N=° d'ordre : 34/2012-M/EL

République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifiques

Université des sciences et de la technologie Houari Boumediene

Faculté d'ELECTRONIQUE ET D'INFORMATIQUE

MEMOIRE

Présenté pour l'obtention du diplôme de MAGISTER
En : ELECTRONIQUE
Spécialité : **Communication Parlée**
Par

Mohamed LICHOURI

THÈME

Vers un Système de Compréhension d'énoncé en Communication Homme-Machine

Soutenu publiquement le 15/04/2012 devant le jury composé de :

Mr A.GUESSOUM	Professeur	à l'U.BLIDA	Président
Mr A.DJERADI	Professeur	à l'USTHB	Directeur de mémoire
Mr S.LARABI	Professeur	à l'USTHB	Examinateur
Mr H.TEFFAHI	Maitre de Conférence/A	à l'USTHB	Examinateur
Mme L.FALEK	Maitre de Conférence /A	à l'USTHB	Examinateur

Remerciements

Mes remerciements vont tout d'abord à Mr Amar DJERADI, PROFESSEUR à la Faculté d'Electronique et d'Informatique de l'USTHB, qui a suivi et dirigé d'une façon continue mes travaux de recherche, pour la confiance qu'il m'a témoignée, pour la patience et la gentillesse qu'il a manifestées à mon égard, pour son encouragement et son soutien moral. Je lui exprime ma profonde gratitude pour son effort à organiser la soutenance de ma mémoire avec un Jury d'expert.

Je tiens à remercier Mr GUESSOUM, Professeur à la Faculté d'Electronique, U.BLIDA, pour l'honneur qu'il m'a fait de présider ce Jury.

Je voudrai également exprimer mes remerciements à Mr Slimane LARABI, Professeur à la Faculté d'Electronique et d'Informatique, USTHB, Mr Houcine TEFFAHI, Maître de Conférences à la Faculté d'Electronique et d'Informatique, USTHB et Mme Leila FALEK, Maître de Conférences à la Faculté d'Electronique et d'Informatique, USTHB pour avoir bien voulu accepter de faire partie de ce Jury.

Mes remerciements vont aussi à Mlle Fatmazohra CHELALI, chercheuse au Laboratoire de Communication Parlée et Traitement du Signal à la Faculté d'Electronique et d'Informatique pour m'avoir bien encourager le long de mon travail.

Je ne saurais oublier ma famille qui m'a été d'un grand soutien et qui n'a ménagé ni sa patience ni ses encouragements pour que ce travail puisse un jour aboutir.

Je tiens à exprimer toute ma gratitude à Wahib DJAOUTI, Chakib BERZALI, Rabie SBAA et tous ceux qui ont contribué, de près ou de loin, à la concrétisation de ce travail

Dédicace

Je dédie tout particulièrement ce modeste travail à mes parents, à

m*es frères et sœurs, et toute ma famille.*

A mes amis, et à tous mes collègues.

Résumé

Le travail présenté dans ce mémoire vise à réaliser un système de compréhension Automatique de l'énoncé spécifique à un domaine, en premier lieu. C'est pourquoi, nous nous intéressons au Système de Compréhension humaine. Ce dernier, nous a permis de concevoir deux approches de Compréhension : Conceptuelle (ou Déterministe) et Thématique (ou Probabiliste).

En deuxième lieu, on vise la mise au point de l'architecture de base d'un Système Générale de Compréhension d'énoncé. Dans ce sens, nous avons considéré deux applications distinctes : (i) Une application d'interrogation d'une base de données de Scolarité, (ii) Application d'Extraction d'Information (Diagnostique Médical).

Mots-clés: Compréhension de l'énoncé, Conceptuelle, Thématique, Déterministe, Probabiliste, Scolarité, Médical.

TABLE DES MATIERES

LISTE DES FIGURES

Chapitre 1

Chapitre 2

Chapitre 3

Chapitre 4

LISTE DES TABLEAUS

Chapitre 1

Chapitre 2

Chapitre 3

Chapitre 4

Liste des abréviations

API	Application Programming Interface
ARISE	Automatic Railway Information Systems for Europe
AT&T	American Telephone & Telegraph
ATIS	Automatic Terminal Information Service
CACAO	Compréhension Automatique par segments Conceptuels Assistée par Ordinateur
CHRONUS	Conceptual Hidden Representation of Natural Unconstrained Speech
DTMF	Dual-Tone Multi-Frequency
EM	Maximum Expectation
ES	Expert System
GUI	Graphical User Interface
HMM	Hidden Markov Model
LIMSI	Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur
LSA	Latent Semantic Analysis
LSI	Latent Semantic Indexing
MADHS	Multi-Agent Diagnosis Helping System
MASK	Multimedia Automated Service Kiosk
MDSS	Medical Decision-Support Systems
MIT	Massachusetts Institute of Technology
NLTK	Natural Language Toolkit
PABX	Private Automatic Branch eXchange
PLSA	Probabilistic Latent Semantic Analysis
SC	Segment Conceptuelle
SEDMD	Systeme Expert de Dignostique Médicale à Distance
SGBD	Système de Gestion de Base de Données
SQL	Structured Query Language
SVD	Singular Value Decomposition
SVI	Systèmes Vocale Interactifs
SyCE	Systéme de Compréhension d'Énoncé
SyCE-CHM	Systéme de Compréhension d'Énoncé en Communication Homme-Machine
SyCE-G	Systéme de Compréhension d'Énoncé Généralisé
TALN	Traitement Automatique des Langues Naturelles

TF-IDF	Term Frequency-Inverse Document Frequency
TINA	Telecommunication Information Networking Architecture

Introduction générale

Introduction Générale

Contexte Générale

Un système d'interaction vocal permet à un utilisateur humain de coopérer avec la machine pour mener à terme une tache en utilisant un dialogue en langage naturel. Dans les visions futures de la technologie de l'interaction, la machine parlante est vue comme omnisciente[1], très articulée, et souvent anthropomorphes[2], avec la possibilité de fournir tous types d'informations utiles, de réagir à un large éventail de situations et de problèmes, et de reconnaître des gestes et des émotions. Cependant, la réalité actuelle des systèmes de dialogue est nettement inférieure à celle de leurs homologues dans la science-fiction. Tous de même, dans le commerce les systèmes disponibles sont en mesure d'automatiser une variété de services à la clientèle, telles que fourniture d'information de vol, les prévisions météo, résultats sportifs et les cours des actions, et ils peuvent également supporter les transactions telles que la réservation des hôtels, location de voitures, des paiements, ou le téléchargement de sonneries pour les téléphones mobiles. Ces systèmes sans opérateurs humains sont des tâches banales qui sont de nature répétitive et peuvent donc être facilement automatisées, et pour lesquels la parole est aussi un mode de communication naturel. Toutefois, les laboratoires de recherche à travers le monde sont continuellement entraînent de repousser les frontières en développant des systèmes avec plus de fonctionnalités avancées. Ces systèmes ont une interface plus conversationnelle et ils accomplissent des tâches plus difficiles telles que la planification de l'évacuation d'une zone de catastrophe. D'autres recherches consistent à rendre les systèmes plus attentifs aux besoins de l'homme ainsi que de montrer les caractéristiques anthropomorphiques dans leurs réponses, par exemple, en permettant aux systèmes de reconnaître les états émotionnels de leurs utilisateurs et par l'affichage de même des émotions appropriées dans le cadre d'un dialogue [1] ou encore **« *rivaliser avec la capacité de l'esprit humain à déterm*iner des réponses précises à des questions en langage naturel. »**[2].

Les systèmes d'interaction vocale, en revanche, bien qu'ils intègrent les technologies de base de la reconnaissance automatique de la parole et de la synthèse vocale, exigent également des façons de comprendre la demande de l'utilisateur et génèrent ensuite les actions ou les réponses appropriées. Ces fonctionnalités sont la responsabilité du module de Compréhension d'Énoncé.

---

[1] Omniscient: Qui sait tout.
[2] Anthropomorphe: (Humanoïde) Qui a la forme, l'apparence humaine

La compréhension de la parole est un sujet pluridisciplinaire qui fait intervenir des aspects informatiques, linguistiques, cognitifs mais aussi de traitement de signal et d'intelligence artificielle. Les applications visées par ce sujet sont aussi diverses que variées. Les plus connues sont celles liées à l'interrogation des bases de données, celles utilisées dans le cadre du pilotage de machines et l'exploration de données.
Le but ultime est de simuler la communication humaine et de réaliser ainsi des systèmes d'interaction multimodale et spontanée. Dans de tels systèmes, la composante de Compréhension d'Enoncé joue le rôle le plus prometteur.

Les travaux présentés dans cette thèse se situent dans le cadre de la Compréhension d'Énoncé en Communication Homme-Machine. Les applications traitées concernent l'interrogation vocale de bases de données et l'exploration de données où le dialogue est finalisé et propre à un domaine donné. Cela signifie que les systèmes visés sont spécifiques à une application donnée et que le vocabulaire utilisé est limité au domaine de cette application.

Dans notre travail, nous nous intéressons plus particulièrement à l'approche statistique pour la CE (entrée de type texte). Cette approche a été utilisée dans de nombreux autres travaux où elle a fait preuve de simplicité et d'efficacité [3], [4], [5], [6]. Elle présente l'avantage de réduire le recours à l'expertise humaine pour établir les règles d'inférence. Dans cette approche, le problème de compréhension est divisé en deux parties : la traduction sémantique et la génération de réponses. La compréhension proprement dite est réalisée au cours de la première étape où on associe à chaque phrase un ensemble de concepts qui permettent de la traduire sémantiquement [6].

Problématique

En effet, notre premier problème est de réaliser un Système de Compréhension d'Énoncé en CHM fonctionnel. Ceci dit, avoir N applications dans M langue nous ramènera à réaliser M*N applications du fait que chaque application est propre à un domaine et une langue donnée. Mais cela est irréalisable d'un point de vue de ressource humain, temporel et financière.

Donc, notre problème majeur est la réalisation d'un Système Générale de Compréhension d'Énoncé qui sera adaptable à n'importe quelle domaine (en infligeant des modifications aux seins des bases de données seulement) et langages (ce dernier s'y fut voire de plus en plus de l'importance par les systèmes de reconnaissance de la parole multilingues ou la détection de la langue, mais pas en compréhension).

Apports de la mémoire

Les travaux de cette thèse s'articulent autour de l'approche statistique pour le SyCE-CHM. En premier lieu, nous proposons l'utilisation d'un algorithme de segmentation de texte en phrase basé sur les travaux de (Kiss et al.,)[7]. Du fait qu'on ne considère que les phrases comme entrée de notre système.

En deuxième lieu, en se basant sur l'aspect cognitif humaine de découper la phrase en mots envers sa compréhension ; on va fait submerger une sous étapes qui est l'analyse lexicale servant en plus de segmenter les phrases en mots d'effectuer une première réduction de dimension en filtrant les termes non signifiants.

En troisième lieu, on va présenter une première version de notre système en adoptant la méthode d'extraction automatique de concept [6] suivant la même architecture mais en utilisant d'autres méthodes de représentation vectorielle servant à extraire le sens caché dans une phrase, en regroupant les mots les plus proches d'un point de vue sémantique en concept en utilisant un algorithme de classification non supervisé. Ces concepts seront utilisées pour étiqueter les phrases (traduction de phrase du langage naturel vers le langage conceptuel), ensuite seront traduit en requête SQL interprétable par un système de gestion de base de données. Cette dernière étape n'est utile que pour les systèmes d'interrogation de base de données (traduction en langage SQL), mais pas pour les systèmes d'exploration de données.

En dernier lieu, on va présenter une nouvelle architecture en se basant sur un autre aspect cognitif humain qui est la catégorisation des textes par thème. En intégrant de plus un autre classificateur supervisé servant à regrouper les phrases en catégorie ayant un thème commun.

Plan du mémoire

Le premier chapitre de ce mémoire est consacré à la présentation des systèmes d'interaction vocale en mettant l'accent sur le problème de la compréhension qui est la maille forte dans la chaine de ces systèmes. En second, un état de l'art des systèmes de Compréhension de la Parole est présenté en s'intéressant depuis les différentes approches existantes (stochastique, linguistique, connexionniste) à l'approche statistique. Enfin, nous passons en revue les principaux modèles de langage utilisés dans la littérature.

Le deuxième chapitre éclate en long et en large les différents composants du système SyCE-CHM, toute en présentant les différents algorithmes qui surgissent au niveau de chaque composant. En second, on va présenter l'approche qu'on va adopter pour notre système.

Dans le troisième chapitre, nous présentons l'environnement SyCE-CHM ainsi que les différents algorithmes utilisées par chaque composant. En second partie, on va submerger notre système dans un milieu réel qui est la réalisation de l'application de Scolarité en se basant totalement sur l'environnement SyCE-CHM. Pour avoir une vision plus complète des capacités de cet environnement, on va le soumettre à un autre domaine qui est celui de la Diagnostic Médicale. Á ce niveau-là, le SyCE-CHM actuel a atteint ces limites du fait qu'il n'est utile que pour les systèmes d'interrogation d'une base de donnée à un certain niveau seulement.

Le dernier chapitre est deviser en deux partie, une servant à améliorer les qualités du SyCE-CHM en incrustant des modifications internes, la second partie est consacrer pour la présenta-

tion d'une nouvelle architecture des SyCE-CHM qui est supposé être indépendante de l'application et du langage (Générale).

Nous terminons ce manuscrit par une conclusion et quelques perspectives concernant les différents travaux effectués.

En complément, cinq annexes sont ajoutées à ce mémoire. Les trois premiers consistent à des tutoriel numérique sur les algorithmes LSA, K-moyennes et le classificateur Bayésien Naïf, alors que les deux restants contiennent le code source complet des deux systèmes.

Chapitre I

Structure Générale d'un Système de Compréhension de la Parole

Chapitre I

Structure Générale d'un Système de Compréhension de la Parole

I.1. Introduction

Nous allons dans un premier temps présenter le contexte d'étude sur lequel nous nous focaliserons, à savoir la compréhension des énoncés oraux spontanés dans les Systèmes Vocal Interactifs SVI et plus généralement dans tous systèmes interactifs dont une des modalités d'entrée est la parole continue. Ces systèmes permettent le plus souvent de consulter une base de données et ils concernent tous une tâche finalisée, bien délimitée tel que les renseignements ferroviaires.

Après avoir fixé le cadre de notre étude, nous étudierons la problématique de la compréhension de la parole en Communication Homme-Machine et plus particulièrement celui de l'énoncé ainsi les difficultés auxquelles un module de compréhension robuste doit faire face.

Enfin, nous exposerons, dans la dernière partie de ce chapitre, un aperçu global de divers systèmes de compréhension que nous ordonnerons selon quatre axes de classification, puis nous présenterons quelques systèmes connus et dont le modèle est proche de celui que nous proposons.

I.2. Contexte d'étude

L'objet d'étude de ce mémoire concerne la modélisation, la conception et l'évaluation d'un Système de Compréhension d'Énoncé (entrée textuel) en Communication Homme-Machine. Nous allons présenter dans cette section les SVI en décrivant leurs domaines d'application et leur fonctionnement général.

I.2.1. Les Systèmes Vocaux Interactifs

I.2.1.1. Domaines d'application

Un Système Vocale Interactif (SVI) est un système qui permet à un ordinateur d'interagir avec les humains grâce à l'utilisation de la voix et les entrées clavier DTMF[1].

Dans les télécommunications, le SVI permet aux clients d'interagir avec la base de données d'une institution via un clavier de téléphone ou un microphone, après quoi ils peuvent servir leurs propres enquêtes en suivant un dialogue interactif avec la machine.

La technologie des SVI est également introduite dans les systèmes automobiles pour une utilisation mains-libres. Le déploiement actuel dans le domaine automobile tourne autour de la navigation par satellite, par onde radio ou par téléphone mobile.

[1] DTMF : Dual-Tone Multi-Frequency

De tels systèmes fonctionnent actuellement sur une tâche finalisée dans un domaine particulier. En effet, de nos jours nous ne savons pas réaliser des systèmes qui ne soient pas focalisés sur une ou quelques tâches données. C'est en effet les connaissances du domaine de la tâche qui permettent de faire face à toutes les difficultés rencontrées lors de la réalisation d'un dialogue. Ces connaissances permettent, entre autres, de contraindre les étapes de reconnaissance et de compréhension de l'énoncé prononcé par l'utilisateur. Un moyen de contourner cette difficulté est de construire un système général permettant de rediriger le dialogue vers un système traitant une tâche particulière. C'est ce qui a été réalisé par Seneff [8] et Carobus [9].

Les prototypes de SVI sont de plus en plus nombreux en laboratoire et commencent petit à petit à être opérationnels dans la vie de tous les jours. Ils peuvent être utilisés par téléphone, à l'aide de bornes vocales ou encore via internet. On voit ainsi apparaître des systèmes opérationnels qui permettent un dialogue interactif avec l'utilisateur via le téléphone. Ces systèmes concernent des applications aussi diverses que la réservation ferroviaire par téléphone (Mise en service par les chemins de fer néerlandais suite au projet européen ARISE [10]) ou encore le routage téléphonique grand public (mis en place en 1992 par AT&T [11]).

I.2.1.2. Architecture générale d'un SVI

Le principe d'un système vocal interactif est assez simple : l'utilisateur formule une requête, celle-ci doit être reconnue par le module de reconnaissance automatique de la parole et comprise par le module de compréhension, ensuite le contrôleur de dialogue détermine s'il doit demander des précisions sur la requête ou s'il doit consulter la base de données pour fournir une réponse, le système génère alors en retour le message de réponse et la synthétise.

Un système vocal est composé de cinq principaux éléments correspondant aux cinq grandes étapes du système que nous venons de citer. La Figure I.1 décrit l'architecture représentative de la majorité des systèmes vocaux [12].

- ✓ **Reconnaissance de la parole** – Ce module a pour but de transcrire le signal vocal énoncé par l'usager en un message orthographique. De plus en plus, les systèmes de reconnaissance automatique de la parole proposent plusieurs solutions candidates classées selon leur score de vraisemblance. Ces solutions peuvent être données sous la forme d'un graphe de mots (appelé aussi treillis de mots) ou de plusieurs suites de mots.

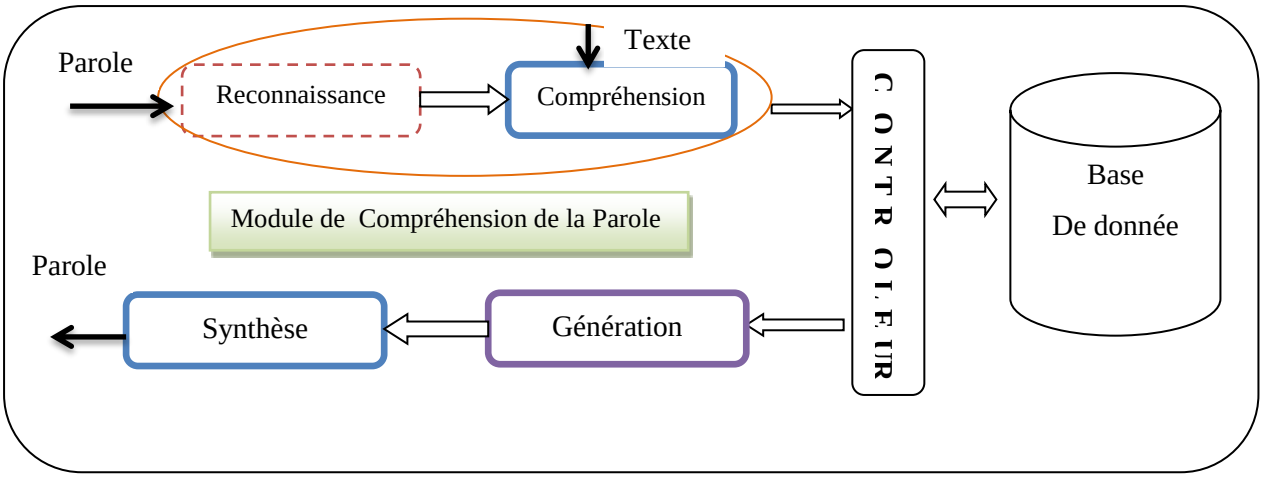


Figure I.1 : Architecture générale d'un système vocale

✓ **Un module de Compréhension –** par abus de langage, la majorité des travaux concernant les systèmes interactifs utilisant comme modalité la parole ont tendance à confondre entre le Système de Compréhension de la Parole[2] et le module de Compréhension. Or comme montré dans la Figure 1.1.-le second n'étant qu'un simple constituant du premier, plus celui de la reconnaissance. Alors, dans ce qui suit ce qui nous intéresse est le module de compréhension, qui produit une représentation sémantique à partir des phrases transcrites par le module de reconnaissance [13].

✓ **Contrôleur de Dialogue --** Il a pour fonction de consulter la base de données pour fournir les informations demandées ou il décide s'il faut demander des précisions à l'usager. Il doit aussi choisir la meilleure stratégie de dialogue à mettre en œuvre pour simplifier la tâche de l'usager et lui permettre de réaliser son but le plus facilement et rapidement possible. Deux taches sont nécessaires pour assurer le fonctionnement d'un tel module, le **gestionnaire de buts de dialogue** et celui **de la tâche** [14].

✓ **Génération de message –** Le processus de génération est effectué généralement en deux étapes : la génération profonde (« Quoi dire ? ») et la génération de surface (« Comment le dire ? »). La génération profonde détermine le contenu du message et la façon dont il sera structuré, ensuite la génération de surface permet de choisir les mots qui composeront le message, faire des transformations linguistiques (anaphores, ellipses, coordinations…) si nécessaire. Comme nous l'avons mentionnés plus haut, l'étape de génération profonde est généralement incluse dans le contrôleur de dialogue [15] [16].

---

[2] Definition:

Spoken language understanding involves two primary component technologies :
Speech Recognition (SR), and Natural Language (NL) understanding.
La Compréhension du Langage Parlé comporte deux modules essentiels:
Reconnaissance de la Parole, et Compréhension du langage naturel (énoncé).

✓ **Synthèse vocale** – Une synthèse vocale permet de créer de la parole artificielle à partir d'un texte. La synthèse vocale transforme le texte en code utilisable par une voix. Il existe à la fois des voix payantes et gratuites, ces dernières sont souvent de moins bonne qualité mais elles sont intéressantes à connaître pour tester l'utilité d'une synthèse vocale ou pour une utilisation occasionnelle. Les voix sont identifiées par des prénoms. La qualité d'une voix dépend de différents critères comme l'intonation (affirmation, interrogation), l'accent, le rythme (vitesse), l'intensité (volume), l'articulation et les parasites. Une voix monocorde (peu d'intonation) sera perçu robotique. Il existe également un caractère individuel subjectif qui influence le choix d'une voix [17].

L'architecture des systèmes vocaux interactifs décrite ici est une représentation simplifiée de la majorité des systèmes. Cependant certains systèmes diffèrent légèrement de cette architecture : par exemple, l'étape de compréhension est parfois réalisée en même temps que l'étape de reconnaissance [18].

I.2.2. Le problème de la compréhension de la parole

La compréhension de la parole constitue une étape clé au sein d'un processus de communication humaine. Dans les dictionnaires, communiquer signifie "transmettre" ou alors faire savoir et faire connaître. Il s'agit de transmettre ou de faire savoir quelque chose, une idée, une pensée ou un état. Communiquer nécessite donc deux acteurs, un émetteur et un récepteur. Le récepteur est celui qui va recevoir ce qu'on veut transmettre et c'est celui sans doute qui a la tâche la plus difficile dans le processus de communication car il s'agit de comprendre ce que l'émetteur veut lui faire savoir. Goethe a dit "Parler est un besoin, écouter est un art." [19].

La parole constitue le moyen de communication le plus simple et le plus utilisé par l'Homme.

Ce mode de communication met en action un certain nombre de facultés humaines qu'il serait très difficile d'imiter artificiellement. Tout d'abord la faculté de parler, d'écouter mais aussi, de voir et de percevoir. Ensuite, viennent les facultés d'analyser, de synthétiser et de comprendre. Enfin, il faut savoir s'exprimer et faire comprendre aux autres nos réactions et nos désirs.

Ce processus très complexe, fait donc intervenir plusieurs capacités humaines qui interagissent entre elles pour délivrer un message et pour comprendre les messages des autres. La communication orale homme-machine, quant à elle fait intervenir deux acteurs, l'homme avec ses facultés et ses défauts et la machine. Les capacités de chacun de ces acteurs sont très différentes et jusqu'au moment de l'écriture de ces lignes, c'est l'homme qui a la charge de s'adapter aux systèmes de communication orale actuels [20]. C'est à lui de faire l'effort et d'adapter ses messages aux exigences de ces systèmes : vocabulaire limité, environnement non bruité, une manière correcte de s'exprimer, etc. La machine, elle, progresse mais le rêve de pouvoir faire comme l'homme reste encore loin.

L'étape de génération des commandes (génération de réponses + présentation du résultat) ne nous sert que pour vérifier et valider l'étape de compréhension.

I.2.2.1. Les différents types d'entrées des systèmes de compréhension

Chez l'homme, comprendre c'est construire une image mentale de tous ce qui l'entoure. Il met en œuvre pour cela ses cinq sens (toucher, vue et ouïe). Il s'agit donc de coordonner et d'analyser cinq types d'entrées en plus de ses croyances et de ses expériences antérieures afin de comprendre tout événement extérieur.

Pour la machine, ceci est trop complexe à réaliser, un système de compréhension en général, peut avoir plusieurs types d'entrées possibles relatifs à différentes modalités, comme la parole, le texte, les gestes, etc. mais leur cohésion et leur analyse et beaucoup plus modeste que ce qui se passe chez l'homme. De plus, pour un système de compréhension de la parole, l'entrée peut se présenter sous plusieurs formes possibles : signal, transcriptions phonétiques, parole reconnue, texte, etc. qui sont toutes issues de la même modalité à savoir, la parole. La chaîne de traitement de la parole est présentée par la Figure I.2, la sortie de chaque étape peut être considérée comme entrée possible au module de compréhension. Le traitement au sein du système de compréhension diffère d'une forme d'entrée à une autre. Cependant, ce système doit fournir en sortie une forme précise du résultat qui donne le sens littéral de ce qui a été prononcé.

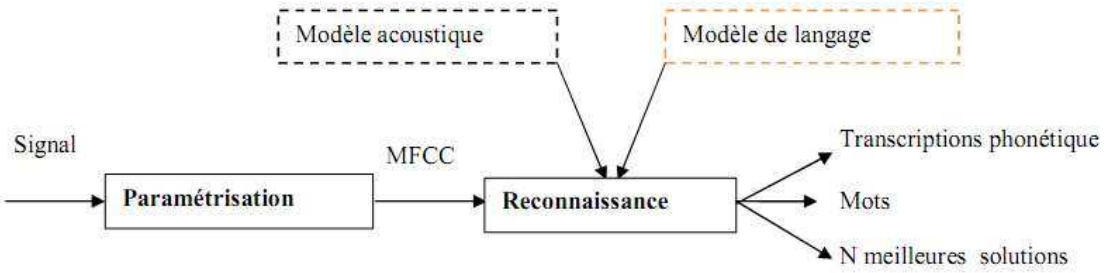


Figure I.2 : Chaine de traitement du signal de la parole

I.2.2.2. Le système de compréhension humain

Lorsqu'une personne prononce une phrase, les mots sont transmis au cerveau de son interlocuteur sous forme de signaux nerveux jusqu'au cortex auditif, qui est une aire sensorielle primaire. L'information auditive gagne ensuite l'aire de Wernicke, où les mots sont reconnus et où le sens de la phrase est décrypté. L'aire de Wernicke est une aire associative de type sensoriel qui intervient dans la perception des mots et des symboles du langage. Une personne souffrant d'une lésion dans cette région n'est pas sourde : elle peut encore entendre les mots et tous les autres sons, mais elle ne peut leur attribuer un sens. L'aire de Wernicke permet d'associer les mots, puis de retrouver leur sens. De ce fait, elle intervient directement dans la mémorisation des signes utilisés dans le langage. Les mécanismes mis en jeu dans cette aire nous intéressent particulièrement parce que c'est l'aire responsable de la compréhension chez l'homme. Cependant, il reste encore une grande part de mystère dans le fonctionnement réel au sein de cette aire comme au sein de la totalité du cerveau humain.

Détaillons tout d'abord l'acheminement d'une information perçue par l'homme. S'il s'agit de parole, les signaux sonores sont reçus dans le cortex auditif primaire et interprétés dans l'aire de Wernicke. Pour un mot lu, ce mot sera reçu dans le cortex visuel primaire et interprété dans la circonvolution angulaire et par la suite dans l'aire de Wernicke. Ensuite, les signaux de l'aire de Wernicke sont acheminés vers l'aire de Broca à travers un circuit nerveux qui relie les deux aires (Figure I.3).

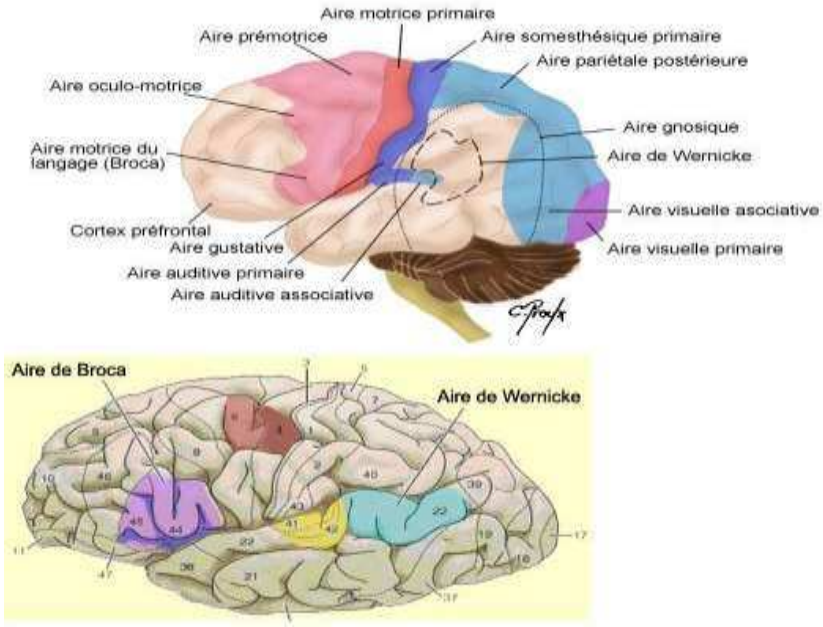


Figure I.3 : Les aires de Broca et de Wernicke.

Afin de produire une réponse, le cerveau active des programmes moteurs de l'aire de Broca pour le contrôle de la formation des mots et transmet des signaux au cortex moteur pour le contrôle des muscles de la parole. Schématiquement ceci peut être résumé d'une manière simplifiée par la Figure I.4.

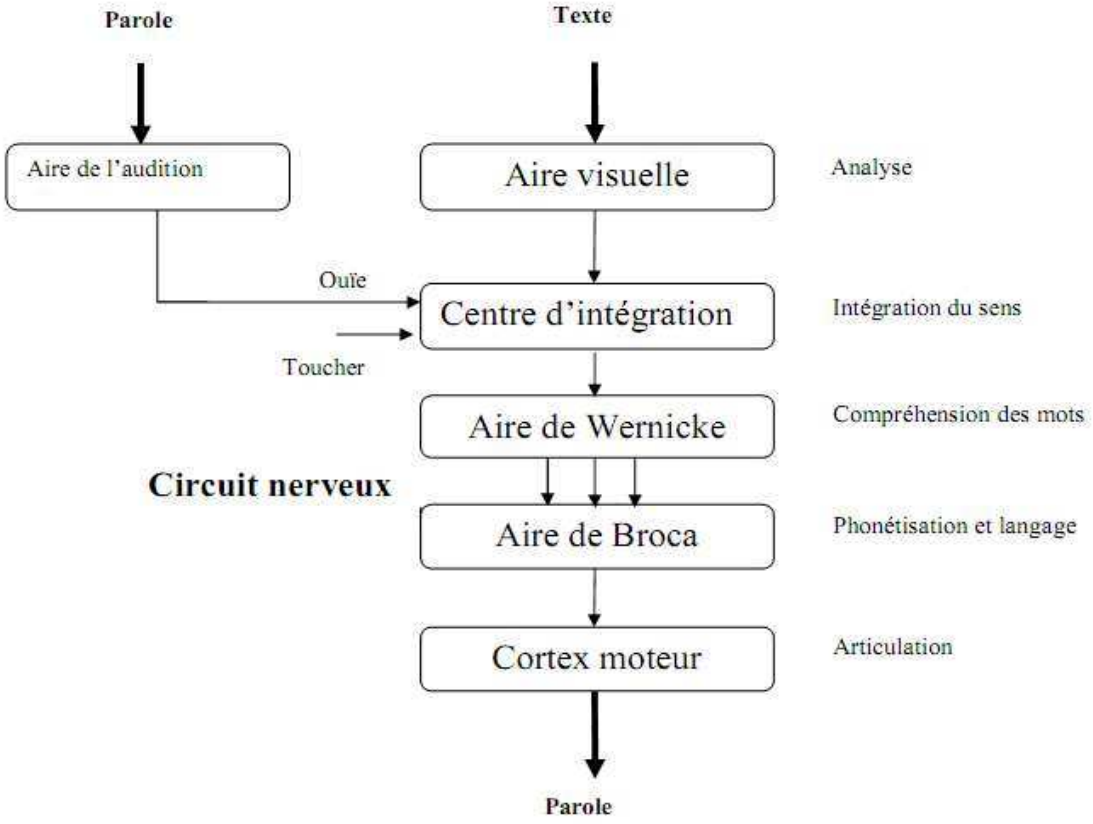


Figure I.4 : La séquence des traitements pour la perception et la production de la parole dans le cerveau humain.

D'une manière plus macroscopique nous trouvons les quatre étapes présentées dans la Figure I.5. La première est l'étape de perception. La deuxième est la construction d'une représentation mentale relative à ce qui a été perçu. La troisième est une étape d'évaluation de cette représentation mentale. Enfin, la quatrième étape consiste à réagir en se basant sur l'image mentale construite et l'évaluation qui lui a été faite.

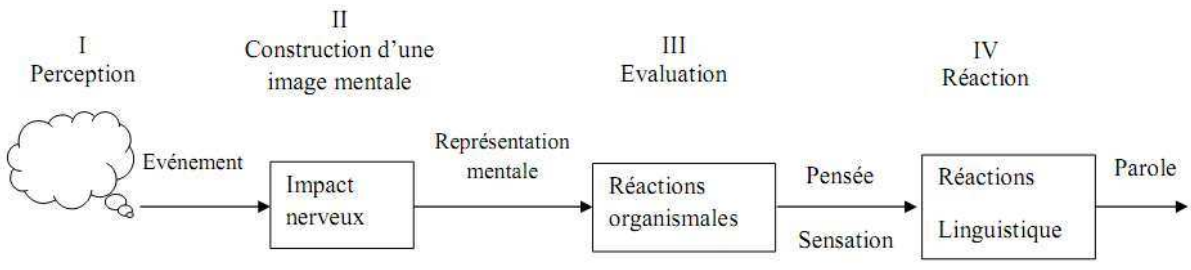


Figure. I.5 : Les différentes étapes de compréhension chez l'homme.

Ces différentes étapes correspondent en fait à des niveaux d'abstraction. Le niveau II représente un niveau d'abstraction plus élevé que le niveau I et au niveau III l'abstraction faite est plus élevée que celle du niveau II.

Dans son livre, Alfred Korzybski [21] a dit :

« **La totalité de notre connaissance fondamentale la plus profonde doit être et ne peut être autre qu'hypothétique ; en effet, ce que nous voyons, entendons, sentons, ce dont nous parlons ou ce que nous inférons n'est jamais *"ça", mais seulement le résultat de nos* abstractions humaines à propos de « ça ».** »

En effet, le langage utilisé pour décrire nos pensées n'est qu'un moyen d'interprétation imposé par notre communauté comme a dit Edward Sapir [22]:

« **Si nous voyons, entendons et éprouvons en général dans une très grande mesure comme nous le faisons c'est parce que les habitudes linguistiques de notre communauté nous prédisposent à certains choix d'interprétation.** »

Il en découle que le processus de compréhension de la parole passe par plusieurs étapes d'abstraction et que notre réponse n'est qu'une traduction limitée de ces abstractions. Si nous essayons de reproduire ce qui se passe chez l'homme d'une manière artificielle, il nous faut considérer les quatre grandes étapes de ce processus et qui sont présentées par la Figure I.5. Premièrement, il s'agit de l'étape de perception pendant laquelle la machine reconnaît le signal de la parole. Deuxièmement, elle construit une image conceptuelle de ce qu'elle a reconnu. Troisièmement, elle construit une réponse en réagissant à l'image construite et à la fin elle exprime sa réponse en utilisant le langage considéré par l'application mère. Il s'agit là exactement des étapes déjà données dans la Figure I.1 et qui décrivent l'architecture d'un système de compréhension automatique de la parole.
Ainsi, comme chez l'homme, le résultat d'un système de compréhension n'est complet que s'il est donné à l'issue de la quatrième étape du processus de compréhension. Cependant avec la machine, nous pouvons considérer que les sorties des étapes 2, 3 et 4 sont aussi valides car elles représentent des étapes de compréhension intermédiaires. En effet, la compréhension proprement dite correspond au travail fait au niveau de la deuxième phase (cf. Figure I.1). La troisième et la quatrième phase concernent la génération de la réponse et non l'acte de compréhension. Ces deux dernières étapes servent essentiellement à montrer que la machine a bien compris, sans leurs résultats notre compréhension serait sans intérêt.

I.2.2.3. Les structures de différentes représentations sémantiques

L'objectif d'un système de compréhension de la parole est d'extraire le sens utile d'un énoncé. Dans le cas des applications restreintes ceci peut être établit par la seule extraction des informations qui sont nécessaires pour l'application en question [23], [24]. Pour représenter le sens de la phrase, nous pouvons procéder à une transformation de la représentation des entrées (le signal de parole, la parole reconnue, sa transcription phonétique ou orthographique) en une autre représentation sémantique interprétable par le module suivant de la chaîne des traitements.
Dans la littérature, nous pouvons trouver plusieurs types de représentations sémantiques permettant de coder d'une façon interne un énoncé. Nous pouvons entre autres citer les

techniques fondées sur la logique (la logique des propositions, la logique des prédicats, la logique modale, etc.), les réseaux sémantiques et les graphes conceptuels [25], les structures de traits et les ensembles d'attributs, etc.

Dans un système de communication homme-machine, le module de compréhension est généralement placé en aval du module de reconnaissance et en amont du module de génération de réponses. Il doit pouvoir transformer la phrase reconnue en entrée en une représentation sémantique qui traduit le sens de cette phrase d'une manière efficace, complète, simple et facilement interprétable par le module suivant. Ce dernier doit ensuite utiliser cette représentation sémantique afin de générer une réponse correcte qui répond à la demande de l'utilisateur.

