'accordent et plus il est vraisemblable que ce composant sera choisi pour étiqueter le groupe. Pour choisir les éléments les plus descriptifs, un seuil est défini. Ce seuil est le seuil de candidat parce qu'il limitera le nombre d'éléments choisis. Le seuil peut être un nombre prédéfini d'éléments qui seront choisis ou une précision seuil prédéfinie à laquelle la valeur de l'élément choisi doit être inférieure. Par conséquent, pour chaque groupe, un ensemble d'éléments candidats est choisi.

Si Q_k est le vecteur d'erreurs de quantification du groupe C_k, ses composants $Q_{k,j}$ sont calculés comme suit:

$$Q_{k,j} = \sum_{i \in c_k} (d_{i,j} - c_{k,j}) \qquad 5.1.4.1$$

Parmi ces éléments, il peut y avoir des éléments qui ne sont pas spécifiques au groupe. Il s'agit des éléments ayant une petite erreur de quantification non seulement pour ce groupe mais aussi pour l'autre groupe. Ils correspondent aux mots qui sont très communs dans la collection entière de documents. Pour cette raison, un deuxième seuil est défini pour choisir parmi les éléments candidats les éléments les plus spécifiques au groupe. Ce seuil pourrait s'appeler le seuil d'importance, qui décrit la valeur minimum qu'un élément doit avoir pour être considéré le plus descriptif. A la fin du processus, chaque groupe est caractérisé par une liste d'éléments qui sont les plus descriptifs et aussi les plus spécifiques. Les mots correspondant à ces éléments sont alors employés pour étiqueter le groupe [Timo et al., 97].

Dans [Rauber et Merkl, 01] ont montré que les étiquettes obtenues sont très descriptives. Des groupes sur la carte sont décrits par les mots qui reflètent leur contenu. Avec ceci, l'utilisateur peut obtenir l'information utile de sorte qu'il peut prendre une décision plus rapide tout en recherchant le document désiré et qu'il peut éviter de choisir les groupes non pertinents.

Néanmoins, cette méthode a également quelques limites. Premièrement, elle exige que les documents soient représentés par la pleine représentation. Pendant le processus de recherche des composants les plus descriptifs, chaque composant du vecteur doit être inspecté. Cette représentation peut être impossible même pour un ensemble moyen de documents ou pour un ensemble de documents dans lequel les documents ont des sujets très différents. Deuxièmement, après les itérations d'apprentissage, considérer à nouveau l'espace d'entrée peut prendre du temps. Pour trouver les étiquettes les plus descriptives d'un groupe sur la carte, il faut trouver les composants les plus communs des vecteurs de document appartenant au groupe. Ceci, à son tour, exige d'examiner tous ces vecteurs d'entrée. Quand il y a beaucoup de documents, ceci peut prendre beaucoup de temps. Troisièmement, il est difficile de fixer les seuils. Le choix des meilleurs seuils reste une question en suspens.

6. Méthodes d'indexation multilingue

Il y a deux types de collection de documents multilingues [Wen et Hsin, 02]. La première contient certaines collections de documents monolingues. La deuxième contient des documents multilingues. Une vue d'ensemble des méthodes d'indexation de documents multilingues a été donnée dans [Douglas, 96] et voici les possibilités pour l'indexation de documents multilingues:

- **Traduction de texte:**

Les documents multilingues sont traduits dans une langue intermédiaire français par exemple. Les documents intermédiaires sont alors indexés et représentés par des vecteurs de caractéristiques. La qualité de la traduction détermine la qualité d'indexation des documents.

- **Thésaurus multilingues:**

Plusieurs types de thésaurus peuvent être construits. Pour une liste de concepts, l'espace de termes est divisé en classes de concepts. Pour une liste de termes, un terme dans une langue est associé au terme correspondant dans une autre langue. On peut aussi construire une hiérarchie de termes où il y a des relations associatives entre les termes. Une ontologie est aussi un thésaurus dans ce cas.

- **Techniques basées sur un corpus:**

C'est une direction de recherche plus directe pour indexer des documents multilingues. Les vecteurs de termes d'une langue sont traduits grâce aux correspondances produites pendant le traitement des ensembles de termes de chaque langue. Dans d'autres expériences, la méthode LSI (Latent Semantic Indexing) est aussi une technique de ce genre. Une limite de ces techniques est que le nombre de calculs mathématiques est grand.

7. Ontologie pour la représentation de documents

Pour utiliser des ontologies dans le domaine du regroupement de textes, deux étapes de prétraitement sont nécessaires. Une ontologie qui décrit le domaine cible doit d'abord être construite, en utilisant un corpus de textes ou des définitions manuelles, ou les deux. Les documents peuvent être indexés à travers cette structure pour produire des vecteurs qui les représentent. Ces vecteurs seront utilisés par une méthode de regroupement comme des entrées. Pour la première tâche la construction de l'ontologie, plusieurs outils peuvent être utilisés pour déterminer les concepts et leurs inter-relations. Pour la seconde, un indexeur est nécessaire pour produire des vecteurs caractéristiques de documents.

Les ontologies ont récemment été utilisées pour représenter des documents. La première recherche de ce genre a été réalisé par A. Hotho et ses collègues avec l'article texte "Ontology-based text clustering" [Hotho et al., 01]. Pour étayer leur proposition, ils ont utilisé le K-Means comme méthode de regroupement. Leur approche, nommée CASA (Concept Selection and Aggregation), utilise une ontologie noyau pour restreindre l'ensemble de traits de documents et proposer automatiquement des agrégations appropriées. Les concepts qui sont sélectionnés depuis l'ensemble de documents doivent représenter un domaine prédéfini. Ensuite, parce que le nombre de concepts est élevé, l'agrégation est réalisée de manière à générer pour des concepts de même catégorie un concept qui les recouvre. Ils proposent par conséquent une méthode heuristique basée sur une hétérarchie qui fournit des vues de concepts afin que le nombre de dimensions de vecteurs résultants puisse être réduit. Cette recherche est très intéressante et les résultats obtenus sont encourageants.

En suivant cette proposition, des recherches ont été réalisées pour appliquer des ontologies domaine spécifiques au regroupement de texte. Par exemple, dans [Smirnov et al., 05] une ontologie est utilisée pour regrouper des clients et des demandes dans la Gestion du Service après vente. Dans [Haiying et al., 05], la tâche de regroupement se base sur une ontologie dans l'analyse de l'expression du gène. Dans [Liang et al., 08] Un modèle OWL pour la création d'ontologies dans le domaine de l'agriculture.

Chapitre 4:

Une ontologie Sonelgaz

1. Introduction

Sonelgaz, est la société algérienne de l'électricité et du gaz, qui s'occupe de la production, le transport et la distribution de l'électricité et du gaz naturel. Cette compagnie reçoit au niveau de ces unités régionales des milliers d'appels au quotidien, en voulant améliorer encore plus la qualité de ses services, elle souhaite automatiser le routage des appels des clients vers les services concernés. La première proposition pour la réalisation de ce projet comprend les étapes décrites par la Figure 4-1.

Et, comme les appels téléphoniques reçus par le système sont de nature multi langue à savoir arabe, anglais et français, dont plusieurs types de thésaurus peuvent être construits pour une liste de concepts et si on cherche à indexer ces fichiers audio après leur transcription, nous ne pouvons pas nous prononcer sur leur conclusion concernant le contenu, dans le cas où il y a deux fichiers qui sont arrangés dans le même groupe malgré que leurs vecteurs correspondants soient semblables.

Donc, il faut que notre solution arrive à présenter l'ensemble des fichiers audio transcrits dans une langue intermédiaire qui serait un système de concepts et non un système de signes, pour que le système de classification SOM arrive à indexer l'ensemble des fichiers audio en respectant le contenu multi langue. Cette langue intermédiaire serait donc une langue basée sur les concepts communs et non sur la disparité des termes dans les différentes langues utilisées dans le système.

L'approche proposée selon le schéma de la Figure 4-1 comprend quatre (4) phases principales :

- **Dans la première phase** : Il s'agit d'abord d'extraire les informations pertinentes des appels reçus ; cette première étape peut être effectuée par un système de reconnaissance de la parole ou encore par un système de détection de mots clés. Toutefois, à ce stade du projet, les tests que nous avons effectués sont basés sur des documents transcrits par les opérateurs, car nous avons commencé par la mise au point de l'ontologie de « Sonelgaz », et un essai de caractérisation des appels (documents audio) en lien de cette ontologie.

- **Dans la deuxième phase,** c'est la création de l'ontologie de domaine sur laquelle seront projetés les documents transcrits, dont nous procéderons à une exposition détaillé dans les parties 1 et 2 de ce chapitre.

- **Dans la troisième phase,** application de la projection d'après le modèle ***DocCore,*** *présentée dans la partie 3 de ce chapitre.*

- **Dans la quatrième phase,** application de l'algorithme SOM en utilisant les vecteurs de concepts obtenus, le détail sera abordé dans le chapitre 5.

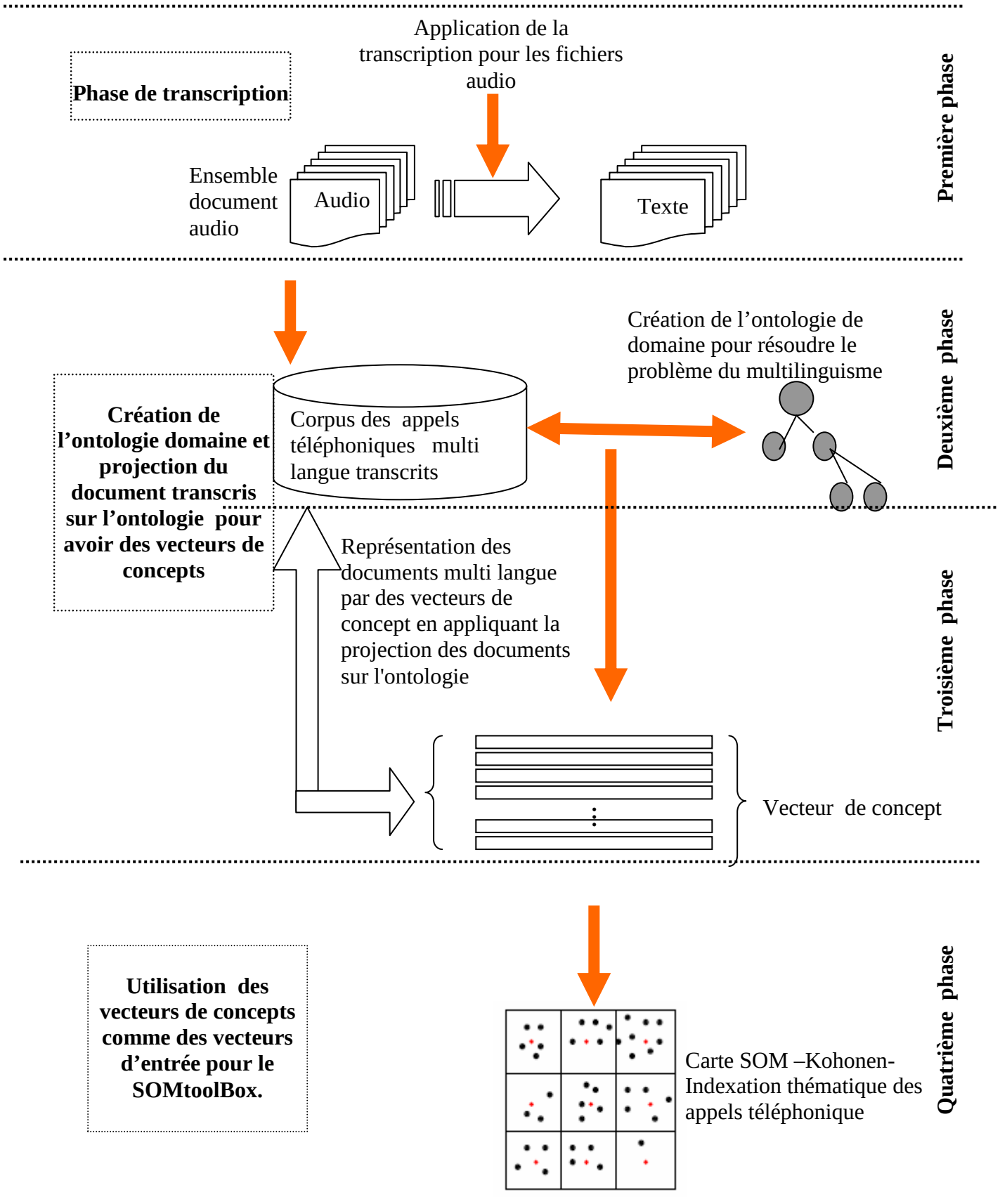


Figure 4-1 : L'architecture Du Système

Partie 1: Les éléments d'une ontologie

1.1 Introduction

Le terme « ontologie », construit à partir des racines grecques *ontos* (ce que existe, l'existant) et *logos* (le discours, l'étude), est un mot que l'informatique a emprunté à la philosophie au début des années 1990. En philosophie, l'Ontologie est une branche fondamentale de la Métaphysique, qui s'intéresse à la notion d'existence, aux catégories fondamentales de l'existant et étudie les propriétés les plus générales de l'être.

L'utilisation d'ontologies en informatique vise à intégrer une couche de connaissances aux systèmes afin de permettre des traitements élaborés de l'information qu'ils manipulent.

La conception d'ontologies est une tâche difficile qui nécessite la mise en place de procédés élaborés afin d'extraire la connaissance d'un domaine, manipulable par les systèmes informatiques et interprétable par les êtres humains.

Dans cette partie, nous allons étudier les différents points de vu sur l'ontologie, ensuite les différents éléments qui la composent et les besoins auxquels elle répond, puis, nous citerons les différentes classifications et les principaux formalismes utilisés pour représenter une ontologie afin de construire notre ontologie de domaine en respectant le problème de multilinguisme.

1.2 Définitions

Dans cette section, nous essayons de donner un éventail de définitions utilisées pour qualifier le concept d'ontologie.

Définition 1:

En Intelligence Artificielle, la définition communément admise est celle énoncée par T. GRUBER qui définit l'ontologie comme une spécification explicite d'une conceptualisation [Gruber, 93]. De ce point de vue, la construction d'une ontologie interviendrait après le travail de conceptualisation. Cette démarche de conceptualisation consiste à identifier, au sein d'un corpus, les connaissances spécifiques au domaine de connaissances à représenter.

Cette définition a fait l'objet d'une précision de la part de B. STUDER qui voit ainsi une ontologie comme la spécification formelle et explicite d'une conceptualisation partagée [Studer, 98]. Dans cette définition, il convient de mesurer la portée de chaque terme utilisé. Ainsi, le terme «spécification explicite» indique qu'une ontologie est un ensemble de concepts, de propriétés, d'axiomes, de fonctions et de contraintes explicitement définis. Le terme «formel» précise que cette conceptualisation doit pouvoir être comprise et interprétée par une machine. Le terme «partagé» précise l'aspect consensuel du vocabulaire employé. Ce qui suppose que l'on doit assurer une réutilisation de la formalisation choisie. Enfin, le terme «conceptualisation» implique également l'aspect intentionnel, lié à un objectif de réalisation.
N. GUARINO affine cette définition dans [Guarino et Giaretta, 95], en considérant qu'une ontologie est une spécification partielle et formelle d'une conceptualisation (introduction du niveau ontologique). Cette remarque fait apparaître les différents rôles attribués aux ontologies suivant les domaines (pour plus de détails, voir [Fürst, 02], [Gruber, 93], [Gruber et al., 95] et [Guarino et Giaretta, 95]).

Définition 2:

L'ontologie peut aussi être définie comme étant un outil permettant de fournir une conceptualisation sommaire d'une information et un ensemble de termes à utiliser dans cette représentation [Parekh et al., 04].

Définition 3:

Une ontologie est un ensemble de concepts clairement explicités et un ensemble de relations entre ces concepts [Chandrasekaran et al., 99]. Une ontologie peut donc être représentée avec une taxonomie (arbre d'héritage) des concepts.

Comme nous l'avons dit précédemment, une ontologie doit présenter les fonctions (ou caractéristiques) du domaine qu'il représente. Elle doit, en outre, classifier les principaux concepts du domaine et rendre évidentes les relations et les contraintes entre ces concepts. Les ontologies sont des représentations de connaissances, contenant des termes et des énoncés qui spécifient la sémantique d'un domaine de connaissance donné dans un cadre opérationnel fixé.

1.3 Composants d'une ontologie :

Une ontologie ne peut être construite que dans le cadre d'un domaine précis de la connaissance, parce que beaucoup de termes n'ont pas le même sens d'un domaine à un autre. Les connaissances traduites par une ontologie sont à véhiculer à l'aide des principaux éléments suivants :

a) **Concept**
b) **Relation**
c) **Les rôles**
d) **Les fonctions**
e) **Les axiomes**
f) **Les instances**

a) Concept :

Qui peut représenter un objet, une idée. Un concept peut être divisé en trois parties :

1. **Un terme** : est un élément lexical qui permet d'exprimer le concept en langue naturelle. Il peut admettre des synonymes.

2. **La notion** : également appelée intension du terme, contient la sémantique du concept, exprimée en terme de propriétés et attributs, et de contraintes.

3**. L'ensemble d'objets** : appelé extension du concept, regroupe les objets manipulés à travers le concept ; ces objets sont appelés instances du concept.

- **Propriétés sur les concepts** :

  - La généricité : un concept est générique s'il n'admet pas une extension, exemple, la vérité.
  - L'identité : permet de conclure si deux entités sont identiques
  - La rigidité : un concept est rigide si toute instance de concept en reste instance dans tous les mondes possibles.

- L'anti-rigidité : toute instance du concept est définie par son appartenance à l'extension d'un autre concept
- Propriétés portant sur deux concepts :
  - **L'équivalence**: deux concepts sont équivalents s'ils ont la même extension.
  - **La disjonction**: deux concepts sont disjoints (incompatibles) si leurs extension sont disjointes.

b) Relation :

Une relation permet de lier les instances de concepts, ou des concepts génériques. Elles sont caractérisées par un (ou plusieurs) terme(s) et une signature qui précise le nombre d'instances de concepts que la relation lie, leurs types et les ordres des concepts.

- **Propriétés sur les relations**

1. Propriétés algébriques :
   - La symétrie
   - La réflexivité
   - La transitivité.
2. La cardinalité : nombre de relations possibles entre les mêmes concepts.
3. Propriétés liant deux relations
   - L'incompatibilité : Deux relations sont incompatibles si elles ne peuvent lier les mêmes concepts.
   - L'inverse : Deux relations binaires sont inverses l'une de l'autre si, quand l'une lie deux instances I1 et I2, l'autre lie I2 et I1.
   - L'exclusivité : deux relations sont exclusives si, quand l'une lie des instances de concepts, l'autre ne le fait pas.

c) Les rôles

Selon [Sowa, 84], « un rôle caractérise une entité par quelque rôle qu'elle joue dans sa relation à une autre entité. Le type « Humain », par exemple, est un type de phénomène qui dépend de la forme interne de l'entité ; mais la même entité peut être caractérisée par des rôles du type, Mère, Employé ou Piéton. ».

d) Les fonctions :

Les fonctions sont aussi des cas particuliers de relations dans lesquelles le $n^{ème}$ élément de la relation est défini à partir des ***n premiers***. Comme exemple de fonctions binaires il y a la fonction ***mère de*** ou ***carré de***, comme fonction ternaire, le prix d'une voiture usagée sur lequel on peut se baser pour calculer le prix d'une voiture d'occasion en fonction de son modèle, de sa date de construction et de son kilométrage.

e) Les axiomes

Les axiomes sont utiles à la structuration de phrases qui sont toujours vraies. Ils permettent de contraindre les valeurs de classes ou d'instances.

f) Les instances

Les instances sont utilisées pour représenter des éléments dans un domaine.

1.4 Importance d'avoir une ontologie

Avant d'évoquer l'importance que peut avoir l'utilisation des ontologies, nous estimons qu'il est utile de donner un aperçu sur les rôles et les fonctions que peut jouer une ontologie. Il s'agit de permettre la communication entre les systèmes, entre les utilisateurs et les machines ou entre simples utilisateurs. Ces rôles sont décrits dans les travaux de [Gruninger et Lee, 02]. Leur utilisation permettrait aussi l'inférence, la réutilisation et l'organisation de la connaissance.

En prenant le domaine de la logique de programmation, les ontologies y jouent, selon [Miller, 00], deux fonctions principales:

- fournir un moyen de visualisation d'un domaine et d'organisation de l'information;
- définir un vocabulaire commun en expliquant la signification des termes et des relations entre ces termes.

Comme nous pouvons le constater, une ontologie permet de clarifier la structure de la connaissance [Chandrasekaran et al., 99]. Ainsi, l'ontologie forme le coeur de tout système de représentation de connaissances pour un domaine. En effet, sans les conceptualisations de la connaissance d'un domaine, il est difficile d'avoir un vocabulaire pour la représenter.

Pour créer une ontologie il semble donc primordial d'effectuer une bonne analyse du domaine en clarifiant les terminologies employées pour cette création. Et l'utilisation de ces ontologies s'explique selon [Parekh et al., 04], par les facteurs suivants:

- l'ontologie peut être construite pour fournir un vocabulaire (partagé et commun) utilisé dans la description d'un ensemble de données et permettre ainsi de définir un vocabulaire compris et partageable par tous;

- l'ontologie peut fournir un schéma conceptuel pour tout type de données sans se soucier de son format, de sa structure ou de sa taille;

- l'ontologie peut être conçue pour comprendre sémantiquement le contenu et la structure des données contenues dans une base de données;

- l'ontologie peut être utilisée pour aider les fournisseurs de données à entrer les métas donnés dans un format sémantiquement valide;

- l'interopérabilité entre les ensembles de données hétérogènes peut être atteinte en utilisant une ontologie partagée;

- l'ontologie peut être vue comme étant le modèle de représentation de connaissances le plus avancé.

1.5 Les types d'ontologies :

Ainsi, il est possible et même conseillé d'utiliser le pluriel pour parler de la notion d'ontologie afin de refléter les multiples facettes qu'elle recouvre.
Selon Van Heijst [Van Heijst et al., 97], on peut distinguer quatre types d'ontologies:

- **Les ontologies du domaine:**

  Elles sont appelées de la sorte parce qu'elles expriment des conceptualisations spécifiques à un domaine. Elles rendent compte du vocabulaire d'un domaine spécifique au travers de concepts et de relations qui modélisent les principales activités, les théories et les principes de base du domaine en question. Elles sont réutilisables pour plusieurs applications concernant le domaine pour lequel elles ont été créées car elles ont été conçues de façon aussi indépendante que possible du type de manipulations qui vont être opérées sur ces connaissances.

- **Les ontologies applicatives ou ontologies d'application:**

  Ce sont les ontologies les plus spécifiques, elles contiennent les connaissances requises pour une application particulière et ne sont pas réutilisables. Elles peuvent en outre inclure une ontologie de domaine.

- **Les ontologies génériques ou ontologies de haut niveau :**

  Elles expriment des conceptualisations valables dans différents domaines de valeur relativement générale comme les notions d'objets, de propriété, de valeur, d'état, ou encore de temporalité. Théoriquement, ces ontologies doivent pouvoir être reliées au sommet des ontologies de domaines.

- **Les ontologies de représentation:**

  Ce type d'ontologies regroupe les concepts utilisés pour formaliser les connaissances. Parmi les ontologies de représentation, on trouve des ontologies qui vont décrire les notions utilisées dans toutes les ontologies pour spécifier les connaissances, telles que les substances, les concepts, les relations etc. Par exemple, la « Frame-Ontology » est une ontologie de représentation. Elle définit de manière formelle les concepts utilisés principalement dans les langages à base de frames: classes, sous classes, attributs, valeurs, relations et axiomes. Selon Guarino, les ontologies de représentation sont en fait indépendantes des différents domaines de connaissances, puisqu'elles décrivent des primitives cognitives communes aux différents domaines.

Par ailleurs, [Uschold et Gruninger, 96] précisent que les ontologies peuvent être de natures variables, suivant le type de langage utilisé et donc allant d'un degré de formalisation zéro à une formalisation totale. Quatre distinctions sont mises au jour :

- **Les ontologies informelles:** elles sont exprimées en langue naturelle. Ainsi, cela peut les rendre plus compréhensibles par l'utilisateur, mais cela rend plus difficile à vérifier l'absence de redondance ou de contradiction dans les ontologies. En d'autres termes, elles sont plus difficiles à valider.

- **Les ontologies semi-informelles:** elles sont exprimées dans une forme de langue naturelle structurée et limitée. Cela permet d'augmenter la clarté de l'ontologie tout en réduisant l'ambiguïté.

- **Les ontologies semi-formelles :** elles sont exprimées dans un langage artificiel et défini formellement.

- **Les ontologies formelles:** elles sont exprimées dans un langage artificiel disposant d'une sémantique formelle, permettant de prouver des propriétés de cette ontologie. L'intérêt de ces ontologies est la possibilité d'effectuer des vérifications sur l'ontologie: complétude, non-redondance, cohérence, etc.

Uschold et Gruninger expliquent également que du degré de formalisation de l'ontologie, dépend le degré d'automatisation dans les diverses tâches impliquant l'ontologie. «Si une ontologie est une aide à la communication entre personnes, alors la représentation de l'ontologie peut être informelle du moment qu'elle est précise et qu'elle capture les intuitions de chacun. Cependant, si l'ontologie doit être employée par des outils logiciels ou des agents intelligents, alors la sémantique de l'ontologie doit être rendue beaucoup plus précise» [Uschold et Gruninger, 96].

Enfin, une dernière classification peut s'effectuer en fonction du niveau de granularité, c'est-à-dire du niveau de détail des objets de la conceptualisation. Ainsi, selon l'objectif opérationnel de l'ontologie, une connaissance plus ou moins fine du domaine est nécessaire et des propriétés considérées comme accessoires dans certains contextes peuvent se révéler indispensables pour d'autres applications. On peut relever alors deux types de granularités :

- **Granularité fine:** cela correspond à des ontologies très détaillées, possédant un vocabulaire riche capable d'assurer une description détaillée des concepts pertinents d'un domaine

- **Granularité large:** cela correspond à un vocabulaire moins détaillé. Les ontologies de haut niveau ont par exemple une granularité large, car les notions sur lesquelles elles portent peuvent être raffinées par des notions plus spécifiques.

1.6 Étapes de mise en œuvre d'une ontologie

Pour mettre en place une ontologie, diverses démarches peuvent être entreprises par les concepteurs. Cependant, quelle que soit la démarche, il convient de respecter quelques principes. Ainsi, on commencera par identifier les connaissances contenues dans un corpus représentatif du domaine. Ce processus porte le nom de *conceptualisation.* Ensuite, une modélisation semi-formelle du domaine sur lequel portera l'ontologie s'impose. Ce processus porte le nom *d'ontologisation* [Kassel et al., 00]. De plus, cette ontologie doit être traduite dans un langage formel et compréhensible par la machine.

Le langage de représentation de l'ontologie doit donc permettre de représenter et de manipuler les connaissances modélisées par l'ontologie. Ce processus de traduction est appelé *opérationnalisation.* Ce processus est détaillé dans la Figure 4-2 et pour plus de détails, se référer à [Fürst, 02], [Gandon, 02].

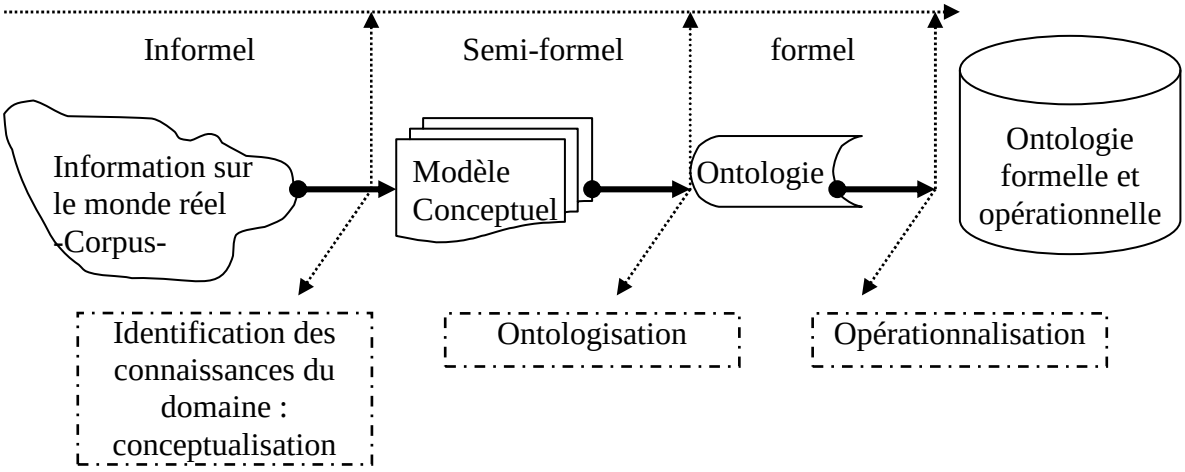


Figure 4-2: Etapes de mise en ouvre d'ontologie selon [Fürst, 02]

La séparation de ces deux processus permettra d'avoir des ontologies pouvant être utilisées comme des composants logiciels dans des systèmes différents. Des travaux relatifs au cycle de vie des ontologies, inspirés du génie logiciel, ont été proposés par [Dieng et al., 01]. Il convient aussi de noter que le processus de mise en œuvre d'une ontologie n'est pas linéaire. Ainsi, il est possible de revenir sur des étapes déjà validées en amont pour bâtir une ontologie opérationnelle adaptée à des besoins spécifiques.