Pour des systèmes vocales permettant de consulter des bases de données, la représentation sémantique doit permettre de générer une requête de type SQL (Structured Query Language) pour interroger la base de données. De ce fait deux problèmes sont donc à résoudre. D'une part il faut trouver la bonne représentation sémantique à utiliser et qui doit être simple mais aussi efficace. D'autre part, il faut bien choisir le mécanisme d'association phrase/sens. Plusieurs méthodes ont été adoptées dans la littérature afin de résoudre ce dernier problème. Quelques-unes de ces méthodes d'association seront présentées en détails dans les sections suivantes

I.3. Etat de l'art des Systèmes de Compréhension

En réalité les Systèmes de Compréhension de la Parole reposent sur différentes approches. On y distingue principalement deux grands courants : d'une part celui des approches symboliques linguistiques, d'autre part celui des approches stochastiques fondées sur l'utilisation d'un grand nombre de données. Afin de rendre robuste le décodage des énoncés oraux, des travaux (Par exemple Goulian [26]; Villaneau [27]) ont souligné l'intérêt d'une combinaison des aspects syntaxiques et sémantiques. D'autres ont montré que le formalisme de la grammaire sémantique de cas [28] ou l'approche conceptuelle [29] est bien adapté à l'interprétation d'énoncés oraux.(Antoine et al.,)[30] ont procédé également à une analyse plus fine des phénomènes linguistiques de l'oral en se focalisant par exemple sur le traitement des incises ou des dislocations, ou encore sur la modélisation de la complexité structurale ([31]; [32]).

Un état de l'art décrivant les différents systèmes de compréhension des principaux laboratoires a été réalisé par De Mori [33] ainsi que dans le cadre de la thèse de Wolfgang Minker [28]. Nous proposons dans cette section de donner un aperçu global des diverses approches de compréhension répertorié selon quatre axes de classification [34] :

- ✓ Niveau de compréhension
- ✓ Nature des connaissances linguistiques
- ✓ Approche stochastique ou non
- ✓ Formalisme de représentation des connaissances linguistiques

Les approches stochastiques sont le plus souvent utilisées dans des systèmes guidés par la sémantique et les approches non stochastiques sont plus largement utilisées dans des systèmes guidées par la syntaxe. Pour les approches stochastiques, la représentation des connaissances est essentiellement réalisée à l'aide de grammaires hors contexte probabilisées, de modèles de Markov cachés et plus rarement avec des réseaux neuronaux. Les approches non stochastiques utilisent divers formalismes, le plus courant est la grammaire hors-contexte. Les règles peuvent être probabilisées ou non.
L'étude des travaux de («Aust et al., »[35], «Fereiros et al., » [36], Boros [37] et Hacioglu [38]) montre que des notions de classes de mots ou de concepts de rebut (filler en anglais) sont très utilisées afin de récupérer les mots inutiles ou inconnus de la compréhension. «Mayfield et al., »[39] et Wang [40] ont montré que l'on peut 'sauter' des mots à l'aide de règles spécifiques dans des grammaires hors-contexte. De telles approches permettent d'obtenir un module de compréhension robuste face à des difficultés telles que la présence de mots inconnus inutiles à la compréhension, la présence d'hésitations et certains types d'erreurs de reconnaissance (insertions ou substitutions de mots).

I.3.1. L'approche statistique de compréhension

I.3.1.1 Modèle théorique

L'approche statistique de la compréhension de la parole permet de construire des systèmes performants en limitant le besoin à l'expertise humaine, notamment pour écrire des règles ou pour élaborer des grammaires. Cependant ces approches nécessitent généralement de gros corpus de données (phrases) servant comme des corpus d'apprentissage. En plus, ces modèles statistiques s'adaptent à tout type d'application, à l'opposé des méthodes grammaticales qui sont généralement spécifiques à un domaine ou à une tâche bien précise. Dans ce dernier cas, les règles établies doivent être changées d'une application à une autre ce qui engendre un travail humain conséquent. Alors qu'avec des approches statistiques, on utilise toujours un même formalisme mathématique et ce sont seulement les données qui changent à chaque fois.

La plupart des systèmes de compréhension de la parole statistiques existant utilisent l'approche stochastique pour réaliser la tâche de décodage conceptuel. Il s'agit de segmenter la phrase d'entrée en une suite de concepts en supposant qu'il y a une correspondance séquentielle entre les mots de la phrase et les concepts [41], [42]. Le décodage conceptuel stochastique repose sur le principe de maximisation de vraisemblance. En effet, supposant que la phrase S est représentée par la séquence d'observations acoustiques A, le but est de trouver la suite des concepts C qui correspond à cette phrase et qui maximise la probabilité à posteriori P(C/A).
Notons $A = a_1, a_2, \ldots, a_N$ et $S = m_1, m_2, \ldots, m_K$ la suite des mots composant la phrase S. Où N et K représentent respectivement le nombre total d'observations acoustiques dans A et le nombre de mots dans la phrase S. On associe à chaque mot m_i de cette phrase un concept c_i. La suite des concepts relatifs à S sera donc représentée par $C = c_1, c_2, \ldots, c_K$. Ainsi, la séquence de concepts optimale qui correspond à la phrase S est donnée par l'équation (I.1) :

$$\hat{\mathrm{C}} = \operatorname{argmax}_{\mathrm{C}} \mathrm{P}(\mathrm{C}|\mathrm{A}) \quad \text{(I.1)}$$

A l'aide de la formule de Bayes, cette équation peut être réécrite sous la forme suivante :

$$\hat{C} = \operatorname{argmax}_{C} P(A|C) \frac{P(C)}{P(A)} \quad \text{(I.2)}$$

Sous l'hypothèse d'indépendance entre la séquence acoustique observée A et la séquence de concepts C, l'équation précédente devient :

$$\hat{C} = \operatorname{argmax}_{C,S} P(A|S)P(S|C)P(A) \frac{P(C)}{P(A)} \quad \text{(I.3)}$$

$$\hat{C} = \operatorname{argmax}_{S,C} P(A|S)\, P(S|C)P(C) \quad \text{(I.4)}$$

Chaque terme de cette équation représente un modèle bien particulier :

La probabilité P(A|S) étant évaluée dans le cadre de la reconnaissance de la parole, l'enjeu de la compréhension est donc la résolution de l'équation :

$$\hat{C} = arg\max_{C} P(S|C)P(C) \quad \text{(I.5)}$$

La probabilité P(S|C) d'une séquence de mots sachant une séquence conceptuelle représente le modèle de réalisation syntaxique. Elle est généralement estimée par des probabilités n-grammes de mots conditionnées par le concept associé au mot courant :

$$\mathrm{P}(\mathrm{S}|C) \simeq \prod_{i=1}^{\mathrm{N}} \mathrm{P}(m_i|m_{i-1}, \dots, m_{i-n+1}, c_i) \quad \text{(I.6)}$$

La probabilité P(C), probabilité a priori d'une séquence de concepts, représente le modèle sémantique. Son estimation est également réalisée par des probabilités m-grammes de concepts selon :

$$\mathrm{P}(C) \simeq \prod_{i=1}^{\mathrm{N}} \mathrm{P}(c_i|c_{i-1}, \dots, c_{i-m+1}) \quad \text{(I.7)}$$

N est la taille du corpus d'apprentissage.

Cette modélisation classique est une chaîne de Markov d'ordre n où seules les n dernières observations sont utilisés pour la prédiction du mot ou du concept suivant. Si on ne considère que l'état précédent donc : n = 1 et que m = 2. En d'autre terme, nous ne considérons respectivement que les probabilités conditionnelles des mots sur les concepts et que les bi-grammes de concepts (probabilité du concept sachant le concept précédent) :

$$\mathrm{P}(\mathrm{S}|C) \simeq \prod_{i=1}^{\mathrm{N}} \mathrm{P}(m_i|c_i) \quad \text{(I.8)}$$

$$\mathrm{P}(C) \simeq \prod_{i=1}^{\mathrm{N}} \mathrm{P}(c_i|c_{i-1}) \quad \text{(I.9)}$$

Les états cachés du HMM seront les concepts qu'on cherche à identifier et les observations seront donc les mots.

$P(c_i/c_{i-1})$ est la probabilité de transition de l'état c_{i-1} à l'état c_i.

$P(m_i/c_i)$ est la probabilité d'émettre l'observation m_i sachant que nous sommes à l'état c_i.

Ceci correspond à un modèle HMM à un seul niveau. Un exemple simplifié de modèle HMM pouvant modéliser ce problème est donné dans la Figure I.6.

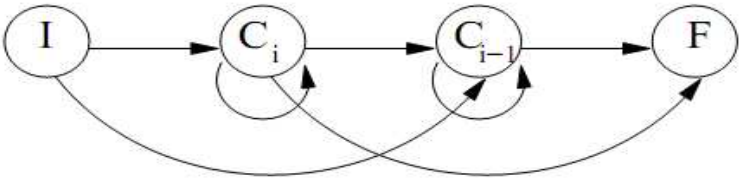


Figure I.6 : Un exemple de modélisation conceptuelle à l'aide d'un HMM à un seul niveau.

Si nous examinons les travaux de Bousquet [34], on remarque qu'à chaque état du modèle HMM à deux niveaux (Figure I.7), un autre HMM spécifique est associé au segment conceptuel SC_i. Et ce, du fait que comme un segment conceptuel peut contenir plus qu'un seul concept (ex : SC= C1 C2), il devient nécessaire de modéliser aussi les concepts à l'intérieur d'un même segment conceptuel.

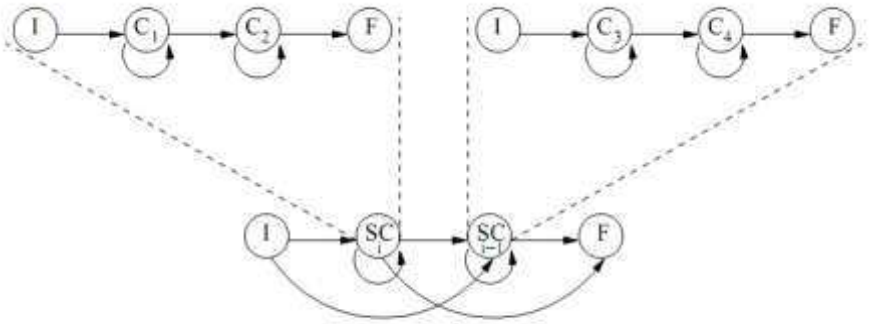


Figure I.7: Un exemple de modélisation des segments conceptuels à l'aide d'un HMM à deux niveaux.

Selon ce graphe, on trouve qu'au niveau un les états cachés sont les SC_i et les observations sont les concepts C_i. Alors qu'au niveau deux, les états cachés sont les concepts C_i et les observations sont les mots m_i.

Selon Minker et Wolfgang [43], les données contextuelles des mots (le contexte de mot) sont aussi considérées pour la modélisation conceptuelle et une interpolation entre le modèle conceptuel simple et celui contextuel est utilisé. Dans ce cas, le contexte ti du mot mi est introduit et la probabilité P(mi/ci) est remplacée par l'interpolation suivante :

$$\lambda \mathrm{P}(m_i, t_i|c_i) + (\lambda - 1)\mathrm{P}(m_i|c_i) \qquad \text{(I.10)}$$

I.3.1.2. Quelques applications

I.3.1.2.1. Une approche hybride : le système TINA

L'analyse d'un énoncé peut se faire de deux manières : (1) une analyse complète portant sur la syntaxe (permettant de comprendre des phrases complexes comme les formes passives ou les propositions relatives), (2) une analyse sémantique fondée sur des mots clefs ou des concepts. La deuxième approche autorise la compréhension d'énoncés dont la syntaxe n'est pas respectée, comme c'est souvent le cas à l'oral.
Le système TINA développé par le MIT reprend ces deux aspects, ce qui lui permet donc de comprendre la majorité des énoncés [44]. Le principe est le suivant :

- L'énoncé est analysé complètement à partir de sa forme syntaxique selon les mécanismes décrits par Seneff [45]. Cette analyse utilise une grammaire hors-contexte transformée de façon automatique en un automate portant des probabilités sur les arcs, ce qui permet d'avantager les constructions les plus courantes. Les nœuds de cet automate font référence à des catégories particulières, qui peuvent être sémantiques (comme les lieux) ou bien syntaxiques (par exemple les verbes ou les adjectifs).

- Si cette analyse ne permet pas de dégager de façon satisfaisante le sens de l'énoncé, on utilise alors la deuxième méthode : on analyse seulement les segments porteurs de sens (appelés concepts) et non plus l'énoncé complet [8]. Certains mots de l'énoncé n'appartenant à aucun concept ne sont pas utilisés lors de l'analyse. Si plusieurs solutions sont possibles, c'est celle qui a consommé le plus de mots qui est retenue.

I.3.1.2.2.Une approche conceptuelle : le système PHOENIX

Le système PHOENIX [46] inclut à la fois le module de reconnaissance de la parole et celui de compréhension. Nous présenterons ici uniquement le module de compréhension.
La représentation sémantique est donnée sous la forme d'un schéma (frame) comportant des pairs « attributs / valeur » (slots dans la littérature anglophone). Les connaissances linguistiques sont représentées par une grammaire hors-contexte partielle non stochastique et essentiellement guidée par la sémantique. En fait chaque paire attribut / valeur est représentée par un automate à états finis indiquant toutes les manières de dire une séquence de mot dont le sens correspond à cette paire. La grammaire est constituée d'un ensemble de sous-grammaires. Chaque automate peut en appeler un autre, ce qui réduit la taille de la grammaire. Ces paires sont équivalentes à la notion de concept que nous retrouverons entre autre dans le système CHRONUS décrit ci-dessous. Les mots ne correspondant à aucune paire ne sont pas interprétés, ce qui permet d'être robuste face aux phénomènes propres à l'oral spontané (extra-grammaticalités, présence d'hésitations…) ainsi qu'à la présence d'erreurs de reconnaissance mineures.
Le principe d'interprétation consiste à détecter les segments correspondant au pair attribut / valeur et à en déduire le schéma. Si plusieurs solutions sont possibles, on retient alors celle

qui a le meilleur score, ce score étant calculé en fonction du nombre de mots de l'énoncé qui ont été pris en compte pour l'interprétation.

I.3.1.2.3.Décodage conceptuel stochastique : le système CHRONUS

Le système CHRONUS (Conceptual Hidden Representation of Natural Unconstrained Speech) est un système de compréhension de la parole fondée sur une approche stochastique et conceptuelle développé par le laboratoire AT&T [47]. Un concept est défini ici comme une unité élémentaire de sens, de tel manière que chaque énoncé peut être représenté par une suite de concepts.

Dans la première version de CHRONUS [18], la compréhension était réalisée directement à partir du signal acoustique et incluait l'étape de reconnaissance de la parole. Nous allons décrire ici la version proposée en 1995 par Pieraccini [47].

Le processus de compréhension se décompose en deux modules correspondant au découpage classique de compréhension littérale et compréhension contextuelle.

Le processus de compréhension littérale, appelé analyse locale dans ces travaux, est réalisé en trois étapes (Figure I.6) :

✓ Le module d'analyse lexicale permet de faire des prétraitements sur l'énoncé à analyser et transforme cet énoncé en un treillis d'hypothèses lexicales. Certains mots sont regroupés (en particulier les locutions), les mots considérés comme synonymes dans le cadre de l'application sont remplacés par la classe de mots correspondante et certains mots comme les villes ou les jours de la semaine sont aussi remplacés par une classe de mots (par exemple le mot New-York est remplacé par la classe ville, lundi par jour). Lorsqu'il y a ambiguïtés, plusieurs solutions sont représentées par le treillis d'hypothèses lexicales.

✓ La seconde étape consiste à découper l'énoncé en concepts à l'aide du module de décodage conceptuel. Les connaissances linguistiques nécessaires pour cette étape sont données par une grammaire sémantique. Cette grammaire est représentée sous la forme d'un modèle de Markov caché dont les états sont les concepts. Les probabilités d'observation des états sont données par un modèle de langage bigramme de classes de mots. Le décodage conceptuel est réalisé à l'aide de l'algorithme de Viterbi classiquement utilisé avec les modèles de Markov cachés. Le résultat obtenu est une segmentation de l'énoncé en concepts.

✓ Un module de générateur de traits transforme la décomposition en concepts obtenue lors de l'étape précédente en un ensemble de paires attributs / valeurs.

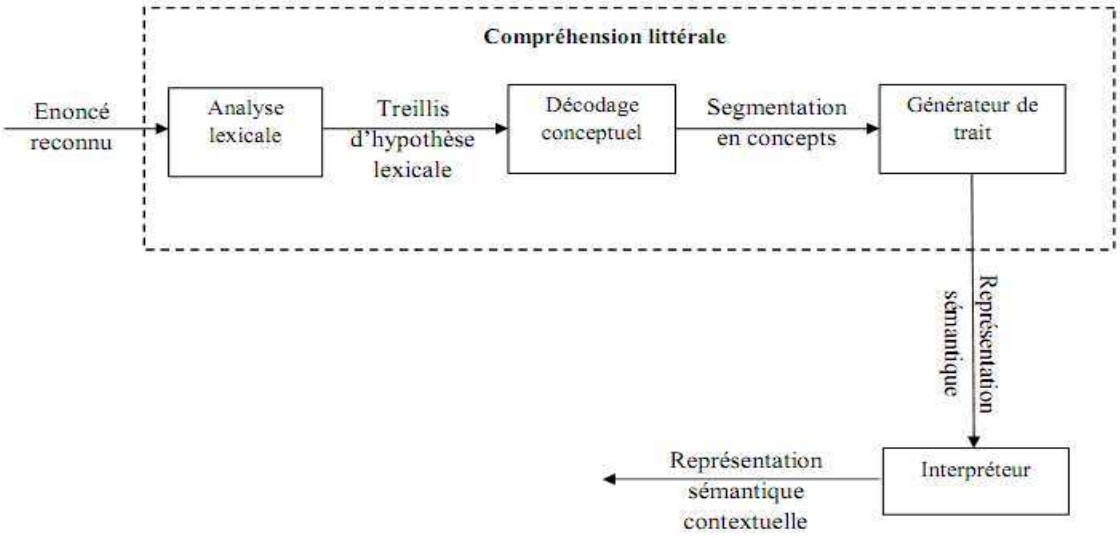


Figure I.8 : Architecture du système CHRONUS [47].

L'étape de compréhension contextuelle est réalisée avec le module appelé interpréteur et il consiste en une analyse globale de l'énoncé. Son rôle consiste entre autre à résoudre les ambiguïtés à l'aide de règles écrites manuellement (par exemple, s'il y a plusieurs villes de départ possibles, on ne considère que la dernière).
CHRONUS a été mis en œuvre sur la tâche de renseignements sur les horaires d'avion dans le cadre du projet ATIS.

I.3.1.2.4. Un modèle fondé sur la grammaire de cas : l'approche du LIMSI

L'approche que nous décrivons maintenant porte sur l'utilisation du formalisme de la grammaire de cas qui est fortement guidée par la sémantique. Elle a donné lieu à la réalisation de deux systèmes de compréhension, le premier utilise une méthode par règles non probabilisées [13] alors que la seconde est stochastique et utilise un modèle de Markov caché [28]. Ces deux systèmes ont été comparés dans Minker [48]. L'approche stochastique a l'avantage d'être plus simple à mettre en œuvre et d'être plus facilement portable d'une application ou d'une langue vers une autre.
La grammaire de cas a été introduite à l'origine par Fillmore et exprime les relations entre un verbe et ses compléments [49]. Cette théorie a été étendue par la suite par Bruce pour un système fondé sur les concepts [50].
Ce formalisme construit une représentation sémantique d'un énoncé sous la forme d'un schéma. Ce derniers correspond à un concept et comporte plusieurs attributs. Un ou plusieurs mots de références permettent d'identifier le concept et le schéma correspondant à ce concept et les attributs du schéma sont instanciés avec les cas identifiés grâce aux marqueurs de cas.

Exemple : dans le domaine du renseignement aérien [51]**:**

« Je voudrais **aller** de Oakland à Denver »

Le mot aller est le mot de référence (en gras dans l'exemple) permettant d'identifier le concept. Les mots soulignés sont les cas de ce concept et sont identifiés grâce à la présence des marqueurs de cas (notés en italique). Les autres mots sont jugés inutiles pour l'interprétation de cet énoncé.

L'approche stochastique a été mise en œuvre à l'aide d'un modèle de Markov caché dont les états sont les concepts, les attributs et les marqueurs. Un état particulier est utilisé pour recueillir les mots inutiles à l'interprétation. Une analyse lexicale est réalisée en prétraitement afin de supprimer les redondances, les informations inutiles et de remplacer certains mots par une classe de mots (par exemple toutes les villes sont étiquetés ville).

Cette approche a été implémentée dans le kiosque multimodal MASK donnant les renseignements ferroviaires, dans ATIS pour les horaires d'avion et dans L'ATIS (version française d'ATIS).

I.3.1.2.5. L'approche basé sur CACAO

Dans l'environnement CACAO (Compréhension Automatique par segments Conceptuels Assistée par Ordinateur) [24], la sémantique est représentée par des segments conceptuels. Un segment conceptuel permet de représenter les groupes de mots ayant un sens commun (par exemple des groupes de mots comme "partir pour Oran" et "aller à Tizi ouzou" font partie du même segment conceptuel direction). Chaque segment conceptuel est représenté par un modèle de Markov caché. On peut donc décomposer tout énoncé en une suite de segments conceptuels à l'aide d'un module de décodage conceptuel en utilisant l'algorithme de Viterbi.

A la fin de cette étape de décodage, seule la meilleure solution est retenue représentant ainsi le sens de l'énoncé. Le sens est ensuite interprété en utilisant les structures de traits, notées SDT. L'exemple donné par la Figure I.9 illustre les résultats de ces deux étapes : le décodage conceptuel et la représentation SDT, obtenus pour une phrase donnée en entrée.

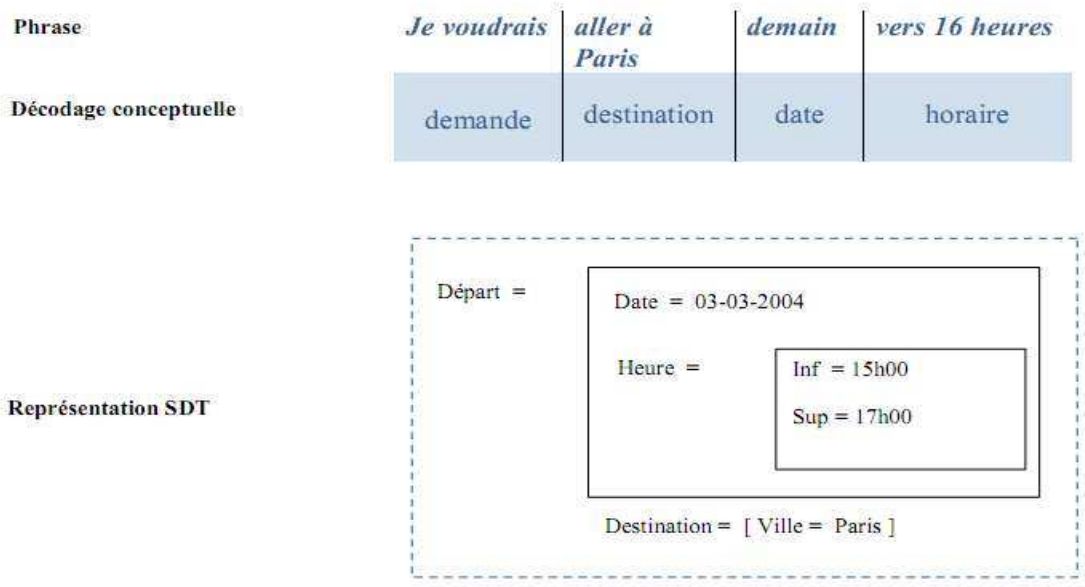


Figure I.9 : Le décodage conceptuel et la représentation SDT utilisés dans l'environnement CACAO.

I.3.2. Le convertisseur de représentation

Nous allons entamer la description de la deuxième composante d'un système de compréhension de la parole selon l'architecture illustrée par la Figure I.10. Il s'agit du module de conversion de représentation (ou gestionnaire de dialogue) qui vient après l'étape de compréhension proprement dite.

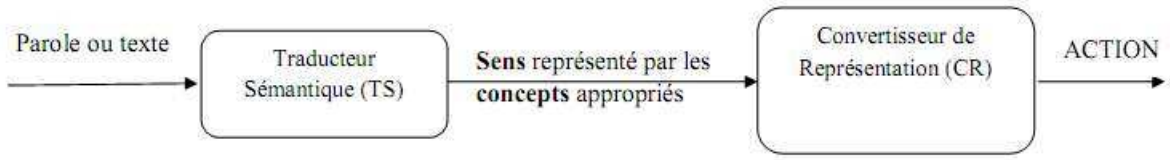


Figure I.10 : Architecture générale d'un système de compréhension automatique de la parole.

L'importance de cette étape vient du fait qu'elle nous permettre de mieux apprécier la validité du sens fournit par le module de compréhension. Elle nous fournit l'action/information liée à notre requête.

Dans la littérature des systèmes de compréhension automatique de la parole, peu de travaux se sont intéressés à cette étape malgré le fait qu'elle soit présente dans l'architecture de tous les systèmes d'interaction orale homme-machine. Cette étape consiste à traduire une représentation conceptuelle en un langage formel cible. Ce langage peut être le langage de commandes Shell, un langage pour l'interrogation des bases de données comme le langage SQL, ou même dans des cas plus évolués un langage de programmation avec au début de simples instructions. Le cas le plus présent dans la littérature est celui de la conversion concepts/commandes SQL. En effet, une application faré des systèmes de compréhension automatique de la parole est l'interrogation des bases de données afin d'extraire une information particulière. Les exemples dans ce cas de figure sont multiples, le cas le plus présent est celui de l'interrogation des bases de données ferroviaires ou aériennes pour la consultation des horaires ou la réservation de places, ou pour la demande de tarifs ou de services.

La conversion concepts/commandes SQL n'est pas immédiate. En effet, ces deux langages sont très différents et généralement les concepts et les paramètres de la base de données à consulter sont très distincts et parfois ils n'offrent pas la même classification des données. Par exemple, si nous trouvons le concept réservation ”et si nous voulons l'exécuter en tant qu'une commande SQL alors il faut procéder à une vérification des places, à une sélection des vols, de services, d'horaire, etc. En plus, il faut trouver quelque chose qui réponde à toutes ces conditions, sinon il nous faut proposer un autre vol similaire. Il y a tant de travail à faire que la conversion en devient très compliquée. Dans « Schwartz et al.,» [52], un compilateur a été conçu pour la traduction concepts/commandes SQL. Les requêtes SQL construites pour la tâche ATIS faisaient parfois une demi-page de long rien que pour deux contraintes. En générant les requêtes SQL avec ce compilateur, les auteurs ont constaté une perte de performance de 3.5% par rapport aux résultats de compréhension. Ceci montre la complexité de la tâche de conversion de représentations.

Dans un travail similaire sur la tâche ATIS [18], la tâche de conversion concepts/commandes SQL a été divisée en deux étapes :

- Une génération d'une requête « patron »: il s'agit de convertir les concepts trouvés par le traducteur sémantique en une table de paires (attribut, valeur). Le nom de l'attribut est obtenu par une traduction directe concept/attribut à l'aide d'un simple mécanisme d'inférence. Par exemple, le concept Question est traduit comme étant l'attribut Requête. L'attribut Objet est toujours présent, il définit le sujet de la requête. La détermination des valeurs de ces attributs est une tâche plus compliquée.

En effet, ces valeurs sont obtenues à l'aide d'un mécanisme d'inférence plus compliqué qui met en œuvre un ensemble de règles spécifiques à chaque concept afin de pouvoir extraire sa valeur. Un exemple d'une requête « patron » est donné dans la table I.1.

Requête	**Liste**
Objet	Avion
Escales	0
Aéroport_origine	ALGER
Aéroport_dest	PARIS
Jour	Dimanche

Table I.1 : Un exemple d'une requête « patron »

- Traduction SQL : ce traducteur génère d'une manière dynamique les requêtes SQL finales afin d'extraire l'information demandée. Au sein de ce module, la requête « patron » est traitée selon l'attribut Objet. Ensuite, chaque attribut dans cette requête est interprété selon la table correspondante dans la base de données à consulter.

I.4. Conclusion

Dans ce chapitre nous avons présenté un état de l'art des Systèmes de Compréhension Automatique de la Parole. Nous avons commencé par une description cognitive du processus de compréhension en faisant à chaque fois une analogie avec les systèmes de compréhension automatique. Ensuite, nous avons donné les méthodes de compréhension existantes (l'approche linguistique, l'approche connexionnistes et statistique). Enfin, nous avons présenté plus particulièrement l'approche statistique pour la compréhension automatique de la parole où nous nous sommes attardés sur les différentes représentations sémantiques adoptées dans ce cas.

Cette dernière approche nous intéresse particulièrement puisqu'elle nous offre la possibilité d'automatiser une grande partie du traitement. A l'opposé, l'approche linguistique nécessite l'élaboration manuelle des grammaires ou des graphes sémantiques ou autres. La méthode statistique présente aussi l'avantage d'être assez transparente par rapport à l'approche connexionniste. En effet, les réseaux de neurones sont généralement considérés comme des

boites noires utilisées pour des objectifs bien précis sans pouvoir expliquer tout le traitement qui s'y réfère.

Dans l'approche statistique, nous avons proposé une diversité de méthodes qui seront plus développés dans les chapitres suivants. Dans ce travail, nous cherchons à élaborer une méthode totalement automatique pour la compréhension de la parole où l'intervention humaine est minime.

Chapitre II :

Traitements et Représentation des unités d’énoncé en Compréhension de l’énoncé

Chapitre II :

Traitements et Représentation des unités d'énoncé en Compréhension de l'énoncé

II.1. Introduction

Dans ce chapitre, on va entamer le problème de la compréhension de l'énoncé, en considérant l'aspect cognitif humain. C'est-à-dire, pour l'humain comprendre est l'aptitude de détecter les différents unités d'énoncés (les mots), leurs attribuer une représentation mentale (concept de sens). Et enfin récupérer le sens latéral de l'énoncé.
Pour notre système, ceci est équivalent à détecter les différents mots de l'énoncé (par segmentation). Ensuite, retrouver les concepts (par classification).

II.2. Système de Compréhension d'Énoncé en CHM

D'après Didierjean [53] on peut énoncer le problème de la compréhension de la manière suivante :
Soit S un énoncé de problème constitué de N objets. Comprendre cet énoncé se résumera en deux étapes.

1- Une analyse descendante de l'énoncé S, afin de trouver les différents objets.
2- Une étape d'identification des relations établis entre ces objets.
3- Une analyse ascendante depuis ces relations, aboutira à la compréhension de l'énoncé.

Par conséquent, la compréhension d'un énoncé textuel est établie en identifiant les termes (objet=terme) appartenant à l'énoncé. La reconstitution des relations entre ces termes engendra la compréhension de l'énoncé.

D'où, la première approche qu'on a considérée (basé sur les concepts) [19] de ce système. Son architecture figurant dans la Figure II.1 comprend quatre blocs : le bloc d'analyseur morphologique, syntaxique, sémantique et pragmatique. L'analyseur morphologique peut se subdiviser en deux sous-analyseurs structurels et lexicaux [54]. Cela revient à intégrer l'analyseur syntaxique dans celui de la sémantique [55].

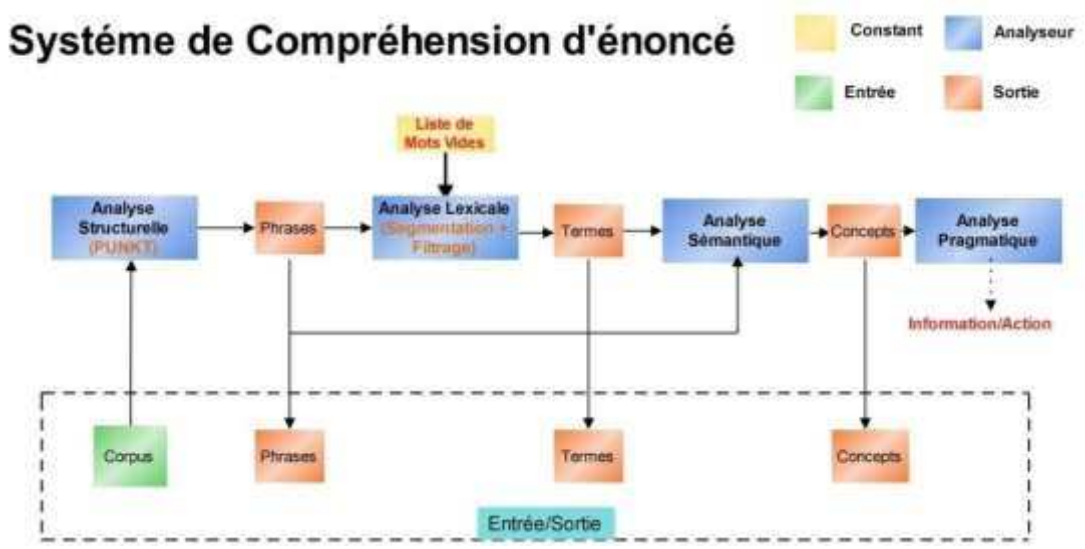


Figure II.1 : L'architecture du système de Compréhension de l'énoncé basé sur l'approche conceptuelle

II.2.1. Analyse Structurale

Dans le domaine de la littérature, le structuralisme cherche à expliquer **les structures sous-jacentes** des textes littéraires, soit en termes d'une grammaire calqué sur celui du langage ou en termes de principe de Ferdinand de Saussure que le sens de chaque mot dépend de sa place dans l'ensemble du système de la langue.
Pour Jean-Dominique Robert.[1]

> « **Il y a analyse structurale…, sur un contenu culturel donné (quel qu'il soit d'ailleurs), … quand on parvient à isoler un ensemble formel d'éléments et de relations, sur lequel il est possible de raisonner sans faire appel à la signification du contenu en question**».

En d'autres termes, le structuralisme n'est pas concerné par le contenu d'un texte ou d'un type de système, mais plutôt des analyses, car il explore les structures sous-jacentes du texte ou du système, ce qui rend le contenu possible. Un des grands principes du structuralisme est : la forme définit le contenu ("**forme est le contenu**"). Ainsi, le structuralisme analyse la manière dont le sens [56] est possible et montre comment elle est transmise quelle que soit la signification réelle [57].
En se basant sur ces éléments et en considérant le traitement mental de l'humain lors de la lecture d'un texte, on peut dire que l'humain instinctivement fait une division du texte en phrases (détection des débuts et fins de phrases). Or ceci est fastidieux, lent, **error-prone**[2], et extrêmement difficile.

[1] Jean-Dominique Robert ; *'Analyse « structurale» et analyse « symbolique». Quelques aperçus capitaux d'un* livre récent sur la *communication'*.
[2] Définition : capable de faire une erreur; "tous les hommes sont sujettes à l'erreur"

Donc se pose le besoin **de Détection des Frontières des Phrases**.[3]

II.2.1.1. Système de Détection de Frontières des Phrases

La grande majorité des applications de traitements du langage naturel présuppose le découpage de textes en phrases. Cette tâche est depuis très longtemps automatisée, on parle alors de reconnaissance de frontières des phrases. La phrase est considérée comme l'unité centrale des processus du traitement du langage naturel. On désigne par phrase la suite des mots qui se trouvent entre des signes de ponctuation dits majeurs tels que le point, le point d'exclamation, le point d'interrogation et d'autres qui précédent ou suivent ces signes. Par la ponctuation ; on désigne l'ensemble des signes qui permettent l'interprétation des textes écrits. Traditionnellement la ponctuation est le module de la compétence langagière le plus difficile à maîtriser, d'une part ; à cause de son caractère écrit, d'autre part à cause de différents styles appliqués par les auteurs dans la littérature.

Le problème de la détection automatique des phrases est le fruit de l'ambiguïté de certains signes de ponctuation. Par exemple un point peut être utilisé pour déclarer la fin d'une phrase mais aussi pour exprimer une abréviation ou un acronyme (ex. : E.D.F. ou U.S.A.), ou même un nombre décimal, par exemple : 3.14 (écriture anglo-saxonne).

II.2.1.2. Travaux effectués dans le domaine de la détection des phrases

Les travaux concernant la détection automatique des phrases représentent un premier pas pour une analyse morphologique ou syntaxique. Les techniques utilisées visent à reconnaître les cas les plus courants. Dans leur majorité ces techniques sont orientées vers la détection des phrases dans des corpus ou dans une langue donnée, ce qui rend difficile de les adapter à un autre type de textes ou à une autre langue sans avoir à modifier l'algorithme. Donc, le développement d'un système basé sur de grandes listes d'abréviations, ou de lexiques spécialisés avec des informations qui ne s'utiliseront plus par la suite, n'est pas la meilleure solution.

L'approche la plus simple consiste à l'usage des règles qui reconnaissent des suites de caractères (par exemple : point – espace – lettre majuscule), accompagnées de longues listes d'exceptions pour les abréviations. Une approche différente a été proposée par [58], au lieu des listes d'abréviations, une analyse morphologique filtre les mots ayant les mêmes suffixes, pour lesquels la probabilité d'être une abréviation est minime ou nulle. De telles approches demandent plusieurs heures de travail pour construire et renouveler les listes de règles et d'abréviations. De plus, elles sont orientées à un type particulier de corpus.

L'approche avec le plus grand taux de détection [59] est basée sur un corpus annoté de 25 millions de mots. Sa performance pour le corpus Brown atteint 99,8%. Pour chaque mot du dictionnaire, ce modèle demande le calcul de probabilités qu'il soit le premier ou le dernier

[3] Donc, les structures sous-jacentes évoquées par l'analyseur structurale n'est autre que les frontières des phrases.

mot d'une phrase. Cette information est inutile aux analyses qui peuvent suivre, morphologiques ou syntaxiques.

De plus la détection n'est qu'un moyen pour permettre à divers outils d'effectuer leur analyse des phrases découpées, donc le coût de calcul doit être minime. C'est pour cette raison que [60] proposent un modèle basé sur la plus grande entropie qui ne nécessite aucun calcul ni le coût d'une information compliquée. L'information utilisée par ce modèle est fondée sur le segment (lexème)[4] qui contient le signe de la ponctuation candidat pour la fin d'une phrase, ainsi que les autres lexèmes, juste avant et juste après celui-là. Pour la construction d'une liste d'abréviations extraite du corpus annoté, un algorithme simple est utilisé. L'approche de la plus grande entropie atteint 97,7% de solution pour le corpus Brown, avec utilisation d'une liste manuellement construite d'abréviations, d'appellations et d'acronymes. Si on enlève cette information supplémentaire, le taux est de 97,5%.

Une autre approche est le système SATZ[5] [61], qui utilise un réseau neuronal pour la désambiguïsation des frontières d'une phrase. Il est basé sur des probabilités d'appartenance d'un mot à une partie du discours principale (prior part-of-speech). Par exemple, le mot « porte » peut être un verbe mais il est plus probable qu'il soit un nom selon les textes utilisés. Ce système utilise un dictionnaire de 30000 entrées lexicales et une information sur 6 différents « lexème » concernant leur contexte: 3 lexèmes avant et 3 après la fin candidate, et sa performance est de 98,5%, sur un corpus composé d'articles du Wall Street Journal. Le système SATZ peut être utilisé pour d'autres types de corpus ou d'autres langues naturelles après un apprentissage. Le problème demeure du fait qu'il existe des types de corpus, ou même des langues, pour lesquelles des dictionnaires spécifiques contenant des informations sur les principales parties du discours ou sur la façon de les détecter automatiquement n'existent pas.

Les auteurs insistent sur le fait que la performance du système n'est pas influencée par la diminution de la taille du dictionnaire.

Même si cette information est vraie, et peut être utile par la suite à un autre traitement du langage, il y a de traitements pour lesquels elle ne s'y applique pas, comme l'alignement des phrases [62] (sentence alignement).

Il faut préciser que le système SATZ ainsi que l'approche de l'entropie maximale, ne distinguent pas les différents usages des signes de ponctuation qui montrent la fin d'une phrase, et qu'ils utilisent les mêmes règles partout. Or, il y a des ambiguïtés qui ne sont pas traitées. Par exemple, un point peut être utilisé pour une abréviation mais ce n'est pas le cas d'un point d'interrogation ou d'un point d'exclamation. Cette théorie, dont les applications comme par exemple l'analyse syntaxique de surface[6] [63], [64] sont largement répandues dans le

---

[4] Le lexème ou le morphème lexical, objet d'étude de la lexicologie, peut correspondre :
- •au radical du mot lorsqu'il est lié à un ou plusieurs morphème (-s) ou
- •au mot lui-même lorsqu'il est libre.

[5] Satz est un mot allemand signifiant la phrase.

[6] La segmentation du texte consiste à diviser un texte en parties syntaxiquement corrélés de mots,

domaine du TALN, consiste à extraire automatiquement de la connaissance linguistique sous forme de règles à partir des corpus annotés.
On va évoquer dans ce qui suit le système qu'on a adopté pour réaliser cette tache (Détection de frontière de phrase).

II.2.1.3. Approche non supervisé de détection de frontière de phrase multilingue

Les auteurs Kiss et Strunk ont présenté une méthode de détection de frontière de phrase qui s'appuie sur des méthodes indépendantes de la langue (multilingue). Celle-ci n'utilise pas d'annotations supplémentaires, étiquetage grammatical « part-of-speech tagging », ou des listes précompilées pour soutenir la détection des frontières de phrases; mais plutôt elle se sert du texte à segmenter pour extraire tous les informations nécessaire.
Dans le même temps, la structure modulaire du système permet d'intégrer des méthodes spécifiques à la langue et des indices pour continuer à améliorer sa précision (afin de supporter d'autre langue). L'algorithme de base a été déterminé expérimentalement sur la base d'un corpus de développement annoté de l'anglais. Le système résultant a été appliqué à d'autres corpus de texte en langue anglaise ainsi que des corpus de dix autres langues: portugais, brésilien, néerlandais, estonien, français, allemand, italien, norvégien, espagnol, suédois et Turks. Sans ajouts ou des modifications au système résultant par le biais de l'expérimentation sur le corpus de développement, la précision moyenne de la détection des frontières de phrases sur des corpus de journaux en onze langues est de 98,74%.
La détection de frontière des phrases sera approchée en déterminant d'abord les abréviations possibles dans le texte. Quantitativement, les abréviations sont une importante source d'ambiguïtés dans la détection des frontières de la phrase, car ils constituent souvent jusqu'à 30% des candidats possibles pour les frontières de phrases dans le texte courant.
Ils ont assumé que les abréviations sont les collocations[7] des mots, et par conséquent les méthodes pour la détection des collocations peut être appliquée avec succès à la détection des abréviations. Firth [65] caractérise les collocations d'un mot comme «**états des lieux ou la coutume habituelle de ce mot** ». Dans des langues que les abréviations sont marqués avec une période suivant, on pourrait dire que l'abréviation est habituellement composé d'un mot (ou une séquence de mots) et une période (point) qui a suivi (U.S.A.). Idéalement, en l'absence d'homographie et les fautes de frappe, une abréviation doit toujours se terminer par une dernière période[8]. Par conséquent, une abréviation est caractérisée [7] comme une collocation

[7] Quand les mots sont utilisés ensemble, régulièrement, les règles sont formés sur leur utilisation n'est pas pour des raisons grammaticales, mais à cause de l'association. «Noir et blanc » apparaissent dans cet ordre en raison de colocation, ils sont toujours dans cet ordre et les mettre dans l'autre sens semble autour de mal. Pour la même raison que nous «faire une erreur» lorsque nous «faire un test ». La raison de l'utilisation de ces verbes avec ces derniers est que nous le faisons toujours, ce qui est de colocation.

[8] Il existe plusieurs types d'abréviations qui ne sont généralement pas marqués par une période finale: des acronymes, comme les abréviations de l'OTAN ou unité kg, tels (kilogramme). Celles-ci ne pré-

très stricte. Des techniques seront modifiées de manière appropriée pour tenir compte de la stricte égalité entre un mot abrégé et la période suivante. Il doit être clair dès le départ que les abréviations ne peuvent pas être réglées, en les énumérant, parce qu'ils forment une classe de mot productives et donc ouverte *"open word class"* [58] Et [66].

De même une caractérisation formelle des abréviations en termes de trois propriétés majeures fut proposée par kiss [7], qui ne s'appuient que sur le type de mot candidat lui-même et non sur le contexte local dans lequel le candidat apparaît.

Tout d'abord, comme cela a été déjà mentionné, une abréviation ressemble à une **collocation** très serré comme quoi le mot abrégé précédant la période et la période forment un lien étroit.

Deuxièmement, les abréviations ont tendance à être plutôt courte. Cela ne signifie pas que nous avons à assumer une limite supérieure fixée pour la durée d'une abréviation possible, mais que la probabilité d'être une abréviation baisse si les candidats deviennent plus longs. En utilisant **la longueur d'un candidat** comme un contrepoids pour le lien de collocation entre candidats et la dernière période permet à notre méthode pour identifier les abréviations assez longue.

Comme une troisième propriété caractéristique, est la présence de périodes de mot internes "**word internal periods**" contenues dans de nombreuses abréviations. Même si les propriétés ci-dessus ont été déterminées expérimentalement, les auteurs croient qu'elles représentent effectivement des traits essentiels des abréviations.

En utilisant seulement ces trois caractéristiques, ce système est capable de détecter les abréviations avec une précision moyenne de 99,38% sur des corpus de journaux en onze langues. Les chiffres présentés ne comprennent pas les **initiales (Mr.)** et le **numéro ordinal (2.5)** parce que ces sous-classes des abréviations ne peuvent pas être découvertes à l'aide de ces caractéristiques et doivent être traités différemment. Le système complet avec des heuristiques spéciales pour les initiales et les nombres ordinaux atteint une précision de 99,20% pour la détection des abréviations, des initiales, et les nombres ordinaux.

La détermination des types d'abréviation produit un grand pourcentage de toutes les frontières de phrase car toutes les périodes survenant après les types non-abréviation peuvent être classées comme des marqueurs de fin de phrase. Une telle homonymie sur le niveau de type, cependant, est insuffisante en soi, car il doit encore être déterminé pour chaque période suivante d'une abréviation si elle sert de marqueurs de fin de phrase en même temps. La détection des initiales et des nombres ordinaux, qui sont représentés par des chiffres suivis d'une période en plusieurs langues, exige également l'application des méthodes à base de jetons, car ces sous-classes des abréviations sont problématiques pour les méthodes de type base. Ces observa-

---

sentent pas un problème pour la détection de frontière de phrase et ne sont donc pas décrits dans le présent article. Nous allons désormais utiliser l'abréviation pour désigner qu'aux classes des abréviations qui sont normalement marqués d'une dernière période.

tions suggèrent un traitement en deux étapes de détection des limites de phrase qui est à la fois à base de type et de jetons.

Dans un premier temps, une résolution est effectuée sur le niveau du type pour détecter les types abréviation et les types de mot ordinaire. Après cette étape, le corpus reçoit une annotation intermédiaire où toutes les instances des abréviations détectées par la première phase sont marquées avec le marqueur **<A>** et toutes les ellipses avec le marqueur **<E>**. Toutes les périodes suivantes, non-abréviations sont supposées être les bornes de phrase et reçoivent l'annotation **<S>**.

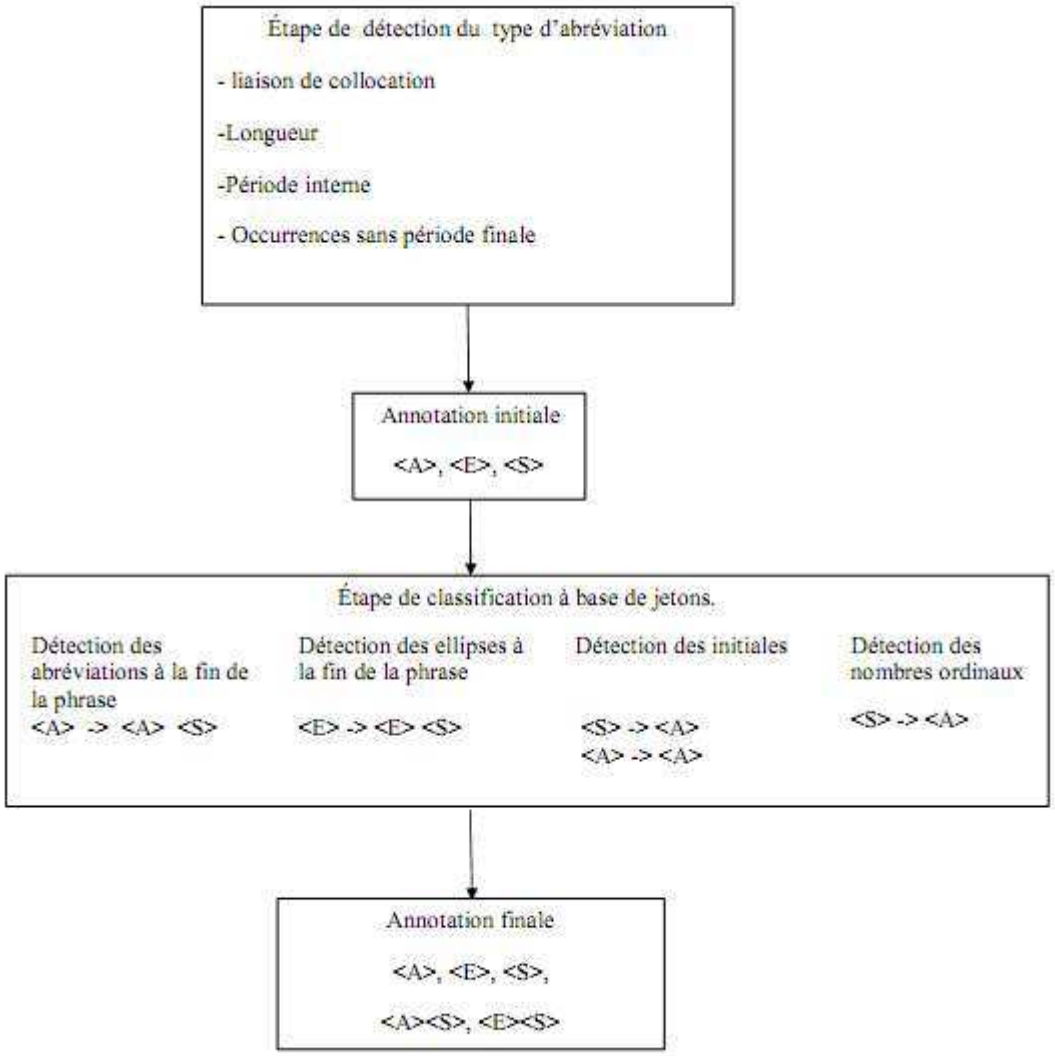


Figure II.2 : L'architecture globale du système Punkt

La seconde étape, à base de jetons emploie des heuristiques supplémentaires sur la base de l'annotation intermédiaire pour affiner et de corriger la sortie du premier classificateur pour chaque jeton individuel. Le classificateur à base de jetons est particulièrement adapté pour

déterminer les abréviations et les ellipses (**…**)[9] à la fin de la phrases et de leur donner la dernière annotation **<A> <S>** ou **<E> <S>**. Mais il est également utilisé pour corriger l'annotation intermédiaire en détectant les initiales et le numéro ordinal qui ne peuvent pas être facilement reconnue par des méthodes de type base et donc souvent mal recevoir l'annotation de la première étape. L'architecture globale du système actuel, baptisé **Punkt** (allemande pour la période), est donnée dans la Figure II.2.

Pour l'évaluation, des versions annotées du corpus de test, ont été crié, dans lequel toutes les périodes (points) ont été clarifiées à la main et étiquetés avec le marqueur correct du tableau 1.

Tableau 1
Les marqueurs utilisés dans l'évaluation
<S> Frontière de phrase
<A> Abréviation
<E> Ellipse
<A> <S> Abréviation à la fin de la phrase
<E> <S> Ellipse à la fin de la phrase

En résumé Punkt est conçu pour lever l'ambiguïté des périodes et des ellipses. Son principe repose sur deux étapes:

Etape 1 : **identification des abréviations et des ellipses**

En utilisant le critère de la langueur :

- Si une collocation suivie par une période est assez long, elle est considérée comme abréviation et reçoit l'annotation **A**. Sinon le candidat appartient à la classe des non abréviations et reçoit l'annotation de fin de phrase **S**.
- Si une période est suivie par une autre pour une certains langueur elle sera considérée comme ellipse et elle reçoit l'annotation **E**.

Etape 2 : **identification de la fin de la phrase**

Parmi les jetons déjà trouvé A, E et S y en a certains qui seront considérés comme une fin de phrase. Cette ambigüité sera levée en considérant le contexte.

- Si après une abréviation A, le mot commencera par une lettre en majuscule, cette abréviation est considérée comme une fin de phrase. Elle recevra l'annotation **A S.** Sinon elle ne sera pas considéré comme tel et gardera l'annotation initiale **A**.
- De même pour l'ellipse E, le même cas est considéré. En premiers cas, il recevra l'annotation **E S**, sinon gardera l'annotation initiale E.
- Pour la liste des non abréviations tel que les initiales (Mr.) et les nombres ordinaux (2.5), on a deux possibilités.
    - ✓ Pour les initiales : s'il est suivi par un espace et que le mot suivant commence par une majuscule (Mr. **L**ICHOURI Mohamed), on doit vérifier les collocations possibles. Si ce

[9] L'ellipse grammaticale : omission d'un mot ou d'un verbe. Souvent cet usage de la figure n'est pas destiné à produire un effet particulier, il s'agit avant tout de faire l'économie d'une répétition souvent par une énumération : « Café, bain, travail... Deux pages par jour, d'accord ? » (Philippe Sollers).

mot appartient à cette liste, donc l'initiale ne sera pas considéré comme fin de phrase et recevra l'annotation des abréviations **A**. si par contre, ce mot n'appartient pas à la liste des collocations il sera considéré comme fin de phrase et recevra l'annotation **S**.

- ✓ Pour les nombres ordinaux : s'il est suivi par un espace et que le mot suivant commence par une lettre majuscule, il sera considéré comme fin de phrase et gardera l'annotation **S.** En revanche, si le mot suivant ne commence pas par une majuscule, ce nombre n'est pas considéré comme fin de phrase.

II.2.2. Analyse Lexicale

Nous sommes maintenant en présence des phrases (matière principale de notre système de compréhension), on doit faire face aux contraintes posées par le système cognitif humain, du fait qu'il ne peut comprendre une phrase sans avoir recours à la découper en mots. C'est le rôle de l'analyseur lexical.

Pour y parvenir, il suffit de détecter les « blancs » ou le caractère « espace » dans une phrase. La boite à outils NLTK du langage Python permet de réaliser cette tâche en utilisant la fonction « Split », en voici un exemple :

```
>>> WordList=text.split()

>>> WordList

['puis-je', 'avoir', 'mon', 'relevée', 'de', 'note', '.', 'donne', 'moi', 'mon', 'diplôme', '.', 'puis-je',
'me', 'renseigner', 'sur', 'ma', 'note', '.', 'quel', 'note', 'ai-je', 'obtenu', 'en', 'examen', '?', 'je',
'souhaite', 'avoir', 'ma', 'certificat', 'de', 'scolarité', '.', 'je', 'veux', 'mon', 'diplôme', '.', 'donner',
'moi', 'ma', 'certificat', 'de', 'scolarité', '.', 'puis-je', 'avoir', 'mon', 'diplôme', '.', 'donne', 'moi',
'ma', 'note', "d'examen", '?', 'veux', 'tu', 'me', 'donner', 'ma', 'certificat', 'STP', '?', 'pouver',
'vous', 'me', 'montrer', 'mon', 'relevée', 'de', 'note', '.', 'je', 'voudrais', 'savoir', 'ma', 'note', '?',
"j'aimerai", 'bien', 'avoir', 'ma', 'certificat', 'de', 'scolarité', '.']
>>>
```

A ce stade on se pose la question : comment trouver le sens du mot et quelle est la phrase d'appartenance ?

Mais une étape intermédiaire entre l'analyseur lexicale et sémantique est nécessaire, car il faut filtrer les termes non signifiants ou indésirable.

En effet ; la segmentation du texte donne :

['je', 'puisse', 'classifier', 'mes', 'mots', 'en', 'concepts', '"', 'classes', '"', 'on', 'va', 'opter', 'pour', 'la', 'représentation', 'matricielle', 'qui', 'tient', 'compte', 'de', 'l', "'", 'information', 'mutuelle', 'et', 'du', 'contexte', '.',]

On remarque alors que tous les trucs ('je','mes','en','.'…) peuvent être filtrés sans endommager trop le sens de la phrase.

Ce filtrage est intéressant, car on aura une réduction de dimension ce qui simplifie la tache de l'analyseur sémantique.

Pour ceci on pourra utiliser ce qu'on appelle par **"Stop Words"**[10] ou la liste des mots vides**.**

Cette liste contient des mots propres à une langue donnée. L'absence ou la présence de ces mots ne changera plus le sens de la phrase. On pourra trouver sur internet plein de liste comme ceci:

"a afin ah ai aie aient aies ailleurs ainsi ait alentour alias allais allaient allait allons allez *alors Ap. Apr. aprs aprs demain arrire as assez..."* [67]

En ce qui concerne le toolkit NLTK, on pourra même utiliser cette fonction:

```
>>> ignored_words = nltk.corpus.stopwords.words('french')
```

Mais pour des raisons de simplifications, on pourra plutôt créer notre liste de "stop words" en utilisant cette fonction:

```
>>> sub_word = [w for w in tokens if len(w)<=3] % Pour tous les mots appartenants à la liste tokens, on ne considérera que ceux avec une longueur inférieur ou égale à trois caractéres.
```

Mais ...On aura surement des ambiguïtés, comme celui-là:

" je ne veut pas aller à 7.00h"

Ou les termes 'ne' et 'pas' seront éliminer ce qui veut dire une contradiction totale.

Or, on pourra bien résoudre ce problème en ne tenant pas compte des phrases négatives**.**

II.2.3. Analyse Sémantique

Dans le cadre de l'analyse sémantique, le fait que deux mots s'écrivent de la même manière ne signifie pas forcément qu'ils ont le même sens !

Par exemple, du point de vue grammatical le mot "vienne" peut prendre un sens différent dans des phrases comme "qu'il vienne ici" (dans ce cas c'est le verbe "venir") et dans d'autres comme "aller en vacances à Vienne" (ici c'est un nom propre). Du point de vue sémantique, le même mot "vienne", peut correspondre à la ville de "Vienne", qui existe en Autriche et en France, ainsi qu'au département de la Vienne (il faudrait dire "dans la Vienne"), etc. Il y a une

---

[10] Some search engines don't record extremely common words in order to save space or to speed up searches. These are known as "stop words."
Saving Space: Consider this sentence:
The way to the school is long and hard when walking in the rain.
The appears three times. To save space, a search engine might replace it with what's called a marker.
The sentence would be stored like this:
* way to * school is long and hard when walking in * rain.
Speeding Searches: Some search engines store every word on a web page but they don't search for certain ones to save time..... **What Are Stop Words?**
By **Danny Sullivan,** Search Engine Watch, Jan 1, 2003

multitude de mots ambigus dans toutes les langues, ce qui peut créer des incompréhensions.
Il est donc fondamental d'analyser le sens des mots pour comprendre ce qu'on dit (ou bien ce que les autres disent). C'est une opération humaine que nous effectuons tous les jours, sans forcément en être conscient, qui pose de nombreux problèmes pour l'analyse automatique sur ordinateurs. En particulier dans les moteurs de recherche (par exemple, Google), les logiciels de traduction (par exemple, Systran) et les correcteurs orthographiques (par exemple, Druide Antidote), qui affichent des résultats contenant de nombreuses erreurs. Toutefois certaines sociétés progressent dans ce secteur (comme par exemple, Tropes d'Acetic, ou encore les "Kaliwatch" de la société Arisem), ce qui nous permet d'espérer que l'ordinateur nous comprendra, vraiment, un jour,
L'analyse sémantique latente (Latent Semantic Analysis) LSA est un modèle statistique développé par les laboratoires Bellcore en 1989, pour faire la recherche documentaire [68]. Mais très vite, grâce à ses performances, son utilisation s'est étendue à d'autres domaines [11][69].

II.2.3.1. Approche utilisé dans notre système

L'approche adoptée est celle du système CHRONUS [47]. Ce système de compréhension utilise une grammaire sémantique de concepts[12] ou chaque énoncé peut être représenté par une suite de ces concepts.
Par contre il ne se compose que d'un module de découpage classique de compréhension littérale du fait que notre système est basé sur une grammaire hors-contexte (ne tient pas compte de l'historique).
Le processus de compréhension littérale est réalisé en trois étapes :

- ✓ L'analyse lexicale qui permet de faire des prétraitements sur l'énoncé à analyser et transforme cet énoncé en un sac de mots signifiants (voir § 2.2 de ce chapitre).

- ✓ L'analyse sémantique qui consiste à regrouper les mots en concepts. Une première étape intermédiaire est nécessaire utilisant un modèle vectoriel qui est une représentation mathématique du contenu d'un document, selon une approche algébrique.

Ce modèle concernait originellement les documents textuels et a été étendu depuis à d'autres types de contenus. Chaque contenu est ainsi représenté par un vecteur V, dont la dimension correspond à la taille du vocabulaire. Chaque coordonnées V_i du vecteur V est pondéré par un terme d'indice i et correspond à l'échantillon de texte (booléen, fréquentiel, TF-IDF,...). Cette représentation est suivie par une deuxième étape intermédiaire qui est l'utilisation d'un classificateur pour regrouper les vecteurs dans une classe « concept ».

11 Utilisation de l'analyse sémantique latente pour tenter d'optimiser l'acquisition par exposition à une langue étrangère de spécialité.
12 Un concept est défini ici comme une unité élémentaire de sens

✓ Une troisième étape consistant à découper l'énoncé en concepts à l'aide du module de décodage conceptuel. Les connaissances linguistiques nécessaires pour cette étape sont données par une grammaire sémantique.

II.2.3.2. Modèles de représentation

II.2.3.2.1. Modèle vectoriel

Comme mentionné en-dessus, ce modèle est basé sur une approche algébrique [13][70] de représentation du contenu d'un document. Elle est utilisée en **recherche d'information**, notamment pour la recherche documentaire, la classification ou le filtrage de données. Ce modèle a été étendu pour pouvoir supporter plusieurs types de contenus. Nous allons adapter cette méthode en considérant que :

- Le type de contenu est la phrase ou l'énoncé.
- Chaque mot de la phrase est représenté par un seul vecteur.
- Chaque vecteur représente un seul sens.

Dans la littérature, on trouve plusieurs représentations vectorielles de mots [71], [72]. Celles-ci ne correspondent pas à notre souhait. En effet, un mot n'est représenté que par son apparition dans la phrase [71], parfois, un mot est représenté par ses caractéristiques lexicales et grammaticales [72]. Dans notre cas, on veut une représentation sémantique des mots. C'est-à-dire que chaque mot sera représenté par un vecteur de caractéristiques. Cette représentation, constitue une étape clé dans le processus de compréhension. En fait, selon cette représentation, le classificateur doit pouvoir regrouper dans la même classe, les mots ayant le même sens. Chaque groupement donnera un concept, on aura ainsi un ensemble de groupements, donc de concepts.

Cette représentation dite sémantiquement significative doit respecter l'hypothèse d'indépendance des vecteurs où chaque terme est indépendant des autres.[14] Ceci dit, avoir des vecteurs indépendants.

---

[13] L'algèbre linéaire est la branche des mathématiques qui s'intéresse à l'étude des espaces vectoriels (ou espaces linéaires), de leurs éléments les vecteurs, des transformations linéaires et des systèmes d'équations linéaires (théorie des matrices).

[14] « Selon Shannon, pour une source sans mémoire ; c'est-à-dire pour laquelle les mots sont indépendant les uns des autres,… » Audio By Mario Rossi.

"Les mots sont indépendants, comme les chats, et ils ne font pas ce que vous voulez. Vous avez beau aimer, les flatter, leur parler doucement, ils s'échappent et partent à l'aventure." Jacques Poulin

II.2.3.2.2. Le Modèle booléen

Le modèle booléen traite un énoncé ou une phrase comme une expression logique. Considérons l'ensemble de phrases S = {p_1, p_2, ..., p_m} où : m représente le nombre de phrases contenu dans le corpus, et l'ensemble de termes T = {t_1, t_2, ...,t_n} où : n représente le nombre de termes signifiants appartenant au corpus.

Pour chaque phrase p_j du corpus, on compte le nombre d'occurrences de chaque terme t_i où :

$$tf_{ij} = \begin{cases} 1, & si\ t_i \in p_j \\ 0, & sinon \end{cases} \quad \text{(II.1)}$$

Ces occurrences forme un vecteur de représentation de phrase tel que :

$$V_{p_j} = \begin{vmatrix} tf_{1j} \\ \vdots \\ tf_{nj} \end{vmatrix} \quad \text{(II.2)}$$

Où : V_{p_j} est le vecteur de représentation de la phrase p_j.

Par analogie pour chaque terme t_i, on aura :

$$V_{t_i} = |tf_{i1} \ldots \quad tf_{im} \quad \text{(II.3)}$$

Où : V_{t_i} est le vecteur de représentation du terme t_i.

L'ensemble de ces vecteurs forme la matrice A souvent appelé matrice d'occurrences terme-phrase.

Cette matrice A est une matrice de dimension (n x m), dont les lignes correspondent aux « termes » et dont les colonnes correspondent aux « phrases ».

$$A = \begin{matrix} p_1 & \cdots & p_m \end{matrix} \begin{pmatrix} tf_{11} & \cdots & tf_{1m} \\ \vdots & \ddots & \vdots \\ tf_{n1} & \cdots & tf_{nm} \end{pmatrix} \begin{matrix} t_1 \\ \vdots \\ t_n \end{matrix} \quad \text{(II.4)}$$

La matrice A est une matrice creuse, de ce fait elle contient trop de bruit. Celui-ci nous engendra les anomalies suivantes :

- Deux termes appartenant à deux phrases différents n'auront pas la même idée et de ce fait ne seraient pas attribués au même concept. Alors que ceci n'est pas toujours vrai pour le cas des synonymes.

  Exemple 1:

  P_1 : Je **veux** ma **note**.

  P_2 : Je **désire** avoir mon **diplôme**.

  Après avoir être soumis aux analyseurs structurel et lexical, on aura la matrice d'occurrence suivante :

	V_{P1}	V_{P2}
V_{T_1} : veux	1	0
V_{T_2} : note	1	0
V_{T_3} : désire	0	1
V_{T_4} : diplôme	0	1

Table II.1 : Matrice d'occurrence basée sur le modèle booléen

En appliquant une mesure de distance euclidienne, on aura :

	V_{T_1}	V_{T_2}	V_{T_3}	V_{T_4}
V_{T_1}	0	**0**	**1.44**	1.44
V_{T_2}	**0**	0	1.44	1.44
V_{T_3}	**1.44**	1.44	0	**0**
V_{T_4}	1.44	1.44	**0**	0

- On remarque que la distance entre les vecteurs V_{T_1} et V_{T_3} est très grande, de ce fait ces deux vecteurs ne vont pas être attribués au même concept (donc une erreur de compréhension).

- Deux termes appartenant à la même phrase auront la même idée et vont appartenir au même concept.
  - On remarque encore que la distance entre les vecteurs (V_{T_1} ,V_{T_2}) et (V_{T_3} ,V_{T_4}) appartenant à la même phrase est nulle, qui en résulte à les attribuer au même concept ($Concept_1$= T_1 T_2 ; $Concept_2$= T_3 T_4)

Du point de vue, recherche d'information, la simplicité du modèle le rend aisément compréhensible pour un utilisateur. Dans le cas de corpus explorés par des spécialistes, ayant notamment une très bonne connaissance du vocabulaire, cela le rend très efficace. Cette simplicité le rend néanmoins peu adapté à des recherches sur des corpus généralistes, surtout lorsque les utilisateurs n'ont pas d'expertise sur ce dernier. En particulier, la formulation des requêtes devient vite laborieuse quand la requête se fait précise (donc longue).
Morphologiquement, on voit que la matrice d'occurrence tirée du modèle booléen ne respecte pas l'hypothèse d'indépendance où pas mal de mots peuvent être exprimés en fonction des autres (dépendant). Le problème de la pertinence de la recherche (pouvoir trier les résultats par ordre d'importance ou de relevance) avec le modèle booléen standard se fut corrigé avec l'utilisation du modèle fréquentiel.

II.2.3.2.3. Approche basée sur la fréquence d'occurrences TF (Term Frequency)

La fréquence d'un terme (term frequency) est le nombre d'occurrences de ce terme dans la phrase considérée, normalisée en la divisant par la somme des nombres d'occurrences de tous les termes de la phrase. Le nombre d'occurrence peut rendre compte de "l'importance" d'un terme dans une phrase. La normalisation du nombre d'occurrences d'un terme rend possible la comparaison de deux phrases de longueurs différentes.

Soit la phrase p_j et le terme ti, alors la fréquence du terme dans la phrase est :

(II.5)

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

Où $\boldsymbol{n_{i,j}}$ est le nombre d'occurrences du terme $\mathbf{t_i}$ dans $\mathbf{p_j}$. Le dénominateur est le nombre d'occurrences de tous les termes dans la phrase $\mathbf{p_j}$.

Pour l'exemple 2:

P_1 : Je **veux connaitre** ma **note**.

P_2 : Je **désire** avoir mon **diplôme**.

On aura la matrice d'occurrence suivante :

	V_{P1}	V_{P2}
V_{T_1}: veux	1/3	0
V_{T_2}: connaitre	1/3	0
V_{T_3}: note	1/3	0
V_{T_4}: désire	0	1/2
V_{T_5}: diplôme	0	1/2

Table II.2 : Matrice d'occurrence basée sur le modèle TF

En se basant sur cet exemple, on voit que les poids des termes dans les deux phrases n'est plus les mêmes. Ceci exposera au mieux leurs importances dans ces deux phrases.

En plus de sa simplicité, le modèle fréquentiel a pu partiellement résoudre le problème de la pertinence envers une requête (n'est pas supporter par celui booléen). Mais l'indépendance entre les termes n'est pas encore atteinte.

II.2.3.2.4. Approche basée sur la pondération TF-IDF

C'est une méthode de pondération souvent utilisée en recherche d'information et en particulier dans la fouille de textes. Cette mesure statistique permet d'évaluer l'importance d'un terme contenu dans une phrase, relativement à une collection ou un corpus. Le poids augmente proportionnellement au nombre d'occurrences du terme dans la phrase TF. Il varie également en fonction de la fréquence du terme dans le corpus IDF. La fréquence inverse de document IDF

(inverse document frequency)[15] est une mesure de l'importance du terme dans l'ensemble du corpus. Dans le schéma TF-IDF, elle vise à donner un poids plus important aux termes les moins fréquents, considérés comme plus discriminants. Elle consiste à calculer le logarithme de l'inverse de la proportion de phrases du corpus qui contiennent le terme :

$$idf_i = \log \frac{|P|}{|\{p_j : t_i \in p_j\}|} \qquad \text{Où} \quad \begin{cases} |P| : \text{nombre total de phrases dans le corpus} \\ |\{p_j : t_i \in p_j\}| : \text{nombre de phrases où le terme ti apparaît} \\ \text{(C'est-à-dire } n_{i,j} \neq 0\text{).} \end{cases} \qquad \text{(II.6)}$$

En prenant l'exemple 3 :

P_1 : Je **veux connaitre** ma **note**.

P_2 : Je **veux** avoir mon **diplôme**.

On aura la matrice suivante :

	V_{P1}	V_{P2}
V_{T_1} : veux	$\frac{2}{3} * \log(\frac{2}{2})$	0
V_{T_2} : connaitre	$\frac{1}{3} * \mathbf{log}(\frac{\mathbf{2}}{\mathbf{1}})$	0
V_{T_3} : note	$\frac{1}{3} * \mathbf{log}(\frac{\mathbf{2}}{\mathbf{1}})$	0
V_{T_4} : désire	0	$\frac{1}{2} * \mathbf{log}(\frac{\mathbf{2}}{\mathbf{1}})$
V_{T_5} : diplôme	0	$\frac{1}{2} * \mathbf{log}(\frac{\mathbf{2}}{\mathbf{1}})$

Table II.3 : Matrice d'occurrence basée sur le modèle TF-IDF

On remarque ici, que ce modèle donne de l'importance aux **termes rares.** Et cela grâce à la IDF, qui est maximale pour les termes rares = log(p/1). D'une part ceci est très encourageant auprès des moteurs de recherches et donne des résultats satisfaisants. Mais l'indépendance entre les termes n'est pas encore atteinte.

II.2.3.2.5. Approche basée sur la mesure d'information mutuelle

Dans la théorie des probabilités et la théorie de l'information, l'information mutuelle de deux variables aléatoires est une quantité mesurant la dépendance statistique de ces variables. Elle se mesure souvent en bit.

L'information mutuelle d'un couple (X, Y) de termes représente leur degré de dépendance au sens probabiliste.

Soit (X, Y) un couple de termes de densité de probabilité jointe données par P(x, y). Notons les distributions marginales P(x) et P(y). Alors l'information mutuelle dans le cas discret est:

[15] C'est une méthode pour apprécier la pertinence d'un document en fonction des critères de recherche de l'utilisateur. Dans notre cas, on considère que chaque document contient une seul phrase. C'est pourquoi on peut appliquer cette méthode pour pondérer notre matrice d'occurrence (terme-phrase).

(II.7)

$$I(\mathrm{X},\mathrm{Y}) = \sum_{x,y} \mathrm{P}(x,y) \log \frac{\mathrm{P}(x,y)}{\mathrm{P}(x)\mathrm{P}(y)}$$

Voici un extrait du résultat obtenue pour notre corpus:

```
[[+0.06, +0.04, +0.06, +0.04, +0.03, +0.06, +0.00, +0.04, +0.02, +0.03,
[+0.04, +0.02, +0.04, +0.02, +0.00, +0.04, +0.05, +0.02, -0.01, +0.00,
[+0.06, +0.04, +0.06, +0.04, +0.03, +0.06, +0.00, +0.04, +0.02, +0.03,
[+0.04, +0.02, +0.04, +0.02, +0.00, +0.04, -0.02, +0.02, +0.03, +0.00,
[+0.03, +0.00, +0.03, +0.00, -0.01, +0.03, -0.03, +0.00, -0.02, -0.01,
[+0.06, +0.04, +0.06, +0.04, +0.03, +0.06, +0.00, +0.04, +0.02, +0.03,
[+0.00, -0.02, +0.00, -0.02, -0.03, +0.00, -0.06, -0.02, -0.04, -0.03,
[+0.04, +0.02, +0.04, +0.02, +0.00, +0.04, -0.02, +0.02, -0.01, +0.00,
[+0.02, -0.01, +0.02, -0.01, -0.02, +0.02, -0.04, -0.01, -0.03, +0.11,
[+0.03, +0.00, +0.03, +0.00, -0.01, +0.03, -0.03, +0.00, -0.02, -0.01,
[+0.06, +0.04, +0.06, +0.04, +0.03, +0.06, +0.05, +0.04, +0.02, +0.03,
[+0.06, +0.04, +0.06, +0.04, +0.03, +0.06, +0.00, +0.04, +0.02, +0.03,
[+0.06, +0.04, +0.06, +0.04, +0.03, +0.06, +0.00, +0.04, +0.02, +0.03,
[+0.06, +0.04, +0.06, +0.04, +0.03, +0.06, +0.00, +0.04, +0.02, +0.03,
[+0.04, +0.02, +0.04, +0.02, +0.00, +0.04, -0.02, +0.02, -0.01, +0.00,
[+0.04, +0.02, +0.04, +0.08, +0.00, +0.04, -0.02, +0.02, -0.01, +0.00,
[+0.06, +0.04, +0.06, +0.04, +0.03, +0.06, +0.00, +0.04, +0.02, +0.03,
```

Figure II.3 : La matrice d'occurrence basée sur la mesure d'information mutuelle

Á l'inverse des autres modéles déjà vues, ce modèle n'est pas utilisé dans les moteurs de recherches. Et ceci est due au fait qu'il n'exprime pas l'importance des termes dans les phrases (les relations entre les termes et les phrases), mais plutôt les relations des termes entre eux. Donc, on peut dire qu'on pourra sans aucun problème créer un thésaurus[16] à l'aide de ce modèle. Mais le problème de la dépendance n'est pas encore résolu.

II.2.3.2.6. Modèle LSA « L'analyse de la sémantique latente »

A. Aspect psychologie cognitive

Le **modèle LSA** est un formalisme qui s'appuie sur une conception multidimensionnelle de la mémoire permanente pour représenter, sur une vaste échelle, les connaissances des sujets humains. Les connaissances sont représentées sous la forme de vecteurs de grandes dimensions, correspondant chacun à un mot ou à un groupe de mots. Elles sont produites à partir d'une analyse automatique (mathématique) du contenu textuel latent de larges corpus de textes. Le but de cette analyse est de représenter le sens des mots, en prenant en compte le contexte dans lequel chaque mot apparaît. En effet, des mots qui apparaissent dans des contextes similaires peuvent être considérés comme étant proches sur le plan sémantique (tout comme des con-

[16] Un thésaurus ou thésaurus de descripteurs, est un type de langage documentaire qui consiste en une liste de termes sur un domaine de connaissances, reliés entre eux par des relations synonymiques, hiérarchiques et associatives.

textes qui contiennent des mots similaires peuvent être considérés comme étant proches sémantiquement).