1.7 Processus de création d'ontologie

Pour que la conceptualisation d'un domaine se fasse de manière non ambiguë, il est nécessaire de préciser les objectifs de celle-ci. En effet, un même terme peut désigner deux concepts différents dans deux contextes différents [Bachimont, 00]. Il est donc impossible de dissocier la représentation des connaissances d'un domaine et la modélisation des traitements que l'on souhaite leur appliquer. Ce qui signifie que toute tâche de modélisation de connaissances doit se faire dans un domaine précis avec un but précis, condition nécessaire à l'unicité de la sémantique associée aux termes du domaine. Selon [Holsapple et Joshi, 02], la conception d'une ontologie est dictée par un certain

nombre de principes qu'il détaille dans ses travaux. De plus, il avance cinq approches utilisées dans la conception d'une ontologie pour un domaine: *l'inspiration* (un point de vue individuel sur le domaine), *l'induction* (cas spécifique dans le domaine), *la déduction* (principes généraux sur le domaine), *la synthèse* (ensemble d'ontologies existantes, chacune fournissant une caractérisation partielle du domaine) et *la collaboration* (des points de vue multiples avec possibilité de couplage entre ontologies).

a) 1ère étape: extraction de termes et analyse

L'un des premiers objectifs à atteindre lorsque l'on construit une ontologie est de définir les primitives du domaine, tout en sachant qu'il n'existe pas de primitives dans un domaine. Les primitives, comme les termes de tête d'un thésaurus, sont arbitraires. Il faut par conséquent, modéliser les primitives nécessaires à la formalisation et à la représentation du problème à résoudre et des connaissances s'y rapportant. La construction des primitives relève du choix du concepteur, mais comment procéder? Pour y parvenir, Bruno Bachimont propose de repartir de l'expression linguistique des connaissances du domaine en utilisant l'extraction de termes à partir d'un corpus spécialisé. En effet, un corpus (constitué de documents propres au domaine concerné) comporte l'expression des notions qu'il faut modéliser. On peut ainsi construire un corpus textuel qui sera la source privilégiée permettant de caractériser les notions utiles à la modélisation d'une ontologie et le contenu sémantique qui lui correspond. Pour ce faire, B. Bachimont utilise ce qu'il appelle «une démarche corpus» et des outils terminologiques pour commencer la modélisation du domaine. Ces outils, pour la plupart, reposent sur la recherche de formes syntaxiques particulières manifestant les notions recherchées comme des syntagmes nominaux pour des candidats termes, des relations syntaxiques marqueurs de relations sémantiques, ou des proximités d'usage - comme les contextes partagés - pour des regroupements de notions.

b) 2ème étape: la normalisation sémantique

À la fin de la première étape, nous avons donc une liste de candidats termes dont les libellés ont un sens pour le spécialiste du domaine. Mais rien n'assure que ce sens soit unique: au Contraire argumentent Jean Charlet, B. Bachimont, et Raphaël Troncy [Charlet et al., 04] car nous sommes dans un fonctionnement linguistique où les significations sont ambiguës, les définitions circulaires dépendent en particulier du contexte interprétatif des locuteurs. Or, dans la modélisation ontologique, on cherche à construire des primitives dont le sens ne dépend pas des autres primitives et est surtout non contextuel. Il faut dès maintenant prendre le chemin du formel en normalisant les significations des termes pour ne retenir, pour chacun d'eux, qu'une seule signification, qu'une seule interprétation par un être humain. C'est ce que permet l'utilisation de la sémantique différentielle, proposée par Bruno Bachimont. Cette sémantique, issue notamment des travaux de François Rastier [Rastier, 87] [Rastier *et al.*, 94], permet de décrire les unités entre elles par les identités qui les unissent et les différences qui les distinguent. Cela donne lieu à une définition de l'unité selon quatre principes différentiels:

Le principe de différence avec **le père:** toute unité se distingue de l'unité parente, sinon il n'y aurait pas lieu de la définir. Il faut donc expliciter la différence qui la distingue de l'unité parente. Il s'agit du principe aristotélicien de définition par la différence spécifique.

Le principe de communauté avec **le père:** toute unité se détermine par l'identité qu'elle possède avec l'unité parente. Il faut expliciter en quoi l'unité fille est identique à l'unité parente. Il s'agit du principe aristotélicien de définition par le genre proche.

Le principe de différence avec **les frères:** toute unité se distingue de ses frères, sinon il n'y aurait pas lieu de la définir. Il faut donc expliciter la différence de l'unité avec chacune des unités sœurs.

Le principe de communauté avec **les frères:** toutes les unités filles d'une unité parente possèdent, par définition, un même trait générique, celle qu'elle partage avec l'unité parente. Mais il faut établir une autre communauté entre unités filles: celle qui permet de définir des différences mutuellement exclusives entre les unités filles. Bruno Bachimont donne cet exemple: l'unité parente est *être humain,* et les unités filles sont *homme* et *femme.* Ces unités partagent le fait d'être des humains. Mais cette propriété ne permet pas de définir en quoi les hommes et les femmes sont différents. On choisit alors comme principe de communauté la sexualité, on peut attribuer à *homme* le trait masculin, et à *femme* le trait féminin. Ces deux traits sont mutuellement exclusifs, car ce sont deux valeurs possibles d'une même propriété.

À la fin de cette étape, nous avons un arbre de primitives conceptuelles valable dans la seule région du monde modélisée où les concepts retenus correspondent bien à ceux de l'ontologie, par définition décontextualisée. Nous avons ce que B. Bachimont appelle une ontologie régionale.

c) 3ème étape: l'engagement ontologique

L'engagement ontologique correspond à l'évolution de l'ontologie régionale vers une ontologie formelle. La sémantique formelle ne considère plus des notions sémantiques mais des extensions, c'est-à-dire l'ensemble des objets qui vérifient les propriétés définies en intension dans l'étape précédente, propriétés ayant une définition formelle à ce niveau. La structure est alors un treillis

Ce treillis de concepts doit être vu comme la possibilité de créer des concepts dits définis en combinant les concepts primitifs.

d) 4ème étape: l'opérationnalisation

Il s'agit de la dernière étape de la création d'une ontologie, elle est généralement le fruit du travail d'un logiciel. Elle débouche sur la manipulation de l'ontologie par une machine. Elle devient un objet informatique. Enfin, cette étape consiste en la représentation de l'ontologie dans un langage de représentation de connaissances, ce qui nous amène directement à nous intéresser aux principaux langages de connaissances utilisés pour la création des ontologies formelles.

1.8 Langages de représentation d'ontologies

Il convient de rappeler que dans le domaine des ontologies, les travaux relatifs à la création de langages utilisés et acceptés par tous ont atteint un certain niveau de maturité. De ce fait, nous avons aujourd'hui un certain nombre de langages tels que *RDF* ou *OWL* qui ont fait l'objet de propositions de la part de l'organisme en charge de la coordination des actions liées au *Web sémantique.* Par ailleurs, il existe une multitude d'outils de représentation d'ontologies variant selon les techniques utilisées et les objectifs affichés par leurs développeurs.

1.9 Outils de représentation d'ontologie

Nous disposons déjà de nombreux outils et environnements de construction d'ontologies [Mizoguchi, 04]. Il convient de rappeler que tous ces outils offrent des supports pour le processus d'ontologisation, mais peu d'entre eux offrent une aide à la conceptualisation.
Dans cette section, nous abordons Protégé 2000 un des outils utilisés dans la création d'ontologies.

Protégé a été développé par le *Stanford Medical Informatics* de l'université de médecine de Stanford depuis 1995. Il est construit autour d'un modèle de connaissances inspiré par le paradigme des frames: classes, slots (attributs) et facets (contraintes sur les attributs) qui sont les primitives de modélisation proposées. Ce modèle autorise une liberté de conception importante, puisque le contenu des formulaires de spécification des classes peut être modifié suivant les besoins, *via* un système de méta-classes, qui constituent des sortes de « patrons» pour les classes du modèle du domaine. Il est adapté à la construction d'ontologies depuis la version Protégé 2000. L'interface très complète ainsi que l'architecture logicielle bien pensée permettant l'insertion de plugins, notamment des plugins pour gérer les représentations sous forme graphique, par exemple OWLViz, ont grandement contribué au succès de Protégé. En quelques années, cet éditeur s'est imposé comme la référence, avec une communauté d'utilisateurs extrêmement importante et active. Ses nombreuses extensions lui permettent en particulier de gérer des langages standards comme RDF et surtout OWL, de créer des axiomes formels de manière intuitive, d'accéder aux ontologies par des interfaces graphiques évoluées, de comparer et fusionner des ontologies avec la suite PROMPT• Il est également possible de faire fonctionner des raisonneurs, comme RACER *(Renamed ABox and Concept Expression Reasonner)* pour le langage OWL par exemple, pour vérifier la cohérence et la consistance de la structure ontologique.

Partie 2 : ONTOLOGIE SONELGAZ

2. 1 Une ontologie Sonelgaz

Sur la base de ce qui est donné sur le processus de mise en oeuvre d'une ontologie, nous avons développé une ontologie dans le domaine de l'énergie électricité et gaz. Dans ce contexte, nous allons présenter les éléments de ce modèle. Nous détaillons également les problèmes liés au **multilinguisme et décrivons les solutions adoptées**. Nous avons utilisé *Protégé 3.4* comme outil de modélisation, un outil aujourd'hui largement utilisé ; c'est un éditeur d'ontologies en code source libre développé en Java à l'université de Stanford (*Protégé ontologie Editor*). Les captures d'écran utilisées pour illustrer cet article ont été créées à partir de *Protégé*.

Il est essentiel de représenter les concepts en minimisant les biais en faveur d'une langue ou famille de langues donnée. C'est-à-dire que, dans la mesure du possible, nous considérons le sens indépendamment de sa réalisation dans une langue particulière. Chaque langue serait donc capable d'exprimer les concepts du domaine pour lesquels elle a des lexicalisations et les concepts pour lesquels elle n'en a pas. Une terminologie qui traduirait simplement les termes dans une langue donnée. Ce modèle devrait, permettre non seulement de prendre en compte des concepts existants dans différentes langues (et, donc, dans différentes cultures), mais aussi de représenter les relations lexicales à la fois à l'intérieur d'une même langue et entre plusieurs langues. Cela permettrait d'établir des équivalences lexicales précises (p. ex. synonymes, traductions), de traiter de manière efficace les termes et concepts et d'optimiser la valeur de l'ontologie pour un grand nombre d'applications.

Les trois niveaux de représentation que nous souhaitons exprimer dans ce modèle sont :

- **Les concepts** (la signification abstraite), par exemple ENERGIE au sens de l'énergie ;
- **Les termes** (formes lexicales spécifiques à une langue), par exemple 'Electricité', 'Electricity', 'الكهرباء'.

Chaque terme est une entité distincte dans chaque langue qui peut être liée à des concepts ou à d'autres termes et à d'autres variantes du même terme.

Ces distinctions nous permettent d'établir les relations hiérarchiques suivantes :

Concept vers terme : *haslexicalization*
(Lie les concepts à leurs réalisations lexicales)
Terme vers sa variante : *hasacronym, hasspellingvariant, hasabbreviation*
(Lient les termes à leurs variantes).

Les variantes de terme ne constituent pas de nouveaux termes mais sont les formes variables du même terme.

Les relations internes à un niveau existent tant au niveau du concept qu'au niveau du terme

Concept vers concept : *is_a* (indique une hiérarchie)

Terme vers terme : *is_synonym_of, is_translation_of*

Nous présentons un modèle OWL qui permet de rendre compte des distinctions conceptuelles et lexicales présentées ci-dessus. La conception des multiples niveaux de représentation lexicale présentés dans cette partie (classes, propriétés, annotations) est donc effectuée dans la version d'OWL connue sous le nom de OWL DL.

2.2 Le modèle de base

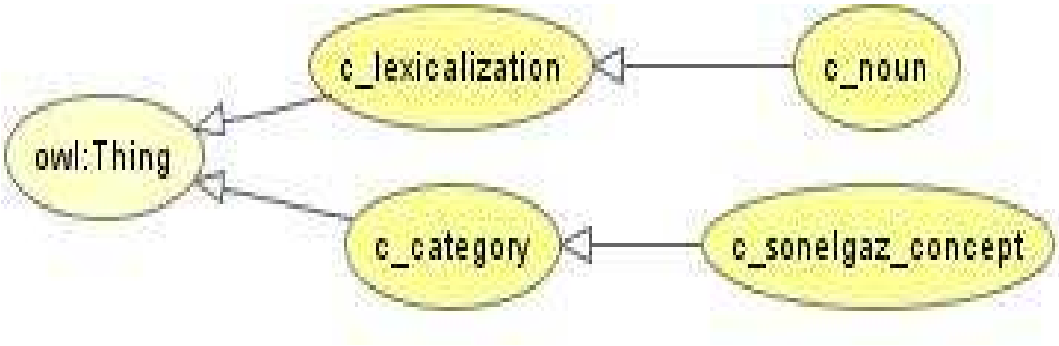


Figure 4-3 : le modèle de base

Le nouveau modèle OWL est fondé sur deux concepts au sommet de la hiérarchie, comme indiqué dans la Figure 4-3. Chaque entité d'une ontologie OWL a un URI (identifiant uniforme de ressource) unique. Dans la Figure 4-3, seule la dernière partie identifiante de l'URI est visible. Chaque URI d'entité dans notre modèle est composé d'un préfixe, c_ (pour les classes), r_ (pour les relations/propriétés), i_ (pour les instances), suivi d'une séquence numérique ou alphanumérique.

La racine de tous les concepts du domaine est le concept c_sonelgaz_concept. Ce noeud a toutes les caractéristiques structurelles de l'ontologie de domaine, à savoir une hiérarchie de classe avec des classes et leurs instances, leurs relations, propriétés, axiomes, contraintes et annotations afférentes au domaine de connaissance. Tous les termes du thésaurus ou plus précisément les descripteurs seront modélisés à l'intérieur de ce noeud.

La classe *c_sonelgaz_concept* est modélisée comme une sous-classe de *c_category*, ce qui implique que tous les concepts du domaine sont également potentiellement des catégories. La classe distincte *c_category* a été créée pour pouvoir prendre en compte des catégories spécifiques qui ne sont pas des concepts de domaine.

Alors que la structure centrale de l'ontologie de domaine est modélisée sous *c_sonelgaz_concept*, les lexicalisations de ces concepts apparaîtront comme des instances de la classe *c_lexicalization*. Cette approche a été choisie plutôt que l'utilisation du rdfs : label pour chaque concept afin de représenter sa lexicalisation dans une langue particulière. Elle permet de gérer la question du multilinguisme. La modélisation des lexicalisations comme des concepts distincts permettra d'établir des relations entre les différentes lexicalisations qui décrivent un concept. Elle fournira ainsi une sémantique plus riche.

2.3 La structure centrale hiérarchique

Dans un premier temps, les termes du domaine de l'électricité et de gaz (plus précisément ses principaux descripteurs) constitueront la structure centrale hiérarchique du modèle. Tous les descripteurs de Sonelgaz seront modélisés comme des sous-classes de *c_sonelgaz_concept* en utilisant le code du terme Sonelgaz pour former un URI de classe (par exemple FSM pour le concept 'FACTURATION SUR MEMOIRE') Figure 4-4. Les relations du thésaurus traditionnel Terme spécifique et Terme générique sont ensuite traduites en relations superclasse OWL et sous-classe OWL.

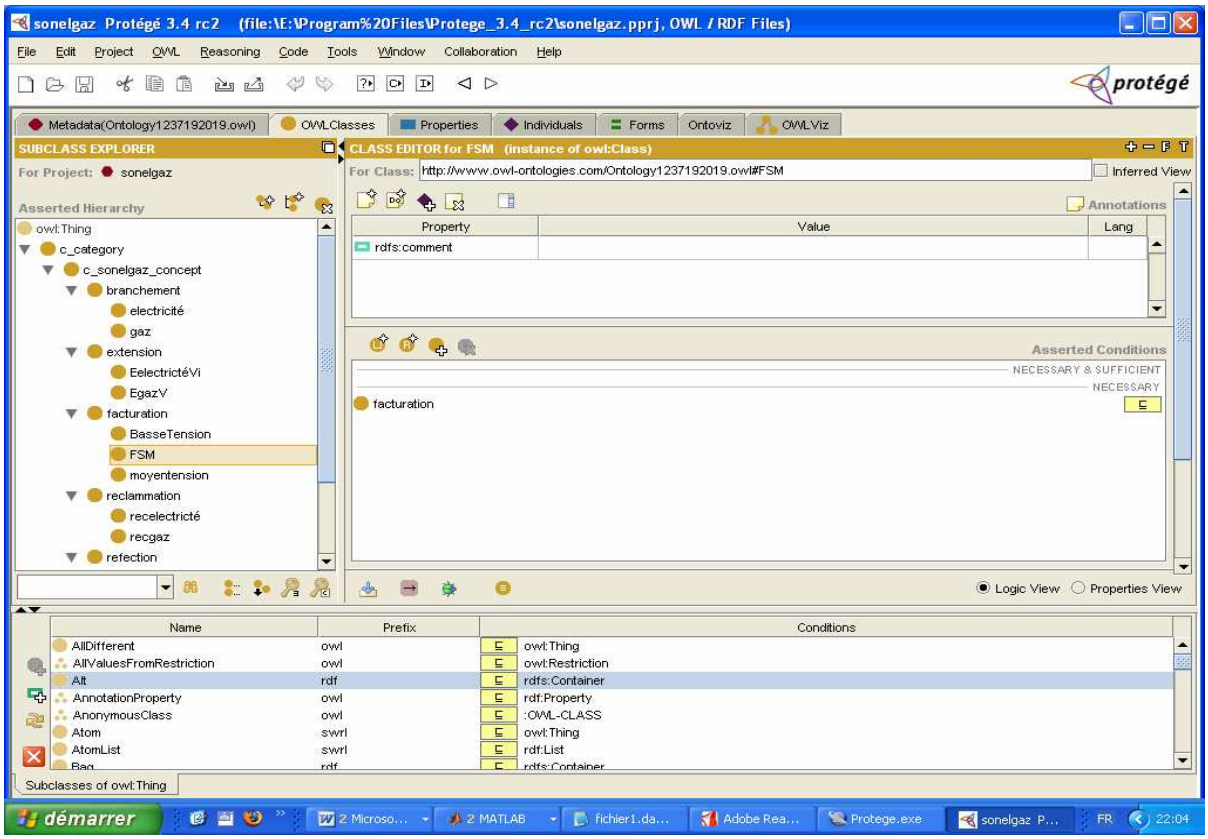


Figure 4-4 : Présentation des concepts de l'ontologie SONELGAZ

2.4 Les lexicalisations

Nous devons à présent introduire les lexicalisations afin de représenter cette structure en plusieurs langues. Toutes ces informations lexicales sont incluses dans le concept *c_lexicalization*. Chaque terme (c'est-à-dire chaque lexicalisation ou mot) qui décrit un concept dans une langue spécifique est modélisé comme une instance de ce concept.

L'instance URI est composée de i_ suivi du code de langue à deux lettres de ce terme, suivi par le terme lui-même (en utilisant des tirets bas pour remplacer les espaces et les caractères spéciaux). Si une forme particulière d'un mot s'avère avoir un homonyme dans une langue, un tiret bas est ajouté, suivi d'un chiffre, L'annotation rdfs : label est utilisée pour fournir le label du terme à afficher. La Figure 4-5 montre une copie d'écran de Protégé avec quelques instances de *c_lexicalization*.

Les instances sont en réalité des instances de *c_noun*, un sous-*concept* de *c_lexicalization*. Cela laisse le modèle suffisamment ouvert pour inclure d'autres formes telles que des verbes, des adjectifs.

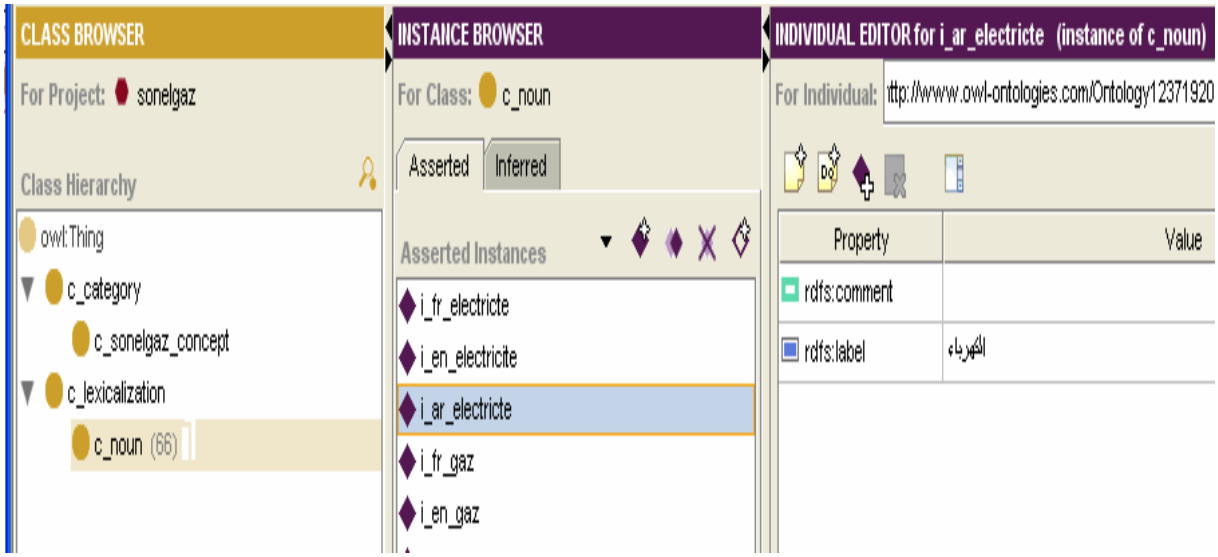


Figure 4-5 : Représentation de la désambiguïsation des termes et URI

La décision de traiter les termes comme des instances plutôt que comme des annotations (p. ex., rdfs : label) est principalement motivée par le fait que les relations en OWL DL peuvent seulement être définies entre deux individus ou entre un individu et un littéral. Afin de pouvoir également exprimer des relations entre les termes (telles que les relations de traduction et de synonymie), les termes doivent être réalisés comme instances. Nous allons présenter brièvement les relations terme à terme, mais auparavant nous allons expliquer la manière d'associer les termes aux concepts du domaine.

a) Associer les lexicalisations à des concepts : l'interface de gestion des relations Concept - Terme

Les termes sont associés au concept dont ils lexicalisent le sens via deux propriétés d'objets OWL, r_has_lexicalization et la relation inverse, r_means, représentées sur la Figure 4-6.

Nous avons modélisé les relations au niveau de c_category puisque nous traitons de la même façon les lexicalisations des catégories et des concepts du domaine. La classe *c_sonelgaz_concept* hérite des relations de *c_category*.

Chaque instance de *c_lexicalization* est liée à une et une seule instance de *c_category* ou de *c_sonelgaz_concept*. Une catégorie ou un concept de domaine sera généralement associé à plusieurs instances de *c_lexicalization* ; au moins une pour chaque langue dans laquelle il est disponible et d'autres pour les synonymes et noms scientifiques.

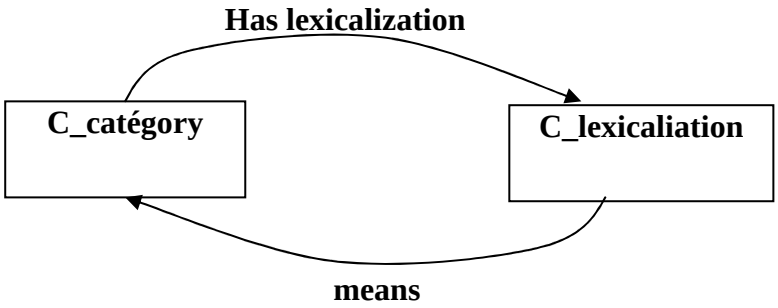


Figure 4-6 : Associer des termes à des concepts

b) Associer les lexicalisations : l'interface de gestion des relations entre termes

Afin d'associer deux termes (ou lexicalisations) entre eux, nous introduisons la propriété r_has_related_term. Cette propriété est la super propriété de toutes les relations entre termes. Il est important de noter que cette relation NE correspond PAS à la relation terme associé des thesaurus classiques, puisqu'elle décrit une relation conceptuelle, pas une relation entre termes. Nous avons initialement identifié deux associations possibles entre termes. Un terme peut avoir :

• Une ou plusieurs traductions ;
• Un ou plusieurs synonymes par langue ;

La Figure 4-7 présente la hiérarchie des propriétés telle qu'elle est modélisée dans Protégé. Le domaine et le type OWL de toutes les propriétés sont définis au niveau de c_lexicalization. r_has_synonym et r_has_translation sont des relations symétriques.

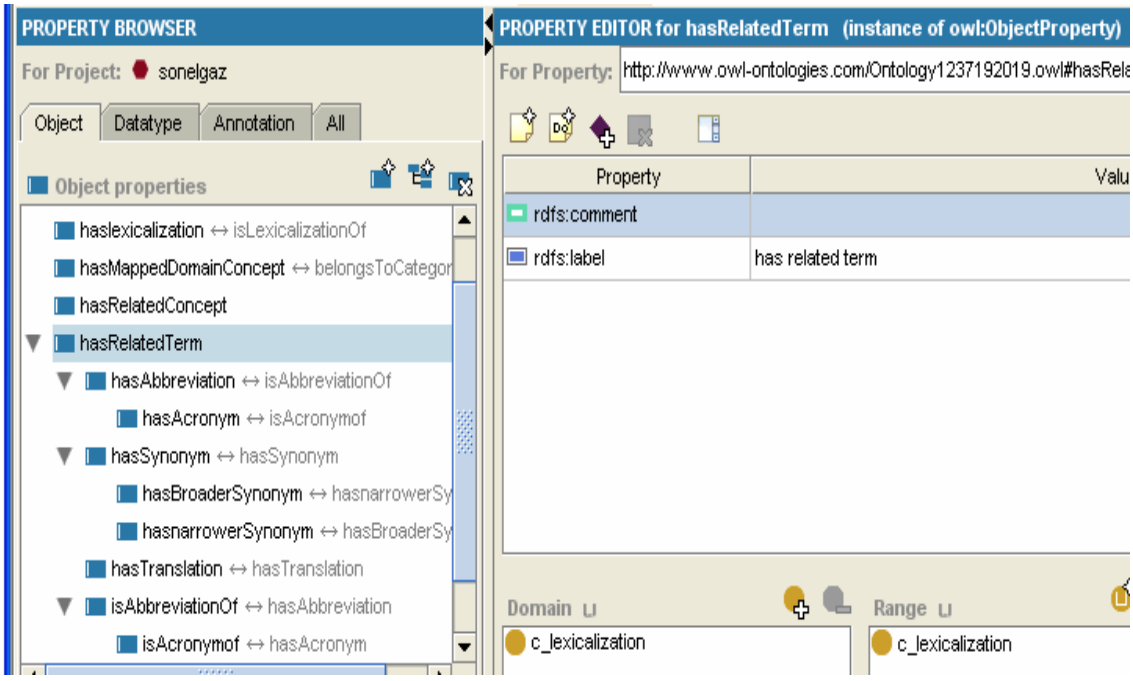


Figure 4-7 : L'organisation hiérarchique des propriétés de termes

c) Gérer les variantes de termes : l'interface de gestion des relations entre termes et variantes

Les termes eux-mêmes peuvent être représentés de différentes manières. Par exemple, le terme Facturation sur mémoire a les variantes suivantes :
• Administration (forme raccourcie) ;
• FSM (abréviation) ;
• Facturation sur mémoire (nom officiel).

Un terme est associé à ses variantes grâce à des propriétés de type de données telles que rdfs : label et à des propriétés spécifiques telles que has acronym, spelling variant et abbreviation. Suivant notre organisation hiérarchique des propriétés, nous avons modélisé ces relations comme des sous-propriétés de la propriété de type de données r_has_term_variant.

Le domaine de ces propriétés est défini dans c_lexicalization, alors que leur type est une simple variante. Cela implique qu'aucune nouvelle relation ne peut être établie entre des acronymes, des abréviations ou des variantes orthographiques. Jusqu'à présent, nous n'avons pas considéré cela comme une limitation à l'expressivité lexicale de notre modèle.

d) Extraction de nouvelles instances

Elle a pour but d'extraire les méta-données qui permettront de représenter les ressources. L'extraction d'instances repose sur des techniques du domaine de l'extraction d'information. De nombreuses plate-formes telles que Gate [Cunningham et al., 02] permettent de définir des patrons d'extraction ou d'utiliser des techniques reposant sur le traitement automatique des langues. L'extraction d'instances de concepts peut se faire à partir de techniques d'extraction d'entités nommées, issues du domaine du traitement automatique des langues [Kiryakov et al., 04]. Une entité nommée est un nom ou syntagme nominal se rapportant à une entité comme, par exemple, une personne, une organisation ou une localisation [Chinchor et Robinson, 98]. Un procédé d'extraction d'instances est décrit dans [Kiryakov et al., 04]. Les entités sont extraites à partir d'une base de connaissance qui, à partir de ressources lexicales, permet la détection automatique des entités.

Partie 3: Projection des documents sur l'ontologie

3.1 Comment peut-on représenter des documents en utilisant une ontologie ?

Dans le but de représenter des documents, nous allons utiliser notre ontologie de domaine. On peut désigner le processus de représentation d'après la Figure 4-8 ce qui permettra de définir les étapes nécessaires pour avoir les vecteurs concepts:

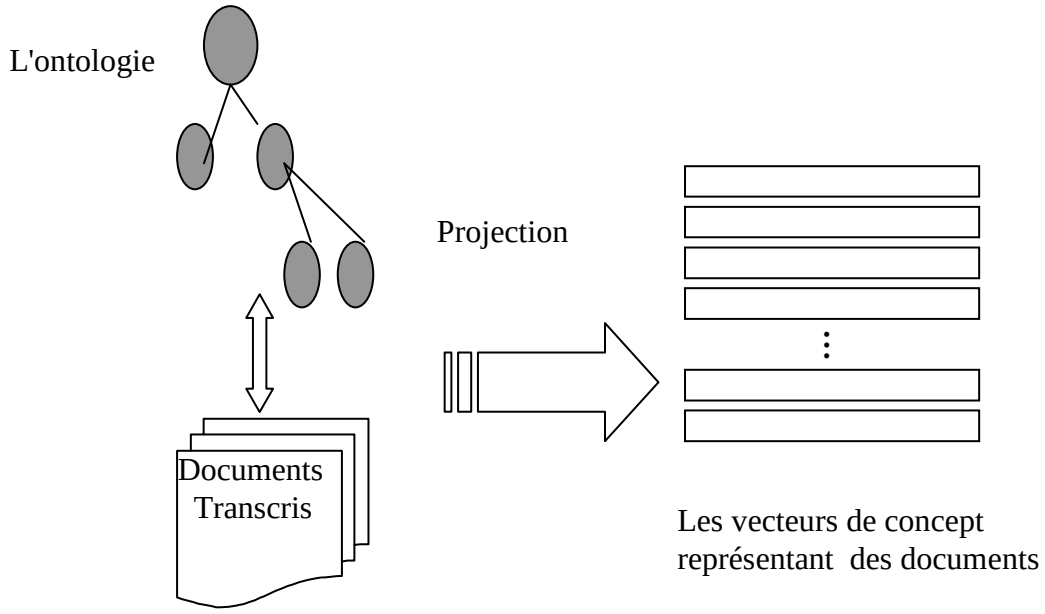


Figure 4-8 : Projection des documents sur une ontologie

3.2 Identification des concepts et des instances existant dans l'ontologie

Cette identification de concepts et d'instances dans les documents à partir d'une ontologie s'appuie sur différentes étapes :

a) Extraction des termes du document :

L'approche généralement suivie consiste à extraire des documents l'ensemble des termes y apparaissant et d'y rechercher les termes contenus dans l'ontologie. L'extraction de termes des documents se fait de la même façon que la recherche du langage de représentation classique. Les termes apparaissants dans un anti-dictionnaire peuvent être supprimés. Les expressions sont extraites soit statistiquement, soit syntaxiquement. L'extraction d'expressions est quasiment obligatoire, car les labels des concepts sont souvent composés de ce type d'éléments.

b) Recherche des termes correspondants à des concepts ou instances de l'ontologie :

Les labels ou termes désignant les concepts ou instances sont recherchés dans les documents. Un concept (et une instance de concept) est en effet défini à partir d'un ou plusieurs labels représentant les variantes lexicales que peuvent prendre les termes définissant les concepts [Vallet et al., 05] [Kiryakov, 04] [Guha, 03].

Les termes sont recherchés dans l'ensemble des termes extraits en favorisant la prise en compte des termes les plus longs et donc des concepts les plus spécifiques [Baziz et al., 05], [Vallet et al., 05]. Par exemple, dans le cas où les termes « Branchement », «Electricité », et « Branchement électricité » apparaissent dans le document, le label retenu et donc le concept correspondant, - sera Branchement électricité car l'expression formée de deux termes est plus précise que le ou les termes seuls. Plusieurs algorithmes ont été définis pour rechercher les termes les plus longs, ils consistent à faire varier la taille d'une fenêtre sur les mots de chacune des phrases des textes.