Un mécanisme d'induction permet d'inférer la similarité entre deux mots sur la base de leur cooccurrence dans une même phrase ou un même paragraphe. De nombreuses expériences ont montré des corrélations entre les performances de 'LSA' et celles de sujets humains sur des tâches diverses, justifiant ainsi la puissance des représentations sous-jacentes. Par exemple, [73] ont testé les représentations produites par l'analyse automatique d'un corpus de 4,6 millions de mots sur la partie 'synonymie' du TOEFL (Test Of English as a Foreign Language), qui est un test standardisé mesurant le niveau d'anglais de sujets non-anglophones. Il consiste à déterminer parmi quatre mots voisins d'un mot donné celui qui est le vrai synonyme. LSA a obtenu un score de 51,5 qui était comparable au score moyen des étudiants non-anglophones postulant à l'entrée aux universités américaines (51,6).

B. Aspect linguistique

LSA est un modèle d'acquisition de connaissances à partir de textes issu de LSI (Latent Semantic Indexing), un outil développé pour la recherche documentaire [68]. L'objectif est de représenter le sens des mots à partir de l'analyse de grands corpus de textes, en prenant en compte le contexte dans lequel chaque mot apparaît. En effet, des mots qui apparaissent dans des contextes similaires peuvent être considérés comme étant proches sur le plan sémantique. Par exemple, le mot «vélo» apparaît dans le contexte de mots comme «pédaler», «randonnée», «guidon», etc. Le mot «bicyclette» apparaît statistiquement dans des contextes similaires. Ces deux mots vont donc être représentés par des objets proches. On voit donc apparaître une double récursivité : deux mots sont similaires s'ils apparaissent dans des contextes similaires ; deux contextes sont similaires s'ils contiennent des mots similaires.

Il s'agit de déterminer le nombre d'occurrences de chaque mot dans chaque contexte. L'unité de contexte retenue sera donc la phrase, qui possède l'avantage d'être de taille correcte tout en étant facilement repérable par une machine.

Une forme d'analyse factorielle est ensuite appliquée à cette matrice afin de représenter chaque mot et chaque phrase par un vecteur de très grandes dimensions. Deux mots apparaissant dans des contextes similaires sont représentés par des vecteurs proches (la mesure de similarité est définie par le cosinus de leur angle).

C. Aspect mathématique

Le processus commence avec la création de la matrice d'occurrence terme par phrases A. S'il y a un total de n termes et m phrases dans le corpus, alors nous aurons une matrice A de dimension n × m pour le corpus (sans perte de la généralité $n \geq m$[17]). Étant donné que chaque terme n'apparaît pas dans chaque phrase, la matrice A est donc creuse[18] [74].

[17] Le nombre totale des termes du corpus est supérieur au nombre de phrases.

[18] Une matrice creuse est une matrice contenant beaucoup de zéros.

La deuxième opération consiste à appliquer une Décomposition en Valeur Singulier 'SVD' de A.

C.1. Décomposition en valeurs singulières

La décomposition en valeurs singulières prend une matrice rectangulaire de données (défini comme A, où A est une matrice n x m) dans lequel les lignes n représentent les termes, et les colonnes m représentent les phrases.

L'importance de la SVD vient du fait qu'elle va nous permettre de passer d'un espace de terme-phrase vers un espace réduit de terme&phrase-concept.

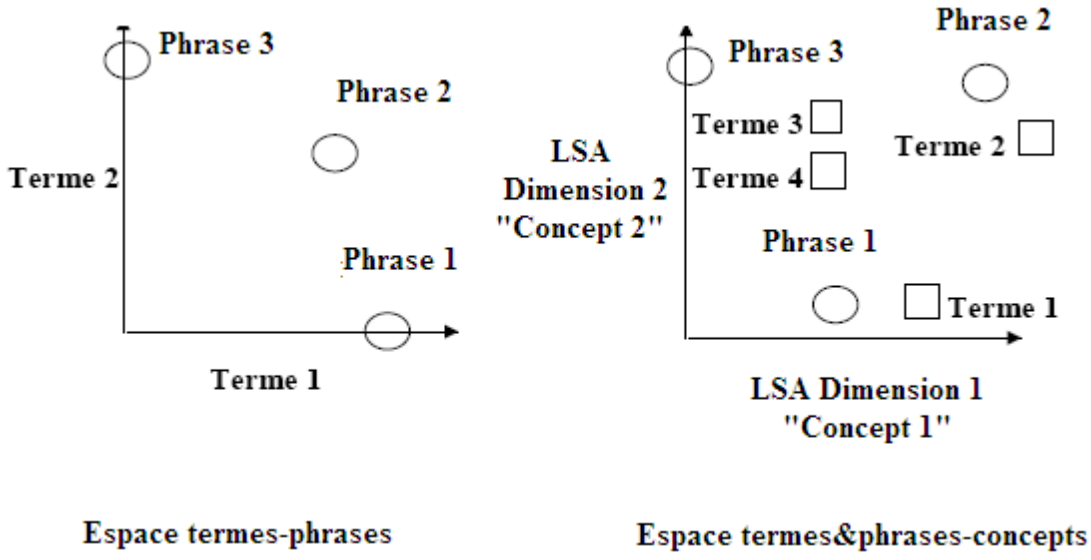


Figure II.4 : Représentation d'un espace terme-phrase & un espace LSA

Dans ce nouvel espace, on peut apprécier les relations sémantiques cachés (latente) entre : les différents termes (concept ou idée), entre les différents phrases (catégorie ou thème) et encore entre les termes et les phrases (moteur de recherche).

Le théorème SVD formule que :

$$A_{nxm} = U_{nxr} S_{rxr} V^{T}_{rxm} \quad \text{Où:} \quad \begin{cases} U^{T}U = \mathbf{I_{nxn}} \\ V^{T}V = \mathbf{I_{mxm}} \quad \text{(U et V sont orthogonaux)} \end{cases} \tag{II.8}$$

Et :

- m : représente le nombre de phrases.
- n : représente le nombre de termes.
- r : est le rang de la matrice A. $r = rank(A) \leq \min(m,n)$ (II.9)

Le calcul de ces matrices se fait comme suit :

Etape 1 : calculer le produit matricielle AA^T et A^TA

Etape 2 : calculer les valeurs propres λ des matrices AA^T et A^TA selon la loi (AA^T- λI=0)

Etape 3 : calculer les valeurs singulières s en calculant la racine des valeurs propres λ.
Etape 4 : calculer les vecteurs propres v de A^TA qui sont la base de la matrice V.
Etape 5 : de même calculer les vecteurs propres de AA^T qui sont la base de la matrice U ou bien calculer U en calculons le produit : $U=AVS^{-1}$.
Après avoir effectué ce calcul, on aura à la fin trois matrices :

- Matrice U de dimension n x r, où n est le nombre de terme.
- Matrice S de dimension r x r.
- Matrice V de dimension r x m, où m est le nombre de phrase.

Par principe, on pourra dire que :

- U: matrice de similitude termes-concept.
- V: matrice de similitude phrases-concept.
- S: ses éléments diagonaux: représentent la «force» de chaque concept [75].

Donc en finale, on est passé d'un espace des similitudes (relations) termes avec les phrases vers un espace des similitudes termes et phrases avec les concepts. Ceci dit, plus le corpus est long, plus le nombre de concept augmente. En pratique, ceci n'est pas intéressant, car nous sommes intéressés par un certain nombre de concepts, les concepts dites parlant ou forts. Donc, nous sommes obligés de diminuer le nombre de concept en utilisant le principe de la réduction de dimension.

C.2. Réduction de dimension

Cette réduction de dimension consiste à diminuer le nombre de concepts à garder. Le critère que nous allons considérer est la force du concept, qui est déjà exprimé par la matrice S.
En d'autres mots, réduire le nombre de concept signifie ne garder que k concepts. Ce qui en résulte à ne garder que les k premiers valeurs singulier de la matrice S. Étant donné qu'ils sont classés par ordre décroissant le long de la diagonale de S et cette ordre est conservée lors de la construction U et V^T. Garder les k premières valeurs singulier est équivalente à garder les k premières lignes de V^T et les k premières colonnes de U. Ceci nous donnera deux matrices orthonormées U_k et V_k et une matrice diagonale S_k. On a alors (Figure II.5).

$$A_k = \cup_k S_k V_k^T \qquad \text{(II.10)}$$

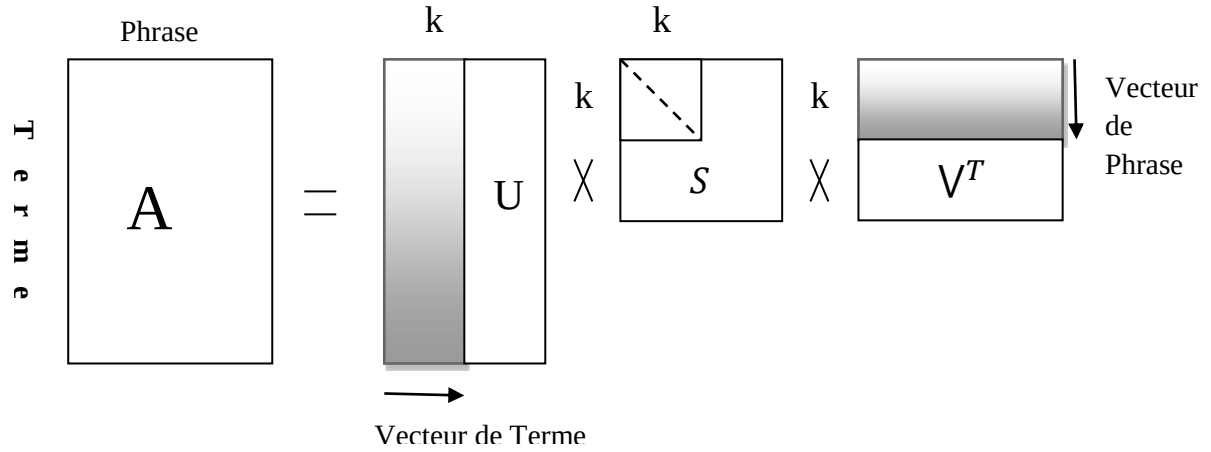


Figure II.5 : La Décomposition en Valeur Singulier 'SVD' de A

Ce processus est appelé la réduction de dimension, et A_k est appelé le rapprochement rang k de A ou la "Réduction SVD" de A. C'est exactement ce qui se fait dans les LSI. Les k premières valeurs singulières sont sélectionnés en tant que moyen pour développer une «sémantique latente » de la représentation de A qui est maintenant libre de dimensions bruyant. Cette représentation «sémantique latente » est une structure de données spécifique dans l'espace de faible dimension dans lequel les phrases et les termes sont intégrées et comparées. Cette structure de données cachées ou «latente» est masqué par des dimensions bruyantes et se manifeste après la SVD.

Algorithme de SVD
• Calcul de A
• Calcul de valeurs singulières et vectrices propres à gauche et à droite.
• Calcul de S, U et V.
• Calcul de « K » : le nombre de valeurs singulières à garder.

D. Aspect pratique

- Étant donné que la LSA s'applique sur la matrice d'occurrence, donc on a décidé de prendre deux matrices. La première est celle tiré du modèle booléen et la seconde du modèle TF-IDF.

- La LSA va nous permettre d'extraire le sens caché en transformant l'espace terme-phrases en deux espaces : Espaces de terme et des phrases en fonction d'une classe LATENTE[19].

- La LSA prendra des mots qui sont sémantiquement similaire mais qui ne l'ont pas dans l'espace vectorielle car il n'apparait pas dans des contextes similaires. Et les projetais dans un espace vectoriel réduit où ils peuvent avoir une grande similarité. (Voir Annexe A) .

[19] Sémantique= sens, Latente = présents mais cachés ⇔ LSA recherche le sens caché

- Les vecteurs de termes sont pris depuis la matrice U. Ces nouveaux vecteurs sont avantageux du fait que :
  - ✓ Deux termes ayant la même idée ont une grande similarité dans l'espace termes-concepts que celui de termes-phrases.
  - ✓ Deux vecteurs ayant une distance proche dans l'espace termes-concepts auront la même idée et donc appartiendront au même concept.
- Les vecteurs de phrases sont pris depuis la matrice V. Ces nouveaux vecteurs sont avantageux du fait que :
  - ✓ Deux phrases ayant le même thème ont une grande similarité dans l'espace phrases-concepts que celui de termes-phrases.
  - ✓ Deux vecteurs ayant une distance proche dans l'espace phrases-concepts auront le même thème et donc appartiendront au même catégorie.

N.B :

On va considérer la nomination de **« concept »** pour le groupement de termes. Idem, pour le groupement de phrases, la nomination de **« catégorie »** est considérée.

On a préféré l'utilisation de la nomination **« terme »** au lieu du **« mot »** dans ce chapitre. Et ceci parce que les termes, par définition, sont les mots ou mots composées ayant un sens particulier dans un *contexte donnée. Encore, le terme est monosomie alors que le mot ne l'est pas. Donc, le terme est directement relié avec le concept, c'est le principe de la terminologie.*

Maintenant, qu'on a réussi à représenter chaque mot par son propre vecteur de représentations. Tous ce qu'il nous reste à faire est de regrouper les différents vecteurs (mots) en un certain nombre de groupement, pour cela nous allons utiliser un algorithme de partitionnement non supervisé K-means. Figure II.6

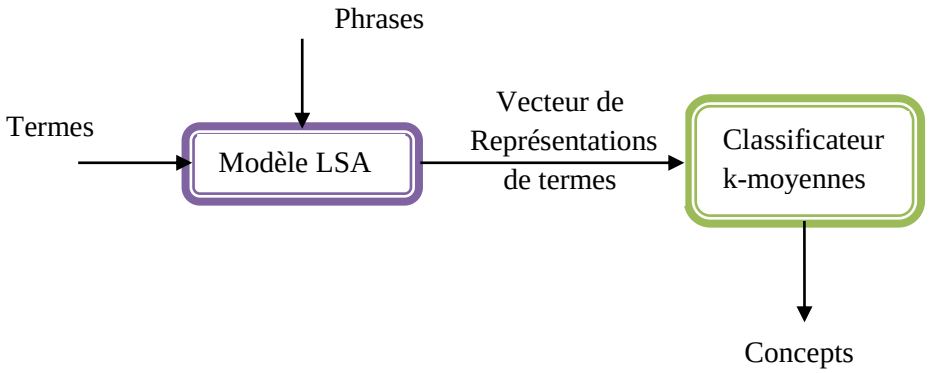


Figure II.6. Analyseur sémantique Conceptuelle

II.2.3.3. Mis au point du classificateur K-means

En statistiques et en apprentissage automatique, l'algorithme K-means permet de partitionner l'espace de représentation des données en K sous-espaces.

Etant donné un ensemble d'observations (x_1, x_2, …, x_n), où chaque observation est un vecteur de dimension d, l'algorithme k-means de regroupement vise à partitionner les n observations dans k sous-ensembles ($k \leq n$) $S = \{S_1, S_2, \ldots, S_k\}$.

Pour notre travail, récupérer les concepts n'est qu'une question de classification des vecteurs de représentations de termes. Notre choix s'est porté vers le classificateur non-supervisé k-moyennes. Ce choix est supporté par deux critères :

- Du fait qu'on vise la généralisation du système de compréhension d'énoncé implique l'affrontement à de nouveaux domaines d'applications. Donc, le besoin d'un classificateur non-supervisé afin d'exploiter ces domaines.
- Un bon classificateurs non-supervisé est celui qui excelle en termes de rapidité de calcule, vitesse de convergence ainsi que le partitionnement des données en groupes.

La classification k-moyennes comporte deux étapes fondamentales : une étape d'initialisation et une étape d'affectation. [76]

Algorithme
Etape d'initialisation :
- **Choisir k vecteur formant les k centroïdes**
Etape d'affectation :
1- Affecter chaque vecteur X_i au groupe C_j de centre G_j telle que $d(X_i, G_j)$ soit minimale. $\mathbf{d}(\boldsymbol{Xi}, \boldsymbol{Gj}) = \sum_{k=1}^{n} \sqrt{(x_{ik} - g_{jk})^2}$
2- Recalculer le centroïde de G_j chaque groupe. G_j est la moyenne du groupe j.
3- Si G_j change, retourne à l'étape 2 sinon STOP.

Bien qu'il puisse être prouvé que la procédure sera toujours fin, l'algorithme des k-moyennes ne signifie pas nécessairement trouver la configuration optimale, la plupart correspondant au minimum de la fonction objectif global. L'algorithme est également très sensible au hasard de la grappe des centres sélectionnés en début. L'algorithme des k-moyennes peut être exécuté plusieurs fois pour réduire cet effet. (Voir Annexe B)

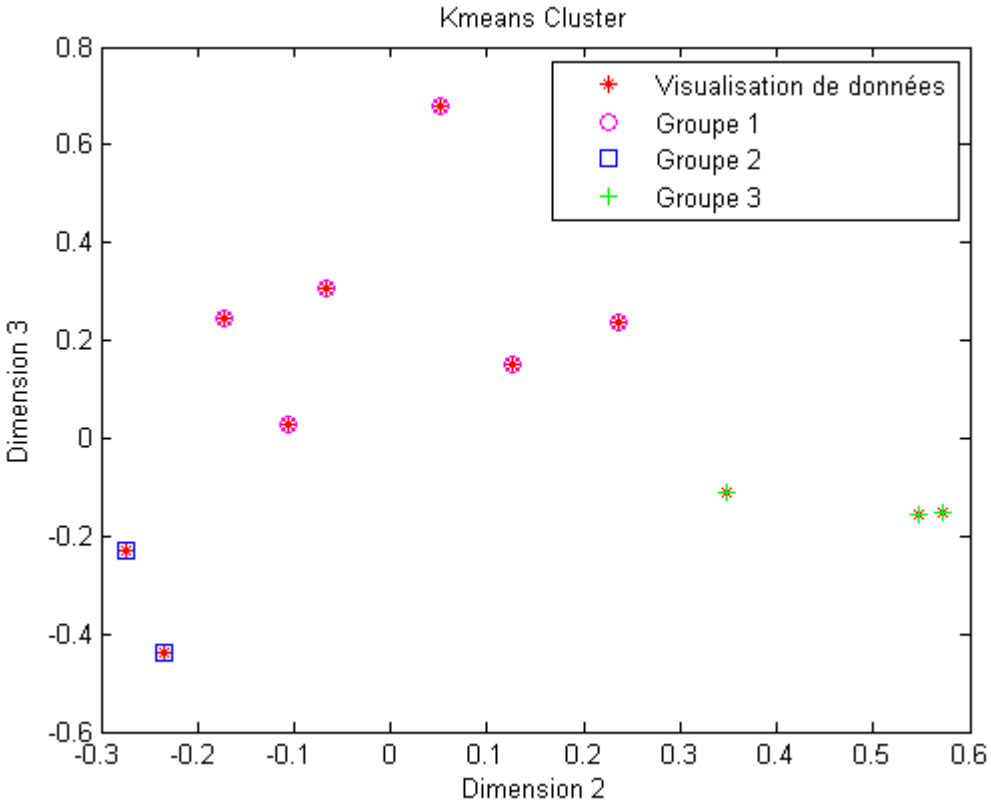


Figure II.7 : Classifications des vecteurs de représentations des termes (LSA_TF-IDF) en trois groupes.

On a considéré dans cet exemple 11 termes à classer en trois groupes. Après avoir effectué une analyse LSA sur la matrice d'occurrence termes-phrases pondérés par le poids TF-idf. On remarque qu'on a une bonne classification car les deux conditions suivantes sont vérifiées :

- La distance intra-classe minimale.
- La distance interclasse maximale.

II.4. Conclusion

On va faire un petit récapitulatif de ce qu'on a fait jusqu'ici :

1- On a pris comme entrée un énoncé bien formé (formel[20]) ; alors qu'en réalité l'entrée du module de l'analyse sémantique n'est autre que la sortie de ce lui de la reconnaissance de la parole (qui n'est pas formel).

2- En prenant comme référence le système cognitif humain qui a tendance à diviser l'énoncé en mots ⇒ **Diviser le texte en phrase** + deviser les phrases en mots.

⇒ **Détection des Frontières de Phrases** + segmentation en mots.

3- On considère qu'il y a des termes non significatifs qu'il faudra filtrées.

4- Maintenant on n'a des mots, il faudra alors trouver les concepts :

* On a recours à une représentation vectorielle, la LSA.
* Regroupement des mots de concepts par le classificateur K-moyenne.

- Désigner un représentant de la classe les mots qui sera le concept.

[20] En mathématiques, logique et informatique, un langage formel est formé :
-D'un ensemble de mots obéissant à des règles logiques strictes (dites grammaire formelle ou syntaxe). -
- D'une sémantique.

Chapitre III :

Système de Compréhension textuel finalisé

Chapitre III :

Système de compréhension textuel finalisé

III.1. Introduction

Ce chapitre est consacré à la description de notre système de compréhension. Il sera dédié à deux applications différentes : Scolarité et Diagnostic Médical.
En premier lieu, nous décrivons tout d'abord l'étape de traduction sémantique qui se résume dans notre cas à une étape d'étiquetage des énoncés par les concepts trouvés précédemment. Ensuite, nous détaillons les différentes étapes nécessaires pour la mise en œuvre d'un module de conversion de représentations. Ce dernier a pour but de convertir les concepts relatifs à une requête en une commande SQL correcte.
Après, nous présentons la dernière étape de notre travail qui consiste à intégrer nos modules de compréhension dans une plate-forme réelle d'interaction textuelle homme-machine. Ceci nous permet d'aboutir à un système de compréhension textuelle finalisé et opérationnel. Enfin, on va appliquer notre système au domaine de la Diagnostic Médical et essayer de quantifier ces performances vis-à-vis de nouvelles applications tout en gardant la même architecture de notre système.

III.2. Le module de compréhension

L'architecture de notre système de compréhension est donnée par la figure III.1 qui montre deux phases dans son fonctionnement semi-autonome de tel système.
Dans la première phase (apprentissage), le but du premier module est d'extraire à partir du corpus textuel la liste de phrases relatives à l'application considérée. Les deuxième et troisième composantes correspondent aux analyseurs lexical et sémantique. L'analyseur lexical permet de segmenter les phrases en termes suivie par un filtrage de ceux non significatifs. L'analyseur sémantique comprend deux sous parties :
La première sert à générer une représentation vectorielle des termes, en utilisant le modèle LSA et la deuxième sert à extraire à partir des vecteurs de termes les concepts relatifs à l'application en question, en utilisant le classificateur non-supervisé k-moyennes.

N.B :
Le caractère Automatique de notre système de compréhension est due en totalité à *l'analyseur sémantique qui sert à trouver les concepts en combinant un* modèle algébrique de représentation suivie par un classificateur non-supervisé.

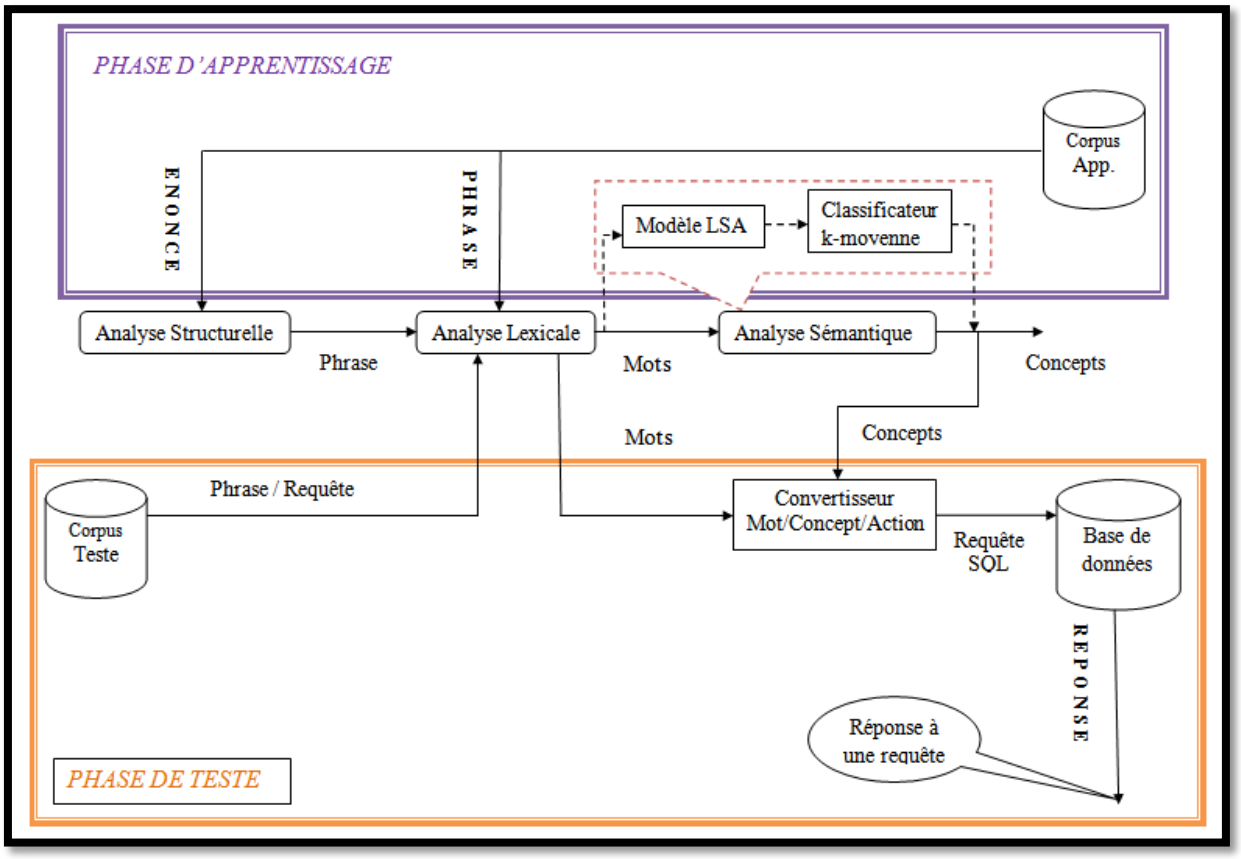


Figure III.1 : Architecture détaillé de notre Système de Compréhension d'Énoncé pour l'application de Scolarité

III.2.1. Description de la phase d'Apprentissage

Nous avons montré précédemment que notre système comprend les analyseurs structurel, lexical et sémantique qui servent en fin de compte à générer les concepts de mots contenue dans le corpus de phrase de notre application.

Les propriétés nécessaires au fonctionnement de ces analyseurs sont:

a- L'analyseur Structurel : suppose que le corpus en entrée n'est constitué que de phrases. Et que ces derniers ne sont que des requêtes entrées par l'utilisateur vis-à-vis de l'application considérée. Enfin, le caractère « \n » saut de ligne est le seul indicateur à la segmentation de texte en phrases.

b- L'analyseur lexical : Utilise le caractère « blanc » comme indicateur à la segmentation des phrases en mots, suivie par une étape de filtrage de données (première réduction de dimension). Lors de cette étape, l'analyseur :

- Ne tient pas compte des caractères alphanumérique, ou il les supprime car insignifiants lors de cette phase.
- Les ponctuations sont supprimées.
- Tous mots ayant une longueur inférieure à trois caractères sont considérés comme mots insignifiant pour notre application, donc supprimés.

- Tous mots figurants dans la liste des mots vides « **stop-words** » de la langue française sont eux même supprimé.

Ces propriétés déjà considérées par cette analyseur, nous ramènera vers la première limitation de notre système à ce stade[1], car il ne tient pas compte des requêtes négatives. On ne peut donc comprend la négation par exemple:

« Je **ne** veux **pas** la note du première examen mais celui du deuxième ».

En sortie de cet analyseur, on aura obtenu la suite des mots :{veux, note, première, examen, deuxième}, ce qui veut dire que le système a tendance à comprendre que l'utilisateur veux sa note du premier et deuxième examen. Or, ce n'est pas exactement ce que demande l'utilisateur.

c- L'analyseur sémantique, fonctionne de la manière suivante :

Lors de la création de la matrice d'occurrence termes-phrases, les mots sont traités sous le même ordre induit par l'analyse précédente. L'utilisation du modèle LSA est appliquée sur une matrice d'occurrence pondérée avec la méthode TF-IDF. En sortie, on ne tiendra compte que de la matrice U représentant les vecteurs de termes où chaque vecteur-colonne de cette matrice représente un terme. Précisant que les vecteurs de termes (matrice U) sont eux-mêmes répartie sous le même ordre des termes, d'où, ceci facilite la distinction des différents pairs vecteur-terme lors de l'étape suivante.

La deuxième étape de cette analyse est basée sur le classificateur k-moyennes. Celui-ci, utilise la distance euclidienne [77].

$$\text{distance}(x,y) = \{\sum_i (x_i - y_i)^2\}^{½} \qquad \text{(III.1)}$$

La méthode d'agrégation[2] utilisée par ce classificateur est celle de la distance des barycentres non pondérée (assez simple, elle consiste à mesurer les distances entre barycentres de clusters) [78]. La qualité des résultats obtenus à partir de ce classificateur est limité par le nombre de cluster (groupe) k choisit au départ.

La méthode des k-moyennes et ses variantes résolvent une tâche dite non supervisée, c'est-à-dire qu'elle ne nécessite aucune information sur les données. La segmentation peut être utile pour découvrir une structure cachée qui permettra d'améliorer les résultats de méthodes d'apprentissage supervisé (classification, estimation, prédiction.

En choisissant une bonne notion de distance, la méthode peut s'appliquer à tout type de données (mêmes textuelles). Cette méthode ne nécessite que peu de transformations sur les données (excepté les normalisations de valeurs numériques), il n'y pas de champ particulier à identifier, les algorithmes sont faciles à implanter et sont, en règle générale, disponibles dans les environnements de fouille de données (data mining).

[1] Ceci sera remédié lors de la phase de test en ce qui suit.
[2] Assemblage de clusters en un.

Les performances de cette méthode (la qualité des groupes constitués) sont dépendantes du choix d'une bonne mesure de similarité ce qui est une tâche délicate surtout lorsque les données sont de types différents.

La méthode est sensible au choix des bons paramètres, en particulier, le choix du nombre k de groupes à constituer. Un mauvais choix de k produit de mauvais résultats. Ce choix peut être fait en essayant plusieurs valeurs de K qui semblent raisonnables. Par exemple, on essaie toutes les valeurs entre 2 et 10, et on choisit celui avec les meilleurs résultats [79].

Il est difficile d'interpréter les résultats produits, i.e. d'attribuer une signification aux groupes constitués. Ceci est général pour les méthodes de classification.

- Dire que le classificateur nous a générer de bon classe en fonction des paramètres déjà choisit (distance, k, ...) est basé sur trois critères :
  - La séparation des classes (distance inter-class maximal).
  - Distance intra-classe minimale.
  - Pouvoir donner une pré-signification[3] aux groupes constitués.
- Assigner les mots au concept approprié (groupe trouvé par le classificateur) en utilisant les pairs mot-vecteur.

Á ce stade-là, la phase d'apprentissage est finie lors de la récupération des différents concepts de mots. Une évaluation des résultats obtenus est nécessaire.

III.2.1.2. Résultats obtenus

Les résultats obtenus à cette étape sont les concepts de mots obtenus à partir du corpus d'entrée dans le cas de l'application de Consultation Scolaire.

A. Corpus d'Apprentissage de l'application de Consultation Scolaire

Nous sommes en train de travailler dans le cadre d'un dialogue H.M. et le corpus en comporte 21 phrases (requêtes). Les cibles de ces requêtes sont soit une recherche de Note, Diplôme ou Certificat de Scolarité. Le critère de choix de collection de ces requêtes en corpus est subjectif, où j'ai essayé en premier lieu de considérer tous les cas possibles (de mon point de vue) pour interroger une base de données de scolarité. En second lieu, je n'ai considéré que des phrases formels syntaxiquement (selon une grammaire) et lexicalement (selon les mots). Ce qui nous emmènera à ne pas tenir compte des autres phrases telles que celle tiré depuis un module de Reconnaissance de la Parole.
La figure suivante montre la liste de phrases considérées lors de cette phase.

[3] Action de signifier à l'avance ; signe indicatif d'un événement futur.

Puis-je avoir mon relevée de note.
Donne-moi mon diplôme.
Puis-je me renseigner sur ma note.
Quelle note ai-je obtenu en examen.
Je souhaite avoir mon certificat de scolarité.
Je veux mon diplôme.
Donnez-moi mon certificat de scolarité.
Puis-je avoir mon diplôme.
Donnez-moi ma note d'examen.
Veux-tu me donner mon certificat STP.
Pouvez-vous me montrer mon relevée de note.
Je voudrais savoir ma note.
J'aimerais bien avoir mon certificat de scolarité.
Pouvez-vous me donner mon diplôme.
Je veux bien avoir la moyenne du module de math.
Je veux savoir le résultat d'examen du physique.
Combien j'ai eu en chimie.
J'aimerais bien avoir ma note du math.
Veuillez me donner la note du module de chimie.
Donne-moi les résultats des modules de math et d'informatique.
Quelle est le résultat d'examen du module d'informatique.

Figure III.2 : Le corpus de l'application de Consultation Scolaire

Á partir de ce corpus de phrase, voici la liste des concepts obtenus manuellement.

Concept	**Liste des mots formant le concept**
Ordre	renseigner, obtenu, souhaite, veux, donnez, donner, pouvez, montrer, voudrais, savoir, aimerais, veuillez
Information	note, certificat, diplôme, relevée, scolarité
Valeur	module, examen, résultat, modules
Module	math, physique, chimie, informatique

Table III.1 : Liste des Concepts Manuelles de l'application de Consultation Scolaire

On a pu obtenir quatre concepts, le concept « ordre » regroupant tous les termes qui expriment une demande ou un ordre. Le concept « information » regroupant les termes liées directement aux taches de l'application, et les concepts « valeur » et « module » qui regroupent des termes liées indirectement aux taches de l'application.

En considérant maintenant notre système, appliquer l'analyseur LSA sur la matrice d'occurrences (de ce corpus) pondérée avec le poids TF-idf, nous fournira les vecteurs de représentations de termes depuis la matrice U. Ces derniers seront regroupés en utilisant le classificateur k-moyennes, les classes obtenues sont représentées dans la table III.2.
En premier lieu, on a gardé le même nombre de concept soit k=4.

K=4 (Booléen)	
Concept	**Liste des mots formant le concept**
Concept n°1	Module, résultat, math, informatique, examen
Concept n°2	Certificat, scolarité, donner, diplôme, donne
Concept n°3	Moyenne, veux, donnez, physique, souhaite, résultats, modules
Concept n°4	Voudrais, aimerais, note, pouvez, renseigner, veuillez, montrer, chimie, savoir, relevée, obtenu

Table III.2 : Liste des Concepts Automatiques de l'application de Consultation Scolaire (K=4), pour une matrice booléenne

Et en considérant une matrice pondérée par la TF-IDF

K=4 (TF-IDF)	
Concept	**Liste des mots formant le concept**
Concept n°1	Moyenne, module, veux, donner, aimerais, physique, note, pouvez, résultat, renseigner, math, informatique, veuillez, montrer, résultats, chimie, examen, relevée, modules, obtenu
Concept n°2	Certificat, scolarité, donner, souhaite
Concept n°3	Voudrais, savoir
Concept n°4	Diplôme, donne

Table III.3 : Liste des Concepts Automatiques de l'application de Consultation Scolaire (K=4), pour une matrice TF-IDF

Avant de valider les résultats obtenus, on remarque au premier coup d'œil qu'une mauvaise classification s'est produite car chevauchement de classe. Un premier alternative d'améliorer les résultats obtenus est de changer le nombre k des concepts, soit k= 2. Ce choix est supporté par l'hypothèse que pour n'importe quelle application d'interrogation d'une base de données, le nombre de champ nécessaire à la requête est de 2. Un champ de « demande » et un champ d' « information ».
Les résultats obtenus sont présentées dans la table III.3.

K=2	
Concept	**Liste des mots formant le concept**
Concept n°1	Voudrais, aimerais, note, pouvez, renseigner, veuillez, montrer, chimie, savoir, examen, relevée, obtenu
Concept n°2	Moyenne, module, certificat, scolarité, veux, donnez, donner, physique, résultat, math, informatique, souhaite, diplôme, résultats, modules, donne

Table III.4 : Liste des Concepts pour l'application de Consultation Scolaire (K=2), pour une matrice booléenne

De même, en considérant une matrice TF-IDF

K=2	
Concept	**Liste des mots formant le concept**
Concept n°1	Diplôme, donne
Concept n°2	informatique, moyenne, module, veux, aimerais, physique, résultat, renseigner, pouvez, math, veuillez, montrer, résultats, chimie, examen, relevée, aimerais, obtenu, note, voudrais, savoir, certificat, scolarité, donner, souhaite, donnez

Table III.5 : Liste des Concepts pour l'application de Consultation Scolaire (K=2), pour une matrice TF-IDF

Maintenant que nous avons ces deux résultats, une étape de validation est nécessaire en considérant comme référence les concepts tirés manuellement.

Dans l'évaluation externe, les résultats du classificateur sont évaluées en fonction des données qui n'ont pas été utilisées pour la classification, tels que les étiquettes de classe connus et des références externes (External Benchmarks [80]). Ces références se composent d'un ensemble d'éléments pré-classifiées, et ces ensembles sont souvent créés par des humains (les experts). Ainsi, les ensembles de référence peut être considéré comme un étalon-or pour l'évaluation. Ces types de méthodes d'évaluation de mesurer à quel point le regroupement est de la classe de référence prédéterminé. L'une des mesures considérant l'évaluation externe est la mesure du taux de reconnaissance des concepts obtenus par le classificateur k-moyenne seulement à l'aide de la mesure F1[4] qui est définis par :

$$F = \frac{2 \times précision \times rappel}{précision + rappel} \quad \text{(III.2)}$$

Dans la tache de la classification, les deux termes « précision » et « rappel » sont définie en fonction des mesures de comparaison externe. Ces mesures sont :

- La vraie positive (true positive TP) : désigne le nombre de résultat correctement présent (nombre de terme correctement classifié).
- La vraie négative (true negative TN) : désigne le nombre de résultat correctement absent.
- Le faux positif (false positive FP) : désigne le nombre de résultat faussement présent (nombre de terme mauvaisement classifié).
- Le faux négatif (false negative FN) : désigne le nombre de résultat faussement absent (missing result).

Les formules de la précision et le rappel :

$$précision = \frac{TP}{TP + FP} \quad \text{(III.3)}$$

$$rappel = \frac{TP}{TP + FN} \quad \text{(III.4)}$$

On considère que le mot a été bien reconnu s'il a été attribué au bon concept.

[4] La F-measure de Van Rijsbergen [81] dite aussi efficacité globale.

Pour le cas de classification multi-classe (N_classe>2), le calcul des mesures précision, rappel et F1 pourrais se faire de deux manières :

- Soit en calculant ces mesures pour chaque classe à part, suivie par un calcul de leurs moyennes (sommes / N_classe).
- Soit de sommer tous les TPs, TNs, FPs et FNs pour toutes les classes, suivies par un simple calcul de mesures précédentes. Cette méthode s'appelle « macro-average ».

Voici les résultats du classificateur, ainsi que la liste des termes significatifs.

% Nombre de Concept est 4
K=4 (booléen):
x = 2 13 15 16 24 0 0 0 0 0
y = 3 4 8 19 27 0 0 0 0 0
z = 1 5 6 10 17 21 26 0 0 0
q = 7 9 11 12 14 18 20 22 23 25 28
K=4 (tf-idf) :
x = 1 2 5 8 9 10 11 12 13 14 15 16 18 20 21 22 24 25 26 28
y = 3 4 6 17 0 0 0 0 0 0
z = 7 23 0 0 0 0 0 0 0 0
q = 19 27 0 0 0 0 0 0 0 0

Nombre de conccept égale à 2 :
K=2 (booléen) :
x = 7 9 11 12 14 18 20 22 23 24 25 28
y = 1 2 3 4 5 6 8 10 13 15 16 17 19 21 26 27
k=2 (tf-idf) :
x = 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 20 21 22 23 24 25 26 28
y = 19 27 0 0 0 0 0 0 0 0

1-moyenne, 2-module, 3-certificat, 4-scolarite, 5-veux, 6-donnez, 7-voudrais, 8-donner, 9-aimerais, 10-physique, 11-note, 12-pouvez, 13-resultat, 14-renseigner, 15-math, 16-informatique, 17-souhaite, 18-veuillez, 19-diplôme, 20-montrer, 21-resultats, 22-chimie, 23-savoir, 24-examen, 25-relevee, 26-modules, 27-donne, 28-obtenu

Figure III.3 : La liste des termes significatifs et les résultats du classificateur.

x, y, z et q sont les classes récupérées auprès du classificateur. Chaque numéro dans ces classes indique un terme dans la liste en dessous.

Voici un extrait du calcul de mesures précédentes :

```
# Classe 1: Concept 1
y1=[5, 6, 7, 8, 9, 12, 14, 17, 18, 20, 23, 28]
pred1= [ 7,9 ,  11,  12,  14,  18,  20,  22 ,  23 ,  25,  28]
TP1=7 # (true positive) Correct result 7,9,12,14,18,23,28
FN1=4 # (false negative) Missing result 5,6,8,20
FP1=4  # (false positive) Unexpected result 11,20,22,25
TN1=13 # (true negative) Correct absence of result 2,11,13,15,16,19,21,24,26,27
pre1= TP1/(TP1+FP1)
rec1= TP1/(TP1+FN1)
fscore1= 2* pre1 *rec1 /(pre1+rec1)
```

Figure III.4 : Un extrait du calcul des mesures

Voici les résultats des mesures de validations obtenues pour k=4 et k=2 pour les deux types de matrices. Ces résultats contiennent les précisions, rappels et F1 pour chaque classe ainsi que pour toutes les classes.

Résultats:
Concept1 precision: 0.636363636364 Concept3 precision: 0.6
Concept1 rappel: 0.636363636364 Concept3 rappel: 0.6
Concept1 F-measure: 0.636363636364 Concept3 F-measure: 0.6

Concept2 precision: 0.75 Concept4 precision: 0.142857142857
Concept2 rappel: 0.75 Concept4 rappel: 0.25
Concept2 F-measure: 0.75 Concept4 F-measure: 0.181818

ConceptAll micro_avg_precision: 0.518518518519
ConceptAll micro_avg_rappel: 0.583333
ConceptAll micro_avg_F-measure: 0.549019607843

Figure III.5 : Les mesures de validations pour k=4 (booléen)

Donc en utilisant une matrice booléenne avec k=4 concept, on est en mesure d'atteindre 54.90% de bonne classification. Et pour la matrice TF-IDF :

Résultats:
Concept1 precision: 0.4 Concept3 precision: 0.0
Concept1 rappel: 0.666666 Concept3 rappel: 0.0
Concept1 F-measure: 0.5 Concept3 F-measure: 0.0

Concept2 precision: 0.5 Concept4 precision: 0.0
Concept2 rappel: 0.4 Concept4 rappel: 0.0
Concept2 F-measure: 0.44444 Concept4 F-measure: 0

ConceptAll micro_avg_precision: 0.357142857143
ConceptAll micro_avg_rappel: 0.4
ConceptAll micro_avg_F-measure: 0.377358490566

Figure III.6 : Les mesures de validations pour k=4 (TF-IDF)

On remarque que les taux de classifications s'est dégradé jusqu'à les 37.73% (qui n'est pas très encourageant). Maintenant, on va voir si on peut améliorer ces taux de classifications en considérant le nombre de k=2. Voici les résultats pour la matrice booléenne.

Concept1 precision: 0.166666666667 Concept3 precision: 0.75
Concept1 rappel: 0.285714285714 Concept3 rappel: 0.545454
Concept1 F-measure: 0.210526315789 Concept3 F-measure: 0.631578947368

ConceptAll micro_avg_precision: 0.5
ConceptAll micro_avg_rappel: 0.48275862069
ConceptAll micro_avg_F-measure: 0.491228070175

Figure III.7 : Les mesures de validations pour k=2 (Booléen)

On n'y voit bien que le taux de classification s'est amélioré pour atteindre les 49.12%. Allons-nous examiner les résultats obtenus avec la matrice TF-IDF.

Résultats:
Concept1 precision: 0.5 Concept3 precision: 0.807692307692
Concept1 rappel: 0.142857142857 Concept3 rappel: 0.954545454545
Concept1 F-measure: 0.22222222 Concept3 F-measure: 0.875

ConceptAll micro_avg_precision: 0.785714285714
ConceptAll micro_avg_ rappel: 0.758620689655
ConceptAll micro_avg_F-measure: 0.771929824561

Figure III.8 : Les mesures de validations pour k=2 (TF-IDF)

On remarque que le taux de classification est au-dessus de celui précédent, avec un pourcentage de 77.19% (qui est le meilleur taux jusqu'ici).
Dans le récapitulatif suivant, on va considérer les mesures de validations « micro average ». Du fait qu'elle exprime le taux de classification totale (pour toutes les classes).

Méthode	Extraction de Concept			
Application	Scolarité			
Modèle	LSA + k-moyenne			
Type de matrice	Booléenne		TF-IDF	
Paramètres	K=4	K=2	K=4	K=2
Précision (%)	51.58	50	35.71	78.75
Rappel (%)	58.33	48.27	40	75.86
F1[5] (%)	54.90	49.12	37.73	77.19

Table III.4 : Résultats obtenus par la méthode Extraction de Concepts avec l'application de Consultation Scolaire

On remarque que pour l'application de scolarité, on a un taux de classification de l'ordre de 77.19% avec k=2 pour une matrice TF-IDF. Mais tous de même on va travailler avec les concepts obtenus avec la valeur de k=4[6] où le taux vaut 54.90% en utilisant la matrice booléenne. Et ceci est dû aux spécificités de cette application d'interrogation de base de données, qui sont le non possibilité de répondre à une requête ou la réalisation d'une tache avec seulement deux champs (concepts). Néanmoins, pas pour tous les cas possibles, ce qui nous s'en sorte du langage naturel.
Prenons ces deux exemples d'une même requête en langage naturel, formulé différemment :

1- Je veux ma note.
2- Je veux ma note de math.

[5] La F-measure de Van Rijsbergen [81] dite aussi efficacité globale,
[6] On a considéré quatre concepts pour être compatible avec une requête SQL.

En considérant seulement deux concepts (k=2), on pourrait répondre à la première requête mais pas à la deuxième (car on aurait trois cas possibles soit : « veux note », « veux math » ou « note math »).
Donc plus on a de concept[7], plus on peut résoudre n'importe quelle évènement.
Enfin, puisque au début de ce travail, on a supposé une réduction de l'influence de l'expertise humaine, donc c'est ici qu'on va intervenir pour corriger les concepts (car on ne pourra pas avoir un taux de classification de 100%, donc une correction des concepts ne serait pas très exigeante au terme de ressource humaine). Cette correction est nécessaire pour réaliser un bon système de compréhension (continuer avec des concepts non homogène engendra un système non performant).

III.2.2. Phase de test

C'est au cours de cette phase que l'interprétation des requêtes est effectuée (Convertisseur Mot/Concept). En effet, disposant de l'ensemble des concepts qui régissent l'application, nous pouvons attribuer à chaque requête les concepts appropriés. En temps normal, pour montrer la validité des concepts obtenus par les méthodes de classification non supervisée, on aurais dû les utiliser à la place des concepts établis manuellement, mais tous de même, étant donné que le but de départ été de réaliser un système de compréhension de l'énoncé indépendant de l'application mais pas totalement autonome, on s'y permet de corriger un peu les concepts déjà obtenu par le classificateur, en adoptant la représentation vectorielle LSA, de telle manière qu'ils soient interprétable. Ceci dit, minimiser les chevauchements interclasse, nous permet de maximiser les performances de notre système de compréhension.
On considère toujours l'application de Consultation Scolaire. Les listes de concepts relatives à cette application sont données dans le Table-III.1. Les concepts ne présentent aucune répétition ni chevauchement de mots. L'étape d'étiquetage dans ce cas peut être réalisée d'une manière déterministe en attribuant tout simplement un concept par terme. Autrement dit, on remplacera chaque terme par le concept auquel il appartient.
Le seul problème qui reste est celui des mots hors-vocabulaire. Puisque ces mots n'appartiennent à aucun concept, ils ne peuvent pas participer à l'étiquetage des requêtes (passage d'une suite de terme « langage naturel » à la suite de concept « langage conceptuelle »).
Ceci sera considérer plus loin dans ce chapitre.

III.2.2.1 Construction des requêtes SQL

Ce module a pour objectif de construire la requête SQL correspondante à la phrase initiale étant donnée la liste des concepts appropriés. Il est clair que la génération des requêtes SQL est une tâche spécifique à l'application et à la base de données exploitée. Par conséquent, le traitement sous-jacent à cette tâche ne peut être automatisé.[8]

[7] Pour le choix du nombre de concept idéal, voir chapitre 4.
[8] Ce module est propre à une application donnée, donc impossibilité de le généraliser.

Ce problème peut se traité en deux étapes: une étape de représentation générique assuré par un moteur d'inférence et une étape de génération des requêtes SQL.

III.2.2.1.1. Représentation générique

La fonction principale du module de représentation générique est de transformer la représentation conceptuelle de la phrase en une représentation proche d'une requête SQL. En effet, il s'agit d'une requête SQL dont les objets ne sont pas encore identifiés. Ainsi, dans cette étape on génère une requête du style :

select Note
from users_note
where inscription;

Les objets en italiques de cette requête ne sont pas encore instanciés à ce stade. Ils correspondent aux différents concepts sémantiques qui représentent la phrase de départ. Le travail réalisé dans cette étape consiste à placer correctement ces différents concepts dans la requête SQL dite générique. Dans cet exemple, il s'agit d'une phrase qui a été étiquetée par les concepts {Note, users_note, Inscription} de l'application Scolarité. La requête générique présentée indique par exemple qu'il y a une condition sur le code d'inscription mais qu'on ne connaît pas encore sa valeur.
La génération de cette forme de requêtes est assurée par un moteur d'inférence. Ce dernier est relatif à la liste des concepts de l'application traitée. Le fait d'organiser les concepts obtenus en hiérarchie peut faciliter considérablement le travail de ce moteur.
En effet, plus les concepts respectent la structure d'une requête SQL plus la construction des requêtes génériques sont simplifiée.
Dans la suite, nous présentons tout d'abord la hiérarchie des concepts obtenus pour l'application Scolarité. Ensuite, nous exposons la structure de la base de données exploitée. Enfin, nous décrivons le moteur d'inférence utilisé pour la construction des requêtes génériques.

a- Hiérarchie des concepts

Les concepts obtenus peuvent être organisés en une structure d'arbre. Dans notre cas, cette structure doit ressembler le plus possible à celle d'une requête SQL. Ceci permet de faciliter le travail de conversion de représentation. La figure III.9 représente la hiérarchisation des concepts relatifs à l'application Scolarité. Dans cette figure, les concepts sont donnés au niveau des feuilles de l'arborescence (donnés en gras italique).
Les autres niveaux correspondent à des degrés d'abstraction supérieurs. Ces derniers ont été conçus manuellement dans le but de retrouver la structure qui correspond le plus à celle des requêtes SQL en groupant ensemble les concepts qui se ressemblent. La correspondance entre concepts et requête SQL est donnée en bas de la figure.

Nous distinguons quatre niveaux de hiérarchisation. Le sommet de l'arbre qui correspond à la totalité de la requête. Ensuite, au deuxième niveau, nous divisons la requête en deux parties : l'introduction et le corps de la requête. La première partie correspond à tout ce qui concerne le début des phrases : demande, formules de politesse, souhaits, actions à faire, etc. La deuxième partie contient la demande proprement dite c'est-à-dire l'objet recherché et les conditions qui lui sont relatives. Dans une requête SQL nous pouvons trouver trois composantes principales qui sont : l'objet recherché, les tables à consulter et les conditions. Nous reprenons cette même décomposition et nous définissons donc trois sous niveaux supplémentaires pour représenter chacune de ces trois composantes. Ainsi, nous construisons la hiérarchie sous laquelle nous devons organiser nos concepts et ce quel que soit l'application traitée. Le fait d'adopter cette même hiérarchie pour toutes les applications envisagées nous permet d'établir un traitement automatique pour la construction des requêtes génériques. Il suffit ensuite de changer le nombre et les noms des concepts relatifs à chaque arborescence pour basculer d'une application à une autre.

Chaque hiérarchie est composée de quatre parties correspondantes à celles d'une commande SQL. La première partie correspond à la formulation d'une demande qu'on représente avec le mot clé “Select”. Ensuite, nous retrouvons les concepts qui expriment l'objet demandé ainsi que quelques critères de choix qui permettent de mieux le cerner. Ces critères peuvent être employés pour identifier les tables à consulter. Enfin, la dernière partie correspond aux conditions que doit vérifier l'objet demandé. Elles correspondent tout simplement à la partie “**Condition**” dans une requête SQL. Il est à noter que les concepts utilisés pour identifier les tables à interroger peuvent aussi figurer comme des conditions dans la requête SQL finale [6].

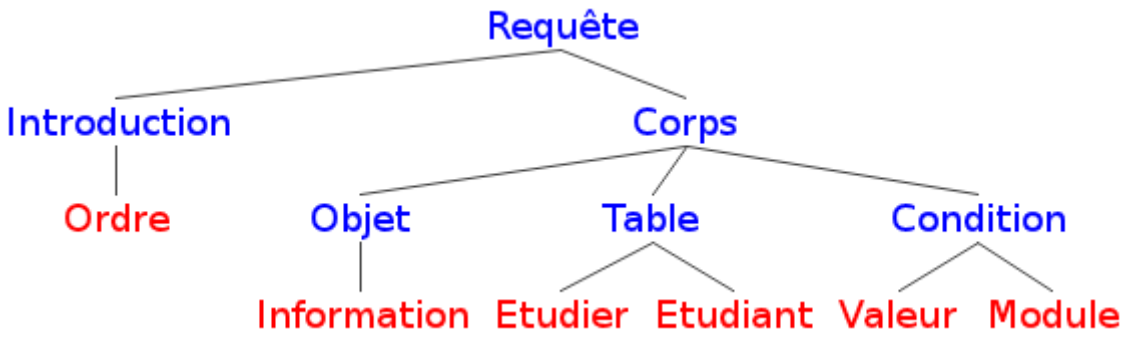


Figure III.9 : Hiérarchie des concepts obtenus pour l'application Scolarité

b- Etiquetage de requête

L'étiquetage consiste à passer du langage naturel au langage de concept. Un exemple expliquant le principe de l'étiquetage :

	Requête 1	Requête 2
La requête en langage naturel	Je **voudrais** avoir mes **notes**	**Donner** moi ma **note**
Les termes	voudrais, notes	donner, note
Les concepts propres à chaque terme => La requête en langage conceptuel	Introduction Information	Introduction Information

Table III.5. Exemples d'étiquetage de requête

On voit bien à partir de ces exemples, que deux requêtes différentes ayant la même idée ont été étiquetées avec la même requête en langage conceptuel. Ceci engendre une autre réduction de dimension (après filtrage et LSA) sans perdre aucune information. Afin de réaliser cette tâche d'étiquetage, une grammaire est nécessaire. Voici la grammaire qu'on a considérée afin de concevoir cette hiérarchie de concept et ceci en analysant les différentes phrases (requête) du corpus. La grammaire formelle **G** est définit par le quadruplet {V_N, V_T, R, P) tel que:

V_N→ {A, B, C, D, E}

V_T→ {montrer, indiquer, sélectionner, trouver, donner, afficher, présenter, prendre, passer, chercher, souhaite, faut, désire, désirerais, peux, pourrais, veux, voudrais, possible, aimerais, souhaiterais, note, certificat, diplôme, module, examen, test, électronique générale, électronique numérique, traitement de signal, électricité, mesure, champ, antenne, capteur, réseaux, fibre optique, télévision, système de télécommunication.}

R = P

P→AB | C | D | E

B→C | C D | C D E | C E

A→ montrer, indiquer, sélectionner, trouver, donner, afficher, présenter, prendre, passer, chercher, souhaite, faut, désire, désirerais, peux, pourrais, veux, voudrais, possible, aimerais, souhaiterais.

C→ note, certificat, diplôme.

D→ module, examen, test.

E→électronique générale, électronique numérique, traitement de signal, électricité, mesure, champ, antenne, capteur, réseaux, fibre optique, télévision, système de télécommunication.

Figure III.10 : Grammaire considérés par l'application de Consultation Scolaire

Où : A : Concept « Ordre », C : Concept « Information »
D : Concept « Valeur », E : Concept « Module »

Cette grammaire est le fruit des grammaires G1, G2 et G3 propres aux classes Note, Certificat et Diplôme.

- La grammaire de G1 de la Note, considère les mêmes règles que G sauf que pour V_T et C on ne trouve pas les terminaux « certificat, scolarité et diplôme ».
- La grammaire G2 du Certificat, assigne quelques modifications à G :
  1. $V_N \rightarrow$ {A, B, C} et C ne contient que le terme « certificat ».
  2. P→AC|C.
- La grammaire G3 du Diplôme, considère les mêmes règles que G2 sauf que pour C il ne contiendra maintenant que le terme « Diplôme ».

Donc pour affiner cette grammaire pour qu'elle supporte autant de tache possible pour l'application de scolarité, il suffit d'ajouter d'autres grammaires (nouvelles règles).

D'une manière plus illustrative, supposons que nous ayons la phrase en entrée suivante :

Je veux ma note du module de champ.

Etiqueter cette phrase consiste à appliquer une analyse descendante, pour savoir à qu'elle langage (grammaire) elle appartient. Et par la suite lui attribuer les concepts trouvés lors de cette analyse (Cette suite de concept correspond exactement à l'idée exprimé par la classe possédant la grammaire en question).

Donc cette phrase sera étiquetée par les concepts « A, C, D, E » respectivement « Ordre, Information, Valeur, Module ». Ceci permet d'obtenir l'allure générale de la requête générique correspondante à la phrase d'entrée. Afin de construire la requête finale, une connaissance de la structure de la base de données exploitée est nécessaire. Ceci est expliqué plus en détail dans ce qui suit.

c- Bases de données exploitées

Le corpus de l'application Scolarité concerne la consultation d'une base de données de scolarité. La structure de la base de données considérée contient donc des informations concernant les notes, l'attestation et le diplôme. La structure de cette base de données est exposée dans la figure III.11. Elle met en évidence les six tables suivantes :

- Etudiants (inscription, nom, date de naissance, code degré*, année_obtention*)
- Degré (code_degré, désignation)
- Module (code_module, designation)
- Etudier (inscription, code_module ,code_sp , note)
- Spécialité (Code_sp, désignation)
- Année (année_scolaire)

La table Etudiant a pour but d'identifier chaque étudiant de la base par un nom, une date de naissance, un matricule et une année. Celle du Degré consiste à donner une désignation à un statut (ex : 001->Ingénieur). De même, la table Module affecte à chaque module un code

spécial (ex : 006->Traitement de signal). Encore, la table Spécialité, elle aussi attribue à chaque spécialité un code unique. La table de l'année contribue l'année scolaire. Enfin, la table Etudier a pour but d'associer à chaque étudiant une ou plusieurs notes par module.

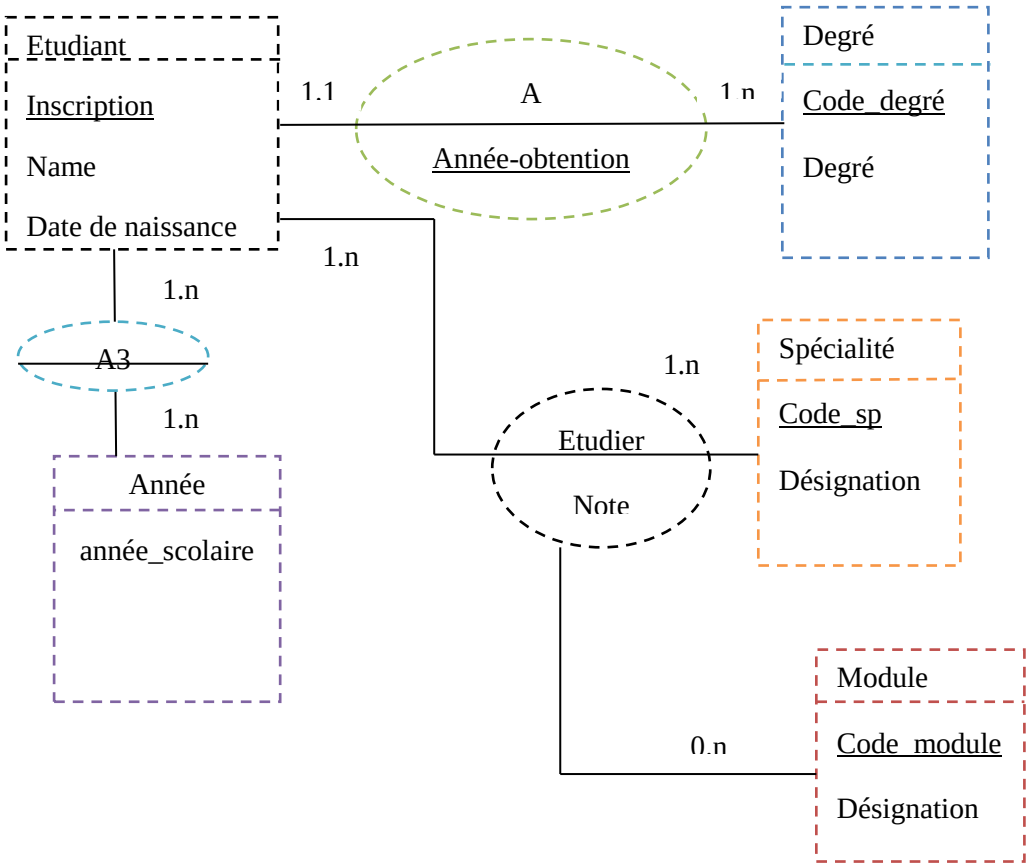


Figure III.11 : Structure de Base de données de notre application de Scolarité

Dans cette structure, on trouve les tables ainsi que leurs relations respectives. Ces relations sont annotées par les termes : (1.1), (1.n) et (0.n). Ces derniers souvent appelées cardinalités ou fréquences, expriment le nombre de cas possibles allant du minimum au maximum.
Par exemple la liaison A3 entre la table Année et la table Etudiant est annotée par (1.1) dans les deux sens. Ceci veut dire qu'au minimum un seul étudiant (maximum n étudiant) étudie dans l'année scolaire courant (le sens de lecture de haut en bas). Inversement au minimum dans une seule année scolaire (maximum n année), on trouve l'étudiant sélectionné.

d- Moteur d'inférence

Pour construire les requêtes génériques, un mécanisme d'inférence a été mis en place. Un moteur d'inférence permet aux systèmes experts de conduire des raisonnements logiques et de dériver des conclusions à partir d'une base de faits et d'une base de connaissances. Les bases de faits sont les concepts présents et celle de connaissances représentes l'ensemble de règles reliant une requête conceptuelle et la requête générique. En d'autre terme, ce moteur nous permet de passer d'une requête conceptuelle en une autre générique.

Donc, il s'agit tout d'abord d'identifier les concepts présents dans la phrase et faire leur mise en correspondance avec la hiérarchie des concepts. Tous les concepts exprimant une de-

mande sont d'abord remplacés par le mot clé « **select** ». Ensuite, les parties « **Objet** » et « **Table** » de la requête sont remplacés par les différents concepts qui les représentent. Dans ce cas, quelques concepts nécessitent d'affiner encore la requête. Examiner les concepts de la partie « Conditions » sert pour ajouter si nécessaire d'autres tables et les conditions de jointures correspondantes. Sinon, la consultation est limitée à une seule table avec des conditions internes sur ses attributs seulement.

Exemple :

Je veux ma note → Introduction Objet → select Information

La condition interne dans ce cas est le matricule d'inscription.

On a essayé de mettre en œuvre un moteur d'inférence le plus indépendant possible de l'application. Ce moteur sera paramétré par les concepts même si certains ajustements sont dépendants de l'application. Ainsi, l'automatisation de cette tâche s'est révélée possible grâce à la hiérarchie des concepts qui a été construite en respectant la structure d'une requête SQL. La table III.6 montre un exemple de requête générique simple obtenue avec l'application Scolarité. Dans certains cas, l'obtention des concepts représentatifs de chacune des parties d'une requête SQL n'est pas possible. Ceci peut être dû à une mauvaise identification des concepts ou alors à des phrases incomplètes où beaucoup d'information reste implicite. Par exemple, si nous ne trouvons pas de concepts relatifs à la partie « Objet », nous la remplaçons par « * » et on aura donc un « select * from Table » au lieu de « select Objet from Table ».

Phrase de départ	Je veux ma note du module de champ
Etiquetage	Ordre, Information, Valeur, Module
Requête générique	**select** Information **from** Etudier **where** Module

Table III.6 : Exemple d'une requête générique obtenue avec l'application Scolarité

III.2.2.2. Génération des requêtes SQL

La génération des requêtes SQL constitue la dernière étape du processus de compréhension. Elle consiste à utiliser la phrase de départ ainsi que la requête générique afin d'instancier les valeurs des différents concepts présents dans cette dernière. Ceci est réalisé à l'aide d'un mécanisme de mise en correspondance.

L'implémentation de ce mécanisme de mise en correspondance est spécifique à chaque application. Il est donc quasiment impossible d'automatiser cette tâche. Au cours de cette étape, nous revenons à la phrase initiale pour associer à chaque concept de la requête générique la valeur exacte permettant son interprétation par un SGBD. Ceci est fait concept par concept. Les conditions sont aussi traitées cas par cas. De plus, quelques cas particuliers sont

toujours à envisager : par exemple, le manque d'informations dans les phrases initiales ou la structuration incohérente d'une requête générique, etc.

Comme nous construisons des requêtes SQL relatives à des bases de données spécifiques, nous avons toujours besoin de retourner à ces bases pour connaître les noms des tables et leurs attributs. En fait, en plus de la requête générique et de la phrase de départ, nous avons aussi besoin de tenir compte d'un troisième paramètre très important qui est la base de données à interroger. La mise en correspondance faite entre les deux premiers n'est pas suffisante pour générer une requête SQL correcte, il faut aussi trouver leur correspondance dans la structure de la base de données considérée.

Après avoir généré la requête SQL relative à une phrase en entrée, il suffit de l'exécuter par un **système de gestion de base de données** SGBD pour obtenir un résultat final. Dans la table III.7 l'exemple présenté dans la section précédente (dans la table III.6) sera complété en donnant les requêtes SQL finales obtenues dans l'application Scolarité.

La phrase	Je **voudrais** bien avoir ma **note** du **module** de **champ**
La suite de concept	**Ordre**, **Information**, **Valeur**, **Module**
Requête générique	**select Information** from **Etudier** where **module**
Requête SQL	**select note** from **Etudier** where **inscription&module**

Table III.7 : Exemple d'une requête générique et SQL obtenues avec l'application Scolarité

III.2.3. L'intégration dans une plate-forme réelle

La dernière étape de notre travail consiste à intégrer les modules de notre système de compréhension tournant autour de l'application Scolarité dans une plate-forme réelle d'interaction textuelle homme-machine : le cas d'une interface graphique GUI comportant un simple champ textuel, invitant l'utilisateur à saisir sa requête (voir annexe F).

Notre système de compréhension d'énoncé est exploitable et opérationnel (des captures d'écrans ainsi qu'un test est présenté en annexe F). En effet, il permet de prendre un texte en entrée et de délivrer en sortie une requête SQL. On considère que la compréhension a abouti lorsque la requête SQL générée satisfait la demande saisis en entrée. Autrement dit, si elle est exécutée par un SGBD, la réponse obtenue doit répondre exactement à ce qui a été demandé par le locuteur.

Les avantages de cette plateforme est son « easy to use » où l'utilisateur n'aura qu'à saisir sa requête dans le champ textuel et cliquer sur entrée. Si la compréhension s'y bien faite, l'utilisateur aura la possibilité de dialoguer encore avec le système dépendamment de la requête formulé.

L'inconvénient de cette plateforme est dû essentiellement au fait qu'elle ne peut être lancé qu'à partir une machine ayant le Python et le NLTK installé. Encore, il y a le manque d'une

interface vocale à l'entrée qui aurait pu ajouter de plus à ce système. Ces deux problèmes seront considérés comme future perspectives à short terme.

III.2.3.1. Gestionnaire des termes hors-vocabulaire

Il sert à attribuer un sens aux nouveaux termes (inexistants auparavant dans le corpus de mots).

Pour se faire, rappelons que les concepts de termes ont un ordre bien précis « A, C, D, E », ce qui est assurée par la grammaire G (de telle manière que seulement les phrases syntaxiquement correctes « appartenant à la grammaire en question » vont être traité par le moteur d'inférence ainsi que ce gestionnaire). Donc ce dernier aura quatre cas à résoudre, selon le mot nouveau w' et les concepts C, D, E :

- 1ére cas : « w' C D E », si on trouve le terme dans un contexte où il est en début de la phrase suivie par les trois autres concepts, on déduira que l'action voulue est une demande de Note d'un Module.
- 2éme cas : « w' C D », l'action voulue est soit une demande de Note ou Diplôme.
- 3éme cas : « w' C E », aussi la Note d'un Module
- 4éme cas : « w' C », idem
- 5éme cas : « w' D E », la Note d'un Module
- 6éme cas : « w' D », la Note
- 7éme cas : « w' E », la Note d'un Module

Le Table suivant explique clairement l'idée derrière ce gestionnaire (le ou les termes en rouges sont méconnus à notre système):

Cas	Requête en langage naturel	Etiquetage	Requête en Concept Proche	Information Considéré
1	j'espère avoir ma **note** du **module** de **champ**	W'**CDE**	A**CDE**	La Note du module de champ
2	j'espère avoir ma **note** d'**examen**	W'**CD**	A**CD**	La Note d'un examen
3	j'espère avoir ma **note** du **champ**	W'**CE**	A**CE**	La Note du champ
4	j'espère avoir ma **note**	W'**C**	A**C**	La Note
5	combien j'ai eu en **module** de **champ**	W'**DE**	A**DE**	La Note du module de champ
6	combien j'ai eu en **examen**	W'**D**	A**D**	La Note d'un examen
7	combien j'ai eu en **champ**	W'**E**	A**E**	La Note du champ

Table III.8 : Exemplaire de la Gestion des termes hors-vocabulaire

C'est 7 cas sont propre à ce gestionnaire et à cette application, car ils sont extraits à partir d'une analyse des requêtes (phrases) ayant des termes nouveaux. Cette analyse est tirée d'une simple déduction à une simple question :

Lequel des concepts existants, pourra prendre la place de ce terme sans déformer l'idée de base?

Pour répondre à cette question, il suffit d'essayer tous les concepts un à un et garder celui le plus relevant. En répétant cette opération avec tous les cas possible on aura le gestionnaire de termes hors vocabulaire.

La tâche du gestionnaire est réalisée en deux étapes :

- une première étape d'étiquetage de requête : le nouvel terme sera détecté à ce niveau où l'étiqueteur va le laisser tel qu'il est, et passera au terme suivant jusqu'à la fin de la requête. Cela aura pour effet de garder l'ordre des termes (contexte). Donc, en sortie on aura nos concepts ainsi qu'un nouvel terme dans un contexte donnée.
- La deuxième étape consiste à attribuer cette nouvelle requête à l'une des sept cas possibles en considérant le contexte du nouvel terme. Après on aura directement l'information considérée et depuis la requête SQL appropriée.

III.2.4. Pré-conclusion

Dans les sections précédentes, on a pu voir et définir tous les composants d'un système SyCE-CHM et ce en deux phases :

- Phase d'apprentissage, consistant à extraire les concepts appropriés à une application donnée.
- Une phase de test, qui s'intéresse plutôt à la façon d'aboutir au sens de la requête entré par l'utilisateur et ce en utilisant ces concepts pour traduire cette requête du langage naturel vers le langage SQL (compréhensible par le SGBD) en passant par une représentation intermédiaire (langage des concepts).Comme mentionné plutôt, on considère que la compréhension est concluante, quand l'action générer par le SGBD correspond à la demande de l'utilisateur.

Pour donner plus de poids à notre système, on a voulue mesurer ses aptitudes en termes de portabilité et d'adaptabilité vis-à-vis d'autre domaine d'application. Pour cela, nous allons utiliser ce système pour une nouvelle application. Cette dernière permet de diagnostiqué une maladie et va attribuer un traitement propre.

III.3. Les systèmes utilisées en Diagnostic Médical

Comme tout programme destiné à servir de consultant aux médecins, ils doivent avoir un capital de connaissances Médicales, exprimées en termes de descriptions de maladies possibles. Selon l'étendue du domaine clinique, le nombre d'hypothèses dans la base de données peut varier. Dans la représentation la plus simple, chaque hypothèse de maladie identifie l'ensemble des disfonctionnements des sujets. La version la plus simple de ces programmes fonctionne de la façon suivante :

1. Pour chaque maladie diagnostiquée, il faut vérifier si les résultats donnés sont prévisibles.
2. Évaluer ce système en comparant les résultats donnés par rapport aux résultats connus à l'avance.
3. Hiérarchiser les maladies diagnostiquées en fonction de leurs scores.

Le système diagnostiquant les maladies osseuses comprend trois grandes parties: une interface utilisateur (IU), un système de gestion de base de données (SGBD), et un système expert (ES). Le SGBD est utilisé pour la manipulation des données des patients différents. Un certain nombre de cas de patients sont sélectionnés en tant que prototype et stockés dans une base de données. Le diagnostic est effectué par l'ES, appelé XBONE, fondées sur des données patient, la base de connaissances (KB) contient les connaissances heuristiques pour le diagnostic des maladies des os. La connaissance est représentée par un formalisme intégrée qui combine les règles de production et un réseau de neurones [82].

On peut aussi citer le système Gardian, c'est un système intelligent de surveillance des patients après une chirurgie cardiaque dans une unité de soins intensifs. La base de connaissances est testée et vérifiée.

Le MDSS est aussi un système qui soutient le diagnostic et le traitement du diabète. Dans ce système on détecte la maladie, sa gravité, et de ses complications potentielles, basée sur les symptômes du patient et des examens de laboratoire [83].

Le système (MADHS) contient principalement quatre types d'agents: Coordonnateur, examinateurs, spécialistes et conjoint décideur. Cette approche est similaire à la conception du système e-médecine présenté par Tian et Tianfield [84]. Mais le coordonnateur de MADHS joue plusieurs rôles en même temps: courtier, agent d'administration, l'agent contrôleur et agent d'interface. La décision conjointe Maker peut être considéré comme une sorte de «agent de décision », tandis que les examinateurs et les spécialistes peuvent être considérés comme des «agents de diagnostic » dans la conception et Tian de Tianfield [85].

Le système EasyDiagnosis est un système qui permet de fournir une analyse instantanée en ligne, des grands symptômes médicaux, dans un format convivial. Ce logiciel donne une description des symptômes particuliers [86]. En voici un extrait :

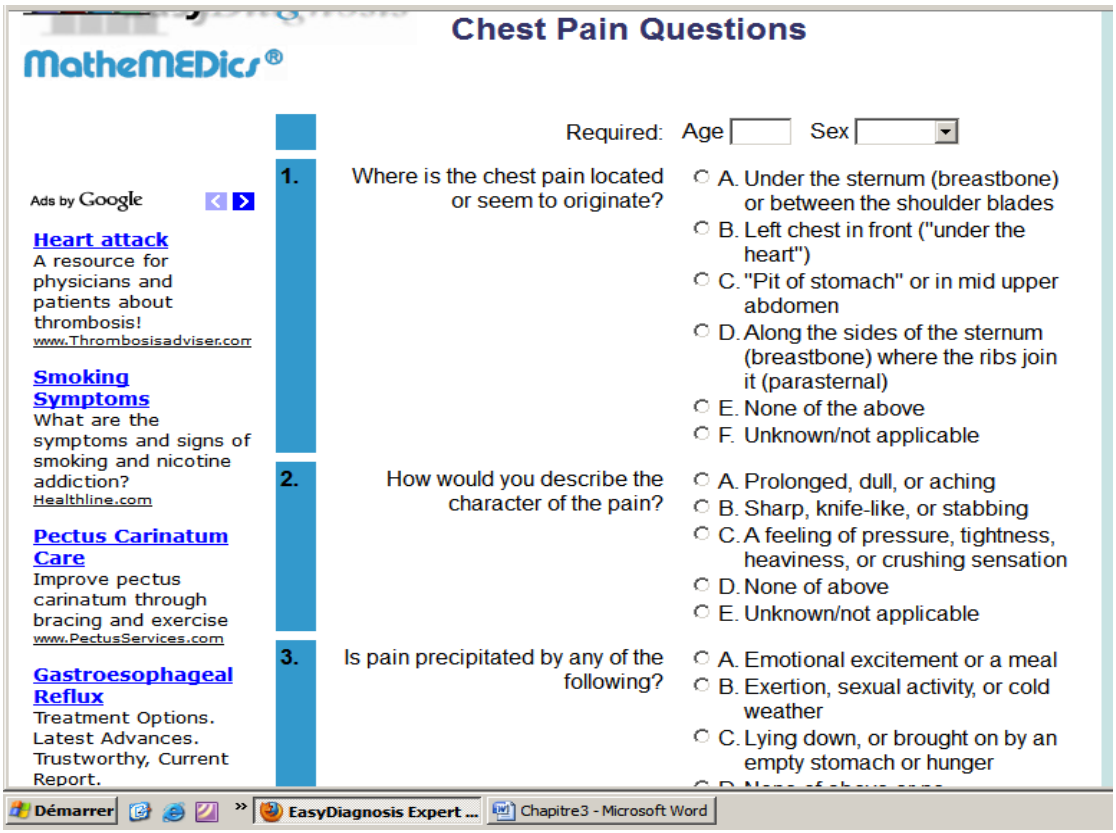


Figure III.12 : Extrait du questionnaire d'EasyDiagnosis

Le système ELIZA est un programme qui simulait un psychothérapeute rogérien, fonctionnant au sein du système MAC en temps partagé au MIT qui rend certains types de conversation en langage naturel entre l'homme et l'ordinateur possible. Les phrases d'entrée sont analysées sur la base des **règles de décomposition qui sont déclenchées par des mots clés figurant dans le texte d'entrée**. Les réponses sont générées par des règles remontage associés aux règles de la décomposition choisie. Les problèmes techniques de base avec laquelle est ELIZA concernés sont:

- L'identification de mots-clés,
- La découverte d'un contexte minimal,
- Le choix des transformations appropriées,
- Génération de réponses en l'absence de mots-clés, [4]

Wed May 18 10:16:03 2011 : qui êtes-vous – Je m'appelle Élizia. Et si on continuait à parler de vous ?
Wed May 18 10:16:26 2011 : qu'est ce que tu veux que je te dise – Merci. Très peu pour moi. Reprenez-vous s'il vous plaît !
Wed May 18 10:16:53 2011 : tu n'es pas bon du tous – Ca vous amuse de me tutoyer ?
Wed May 18 10:17:20 2011 : ça c'est un bon reflexe, je te facilite – Votre intérêt pour moi est sordide.
Wed May 18 10:17:33 2011 : tu le crois – Que pensez-vous de la phrase 'La religion est l'opium du peuple' ?
Wed May 18 10:18:03 2011 : est ce que je suis sensé répondre à cette question – Que cherchez-vous vraiment quand vous me parlez ?
Wed May 18 10:18:15 2011 : te connaitre – Je vous vouvoie, faites de même je vous prie !
Wed May 18 10:18:30 2011 : me connaitre – Je suis ravie que vous parveniez à verbaliser que vous connaitre ?
Wed May 18 10:19:07 2011 : j'ai toujours été très complice avec ma sœur – La famille est un facteur important dans la vie de tous.
Wed May 18 10:19:18 2011 : pourquoi – Vous êtes un peu trop taciturne.
Wed May 18 10:19:32 2011 : ce n'est pas une réponse – On parle toujours mal quand on a rien à dire.
Wed May 18 10:31:42 2011 : peut-on manger demain avec un ami décédé l'année dernière et pourquoi – Ah ! Encore un patient qui ne veut pas guérir ses problèmes, mais qui veut tester mon intelligence...