Pour chaque concept candidat formé en combinant les mots adjacents dans le texte, on interroge d'abord l'ontologie en utilisant les mots tels qu'ils se présentent dans le texte. S'ils ne correspondent pas à des entrées dans l'ontologie, on utilise leurs formes de base. Ceci permet de résoudre partiellement les problèmes liés aux variations de la morphologie des mots.

On peut s'interroger sur l'intérêt d'extraire les termes composés ou multi mots dans le texte des documents. A partir du moment qu'à l'issue d'une indexation classique par mots simples, on retrouve tous les mots y compris ceux formant des termes composés. En fait, cette démarche est importante dans la mesure où elle permet de réduire l'ambiguïté lors de l'affectation d'un terme à un concept de l'ontologie. En effet, les termes composés sont en général monosémiques (ils n'ont qu'un seul sens) même si les mots qui les composent peuvent être ambigus.

On peut noter ici l'intérêt d'utiliser une ontologie dont l'espace conceptuel possède une terminologie suffisamment vaste pour couvrir celle des documents de la collection.

c) Pondération des termes :

Une fois les termes extraits du document, il s'agit de leur affecter un poids qui détermine leur importance dans le document, plusieurs méthodes de pondération qui sont en général des variantes de *TF.IDF* sont utilisées. On peut se demander si ces méthodes sont valables aussi quand il s'agit de pondérer des termes composés de plusieurs mots. Nous avons donc utilisé une méthode de pondération qui tient compte de la longueur du terme (en nombre de mots) et du critère TF.IDF. Cette méthode de pondération *CF.IDF* s'appuie sur deux intuitions. La première fait l'hypothèse que les termes composés de plusieurs mots, sont plus riches sémantiquement que les mots qui les composent. Ainsi, "état de l'art" est plus précis que "état" et "art" pris séparément. La deuxième, suppose que les mots composant les termes peuvent renvoyer à ces derniers même lorsqu'ils sont utilisés isolément, après un certain nombre d'occurrences. Par exemple, dans un document traitant des bases de données, l'auteur peut utiliser simplement "base" pour désigner "bases de données". Ce qui représente une forme de simplification ou d'abréviation communément admise.

Dans cette méthode de pondération *CF.IDF*, chaque terme extrait représentera forcément un concept (noeud) de l'ontologie étant donné qu'on a utilisé l'ontologie pour les identifier. Pour un terme *K* composé de n mots, sa fréquence dans un document dépend du nombre d'occurrences du terme lui-même, et de celui de tout ses sous-termes dérivés. Formellement

$$cf(T) = count(T) + \sum_{sc \in sub\_terms(T)} \frac{Length(ST)}{Length(T)}.count(ST) \qquad \text{c.1}$$

Où *Length(T)* représente le nombre de mots dans *T* et *sub_terms(T)* le nombre de tous les sous-termes (qui doivent correspondre à leur tour à des concepts de l'ontologie) dérivés de *T* : sous-termes de n-1 mots de *c*, sous-termes de n-2,.. et tous les mots uniques de *T*.

Par exemple, pour le terme *"branchement moyen tension"* composé de 3 mots, sa fréquence est calculée comme suit :

f("elastic potential energy") *= count("branchement moyen tension") + 2/3 count("moyen tension") + 1/3 count("branchement ") + 1/3 count("moyen ") + 1/3 count("tension ").*

D'autres méthodes de calcul de fréquence sont proposées dans la littérature. Elles utilisent des analyses statistiques et/ou syntaxiques [Croft et Turtle, 91], [Huang et Robertson, 01]. Elles consistent en général à additionner les fréquences des mots uniques, les multiplier ou à multiplier le nombre d'occurrences du terme par le nombre de mots qu'il contient. Dans notre cas, nous avons utilisé la méthode de pondération locale de l'équation précédente. Le poids global d'un terme *T* dans un document d_j, *Weight(T,* d_j*)*, est alors calculée comme suit :

$$Weight\ (T, dj) = cf\ (T).\ln(N / df) \qquad \text{c.2}$$

N étant le nombre total de documents et *df* (document frequency) le nombre de documents où le terme *T* apparaît. Si le terme apparaît dans tous les documents, sa fréquence globale est nulle. Nous avons utilisé une fréquence seuil égale à 2 pour sélectionner les termes.

Notons ici que la pondération *CF.IDF* généralise le critère classique *TF.IDF*. En effet, dans le cas de mots simples (l'ensemble des sous termes, sub-terms est vide donc), les deux méthodes se confondent :

$$\sum_{sc \in sub\_terms(T)} \frac{Length(ST)}{Length(T)}.count(ST) = 0 \qquad \text{c.3}$$

Une fois les termes importants extraits du document, ils sont utilisés pour construire le noyau sémantique de ce document. Comme chaque terme extrait peut avoir plusieurs sens, donc correspondre à plusieurs concepts ou noeuds dans l'ontologie, des mesures de similarité entre les différents sens des termes sont calculées en vue de sélectionner, pour chaque terme, le meilleur sens correspondant dans l'ontologie. La mesure de similarité entre deux noeuds représente une valeur condensée résultant de la comparaison de deux sens possibles pour deux termes (donc deux concepts candidats) en utilisant la distance entre les positions des deux concepts candidats dans l'ontologies ou encore les relations sémantiques de l'ontologie. Cette valeur n'a pas de sens précis mais exprime le degré du lien entre les deux concepts candidats. Nous l'explicitons dans la section suivante.

d) Désambiguïsation des termes dans le modèle *DOCCORE* :

Les termes peuvent cependant se rapporter à plusieurs concepts. Dans ce cas, un mécanisme de désambiguïsation du terme est mis en place afin d'identifier quel est le concept abordé dans le document. Il existe un grand nombre de techniques de désambiguïsation [Sanderson, 00].

Dans cette section on va utiliser un algorithme du modèle DocCore [Baaziz et al., 05] ou des mesures de proximité sémantique ont été proposées dans la littérature (une douzaine) utilisant des structures de réseaux sémantiques ou hiérarchiques. Ses mesures sont soit basées sur le chemin (path based measures) entre les deux concepts à comparer telles que définies par exemple dans [Rada et al., 89] [Leacock et al., 98] [Jiang et al., 97], sur la notion de contenu d'information (Information Content ou IC) telle que définie par Wu et Palmer [Wu et al., 94] et Resnik [Resnik, 99], sur une combinaison du chemin et du contenu d'information [Lin, 98] ou sur l'algorithme de Lesk .

Nous décrirons la mesure de Patwardhan [Patwardhan et al., 03] que nous désignerons parfois par Lesk adapté telle que nous l'avons utilisée dans la suite.

Formellement, étant donné un ensemble de relations de l'ontologie $R = \{R_1, R_2, ...R_n\}$ et deux termes T_k et T_l auxquels sont affectés deux sens α et β : C_α^k et C_β^l. La similarité sémantique entre C_α^k et C_β^l, notée P_{kl} (C_α^k, C_β^l) ou *overlaps* (C_α^k, C_β^l) est définie comme suit

$$P_{Kl}(C_\alpha^k, C_\beta^l) = \sum_{i,j \in \{1...,n\}} R_i(C_\alpha^k) \cap R_j(C_\beta^l) \qquad \text{d.1}$$

Elle représente l'intersection de Lesk adaptée (nombre de mots communs qui est élevé au carré dans le cas des mots successifs) entre les informations retournés par les relations R_i, quand elles sont appliquées aux concepts C_α^k et C_β^l ($R_j(C_{\alpha 1}^k), R_j(C_\beta^l)$). Les informations que les relations retournent, peuvent être des phrases (quand il s'agit de définitions ou glossaire du concept), ou un ensemble de termes décrivant le concept de manière alternatives (termes synonymes).

L'utilisation de ce nombre relativement élevé de relations a pour but de couvrir au maximum les différents types de liens que deux concepts peuvent partager. On peut ainsi détecter des liens non explicites entre concepts.

A l'issue de cette étape, on obtient des valeurs de proximité sémantiques entre tous les concepts candidats. Les concepts candidats étant les différents sens possibles (ou entrées possibles dans l'ontologie) pour lesquels un terme extrait peut être affecté.

e) Choix du concept :

La dernière étape de l'approche concerne la construction du meilleur réseau sémantique qui représente au mieux le contenu du document. A cette étape, nous connaissons l'ensemble des termes noté :

$$D_T : D_T = \{T_1, T_2, \ldots, T_m\} \qquad \text{e.1}$$

L'ensemble des termes est sélectionné suivant la phase d'extraction de termes. Comme évoqué précédemment, les termes extraits peuvent être composés de mots uniques ou multiples de plus, chaque T_i de l'ensemble D_T, peut avoir un certain nombre de sens noté S_i, représentés par les instances de notre ontologie :

$$S_i = \{C_1^i, C_2^i, \ldots\ldots, C_n^i\} \qquad \text{e.2}$$

Le terme T_i *a alors* |Si |=n sens, il peut donc représenter n concepts, c'est-à-dire, n entrées différentes dans notre ontologie. Si nous choisissions un sens pour chaque terme de D_T, nous aurions toujours un ensemble *SN(j)* de *m* éléments, car nous sommes sûrs que chaque terme de D_T a au moins un sens étant donné qu'il appartient à l'ontologie. Nous définissons les noeuds d'un réseau sémantique, *SN(j)* comme suit :

$$SN(j)=\{C_{j1}^1, C_{j2}^2, C_{j3}^3, \ldots, C_{jm}^m\} \qquad e.3$$

Il représente la j^{ème} configuration des sens des termes de D_T. $j_1, j_2, .., j_m$ désignent des indexes de sens pris entre un et le nombre de sens possibles pour respectivement les termes T_1, $T_{2,..,}$ T_m. Concernant les *m* termes de D_T, plusieurs réseaux sémantiques peuvent être construits en utilisant toutes les combinaisons possibles de sens. Le nombre total de réseaux sémantiques possibles *Nb_SN*, dépend du nombre de sens que chaque terme de D_T peut avoir :

$$Nb\_SN = |S_1| \,.\, |S_2| \ldots .|S_m| \qquad e.4$$

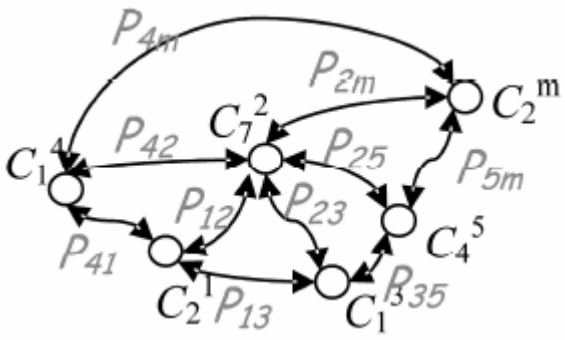


Figure 4-9 : Exemple de réseau sémantique construit à partir d'une configuration de concepts candidats.

Par exemple, la Figure 4-9 représente un réseau sémantique possible contenant les noeuds $\{C_2^1, C_7^2, C_1^3, C_1^4, C_4^5, C_2^m\}$ résultant de la combinaison du 2ème sens du premier

terme T_1, du 7ème sens de T_2,.., du $2^{ème}$ sens de T_m (nous supposons que les liens nuls ne sont pas représentés). Les liens entre les concepts ou noeuds (P_{ij}) représentés dans la Figure 4-9 représentent les valeurs (réelles) de proximité sémantique entre les noeuds. Elles sont calculées en utilisant des mesures de similarité.

Le problème auquel on est confronté maintenant est comment choisir, parmi les différentes configuration possibles (*Nb_SN* configurations*)* de réseaux sémantiques, celle qui représente au mieux le contenu du document. Pour ce faire, nous prenons le problème dans le sens inverse : on suppose que l'on a le meilleur réseau et on cherche à identifier chacun de ces noeuds. L'hypothèse à faire sur les noeuds est que, chaque noeud du meilleur réseau à construire, doit représenter l'entrée adéquate dans l'ontologie, c'est à dire, le sens approprié pour le terme correspondant. Ceci passe par une désambiguïsation des termes extraits à l'étape a, en se servant de l'information sur la proximité sémantique entre les sens des termes calculée à l'étape d.

Le principe de cette désambiguïsation consiste à supposer que, parmi les différents concepts candidats pour un terme donné, le plus adéquat (vraisemblable) est celui qui a le plus de lien avec les autres concepts du même document que lui. En répétant cette règle à tous les éléments de D_T, on se retrouve avec des termes qui se désambiguïsent mutuellement et de manière globale, par rapport au contexte du document.

Pour formaliser cette idée, on affecte à chaque concept candidat un score (*C_score*). Le score d'un concept candidat est égal à la somme des valeurs de similarité qu'il a obtenu avec les autres concepts candidats, sauf ceux qui sont dans le même ensemble de sens que le sien : pour un terme T_i, le score de son $k^{ème}$ sens est alors calculé comme suit :

$$C\_score(C_k^i) = \sum_{\substack{l\in[1..m],i<>i \\ j\in[1..n]}} P_{i,l}(C_k^i, C_j^l) \qquad \text{e.5}$$

Sachant que m représente le nombre de termes de D_T et n le nombre de sens qui est propre à chaque T_i comme défini dans l'équation (e.5).

Le meilleur concept C_i qui représente au mieux le sens du terme T_i est celui qui maximise *C_score* :

$$C_i = Best\_score(T_i) = \underset{k=1..n}{ArgMax} \left\| C\_score(C_k^i) \right\| \qquad \text{e.6}$$

Le concept ainsi sélectionné, représentera un noeud dans le noyau sémantique.

Le noyau sémantique final du document est $\{S_{j1}^{1}, S_{j2}^{2}, S_{j3}^{3}, \ldots, S_{jm}^{m}\}$, sachant que les pondérations des noeuds correspondent respectivement à (*Best_Score(T_1), Best_Score(T_2), Best_Score(T_3), Best_Score(T_m)*). Les valeurs des liens : $\{P_{12}, P_{13}, .., P_{ij}, P_{m-1, m}\}$ sont connues car déjà calculées dans la phase précédente.

3.3 Projection de document sur l'ontologie :

Dans l'absence du code source de l'algorithme de DocCore, on propose de suivre les étapes suivantes pour l'obtention du vecteur concepts d'un document :

1. **Corpus =D_1,D_2……..D_n où n est le nombre des documents du corpus.**
2. **Chaque Document de notre corpus :Di ={T_1,…..T_m} où T c'est le terme et m nombre de terme sélectionnés pour ce document avec la formule CF.IDF**
3. **Chaque Di est représenté par un ensemble de concepts où chaque concept est le meilleur concept Ci qui représente au mieux le sens du terme Ti (application de l'algorithme DocCore)**
4. **chaque document est écrit sous forme de concept : Di= {C_1, C5, C1, C6……C_j} où le j est le nombre de concept correspond au nombre de terme du document Di**
5. **arranger l'ensemble des concepts de notre ontologie sous forme de vecteur, Ontologie = {$concept_1$, $concept_2$, $concept_3$……..$concept_{k}$} où k c'est le nombre de concepts de l'ontologie.**
6. **on va appliquer d'autres méthodes de calcul de fréquence en utilisant la quatrième présentation obtenue par le modèle DocCore.**

• **La mesure *cf.idf* $_{\text{DocCore}}$ *(Concept Frequency – Inverse Document Frequency) basé sur vecteur DocCore :***

$$cf.idf(d,k) = cf(d,k) * \log\left(\frac{N}{df(k)}\right)$$

$$cf(d,k) = \sum_{i=1}^{P} freq(Ti)$$

- ***cf(d,k)* est la fréquence du concept *k* dans le document *d* (la mesure locale)**
- ***freq(Ti)* est la fréquence du terme *i* du concept *k* dans le document *d, en utilisant le vecteur du deuxième point***
- ***N* est le nombre de documents de la collection**
- ***df(k)* est le nombre de documents où le concept *k* apparaît**
- $\log\left(\frac{N}{df(k)}\right)$ **est la mesure globale qui nous montre la relation du concept dans l'ensemble de documents.**

Soit un corpus comprenant 355 documents et soit par exemple le document D1, l'ensemble des termes extraits de ce document est :

1. D1= {branchement électricité, branchement gaz, facturation devis, adresse, wilaya……}
2. En utilisant l'algorithme DocCore on obtient une représentation de chaque terme du document D1 par son meilleur concept Ci qui représente au mieux le sens du terme T_i .

D1= {branchement, branchement, facturation, branchement, branchement, branchement …….}

3. Le vecteur concept représentant les concepts de notre ontologie est sonelgaz = {branchement, facturation, réclamation, réfection, extension, sécurité}. (voir Figure 4-4).
4. En appliquant la formule $cf.idf(d,k) = cf(d,k) * \log\left(\frac{N}{df(k)}\right)$ sur les vecteurs du point 2) on obtient D1={ 2*log(355/56),1*log(355/46),0,0,0,0}

3.4 La classification par l'ontologie

A la fin de ce processus chaque concept de notre ontologie à au moins une liaison avec un document de notre corpus, à base de cette représentation il ne reste qu'à rechercher une classification thématique de l'ensemble des documents en utilisant les vecteurs de concepts obtenus pour chaque document.

Nous avons utilisé une base qui comprend 355 documents. A partir des résultats qui découlent de l'algorithme ci-dessus, la présentation de l'ensemble des documents du corpus par des vecteurs de concept de notre ontologie forme le tableau suivant :

	Concept de l'ontologie					
docum	**branchement**	**facturation**	**réclamation**	**réfection**	**extension**	**sécurité**
D1	62,25	4.60	2.10	1.20	2.80	0.90
D2	62.94	1.60	0.90	1.50	4.30	5.10
D3	68.01	1.50	3.20	1.20	2.50	2.51
D4	40.37	1.50	1.50	2.00	3.50	4.00
D5	68.14	5.50	3.30	4.40	1.30	1.50
D6	62.61	4.20	5.40	4.60	0.20	8.80
D7	1.33	31.86	2.20	1.50	5.20	1.20
D8	3.25	60.96	3.00	8.20	2.50	2.20
⋮	⋮	⋮	⋮	⋮	⋮	⋮
D52	8,73	8,53	1,65	10,41	6,44	4,27
D53	56,53	1,55	8,82	8,96	2,91	14,97
D54	23,88	13,52	7,34	3,82	8,94	3,8
D55	78,82	8,03	2,7	6,68	4,51	4,92
D56	66,27	16,16	13,21	5,46	2,46	8,15
D57	86,74	8,07	7,18	4,73	5,55	1,77
D58	28,37	8,57	18,35	8,29	10,23	10,66

D59	62,05	19,37	16,7	9,33	6,01	8,05
D60	57,69	16,31	5,45	2,32	3,94	3,13
D61	35,08	5,94	0,41	8,05	4,88	24,18
D62	44,45	2,87	1,94	6,7	2,3	12,44
D63	51,42	1,15	14,74	4,91	5,76	2,98
D64	11,18	0,38	6,6	9,8	23,27	3,68
D65	40,5	3,13	0,47	0,78	6,24	0,75
D66	91,85	13,5	8,12	19,37	6,31	17,06
D67	16,48	5,03	7,16	12,85	20,45	25,87
D68	49,5	0,63	11,79	6,56	8,74	7,25
⋮	⋮	⋮	⋮	⋮	⋮	⋮
D355	6,02	4,24	2,45	5,37	4,02	44,72

Tableau 4-1 : Résultat de la projection des documents sur l'ontologie.

3.5 Conclusion sur l'approche

En appliquant cette approche on conclut que les caractéristiques de documents sont les concepts. Pour chaque document dans la collection, on compte l'apparition des termes représentant chaque concept de l'ontologie et on calcule le poids de ce concept en utilisant la mesure $CF.IDF_{DocCore}$. Chaque concept dans l'ontologie est représenté par plusieurs termes. Si un des sous concepts de concept A est apparu dans un document, on dit que le concept A est apparu aussi dans ce document. Le vecteur représentant un document est le vecteur des poids de concepts (vecteur concept).

Cette approche se concentre sur les éléments sémantiques de document. Grâce à chaque concept contenant un ensemble de termes, la dimension du vecteur représentant le document est plus petite que celle dans une autre méthode et elle est fixe (elle est égale le nombre de concept dans l'ontologie) Figure 4-10. Mais l'élément le plus intéressant est qu'on utilise une seule ontologie pour représenter les documents dans des langages différents.

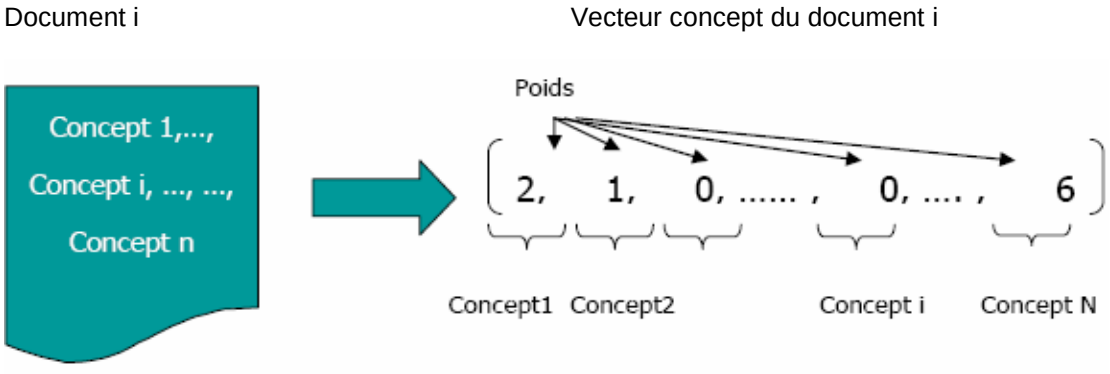


Figure 4-10 : Représentation d'un document par un vecteur des concepts

Chapitre 5:

Visualisation des résultats

1. Introduction :

Le résultat de la projection des documents sur l'ontologie, qui est un vecteur de concepts pondérés selon la méthode CF.IDF est utilisé comme entrée au réseau de neurone de type SOM, pour visualiser le résultat de la classification. En effet, bien que la projection d'un document sur l'ontologie nous fournit un vecteur qui renseigne sur le thème du document, mais la visualisation du résultat effectif de la classification, est réalisée grâce aux cartes de Kohonen.

Partie 1: La classification par les SOM

1.1 Introduction de l'algorithme de SOM

Teuvo Kohonen a pour la première fois présentée la SOM en 1982. Jusqu'à maintenant, la SOM a été largement appliquée dans beaucoup de domaines de l'informatique, comme le traitement d'images ou la fouille de données… Dans les années 90, Kohonen et ses collègues ainsi que quelques autres groupes de recherche indépendants ont appliqué les SOM à l'exploitation de données des textes. Dirigés par Kohonen, plusieurs projets de cette nature ont été mis en oeuvre pendant cette période. Avec WEBSOM, Kohonen et ses collègues ont montré que la SOM peut être employée pour regrouper les documents numériques d'une manière non supervisée. En outre elle peut aider l'utilisateur à visualiser les groupes de documents classifiés par la SOM. Dans cette section, nous reformulerons l'approche SOM avec les symboles et les conventions appropriés aux nos besoins et objectifs.

1.2 Structure de SOM

La carte auto organisatrice est un type de réseaux neuronaux à une couche, appelée la couche de sortie. Dans cette couche, des neurones sont organisés selon une certaine topologie où un neurone peut avoir 4, 6, 8, ou plus de voisins. Pendant le processus d'apprentissage, le nombre de voisins qu'un neurone a peut influencer la similitude entre les documents appartenant à ce neurone et les documents appartenant à ses voisins. Dans les cartes montrées ci-dessous, les points sur la grille représentent les neurones. Le neurone au centre du cercle a 4 voisins au maximum. Chaque neurone a deux directions de relations voisines, de sorte qu'il peut avoir deux voisins dans la direction verticale et deux voisins dans la direction horizontale. Les neurones dans les coins de la carte ont moins de voisins (2 voisins) ; les autres neurones dans les frontières de la carte ont 3 voisins. Au cas où un neurone aurait 4 voisins, ses neurones voisins forment un carré dont les sommets sont les voisins. Ce type de topologie s'appelle également la topologie carrée.

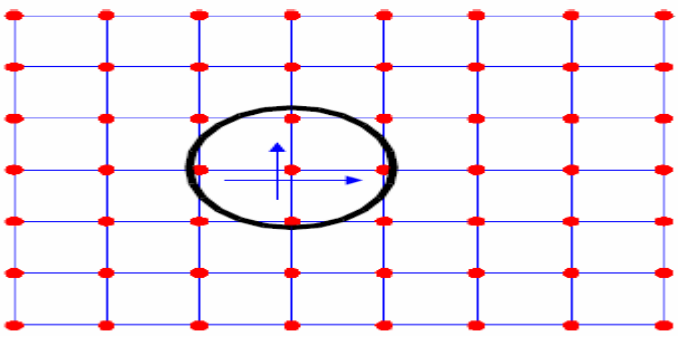


Figure 5-1 : Une 7x8 SOM de topologie carrée**.**

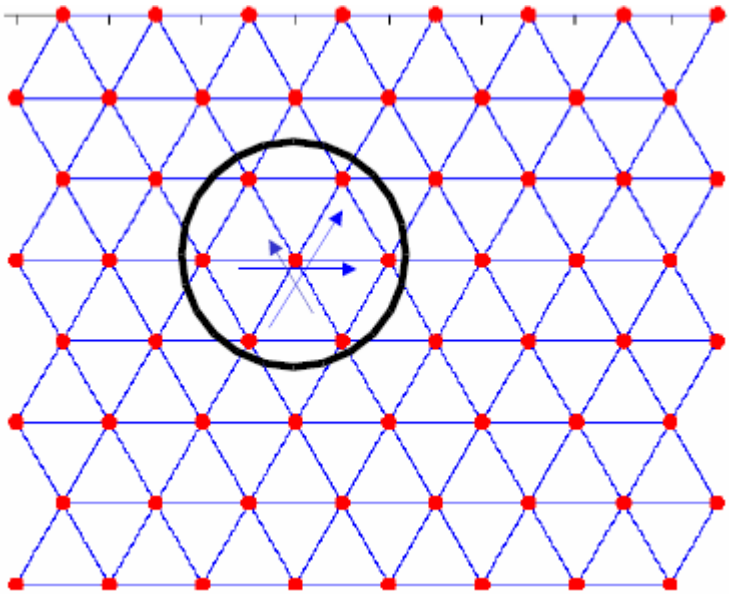


Figure 5-2: Une 8x8 SOM de topologie hexagonale.

La topologie hexagonale est un autre type de topologie où chaque neurone peut avoir trois directions de relation voisine : Ouest-Est, Sudouest-Nordest, et Sudest-Nordouest de sorte qu'un neurone a au plus 6 voisins.

Un neurone sur la carte a les caractéristiques suivantes :

• Sa position sur la carte. C'est important parce qu'elle peut déterminer le nombre de voisins qu'il peut avoir (voir les deux Figures ci-dessus). Dans une carte à deux dimensions on caractérise sa position comme une combinaison des coordonnées initialisées à la construction de la carte. Ensuite, la position ne sera jamais changée.

• Un vecteur représentant le neurone, appelé le vecteur de neurone. Ce vecteur a un nombre de dimensions égal à celui des vecteurs d'entrée. Les composants de ce vecteur, après avoir été formé (voir la section « apprentissage » ci-dessous), refléteront mieux les composants correspondants des vecteurs d'entrée appartenant à ce neurone. Il signifie qu'un neurone représente un groupe d'entrées semblables. Pendant le processus d'apprentissage, les composants d'un vecteur de neurone seront modifiés.

Le vecteur représentant un neurone a un nombre de dimensions égal à M, qui est le nombre de dimensions des vecteurs d'entrée. Si la carte a deux dimensions, le nombre de neurones dans chaque dimension est X et Y, avec X.Y=G (G est le nombre de groupes sur la carte). Le neurone C_k , *avec* $0 <= k < G$ peut être représenté par les index x, y : $C_{x,y}$,*avec* $0<=x<X$, $0<= y< Y$ *où* $k= x.X+ y$. Le vecteur de neurone correspondant au neurone C_k ou $C_{x\,y}$ est c_k ou $c_{x\ y}$, (c minuscules). Le j-ième composant dans le vecteur représentant le k-ième neurone est alors représenté par C_{kj} ou $C_{x\,yj.}$

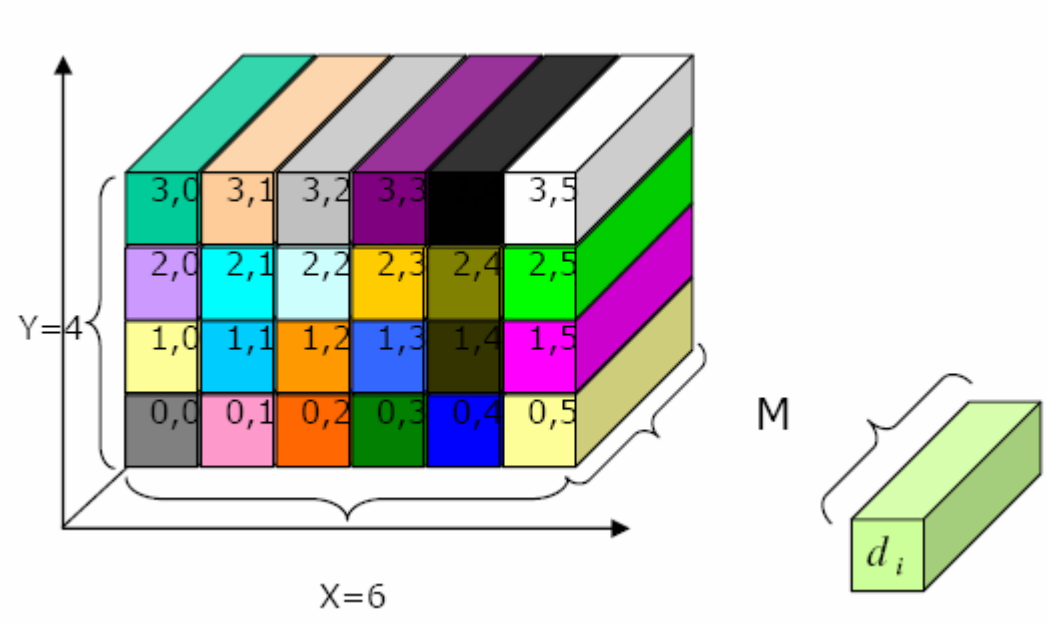


Figure 5-3: Un exemple d'une carte de topologie carrée. Les vecteurs de neurones ont M dimensions. La carte a deux dimensions avec 6 neurones verticaux et 4 neurones horizontaux. Chaque neurone a une couleur différente des autres. Ceci signifie que les vecteurs de neurones sont différents.

1.3 Apprentissage

Nous considérons maintenant le processus d'apprentissage de la SOM. Le but de ce processus est d'établir graduellement les vecteurs de neurones qui représentent le mieux les groupes d'entrées. D'une part, en raison de la particularité de l'algorithme de SOM, un vecteur de neurone sera très semblable aux vecteurs de ses voisins. Cette similitude est exprimée par les distances courtes entre eux dans l'espace de données.

Voici les étapes de ce processus :

i. Initialiser des vecteurs de neurones. Il y a beaucoup de méthodes possibles: l'initialisation aléatoire, l'initialisation basée sur K-moyennes… L'initialisation aléatoire est la plus utilisée parmi ces méthodes. Des composants des vecteurs de neurones seront affectés des valeurs numériques aléatoires.
ii. Pour la t-ième itération du processus d'apprentissage, choisir aléatoirement parmi N entrées une entrée d_i.
iii. Chercher le neurone ayant le vecteur de neurone le plus semblable à l'entrée d_i. Ce neurone s'appelle également le gagnant pour l'entrée d_i à la t-ième itération. Nous emploierons le terme w_t pour représenter le gagnant au moment t.

iv. Déterminer le voisinage du gagnant et mettre à jour les vecteurs de neurones à l'intérieur du voisinage.

v. Si la condition d'arrêt n'est pas satisfaite, continuer la prochaine itération (t=t+1) et répéter depuis le pas ii.

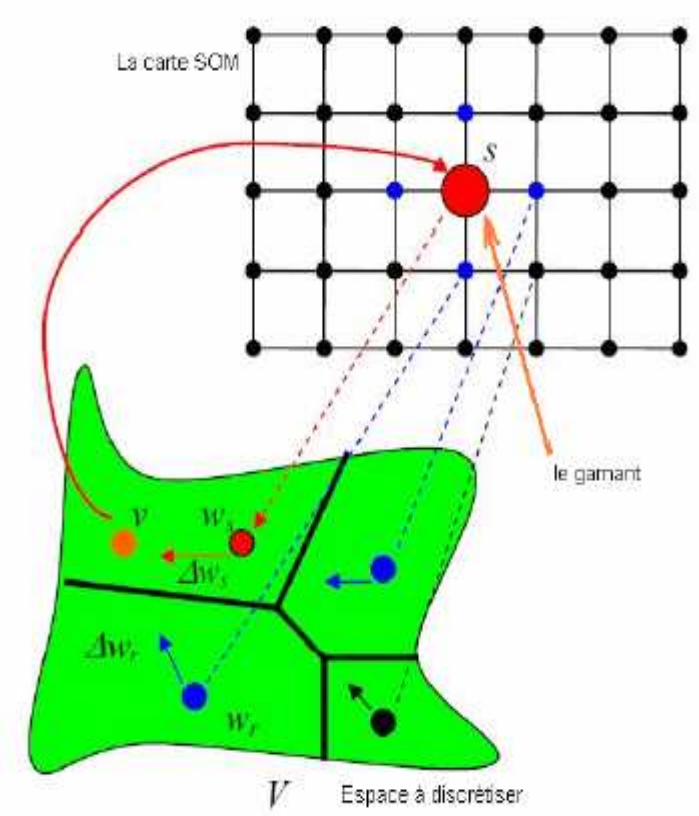


Figure 5-4 : Mettre à jour un vecteur neurone

L'entrée de ce processus est un ensemble de N vecteurs. Au début de ce processus, on initialise aléatoirement les vecteurs de neurones.

Dans ces 5 étapes, nous avons mis notre attention sur l'étape iv à cause de son importance. Considérons comment le vecteur de neurone c_k est mis à jour.

1.4 Comment peut-on mettre à jour des vecteurs de neurones dans la carte?

Supposons qu'on est la t-ième itération du processus. On a trouvé le gagnant wi. On veut mettre à jour le vecteur de neurone c_k. On calcule les éléments du vecteur de neurone c_k par la formule suivante :

$$c_k(t+1) = c_k(t) + \gamma(t) * h(c_k(t), w_t, t) * (d_i - c_k(t)) \qquad 1\text{-}4\text{-}1$$

$c_k(t+1)$ Est le vecteur de neurone à la (t+1) ième itération.

$c_k(t)$ Est le vecteur de neurone à la tième itération.

$\gamma(t)$ Est le taux d'apprentissage.

$h(c_k(t), w_t, t)$ Est la fonction du voisinage.

$(d_i - c_k(t))$ Est le vecteur d'entrer.

Dans cette formule, il y a deux distances qui doivent être calculées : distance locale et distance de données. La distance locale est la distance entre 2 neurones sur la carte, qui est habituellement un plan à deux dimensions. La distance entre deux neurones sur la carte est toujours fixe. Dans la formule 1-4-1, le terme qui concerne la distance locale est $\gamma(t) * h(c_k(t), w_t, t)$ qui est la multiplication du **taux d'apprentissage** $\gamma(t)$ et de la fonction du **voisinage** $h(c_k(t), w_t, t)$. Nous pouvons voir que ces deux termes sont des fonctions du temps. La forme de ces deux fonctions peut être déterminée par le programmeur en respectant les règles suivantes :

- **Le taux d'apprentissage :** Ce taux commence par une grande valeur et il va être diminué quand le temps augmente. Au début du processus d'apprentissage, les vecteurs de neurones sont initialisés aléatoirement. Sa position est la plus loin que la position optimale. À chaque étape, on doit déplacer (mettre à jour) les vecteurs de neurones à une autre position qui est plus optimale. Si la valeur de taux d'apprentissage est grande, la distance entre la position actuelle et la position optimale du vecteur de neurone est donc grande. Quand le temps augmente, ce taux est diminué. C'est-à-dire que les vecteurs de neurones sont plus proches de la position optimale. On peut calculer ce taux par la formule suivante :

$$\gamma(t) = \gamma(o) * e^{\frac{-t}{\varepsilon}} \qquad \text{1-4-2}$$

  - γ (t) est le taux d'apprentissage initial
  - ε est le coefficient prédéfini

- **La fonction de voisinage :** $h(c_k(t), w_t, t)$ commence par une valeur élevée puis diminue quand le temps augmente. Cette fonction vient du fait que dans le vrai système de neurone (dans le cerveau de l'animal), quand un neurone est excité, d'autres parmi les neurones autour de lui seront également excités. Ils forment un voisinage du neurone central et autour de ce voisinage il y a des neurones qui sont inhibés. Ce terme dépend strictement de la distance locale. Pour représenter ce phénomène, la fonction de voisinage doit avoir la forme d'un chapeau mexicain :

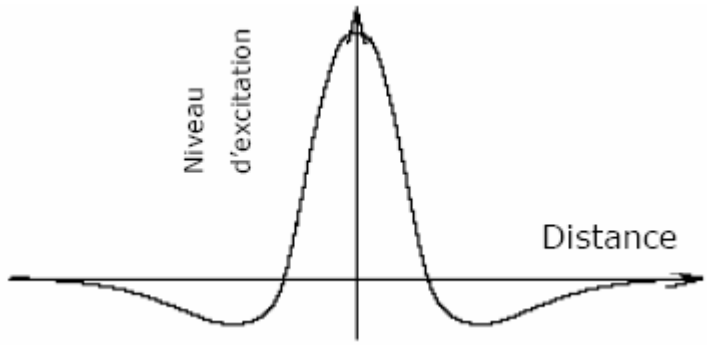


Figure 5-5: La forme du chapeau mexicain.

Généralement il y a beaucoup de propositions pour les formules de ces deux fonctions. Nous présentons ici les plus utilisées :

Le taux d'apprentissage $\gamma(t)$ **:** la formule 1-4-2

$$\gamma(t) = \gamma(o) * e^{\frac{-t}{\varepsilon}}$$

La fonction de voisinage $h(c_k(t), w_t, t) = e^{\frac{\|c_k - w_t\|^2}{2.\sigma(t)^2}}$ 1-4-3

ou, $\sigma(t) = \sigma(0) e^{\frac{1}{\varepsilon}}$, $\sigma(0)$ et ε sont les paramètres prédéfinis.

La distance de données dans la formule I.4.1 est calculée par le terme $(d_i - c_k(t))$ où d_i est un vecteur d'entrée (naturellement, il ne sera pas changé dans le processus d'apprentissage) et $c_k(t)$ est le vecteur de k-ième neurone à la t-ième itération.

1.5 Classification

Quand le processus d'apprentissage est terminé, nous nous intéressons au processus de classification. Actuellement, chaque vecteur de neurone a sa propre position optimale dans l'espace de données, de sorte qu'il puisse être le représentant pour un groupe de vecteurs d'entrée. L'objectif de la tâche de classification est de lier chaque vecteur d'entrée à son vecteur de neurone correspondant. Pour une entrée, nous trouverons son neurone en recherchant le vecteur de neurone ayant la plus petite distance au vecteur d'entrée. Puisqu'il y a beaucoup d'entrées mais seulement quelques neurones, plusieurs entrées tomberont dans un même neurone. On dit que ces entrées forment un groupe d'entrées ayant les plus petites distances à ce neurone par rapport aux distances à d'autres neurones.

Après le processus de classification, un neurone C_k est un ensemble d'index i des entrées appartenant au neurone et le vecteur de neurone c_k est le centre de la gravité de ces entrées dans l'espace de données.

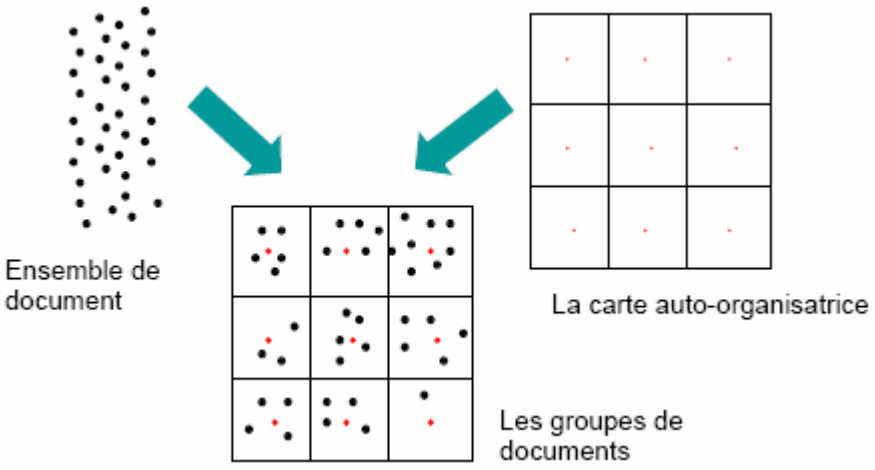


Figure 5-6 : Classification des documents

Partie 2 : La visualisation par les SOM

2.1 Classification basée sur La SOM

On peut trouver plusieurs implémentations de ce modèle sur Internet (le code source ou le package binaire). Dans ce cas, on a choisi d'utiliser le package SOMToolbox qui a été développé par Teuvo Kohonen (l'auteur de la SOM) et ses confrères. Ce package est téléchargeable à partir de la page **'http://www.cis.hut.fi/projects/somtoolbox/'**

Avant d'utiliser l'algorithme SOM, nous allons répartir notre corpus sur deux groupes. Le premier groupe sera utilisé pour l'entraînement de la carte; il comporte 51 documents, Le deuxième groupe est celui du test, et contient le reste des documents (304).

a) Construction de la carte auto-organisatrice

La première étape de ce modèle est l'initialisation de la carte SOM. Dans la SOMToolbox propose la commande «`som_randinit` » pour initialiser aléatoirement cette carte. Pour le faire, il faut fournir quelques arguments :

- La taille de la carte
- Les données d'apprentissage
- Le fichier de sortie
- Le type de la carte (carré ou hexagone)
- Le type de la fonction du voisinage (bubble ou gaussian)

Par exemple :

```
> sD = som_read_data('fichier1.data');
> msize = [6 6];
> insize = size(sD,1);
> lattice = 'rect'; % 'rect' ou 'hexa'
> shape = 'sheet'; % 'sheet', 'cyl', ou 'toroid'
> sMap = som_map_struct(6,'msize',msize, lattice, shape);
%Initialisation des poids de la carte topologique
sMap = som_randinit(sD, sMap);% som randinit ou som lininit
```

En utilisant les commandes ci-dessus, on a construit une carte carrée ayant 36 noeuds. Cette carte sera utilisée comme l'entrée dans l'étape de l'apprentissage.

On peut donner le programme suivant pour représenter la carte dans l'espace d'entrée (initialisation de la carte : poids initiaux des neurones) et l'espace de sortie. En effet, à chaque unité de la carte correspondent deux types de coordonnées (Figure 5-7):
- le neurone dans l'espace d'entrée
- la position sur la carte dans l'espace de sortie

```
> subplot(1, 3,1)
> som_grid(sMap, 'coord',sMap.codebook(:,[1 2]))
> title('Map in input space')
> subplot (1, 3, 2)
> som_grid (sMap)
> axis([0 11 0 11]),view(0,-90),title('Map in output space')
```

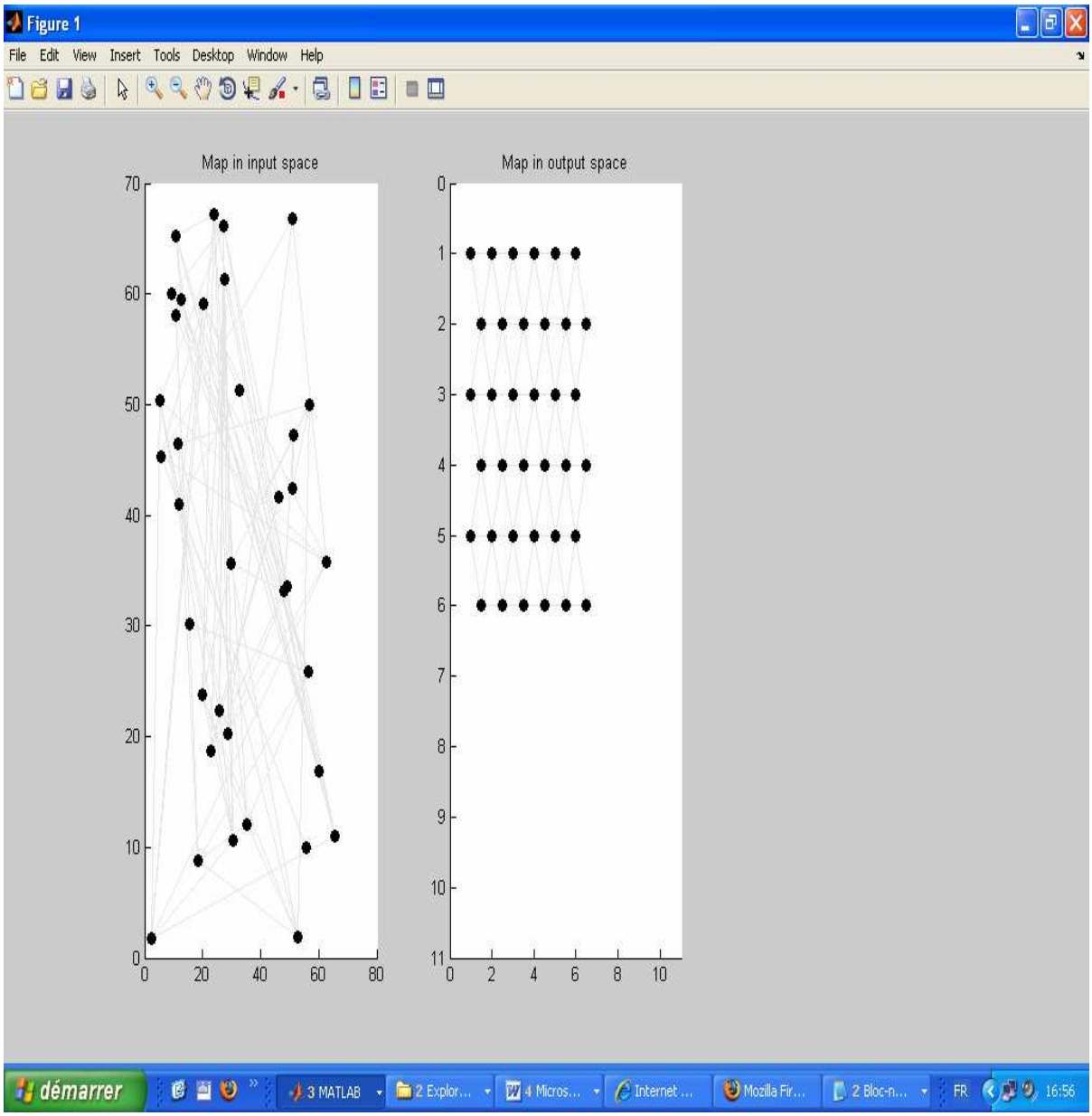


Figure 5-7: Représentation de la carte dans les deux espaces d'entrée et de sortie

b) Processus de l'apprentissage

Dans cette étape, on entraîne la carte à l'aide des données d'apprentissage. Avec la SOMToolbox, on utilise la commande «`som_batchtrain` ». Les arguments principaux :

- Les données d'entrées pour l'apprentissage
- La carte initiée
- Le taux d'apprentissage
- Le nombre de l'étape d'apprentissage
- Etc.

Par exemple :

1- Entraînement de la carte : Phase 1 ('Auto Organisation')

```
>  epochs = 500;
>  radius_ini = 3;
>  radius_fin = 1;
>  Neigh = 'gaussian'; % 'gaussian', 'cutgauss', 'bubble' ou 'ep'
>  tr_lev = 3;
>  [sMap,sT] =
som_batchtrain(sMap,SD,'trainlen',epochs,'radius_ini',radius_ini,'rad
ius_fin',radius_fin,'neigh',Neigh,'tracking',tr_lev);
> xlabel('AUTO ORGANISATION')
```

La commande `som_batchtrain` entraîne la carte avec un taux d'apprentissage (dans cet exemple, on a assigné une valeur 3) qui diminue progressivement de 3 jusqu' à 1. Après 500 boucles, ce processus sera arrêté. À partir de celà, on peut utiliser cette carte pour classifier nos documents. La fonction de voisinage utilisée est la fonction «`gaussian` » (Figures 5.8-5.9).

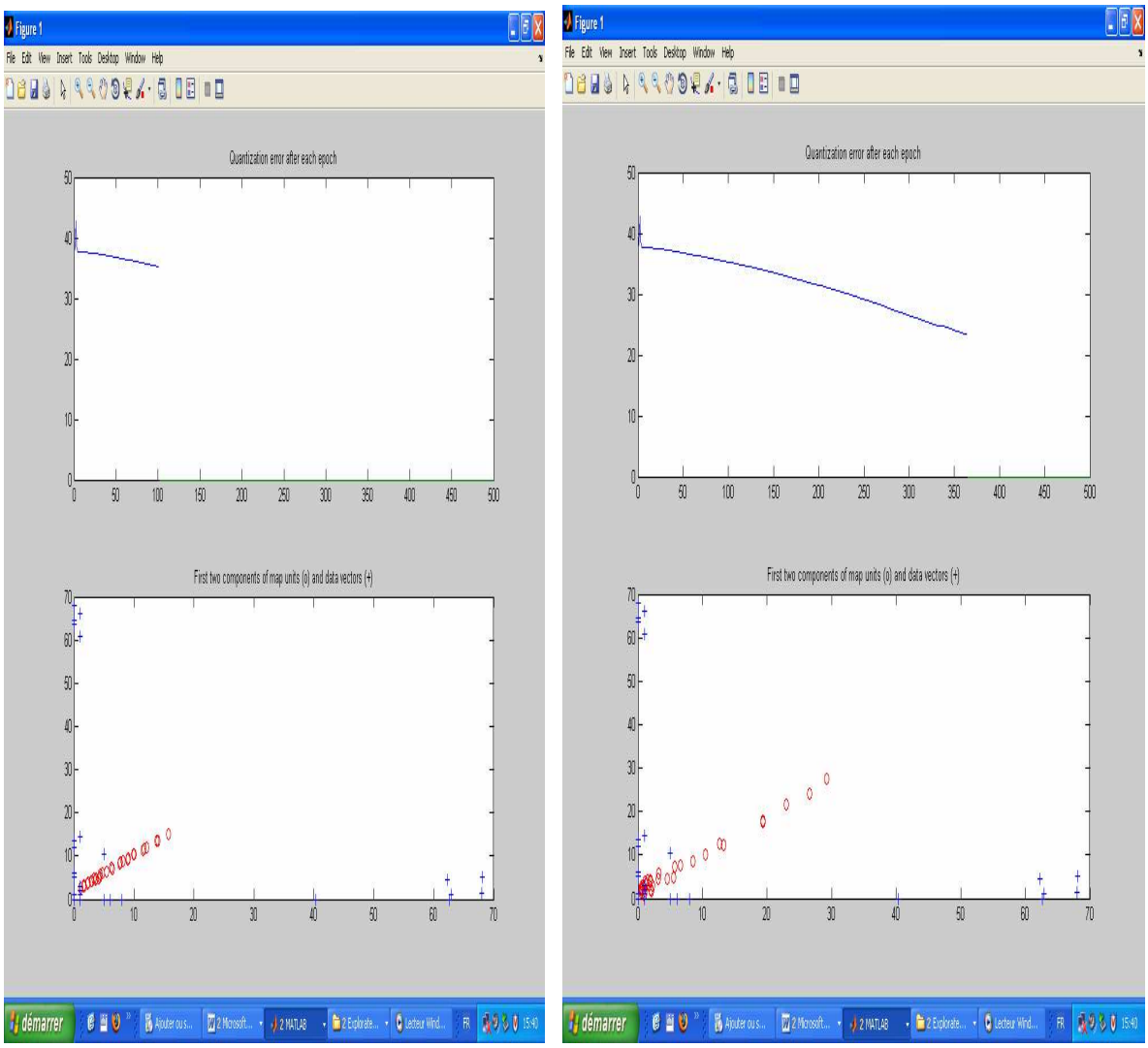


Figure 5-8 : Phase de l'entraînement –auto organisation pour époch=500 avec un courbe de quantification de l'erreur par époch

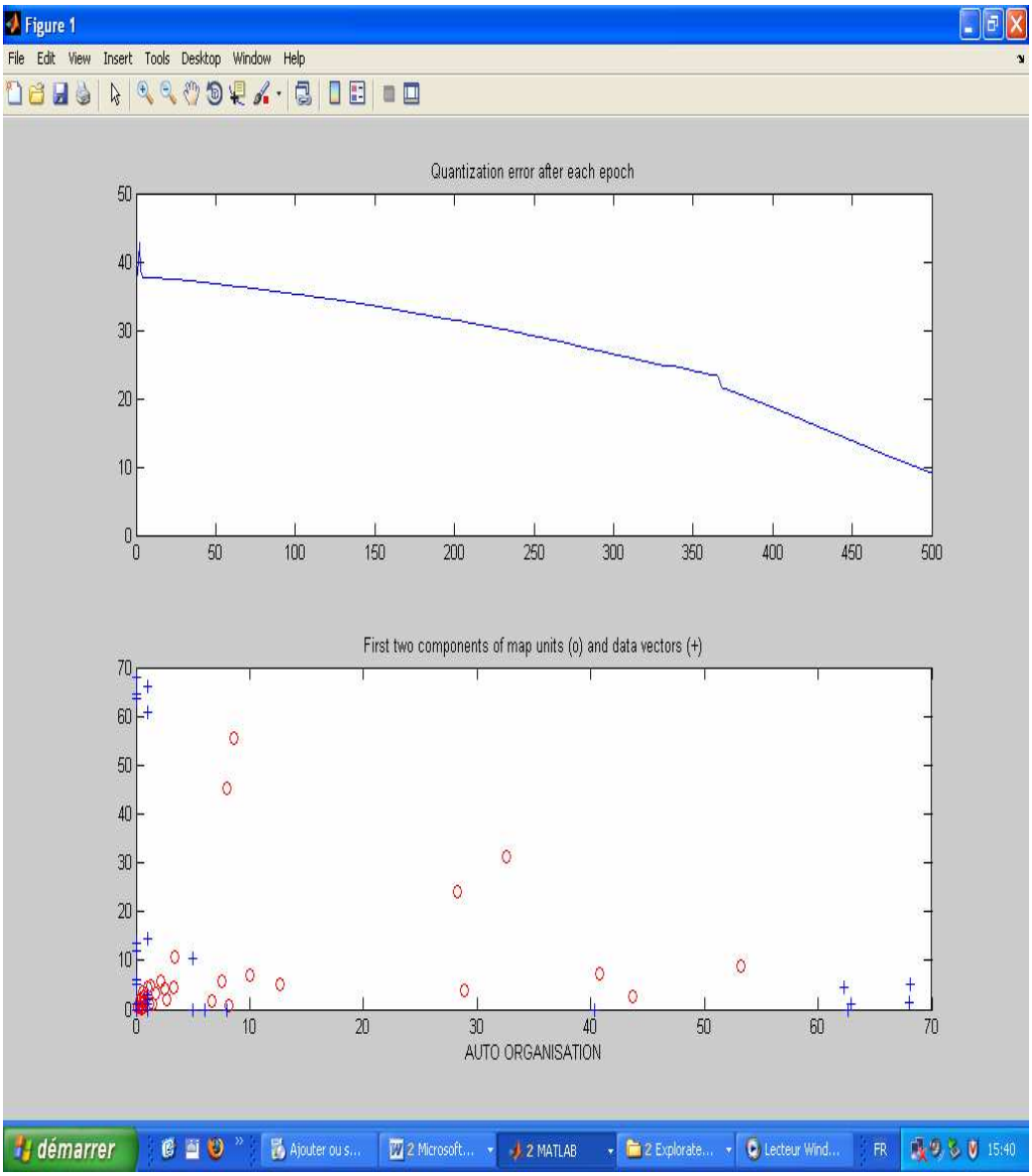


Figure 5-9: Présentation de l'auto organisation pour 500 époch (phase1)

2- Entraînement de la carte : Phase 2 (Convergence)

```
>  epochs = 500;
>  radius_ini = 1;
>  radius_fin = 0.1;
>  [sMap,sT] = som_batchtrain(sMap,
sD,'trainlen',epochs,'radius_ini',radius_ini,'radius_fin',radius_fin
, 'neigh',Neigh,'tracking',tr_lev);
>  xlabel('CONVERGENCE')
```

La phase de création de la carte et de l'apprentissage peut être éffectuée par la commande sMap = som_make(sD); en spécifiant les paramètres comme :epoch, l'algorithme,la taille de la carte…(Figures 5-10,5-11).

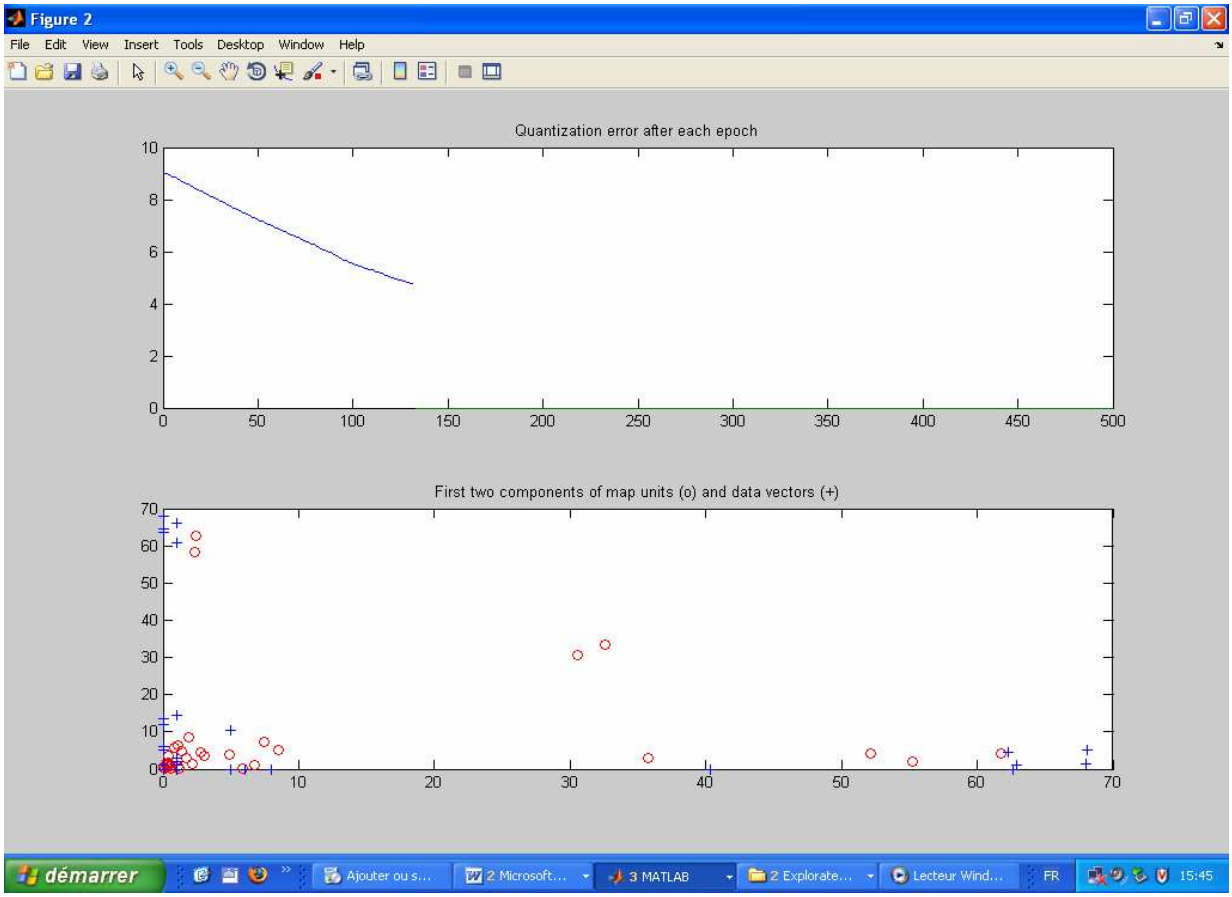


Figure 5-10 : Présentation de la phase de convergence à l'instant epoch=150

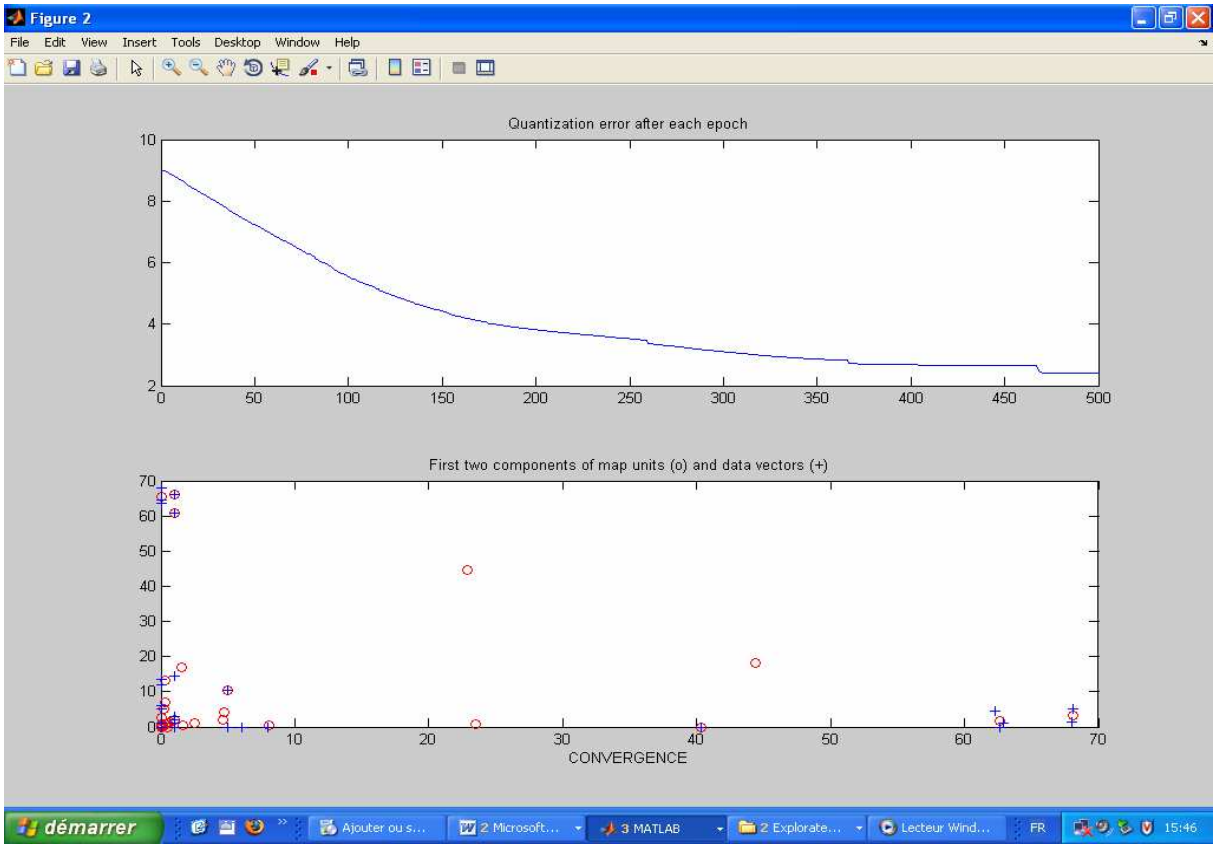


Figure 5-11 : Présentation de la phase de convergence epoch=500

c) Visualisation de la carte

Jusqu'à ce moment, on a une carte complète. Pour la visualiser on utilise la commande suivante