Figure III.13 : Exemple d'une conversation avec ELIZA

La majorité de ces travaux sont basées en générale sur une base de données comportant l'historique, symptômes et examen de laboratoire du patient (analyse, imagerie Médical...) ([82], [87], [83]), alors que d'autre utilise des règles grammaticaux ([88], [85], [86], [89]). Parmi ces travaux, ceux qui nous intéressent le plus sont les systèmes ELIZA et EasyDiagnosis, car on veut réaliser un système de diagnostic Médical (EasyDiagnosis) négligeant le dialogue inutile (ELIZA), ou le patient aura la description de sa maladie ou de son état le plus rapidement possible et en bref.

III.3.3. Architecture du système SEDMD basé sur l'environnement SyCE-CHM

En (§ 2.4 de ce chapitre), on a dit qu'il serait préférable de mesurer les aptitudes de notre système de compréhension, en termes de portabilité et d'adaptabilité vis-à-vis le domaine du diagnostic Médical. C'est pourquoi, on prévoit toujours de garder la même architecture de l'environnement SyCE-CHM basé sur la méthode d'extraction de concept (voir § 2 de ce chapitre – Figure III.1). Et cela en considérant seulement les modifications suivantes :

- Le nombre de groupe k – Ceci est un paramètre essentiel pour le classificateur k-moyenne. Dans ce sens, on a trouvé que si on pose k=4 (cas idéale), on pourra décrire au mieux la requête du patient, voici donc la hiérarchie des concepts figure III.14.

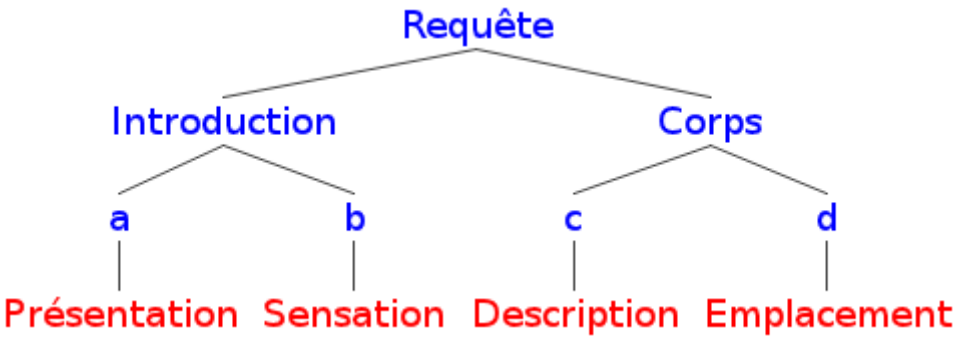


Figure III.14 : Hiérarchie des concepts obtenus pour l'application SEDMD

- La phase de Test – Dans cette phase, on considère que la sortie du décodeur Mot/Concept/Action n'est pas une requête SQL mais plutôt la réponse finale (Diagnostic de la maladie en fonction des symptômes), voici l'architecture de cette phase en figure III.15.

- Moteur d'inférence – Dans ce cas précise l'utilisation d'un moteur d'inférence n'est pas envisageable du fait qu'on ne peut englober toutes les possibilités de combinaison des concepts obtenus, et ceci avec seulement une dizaine de maladies, alors que peut-on dire si on élargie le nombre de maladie. Ce qui nous ramènera à utiliser une approche basé sur la recherche par mots clés (manuel)[9].

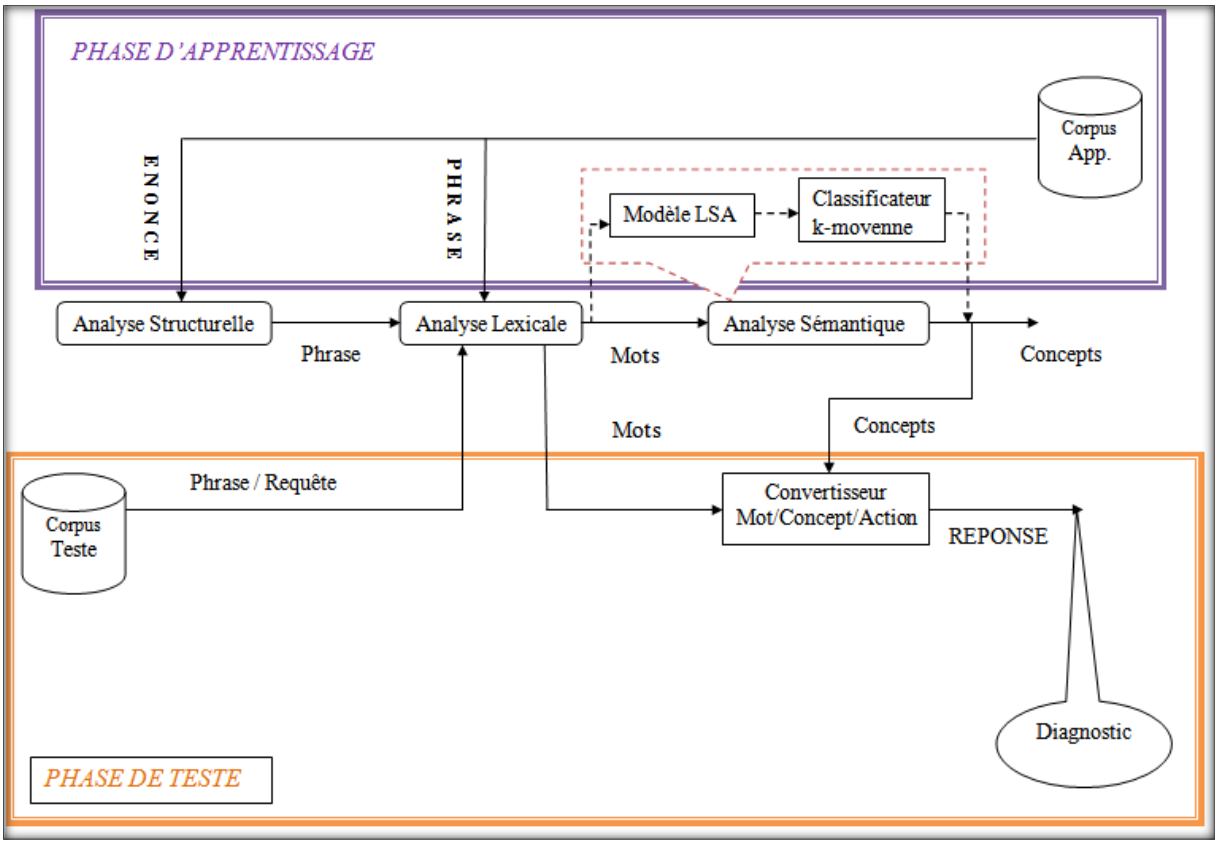


Figure III.15 : Architecture du système de diagnostic médical SEDMD

[9] Du fait qu'on doit sélecter les mots clés à utiliser manuellement.

Dans cette architecture, les analyseurs sont inchangeables, les modifications seront apportées aux niveaux des corpus d'apprentissage, de test et le convertisseur, sans oubliées essentiellement le nombre de concepts k.

III.3.3.1. Résultats obtenus

Les résultats en question sont les concepts de termes obtenus à partir du corpus d'entrée après être traité par chacune des analyseurs structurelle, lexicale et sémantique.

Le corpus en question comporte 27 phrases. La cible de ces phrases est une description de la maladie ainsi que les soins appropriés. Le critère de choix de collection de ces requêtes en corpus est subjectif, où j'ai essayé de considérer sept cas possibles de maladies pour après la généraliser pour d'autres maladies.
La figure suivante montre la liste de phrases considérées lors de cette phase

j'ai eu des difficultés respiratoire, a parler ainsi qu'à avaler.
j'ai des douleurs à l'estomac, vomissement et de la diarrhée.
je me sens faible, des accélérations du rythme cardiaque, de l'anxiété, détresse et perte de conscience.
j'ai une légère fièvre, des maux de tête, des maux de gorge et une voix rauque.
je me sens de la fièvre, des maux de tête, des maux de gorge, une voix rauque ainsi des complications respiratoires.
j'ai des selles fréquentes et liquides avec perte importante de liquide.
j'ai de la fièvre, des vomissements et des douleurs abdominales.
je me sens souvent fatigue même en ayant bien dormi et on a de la peine à se lever le matin.
j'ai des diminutions de l'énergie en général.
j'ai des difficultés à se concentrer.
j'ai une fièvre élevé, des maux de tête importants, une fatigue extrême et des douleurs dans les articulations.
j'ai une toux sèche, des maux de gorge, un rhume ainsi que des nausées.
j'ai de la fièvre a un niveau élevé, et également avoir des frissons à cause de cette fièvre.
j'ai des douleurs au niveau des articulations.
j'ai de forts maux de tête.
j'ai des courbatures.
j'ai des écoulements nasals, éternuement, larmoiement des yeux.
j'ai une perte d'appétit, fièvre, maux de tête et de l'irritation de la gorge.
j'ai des troubles digestifs, des ulcères, des nausées et vomissements.
je m'énerve rapidement.
j'ai des problèmes cardiaques.
j'ai divers troubles du sommeil.
j'ai des troubles psychiques : nervosité, colère excessive et inhabituelle, crise d'angoisse, peurs excessives, dépression.

j'ai un problème de chute des cheveux.
j'ai des problèmes de concentration.
je suis fatigue, pâleur de la peau et des muqueuses, des palpitations ainsi que des essoufflements.
j'ai des maux de tête, des vertiges et des bourdonnements d'oreille.

Figure III.16 : Le corpus de l'application de Diagnostic Médical à Distance

Á partir de ce corpus de phrases, voici les concepts qu'on a récupéré manuellement depuis le corpus d'apprentissage de l'application de diagnostic Médical.

Concept	**Liste des termes formant le concept**
Présentation	Sens, cause, matin, niveau,
Sensation	Pâleur, peine, nausées, larmoiement, perte, frissons, sèche, crise, lever, écoulement, énerve, irritation, vomissement, dépression, selles, phobies, nervosité, avaler, diarrhée, éternuement, essoufflement, détresse, parler, vomissements, bourdonnements, muqueuses, peurs, douleurs, angoisse, maux, colère, sommeil, anxiété, fièvre, palpitations, concentration, rhume, appétit, fatigue, courbatures, concentrer, vertiges, dormi.
Description	Rauque, cardiaques, excessive, troubles, élevé, psychiques, importante, difficultés, importants, cardiaque, conscience, respiratoires, énergie, respiratoire, inhabituelle, difficulté, diminution, extrême, liquides, accélérations, rapidement, excessives, rythme, complications, problèmes, générale, accélération, fréquentes, souvent, également, liquide, légère, problème, chute, forts, faible.
Emplacement	Abdominales, yeux, voix, gorge, toux, estomac, oreille, cheveux, ulcères, peau, nasal, articulations, digestifs, tête.

Table III.9 : Les Concepts Manuelles de l'applications de Diagnostic Médical

Maintenant, en considérant l'utilisation de la LSA sur une matrice d'occurrences booléenne et TF-IDF suivie par une classification k-moyennes pour les valeurs de k=2 et k=4, on aura les résultats suivants :

(4 concepts manuelles)
G1/12, 34, 73, 90
G2/3, 4, 7, 8, 16, 18, 20, 22, 23, 24, 25, 27, 28, 31, 33, 40, 43, 44, 53, 54, 55, 56, 57, 58, 60, 61, 62, 63, 64, 66, 67, 68, 71, 72, 74, 77, 81, 85, 88, 89, 93, 94, 96
G3/1, 2, 5, 9, 10, 11, 14, 17, 19, 21, 22, 26, 29, 30, 32, 35, 36, 37, 38, 41, 42, 45, 48, 49, 50, 51, 65, 69, 70, 75, 76, 78, 79, 84, 86, 91, 92,
G4/6, 13, 15, 39, 46, 47, 52, 59, 80, 82, 83, 87, 95, 97
Résultat: matrice booléenne

K=4 :
x= les éléments restants
y= 2, 15, 39, 63, 68, 97
z= 3, 4, 6, 12, 16, 22, 27, 34, 43, 50, 55, 57, 71, 73, 78, 79, 82, 96
q= 10, 21, 41, 60, 85, 87, 90

K=2 :
x= les éléments restant
y= 2, 15, 39, 63, 68, 97

Résultat : matrice TF-IDF

K=4 :

x= 69, 72

y= 1, 26, 61, 75

z= les éléments restants

q= 49, 89

K=2 :

x= 69, 72

y= les éléments restants

Liste des termes :

['1-cardiaques', '2-rauque', '3-paleur', '4-peine', '5-excessive', '6-abdominales', '7-nausees', '8-larmoiement', '9-troubles', '10-eleve', '11-perte', '12-sens', '13-yeux', '14-psychiques', '15-voix', '16-frissons', '17-importante', '18-seche', '19-difficultes', '20-crise', '21-importants', '22-lever', '23-ecoulement', '24-enerve', '25-irritation', '26-cardiaque', '27-vomissement', '28-depression', '29-conscience', '30-respiratoires', '31-selles', '32-energie', '33-phobies', '34-cause', '35-respiratoire', '36-inhabituelle', '37-difficulte', '38-diminution', '39-gorge', '40-nervosite', '41-extreme', '42-avaler', '43-diarree', '44-eternuement', '45-liquides', '46-toux', '47-estomac', '48-acceleartions', '49-rapidement', '50-essoufflement', '51-excessives', '52-oreille', '53-detresse', '54-parler', '55-vomissements', '56-bourdonnements', '57-muqueuses', '58-peurs', '59-cheveux', '60-douleurs', '61-rythme', '62-angoisse', '63-maux', '64-colere', '65-complications', '66-sommeil', '67-anxiete', '68-fievre', '69-problemes', '70-general', '71-palpitations', '72-concentration', '73-matin', '74-rhume', '75-acceleration', '76-frequentes', '77-appetit', '78-souvent', '79-egalement', '80-ulceres', '81-liquide', '82-peau', '83-nasal', '84-legere', '85-fatigue', '86-probleme', '87-articulations', '88-chute', '89-courbatures', '90-niveau', '91-forts', '92-faible', '93-concentrer', '94-vertiges', '95-digestifs', '96-dormi', '97-tete'

Figure III.17 : La liste des termes significatifs et les résultats du classificateur.

III.3.3.2. Validation des résultats

Voici un extrait du calcul de mesures précédentes :

 # Classe 3: Concept 3
y3=[1, 2, 5, 9, 10, 11, 14, 17, 19, 21, 22, 26, 29, 30, 32, 35, 36, 37, 38, 41, 42, 45, 48, 49, 50, 51, 65, 69, 70, 75, 76, 78, 79, 84, 86, 91, 92]
pred3= [1, 26, 61, 75]
TP3=3 **# (true positive) Correct result 1, 26, 75**
FN3= 34**# (false negative) Missing result 5, 9, 10, 11, 14, 17, 19, 21**…
FP3=1 **# (false positive) Unexpected result 61**
TN3=54**# (true negative) Correct absence of result**

Figure III.18 : Un extrait du calcul des mesures

Voici les résultats des mesures de validations obtenues pour k=4. Ces résultats contiennent les précisions, rappels et F1 pour chaque classe ainsi que pour toutes les classes.

Matrice booléenne	
K=4	
Concept1 precision: 0.0	Concept2 precision: 0.439393939394
Concept1 recall: 0.0	Concept2 recall: 0.674418604651
Concept1 F-measure: 0	Concept2 F-measure: 0.532110091743
Concept3 precision: 0.428571428571	Concept4 precision: 0.111111111111
Concept3 recall: 0.0810810810811	Concept4 recall: 0.142857142857
Concept3 F-measure: 0.136363636364	Concept4 F-measure: 0.125
ConceptAll micro_avg_precision: 0.350515463918	
ConceptAll micro_avg_recall: 0.336633663366	
ConceptAll micro_avg_F-measure: 0.343434343434	
K=2	
Concept1 precision: 0.0	Concept2 precision: 0.95652173913
Concept1 recall: 0.0	Concept2 recall: 0.936170212766
Concept1 F-measure: 0	Concept2 F-measure: 0.94623655914
ConceptAll micro_avg_precision: 0.936170212766	
ConceptAll micro_avg_recall: 0.897959183673	
ConceptAll micro_avg_F-measure: 0.916666666667	
Matrice TF-IDF	
K=4	
Concept1 precision: 0.0	Concept2 precision: 0.449438202247
Concept1 recall: 0.0	Concept2 recall: 0.93023255814
Concept1 F-measure: 0	Concept2 F-measure: 0.606060606061
Concept3 precision: 0.75	Concept4 precision: 0.0
Concept3 recall: 0.0810810810811	Concept4 recall: 0.0
Concept3 F-measure: 0.146341463415	Concept4 F-measure: 0
ConceptAll micro_avg_precision: 0.443298969072	
ConceptAll micro_avg_recall: 0.438775510204	
ConceptAll micro_avg_F-measure: 0.441025641026	
K=2	
Concept1 precision: 0.0	Concept2 precision: 1.0
Concept1 recall: 0.0	Concept2 recall: 1.0
Concept1 F-measure: 0	Concept2 F-measure: 1.0
ConceptAll micro_avg_precision: 0.979591836735	
ConceptAll micro_avg_recall: 0.96	
ConceptAll micro_avg_F-measure: 0.969696969697	

Figure III.19 : Les mesures de validations pour k=4 et k=2 pour matrice booléenne et TF-IDF pour l'application de diagnostic médical

Dans le récapitulatif suivant, on va considérer les mesures de validations « micro average ». Du fait qu'elle exprime le taux de classification totale (pour toutes les classes).

Méthode	Extraction de Concept			
Application	Diagnostic Médical			
Modèle	LSA + k-moyenne			
Type de matrice	Matrice Booléenne		Matrice TF-IDF	
Paramètres	K=4	K=2	K=4	K=2
Précision (%)	35.05	93.61	44.32	97.95
Rappel(%)	33.66	89.79	43.87	96
F1[10] (%)	34.34	91.66	44.10	96.96

Table III.11 : Résultats obtenus par la méthode Extraction de Concepts avec l'application Diagnostic Médical

En considérant ce récapitulatif, on remarque qu'on a les meilleurs concepts possibles avec l'utilisation de la LSA sur une matrice pondéré avec la TF-IDF. Le taux de classification est aux alentours de 44.10%.

III.4. Résultats expérimentales

Le taux de compréhension qu'on a atteint avec l'application de Consultation Scolaire basé sur l'approche d'extraction de concept sur 60 requêtes est de l'ordre de 73.33% qui est beaucoup mieux à celui basé sur l'approche de recherche par mots clés qui est de l'ordre de 68.33%.

Un extrait de cette expérience est représenté dans ce Table (pour les résultats totales, voir annexe F) :

	Système de Consultation basé sur les Concepts	Résultats	mots clés
Utilisateur 1	je veux savoir ma note	OK	ok
	combien j'ai eu en télévision	OK	no
	pouvez-vous me donner ma note de fibre optique	OK	ok
	j'aimerai bien voir mon relevé de note	OK	ok
	s'il vous plait, donnez-moi mon certificat de scolarité	OK	ok
	donne-moi la note de micro-onde	OK	ok
	est-ce que j'ai la moyenne en champ	OK	no
	je veux connaitre mes résultats	NO	no
	excusez-moi, je veux avoir mon certificat de scolarité	Ok	ok
	mes résultats sont-elles acceptable	NO	no
Utilisateur 2	donnez-moi mon certificat scolarité	Ok	ok
	je veux savoir ma note en antenne	ok	ok
	je voudrai savoir ma note en télévision	ok	ok
	donne-moi ma note de micro-onde	ok	ok
	quelle est ma moyenne	NO	no

[10] La F-measure de Van Rijsbergen [81] dite aussi efficacité globale,

	pouvez-vous me donner mon certificat scolarité	ok	ok
	je voulais connaitre ma note en fibre optique	ok	ok
	je cherche mon résultat	NO	no
	J'aimerai bien avoir mon certificat de scolarité si elle est prête	ok	ok
	serait-ce possible d'avoir mon diplôme aujourd'hui	ok	ok

Table III.12 : Extrait de l'expérience Expérimentale

Cette expérience a été réalisée en considérant deux manipulations, une sur notre système (basée sur les concepts) et la deuxième une recherche par mots clés. On a pu avoir 44 phrases "OK" à partir de "60" phrases en entrée, donc un taux de compréhension de 44/60=73,33% pour le premier système. Pour celui de la recherche par mots clés, on a eu 41 phrases "OK" à partir de "60" phrases en entrée, donc un taux de compréhension de 41/60=68,33%.

Et afin d'ajouter plus d'élégance à notre système, on a ajouté une modalité en sortie qui est un synthétiseur de la parole.

D'autre part, pour donner plus de crédibilité à cette approche de compréhension de l'énoncé basé sur l'extraction de concept, on a voulu la soumettre à une autre application qui est celui du Diagnostic Médical à Distance. Mais certes ceci est irréalisable à cause de la richesse linguistique du domaine Médical et encore parce qu'on ne peut créer une grammaire générative aux requêtes (symptôme) du patient. C'est pourquoi l'introduction de la deuxième approche basé sur la seconde propriété du processus de compréhension cognitif humain.

III.5. Conclusion

Dans ce chapitre, nous avons décrit les différents composants de notre système de compréhension de l'énoncé SyCE-CHM (basé sur l'approche d'extraction de concept) et ce en considérant les deux phases de traitement, celle de l'apprentissage ainsi que celle du test en submergeant leurs différents modules (y compris ceux qui sont commun entre les deux phases).

En second, en s'inspirant de l'architecture générale de l'environnement SyCE-CHM, on a fait surgir l'application de Scolarité pour valider ainsi quantifier les performances de cet environnement en milieu réel. Ceci dit l'application de cette architecture [6] a bien prouvé sa réputation en ce qui concerne les systèmes de compréhension de l'énoncé pour l'interrogation d'une base de données SGBD.

C'est pourquoi, on a voulus étendre son axe de spécialité vers les systèmes de diagnostic Médical utilisant une base de connaissances. Á titre d'exemple, on a pris un Système Expert de Diagnostic Médical à Distance (où, le patient n'est plus à côté du système faisant le diagnostic, et encore sans l'intervention du médecin). Mais à ce stade cette architecture a approuvé des limites et ce au niveau de la phase d'apprentissage avec un taux de classifications de 44.10% et celle du test qui n'est plus réalisable à cause de la richesse linguistique du domaine Médical. D'une autre part, on ne peut formaliser la requête du patient avec une suite

de concept. Une autre alternative est de considérer une annotation manuelle par mots clés (non souhaité).

En fin, on peut conclure que cette architecture SyCE-CHM est tous a fait conseiller pour des systèmes d'interrogation de base de données mais à un champ d'application particulier où les services à fournir sont distincts et limités. Néanmoins, y en a bien des limitations à remédier et ce en terme de taux de classifications de concepts qui est aux alentours de 54.90 %. Mais ceci ne veut pas dire que le système est faillible car une simple modification des concepts nous à ramener vers un taux de compréhension de 73.33%.

En effet, c'est pour ça qu'on va entamer le prochain chapitre et ce pour discuter et planifier une architecture d'un système SyCE-CHM portable et adaptable à toute application, domaine et environnement.

Chapitre IV :

Système de Compréhension utilisant une Analyse Stochastique des Énoncés

Chapitre IV :
Système de Compréhension utilisant une Analyse Stochastique des Énoncés

IV.1. Introduction

La généralisation des systèmes de compréhension d'énoncés reste toujours restreinte par la langue. Ces systèmes dépendent des systèmes de reconnaissances de la parole multilingue et de leurs fiabilités (mono, multi-locuteur).
Afin de satisfaire cette ambition, en ce qui suit deux partie sont nécessaire, la première visant à remédier aux différents problèmes rencontrés par le précédent système SyCE-CHM[1] :

- Un taux de classification en phase d'apprentissage, relativement faible. Ce qui engendre une intervention de l'expertise humaine pour corriger les classes (non souhaitable).
- Une faible tolérance envers les nouveaux termes, car une étude plus approfondie sur le domaine d'application est nécessaire. Cette étude va nous aider à concevoir un gestionnaire de termes hors-vocabulaires plus performants. Or, au début de ce travail, on a supposé une intervention minime de l'expertise humaine. Donc cette démarche n'est plus valable pour notre cas.
- Á chaque nouvelle application, on est obligé de reconstruire une nouvelle grammaire pour le générateur des requêtes SQL. Qui est fastidieux et couteux.
- Le non possibilité de considérer des applications d'extraction d'information (où l'information n'est plus dans une base de données mais plutôt dans une base de connaissances).

La deuxième partie visant à la généralisation du système SyCE-CHM envers différents langages ([90], [91], [92]) et domaines. Ceci dit soit par améliorer le système en cours ou bien proposer un nouvel système.

IV.2. Remédiation [2] des problèmes du SyCE-CHM

Les problèmes qu'on a rencontrés avec ce système se situent au niveau du classificateur k-moyennes ainsi qu'au modèle algébrique LSA. C'est pourquoi dans ce qui suit, on va essayer de résoudre ces problèmes et voir s'ils peuvent affectés le fonctionnement de notre système (positivement ou négativement).

---

[1] Le système basé sur les concepts de termes n'est utiles et intéressant, que qu'on en a une seul application à la fois.
[2] Mise en œuvre des moyens permettant de résoudre des difficultés d'apprentissage repérées au cours d'une évaluation

IV.2.1. Classificateur k-moyennes

Pour le classificateur, on s'attendait à des taux de reconnaissances (classification) relativement grande par rapport à la taille du corpus d'apprentissage, alors qu'en réalité on n'a pas pu franchir le plafond de 34% et ce avec une valeur initial de k=4.
Pour mieux localiser la source du problème au niveau du classificateur, une performance minimum serait fortement demandé, sinon réattribuer les paramètres du classificateur (mesure de similitude –distance--, nombre de groupe initial k). La performance minimum requise est quand k=2.
Pourquoi exactement cette valeur ? Quelle est l'étendu des systèmes expert ? Ou bien qu'est-ce qu'on attend de ces systèmes ?
Pour dire vrais, ou n'attend qu'une simple effectuation des services supportés (c.à.d. réaliser la tache demandé). Or, la majorité de ces systèmes sont basés sur une topologie : Question/réponse ou requête/réponse. Ceci dit, n'importe quelle application basée sur un système expert doit avoir au minimum deux concepts (concepts_Introduction) et (concepts_Objet). Donc k=2.

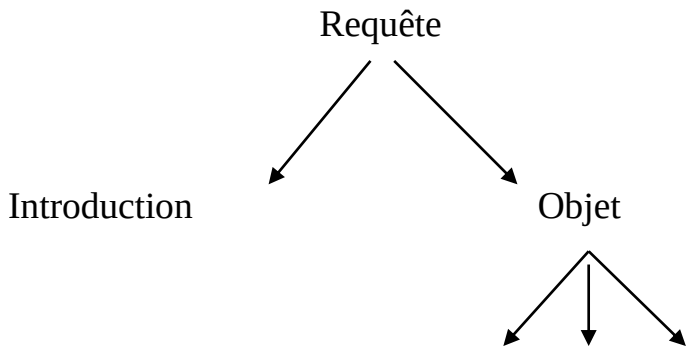


Figure IV.1 : Hiérarchie de concept minimale pour un système expert

En choisissant cette valeur de k, le taux de classifications a pu atteindre 81.35%. Mais cette solution n'est pas envisageables (voir § II.2.1.2 p.58 du chapitre 3). Donc dans ce qui suit, on va considérer le nombre de concept k=4 pour les deux applications.

IV.2.2. Limitation du modèle LSA

Malgré que la LSA nous ait permis d'extraire le sens caché des termes et les phrases ainsi que la similarité entre deux termes ou deux phrases [93], ceci n'est utile et performant que dans certaines situations.
Les limites de LSA sont :
a) L'impossibilité (dans le modèle de base) de prendre en compte la Polysémie (c'est-à-dire les sens multiples d'un mot), car un mot ne correspond qu'à un seul point de l'espace sémantique.

b) Les dimensions qui en résultent peuvent être très difficile à interpréter ce qui induit nécessairement des erreurs. On ne sait pas ce que les similitudes entre les termes résultants signifient vraiment.
c) L'entrée est un sac-de-mots que nous n'avons aucune information sur la structure du texte.
d) Les termes ambigus créent des bruits dans l'espace vectoriel.
e) Il n'y a pas moyen de définir la dimension optimale de l'espace vectoriel [94].

Á partir de ces limites, il apparait la faiblesse de ce modèle. Nous allons voir dans ce qui suit comment l'amélioration de ce modèle peut donner de meilleurs résultats (Analyse sémantique latente probabiliste).

IV.2.3. Analyse sémantique latente probabiliste PLSA

L'Analyse Sémantique Latente (LSA) et sa branche probabiliste (PLSA) sont des méthodes de traitement statistique du langage naturel. Cette analyse permet d'établir des relations entre un ensemble de phrases et les termes contenus dans ces phrases, on obtient donc des classes latentes liées aux phrases et aux termes (On est passé d'une liaison directe Terme→Phrase à une indirecte Terme→Classe Latente→Phrase).
La LSA utilise une matrice des occurrences termes-phrases. Il s'agit d'une matrice creuse dont les lignes correspondent aux termes et les colonnes correspondent aux phrases. Le nombre d'apparitions d'un terme dans chaque phrase est normalisé en utilisant la pondération TF-IDF. Cette méthode met en avant les termes qui apparaissent souvent dans une phrase mais rarement entre différents phrases. Les termes choisis caractérisent donc les phrases auxquels ils appartiennent.
La matrice des occurrences est ensuite décomposée en valeurs singulières (SVD), produisant un «espace sémantique», où les termes d'origine et les phrases sont représentés comme des vecteurs et le «sens» d'un terme est représenté par son vecteur approprié. Une comparaison statistique et une classification des termes est alors possible dans cette «espace sémantique», par le calcul d'une mesure de similarité entre vecteurs de «sens» (généralement la distance euclidienne)[3].
La PLSA améliore la méthode LSA en introduisant une approche probabiliste pour la réduction de la matrice des occurrences, suivant le principe de vraisemblance. Ce «modèle d'aspect» effectue un mélange de décompositions issues de l'analyse des classes latentes, en associant une variable de classe inobservée à chaque observation. La méthode utilise un algorithme EM tempéré pour l'estimation du maximum de vraisemblance de chaque variable latente, afin d'éviter le sur-apprentissage. En résumé la LSA est une simple analyse algébrique,

[3] Idem, le sens d'une phrase est représenté par la moyenne des vecteurs des termes apparaissant dans la phrase en question. Une comparaison statistique et une classification des phrases est alors possible dans ce «espace sémantique», par le calcul d'une mesure de similarité entre vecteurs de «sens» (généralement le cosinus).

alors que la PLSA est une analyse algébrique-statistique (combinant une méthode algébrique avec une statistique).

Modèle PLSA

Le modèle PLSA est un modèle statistique qui a été appelé le modèle aspect [95]. Ce dernier est un modèle probabiliste qui associe une variable de classe latente $z \in Z = \{z_1, z_2, \dots, z_k\}$ (ensemble des variables de classes) pour chaque observation (une observation est l'occurrence d'un terme $t \in T = \{t_1, t_2, \dots, t_n\}$ (ensemble des termes dans le corpus) dans une phrase $p \in P = \{p_1, p_2, \dots, p_m\}$ (ensemble de phrases du corpus)). En termes d'un modèle génératif il peut être défini de la manière suivante:

- Une phrase p_j est sélectionnée selon la probabilité $P(p_j)$.
- Une classe latente (caché) z_k est choisi selon la probabilité $P(z_k/p_j)$.
- Un terme t_i est généré avec la probabilité $P(t_i/z_k)$.

En conséquence, on obtient une paire observée (t, p), tandis que la variable de la classe latente z est cachée. [5]

Exprimer ceci en un modèle de probabilité conjointe se résulta par :

$$P(p,t) = P(p)P(t/p) \qquad \text{(IV.1)}$$

Où :

$$P(t/p) = \sum_{z \in Z} P(t/z)P(z/p) \qquad \text{(IV.2)}$$

Essentiellement, pour calculer (IV.2) il faut sommer sur les possibles choix de z qui pourrait avoir généré l'observation t (considérer les probabilités d'appartenance de ce terme à tous les classes z).

Ce modèle est basé sur deux hypothèses d'indépendance:

- Les paires d'observation (p, t) sont censées être produites de façon indépendante, ce qui correspond essentiellement à l'approche de « la valise de termes».
- L'hypothèse d'indépendance conditionnelle est faite tel que conditionné par la classe latente z, les termes t sont produites de façon indépendante à la spécification d'identité de phrases p.

Suivant le principe de vraisemblance, on détermine $P(p), P(z/p)$ et $P(t/z)$ par la maximisation de la fonction de vraisemblance.

$$\mathcal{L} = \sum_{s \in S} \sum_{w \in W} n(p,t)\, log\, P(p,t) \qquad \text{(IV.3)}$$

Où n (p, t) désigne la fréquence du terme, c'est à dire, le nombre de fois ou t s'est produite dans p. Il convient de noter qu'une version équivalente symétrique de ce modèle peut être obtenue en inversant la probabilité conditionnelle $P(z/p)$ avec l'aide de la règle de Bayes, qui se traduit par :

$$P(p,t) = \sum_{z \in Z} P(z)P(t/z)P(p/z) \quad \text{(IV.4)}$$

✓ **Estimation par EM**

La procédure standard pour l'estimation du maximum de vraisemblance dans les modèles de variables latentes est l'algorithme de la maximisation de l'Espérance (Maximum-Expectation EM). EM alterne deux étapes: (i) une étape d'espérance(E) où les probabilités a posteriori sont calculés pour les variables latentes z, sur la base des estimations actuelles des paramètres,(ii) une étape de maximisation(M), où les paramètres sont mis à jour compte tenu des probabilités a posteriori calculées à l'étape E précédente.

Pour le modèle aspect dans la règle de paramétrisation symétrique de Bayes, l'étape E donne : (IV.5)

$$P(z/p,t) = \frac{P(z)P(p|z)P(t|z)}{\sum_{z'} P(z')P(p|z')P(t|z')}$$

P(z/p, t) est la probabilité qu'un terme t dans une phrase s s'explique par le facteur correspondant à z. Par des calculs classiques, on arrive à des équations de ré-estimation suivantes l'étape M :

(IV.6)

$$P(t/z) = \frac{\sum_p n(p,t)P(z/p,t)}{\sum_{p,t'} n(p,t')P(z/p,t')'}$$

(IV.7)

$$P(p/z) = \frac{\sum_s n(p,t)P(z/p,t)}{\sum_{p',t} n(p',t)P(z/p',t)'}$$

(IV.8)

$$P(z) = \frac{\sum_{p,t} n(p,t)P(z/p,t)}{\sum_{p,t} n(p,t)}$$

IV.2.4. Intégration du Modèle PLSA au SyCE-CHM

Pour utiliser le modèle PLSA dans le SyCE-CHM, on a supprimé le classificateur k-moyennes, puisque notre modèle PLSA fournira en sa sortie les probabilités d'appartenance des termes t du corpus au classe latente z. Ce qui facilitera de loin la reconstruction des concepts par ordre de probabilité (Voir Annexe D). Cela veut dire que l'attribution d'un terme t_i à une classe latente z_k nécessite que la probabilités $P(t_i/z_k)$ est maximale.

A. Listes des Concepts trouvés par la méthode PLSA

1) Application de Consultation Scolaire

En appliquant la méthode PLSA sur une matrice d'occurrence booléenne, voici les résultats qu'on a trouvé. Pour un nombre de concept égal à 4.

Ptz : [[1.0, 1.0, 0.0, 0.0], [1.0, 0.0, 0.0, 0.0], [2.0, 0.0, 0.0, 2.0], [0.5, 0.0, 0.0, 0.5], [2.0, 0.0, 0.0, 1.0], [4.0, 0.0, 0.0, 0.0], [1.0, 0.0, 0.0, 3.0], [1.0, 0.0, 0.0, 0.0], [1.0, 0.0, 0.0, 0.0], **[6.0, 0.0, 0.0, 2.0]**, [0.0, 1.0, 0.0, 0.0], [2.0, 1.0, 0.0, 0.0], [1.0, 0.0, 0.0, 0.0], [0.0, 1.0, 0.0, 0.0], [3.0, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0, 1.0], [1.0, 0.0, 0.0, 0.0], [3.0, 0.0, 1.0, 0.0], [0.0, 0.0, 0.0, 1.0], [1.0, 0.0, 0.0, 0.0], [1.0, 0.0, 0.0, 1.0], [2.0, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0, 4.0], [0.0, 0.0, 1.0, 1.0], [3.0, 0.0, 0.0, 3.0], [2.0, 0.0, 0.0, 0.0], [1.0, 0.0, 0.0, 0.0]]

G1 :1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 13, 15, 17, 18, 20, 21, 22, 25, 26, 27
G2 :1, 11, 14
G3 :24
G4 :3, 4, 7, 16, 19, 21, 23, 24, 25

Figure IV.2 : Les résultats du modèle PLSA sur une matrice Booléenne

Ces résultats représentent les probabilités d'appartenances de chaque terme t à l'une des quatre concepts z. Afin de connaitre la classe (concept) d'un terme, on va considérer la classe avec la probabilité maximale.