>

```
som_show(sMap,'umat','all','comp',[1:6],'empty','Labels','norm','d');
```

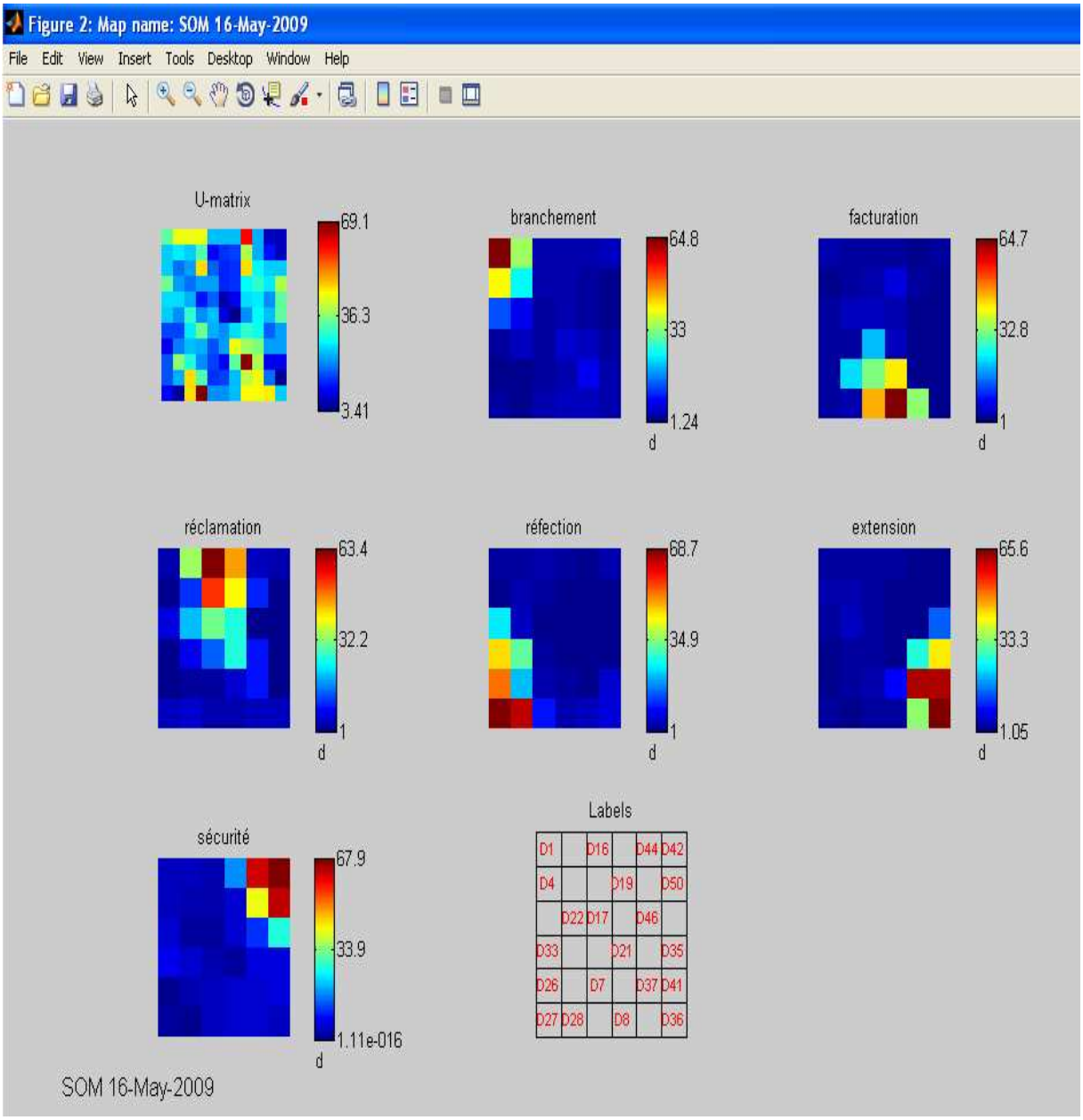


Figure 5-12 : Visualisation de la carte

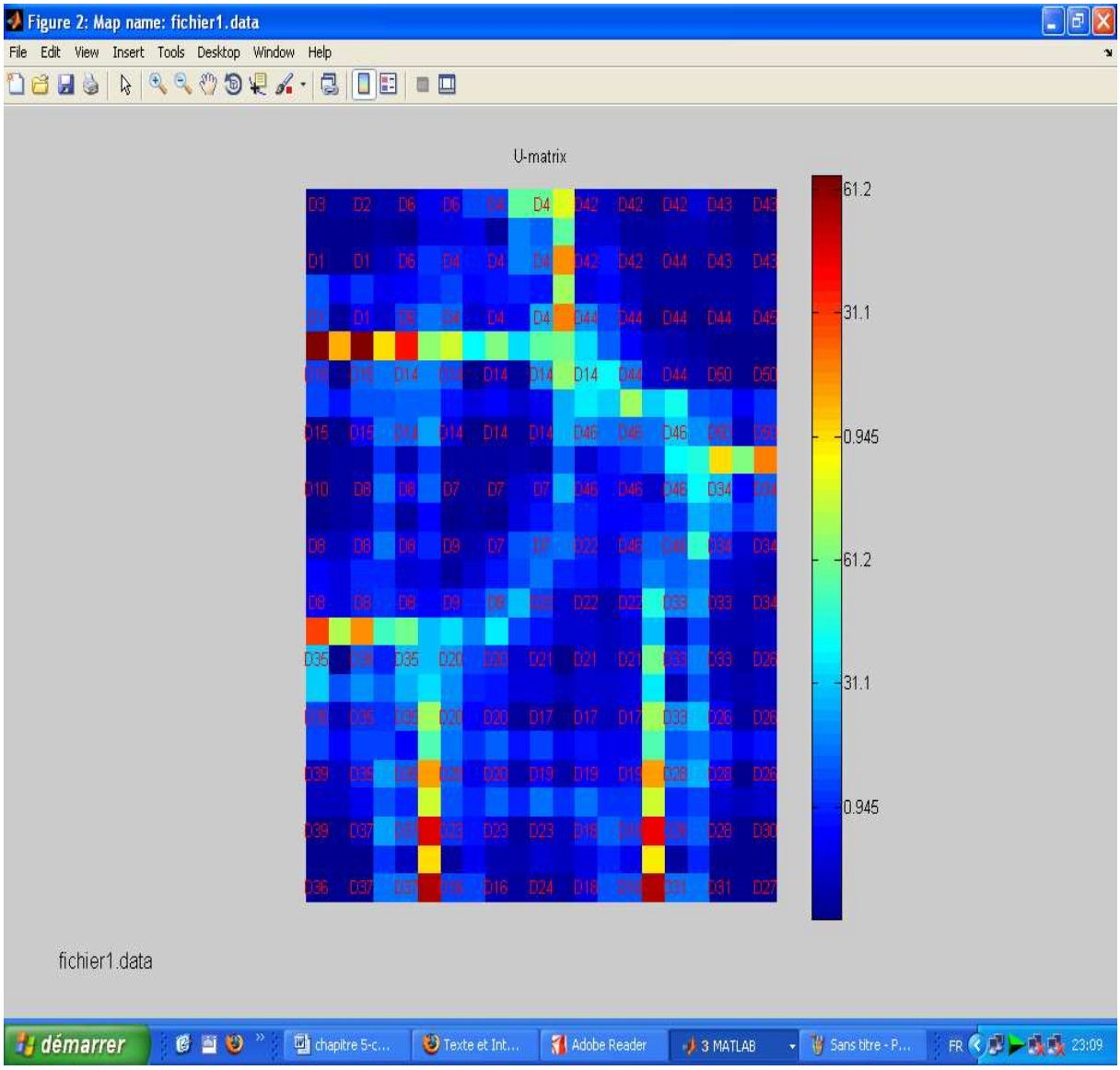
La Figure 5-12 présente la carte des documents classifiés où se trouve la répartition des documents qui correspondent à chaque présentation d'une variable. Par exemple : pour le variable branchement on trouve les documents : D3, D4 sur la matrice labels, qui sont en relation avec la présentation des couleurs affichées sur la carte de la variable « branchement ».

d) Application de la classification sur le fichier de données (groupe de test).

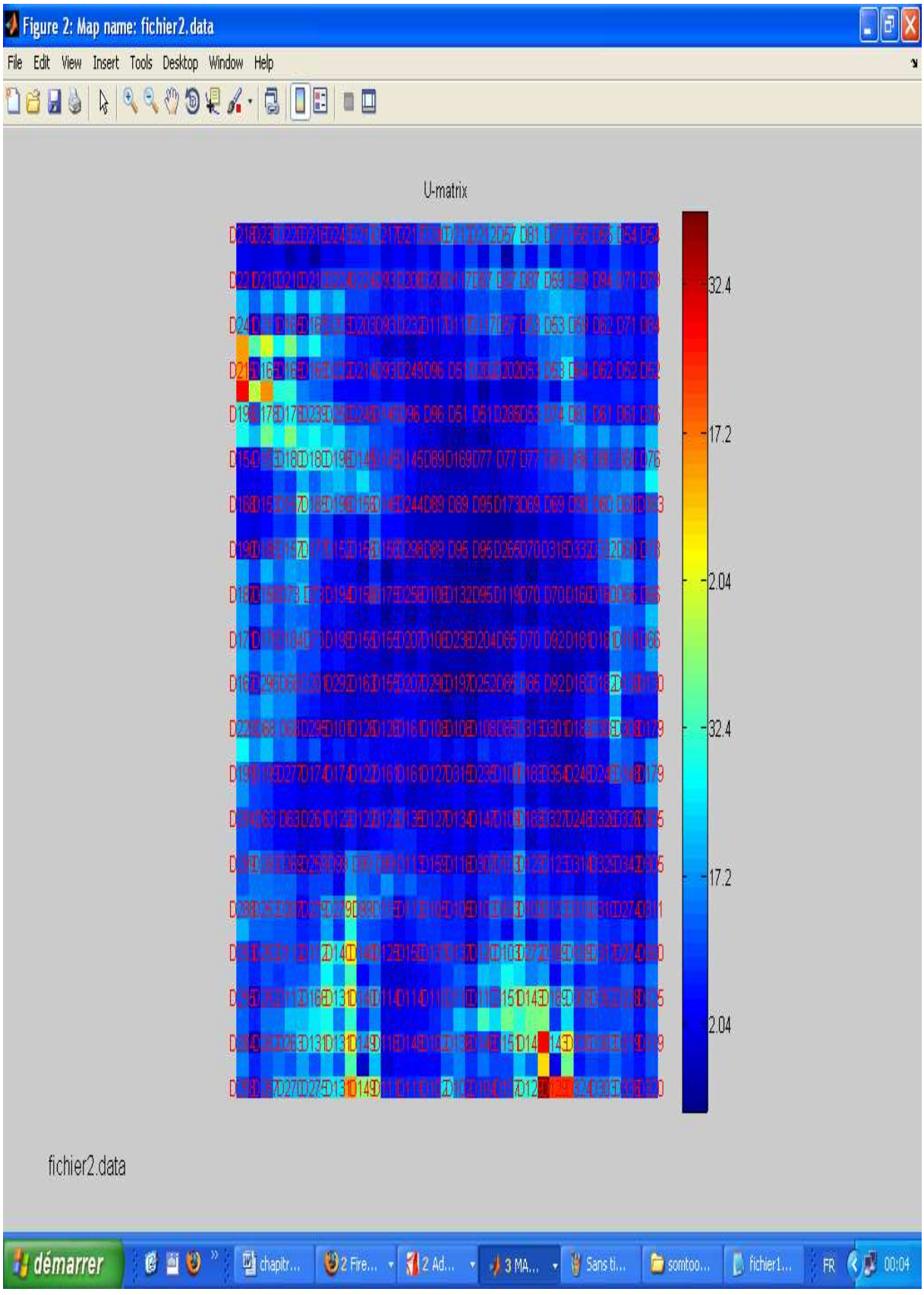
Pour bien visualiser le résultat de la classification, sur les données de test on utilise les commandes suivantes :

```
> sD1 = som_read_data('fichier2.data');
> sM = som_supervised(sD1,'big','lattice','rect','neigh','gaussian');
> som_show(sM,'umat','all');
> som_show_add('label',sM.labels,'TextSize',10,'TextColor','r')
> sD2 = som_label(sD1,'clear','all');
> sD2 = som_autolabel(sD2,sM);        % classification
```

Fichier2.data c'est le groupe de test lu à travers la commande som_read_data puis vient l'application da la carte de classification obtenue précédemment par les commmandes som_supervised , som_show et som_label. (Figure 5-13).



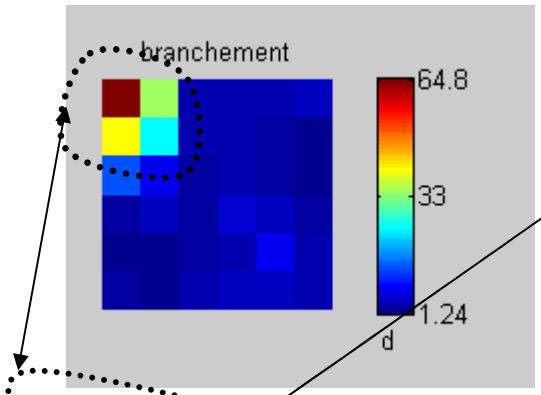
La Figure 5-13 : La classification des documents de l'entraînement (fichier1.data)



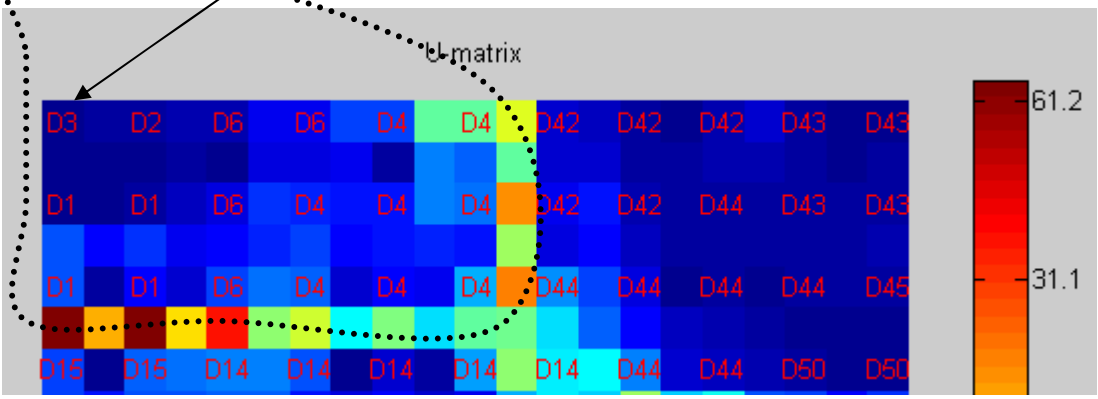
La Figure 5-14: La classification des documents de test (fichier2.data)

2.2 Lecture des résultats :

Le rapprochement entre le variable branchement avec la présentation de la classification des documents de l'entraînement U-matrix est expliqué par leurs valeurs qui sont affichées dans le tableau qui nous montre une convergence vers le variable branchement. On conclut que Les documents qui expriment les mêmes idées en différentes langues sont arrangés dans le même groupe (Figure 5-15).



Les vecteurs des documents						
doc	**Branc.**	**Factu.**	**Récla**	**Réfec.**	**exte**	**sécur**
D1	62,25	4.60	2.10	1.20	2.80	0.90
D2	62.94	1.60	0.90	1.50	4.30	5.10
D3	68.01	1.50	3.20	1.20	2.50	2.51
D4	40.37	1.50	1.50	2.00	3.50	4.00
D5	68.14	5.50	3.30	4.40	1.30	1.50
D6	62.61	4.20	5.40	4.60	0.20	8.80



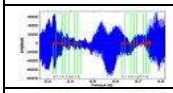
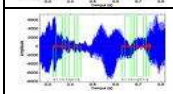
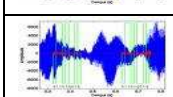
Signal audio	doc	texte	langue
	**D1**	Bonjour je veux un branchement d'électricité pour ma maison mon adresse est ……	français
	**D2**	Hello I want a connection of electricity and gas, and billing of the estimate I have electricity connections since two year I live	Anglais
	**D3**	J'ai un branchement d'électricité et je veux un branchement gaz pour la même adresse ma référence est ……	français

Figure 5-15 : rapprochement entre classification et les documents sources

Pour évaluer les résultats obtenus, dans chaque noeud de la carte, on compte le nombre de documents de chaque variable (**branchement, facturation, réclamation, réfection, extension, sécurité**)

Les tests ont été faits à partir de la fonction de voisinage que le package SOMtoolbox propose (gaussian). Les résultats de ces tests sont reportés dans le tableau 5-1. Pour chaque test, nous avons essayé de faire passer la valeur initiale du taux d'apprentissage de 0.01 à 3.

Nombre de document /variable						
Taux appren	Branchement	facturation	réclamation	réfection	Extension	sécurité
0.01	11	0	9	7	9	14
0.05	11	1	11	10	16	16
1	15	14	20	22	21	25
1.5	17	14	25	25	25	22
2	22	25	23	35	49	38
2.5	37	47	46	34	53	46
3	47	55	51	37	55	59

Tableau 5-1 : Résultat du test

Selon le résultat obtenu, on peut remarquer que les groupes sont divisés de manière de plus en plus claire quand ce taux est augmenté. Comme on a abordé dans ce chapitre, le taux d'apprentissage nous montre la distance entre la position actuelle et la position optimale du vecteur de neurone. Au début, les vecteurs de neurone sont initiés aléatoirement. Le vecteur de neurone est donc plus loin de la position optimale. Alors, au début, le taux d'apprentissage doit être grand. Dans le cas contraire, après le processus d'apprentissage, les vecteurs de neurone ne peuvent pas atteindre la position optimale.

2.3 Etude de la qualité d'apprentissage des cartes topologiques :

La qualité d'apprentissage était évaluée en fonction de deux critères: la résolution de la carte et la préservation de la topologie en utilisant la fonction som_quality() de la manière suivante : [qe_conv,te_conv]=som_quality(sMap,sD) calculée sur les deux ensembles de données d'apprentissage et de test. Dans les tableaux 5-2 et 5-3 ci-dessous, chaque ligne indique le résultat obtenu pour un nombre d'itération définit.

L'erreur de quantification (qe_conv) est la distance moyenne entre chaque vecteur de données et l'unité de meilleur appariement qu'il déclenche. Elle reflète la résolution de la carte topologique.

$$\frac{1}{X}\sqrt{\sum_{x=0}^{X-1}(x_i - w_{bmui})^2} \qquad 2.3.1$$

Ou : x_i est le vecteur d'entrée, $_{bmui}$ est l'indice du best matching unit de x, et w est le poids du best matching unit.

L'erreur topographique (te_conv) est la proportion de vecteurs de données pour laquelle le premier et le second bmus ne sont pas adjacents. Elle rend compte de la préservation de la topologie.

$$\frac{1}{X}\sum_{x=0}^{X-1} d_{node}(bmu1 - bmu2) \neq 1 \qquad 2.3.2$$

	Données d'entraînement		Données de test	
Nombre d'itérations	Erreur de quantification	Erreur topographique	Erreur de quantification	Erreur topographique
100	3,5108	0,0439	4,3351	0,0642
500	3,4102	0,0273	4,0242	0,0509
1000	3,3056	0,0471	3,8653	0,0789
2000	3,2501	0,0446	3,7812	0,0635
3000	3,2152	0,0504	3,6522	0,0752
4000	2,8213	0,0457	3,6111	0,0866
5000	2,7108	0,0439	3,5985	0,0704

Tableau 5-2 : estimation de la qualité d'apprentissage

La qualité d'apprentissage était mesurée à partir de la différence entre les deux meilleurs essais d'entraînement successifs, calculée comme: $(E^{(t-1)}-E^{(t)})/E^{(t)}$, où E est la mesure et *t* l'indice temporel de l'essai. L'évolution de l'erreur était comparée à un seuil S, utilisé comme critère d'arrêt. Ici, une valeur de 0.01 était choisie pour S. La qualité d'entraînement était estimée sur les données de test de manière à sélectionner les essais d'apprentissage ayant la meilleure capacité de généralisation.

	Données d'entraînement		Données de test	
Nombre d'itérations	Erreur de quantification	Erreur topographique	Erreur de quantification	Erreur topographique
100	-	-	-	-
500	0,0287	0,377	0,0717	0,207
1000	0,0307	-0,722	0,0395	-0,549
2000	0,0168	0,052	0,0218	0,195
3000	0,0107	-0,128	0,0341	-0,185
4000	0,1225	0,092	0,0113	-0,151
5000	0,0392	0,040	0,0035	0,187

Tableau 5-3 : Evolution des mesures d'erreur

Le seuil de diminution de l'erreur était atteint pour 4000 itérations d'entraînement. Un essai supplémentaire a été réalisé avec 5000 itérations. La stagnation de l'erreur de quantification confirme la bonne résolution de la carte sans l'améliorer significativement. L'erreur topographique indique le meilleur degré de préservation de la topologie obtenu.

2.4 Evaluation du SRI :

L'évaluation des systèmes de recherche consiste à vérifier si le système est capable de satisfaire les besoins des utilisateurs réels et potentiels non seulement d'une manière individuelle, mais plutôt collective. L'objectif d'une telle évaluation est d'améliorer le processus de recherche d'information aussi bien dans des cas particuliers que dans des cas plus généraux. Ce type d'évaluation s'est basé généralement sur les mesures du taux de rappel et de précision.

Pour une requête, les documents dans un corpus forment deux partitions selon deux caractéristiques :

- les documents récupérés et les documents non récupérés.
- les documents pertinents et les documents non pertinents.

Pour mesurer les performances qualitatives des SRI, on procède à la comparaison des ensembles (documents pertinents, documents non pertinents) et (documents retrouvés, documents non retrouvés) sur l'ensemble des requêtes. Il existe à cet effet de nombreuses mesures, chacune mettant en évidence telle ou telle propriété du système. Les mesures les plus classiques sont les mesures de rappel et de précision.

Mesure de précision : la précision mesure la capacité du système à rejeter tous les documents non pertinents à une requête. Il est donné par le rapport entre l'ensemble des documents sélectionnés pertinents et l'ensemble des documents sélectionnés.

Mesure de rappel : le rappel mesure la capacité du système à retrouver tous les documents pertinents répondant à une requête. Il est donné par le rapport entre les documents retrouvés pertinents et l'ensemble des documents pertinents de la base.

Les taux de précision et de rappel sont donnés par les formulations suivantes :

Précision=R+/M

Rappel= R+/R

Où :

- R désigne le nombre de documents pertinents dans toute la collection.
- M est le nombre de documents sélectionnés par le système.
- R+ est le nombre de documents pertinents sélectionnés par le système.

L'utilisation des collections de documents de l'entraînement avec la carte de classification Figure 5-13 nous a permet de reporté les valeurs de mesure de rappel et précision comme suit :

	Pertinent	Précision	rappel
D1	P	1	0,03
D2	P	1	0,05
D3	P	1	0,08
D4	P	1	0,11
D5		0,80	0,11
D6	P	0,83	0,13
D7	P	0,86	0,16
D8	P	0,88	0,18
D9	P	0,89	0,21
D10	P	0,90	0,24
D11		0,82	0,24
D12		0,75	0,24
D13		0,69	0,24
D14	P	0,71	0,26
D15	P	0,73	0,29
D16	P	0,75	0,32
D17	P	0,76	0,34
D18	P	0,78	0,37
D19	P	0,79	0,39
D20	P	0,80	0,42
D21	P	0,81	0,45
D22	P	0,82	0,47
D23	P	0,83	0,50
D24	P	0,83	0,53
D25		0,80	0,53
D26	P	0,81	0,55
D27	P	0,81	0,58
D28	P	0,82	0,61
D29		0,79	0,61
D30	P	0,80	0,63
D31	P	0,81	0,66
D32		0,78	0,66
D33	P	0,79	0,68
D34	P	0,79	0,71
D35	P	0,80	0,74
D36	P	0,81	0,76
D37	P	0,81	0,79
D38		0,79	0,79
D39	P	0,79	0,82
D40		0,78	0,82
D41		0,76	0,82
D42	P	0,76	0,84
D43	P	0,77	0,87
D44	P	0,77	0,89
D45	P	0,78	0,92
D46	P	0,78	0,95
D47		0,77	0,95
D48		0,75	0,95
D49	P	0,76	0,97
D50	P	0,76	1,00

Tableau 5-4 : Valeur de meure de rappel et précision

On considère d'abord le premier document d1 restitué par le système. A ce point, on a retrouvé un document pertinent parmi les 38 existants. Donc on a un taux de rappel de 0.03. La précision est 1/1. Le point de la courbe est donc (0.03, 1.0). On considère ensuite les deux (2) premiers documents restitués. Le taux est de 2/38 et la précision est cette fois de 2/2 (deux document sur deux est bon). Le point est donc (0.05, 1).

Ce processus est répété jusqu'à épuisement de la liste des réponses (qui peuvent être très longue en incluant tous les documents de la base). Les points de la courbe sont alors représentés comme dans la Figure 5-16.

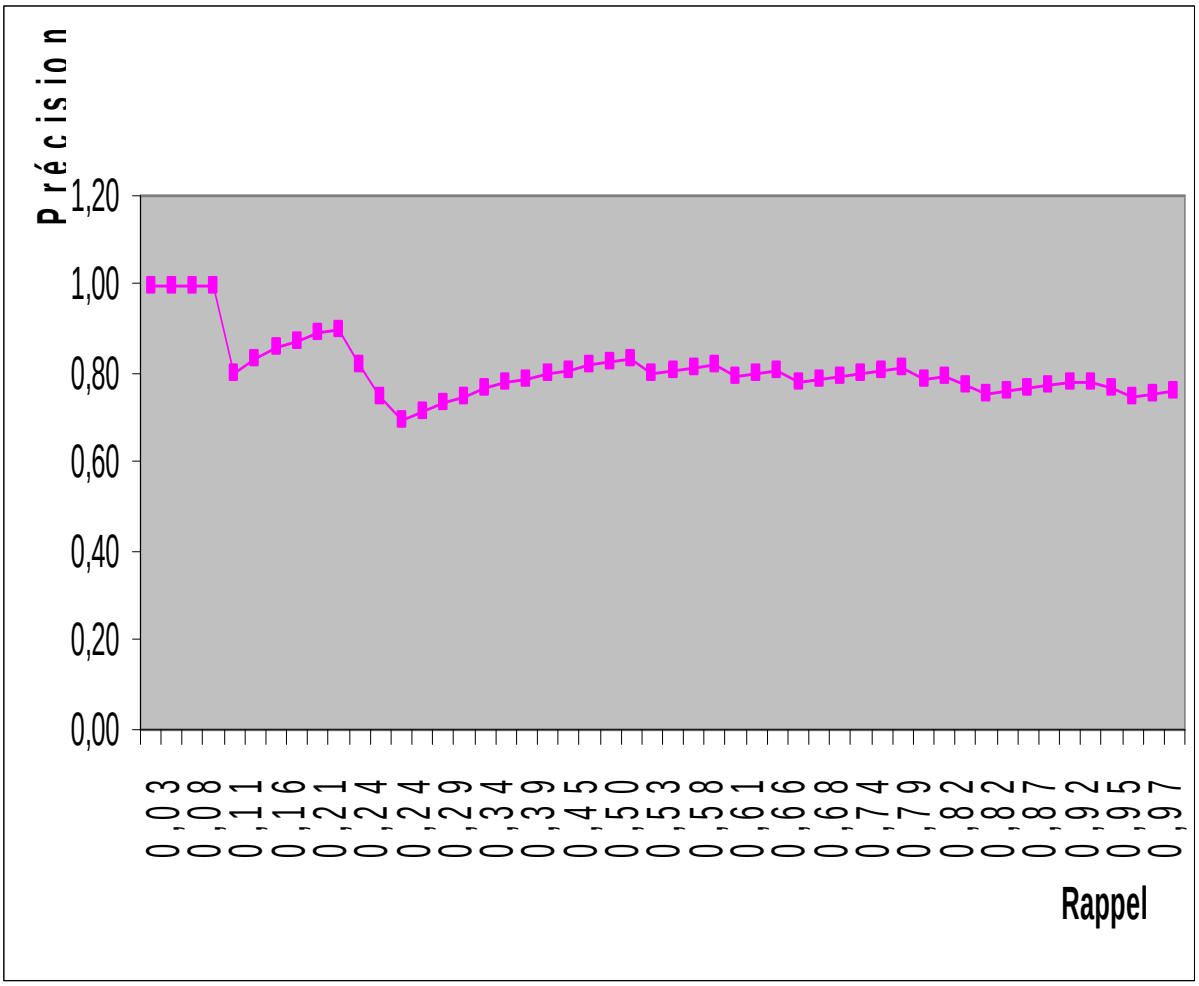


Figure 5-16: Courbe rappel précision

A partir de ces résultats, Il sera possible d'obtenir un système précis (mesure de précision de 0,76), et un rappel fort (mesure rappel de 1, soit la totalité des documents pertinents).

2.5 Conclusion

Après les expériences de l'utilisation de l'ontologie, on peut tirer quelques conclusions :

• Le nombre de concepts dans l'ontologie est fixe. La dimension des vecteurs représentant les documents est donc invariable. L'ontologie dépend du domaine utilisé et les concepts qui apparaissent dans les documents doivent être dégagés. En outre, chaque concept est représenté par plusieurs termes différents. Pour cette raison, les concepts non apparentés considérés comme des valeurs nulle.

• À l'aide de l'ontologie, on peut utiliser la même façon pour traiter les documents dans plusieurs langages. Les termes sont dépendants du langage utilisé mais les concepts ne le sont pas.

• Il est aisé d'expliquer les résultats. Quand deux documents se trouvent dans un même groupe, ils doivent avoir beaucoup de concepts similaires.

• Il est difficile de conserver une ontologie, une fois conçue. Quand les nouveaux concepts apparaissent, il faut les ajouter à cette ontologie. Sinon, le système ne peut les identifier.

Conclusion générale

Notre travail se situe dans le contexte de la recherche d'information multilingue : il faut retrouver tous les documents audio relatifs à un thème donné quelque soient leurs langues.

Notre approche est centrée autour d'une ontologie de domaine ou on fait appel aux cartes auto organisatrices SOM (Self Organizing Map) pour visualiser les résultats.

La première étape de notre système est la transcription des appels téléphoniques, les résultats obtenus sont des fichiers textuels de différentes langues, cette première étape peut être effectuée par un système de reconnaissance de la parole ou encore par un système de détection de mots clés, et pour indexer thématiquement les fichiers transcrits, nous avons procédé à la résolution du problème de multilinguisme. La solution réalisée arrive à présenter l'ensemble des fichiers audio transcrits dans une langue intermédiaire qui est un système de concepts et non un système de signes, c'est dans la deuxième étape qu'on a conçu manuellement une ontologie de domaine « Sonelgaz » où les instances des termes sont de type multi langues qui peuvent être liées aux d'autres concepts ou d'autres termes et d'autres variantes du même terme.

L'ontologie Sonelgaz exprime les conceptualisations spécifiques au domaine de l'électricité et du gaz, elle rend compte du vocabulaire spécifique à travers de concepts, termes et de relations qui modélisent les principales activités, les théories et les principes de base du domaine en utilisant les relations de Terme vers sa variante, Concept vers terme, Concept vers concept et Terme vers terme.

Pour modéliser la présentation conceptuelle des documents transcrits, nous projetons ces derniers sur notre ontologie de domaine « Sonelgaz » , c'est dans la troisième étape ou on fait l'extraction des termes du document et la Recherche des termes correspondants à des concepts ou instances de l'ontologie. Comme chaque terme extrait peut avoir plusieurs sens, donc correspondre à plusieurs concepts dans l'ontologie, il convient de procéder au calcul des mesures de similarité entre les différents sens des termes, en vue de sélectionner, pour chaque terme, le meilleur sens correspondant dans l'ontologie.

Le mécanisme de désambiguïsation du terme est mis en place afin d'identifier quel est le concept abordé dans le document, Le modèle *DocCore* est l'un des méthodes de mesure de similarité entre deux noeuds représente une valeur condensée résultant de la comparaison de deux sens possibles pour deux termes (donc deux concepts candidats).

L'algorithme de projection réalisé permet d'obtenir des vecteurs de concepts représentant les documents transcrits ou les éléments du vecteur sont des valeurs qui indiquent les poids de chaque concept dans l'ontologie.

Pour classifier les vecteurs de concepts obtenus on a utilisé l'algorithme de kohonen – SOM- ou les résultats de classification sont obtenus d'une manière non supervisée et la visualisation de la carte est possible ce qui permet l'interprétation et l'évaluation de notre système de recherche d'information.

La classification obtenue par l'algorithme SOM permet la visualisation des vecteurs de documents arrangés en groupe de variable, ils doivent être très semblables et donc partager beaucoup de concepts qu'il est possible de lister.

Comme suite à ce travail, nous pouvons envisager les perspectives suivantes :

- Développer la partie de la transcription par un système de détection de mots clés pour prendre en charge les mots dans une grande variante de langues.

- Enrichir l'ontologie avec plusieurs langue pour s'adopte au résultat de la transcription.

- Implémenter la partie de la mise à jour de notre ontologie dans le cas ou un nouveau terme où concept est apparaît.

- Implémenter le mécanisme de désambiguïsation du terme avec les variantes de mesures soit sur le chemin entre deux concepts, soit sur la notion de contenu d'information, ou sur une combinaison du chemin et du contenu d'information, pour sélectionner le meilleur mécanisme qui donne plus de précision dans le sens de terme correspondant dans l'ontologie.

Bibliographie

[Abberley et al., 99] D. Abberley**,** S. Renals, Dan Ellis, and T. Robinson. The THISL SDR system at trec-8. In Proceedings of the 8th Text Retrieval Conference TREC-8, Gaithersburg, MD, November 1999.

[Abdillahi et al., 06] N. Abdillahi, P. Nocéra, et J.-F. Bonastre, 2006. Towards Automatic Transcription of Somali Language. Dans les actes de Language Resource and Evaluation Conference (LREC).

[ACA, 1798] Dictionnaire de L'Académie française, 5th Edition (1798), Page 767 http://colet.uchicago.edu/cgi-bin/getobject_?p.16:32./projects/artflb/ databases/ artfl/dicos/acad1798/IMAGE/

[Afnor, 96] Afnor, Information et documentation, principes généraux pour l'indexation des documents, volume Documentation, présentation des publications, traitements documentaire et gestion de bibliothèques. Association Française de Normalisation, 1996.

[Ahmad et al., 01] K.Ahmad, B. L.Vrusias, and A.Ledford, (2001). Choosing feature sets for training and testing self organising maps: A case study. Neural Computating & Applications, 10(1):56--66.

[Allan, 02] J. Allan, 2002. Topic Detection and Tracking : Event-Based Information Organization. Kluwer.

[Al-Zabibi, 90] M. Al-Zabibi, "An Acoustic–Phonetic Approach in Automatic Arabic Speech Recognition," The British Library in Association with UMI, 1990.

[Bachimont, 00] B. Bachimont, (2000). Engagement sémantique et engagement ontologique: conception et réalisation d'ontologies en ingénierie des connaissances. In Charlet, J., Zacklad, M., Kassel, G., and Bourigault, D., editors, Ingénierie des connaissances: évolutions récentes et nouveaux défis, pages 305-323. Eyrolles.

[Bachimont, 01] B. Bachimont, (2001). Modélisation linguistique et modélisation logique des ontologies: l'apport de l'ontologie formelle. In Actes des journées francophones d'Ingénierie des Connaissances (IC'2001), pages 349-368. Presse Universitaire Grenobloise.

[Baeza Yates et Ribeiro Neto, 00] R. Baeza-Yates and B. Ribeiro-Neto, Modern Information Retrieval, ACM Press,Addison Wesley, 2000.

[Baeza-Yates et al., 99] Ricardo A. Baeza-Yates, Berthier A. Ribeiro-Neto: Modern Information Retrieval ACM Press / Addison-Wesley 1999.

[Baloul ,03] S. Baloul, "Développement d'un système automatique de synthèse de la parole à partir du texte arabe standard voyellé," Thèse de Doctorat, Université de Maine, Le Mans, 2003.

[Bar-Hillel, 58] Y. Bar-Hillel, 1958. The mechanization of literature searching. Mechanization of Thought Processes 10, 4–8.

[Barras, 96] C. Barras, "Reconnassance de la parole continue : adaptation au locuteur et contrôle temporel dans les modèles de markov cachés," Ph.D. Thesis, Paris VI University, 1996.

[Baum et al., 70] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. Annals of Mathematical Statistics, 41 : 164_171, 1970.

[Baziz et al., 05] M. Baziz, M. Boughanem, N.Aussenac-Gilles, C.Chrisment, Semantic Cores for Representing Documents in IR, Proceedings of the 20th ACM Symposium on Applied Computing, pp 1020-1026, ACM Press ISBN: 1-58113-964-0, 2005.

[Bazzi et Glass, 00] I. Bazzi et J. Glass, 2000. Modeling Out-Of-Vocabulary Words for Robust Speech Recognition. Dans les actes de International Conference on Spoken Language Processing (ICSLP).

[Becker et Kuropka, 03] J. Becker et D. Kuropka, 2003. Topic-based vector space model. Dans les actes de Business Information Systems (BIS), 7–12. Church et Gale, 1995) K. Church etW. Gale, 1995. Poisson mixtures. Natural Language Engineering 1(2), 163–190.

[Beigbeder et al., 05] Amélie Imafouo, Michel Beigbeder : Scalability Influence on Retrieval Models: An Experimental Methodology. ECIR 2005: 388-402

[Belew, 00] Richard K. Belew. Finding Out About: A Cognitive Perspective on Search Engine Technology and the WWW. New York: Cambridge University Press, 2000. Review published in Information Retrieval, Vol. 5, Issue 2-3, April-July 2002

[Belkin et al., 92] Nicholas J. Belkin, Peter Ingwersen, Annelise Mark Pejtersen: Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Copenhagen, Denmark, June 21-24, 1992 ACM 1992

[Blair et al., 85] D.C.Blair, M.E. Maron, An evaluation of retrieval effectiveness for a full-text document-retrieval system. Commun. of the ACM, 28, 1985, pp. 289-299.

[Blei et al., 03] D. Blei, A. Ng, et M. Jordan, 2003. Latent dirichlet allocation. Journal of Machine Learning Research 3, 993–1022.

[Borko, 78] H. Borko et C.L. Bernier.Indexing concepts and methods,New York, Academic Press, 1978.

[Bouroche et Saporta, 80] Bouroche J.M. & Saporta G. (1980) L'Analyse des Données, Que sais-je, PUF.

[Brown et al., 01] E. W. Brown, S. Srinivasan, A. Coden, D. Ponceleon, J. W. Cooper, et A. Amir, 2001. Toward Speech as a Knowledge Resource. IBM Systems Journal 40, 985–1001.

[Byrne et al., 04] W. Byrne, D. Doermann, M. Franz, S. Gutsman, J. Hajic, M. Picheny,J. Psutka, B. Ramabhadran, D. Soergel, T. Ward, et W. Zhu, 2004. Automatic recognition of spontaneous speech for access to multilingual oral history archives. IEEE Transactions on Speech and Audio Processing 12(4), 420–435.

[Callan et al., 92] J. P. Callan, W. B. Croft, et S. M. Harding, 1992. The INQUERY Retrieval System. Dans les actes de Database and Expert Systems Applications (DEXA), 78–83. (Garofolo et al., 1999) J. Garofolo, C. Auzanne, et E. Voorhees, 1999. The TREC spoken document retrieval track : A success story. Dans les actes de Text REtrieval Conference (TREC), Volume 8, 16–19.

[Carnegie Mellon University] Carnegie Mellon University. Sphinx-4. Available: http://cmusphinx.sourceforge.net.

[Chandrasekaran et al., 99] B.Chandrasekaran, J.Josephson, and V.Benjamins, (1999). What are ontologies, and why do we need them? IEEE Intelligent Systems, 14(1) :20-26.

[Charlet et al., 04] Ontologies pour le Web sémantique.In *Revue i3*, numéro Hors Série « Web sémantique ».

[Chollet et al., 94] G. Chollet (1994) , "Evaluation of ASR systems, algorithms and databases.", In Antonio Bubio-Ayuso, editor, New Advances and Trends in Speech Recognition and Coding. NATO-ASI, Bubion, 1994

[Chollet et al., 95] G. Chollet, J.-L. Cochard, Ph. Langlais, and R. van Kommer , "Swiss-French Polyphone: a telephone speech database to develop interactive voice servers.",In Linguistic Databases, Gröningen, 1995

[Cleverdon, 67] C.W.Cleverdon, The Cranfield tests on index language devices. Aslib Proceedings 19(6), 173-193, 1967.

[Coden et al., 02] A. R. Coden, E. Brown, et S. Srinivasan, 2002. Acm sigir 2001 workshop "information retrieval techniques for speech applications". Dans les actes de ACM Special Interest Group on Information Retrieval (SIGIR), New York, NY, USA, 10–13. ACM Press.

[Corcho et al., 01] Corcho, O., Lôpez, F. M., and Pérez, A. G. (2001). Ontoweb: Technical roadmap. www.ontoweb.org.

[Crestani et van Rijsbergen, 95] F. Crestani et C. van Rijsbergen, 1995. Information Retrieval by Logical Imaging. Journal of Documentation 51(1), 3–17.

[Croft et al., 91] W. Bruce Croft, Howard R. Turtle, David D. Lewis: The Use of Phrases and Structured Queries in Information Retrieval. SIGIR 1991: 32-45

[Crouch et al., 89] D. B.Crouch, C. J.Crouch, and G.Andreas, 1989. The use of cluster hierarchies in hypertext information retrieval. In Proceedings of the Second Annual ACM Conference on Hypertext (Pittsburgh, Pennsylvania, United States). HYPERTEXT '89. ACM Press, New York, NY, 225-237.

[Crouch et al., 92] Carolyn J. Crouch, Bokyung Yang: Experiments in Automatic Statistical Thesaurus Construction. SIGIR 1992: 77-88

[Davis et Mermelstein, 80] S. Davis and P Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. IEEE Trans. Acoust. Speech and Audio Processing, 28(4) :357_366, 1980.

[De Jong et al., 99] F. de Jong, J.L. Gauvain, J. den Hartog, and K. Netter. Olive : Speech based video retrieval. In Proc. CBMI'99, Toulouse, Oct 1999.

[Deerwester et al., 90] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, et R. Harshman, 1990. Indexing by Latent Semantic Analysis. Journal of the Society for Information Science 41(6), 391–407.

[Deller et al., 93] J. Deller, J. Proakis, J.H. Hansen, "Discrete-Time Processing of Speech Signal," Macmillan, NY, 1993.

[Dempster et al., 77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society Series B,39(1) :1_38, november 1977.

[Deshmukh et al., 99] N. Deshmukh, A. Ganapathiraju, J. Hamaker, J. Picone, and M. Ordowski, "A public domain speech-to-text system," in Proc. 6th European Conf. Speech Communication and Technology, vol. 5, Budapest, Hungary, Sept. 1999, pp. 2127–2130.

[Didier Nakache, 07] "Extraction automatique des diagnostics à partir des comptes rendus médicaux textuels ", thèse pour obtenir le grade de docteur en informatique présentée et soutenue publiquement le 26 septembre 2007. Conservatoire National des Arts et Métiers , préparée au sein du laboratoire CEDRIC – équipe ISID, sous la direction de d'Elisabeth Métais . Année 2007.
[Dieng et al., 01] R.Dieng, O.Corby, F.Gandon, A.Giboin, J.Golebiowska, N.Matta, and M.Ribière, (2001). *Méthodes et outils pour la gestion des connaissances: une approche pluridisciplinaire du knowledge management.* Dunod Ed., *2nde* edition.
[Dittenbach et al., 01] M. Dittenbach, and A. Rauber, and D. Merkl: Recent Advances with the Growing Hierarchical Self-Organizing Map. Allinson, N. and Yin, H. and Allinson, L.and Slack, J. (eds.) Advances in Self-Organizing Maps: Proceedings of the 3rd Workshop on Self-Organizing Maps June 13-15 2001, Lincoln, England, Springer, 2001.

[Douglas, 96] Douglas W. Oard and Bonnie J. Dorr University of Maryland, College Park. A Survey of Multilingual Text Retrieval,1996.

[El-Imam, 89] Y.A. El-Imam, "An unrestricted vocabulary Arabic speech synthesis system", IEEE Transactions on Acoustic, Speech, and Signal Processing vol. 37 , no. 12,pp.1829–1845, 1989.

[Elshafei et al., 91] M. Elshafei, "Toward an arabic text-to-speech system," The Arabian J. Science and Engineering vol. 4B no. 16, pp. 565–583, 1991.

[Fabrizio, 02] Fabrizio Sebastiani, Consiglio Nazionale delle Ricerche, Italy, Machine Learning in Automated Text Categorization ,2002

[Farquhar et al., 95] Farquhar, A., Fikes, R., Pratt, W., and Rice, J. (1995). Collaborative ontology construction for information integration. Technical report, Knowledge Systems Laboratory (Stanford).

[Farquhar et al., 97] A..Farquhar, R.Fikes, and J.Rice, (1997). The ontolingua server: a tool for collaborative ontology construction. International journal of Human Computer studies, 46(6) :707-727.

[Favre, 03] B. Favre, 2003**.** Indexation Multimédia: Caractérisation du Déséquilibre entre les Modalités Texte et Parole. Mémoire de Master, Université d'Avignon.
[Féraud, 1787] J-F. Féraud: « Dictionnaire critique de la langue française » (Marseille, Mossy 1787-1788). CLASSIFICATION(PageA458a).http://colet.uchicago.edu/cgi-bin/getobject_?p.0:918./projects/artflb/databases/artfl/dicos/feraud/IMAGE/

[Frakes et al., 92] William B. Frakes and Ricardo Baeza-Yates (eds.), 1992. Informa-tion Retrieval Data Structures & Algorithms. Prentice-Hall. ISBN

[Frieze et al., 98] A.Frieze, R.Kannan, and, S. Vempala, Fat Monte-Carlo Algorithms for Finding Low-Rank Approximations, Proceedings of 39th Symposium on Foundations of Computer Science, 370-378, 1998.

[Fritzke, 91] B. Fritzke, Let it grow - self-organizing feature maps with problem dependent cell structure, In Kohonen T et al. (eds.), Artificial Neural Networks, Vol1, North-Holland, 1991.
[Fürst, 02] F.Fürst, (2002). L'ingénierie ontologique. Rapport de recherche *n002-07.*

[Gandon, 02] F.Gandon, (2002). Ontology engineering: a survey and a return on experience. Technical report, INRIA. Rapport de recherche 4396.
[Garofolo et al., 98] John S. Garofolo, Ellen M. Voorhees, Vincent M. Stanford, Cedric G. P.Auzanne, and Bruce Z. Lund. 1998 spoken TREC-7 document retrieval track overview and results. In Text REtrieval Conference, pages 79_90, 1998.

[Gauvain et al., 00] J.L. Gauvain, L. Lamel, C. Barras, G. Adda, and Y. Kercadio. The LIMSI SDR system for TREC-9. In Proc. of the Text Retrieval Conference, TREC-9, pages 335_341, Gaithersburg, Nov 2000.
[Gauvain et Lamel, 02] J.L. Gauvain and L.F. Lamel. Systèmes de reconnaissance, de compréhension et de dialogue. In J. Mariani, editor, Reconnaissance de la parole Traitement automatique du langage parlé, volume 2, pages 47_83. Hermes Lavoisier, 2002.
[Gauvain et Lamel, 03] Jean-Luc Gauvain and Lori Lamel. Large vocabulary speech recognition based on statistical methods. In W. Chou and F. Juang, editors, Pattern Recognition in Speech and Language Processing, chapter 5, pages 149_189. CRC Press, 2003.

[Gauvain et Lee, 94] J. Gauvain and C. Lee. Maximum a-posteriori estimation for multivariate gaussian mixture observations of markov chains. IEEE Transactions on Speech and Audio Processing, 2, 1994.
[Godfrey et al., 92] J. Godfrey, E. Holliman, et J. McDaniel, 1992. SWITCHBOARD : Telephone speech corpus for research and development. Dans les actes de International Conference on Acoustics, Speech and Signal Processing (ICASSP), 517–520.
[Grossman et al., 98] David Grossman and Ophir Frieder, Ad Hoc Information Retrieval: Algorithms and Heuristics, Kluwer Academic Publishers, 1998. [Lawrence & Lee Giles 99] S. Lawrence and C. Lee Giles,

Accessibility of Information on the Web, Nature, 400, pp. 107-109, 1999.
[Gruber et al., 95] T. R.Gruber, N. Guarino, and R. Poli, (1995). Formal ontology in conceptual analysis and knowledge representation. Special issue of the International Journal of Human and Computer Studies, 43(5-6) :625-640. citeseer .ist. psu.edu/ guarino95formal.html.

[Gruber, 93] T. R. Gruber. A translation approach to portable ontologies. Knowledge Acquisition, 5(2):199-220, 1993.

[Gruninger et Lee, 02] M. Gruninger, and J. Lee, (2002). Ontology applications and design. ACM Communication, 45(2) :39-41.

[Guarino et Giaretta, 95] N. Guarino, and P.Giaretta, (1995). Ontologies and knowledge bases : towards a terminological clarification. In Towards very large knowledge bases, pages 25-32.

[Guha, 03] R. V.Guha, R. McCool, E. Miller, Semantic search, Proceedings of the 12th International World Wide Web Conference, pp 700-709, 2003.

[H. Satori et al., 07] H. Satori, M. Harti, N. Chenfour: Introduction to Arabic Speech Recognition Using CMUSphinx System CoRR abs/0704.2083: (2007)
[Haiying et al., 05] Haiying Wang, Francisco Azuaje, Olivier Bodenreider. An Ontology-Driven Clustering Method for Supporting Gene Expression Analysis, cbms, pp. 389-394, 18th IEEE Symposium on Computer-Based Medical Systems (CBMS'05), 2005.
[Hansen et al., 04] J. H. Hansen**,** R. Huang, P. Mangalath, B. Zhou, M. Seadle, et J. R. Deller Jr, 2004. SPEECHFIND : Spoken Document Retrieval for a National Gallery of the SpokenWord. Dans les actes de Nordic Signal Processing Symposium (NORSIG).

[Harman, 92] D. K. Harman**,** (ed.), NIST Special Publication 500-207: The First Text REtrieval Conference (TREC-1), 1992.

[Hermansky, 90] H. Hermansky. Perceptual linear predictive (plp) analysis of speech. Journal of the Acoustical Society of America, 87(4) :1738_1752, 1990.
[Hofmann, 00] T. Hofmann, 2000. Learning the similarity of documents : An information-geometric approach to document retrieval and categorization. Advances in Neural Information Processing Systems 12, 914–920.

[Holsapple et Joshi, 02] C. W. Holsapple, and K. D. Joshi, (2002). A collaborative approach to ontology design. Commununications ACM, 45(2) :42-47.

[Hong, 02] G. Z. Hong "Speech Recognition Techniques for Digital Video Library," University of Hong Kong , 2002.

[Honkela et al., 96] T. Honkela, S. Kaski, K.Lagus, T.Kohonen (1996), Exploration of full-text database with self-organizing maps. In IEEE International Conference on Neural Networks-ICNN'96, p. 56-61.

[Hotho et al., 01] A.Hotho, A. Maedche, & S.Staab, (2001). Ontology-based text clustering. In Proceedings of the IJCAI-2001 Workshop "Text Learning: Beyond Supervision", August, Seattle, USA.

[Huang et al., 93] X. Huang, F. Alleva, H. W. Hon, M. Y. Hwang, and R. Rosenfeld, "The SPHINX-II speech recognition system: an overview," Computer Speech and Language, vol. 7, no. 2, pp. 137–148, 1993.

[Huang et al., 90] X.D. Huang, Y. Ariki, M.A. Jack, "Hidden Markov models for speech recognition," Edinburgh: Edinburgh University Press, C, 1990.

[Huang et Robertson, 01] X. Huang and S.E. Robertson, "Comparisons of Probabilistic Compound Unit Weighting Methods", Proc. of the ICDM'01 Workshop on Text Mining, San Jose, USA, Nov. 2001.
[Huang, 89] X.D. Huang, "The SPHINX-II Speech Recognition System: An Overview," Computer Speech and Language, Vol. 2, 1993; K. F. Lee, "Automatic Speech Recognition the Development of the SPHINX System," Kluwer Academic Publishers, 1989.

[Jacquemin et al., 02] C. Jacquemin, B.Daille, J. Royanté, and X. Polanco, 2002. In vitro evaluation of a program for machine-aided indexing. Inf. Process. Manage. 38, 6 (Nov. 2002), 765-792.
[Jalam, 03] Radwan Jalam : "Apprentissage automatique et catégorisation de textes multilingues", thèse pour obtenir le grade de docteur en informatique présentée et soutenue publiquement par le 4 juin 2003. université lumière lyon2, préparée au sein du laboratoire ERIC, équipe de recherche en ingénierie des connaissances, sous la direction de jean-hugues Chauchat. Année 2003.
[Jiang et al., 97] J.Jiang & D.Conrath (1997) Semantic similarity based on corpus statistics and lexical taxonomy. In Proceedings on International Conference on Research in Computational Linguistics, Taiwan, 1997.

[Johnson et al., 00] S. E. Johnson**,** P. Jourlin, K. Spärck-Jones, et P. C. Wooland, 2000. Audio Indexing and Retrieval of Complete Broadcast News Shows. Dans les actes de Recherche d'Information Assistée par Ordinateur (RIAO).

[Jolliffe, 86] I.T. Jolliffe (1986) Principal Component Analysis, Springer Verlag.
[Kanerva et al., 00] P. Kanerva, J. Kristofersson, et A. Holst, 2000. Random Indexing of Text Samples for Latent Semantic Analysis. Dans les actes de Annual Conference of the Cognitive Science Society (CogSci), Volume 1036.

[Kaski et al., 02] S.Kaski, Oja, M., Kohonen, T, Bibliography of Self-Organizing Map (SOM) Papers: 1998-2001 Addendum (2002).

[kaski, 99] S.Kaski, Dimensionality Reduction by Random Mapping In Proc. of the International Joint Conference on Neural Networks, pages 413–418, Anchorage, Alaska, May 1999.
[Kassel et al., 00] Kassel, G., Abel, M., Barry, C., Boulitreau, P., Irastorza, C., and Perpette, S. (2000). Construction et exploitation d'une ontologie pour la gestion des connaissances d'une équipe de recherche. In Actes des journées francophones d'Ingénierie des Connaissances (IC'2000), pages 251-259.

[Kiryakov, 04] Kiryakov A., Popov B., Terziev I., Manov D., Ognyanoff D., Semantic annotation, indexing, and retrieval, Journal of Web Semantics, 2(1), pp 49-79, 2004.
[Kohonen et al., 00] Teuvo Kohonen, Samuel Kaski, Krista Lagus, Jarkko Salojarvi, Jukka Honkela, Vesa Paatero and Antti Saarela, Self organization of a massive document collection.IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 11, NO.3, May 2000

[Kohonen, 82] Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. Biological Cybernetics, 43:59-69.
[Lagus, 97] Lagus, K. (1997). Map of WSOM'97 abstracts--alternative index. In Proceedings of WSOM'97, Workshop on Self-Organizing Maps, Espoo, Finland, June 4-6, pages 368- 372. Helsinki University of Technology, Neural Networks Research Centre, Espoo, Finland.

[Lancaster, 68] Lancaster, F.W., Evaluation of the MEDLARS Demand Search Service, National Library of Medicine, Bethesda, Maryland, 1968**.**

[Lancaster, 91] F.W. Lancaster. Indexing and abstracting in theory and practise London, the library association

[Lavrenko, 02] V. Lavrenko, 2002. A Generative Theory of Relevance. Thèse de Doctorat,University of Massachusetts.

[Leacock et al., 98] C.Leacock, G. A. Miller, and Chodorow, M. 1998. Using corpus statistics and WordNet relations for sense identification. Comput. Linguist. 24, 1 (Mar. 1998), 147-165.

[Lebart et al., 82] L. Lebart, A.Morineau, N. Tabard (1982). Techniques de la description statistique. Dunod, Paris.1982.

[Lebart et al.,00] L. Lebart, A. Morineau, M. Piron, « Statistique exploratoire multidimensionnelle », Dunod, 2000 ; section 3.5, chapitre 3.

[Lebart et SALEM.,94] L.Lebart & A.Salem, *Statistique textuelle*. Paris: Dunod.1994.

[Leclavé et al., 00] A. Le Clavé, J. Savoy, Database merging strategy based on logistic regression. IPM, pp. 341-359, 2000.

[Lee et al., 90] K. F. Lee, H. W. Hon, and R. Reddy, "An overview of the SPHINX speech recognition system," IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 38, no. 1, pp. 35–45, Jan. 1990.
[Leggetter et Woodland, 95] C.J. Leggetter and P.C. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. Computer Speech and Language, 9(2), 1995.
[Lewis, 92a] David D. Lewis. 1992a. An evaluation of phrasal and clustered representations on a text categorization task. In SIGIR '92 : Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval, pages 37–50, New York, NY, USA. ACM Press.

[Lewis, 92b] David D. Lewis. 1992b. Text representation for intelligent text retrieval : a classification-oriented view. pages 179–197.
[Li et al., 02] X. X. Li, Y. Zhao, X. Pi, L. H. Liang, and A. V. Nefian, "Audio-visual continuous speech recognition using a coupled hidden Markov model," in Proc. 7th International Conf. Spoken Language Processing, Denver, CO, Sept. 2002, pp. 213–216.

[Liang et al., 08] A.C. Liang, Boris Lauser, Margherita Sini, Johannes Keizer et Stephen Katz, D'AGROVOC à l'Agricultural Ontology Service / Concept Server Un modèle OWL pour la création d'ontologies dans le domaine de l'agriculture.2008

[Lin, 98] D. Lin. (1998) An information-theoretic definition of similarity. In Proceedings of 15th International Conference On Machine Learning, 1998.

[Lopez, 99] F. M.Lôpez, (1999). Overview of methodologies for building ontologies. In *Proceedings of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5).*

[MacFarlane et al., 00] Andy MacFarlane, Julie A. McCann, Stephen E. Robertson: Parallel Search Using Partitioned Inverted Files. SPIRE 2000: 209-220.

[Manjunath et al., 02] B. Manjunath**,** P. Salembier, et T. Sikora, 2002. Introduction to MPEG-7 : Multimedia Content Description Interface. Wiley & Sons.

[Maron et Kuhns, 60] M. E. Maron and J. L. Kuhn, On relevance, probabilistic indexing and information retrieval, Journal of the ACM, 7, 1960, pp 216-244. Reedited in Readings in Information Retrieval, K. Spark Jones and P. Willett, Morgan Kaufman, pp. 39-46, 1997.

[Merkl, 97] Dieter Merkl, Exploration of Text Collections with Hierarchical Feature Maps, in Research and Development in Information Retrieval, 186-195, 1997.

[Miller, 00] L. Miller, (2000). Ontologies and metadata. In Semantic Web Technologies Workshop. A Draft Discussion.

[Mizoguchi, 04] R. Mizoguchi, (2004). Tutorial on ontological engineering. part 2 : Ontology development, tools and languages. New Generation Computing, 22 :61-96.

[Mooers, 48] C.N.Mooers**,** Application of Random Codes to the Gathering of Statistical Information, MIT Master's Thesis, 1948.

[Moore, 65] G. Moore, Cramming more components onto intergrated circuits, Electronics Magazine,38(8), pp. 114-117, 1965.

[Mozer, 84] M.C.Mozer, Inductive information retrieval using parallel distributed computation ICS T.R. 84 06. La Jolla: UCSD, 1984.

[Muhammad, 90] A. Muhammad, "Alaswaat Alaghawaiyah," Daar Alfalah, Jordan, 1990 (in Arabic).

[Mulhem et Nigay, 96] P. Mulhem and L. Nigay. Interactive information retrieval systems: From user centered interface design to software design. In Proc. of ACM SIGIR'96, Zürich, Switzerland, pp.326-334, 1996.

[Nakache, 06] Didier Nakache**,** Elisabeth Métais: "Indicators and methodology for evaluation" In KES 05, Melbourne, septembre 2005.

[Nie et al., 99] Fuji Ren, Lixin Fan, Jian-Yun Nie, SAAK Approach: How to Acquire Knowledge in an Actual Application System, IASTED International Conference on Artificial Intelligence and Soft Computing, Honolulu, 1999, pp.136-140.

[Norman, 86] D.Norman, Cognitive Engineering, Chapter 3 in User Centered System Design, New Perspectives in Human Computer Interaction, Hilldale, New-Jersey: Lawrence Erlbaum Associates, pp. 31-61, 1986.

[Ounis et Pasça, 98] I. Ounis and M. Pasça, RELIEF: Combining expressiveness and rapidity into a single system, ACM SIGIR 1998, Melbourne, Australia, pp. 266-274, 1998.