Exemple :

On prend le 10^{éme} terme, ses probabilités sont : **[6.0, 0.0, 0.0, 2.0]**. Cela se traduit par, le fait que ce terme va appartenir à la première classe (probabilité =6.0).

N.B : En cas d'équiprobabilité, on va présenter le même élément dans les deux classes respectives. En suivant ce principe, on aura les concepts suivants :

K=4(Booléenne)	
Concept	**Liste des mots formant le concept**
Concept n°1	Informatique, moyenne, module, certificat, scolarité, veux, donner, aimerais, physique, voudrais, résultat, pouvez, math, souhaite, diplôme, résultats, chimie, savoir, donne, aimerai, obtenu.
Concept n°2	Informatique, voudrais, pouvez
Concept n°3	relevée
Concept n°4	Module, certificat, donner, veuillez, montrez, chimie, examen, relevée, donne

Table IV.1 : Listes des Concepts trouvés par la méthode PLSA sur une Matrice Booléenne pour l'application de Consultation Scolaire (K=4)

En calculant les mesures de validations précision, rappel et F1 ; on trouve :

Concept1 precision: 0.4
Concept1 recall: 0.666666666667
Concept1 F-measure: 0.5

Concept2 precision: 0.444444444444
Concept2 recall: 0.8
Concept2 F-measure: 0.571428571429

Concept3 precision: 1.0
Concept3 recall: 0.25
Concept3 F-measure: 0.4

Concept4 precision: 0.0
Concept4 recall: 0.0
Concept4 F-measure: 0

ConceptAll micro_avg_precision: 0.40625
ConceptAll micro_avg_recall: 0.52
ConceptAll micro_avg_F-measure: 0.456140350877

Figure IV.3 : Les mesures de validations pour PLSA(Booléenne) scolaire

En considérant le modèle PLSA sur une matrice booléenne et avec k=4, le taux de classification est diminué de 61.53% avec la LSA vers 45.61%.
De même, en considérant une matrice d'occurrence TF-IDF, voici la liste des résultats obtenus à partir la PLSA.

Ptz : [[0.29999999999999999, 0.0, 0.0, 0.0], [0.65000000000000002, 0.0, 0.0, 0.0], [0.52000000000000002, 0.0, 0.0, 0.47999999999999998], [0.0, 0.63636363636363635, 0.36363636363636359, 0.0], [0.78999999999999992, 0.0, 0.0, 0.23999999999999999], [0.51000000000000001, 0.0, 0.0, 0.0], [0.0, 0.60999999999999999, 0.0, 0.0], [0.0, 0.0, 0.0, 0.82000000000000006], [0.0, 0.0, 0.0, 0.0], [0.38, 0.0, 0.0, 0.0], [0.98000000000000009, 0.0, 0.19, 0.0], [0.67999999999999994, 0.0, 0.0, 0.0], [0.57999999999999996, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0, 0.42999999999999999], [0.66000000000000003, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0, 0.0], [0.42999999999999999, 0.0, 0.0, 0.0], [0.34000000000000002, 0.0, 0.0, 0.0], [1.4299999999999999, 0.0, 0.0, 0.0], [0.38, 0.0, 0.0, 0.0], [0.29999999999999999, 0.0, 0.0, 0.0], [0.72999999999999998, 0.0, 0.0, 0.0], [0.28999999999999998, 0.0, 0.46999999999999997, 0.0], [0.0, 0.0, 0.0, 0.42999999999999999], [0.63, 0.0, 0.0, 0.0], [0.29999999999999999, 0.0, 0.0, 0.0], [1.0700000000000001, 0.0, 0.0, 0.0], [0.42999999999999999, 0.0, 0.0, 0.0]]

G1 :1, 2, 3, 5, 6, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 22, 25, 26, 27, 28
G2 : 7, 9, 16,
G3 : 4, 9, 16, 23,
G4 :8, 9, 14, 16, 24,

Figure IV.4 : Les résultats du modèle PLSA sur une matrice TF-IDF pour l'application de consultation scolaire

En ce qui suit les concepts obtenus correspondants.

K=4 (TF-IDF)	
Concept	**Liste des mots formant le concept**
Concept n°1	Moyenne, module, certificat, veux, donnez, aimerais, physique, note, résultat, math, informatique, souhaite, veuillez, diplôme, montrer, résultats, chimie, relevée, modules, donne, obtenu
Concept n°2	Voudrais, aimerais, informatique
Concept n°3	Scolarité, aimerais, informatique, savoir
Concept n°4	Donner, aimerais, renseigner, informatique, examen

Table IV.2 : Listes des Concepts trouvés par la méthode PLSA sur une Matrice TF-idf pour l'application de Consultation Scolaire

Et pour les mesures de validations

Concept1 precision: 0.363636363636
Concept1 recall: 0.666666666667
Concept1 F-measure: 0.470588235294

Concept2 precision: 0.25
Concept2 recall: 0.2
Concept2 F-measure: 0.22222222

Concept3 precision: 0.2
Concept3 recall: 0.25
Concept3 F-measure: 0.22222222

Concept4 precision: 0.3333333
Concept4 recall: 0.25
Concept4 F-measure: 0.285714285714

ConceptAll micro_avg_precision: 0.33333333
ConceptAll micro_avg_recall: 0.44
ConceptAll micro_avg_F-measure: 0.379310344828

Figure IV.5 : Les mesures de validations pour PLSA (TF-IDF) scolaire

Depuis ces résultats, on remarque que pour ce cas précis, on a obtenus un taux de 37.93% (le plus faible jusque-là). Mais tous de même, on voit qu'elle nous a donné des concepts bien répartie, où il n'y avait pas des concepts vides.

2) Application de Diagnostic Médical à Distance

Les résultats trouvés sont présentés dans la figure IV.6.

G1 : 24,32,47,52,62,67,77, 3, 4, 5, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 20, 21, 22, 23, 25, 27, 28, 30, 31, 33, 34, 36, 38, 39, 41, 42, 43, 48, 49, 50, 51, 53, 54, 56, 57, 58, 59, 60, 63, 64, 65, 68, 69, 70, 72, 73, 78, 79, 80, 81, 83, 84, 86, 87, 88, 89, 90, 91, 95, 96
G2 : 1,2,6,24,32,40,44,45,47,52,62,67,77
G3 : 9,18,19,24,32,35,37,46,47,52,55,61,62,66,67,74,75,77,82,85,93,94
G4 : 24,26,29,32,47,52,62,67,71,76,77,92

ptz [[0.0, 0.33000000000000002, 0.0, 0.0], [0.24000000000000002, 0.0, 0.0, 0.0], [0.17999999999999999, 0.0, 0.0, 0.0], [1.0, 0.0, 0.0, 0.0], [0.14000000000000001, 0.0, 0.0, 0.0], [0.0, 0.27000000000000002, 0.0, 0.0], [0.20000000000000001, 0.0, 0.0, 0.14999999999999999], [0.33000000000000002, 0.0, 0.0, 0.0], [0.0, 0.10000000000000001, 0.60999999999999999, 0.0], [0.26000000000000001, 0.0, 0.0, 0.0], [0.42999999999999999, 0.0, 0.0, 0.0], [0.14999999999999999, 0.0, 0.0, 0.0], [0.33000000000000002, 0.0, 0.0, 0.0], [0.14000000000000001, 0.0, 0.0, 0.0], [0.24000000000000002, 0.0, 0.0, 0.0], [0.17000000000000001, 0.0, 0.0, 0.0], [0.27000000000000002, 0.0, 0.0, 0.0], [0.0, 0.0, 0.19, 0.0], [0.0, 0.0, 0.27000000000000002, 0.0], [0.14000000000000001, 0.0, 0.0, 0.0], [0.16, 0.0, 0.0, 0.0], [0.14999999999999999, 0.0, 0.0, 0.0], [0.33000000000000002, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0, 0.0], [0.20999999999999999, 0.0, 0.0, 0.0], [0.0, 0.26000000000000001, 0.0, 0.13], [0.27000000000000002, 0.0, 0.0, 0.0], [0.14000000000000001, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0, 0.16], [0.13, 0.0, 0.0, 0.0], [0.27000000000000002, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0, 0.0], [0.14000000000000001, 0.0, 0.0, 0.0], [0.17000000000000001, 0.0, 0.0, 0.0], [0.0, 0.0, 0.27000000000000002, 0.0], [0.14000000000000001, 0.0, 0.0, 0.0], [0.0, 0.0, 0.55000000000000004, 0.0], [0.40999999999999998, 0.0, 0.0, 0.0], [0.29999999999999999, 0.0, 0.11, 0.0], [0.0, 0.14000000000000001, 0.0, 0.0], [0.16, 0.0, 0.0, 0.0], [0.27000000000000002, 0.0, 0.0, 0.0], [0.27000000000000002, 0.0, 0.0, 0.0], [0.0, 0.33000000000000002, 0.0, 0.0], [0.27000000000000002, 0.0, 0.0, 0.0], [0.0, 0.0, 0.19, 0.0], [0.0, 0.0, 0.0, 0.0], [0.16, 0.0, 0.0, 0.0], [1.1000000000000001, 0.0, 0.0, 0.0], [0.17999999999999999, 0.0, 0.0, 0.0], [0.14000000000000001, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0, 0.0], [0.16, 0.0, 0.0, 0.0], [0.27000000000000002, 0.0, 0.0, 0.0], [0.0, 0.0, 0.42000000000000004, 0.0], [0.27000000000000002, 0.0, 0.0, 0.0], [0.17999999999999999, 0.0, 0.0, 0.0], [0.14000000000000001, 0.0, 0.0, 0.0], [0.46999999999999997, 0.0, 0.0, 0.0], [0.68000000000000005, 0.0, 0.0, 0.0], [0.0, 0.0, 0.26000000000000001, 0.13], [0.0, 0.0, 0.0, 0.0], [0.71999999999999997, 0.0, 0.080000000000000002, 0.0], [0.14000000000000001, 0.0, 0.0, 0.0], [0.13, 0.0, 0.0, 0.0], [0.0, 0.0, 0.66000000000000003, 0.0], [0.0, 0.0, 0.0, 0.0], [0.58999999999999997, 0.0, 0.0, 0.0], [0.52000000000000002, 0.0, 0.26000000000000001, 0.0], [0.40999999999999998, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0, 0.17999999999999999], [0.66000000000000003, 0.0, 0.0, 0.0], [0.14999999999999999, 0.0, 0.0, 0.0], [0.0, 0.0, 0.19, 0.0], [0.0, 0.0, 0.33000000000000002, 0.0], [0.0, 0.0, 0.0, 0.27000000000000002], [0.0, 0.0, 0.0, 0.0], [0.14999999999999999, 0.0, 0.0, 0.0], [0.17000000000000001, 0.0, 0.0, 0.0], [0.25, 0.0, 0.0, 0.0], [0.27000000000000002, 0.0, 0.0, 0.0], [0.0, 0.0, 0.17999999999999999, 0.0], [0.33000000000000002, 0.0, 0.0, 0.0], [0.17000000000000001, 0.0, 0.0, 0.0], [0.0, 0.0, 0.22, 0.10000000000000001], [0.46999999999999997, 0.0, 0.0, 0.0], [0.48999999999999999, 0.0, 0.0, 0.0], [0.46999999999999997, 0.0, 0.0, 0.0], [1.1000000000000001, 0.0, 0.0, 0.0], [0.51000000000000001, 0.0, 0.0, 0.0], [0.55000000000000004, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0, 0.16], [0.0, 0.0, 0.55000000000000004, 0.0], [0.0, 0.0, 0.27000000000000002, 0.0], [0.25, 0.0, 0.0, 0.0], [0.14999999999999999, 0.0, 0.0, 0.0], [0.68000000000000005, 0.0, 0.0, 0.0]]

Figure IV.6 : Les résultats du modèle PLSA sur une matrice TF-IDF pour l'application de Diagnostic Médical à Distance

Et pour les mesures de validations

Concept1 precision: 0.0	Concept3 precision: 0.307692307692
Concept1 recall: 0.0	Concept3 recall: 0.108108108108
Concept1 F-measure: 0.0	Concept3 F-measure: 0.16
Concept2 precision: 0.4571428571	Concept4 precision: 0.166666666667
Concept2 recall: 0.744186046512	Concept4 recall: 0.142857142857
Concept2 F-measure: 0.56637168	Concept4 F-measure: 0.153846153846

ConceptAll micro_avg_precision: 0.324786324786
ConceptAll micro_avg_recall: 0.387755102041
ConceptAll micro_avg_F-measure: 0.353488372093

Figure IV.7 : Les mesures de validations pour PLSA (TF-IDF) Diagnostic

Le taux de classification qu'on a ou obtenir en utilisant le modèle PLSA sur une matrice TF-IDF est d'environ 35.34%.

Voici le récapitulatif des résultats trouvés en considérant le modèle PLSA.

Méthode	Extraction de Concept		
Application	Consultation Scolaire		Diagnostique Médicale
Modèle	PLSA		PLSA
Type de matrice	Booléenne	TF-IDF	TF-IDF
Paramètres	K=4	K=4	K=4
Précision (%)	40.62	33.33	32.47
Rappel (%)	52	44	38.77
F1[4] (%)	45.61	37.93	35.34

Table IV.3 : Résultats obtenus par la méthode Extraction de Concepts avec les applications Consultation Scolaire et Diagnostic Médical

Le Test du SyCE-CHM actuel sur l'application de Consultation Scolaire a permis d'avoir des taux de classifications de l'ordre de 45.61% en utilisant une matrice d'occurrence Booléenne et un taux de 37.93% en pondérant cette matrice par le poids TF-IDF.
Pour l'application de Diagnostic Médicale à Distance, on a eu des taux de reconnaissances de 35.34% en utilisant la matrice TF-idf.
En comparant ces résultats avec ceux du chapitre 3, on remarque que le système vu précédemment (articulé sur la LSA) nous a données les meilleures classes.

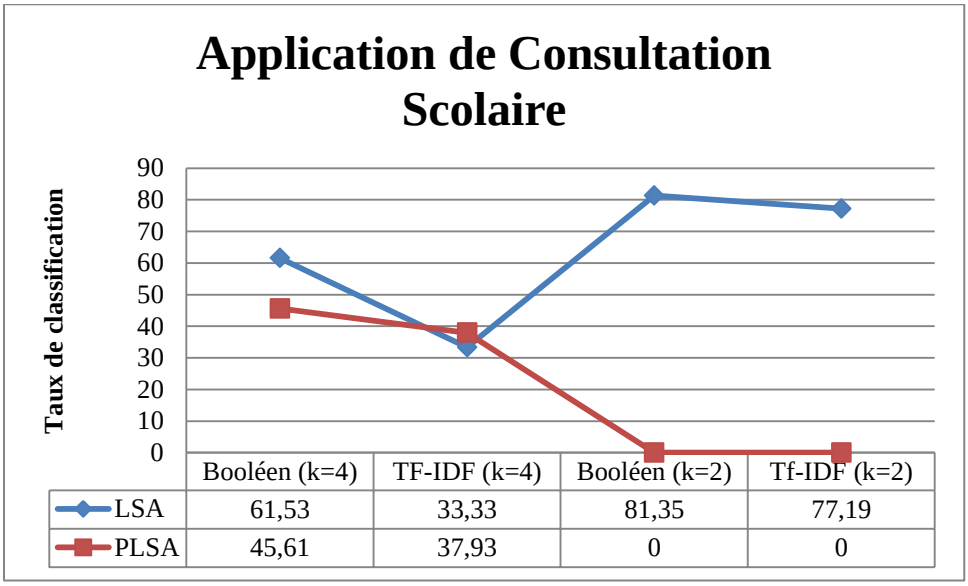


Figure IV.8 : Taux de classification pour l'application de Consultation Scolaire vis-à-vis les différents modèles

[4] La F-measure de Van Rijsbergen [81] dite aussi efficacité globale,

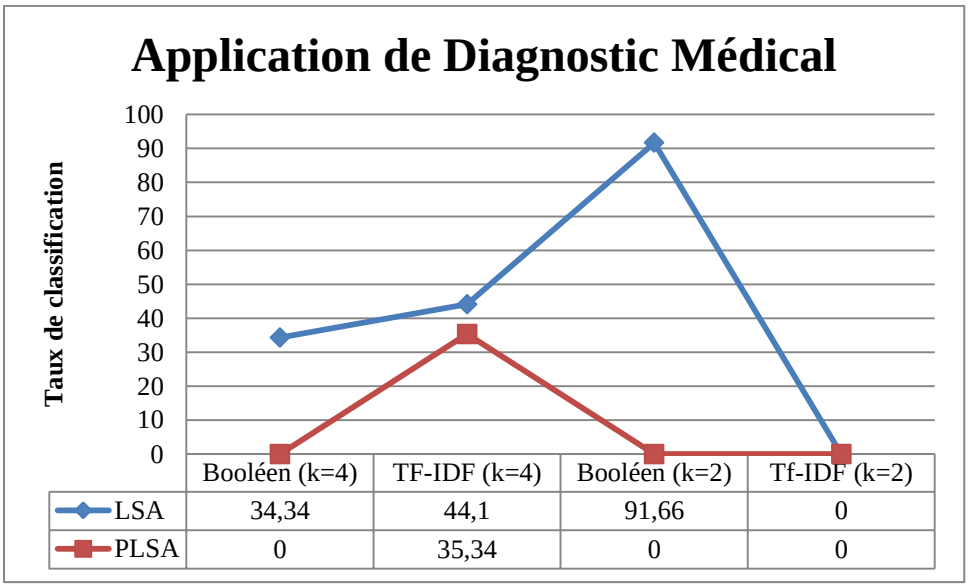


Figure IV.9 : Taux de classification pour l'application de Diagnostic Médical vis-à-vis les différents modèles

Les deux graphes précédents représentent la variation du taux de classification pour les deux applications en considérants les différents modèles (LSA et PLSA).

Ce qu'on peut déduire est qu'il n'y a pas vraiment une configuration optimal à considérer pour notre système afin d'extraire les différents concepts à partir du corpus de phrase. Donc un compromis est mis en jeu, et ce, en considérant les deux modèles pour la phase d'apprentissage. Et ensuite faire une comparaison entre les concepts trouvés, appliquer un petit ajustement afin de récupérer des concepts significatives.

IV.2.5. Pré-Conclusion

Á propos de ce SyCE-CHM actuel basé sur la méthode d'extraction de concept utilisant soit le modèle PLSA et/ou la LSA pour l'apprentissage et un moteur d'inférence lors de celle du test, on peut clairement confirmer que pour n'importe quelles systèmes d'interrogation de base de donnée utilisant comme vecteur le texte, cette architecture est la meilleur (avec quelques limitations à surpasser), du fait que tous ce qu'on aura à faire est de changer la base de donnée ainsi que le moteur d'inférence.

Cette confirmation n'est pas vraie pour le cas des systèmes de diagnostics à cause de leurs richesses linguistiques et l'impossibilité de concevoir un moteur d'inférence fiable lors de la phase de test.

D'une autre part, ce qu'on espère vraiment est la généralisation du système de comparaison pour qu'il soit multilingue et multi-domaines. En bref, un seul système, une seul architecture et plusieurs domaines d'applications. C'est pourquoi, une autre architecture est sollicitée. C'est ce qu'on va voir après.

IV.3. SyCE-CHM Multilingue et multi-domaine

Ceci dit avoir recours à de nouvelles connaissances pour atteindre cette performance. Ces derniers seront acquis à partir des réactions du système cognitif humaines.

IV.3.1. Modèle Cognitif

Si nous examinons les études faites sur le système cognitif humain, on se rend compte que dans la majorité des cas comprendre une phrase, un paragraphe ou un texte ne passe pas obligatoirement par la compréhension de chacun des mots, qui implique la possibilité de comprendre une phrase, un paragraphe ou un texte rien qu'en identifiant son sujet ou thème.
L'humain peut comprendre donc une phrase de deux façons :

- Soit comprendre le sens de la phrase en combinant le sens des différents constituants (mots) après avoir découper la phrase en mots. « Base du premier système SyCE-CHM proposé»
- Soit comprendre le sens d'une phrase en identifiant seulement le sujet ou thème,

Exemple :

Ce texte parle, de l'effet de la sécheresse sur l'économie du pays.

En final, ceci nous ramènera à une nouvelle branche des systèmes de fouilles de données qui est l'extraction du thème ou plus généralement la classification des phrases par thème ou sujets commun.

IV.3.2. Modèle Statistique

Notre nouveau système SyCE-CHM va dépendre d'une nouvelle méthode qui n'est plus la méthode d'extraction de concept, mais plutôt de la méthode de classification de phrases de note corpus.
Ceci engendrera une modification plus ou moins légère de l'architecture précédente.

IV.3.2.1. Phase d'apprentissage

Cette phase garde la même topologie, qui est une suite d'analyses, commençant par une analyse structurelle (pour diviser le texte en phrase), une analyse lexicale (pour segmenter ces phrases et filtrer les mots insignifiants) et enfin une analyse sémantique qui sert à extraire les concepts pas des mots mais plutôt des phrases en utilisant deux approches :

- Modèle LSA et Classificateur k-moyenne.
- Modèle PLSA.

L'utilisation de ces deux approches va nous permettre une comparaison de ces deux modèles en face de cette nouvelle connaissance, et on pourra conclure et définir celui qui est le plus adapté à chacune des applications.

Le premier système nous a permis de générer en sortie des concepts de mots, alors comment peut-on avoir des concepts de phrases avec les mêmes constituants ?
La réponse se situe au niveau de la définition de la LSA[5] ou plus précisément de la SVD, du fait qu'après avoir effectué la décomposition de la matrice d'occurrence A en trois matrices U, S et V^T selon la loi:

$$A = U * S * V^T \quad \text{(IV.9)}$$

Où : **A** : est la matrice d'occurrences terme-phrases.
U : la matrice contenant les vecteurs de mots.
S : la matrice des poids s des concepts.
V^T : La matrice contenant les vecteurs de phrases.

Donc, c'est au niveau des échantillons du classificateur qu'il y aura un changement par rapport à la méthode précédente. En effet, on n'utilisera plus la matrice U comme entrée du classificateur k-moyenne mais plutôt la matrice V^T des phrases.
Ce qui nous ramènera en sortie en face de concepts de phrases ou plus précisément des catégories de phrases (on considère dorénavant le mot catégorie au lieu de celui de concept). Ces catégories contiendront des phrases avec non plus un sens commun mais un sujet commun.

IV.3.2.2. Phase de test

Cette phase utilise le classificateur supervisé de Bayes. Ce dernier recevra en entrée une phrase en langage naturel et fournira en sortie la classe d'appartenance de cette phrase et nous rappelons que la classe est un ensemble de phrase ayant le même sujet ou même thème.

IV.3.2.2.1. Classificateur supervisé de Bayes

La classification Bayésiens naïve de textes est une approche probabiliste de classification simple. Cette approche est basée sur un modèle probabiliste dérivant du théorème de Bayes qui fait l'hypothèse que les mots qui apparaissent dans un document sont indépendants les uns des autres. Ce qui n'est pas tout à fait le cas dans la pratique.
De manière abstraite, le modèle probabiliste pour un classificateur Bayésiens est un modèle conditionnel. Il se base sur la règle de bayes qui s'énonce de la manière suivante :

$$P(A|B_1, B_2, \dots, B_n) = \frac{P(B_1, B_2, \dots, B_n|A) \times P(A)}{P(B_1, B_2, \dots, B_n)} \quad \text{(IV.10)}$$

La probabilité d'avoir l'évènement A étant donné $B_1, B_2, \dots, B_n$, est donné par le rapport entre la probabilité d'avoir les évènements $B_1, B_2, \dots, B_n$ étant donné A et la probabilité que $B_1, B_2, \dots, B_n$ se soient produits. Tant que le dénominateur ne dépend pas de l'évènement A,

[5] Idem pour la PLSA, à la différence que celle la considère plutôt la probabilité p(s/c) au lieu de p(w/c). Cette dernière nous fournira la probabilité d'appartenance de chaque mots à chaque concepts alors que la première la probabilité d'appartenance de chaque phrases à chaque concepts.

on peut considérer la probabilité $P(B_1, B_2, \dots, B_n)$ comme étant constante. Le numérateur peut être écrit encore de la manière suivante :

$$P(B_1, B_2, \dots, B_n|A) \times P(A) \quad \text{(IV.11)}$$
$$= P(A, B_1, B_2, \dots, B_n)$$
$$= P(A) \times P(B_1|A) \times P(B_2, \dots, B_n|A, B_1)$$
$$= P(A) \times P(B_1|A) \times P(B_2|A, B_1) \times P(B_3, \dots, B_n|A, B_1, B_2)$$

La décomposition de $\mathrm{P(A, B_1, B_2, \dots, B_n)}$ se termine lorsqu'on a parcouru l'ensemble des classes $\mathrm{B_1, B_2, \dots, B_n}$.

Le caractère "naïf" de ce théorème vient du fait qu'on suppose l'indépendance des différentes classes B_i, B_j. Ce qui en d'autres termes se traduit par :

$$P(B_i|A, B_j) = P(B_i|A) \quad \text{(IV.12)}$$

Cette hypothèse permet également d'écrire :

$$P(A, B_1, B_2, \dots, B_n) = P(A) \times P(B_1|A) \times P(B_2|A) \times \dots \times P(B_n|A) \quad \text{(IV.13)}$$
$$= P(A) \prod_{i=1}^{n} P(B_i|A)$$

Ce théorème a beaucoup d'applications dans le domaine du traitement de l'information notamment en traitement de la parole, traitement des images et bien d'autres. Nous nous limiterons ici à l'application de ce théorème à la catégorisation de phrases.

Supposons que nous disposons de n catégories C de phrases, le problème revient à déterminer à quelle catégorie C_i sera associée une phrase P revient à calculer la probabilité d'appartenance de cette phrase à la catégorie C_i. On peut calculer cette probabilité de la façon suivante :

(IV.14)

$$P(C_i|P) = \frac{P(P|C_i) \times P(C_i)}{P(P)}$$

Dans cette formule, $P(C_i|P)$ représente la probabilité d'appartenance de la phrase P à la catégorie C_i qui peut être également déterminée en évaluant la fréquence d'apparition des termes de la phrase P qui sont associés à la catégorie $\mathrm{C_i}$.

$P(P|C_i)$ est la probabilité selon laquelle, pour une catégorie donnée, les termes de la phrase P sont associés à la catégorie C_i. $P(C_i)$ est la probabilité qui associe la phrase P à la catégorie C_i indépendamment du contenu de la phrase. $P(P)$ est la probabilité propre de la phrase P.

Pour réellement déterminer à quelle catégorie une phrase appartient, il faut calculer $P(C_i|P)$ pour chacune des catégories. Étant donné que $P(P)$ reste constant pour toutes les catégories, déterminer $P(C_i|P)$ se résume juste au calcul de $P(P|C_i) \times P(C_i)$.

Comment calcule-t-on ces probabilités ?

En considérant que la phrase P est composée d'un ensemble de termes que nous noterons $t_1, \ldots, t_n$, calculer $P(P|C_i)$ reviendrait à calculer le produit des probabilités d'apparition de chaque terme t_i dans la catégorie Ci. Ce calcul se justifie par l'hypothèse selon laquelle tous les termes apparaissent indépendamment les unes des autres dans une phrase. Ce qui permet finalement d'écrire :

$$P(P|C_i) = P(t_1|C_i) \times P(t_2|C_i) \times \ldots \times P(t_n|C_i) \qquad \text{(IV.15)}$$

Pour chacune des catégories, $P(t_i|C_i)$ est le rapport entre le nombre de fois que le terme t_i apparaît dans la catégorie Ci et le nombre total de mots que comprend la catégorie Ci. P(Ci) est calculé en divisant le nombre total de termes pour la catégorie Ci par la somme du nombre total de termes dans toutes les catégories. D'où la formulation suivante :

$$P(C_i|P) = P(t_1|C_i) \times \ldots \times P(t_n|C_i) \times P(C_i) \qquad \text{(IV.16)}$$

Ce calcul est effectué pour chaque catégorie et on considère la probabilité la plus élevée pour choisir à quelle catégorie sera associée la phrase qu'on souhaite classer.

Le calcul ainsi présenté se justifie par l'hypothèse selon laquelle tous les termes apparaissent indépendamment les unes des autres dans une phrase. D'où le caractère naïf de la classification.

La probabilité déjà vus $P(ti|Ci) = N(ti, ci)/\sum_j N(tj, ci)$ est très limitée envers les nouveaux termes car $N(ti, ci)$ **vaut 0 ⇔** $P(ti|Ci)$ **= 0 ⇔ Pas de Classe**

Donc, pour remédier à ce problème on va utiliser un estimateur « M-estimâtes » qui est définie par : (IV.17)

$$P(ti|Ci) = \frac{n_c + mp}{n + m}$$

Où : **n** : la fréquence d'apparition du terme t_i

n_c : le nombre de fois ou t_i apparaît avec Ci « **N (ti, ci)** »

p : probabilité à priori de $\boldsymbol{P(ti|Ci)}$

m : une constante

À la sortie de ce classificateur on aura l'action approprié à exécuter tel que: « chaque concept correspond à une action précis ». (Voir Annexe C)

IV.3.3. Résultat après l'intégration de ce système

Pour l'application de Consultation Scolaire, on a eu un taux de reconnaissance de 72.72% en utilisant le modèle LSA (Figure IV.10) et pour celle du Diagnostic Médicale 51.85% en utilisant le modèle PLSA sur une matrice d'occurrences pondéré par le poids TF-IDF.[6]

[6] Les résultats complets seront présentés ainsi que la liste des concepts en annexe C. Le source code serait présenté en annexe D.

IV.3.3.1. Application de Consultation Scolaire

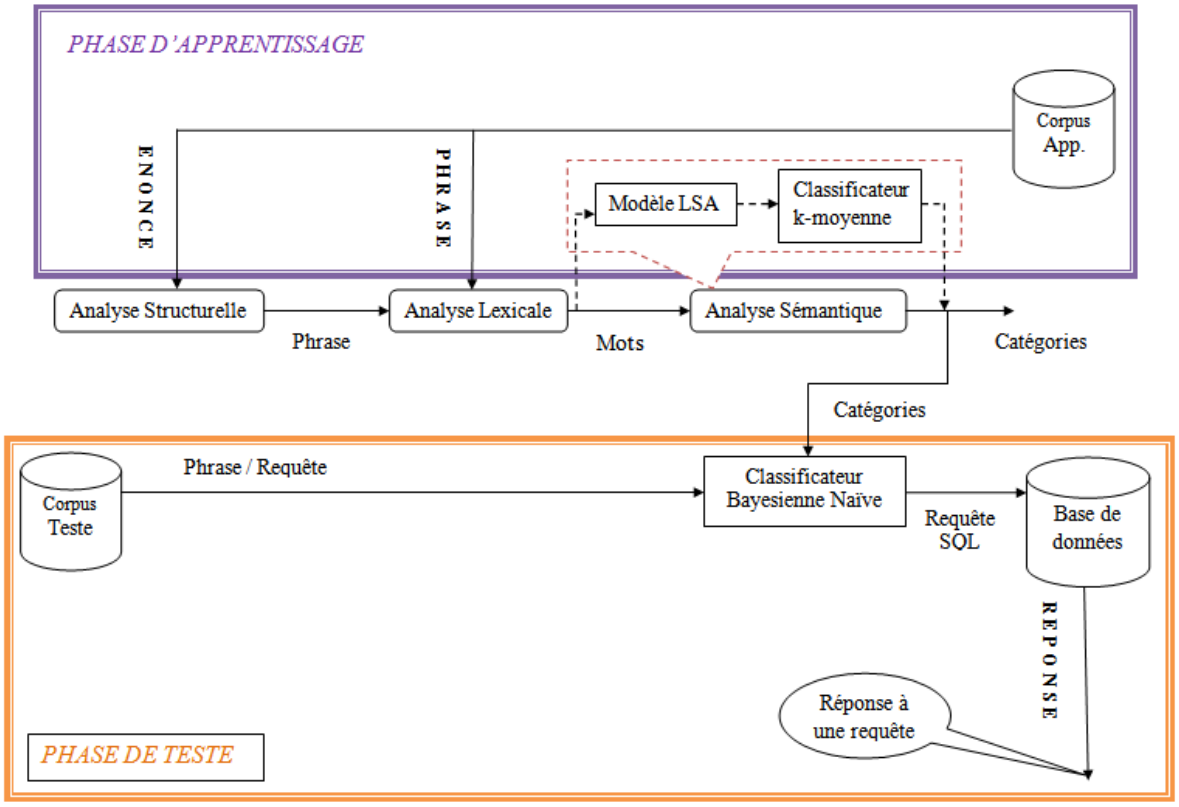


Figure IV.10 : Architecture Générale du SyCE-CHM basé sur l'approche de la Classification de Texte pour l'application de Consultation Scolaire

Pour cette application, on remarque que la seule chose qui s'est changé est le type d'échantillon (les phrases au lieu des termes). Dans la phase de test, un classificateur bayésien est considéré.

Nous allons rappeler que le nombre de catégories (classes de phrases) est fixé à **3**. Et cela, à cause des trois thèmes apparus, qui sont la note, le certificat et le diplôme.

Voici la liste des catégories de phrases annotées manuellement :

Catégorie	Liste des phrases formant la catégorie
Note	puis-je avoir mon relevée de note. puis-je me renseigner sur ma note. quelle note ai-je obtenu en examen. donne-moi ma note d'examen. pouvez-vous me montrer mon relevée de note. je voudrais savoir ma note. je veux bien avoir la moyenne du module de math. je veux savoir le résultat d'examen du physique. combien j'ai eu en chimie. j'aimerais bien avoir ma note du math. veuillez me donner la note du module de chimie. donner moi les résultats du module de math et informatique. résultat d'examen du module d'informatique.

Diplôme	donne-moi mon diplôme. je veux mon diplôme. puis-je avoir mon diplôme. pouvez-vous me donner mon diplôme.
Certificat	je souhaite avoir mon certificat de scolarité. donner moi mon certificat de scolarité. veux-tu me donner mon certificat STP. j'aimerai bien avoir mon certificat de scolarité. est-ce que je peux avoir mon certificat de scolarité.

Table IV.4 : Listes des Catégories Manuelles pour l'application de Consultation Scolaire

En tenant compte des modèles LSA et PLSA sur deux types de matrices booléenne et TF-IDF avec k=3, on aura les résultats suivants :

LSA	
Booléenne	TF-IDF
x = 1, 3, 4, 11, 12, 18, y = 2, 5, 6, 7, 8, 10, 13, 14, 15, 16, 17, 20,21 z = 9, 19	x = 1, 3, 4, 9, 10, 11, 12, 15, 16, 17, 18, 19, 20, 21 y = 2, 6, 8, 14 z = 5, 7,13
PLSA	
Booléenne	TF-IDF
x=2,3,4,5,6,7,10,11,14,15,16,18,19,20,21 y=1,8, z=3, 9, 12, 13, 17,	x=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,20 y= 19, z=21
Manuelle : x=1,3,4,9,11,12,15,16,17,18,19,20,21 y=2, 6, 14 z=5, 7, 10, 13	

Table IV.5 : Les résultats des différents modèles pour l'application de scolarité

Voici un exemple des listes des concepts correspondantes à ces résultats sont présentées dans ce qui suit :

LSA (Booléenne)	
Catégorie	**Liste des phrases formant la catégorie**
Catégorie n°1	1-puis je avoir mon relevée de note 3-puis je me renseigner sur ma note 4-quelle note ai-je obtenu en examen 11-pouvez-vous me montrer mon relevée de note 12-je voudrais savoir ma note 18-J'aimerais bien avoir ma note du math
Catégorie n°2	2-donne-moi mon diplôme 5-je souhaite avoir mon certificat de scolarité 6-je veux mon diplôme 7-donnez-moi mon certificat de scolarité 8-puis j'avoir mon diplôme

	9-donne-moi ma note d'examen 10 veux-tu me donner mon certificat STP 13-j'aimerais bien avoir mon certificat de scolarité 14-pouvez-vous me donner mon diplôme 15-je veux bien avoir la moyenne du module de math 16-je veux savoir le résultat d'examen du physique 17-combien j'ai eu en chimie 20-donne-moi les résultats des modules de math et d'informatique 21-quelle est le résultat d'examen du module d'informatique
Catégorie n°3	9- donne-moi ma note d'examen 19-veuillez me donner la note du module de chimie

Table IV.6 : Listes des Catégories trouvés par la méthode LSA sur une matrice booléenne pour l'application de Consultation Scolaire

Passons maintenant aux liste de catégories trouvés par la méthode PLSA sur une matrice booléenne.