[Pallett et al., 90,92-95]
D.S. Pallett (1990), "Issues in Spoken Language System Performance Assessment in the United States.", International Symposium on International Coordination and Standardization of Speech Databases and Assessment Techniques for Speech Input/Output, Kobe, Nov. 1990" Resource Management Corpus - Continuous Speech Recognition.",
D.S. Pallett and J.G. Fiscus (1992) , September 1992 Test Set Benchmark Test Results," Final review of the DARPA ANNT Speech Program, September"Benchmark Tests for the DARPA Spoken Language Program.",
D.S. Pallett, J.G. Fiscus, W.M. Fisher and J.S. Garofolo (1993) , DARPA Human Language Technology Workshop"Benchmark Tests for the Spoken Language Program.",
D.S. Pallett, J.G. Fiscus, W.M. Fisher, J. Garofolo, B. Lund, A. Martin and M.A. Przybocki. (1994) , DARPA Workshop on Human Language Technology, November 1994"RoutePlanner, un système de navigation flexible.", M. Pallme, G. Maggia (1995) , SIA/FIEV/EQUIP'AUTO, 17-18 octobre 1995

[Parekh et al., 04] V. Parekh, J.-P. J.Gwo, and T.Finin, (2004). Ontology based Semantic Metadata for GeoScience Data. In Proceedings of the International Conference of Information and Knowledge Engineering, pages 485-490. The International MultiConference in Computer Science and Computer Engineering.

[Patwardhan et al., 03] S. Patwardhan,S. Banerjee, and T. Pedersen : Using measures of semantic relatedness for word sense disambiguation. In Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics CICLING, Mexico City, 2003.

[Pavel, 02] Pavel Berkhin, Accrue Software, San Jose, CA, 2002, Survey Of Clustering Data Mining Techniques.

[Ponte et Croft, 98] J. M. Ponte et W. B. Croft, 1998. A Language Modeling Approach to Information Retrieval. Dans les actes de ACM Special Interest Group on Information Retrieval (SIGIR), 275–281.

[Preece et al., 94] J. Preece, Y. Rogers, H. Sharp, D. Benyonm, S. Holland and T. Carey. Human Computer Interaction. Addison-Wesley, 1994.

[Przybocki et al., 98] M. Przybocki, J. Fiscus, J. Garofolo, et D. Pallett, 1998. HUB-4 Information Extraction Evaluation. Dans les actes de DARPA Broadcast News Transcription and Understanding Workshop, 13–18.

[Rada et al. ,89] R.Rada, H.Mili, E.Bicknell, , and M.Blettner, (1989). Development and application of a metric on semantic nets. IEEE Transaction on Systems, Man, and Cybernetics, 19(1):17–30.

[Rastier et al., 94] RASTIER, F., M. CAVAZZA et A. ABEILLÉ (1994), *Sémantique pour l'analyse*, Paris, Masson, 240 p.

[Rastier, 87] RASTIER, F., Sémantique interprétative, Paris : Presses Universitaires de France, 1987, (2e éd. revue et augmentée 1996).

[Rauber et Merkl, 01] A. Rauber, and D. Merkl. Automatic Labeling of Self-Organizing Maps for Information Retrieval. In Journal of Systems Research and Information Systems (JSRIS), 10(10):23–45. December 2001.

[Ravishankar,96] M. K. Ravishankar, "Efficient algorithms for speech recognition," PhD Thesis (CMU Technical Report CS-96-143), Carnegie Mellon University, Pittsburgh, PA, 1996.

[Réhel, 05] Catégorisation automatique de textes et cooccurrence de mots provenant de documents non étiquetés. Réhel, Simon. Thèse. Maître ès sciences (M.Sc.). Université Laval http://www.theses.ulaval.ca/2005/22376/22376.html

[Renals et Hochberg, 99] S. Renals and M. Hochberg, ``Start-synchronous search for large vocabulary continuous speech recognition," IEEE Trans. Speech and Audio Processing, in press.1999

[Resnik, 95] P. Resnik, "Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language", Journal of Artificial Intelligence Research (JAIR), 11, pp. 95-130, 1999.

[Ribeiro-Neto et Muntz, 96] B. Ribeiro-Neto et R. Muntz, 1996. A Belief Network Model for IR. Dans les actes de ACM Special Interest Group on Information Retrieval (SIGIR), 253–260.

[Rijsbergen, 79] C. J. Van Rijsbergen**,** Information retrieval. London: Butterworth, (1975).

[Robertson et al., 94] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In NIST Special Publication 500-226 : Overview of the Third Text REtrieval Conference (TREC-3), November 1994.

[Robertson et Spärck-Jones, 88] S. Robertson et K. Spärck-Jones, 1988. Relevance Weighting of Search Terms. Taylor Graham Series In Foundations Of Information Science 27, 143–160.

[Robertson et Walker, 94] S. Robertson et S. Walker, 1994. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. Dans les actes de ACM Special Interest Group on Information Retrieval (SIGIR), 232–241. Springer-Verlag New York, Inc. New York, NY, USA.

[Robinson et al. ,96] T.Robinson, M.Hochberg, and S.Renals, The use of recurrent networks in continuous speech recognition. In Lee, C. H., Paliwal, K. K., and Soong, F. K., editors, Automatic Speech and Speaker Recognition - Advanced Topics, chapter 10, pages 233{258. Kluwer Academic Publishers, (1996).

[Robinson et al.,98] T.Robinson, and J.Christie, Timerst search for large vocabulary speech recognition. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), (1998). , pp. 829-832.

[Robinson et al.,99] T.Robinson, D.Abberley, D.Kirby, and S.Renals, Recognition, indexing and retrieval of british broadcast news with the THISL system. In Proceedings of 6th European Conference on Speech Communication and Technology, Budabest, Hungary, (1999).

[Robinson,94] A. J. Robinson, ``The application of recurrent nets to phone probability estimation," *IEEE Trans. Neural Networks*, vol. 5, pp. 298-305, 1994.

[Rowley, 88] J.E. Rowley. Abstracting and indexing (2nd edition). Londres, Clive Bingley

[Russell, 00] Bertrand Russell 1921-1970: The Ghost of Madness,Ray Monk, London: Jonathan Cape 2000.

[Salton et al., 71] G. Salton, A Comparison between manual and automatic indexing methods. Journal of the American Documentation, 20(1), pp. 6171, 1971.

[Salton et al., 75]G. Salton**,** A. Wong, et C. S. Yang, 1975. A Vector Space Model for Automatic Indexing. Communications of the ACM 18(11), 613–620.

[Salton et al., 83]G. Salton, E. Fox, et H. Wu, 1983. Extended Boolean Information Retrieval. Communications of the ACM 26(11), 1022–1036.

[Salton, 71] G. Salton, The SMART Retrieval System. Prentice Hall, Englewood Cliffs, NJ, 1971

[Salton, 75] G. Salton, A. Wong, and C. S. Yang. 1975. A vector space model for automatic indexing. Communications of the ACM, 18(11): 613–620.

[Salton, 83] G. Salton et M.J. McGill. Introduction to modern information retrieval McGraw Hill Book Company, New York, 1983

[Sanderson et Shou, 02] M. Sanderson et X. M. Shou, 2002. Speech and Hand Transcribed Retrieval. Dans les actes de ACM Special Interest Group on Information Retrieval (SIGIR). Springer.

[Sanderson, 00] M.Sanderson, Retrieving with good sense, Information Retrieval Vol. 2 No. 1, pp 49-69, 2000.

[Savoy et Berger, 05] J. Savoy et P. Berger, 2005. Report on CLEF-2005 Evaluation Campaign : Monolingual, Bilingual, and GIRT Information Retrieval. Dans les actes de Cross Language Evaluation Forum (CLEF).

[Sebastiani, 04] Fabrizio Sebastiani: "Text Categorization" - In Alessandro Zanasi (ed.), Text Mining and its Applications, WIT Press, Southampton, UK. Forth-coming.

[Sebrechts et al., 99] M. Sebrechts, J. Vasilakis, M. Miller, John Cugini and S. Laskowski, Visualization of Search Results:A Comparative Evaluation of Text, 2D, and 3D Interfaces, ACM SIGIR 1999, Berkeley, USA, pp. 3-10, 1999.

[Smeaton et Agosti, 96] A. F. Smeaton and M. Agosti, Information Retrieval and Hypertext, Kluwer Academic Press, 1996.

[Smeaton et al., 98] A. F. Smeaton, M. Morony, G. Quinn, and R. Scaife. Taisceala i: Information retrieval from an archive of spoken radio news. In Proc. Second European Digital Libraries Conference, LNCS 1513, pp. 429-442, 1998.

[Smirnov et al., 05] A..Smirnov, M.Pashkin, N.Chilov, T.Levashova, A .Krizhanovsky., A.. Kashevnik 2005. Ontology-Based Users and Requests Clustering in Customer Service Management System. In: (V.Gorodetsky, J.Liu, V.Skormin, eds.) Autonomous Intelligent Systems: Agents and Data Mining: International Workshop, AIS-ADM 2005.

[Song et Croft, 99] F. Song and W. B. Croft. A General Language Model for Information Retrieval, ACM CIKM'99, USA, pp. 316-321, 1999.

[Soule Dupuy, 90] Chantal Soule-Dupuy**.** Systèmes de recherche d'information : le système Videotex Infodiab. Mecanismes d'indexation et d'interrogation. Thèse de Doctorat, Université Paul Sabatier, Toulouse, France.

[Sowa, 84] J.Sowa, Conceptual Structures: Information Processing In Mind and Machine. Addison-Wesley, 1984.

[Spärck-Jones et al., 00] K. Spärck-Jones, S. Walker, et S. Robertson, 2000. A probabilistic model of information retrieval : development and comparative experiments. Information Processing and Management : an International Journal 36(6), 779–808.

[Spärck-Jones, 79] Karen Spärck-Jones: Experiments in relevance weighting of search terms. Inf. Process. Manage. 15(3): 133-144, 1979.

[Studer, 98] B. F. Studer, (1998). Knowledge engeeneering : Principles and methods. Data and Knowledge Engineering, 25 :161-197.

[Thong et al., 00] J.-M. V. Thong, D. Goddeau, A. Litvinova, B. Logan, P. Moreno, et M. Swain, 2000. SpeechBot a Speech Recognition Based Audio Indexing System. Dans les actes de Recherche d'Information Assistée par Ordinateur (RIAO).

[Timo et al., 97] Timo Honkela, Samuel Kaski, Krista Lagus and Teuvo Kohonen: WEBSOM - Self-Organizing Maps of Document Collections. Proceedings of WSOM'97, Workshop on Self-Organizing Maps, Espoo, Finland, June 4-6. Helsinki University of Technology, Neural Networks Research Centre, 1997.

[Turenne, 01] Nicolas Turenne : "Etat de l'art de la classification automatique pour l'acquisition de connaissances à partir de textes." UMR INRA-INAPG – Biométrie et Intelligence Artificielle (BIA). INRIA, Technical report. 2001**.**

[Turtle et Croft, 90] H. Turtle and W. B. Croft, Inference Networks for Document Retrieval, ACM SIGIR 90, Belgium, pp. 1-24, 1990.

[Ultsch, 03] A. Ultsch, Maps for the Visualization of high-dimensional Data Spaces , in Proc. Workshop on Self organizing Maps, pp 225 - 230, Kyushu, Japan, 2003.

[Uschold et Gruninger, 96] Ontologies: Principles, Methods and Applications".Knowledge Engineering Review, vol.11, n°2, p. 93-136,1996

[Vallet et al., 05] D.Vallet, M.Fernández, P.Castells, An Ontology-Based Information Retrieval Model, Proceedings of the 2nd European Semantic Web Conference, pp 455-470, 2005.

[Van Heijst et al., 97] G. van Heijst, A. Th. Schreiber and B. J. Wielinga., Using explicit ontologies for KBS development. International Journal of Human-Computer Studies 1997.

[Van Rijsbergen, 89] C. J. van Rijsbergen, Toward a New Information Logic, ACM SIGIR'89, USA, pp. 77-86, 1989.

[Wechsler et al., 98] M.Wechsler, E. Munteanu, et P. Schauble, 1998. New Techniques for Open Vocabulary Spoken Document Retrieval. Dans les actes de ACM Special Interest Group on Information Retrieval (SIGIR), 20–27. ACM Press.

[Wei et Croft, 06] X. Wei et W. Croft, 2006. LDA-Based Document Models for Ad-Hoc Retrieval. Dans les actes de ACM Special Interest Group on Information Retrieval (SIGIR), 178–185. ACM Press New York, NY, USA.

[Wellish, 91] H.H. Wellish. Indexing from A to Z. Bronx, NY, H.W. Wilson. Retrieved August 10, 2006, from http://www.asindexing.org/site/history.shtml .1991.

[Wen et Hsin, 02] Wen-Cheng Lin and Hsin-Hsi Chen, Department of Computer Science and Information Engineering, National Taiwan University. Merging Mechanisms in Multilingual Information Retrieval,2002

[Wiki,06] http://fr.wikipedia.org/wiki/Classification_automatique

[Wilkinson et Hingston, 91] R. Wilkinson et P. Hingston, 1991. Using the Cosine Measure in a Neural Network for Document Retrieval. Dans les actes de ACM Special Interest Group on Information Retrieval (SIGIR), 202–210. ACM Press New York, NY, USA.

[Wong et al., 85] S. K. M. Wong, W. Ziarko, et P. C. N. Wong, 1985. Generalized Vector Spaces Model in Information Retrieval. Dans les actes de ACM Special Interest Group on Information Retrieval (SIGIR), 18–25.

[Wu et al., 94] Z.Wu, M.Palmer, Verb semantics and lexical selection, Proceedings of the 32nd annual meeting of the Association for Computational Linguistics, pp 133-138, 1994.

[Yong, 05] Yong Wang, A Dissertation Submitted to the Faculty of Mississippi State University, Incorporating semantic and syntactic information into document representation for document clustering, 2005

[Young, 94] S. Young, "The HTK hidden Markov model toolkit: Design and philosophy," Cambridge University Engineering Department, UK, Tech. Rep. CUED/FINFENG/TR152, Sept. 1994

[Yu et Seide, 04] P. Yu et F. Seide, 2004. A Hybrid Word/Phoneme-Based Approach for Improved Vocabulary-Independent Search in Spontaneous Speech. Dans les actes de International Conference on Spoken Language Processing (ICSLP).

Annexes :

1 : Transcription des messages

2 : Programme principale (matlab)

Annexe 1 : Transcription des messages

Le traitement automatique du langage parlé, le paradigme d'évaluation tel que défini dans les projets américains (DARPA (Defence Advanced Research Project Agency) [Pallett et al., 90,92-95] et européens [Chollet et al., 94, 95] a pris une importance considérable et a montré qu'il était un facteur d'amélioration constante des performances obtenues : on est passé en dix ans de systèmes capables de reconnaître des mots isolés, un seul locuteur prononçant un lexique d'une cinquantaine de mots, à des systèmes capables de reconnaître en parole continue n'importe quel locuteur utilisant un vocabulaire de 60 000 mots et plus; ou encore à des systèmes capables de conduire un dialogue avec un vocabulaire d'un millier de mots à travers le téléphone, sur un sujet bien défini (renseignements concernant les horaires de train ou d'avion).

Ces démonstrateurs développés pour la plupart aux Etats Unis, au Canada (CRIM) et en Europe (Philips-Aix-la-Chapelle en Allemagne, l'Université de Cambridge en Grande Bretagne, LIMSI-CNRS en France) obtiennent des taux d'erreur, au niveau mot, inférieur à 10% pour la dictée de textes lus et à 19% pour la parole spontanée. De tels progrès sont directement liés à l'accroissement de la puissance de calcul qui a permis de développer des modèles de plus en plus complexes et d'analyser les données des corpus pour estimer les paramètres de ces modèles, mais ils sont également dus à l'organisation de campagnes d'évaluation systématiques et à la disponibilité croissante de très grands corpus (textes et parole) que ces campagnes de tests ont aidé à mettre à disposition.

La transcription automatique des appels téléphoniques pose des problèmes spécifiques. Citons les principaux : la longueur des fichiers de parole qui ne sont pas préalablement segmentés en phrases, les fréquents changements de locuteurs, parfois non natifs, la superposition de parole et de musique, les alternances de parole large bande et de parole téléphonique, la présence de différents types de bruits et la parole simultanée.

La Reconnaissance Automatique de la Parole (RAP) est une technologie informatique permettant à un logiciel d'interpréter une langue naturelle humaine. Elle permet à une machine d’extraire le message oral contenu dans un signal de parole. Cette technologie utilise des méthodes informatiques des domaines du traitement du signal et de l’intelligence artificielle [Barras, 96]. Vu l’importance de la RAP, plusieurs systèmes ont été développés pour la reconnaissance vocale, parmi les plus connus: Dragon Naturally Speaking, IBM Via voice, Microsoft SAPI et d’autres. Il y a aussi des Open Sources comme HTK [Young, 94], ISIP [Deshmukh et al., 99], AVCSR [Li et al., 02] et CMU Sphinx [Lee et al., 90] [Huang et al., 93] [Ravishankar, 96]. Nous nous sommes intéressés à ce dernier qui est un système basé sur les Modèles de Markov Cachés (MMC) [Huang et al., 90] ; nous avons constaté que le système de reconnaissance de la parole CMU Sphinx 4 est librement disponible (Open Source) et il est actuellement l’un des systèmes de reconnaissance de parole les plus puissants. Le CMU Sphinx permet à des groupes de recherche de développer et de conduire des applications de recherches dans la reconnaissance de la parole. Pour ces raisons et d’autres, nous avons choisi ce système pour développer notre application de transcription.

- **Présentation du CMU Sphinx 4**

Sphinx est un projet lancé par l'université Carnegie Mellon (CMU) dans le but de concevoir un environnement pour la recherche dans le domaine de la reconnaissance automatique de la parole. CMU Sphinx 4 est une librairie de classes et d'outils disponibles en langage de programmation Java. Cette librairie est gratuite à télécharger, elle vise principalement à faciliter la construction des systèmes de reconnaissance vocale.

CMU Sphinx-4 est un système de RAP basé sur les Modèles de Markov Cachés (HMM). Il a été créé conjointement par le groupe Sphinx à l'université CMU, les laboratoires Sun Microsystems et Hewlett-Packard company [Hong, 02] [Carnegie Mellon University] [Huang, 89].

SphinxTrain est l'outil crée par CMU pour le développement des modèles acoustiques. C'est un ensemble de programmes et documentations pour réaliser et construire des modèles acoustiques pour n'importe quelle langue.

- **Architecture**

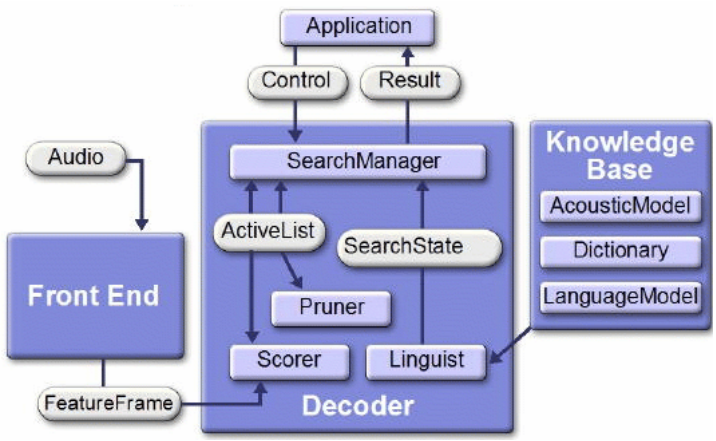


Figure A: Architecture du CMU Sphinx-4.

Sphinx-4 présente un ensemble d'outils de reconnaissance vocale (voir Figure A) flexibles, modulaires et extensibles formant, un véritable banc d'essais et un puissant environnement de recherche pour les technologies de reconnaissance automatique de la parole.

• **FrontEnd** : découpe la voix enregistrée en différentes parties et les prépare pour le décodeur. Il est responsable de la génération des vecteurs caractéristiques représentant les caractéristiques du signal vocal.

• **Features** : et utilisé pour estimer les paramètres du modèle acoustique.

• **Linguist** : ou base de connaissances qui est l'information qu'utilise le décodeur pour déterminer les mots et les phrases prononcées, elle est composée de :

- **Dictionary**.

- **AcousticModel** : modèle acoustique, un modèle statistique décrivant la distribution des données de phonèmes.

- **LanguageModel** : un modèle de langage, c'est un modèle qui défini l'usage des mots dans une application. Chaque mot dans le modèle de langage doit être dans le dictionnaire de prononciation, pour cela il donne la probabilité d'apparition d'un mot donné, basée sur des connaissances tirées du dictionnaire. Le choix d'un modèle de langue dépend de l'application.

• **SearchGraph** : contient toutes les séquences de phonèmes possibles basées sur le LanguageModel.

• **Decoder** : ou Décodeur qui est le coeur de Sphinx-4 ; c'est lui qui traite les informations reçues depuis le FrontEnd, il les analyse et les compare à la base de connaissances pour donner un résultat à l'application.

- ## Installation

a. Sphinx-4

Sphinx-4 peut être téléchargé de l'internet soit sous forme binaire soit sous forme source code [http://cmusphinx.sourceforge.net/sphinx4]. Il a été compilé et testé sur plusieurs versions de Linux et sur Windows. L'exécution de Sphinx-4 demande des logiciels supplémentaires qui sont :

- Java 2 SDK, Standard Edition 5.0 [http://java.sun.com].
- Java Runtime Environement (JRE)
- Les différentes librairies qui composent Sphinx-4.
- Ant : L'outil pour faciliter la compilation en automatisant les tâches répétitives [http ://ant.apache.org.].

b. Sphinxtrain

SphinxTrain téléchargeable dont le lient se trouve dans tools du site de CMU Sphinx [http://cmusphinx.sourceforge.net]. Les différentes librairies qui composent SphinxTrain :

• ActivePerl : L'outil pour éditer des scriptes pour SphinxTrain et permet de travailler dans un Unix-like environnement pour Windows plateforme [http://www.activestate.com].

• Microsoft Visual Studio : Pour compiler les sources en C afin de produire les Exécutables.

- **Reconnaissance de la langue arabe**

La langue arabe est une langue sémitique, elle est parmi les langues les plus anciennes dans le monde [Al-Zabibi, 90]. L'arabe classique standard a 34 phonèmes parmi lesquels 6 sont voyelles et 28 sont des consonnes [Muhammad, 90]. Les phonèmes arabe se distinguent par la présence de deux classes qui sont appelées pharyngales et emphatiques. Ces deux classes sont caractéristiques des langues sémitiques comme l'hébreu [Deller et al., 93] [Elshafei et al., 91].

Les syllabes permises dans la langue arabe sont : CV, CVC et CVCC. Où le V désigne voyelle courte ou longue et le C représente une consonne [Muhammad, 90]. La langue arabe comporte cinq types de syllabes classées selon les trais ouvert/fermé et court/long. Une syllabe est dite ouverte (respectivement fermée) si elle se termine par une voyelle (respectivement une consonne). Toutes les syllabes commencent par une consonne suivie d'une voyelle et elles comportent une seule voyelle. La syllabe CV peut se trouver au début, au milieu ou à la fin du mot [El-Imam, 89] [Baloul ,03].

Annexe 2 :

Programme principale :

```
clear all
close all
echo on
clc
Gaussiennes','labels',labs,'comp_names',cnames);
sD = som_read_data('fichier1.data');
echo on
msize = [6 6];
insize = size(sD,1);
lattice = 'rect'; % 'rect' ou 'hexa'
shape = 'sheet'; % 'sheet', 'cyl', ou 'toroid'
sMap = som_map_struct(6,'msize',msize, lattice, shape);
%Initialisation des poids de la carte topologique
sMap = som_randinit(sD, sMap);% som randinit ou som lininit
%Entrainement de la carte : Phase 1 (Auto organisation)
  figure('Position',[100 100 500 500])
  epochs = 1000;
  radius_ini = 3;
  radius_fin = 0.2;
  Neigh = 'gaussian'; % 'gaussian', 'cutgauss', 'bubble' ou 'ep'
  tr_lev = 3 ;
  [sMap,sT] = som_batchtrain(sMap,
sD,'trainlen',epochs,'radius_ini',radius_ini,'radius_fin',radius_fin,'neigh',N
eigh,'tracking',tr_lev);
  xlabel('AUTO ORGANISATION')
  pause % Strike any key to continue...
%Entrainement de la carte : Phase 2 (Convergence)
  epochs = 1000;
  radius_ini = 0.2;
  radius_fin = 0.1;
  figure('Position',[100 100 500 500])
  [sMap,sT] = som_batchtrain(sMap,
sD,'trainlen',epochs,'radius_ini',radius_ini,'radius_fin',radius_fin,
'neigh',Neigh,'tracking',tr_lev);
  xlabel('CONVERGENCE')
  pause % Strike any key to continue...
  sMap = som_autolabel(sMap,sD,'vote');
  som_show(sMap ,'umat','all','comp',[1:6],'empty','Labels','norm','d');
  som_show_add('label',sMap.labels,'textsize',8,'textcolor','r','subplot',8);
 pause % Strike any key to continue...
subplot(1, 3,1)
som_grid(sMap, 'coord',sMap.codebook(:,[1 2]))
title('Map in input space')
subplot (1, 3, 2)
som_grid (sMap)
axis([0 11 0 11]),view(0,-90),title('Map in output space')
  pause % Strike any key to continue...
sM = som_supervised(sD,'big','lattice','rect','neigh', 'gaussian');
som_show(sM,'umat','all');
som_show_add('label',sM.labels,'TextSize',10,'TextColor','r')
sD2 = som_label(sD,'clear','all');
sD2 = som_autolabel(sD2,sM);       % classification
```

```
pause % Strike any key to continue...
sD1 = som_read_data('fichier2.data');
sM = som_supervised(sD1,'big','lattice','rect','neigh', 'gaussian');
som_show(sM,'umat','all');
som_show_add('label',sM.labels,'TextSize',10,'TextColor','r')
sD2 = som_label(sD1,'clear','all');
sD2 = som_autolabel(sD2,sM);       % classification
```

Code source des fonctions utilisé dans le programme principale : som_read_data, som_map_struct, som_randinit, som_batchtrain, som_show, som_supervised.

```
function sData = som_read_data(filename, varargin)
error(nargchk(1, 3, nargin))  % check no. of input args is correct
dont_care       = 'NaN';  % default don't care string
comment_start   = '#';    % the char a SOM_PAK command line starts with
comp_name_line  = '#n';   % string denoting a special command line,
                          % which contains names of each component
label_name_line = '#l';   % string denoting a special command line,
                          % which contains names of each label
block_size      = 1000;   % block size used in file read
kludge          = num2str(realmax, 100); % used in sscanf
% open input file
fid = fopen(filename);
if fid < 0
  error(['Cannot open ' filename]);
end
% process input arguments
if nargin == 2
  if isstr(varargin{1})
    dont_care = varargin{1};
  else
    dim      = varargin{1};
  end
elseif nargin == 3
  dim       = varargin{1};
  dont_care = varargin{2};
end
% if the data dimension is not specified, find out what it is
if nargin == 1 | (nargin == 2 & isstr(varargin{1}))
  fpos1 = ftell(fid); c1 = 0;      % read first non-comment line
  while c1 == 0,
    line1 = strrep(fgetl(fid), dont_care, kludge);
    [l1, c1] = sscanf(line1, '%f ');
  end
  fpos2 = ftell(fid); c2 = 0;      % read second non-comment line
  while c2 == 0,
    line2 = strrep(fgetl(fid), dont_care, kludge);
    [l2, c2] = sscanf(line2, '%f ');
  end
  if (c1 == 1 & c2 ~= 1) | (c1 == c2 & c1 == 1 & l1 == 1)
    dim = l1;
    fseek(fid, fpos2, -1);
  elseif (c1 == c2)
    dim = c1;
```

```
    fseek(fid, fpos1, -1);
    warning on
    warning(['Automatically determined data dimension is ' ...
         num2str(dim) '. Is it correct?']);
  else
    error(['Invalid header line: ' line1]);
  end
end
if dim < 1 | dim ~= round(dim)
  error(['Illegal data dimension: ' num2str(dim)]);
end
sData       = som_data_struct(zeros(1, dim), 'name', filename);
lnum        = 0;                                    % data vector counter
data_temp   = zeros(block_size, dim);
labs_temp   = cell(block_size, 1);
comp_names  = sData.comp_names;
label_names = sData.label_names;
form        = [repmat('%g',[1 dim-1]) '%g%[^ \t]'];

limit       = block_size;
while 1,
  li = fgetl(fid);                          % read next line
  if ~isstr(li), break, end;                % is this the end of file?

  % all missing vectors are replaced by value realmax because
  % sscanf is not able to read NaNs
  li = strrep(li, dont_care, kludge);
  [data, c, err, n] = sscanf(li, form);
  if c < dim % if there were less numbers than dim on the input file line
    if c == 0
      if strncmp(li, comp_name_line, 2) % component name line?
    li = strrep(li(3:end), kludge, dont_care); i = 0; c = 1;
    while c
      [s, c, e, n] = sscanf(li, '%s%[^ \t]');
      if ~isempty(s), i = i + 1; comp_names{i} = s; li = li(n:end); end
    end

    if i ~= dim
      error(['Illegal number of component names: ' num2str(i) ...
         ' (dimension is ' num2str(dim) ')']);
    end
      elseif strncmp(li, label_name_line, 2) % label name line?
    li = strrep(li(3:end), kludge, dont_care); i = 0; c = 1;
    while c
      [s, c, e, n] = sscanf(li, '%s%[^ \t]');
      if ~isempty(s), i = i + 1; label_names{i} = s; li = li(n:end); end
    end
      elseif ~strncmp(li, comment_start, 1) % not a comment, is it error?
    [s, c, e, n] = sscanf(li, '%s%[^ \t]');
    if c
      error(['Invalid vector on input file data line ' ...
         num2str(lnum+1) ': [' deblank(li) ']']),
    end
      end
    else
      error(['Only ' num2str(c) ' vector components on input file data line '
...
```

```
          num2str(lnum+1) ' (dimension is ' num2str(dim) ')']);
    end

  else

    lnum = lnum + 1;                % this was a line containing data vector
    data_temp(lnum, 1:dim) = data'; % add data to struct

    if lnum == limit       % reserve more memory if necessary
      data_temp(lnum+1:lnum+block_size, 1:dim) = zeros(block_size, dim);
      [dummy nl] = size(labs_temp);
      labs_temp(lnum+1:lnum+block_size,1:nl) = cell(block_size, nl);
      limit = limit + block_size;
    end

    % read labels

    if n < length(li)
      li = strrep(li(n:end), kludge, dont_care); i = 0; n = 1; c = 1;
      while c
    [s, c, e, n_new] = sscanf(li(n:end), '%s%[^ \t]');
    if c, i = i + 1; labs_temp{lnum, i} = s; n = n + n_new - 1; end
      end
    end
  end
end

% close input file
if fclose(fid) < 0, error(['Cannot close file ' filename]);
else fprintf(2, '\rdata read ok         \n'); end

% set values
data_temp(data_temp == realmax) = NaN;
sData.data        = data_temp(1:lnum,:);
sData.labels      = labs_temp(1:lnum,:);
sData.comp_names  = comp_names;
sData.label_names = label_names;

return;
```

```
function sMap = som_map_struct(dim, varargin)
% default values
sTopol     = som_set('som_topol','lattice','hexa','shape','sheet');
neigh      = 'gaussian';
mask       = ones(dim,1);
name       = sprintf('SOM %s', datestr(now, 1));
labels     = cell(prod(sTopol.msize),1);
for i=1:length(labels), labels{i} = ''; end
comp_names = cell(dim,1);
for i = 1:dim, comp_names{i} = sprintf('Variable%d', i); end
comp_norm  = cell(dim,1);
% varargin
i=1;
while i<=length(varargin),
  argok = 1;
  if ischar(varargin{i}),
    switch varargin{i},
      % argument IDs
     case 'mask',       i=i+1; mask = varargin{i};
     case 'msize',      i=i+1; sTopol.msize = varargin{i};
     case 'labels',     i=i+1; labels = varargin{i};
     case 'name',       i=i+1; name = varargin{i};
     case 'comp_names', i=i+1; comp_names = varargin{i};
     case 'comp_norm',  i=i+1; comp_norm = varargin{i};
     case 'lattice',    i=i+1; sTopol.lattice = varargin{i};
     case 'shape',      i=i+1; sTopol.shape = varargin{i};
     case {'topol','som_topol','sTopol'}, i=i+1; sTopol = varargin{i};
     case 'neigh',      i=i+1; neigh = varargin{i};
      % unambiguous values
     case {'hexa','rect'}, sTopol.lattice = varargin{i};
     case {'sheet','cyl','toroid'}, sTopol.shape = varargin{i};
     case {'gaussian','cutgauss','ep','bubble'}, neigh = varargin{i};
     otherwise argok=0;
    end
  elseif isstruct(varargin{i}) & isfield(varargin{i},'type'),
    switch varargin{i}(1).type,
     case 'som_topol', sTopol = varargin{i};
     otherwise argok=0;
    end
  else
    argok = 0;
  end
  if ~argok,
    disp(['(som_map_struct) Ignoring invalid argument #' num2str(i+1)]);
  end
  i = i+1;
end

% create the SOM
codebook = rand(prod(sTopol.msize),dim);
sTrain = som_set('som_train','time',datestr(now,0),'mask',mask);
sMap = som_set('som_map','codebook',codebook,'topol',sTopol,...
                         'neigh',neigh,'labels',labels,'mask',mask,...
```

```
                        'comp_names',comp_names,'name',name,...
                        'comp_norm',comp_norm,'trainhist',sTrain);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function sMap = som_randinit(D, varargin)
% data
if isstruct(D),
  data_name = D.name;
  comp_names = D.comp_names;
  comp_norm = D.comp_norm;
  D = D.data;
  struct_mode = 1;
else
  data_name = inputname(1);
  struct_mode = 0;
end
[dlen dim] = size(D);

% varargin
sMap = [];
sTopol = som_topol_struct;
sTopol.msize = 0;
munits = NaN;
i=1;
while i<=length(varargin),
  argok = 1;
  if ischar(varargin{i}),
    switch varargin{i},
     case 'munits',     i=i+1; munits = varargin{i}; sTopol.msize = 0;
     case 'msize',      i=i+1; sTopol.msize = varargin{i};
                               munits = prod(sTopol.msize);
     case 'lattice',    i=i+1; sTopol.lattice = varargin{i};
     case 'shape',      i=i+1; sTopol.shape = varargin{i};
     case {'som_topol','sTopol','topol'}, i=i+1; sTopol = varargin{i};
     case {'som_map','sMap','map'}, i=i+1; sMap = varargin{i}; sTopol =
sMap.topol;
     case {'hexa','rect'},          sTopol.lattice = varargin{i};
     case {'sheet','cyl','toroid'}, sTopol.shape = varargin{i};
     otherwise argok=0;
    end
  elseif isstruct(varargin{i}) & isfield(varargin{i},'type'),
    switch varargin{i}.type,
     case 'som_topol',
      sTopol = varargin{i};
     case 'som_map',
      sMap = varargin{i};
      sTopol = sMap.topol;
     otherwise argok=0;
    end
  else
    argok = 0;
  end
  if ~argok,
    disp(['(som_topol_struct) Ignoring invalid argument #' num2str(i)]);
  end
  i = i+1;
end
```

```
if ~isempty(sMap),
  [munits dim2] = size(sMap.codebook);
  if dim2 ~= dim, error('Map and data must have the same dimension.'); end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% create map

% map struct
if ~isempty(sMap),
  sMap = som_set(sMap,'topol',sTopol);
else
  if ~prod(sTopol.msize),
    if isnan(munits),
      sTopol = som_topol_struct('data',D,sTopol);
    else
      sTopol = som_topol_struct('data',D,'munits',munits,sTopol);
    end
  end
  sMap = som_map_struct(dim, sTopol);
end

if struct_mode,
  sMap = som_set(sMap,'comp_names',comp_names,'comp_norm',comp_norm);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% initialization

% train struct
sTrain = som_train_struct('algorithm','randinit');
sTrain = som_set(sTrain,'data_name',data_name);

munits = prod(sMap.topol.msize);
sMap.codebook = rand([munits dim]);

% set interval of each component to correct value
for i = 1:dim,
  inds = find(~isnan(D(:,i)) & ~isinf(D(:,i)));
  if isempty(inds), mi = 0; ma = 1;
  else ma = max(D(inds,i)); mi = min(D(inds,i));
  end
  sMap.codebook(:,i) = (ma - mi) * sMap.codebook(:,i) + mi;
end

% training struct
sTrain = som_set(sTrain,'time',datestr(now,0));
sMap.trainhist = sTrain;

return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function [sMap,sTrain] = som_batchtrain(sMap, D, varargin)
error(nargchk(2, Inf, nargin));  % check the number of input arguments

% map
struct_mode = isstruct(sMap);
if struct_mode,
  sTopol = sMap.topol;
else
  orig_size = size(sMap);
  if ndims(sMap) > 2,
    si = size(sMap); dim = si(end); msize = si(1:end-1);
    M = reshape(sMap,[prod(msize) dim]);
  else
    msize = [orig_size(1) 1];
    dim = orig_size(2);
  end
  sMap   = som_map_struct(dim,'msize',msize);
  sTopol = sMap.topol;
end
[munits dim] = size(sMap.codebook);

% data
if isstruct(D),
  data_name = D.name;
  D = D.data;
else
  data_name = inputname(2);
end
nonempty = find(sum(isnan(D),2) < dim);
D = D(nonempty,:);                    % remove empty vectors from the data
[dlen ddim] = size(D);                % check input dimension
if dim ~= ddim,
  error('Map and data input space dimensions disagree.');
end

% varargin
sTrain = som_set('som_train','algorithm','batch','neigh', ...
         sMap.neigh,'mask',sMap.mask,'data_name',data_name);
radius     = [];
tracking   = 1;
weights    = 1;

i=1;
while i<=length(varargin),
  argok = 1;
  if ischar(varargin{i}),
    switch varargin{i},
     % argument IDs
     case 'msize', i=i+1; sTopol.msize = varargin{i};
     case 'lattice', i=i+1; sTopol.lattice = varargin{i};
     case 'shape', i=i+1; sTopol.shape = varargin{i};
     case 'mask', i=i+1; sTrain.mask = varargin{i};
     case 'neigh', i=i+1; sTrain.neigh = varargin{i};
```

```
      case 'trainlen', i=i+1; sTrain.trainlen = varargin{i};
      case 'tracking', i=i+1; tracking = varargin{i};
      case 'weights', i=i+1; weights = varargin{i};
      case 'radius_ini', i=i+1; sTrain.radius_ini = varargin{i};
      case 'radius_fin', i=i+1; sTrain.radius_fin = varargin{i};
      case 'radius',
       i=i+1;
       l = length(varargin{i});
       if l==1,
         sTrain.radius_ini = varargin{i};
       else
         sTrain.radius_ini = varargin{i}(1);
         sTrain.radius_fin = varargin{i}(end);
         if l>2, radius = varargin{i}; end
       end
      case {'sTrain','train','som_train'}, i=i+1; sTrain = varargin{i};
      case {'topol','sTopol','som_topol'},
       i=i+1;
       sTopol = varargin{i};
       if prod(sTopol.msize) ~= munits,
         error('Given map grid size does not match the codebook size.');
       end
       % unambiguous values
      case {'hexa','rect'}, sTopol.lattice = varargin{i};
      case {'sheet','cyl','toroid'}, sTopol.shape = varargin{i};
      case {'gaussian','cutgauss','ep','bubble'}, sTrain.neigh = varargin{i};
      otherwise argok=0;
     end
   elseif isstruct(varargin{i}) & isfield(varargin{i},'type'),
     switch varargin{i}(1).type,
      case 'som_topol',
       sTopol = varargin{i};
       if prod(sTopol.msize) ~= munits,
         error('Given map grid size does not match the codebook size.');
       end
      case 'som_train', sTrain = varargin{i};
      otherwise argok=0;
     end
   else
     argok = 0;
   end
   if ~argok,
     disp(['(som_batchtrain) Ignoring invalid argument #' num2str(i+2)]);
   end
   i = i+1;
end

% take only weights of non-empty vectors
if length(weights)>dlen, weights = weights(nonempty); end

% trainlen
if ~isempty(radius), sTrain.trainlen = length(radius); end

% check topology
if struct_mode,
   if ~strcmp(sTopol.lattice,sMap.topol.lattice) | ...
     ~strcmp(sTopol.shape,sMap.topol.shape) | ...
```

```
    any(sTopol.msize ~= sMap.topol.msize),
    warning('Changing the original map topology.');
  end
end
sMap.topol = sTopol;

% complement the training struct
sTrain = som_train_struct(sTrain,sMap,'dlen',dlen);
if isempty(sTrain.mask), sTrain.mask = ones(dim,1); end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% initialize

M        = sMap.codebook;
mask     = sTrain.mask;
trainlen = sTrain.trainlen;

% neighborhood radius
if trainlen==1,
  radius = sTrain.radius_ini;
elseif length(radius)<=2,
  r0 = sTrain.radius_ini; r1 = sTrain.radius_fin;
  radius = r1 + fliplr((0:(trainlen-1))/(trainlen-1)) * (r0 - r1);
else
  % nil
end

Ud = som_unit_dists(sTopol);
Ud = Ud.^2;
radius = radius.^2;
% zero neighborhood radius may cause div-by-zero error
radius(find(radius==0)) = eps;
Known = ~isnan(D);
W1 = (mask*ones(1,dlen)) .* Known';
D(find(~Known)) = 0;

% constant matrices
WD = 2*diag(mask)*D';    % constant matrix
dconst = ((D.^2)*mask)'; % constant in distance calculation for each data
sample
                         % W2 = ones(munits,1)*mask'; D2 = (D'.^2);

% initialize tracking
start = clock;
qe = zeros(trainlen,1);

blen = min(munits,dlen);

% reserve some space
bmus = zeros(1,dlen);
ddists = zeros(1,dlen);

for t = 1:trainlen,
  i0 = 0;
  while i0+1<=dlen,
    inds = [(i0+1):min(dlen,i0+blen)]; i0 = i0+blen;
    Dist = (M.^2)*W1(:,inds) - M*WD(:,inds);
```

```
    [ddists(inds), bmus(inds)] = min(Dist);
  end

  % tracking
  if tracking > 0,
    ddists = ddists+dconst; % add the constant term
    ddists(ddists<0) = 0;   % rounding errors...
    qe(t) = mean(sqrt(ddists));
    trackplot(M,D,tracking,start,t,qe);
  end

  switch sTrain.neigh,
   case 'bubble',   H = (Ud<=radius(t));
   case 'gaussian', H = exp(-Ud/(2*radius(t)));
   case 'cutgauss', H = exp(-Ud/(2*radius(t))) .* (Ud<=radius(t));
   case 'ep',       H = (1-Ud/radius(t)) .* (Ud<=radius(t));
  end


  P = sparse(bmus,[1:dlen],weights,munits,dlen);


  S = H*(P*D);
  A = H*(P*Known);


  % only update units for which the "activation" is nonzero
  nonzero = find(A > 0);
  M(nonzero) = S(nonzero) ./ A(nonzero);

end; % for t = 1:trainlen

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Build / clean up the return arguments

% tracking
if tracking > 0, fprintf(1,'\n'); end

% update structures
sTrain = som_set(sTrain,'time',datestr(now,0));
if struct_mode,
  sMap = som_set(sMap,'codebook',M,'mask',sTrain.mask,'neigh',sTrain.neigh);
  tl = length(sMap.trainhist);
  sMap.trainhist(tl+1) = sTrain;
else
  sMap = reshape(M,orig_size);
end

return;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% subfunctions

%%%%%%%%
function [] = trackplot(M,D,tracking,start,n,qe)
```

```
  l = length(qe);
  elap_t = etime(clock,start);
  tot_t = elap_t*l/n;
  fprintf(1,'\rTraining: %3.0f/ %3.0f s',elap_t,tot_t)
  switch tracking
   case 1,
   case 2,
    plot(1:n,qe(1:n),(n+1):l,qe((n+1):l))
    title('Quantization error after each epoch');
    drawnow
   otherwise,
    subplot(2,1,1), plot(1:n,qe(1:n),(n+1):l,qe((n+1):l))
    title('Quantization error after each epoch');
    subplot(2,1,2), plot(M(:,1),M(:,2),'ro',D(:,1),D(:,2),'b+');
    title('First two components of map units (o) and data vectors (+)');
    drawnow
  end
  % end of trackplot

function h=som_show(sMap, varargin)

error(nargchk(1,Inf,nargin))     % check no. of input args

if isstruct(sMap),               % check map
  [tmp,ok,tmp]=som_set(sMap);
  if all(ok) & strcmp(sMap.type,'som_map')
    ;
  else
    error('Map struct is invalid!');
  end
else
  error('Requires a map struct!')
end

munits=size(sMap.codebook,1); % numb. of map units
d=size(sMap.codebook,2);      % numb. of components
msize=sMap.topol.msize;       % size of the map
lattice=sMap.topol.lattice;   % lattice

if length(msize)>2
  error('This visualizes only 2D maps!')
end

if rem(length(varargin),2)
  error('Mismatch in identifier-value  pairs.');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  read in optional arguments

if isempty(varargin),
  varargin = { 'umat','all','comp','all'};
end


try
  [Plane, General]= check_varargin(varargin, munits, d, sMap.name);
```

```
catch
  error(lasterr);
end

% Set default values for missing ones

% No planes at all (only general properties given in varargin):
% set default visualization

if isempty(Plane)
  varargin = [varargin, { 'umat','all','comp','all'}];
  % and again we go...
  try
    [Plane, General]= check_varargin(varargin, munits, d, sMap.name);
  catch
    error(lasterr);
  end
end

% set defaults for general properties

if isempty(General.colorbardir)
  General.colorbardir='vert';
end

if isempty(General.scale)
  General.scale='denormalized';
end

if isempty(General.size)
  General.size=1;
end

if isempty(General.edgecolor)
  General.edgecolor='none';
end

n=length(Plane);                      % the number of subfigures
% get the unique component indices
c=General.comp(General.comp>0);
c=setdiff(unique(c),[0 -1]);
c=c(~isnan(c));
% estimate the suitable dimension for
if isempty(General.subplots),
  y=ceil(sqrt(n));                    % subplots
  x=ceil(n/y);
else
  y = General.subplots(2);
  x = General.subplots(1);
  if y*x<n,
    error(['Given subplots grid size is too small: should be >=' num2str(n)]);
  end
end

clf;                                % clear figure
for i=1:n,                          % main loop
  h_axes(i,1)=subplot(x,y,i);       % open a new subplot
```

```
  % Main switch: select function according to the flags set in comps
  switch Plane{i}.mode
  case 'comp'
    %%% Component plane
    tmp_h=som_cplane(lattice,msize, sMap.codebook(:,General.comp(i)), ...
             General.size);
    set(tmp_h,'EdgeColor', General.edgecolor);
    set(h_axes(i),'Tag','Cplane');
    h_label(i,1)=xlabel(sMap.comp_names{General.comp(i)});
  case 'compi'
    %%% Component plane (interpolated shading)

    tmp_h=som_grid(lattice, msize, 'surf', sMap.codebook(:,Plane{i}.value),
...
    'Marker', 'none', 'Line', 'none');
    set(h_axes(i),'Tag','CplaneI');
    h_label(i,1)=xlabel(sMap.comp_names(Plane{i}.value));
    vis_PlaneAxisProperties(gca,lattice,msize,NaN);

  case 'color'
    %%% Color plane

    tmp_h=som_cplane(lattice,msize,Plane{i}.value,General.size);
    set(tmp_h,'EdgeColor','none');
    set(h_axes(i),'Tag','Cplane');
    h_label(i,1)=xlabel(Plane{i}.name);
  case 'colori'
    %%% Color plane (interpolated shading)
    tmp_h=som_grid(lattice, msize, 'surf', Plane{i}.value, 'Marker', 'none',
...
    'Line', 'none');
    set(h_axes(i),'Tag','CplaneI');
    h_label(i,1)=xlabel(Plane{i}.name);
    vis_PlaneAxisProperties(gca,lattice,msize,NaN);

  case 'empty'
    %%% Empty plane

    tmp_h=som_cplane(lattice,msize,'none');
    h_label(i,1)=xlabel(Plane{i}.name);
    set(h_axes(i),'Tag','Cplane');

  case 'umat'
    %%% Umatrix

    u=som_umat(sMap.codebook(:,Plane{i}.value),sMap.topol,'median',...
    'mask',sMap.mask(Plane{i}.value)); u=u(:);
    tmp_h=som_cplane([lattice 'U'],msize,u);
    set(tmp_h,'EdgeColor','none');
    set(h_axes(i),'Tag','Uplane');
    h_label(i,1)=xlabel(Plane{i}.name);

  case 'umati'
    %%% Umatrix (interpolated shading)

    u=som_umat(sMap.codebook(:,Plane{i}.value),sMap.topol,'mean',...
    'mask',sMap.mask(Plane{i}.value)); u=u(1:2:end,1:2:end);
```

```
    u=u(:);
    tmp_h=som_grid('rect', msize, 'surf', u, ...
    'Marker', 'none', 'Line', 'none', ...
    'coord', som_vis_coords(lattice,msize));
    set(h_axes(i),'Tag','UplaneI');
    h_label(i,1)=xlabel(Plane{i}.name);
    vis_PlaneAxisProperties(gca,lattice,msize,NaN);

  otherwise
    error('INTERNAL ERROR: unknown visualization mode.');
  end
  set(h_label,'Visible','on','verticalalignment','top');
  set(gca,'plotboxaspectratio',[msize(2) msize(1) msize(1)]);
  if General.comp(i)> -1 & ~strcmp(General.colorbardir,'none'),
    h_colorbar(i,1)=colorbar(General.colorbardir);           % colorbars
  else
    h_colorbar(i,1)=-1;
    General.comp(i)=-1;
  end
end         %% main loop ends

% Set window name

set(gcf,'Name',[ 'Map name: ' sMap.name]);

SOM_SHOW.subplotorder=h_axes;
SOM_SHOW.msize=msize;
SOM_SHOW.lattice=lattice;
SOM_SHOW.dim=d;
SOM_SHOW.comps=General.comp;
SOM_SHOW.comp_norm=sMap.comp_norm; %(General.comp(find(General.comp>0)));
set(gcf,'UserData', SOM_SHOW);
set(h_label,'interpreter','none');
h_colorbar=som_recolorbar('all', 3, General.scale);   %refresh colorbars
vis_footnote(General.footnote);  vis_footnote(12);
colormap(General.colormap);
if nargout > 0
  h.plane=h_axes; h.colorbar=h_colorbar; h.label=h_label;
end


%%%%%% SUBFUNCTIONS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [Plane, General]=check_varargin(args, munits, dim, name)

% args: varargin of the main function
% munits: number of map units
% dim: map codebook dimension
% name: map name
% Define some variables (they must exist later)

Plane={};           % stores the visualization data for each subplot
General.comp=[];    % information stored on SOM_SHOW figure (which component)
General.size=[];            % unit size
General.scale=[];           % normalization
General.colorbardir=[];     % colorbar direction
General.edgecolor=[];       % edge colors
```

```
General.footnote=name;      % footnote text
General.colormap=colormap;  % default colormap (used to be gray(64).^.5;)
General.subplots=[];        % number of subplots in y- and x-directions

for i=1:2:length(args),
  %% Check that all argument types are strings

  if ~ischar(args{i}),
    error('Invalid input identifier names or input argument order.');
  end

  %% Lower/uppercase in identifier types doesn't matter:

  identifier=lower(args{i});     % identifier (lowercase)
  value=args{i+1};


  switch identifier
   case {'comp','compi'}
    %%% Component planes: check values & set defaults

    if ~vis_valuetype(value,{'nx1','1xn','string'}) & ~isempty(value),
      error([ 'A vector argument or string ''all'' expected for ''' ...
          identifier '''.'])
    end
    if isempty(value)
      value=1:dim;
    elseif ischar(value),
      if ~strcmp(value,'all')
    error([ 'Only string value ''all'' is valid for ''' ...
        identifier '''.']);
      else
    value=1:dim;
      end
    else
      value=round(value);
      if min(value)<1 | max(value)>dim,
    error([ 'Component indices out of range in ''' identifier '''.'])
      end
    end
    if size(value,1)==1, value=value';end
    comp_=value;
    name=[]; % name is taken form sMap by index in main loop

  case {'umat','umati'}
    %%% Check first the possible cell input

    if iscell(value),
      if ndims(value) ~= 2 | any(size(value) ~= [1 2]) | ...
        ~vis_valuetype(value{2},{'string'}),
    error('Cell input for ''umat'' has to be of form {vector, string}.');
      else
    name=value{2}; value=value{1};
      end
    else
      name='U-matrix'; % no cell: default title is set
    end
```

```
    if ~vis_valuetype(value,{'nx1','1xn','string'}) & ~isempty(value),
      error('Vector, string ''all'', or cell {vector, string} expected for
''umat''.')
    end
    if isempty(value)
      value=1:dim;
    elseif ischar(value),
      if ~strcmp(value,'all')
    error('Only string value ''all'' is valid for ''umat''.')
      else
    value=1:dim;
      end
    else
      value=unique(round(value));
    end
    if min(value)<1 | max(value)>dim,
      error('Component indices out of range in ''umat''.')
    end

    if size(value,1)==1, value=value';end
    comp_=0;

  case 'empty'
    %%% Empty plane: check values & set defaults

    if ~vis_valuetype(value,{'string'}),
      error('A string value for title name expected for ''empty''.');
    end
    name=value;
    comp_=-1;

  case { 'color','colori'}
    %%% Color plane: check values & set defaults

    % Check first the possible cell input
    if iscell(value),
      if ndims(value)~=2 | any(size(value) ~= [1 2]) | ...
        ~vis_valuetype(value{2},{'string'}),
    error([ 'Cell input for ''' identifier ...
          ''' has to be of form {M, string}.']);
      else
    name=value{2}; value=value{1};
      end
    else
      name='Color code'; % no cell: default title is set
    end
    if size(value,1)~=munits | ...
      (~vis_valuetype(value,{'nx3rgb'}) & ...
       ~vis_valuetype(value,{'nx1'}) & ...
       ~vis_valuetype(value,{'nx1xm'}) & ...
       ~vis_valuetype(value,{'nx3xdimrgb'})),
      error(['Mx3 or Mx3xN RGBmatrix, Mx1 or Mx1xN matrix, cell '...
         '{RGBmatrix, string},' ...
         ' or {matrix, string} expected for ''' identifier '''.']);
    end

    % if colormap is fixed, we don't draw colorbar (comp_ flag is -1)
```

```
    % if colormap is indexed, we draw colorbar as in umat (comp_=0)

    if size(value,2)==3
      comp_=-1;
    else
      comp_=0;
    end

    %%%% The next things are general properties of the visualization---
    %%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  case 'size'
    %%% Unit size: check & set

    if ~vis_valuetype(value,{'1x1',[munits 1]})
      error('A munits x 1 vector or a scalar expected for ''size''.')
    end
    if isempty(value),
      General.size=1;
    else
      General.size=value;
    end

   case 'bar'
    %%% Colorbar existence & direction: check & set

    if ~vis_valuetype(value,{'string'})
      error('String value expected for ''bar''.')
    elseif isempty(value)
      value='vert';
    end
    if any(strcmp(value,{'vert','horiz','none'})),
      General.colorbardir=value;
    else
      error('String ''vert'', ''horiz'' or ''none'' expected for ''bar''.');
    end

  case 'norm'
    %%% Value normalization: check & set

    if ~vis_valuetype(value,{'string'})
      error('String ''n'' or ''d'' expected for ''norm''.');
    elseif isempty(value)
      value='n';
    end
    if strcmp(value(1),'n'),
      General.scale='normalized';
    elseif strcmp(value(1),'d'),
      General.scale='denormalized';
    else
      error('String ''n(ormalized)'' or ''d(enormalized)'' expected for
''norm''.');
    end

  case 'edge'
    %%% Edge on or off : check % set
```

```
  if ~vis_valuetype(value,{'string'}) & ~isempty(value),
    error('String value expected for ''edge''.')
  elseif ~isempty(value),
    switch value
    case 'on'
  General.edgecolor='k';
    case 'off'
  General.edgecolor='none';
    otherwise
  error('String value ''on'' or ''off'' expected for ''edge''.')
    end
  end

case 'footnote'
  %%% Set the movable footnote text

  if ~vis_valuetype(value,{'string'})
    if ~isempty(value),
  error('String value expected for ''footnote''.');
    else
  General.footnote=sMap.name;
    end
  else
    General.footnote=value;
  end

 case 'colormap'
  %%% Set the colormap
  if isempty(value)
    General.colormap=gray(64).^2;
  elseif ~vis_valuetype(value,{'nx3rgb'})
    error('Colormap is invalid!');
  else
    General.colormap=value;
  end

 case 'subplots'
  %%% set the number of subplots
  if ~vis_valuetype(value,{'1x2'}) & ~vis_valuetype(value,{'2x1'})
    error('Subplots grid size is invalid!');
  else
    General.subplots=value;
  end

otherwise
  %%% Unknown identifier

  error(['Invalid argument identifier ''' identifier '''!']);
end

%%% Set new entry to the Plane array if the indentifier means
%%% making a new plane/planes

tail=length(Plane);
switch identifier
case {'comp','compi'}
```

```
    for i=1:length(value)
      Plane{tail+i}.mode=identifier;
      Plane{tail+i}.value=value(i);
      Plane{tail+i}.name=name; % not used actually
    end
    General.comp = [General.comp; comp_];
   case {'umat','umati','empty'}
    Plane{tail+1}.mode=identifier;
    Plane{tail+1}.value=value;
    Plane{tail+1}.name=name;
    General.comp = [General.comp; comp_];
   case {'color','colori'},
    for i=1:size(value,3),
      Plane{tail+i}.mode=identifier;
      Plane{tail+i}.name=[name '_' num2str(i)];
      Plane{tail+i}.value=value(:,:,i);
      General.comp = [General.comp; comp_];
    end
    if size(value,3)==1,
      Plane{tail+1}.name=name;
    end
  otherwise
    ; % do nothing
  end
end
```

```
function sM  = som_supervised(sData,varargin)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

D0 = sData.data;
[c,n,classlabels] = class2num(sData.labels(:,1));

%%%%%%%% Checking arguments %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if ~isstruct(sData)
  error('Argument ''sData'' must be a ''som_data'' -struct.');
else
  data_name = sData.name;
  comp_names = sData.comp_names;
  comp_norm = sData.comp_norm;
end

[dlen,dim] = size(sData.data);

% defaults

mapsize = '';
sM = som_map_struct(dim+n);
sTopol = sM.topol;
munits = prod(sTopol.msize); % should be zero
mask = sM.mask;
name = sM.name;
neigh = sM.neigh;
tracking = 1;
algorithm = 'batch';

%%%% changes to defaults (checking varargin) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

i=1;
while i <= length(varargin)
  argok = 1;
  if ischar(varargin{i})
    switch varargin{i},
      % argument IDs
     case 'mask',
      i=i+1;
      mask = varargin{i};
     case 'munits',
      i=i+1;
      munits = varargin{i};
     case 'msize',
      i=i+1;
      sTopol.msize = varargin{i};
      munits = prod(sTopol.msize);
     case 'mapsize',
      i=i+1;
      mapsize = varargin{i};
     case 'name',
      i=i+1;
      name = varargin{i};
     case 'comp_names',
```

```
      i=i+1;
      comp_names = varargin{i};
    case 'lattice',
      i=i+1;
      sTopol.lattice = varargin{i};
    case 'shape',
      i=i+1;
      sTopol.shape = varargin{i};
    case {'topol','som_topol','sTopol'},
      i=i+1;
      sTopol = varargin{i};
      munits = prod(sTopol.msize);
    case 'neigh',
      i=i+1;
      neigh = varargin{i};
    case 'tracking',
      i=i+1;
      tracking = varargin{i};
    case 'algorithm',
      i=i+1;
      algorithm = varargin{i};
  % unambiguous values
    case {'hexa','rect'},
      sTopol.lattice = varargin{i};
    case {'sheet','cyl','toroid'},
      sTopol.shape = varargin{i};
    case {'gaussian','cutgauss','ep','bubble'},
      neigh = varargin{i};
    case {'seq','batch'},
      algorithm = varargin{i};
    case {'small','normal','big'},
      mapsize = varargin{i};
    otherwise argok=0;
    end
  elseif isstruct(varargin{i}) & isfield(varargin{i},'type'),
    switch varargin{i}(1).type,
      case 'som_topol',
       sTopol = varargin{i};
    otherwise argok=0;
    end
  else
    argok = 0;
  end
  if ~argok,
    disp(['(som_supervised) Ignoring invalid argument #' num2str(i+1)]);
  end
  i = i+1;
end
[dlen,dim] = size(D0);

Dc = zeros(dlen,n);

for i=1:dlen
  if c(i)
    Dc(i,c(i)) = 1;
  end
end
```

```
D = [D0, Dc];

% initialization and training

sD = som_data_struct(D,...
                     'name',data_name);

sM = som_make(sD,...
              'mask',mask,...
              'munits',munits,...
              'name',data_name,...
              'tracking',tracking,...
              'algorithm',algorithm,...
              'mapsize',mapsize,...
              'topol',sTopol,...
              'neigh',neigh);

% add labels

for i=1:prod(sM.topol.msize),
  [dummy,class] = max(sM.codebook(i,dim+[1:n]));
  sM.labels{i} = classlabels{class};
end

%sD.labels = sData.labels;
%sM = som_autolabel(sM,sD,'vote');

% remove extra components and modify map -struct

sM.codebook = sM.codebook(:,1:dim);
sM.mask = sM.mask(1:dim);
sM.comp_names = sData.comp_names;
sM.comp_norm = sData.comp_norm;

% remove extras from sM.trainhist

for i=1:length(sM.trainhist)
  if sM.trainhist(i).mask
    sM.trainhist(i).mask = sM.trainhist(i).mask(1:dim);
  end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [numbers, n, names] = class2num(class)
 names = {};
numbers = zeros(length(class),1);

for i=1:length(class)
  if ~isempty(class{i}) & ~any(strcmp(class{i},names))
    names=cat(1,names,class(i));
  end
end

n=length(names);
```

```
tmp_numbers = (1:n)';

for i=1:length(class)
  if ~isempty(class{i})
    numbers(i,1) = find(strcmp(class{i},names));
  end
end
```