PLSA (Booléenne)	
Catégorie	**Liste des phrases formant la catégorie**
Catégorie n°1	2-donne-moi mon diplôme 4-quelle note ai-je obtenu en examen 5-je souhaite avoir mon certificat de scolarité 6-je veux mon diplôme 7-donnez-moi mon certificat de scolarité 10 veux-tu me donner mon certificat STP 11-pouvez-vous me montrer mon relevée de note 14-pouvez-vous me donner mon diplôme 15-je veux bien avoir la moyenne du module de math 16-je veux savoir le résultat d'examen du physique 18-J'aimerais bien avoir ma note du math 19-veuillez me donner la note du module de chimie 20-donne-moi les résultats des modules de math et d'informatique 21-quelle est le résultat d'examen du module d'informatique
Catégorie n°2	1-puis je avoir mon relevée de note 8-puis j'avoir mon diplôme
Catégorie n°3	3-puis je me renseigner sur ma note 9-donne-moi ma note d'examen 12-je voudrais savoir ma note 13-j'aimerais bien avoir mon certificat de scolarité 17-combien j'ai eu en chimie

Table IV.7 : Listes des Catégories trouvés par la méthode PLSA sur une Matrice Booléenne pour l'application de Consultation Scolaire

Le calcul des mesures de validations a permis de récupérer les informations suivantes :

LSA	
Booléenne	
Catégorie1 precision: 0.0	Catégorie 2 precision: 0.384615384615
Catégorie 1 recall: 0.0	Catégorie 2 recall: 0.384615384615
Catégorie 1 F-measure: 0	Catégorie 2 F-measure: 0.384615384615
Catégorie 3 precision: 0.0	
Catégorie 3 recall: 0.0	
Catégorie 3 F-measure: 0	
Catégorie All micro_avg_precision: 0.238095238095	
Catégorie All micro_avg_recall: 0.25	
Catégorie All micro_avg_F-measure: 0.243902439024	
TF-IDF	
Catégorie 1 precision: 0.75	Catégorie 2 precision: 0.928571428571
Catégorie 1 recall: 1.0	Catégorie 2 recall: 1.0
Catégorie 1 F-measure: 0.857142857143	Catégorie 2 F-measure: 0.962962962963
Catégorie 3 precision: 1.0	
Catégorie 3 recall: 0.75	
Catégorie 3 F-measure: 0.857142857143	
Catégorie All micro_avg_precision: 0.95	
Catégorie All micro_avg_recall: 0.95	
Catégorie All micro_avg_F-measure: 0.95	
PLSA	
Booléenne	
Catégorie 1 precision: 0.2	Catégorie 2 precision: 0.6
Catégorie 1 recall: 0.25	Catégorie 2 recall: 0.692307692308
Catégorie 1 F-measure: 0.222222222222	Catégorie 2 F-measure: 0.642857142857
Catégorie 3 precision: 0.0	
Catégorie 3 recall: 0.0	
Catégorie 3 F-measure: 0	
Catégorie All micro_avg_precision: 0.454545454545	
Catégorie All micro_avg_recall: 0.5	
Catégorie All micro_avg_F-measure: 0.47619047619	
TF-IDF	
Catégorie 1 precision: 0.0	Catégorie 2 precision: 0.578947368421
Catégorie 1 recall: 0.0	Catégorie 2 recall: 0.846153846154
Catégorie 1 F-measure: 0	Catégorie 2 F-measure: 0.6875
Catégorie 3 precision: 0.0	
Catégorie 3 recall: 0.0	
Catégorie 3 F-measure: 0	
Catégorie All micro_avg_precision: 0.52380952381	
Catégorie All micro_avg_recall: 0.55	
Catégorie All micro_avg_F-measure: 0.536585365854	

Table IV.8 : Les mesures de validations pour l'application scolaire dans la classification de phrases

Voici le récapitulatif des résultats trouvés :

Méthode	Classification de Phrase			
Application	Consultation Scolaire			
Modèle	LSA		PLSA	
Paramètres (k=3)	Boo-léenne	TF-IDF	Booléenne	TF-IDF
Précision (%)	23.80	95	45.45	52.38
Rappel (%)	25	95	50	55
F1 (%)	24.39	95	47.61	53.65

Table IV.9 : Résultats obtenus par la méthode de la Classification de phrases avec l'application de Consultation Scolaire

Á partir de ces résultats, et pour l'application de consultation scolaire, on remarque que la meilleur approche est celle englobant la LSA sur une matrice pondérée par la TF-IDF avec un taux de classification de 95%.

IV.3.3.2. Application de Diagnostic Médicale à Distance

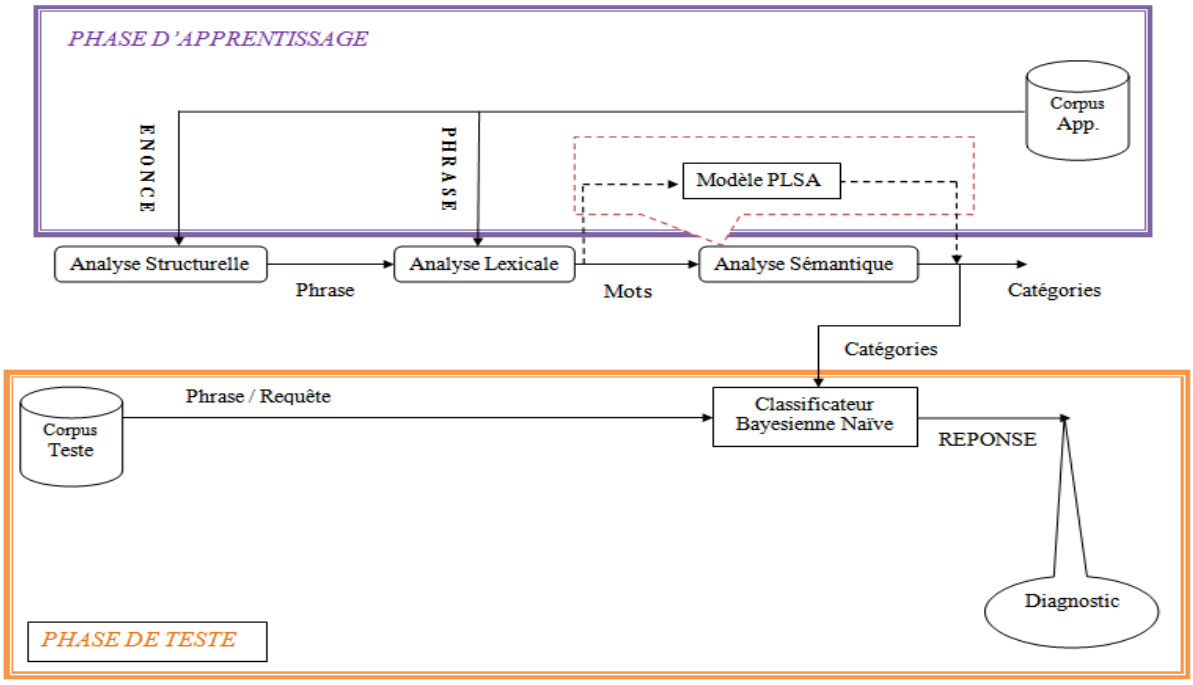


Figure IV.11 : Architecture Générale du SyCE-CHM basé sur l'approche de la Classification de Texte pour l'application de Diagnostic Médicale à Distance (SDMD)

Voici la liste des catégories annotés manuellement pour l'application de diagnostic médical.

Catégorie	Liste des phrases formant la catégorie
Allergie	j'ai eu des difficultés respiratoire, à parler ainsi qu'à avaler. j'ai des douleurs à l'estomac, vomissement et de la diarrhée. je me sens faible, des accélérations du rythme cardiaque, de l'anxiété, détresse et perte de conscience.
Bronchite aiguë	j'ai une légère fièvre, des maux de tête, des maux de gorge et une voix rauque. je me sens de la fièvre, des maux de tête, des maux de gorge, une voix rauque ainsi des complications respiratoires.
Diarrhée	j'ai des selles fréquentes et liquides avec perte importante de liquide. j'ai de la fièvre, des vomissements et des douleurs abdominales.
Fatigue	je me sens souvent fatigué même en ayant bien dormi et on a de la peine à se lever le matin. j'ai des diminutions de l'énergie en général. j'ai des difficultés à se concentrer.
Grippe	j'ai une fièvre élevé, des maux de tête importants, une fatigue extrême et des douleurs dans les articulations. j'ai une toux sèche, des maux de gorge, un rhume ainsi que des nausées. j'ai de la fièvre à un niveau élevé, et également avoir des frissons à cause de cette fièvre. j'ai des douleurs au niveau des articulations. j'ai de forts maux de tête. j'ai des courbatures. j'ai des Écoulement nasal, éternuement, larmoiement des yeux. j'ai une perte d'appétit, fièvre, maux de tête et des irritations de la gorge.
Stress	j'ai des troubles digestifs, des ulcères, des nausées et vomissements. je m'énerve rapidement j'ai des problèmes cardiaques (accélération du rythme cardiaque). j'ai divers troubles du sommeil. j'ai des troubles psychiques : nervosité, colère excessive et inhabituelle, crise d'angoisse, peurs excessives (phobies), dépression. j'ai un problème de chute des cheveux. j'ai des problèmes de concentration.
Anémie	je suis fatigue, pâleur de la peau et des muqueuses, des palpitations ainsi que des essoufflements. j'ai des maux de tête, des vertiges et des bourdonnements d'oreille.

Table IV.10 : Listes des Catégories Manuelles pour l'application de Diagnostic Médical à Distance

En considérant la méthode LSA sur des matrices TF-IDF et Booléenne, on aura les résultats suivants :

LSA (TF-IDF)	
Catégorie	**Liste des phrases formant la catégorie**
Catégorie n°1	16-j'ai des courbatures. 20-je m'énerve rapidement
Catégorie n°2	1-j'ai eu des difficultés respiratoire, a parler ainsi qu'à avaler.

	2-j'ai des douleurs à l'estomac, vomissement et de la diarrhée. 8-je me sens souvent fatigue même en ayant bien dormi et on a de la peine à se lever le matin. 9-j'ai des diminutions de l'énergie en général. 10-j'ai des difficultés à se concentrer 14-j'ai des douleurs au niveau des articulations 17-j'ai des écoulements nasal, éternuement, larmoiement des yeux 19-j'ai des troubles digestifs, des ulcères, des nausées et vomissements. 22-j'ai divers troubles du sommeil 23-j'ai des troubles psychiques : nervosité, colère excessive et inhabituelle , crise d'angoisse, peurs excessives (phobies), dépression. 24-j'ai un problème de chute des cheveux. 26-je suis fatigue, pâleur de la peau et des muqueuses, des palpitations ainsi que des essoufflements
Catégorie n°3	21-j'ai des problèmes cardiaques (accélération du rythme cardiaque) 25-j'ai des problèmes de concentration
Catégorie n°4	6-j'ai des selles fréquentes et liquides avec perte importante de liquide. 18-j'ai une perte d'appétit, fièvre, maux de tête et des irritations de la gorge.
Catégorie n°5	4-j'ai une légère fièvre, des maux de tête, des maux de gorge et une voix rauque. 15-j'ai de forts maux de tête.
Catégorie n°6	3-je me sens faible, des accélérations du rythme cardiaque, de l'anxiété, détresse et perte de conscience.
Catégorie n°7	5-je me sens de la fièvre, des maux de tête, des maux de gorge , une voix rauque ainsi des complications respiratoires . 11-j'ai une fièvre élevé, des maux de tête importants, une fatigue extrême et des douleurs dans les articulations. 12-j'ai une toux sèche, des maux de gorge, un rhume ainsi que des nausées 13-j'ai de la fièvre a un niveau élevé, et également avoir des frisons a cause de cette fièvre. 27-j'ai des maux de tête, des vertiges et des bourdonnements d'oreille.

Table IV.11 : Listes des Catégories trouvés par la méthode LSA sur une matrice TF-IDF par l'approche de classification de texte, application de diagnostic

LSA (Booléenne)	
Catégorie	**Liste des phrases formant la catégorie**
Catégorie n°1	1-j'ai eu des difficultés respiratoire, a parler ainsi qu'à avaler. 9-j'ai des diminutions de l'énergie en général. 10-j'ai des difficultés à se concentrer 16-j'ai des courbatures. 17-j'ai des écoulements nasal, éternuement, larmoiement des yeux 19-j'ai des troubles digestifs, des ulcères, des nausées et vomissements. 20-je m'énerve rapidement 22-j'ai divers troubles du sommeil 23-j'ai des troubles psychiques : nervosité, colère excessive et inhabituelle , crise d'angoisse, peurs excessives (phobies), dépression. 24-j'ai un problème de chute des cheveux.

	25-j'ai des problèmes de concentration
Catégorie n°2	2-j'ai des douleurs à l'estomac, vomissement et de la diarrhée. 7-j'ai de la fièvre, des vomissements et des douleurs abdominales. 8-je me sens souvent fatigue même en ayant bien dormi et on a de la peine à se lever le matin. 13-j'ai de la fièvre a un niveau élevé, et également avoir des frisons a cause de cette fièvre. 14-j'ai des douleurs au niveau des articulations 26-je suis fatigue, pâleur de la peau et des muqueuses, des palpitations ainsi que des essoufflements
Catégorie n°3	3-je me sens faible, des accélérations du rythme cardiaque, de l'anxiété, détresse et perte de conscience. 12-j'ai une toux sèche, des maux de gorge, un rhume ainsi que des nausées.
Catégorie n°4	18-j'ai une perte d'appétit, fièvre, maux de tête et des irritations de la gorge.
Catégorie n°5	4-j'ai une légère fièvre, des maux de tête, des maux de gorge et une voix rauque. 5-je me sens de la fièvre, des maux de tête, des maux de gorge , une voix rauque ainsi des complications respiratoires .
Catégorie n°6	6-j'ai des selles fréquentes et liquides avec perte importante de liquide. 15-j'ai de forts maux de tête. 21-j'ai des problèmes cardiaques (accélération du rythme cardiaque) 27-j'ai des maux de tête, des vertiges et des bourdonnements d'oreille.
Catégorie n°7	11-j'ai une fièvre élevé, des maux de tête importants, une fatigue extrême et des douleurs dans les articulations.

Table IV.12 : Listes des Catégories trouvés par la méthode LSA sur une matrice Booléenne par l'approche de classification de texte, application de diagnostic

Et en considérant l'utilisation de la méthode PLSA sur une matrice TF-IDF, on aura les catégories suivantes :

PLSA (TF-IDF)	
Catégorie	**Liste des phrases formant la catégorie**
Catégorie n°1	25-j'ai des problèmes de concentration
Catégorie n°2	7-j'ai de la fièvre, des vomissements et des douleurs abdominales. 16-j'ai des courbatures. 18-j'ai une perte d'appétit, fièvre, maux de tête et des irritations de la gorge. 23-j'ai des troubles psychiques : nervosité, colère excessive et inhabituelle, crise d'angoisse, peurs excessives (phobies), dépression. 24-j'ai un problème de chute des cheveux. 27-j'ai des maux de tête, des vertiges et des bourdonnements d'oreille.
Catégorie n°3	21-j'ai des problèmes cardiaques (accélération du rythme cardiaque)
Catégorie n°4	2-j'ai des douleurs à l'estomac, vomissement et de la diarrhée. 9-j'ai des diminutions de l'énergie en général. 10-j'ai des difficultés à se concentrer 11-j'ai une fièvre élevé, des maux de tête importants, une fatigue extrême et des douleurs dans les articulations. 13-j'ai de la fièvre a un niveau élevé, et également avoir des frissons à

	cause de cette fièvre 14-j'ai des douleurs au niveau des articulations 23-j'ai des troubles psychiques : nervosité, colère excessive et inhabituelle, crise d'angoisse, peurs excessives (phobies), dépression.
Catégorie n°5	
Catégorie n°6	3-je me sens faible, des accélérations du rythme cardiaque, de l'anxiété, détresse et perte de conscience. 9-j'ai des diminutions de l'énergie en général. 17-j'ai des écoulements nasal, éternuement, larmoiement des yeux. 20-je m'énerve rapidement 22-j'ai divers troubles du sommeil
Catégorie n°7	1-j'ai eu des difficultés respiratoire, a parler ainsi qu'à avaler. 2-j'ai des douleurs à l'estomac, vomissement et de la diarrhée. 4-j'ai une légère fièvre, des maux de tête, des maux de gorge et une voix rauque. 5-je me sens de la fièvre, des maux de tête, des maux de gorge, une voix rauque ainsi des complications respiratoires. 6-j'ai des selles fréquentes et liquides avec perte importante de liquide. 8-je me sens souvent fatigue même en ayant bien dormi et on a de la peine à se lever le matin. 12-j'ai une toux sèche, des maux de gorge, un rhume ainsi que des nausées. 15-j'ai de forts maux de tête. 19-j'ai des troubles digestifs, des ulcères, des nausées et vomissements. 26-je suis fatigue, pâleur de la peau et des muqueuses, des palpitations ainsi que des essoufflements.

Table IV.13 : Listes des Catégories trouvés par la méthode PLSA sur une Matrice TF-IDF pour l'application de Diagnostic Médicale à Distance, par l'approche de classification de texte

Afin d'apprécier cette classification, on est obligée de passer par une étape de validation. Lors de cette étape on aura recours aux mesures de précision, rappel et F1.

LSA	
Booléenne	
Catégorie 1 precision: 0.5 Catégorie 1 recall: 0.333333333333 Catégorie 1 F-measure: 0.4 Catégorie 2 precision: 1.0 Catégorie 2 recall: 1.0 Catégorie 2 F-measure: 1.0 Catégorie 3 precision: 0.25 Catégorie 3 recall: 0.5 Catégorie 3 F-measure: 0.333333333333 Catégorie 4 precision: 0.166666666667 Catégorie 4 recall: 0.333333333333 Catégorie 4 F-measure: 0.222222222222	Catégorie 5 precision: 1.0 Catégorie 5 recall: 0.125 Catégorie 5 F-measure: 0.222222222222 Catégorie 6 precision: 0.545454545455 Catégorie 6 recall: 0.857142857143 Catégorie 6 F-measure: 0.666666666667 Catégorie 7 precision: 0.0 Catégorie 7 recall: 0.0 Catégorie 7 F-measure: 0
Catégorie All micro_avg_precision: 0.444444444444	

Catégorie All micro_avg_recall: 0.444444444444 Catégorie All micro_avg_F-measure: 0.444444444444	
TF-IDF	
Catégorie 1 precision: 1.0 Catégorie 1 recall: 0.333333333333 Catégorie 1 F-measure: 0.5 Catégorie 2 precision: 0.5 Catégorie 2 recall: 0.5 Catégorie 2 F-measure: 0.5 Catégorie 3 precision: 0.5 Catégorie 3 recall: 0.5 Catégorie 3 F-measure: 0.5 Catégorie 4 precision: 0.0 Catégorie 4 recall: 0.0 Catégorie 4 F-measure: 0.0	Catégorie 5 precision: 0.5 Catégorie 5 recall: 0.375 Catégorie 5 F-measure: 0.428571428571 Catégorie 6 precision: 0.333333333333 Catégorie 6 recall: 0.571428571429 Catégorie 6 F-measure: 0.421052631579 Catégorie 7 precision: 0.0 Catégorie 7 recall: 0.0 Catégorie 7 F-measure: 0
Catégorie All micro_avg_precision: 0.37037037037 Catégorie All micro_avg_recall: 0.384615384615 Catégorie All micro_avg_F-measure: 0.377358490566	
PLSA	
TF-IDF	
Catégorie 1 precision: 0.0 Catégorie 1 recall: 0.0 Catégorie 1 F-measure: 0.0 Catégorie 2 precision: 0.2 Catégorie 2 recall: 1.0 Catégorie 2 F-measure: 0.3333333 Catégorie 3 precision: 0.0 Catégorie 3 recall: 0.0 Catégorie 3 F-measure: 0.0 Catégorie 4 precision: 0.0 Catégorie 4 recall: 0.0 Catégorie 4 F-measure: 0.0	Catégorie 5 precision: 0.428571428571 Catégorie 5 recall: 0.375 Catégorie 5 F-measure: 0.4 Catégorie 6 precision: 0.4 Catégorie 6 recall: 0.285714285714 Catégorie 6 F-measure: 0.333333333333 Catégorie 7 precision: 0.166666666667 Catégorie 7 recall: 0.5 Catégorie 7 F-measure: 0.25
Catégorie All micro_avg_precision: 0.285714285714 Catégorie All micro_avg_recall: 0.296296296296 Catégorie All micro_avg_F-measure: 0.290909090909	

Table IV.14 : Le calcul des mesures pour l'application de Diagnostic

Dans ce récapitulatif, on va présenter les mesures de validation pour l'application de diagnostic médicale basé sur l'approche de classification de phrases en utilisant les modèles LSA et PLSA.
Avec le modèle LSA, on a considéré les deux types de matrices. Alors qu'avec la PLSA, on n'a considéré que la matrice TF-IDF (pour celle booléenne le calcul n'était pas faisable).

Méthode	Classification de Phrase		
Application	Diagnostic Médicale à Distance		
Modèle	PLSA	LSA	
Paramètres	K=7	K=7	K=7
Type de matrice	TF-IDF	Booléenne	TF-IDF
Précision (%)	28.57	44.44	37.03
Rappel (%)	29.62	44.44	38.46
F1 (%)	29.09	44.44	37.73

Table IV.15 : Résultats obtenus par la méthode de la Classification de phrases avec l'application de Diagnostic Médical à Distance (SDMD)

Á partir de ces résultats, on remarque que la meilleur configuration est celle du modèle LSA appliqué sur une matrice Booléenne, avec un taux de classification de 44.44%.

IV.3.3.3. Résultats expérimentales

Afin de valider l'architecture de ce système, une phase de test est introduite. Lors de cette phase, l'utilisateur est inviter à entrer sa requête (phrase), et le système aura la tâche de la comprendre ainsi réaliser l'action ou récupérer l'information voulue. Pour que le système puisse comprendre la requête de l'utilisateur, il aura recours à une classification de ces requêtes en thème, où chaque thème est associé avec une action précise.
Comme toujours, on va considérer deux applications.

a) Application de Consultation Scolaire

Voici un extrait de l'expérience mené :

Utilisateur 1	je veux savoir ma note	OK
	combien j'ai eu en television	OK
	j'aimerai bien voir mon relevé de note	OK
	donne moi la note de micro onde	OK
	est ce que j'ai la moyenne en champ	OK
	je veux connaitre mes resultats	OK
	excusez moi,je veux avoir ma certificat de scolarite	OK
	mes resultats sont elles acceptable	OK
	pouvez-vous me donner ma note de fibre optique	OK
	s'il vous plait ,donnez moi ma certificat de scolarité	OK
Utilisateur 2	donnez moi ma certificat scolarité	OK
	je veux savoir ma note en antenne	OK
	pouvez vous me donner ma certificat scolarite	OK
	je voulais connaitre ma note en fibre optique	OK
	je cherche ma resutlat	OK

	je voudrai savoir ma note en electricite	OK
	donne moi ma note de micro onde	OK
	quelle est ma moyenne	OK
	J'aimerai bien avoir ma certificat de scolarité s'elle est prete	OK
	se reste possible d'avoir mon diplôme aujoud'hui	OK

Table IV.16 : Un extrait de l'étape de test pour l'application de consultation scolaire

Cette expérience a considéré l'emploie de 60 requêtes (phrases) introduite par 6 utilisateurs, où chacun d'eux aura à saisir 10 requêtes. Le résultat était que 55 sur 60 requêtes ont été bin compris, donc un taux de compréhension de 91.66%.

b) Application de Diagnostic Médical

Voici un extrait de cette expérience mené :

Utilisateur	**Requête**	**Système**	**Médecin**
Mohamed LICHOURI	j'ai mal aux yeux	Grippe	oui
	je me sens trés fatigué mais je ne peux pas dormir	Allergie	non
	j'ai des nausées et ma téte me fait trés mal	Stress	oui
	j'ai une diarhée et mon ventre me fait mal	Grippe	non
	j'ai trés faim mais je n'ai pas d'appétit	Grippe	oui
	j'ai mal aux doigts et aux pieds	Grippe	oui
	au levet de soleil je ne peux pas bien respirer	Fatigue	non
	je crois que j'ai une grippe car j'ai la gorge toute seche	Grippe	oui
	je ne supporte pas le savon	Grippe	non, allergie
	je ne dors pas beaucoup la nuit	Grippe	oui
Wahib DJAOUTI	j'ai mal au ventre	grippe	non
	j'ai le nez qui coule	grippe	oui
	j'ai le cœur qui bat trop vite	grippe	non, c l'anémie
	j'ai une boule au niveau de l'estomac	grippe	non, c le stress
	je ne peux pas parler	allergie	non
	j'ai mal aux oreilles	anémie	non
	j'ai mal au cou	grippe	non
	j'ai un faible corp	allergie	NON, c la grippe

Table IV.17 : Un extrait de l'étape de test pour l'application de Diagnostic Médical

Dans cette expérience, on a pris 6 personnes, leurs demander d'entrer 10 requêtes chacun, donc un totale de 60 requêtes. Dans la première colonne, on trouve le nom de l'utilisateur,

dans la second sa requête et en troisième, on trouve l'avis de notre système à propos du dépistage de sa maladie. Afin de valider ces résultats, on a pris ces requêtes ainsi que la réponse de notre système à un médecin. Ce dernier, nous a fournis son avis à propos de comportement du système en deux point : oui ou non. En utilisant ces derniers points, on a pu quantifier les performances du système en cours. Donc, sur 60 requêtes, on a eu 25 comme relevant, ce qui implique un taux de compréhension de 25/60= 41.66%.

IV.4 Comparaison des performances des deux systèmes

Afin de conclure ce travail, faisons une comparaison entre les meilleurs résultats des deux systèmes. Un basé sur l'approche d'extraction de concept, alors que l'autre sur l'approche de classification de phrases.

Approche	Extraction de concept		Classification de phrase	
Application	Scolaire	Médical	Scolaire	Médical
Méthode	LSA (K=4)	LSA (K=4)	LSA (K=3)	LSA (K=7)
Type de matrice	Booléenne	TF-IDF	TF-IDF	Booléenne
F1 (%)	54.90	44.10	95	44.44

Table IV.18 : Comparaison entre les performances des deux systèmes en phase d'apprentissage

On remarque bien que les meilleurs résultats ont été atteints par le système basé sur l'approche de classification de phrase. Et cela avec un taux de classification de 95% pour l'application de consultation scolaire, en considérant l'utilisation du modèle LSA sur une matrice pondérée avec le poids TF-IDF et avec k=3 catégorie de phrases. En ce qui concerne l'application de diagnostic, la meilleure performance est celle considérant l'utilisation du modèle LSA sur une matrice booléenne avec k=7 catégories de phrases.

Allant nous examiner maintenant leurs résultats respectifs lors de la phase de test.

Approche	Extraction de concept		Classification de phrase	
Application	Scolaire	Médical	Scolaire	Médical
Taux de Compréhension (%)	73.33		91.66	41.66

Table IV.19 : Comparaison des performances des deux systèmes en phase de test

Comme le but en départ était de concevoir un système général de compréhension de l'énoncé, on trouve qu'en s'aidant de cette comparaison, on peut dire que le deuxième système est le plus apte. Pourquoi ? Parce qu'il nous a fourni les meilleurs résultats en phase d'apprentissage et en test. Est-ce que c'est suffisant comme critère ? En réalité non, car ce qui nous intéresse

le plus est qu'il puisse s'ajuster à la variété des domaines d'applications existants. Rappelons-nous que dans la pratique, on trouve deux classes d'applications :

- Interrogation d'une base de données. Exemple : Consultation Scolaire, État Civile…etc.
- Extraction d'information. Exemple : Diagnostic Médical, e-learning, Conseillé Judiciaire,…etc.

Ceci dit, puisque notre système était capable de manager ces deux classes d'applications, donc il est élu pour être à la base du Système Générale de compréhension d'énoncé.

IV.5. Conclusion

Dans ce chapitre, on a présenté des méthodes pour : (i) l'amélioration du système SyCE-CHM précédents et ce en changeant la valeur des centroïdes initiales (k=2) et en introduisant un nouvel modèle qui est le PLSA (approche alternative de la LSA) ; (ii) la généralisation des systèmes SyCE-CHM afin de répondre aux différentes exigences langagières et applicatives et ce en présentant une nouvelle architecture basé sur l'approche de la classification de texte. Cette dernière a approuvé toutes ces qualités en présentant des taux de reconnaissances et de compréhension largement supérieure au système précédent.

Pour inclure ce chapitre, l'architecture d'un système SyCE-CHM générale doit comporter (selon notre étude) deux phases : (i) une phase d'apprentissage basé sur trois analyseurs : structurelle (segmentation de texte en phrases), lexicale (segmentation des phrases en mots et filtrage des mots insignifiants) et sémantique (représentation des phrases en vecteurs et classification de ces vecteurs en catégories, en utilisant le modèle LSA suivie par le classificateur k-moyenne) ; (ii) une phase de test basé sur un classificateur Bayésienne Naïve pour attribuer à chaque requête en entrée sa classe approprié selon une probabilité). En sortie les données seront soit des informations extrait d'une base de donnée (c'est le cas de l'application de scolarité) ou simplement une phrase descriptif-explicatif (le cas du système de diagnostic médical).

Conclusion Générale
Et
Perspectives

Conclusion Générale

Le travail réalisé au cours de ce mémoire, nous a permis d'examiner les deux aspects de la compréhension de l'énoncé, la première est la mise en place des différents analyseurs et le deuxième est l'application de ce système à deux domaines qui sont : l'interrogation d'une base de données scolaire et le diagnostic médical. Cette tâche consiste à attribuer à chaque symptôme la maladie et soin correspondant. Dans les deux cas, nous avons proposés une méthode de construction de la liste des concepts sémantiques. Ceci a permis d'automatiser la lourde tâche d'étiquetage des grands corpus d'apprentissage.

Nous avons présentés deux systèmes dont les phases :

1) La phase d'apprentissage se manifeste avec trois étape : (i) une étape d'analyse structurelle consistant à segmenter le texte en phrase, (ii) une analyse lexicale servant à segmenter les phrases en mots ainsi leurs filtrages (première réduction de dimension) et (iii) l'analyse sémantique consiste en une présentation vectoriel des termes en utilisant le modèle LSA puis une classification non supervisé pour trouver les concepts, et ce en utilisant l'algorithme des k-moyennes.

2) La phase de test qui contient le module de décodage mot/concept/action sert à traduire chaque requête d'entrée (exprimée en langage naturel) en une requête exprimée dans un langage dit conceptuel puis en une requête SQL compréhensible par le module d'interrogation de la base de données ou en une action.

Ces deux systèmes en questions se distinguent par le fait que le premier est basé sur la méthode d'extraction automatique de concept (de termes) d'où son architecture est celle décrit juste en dessus et un deuxième basé sur la méthode de classification de phrases en catégories ,son architecture différents au niveau de la première phase du fait qu'il ne tient compte que des vecteurs de phrases et non les vecteurs de termes et pour la deuxième phase du fait qu'il utilise maintenant un autre classificateur bayésienne supervisé pour trouver la catégorie approprié à la requête d'entré d'où sa réponse.

Les résultats qu'on a pu atteindre avec le premier système est de 73.33% pour l'application de consultation scolaire seulement. Or, que le second système nous a permis d'aboutir à un taux de compréhension de 91.66% pour l'application de Consultation Scolaire et 41.66% pour l'application de Diagnostic Médical.
Ceci dit, l'architecture du second système basé sur l'approche de classification de phrases en thèmes est plus apte d'être généraliser pour d'autre domaines d'applications.

Perspectives

On envisage d'améliorer la qualité des modèles vectoriels LSA et PLSA et ce en utilisant le modèle LDA « Latent Dirichlet Analysis » ou encore considérer l'utilisation du modèle de Markov caché. En plus, il faudra soigner la méthode d'acquisition du corpus d'apprentissage pour couvrir la totalité des services fournis par une application dans un domaine donnée. L'idéal serait de connaitre à l'avance les constituants (phrases) du corpus d'énoncé.

Du fait qu'on veut généraliser ce système pour qu'il soit compatible avec n'importe quelle application ou domaine, on va opter pour l'ajout de deux autres applications qui sont « état civile » et « conseiller judiciaire ». Omis que ces quatre application ne seront plus séparé mais plutôt intégré dans le même environnement et pour le service multilingue, on envisage d'ajouter un module d'identification de la langue (français, anglais et arabe) en entrée.

Après, ce serait bon d'intégrer les applications discutées dans cette mémoire dans un milieu réel avec bien sûr des entrées et sorties vocale (donc le besoin d'un TTS).

Annexes

Annexe A : LSA entre le Calcul et la Sémantique

1- Considérant l'exemple suivant (extrait du corpus d'apprentissage de l'application de Consultation Scolaire).

1	puis-je avoir mon relevée de note.
2	donne moi mon diplôme.
3	puis-je me renseigner sur ma note.
4	quel note ai-je obtenu en examen ?
5	je souhaite avoir mon certificat de scolarité.
6	je veux mon diplôme.
7	donner moi mon certificat de scolarité.
8	puis-je avoir mon diplôme.
9	donne moi ma note d'examen ?
10	veux tu me donner mon certificat STP ?
11	pouver vous me montrer mon relevée de note.
12	je voudrais savoir ma note ?
13	j'aimerai bien avoir mon certificat de scolarités.

2- Notre Système de Compréhension résout le problème de la sémantique vis-à-vis cette exemple selon deux approches, soit :

- Comprendre la totalité de la phrase en identifiant son thème, sujet ou catégorie.
- Comprendre les mots des phrases et après reconstituer le sens total en combinant ces derniers. Nous considérons ici les concepts de mots.

La première étape consistant à Segmenter les phrases en mots selon le caractère « espace » suivie par un Filtrage des mots insignifiants (appartenant à la liste des Stop-Words ou ayant une langueur inférieur ou égale à 3).

Le résultat fut le suivant :

1	puis-je avoir mon relevée de note.
2	donne moi mon diplôme.
3	puis-je me renseigner sur ma note.
4	quel note ai-je obtenu en examen ?
5	je souhaite avoir mon certificat de scolarité.
6	je veux mon diplôme.

7	donner moi mon certificat de scolarité.
8	puis-je avoir mon diplôme.
9	donne moi ma note d'examen ?
10	veux tu me donner mon certificat STP ?
11	pouvez vous me montrer mon relevée de note.
12	je voudrais savoir ma note ?
13	j'aimerais bien avoir mon certificat de scolarités.

Les mots en rouge sont les mots signifiant pour notre application. (17 mots et 13 phrases)

3- Pour regrouper ces mots en concepts (approche 1) ou phrases en catégories (approche 2), on aura besoin de la matrice d'occurrence terme-phrase afin de récupérer las vecteurs de mots (i.e., phrases). Où chaque élément Cij de cette matrice représente soit une valeur « 1 » (le mot i appartient à la phrase j), « 0 » sinon.

Matrice d'occurrence A :

Terme	Phrase												
	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13
relevée	1	0	0	0	0	0	0	0	0	0	1	0	0
note	1	0	1	1	0	0	0	0	1	0	1	1	0
donne	0	1	0	0	0	0	0	0	0	0	0	0	0
diplôme	0	1	0	0	0	1	0	1	0	0	0	0	0
renseigner	0	0	1	0	0	0	0	0	0	0	0	0	0
obtenu	0	0	0	1	0	0	0	0	0	0	0	0	0
examen	0	0	0	1	0	0	0	0	1	0	0	0	0
souhaite	0	0	0	0	1	0	0	0	0	0	0	0	0
certificat	0	0	0	0	1	0	1	0	0	1	0	0	1
scolarité	0	0	0	0	1	0	1	0	0	0	0	0	1
veux	0	0	0	0	0	1	0	0	0	1	0	0	0
donner	0	0	0	0	0	0	1	0	0	1	0	0	0
pouvez	0	0	0	0	0	0	0	0	0	0	1	0	0
montrer	0	0	0	0	0	0	0	0	0	0	1	0	0
voudrais	0	0	0	0	0	0	0	0	0	0	0	1	0
savoir	0	0	0	0	0	0	0	0	0	0	0	1	0
aimerais	0	0	0	0	0	0	0	0	0	0	0	0	1

Dans cette matrice, les lignes correspondent aux vecteurs de mots alors que les colonnes ceux des phrases.

4- Prenons les mots « donner » et « montrer ». Afin de connaitre s'ils ont la même idée (pour les regrouper dans le même concept) ou pas, considérant la mesure de similarité suivante, qui est l'angle θ entre les deux vecteurs définis par :

$$\theta = cos(V_1, V_2) = \frac{V_1 \times V_2}{\|V_1\| \times \|V_2\|}$$

On va trouver que : cos('donner','montrer') = 0 ⇔ ces deux vecteurs sont orthogonaux donc ils n'ont pas la même idée[1].
Or dans le cas de cette application c'est plutôt l'inverse car ils expriment tous les deux le concept de 'demande'. Donc, on aura besoin de définir un autre espace ou ces deux mots auront plus de similarité.

5- Une transformation est requise afin de passer de l'espace terme-phrase au nouvel espace qui est la SVD. Cette dernière est une méthode algébrique de Décomposition de matrice en Valeur Singulier, ou tout simplement en une multiplication de trois matrices U, S et V^T
En utilisant la SVD, on va passer de la matrice d'origine A vers une nouvelle matrice $A^{'}$.
Dans cette nouvelle matrice on va avoir des mesures de similarité nettement mieux. Utiliser la SVD pour ce but est appelé une Analyse Sémantique Latente LSA.

6- En considérant la matrice A, calculons les trois matrice U, S et V^T :
Etape 1 : calculer le produit matricielle AA^T et A^TA
Etape 2 : calculer les valeurs propres λ des matrices AA^T et A^TA
Etape 3 : calculer les valeurs singulières s en utilisant calculant la racine des valeurs propres λ.
Etape 4 : calculer les vecteurs propres v de A^TA qui sont la base de la matrice V.
Etape 5 : de même calculer les vecteurs propres de AA^T qui sont la base de la matrice U ou bien calculer U en calculons le produit : $U = AVS^{-1}$.
Etape 6 : appliquer une réduction de dimension sur U, S et V^T afin de trouver la nouvel matrice $A^{'}$.

[1] Ceci est due essentiellement au fait qu'ils n'appartiennent pas au même phrase (contexte). (Or ceci n'est pas une nécessité pour dire que deux mots ont le même sens.

Etape 1 :

A=												
1	0	0	0	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	1	1	0
0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	1	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0	0	1	0	0	1
0	0	0	0	1	0	1	0	0	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	1

Table A.1 : Matrice d'occurrence terme-phrase A

AT																
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1

Table A.2 : Matrice transposé A^T

AA^T=																
2	2	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
2	6	0	0	1	1	2	0	0	0	0	0	1	1	1	1	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	3	0	0	0	0	0	0	1	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
0	2	0	0	0	1	2	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	4	3	1	2	0	0	0	0	1
0	0	0	0	0	0	0	1	3	3	0	1	0	0	0	0	1
0	0	0	1	0	0	0	0	1	0	2	1	0	0	0	0	0
0	0	0	0	0	0	0	0	2	1	1	2	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1

Table A.3 : Produit matricielle AA^T

A^TA=												
2	0	1	1	0	0	0	0	1	0	2	1	0
0	2	0	0	0	1	0	1	0	0	0	0	0
1	0	2	1	0	0	0	0	1	0	1	1	0
1	0	1	3	0	0	0	0	2	0	1	1	0
0	0	0	0	3	0	2	0	0	1	0	0	2
0	1	0	0	0	2	0	1	0	1	0	0	0
0	0	0	0	2	0	3	0	0	2	0	0	2
0	1	0	0	0	1	0	1	0	0	0	0	0
1	0	1	2	0	0	0	0	2	0	1	1	0
0	0	0	0	1	1	2	0	0	3	0	0	1
2	0	1	1	0	0	0	0	1	0	4	1	0
1	0	1	1	0	0	0	0	1	0	1	3	0
0	0	0	0	2	0	2	0	0	1	0	0	3

Table A.4 : Produit matricielle A^TA

Etape 2 : Calculer les valeurs propres λ

$A^TA - \lambda I$:

A^T*A-λI=												
2 -λ	0	1	1	0	0	0	0	1	0	2	1	0
0	2-λ	0	0	0	1	0	1	0	0	0	0	0
1	0	2-λ	1	0	0	0	0	1	0	1	1	0
1	0	1	3-λ	0	0	0	0	2	0	1	1	0
0	0	0	0	3-λ	0	2	0	0	1	0	0	2
0	1	0	0	0	2-λ	0	1	0	1	0	0	0
0	0	0	0	2	0	3-λ	0	0	2	0	0	2
0	1	0	0	0	1	0	1-λ	0	0	0	0	0
1	0	1	2	0	0	0	0	2-λ	0	1	1	0
0	0	0	0	1	1	2	0	0	3-λ	0	0	1
2	0	1	1	0	0	0	0	1	0	4-λ	1	0
1	0	1	1	0	0	0	0	1	0	1	3-λ	0
0	0	0	0	2	0	2	0	0	1	0	0	3-λ

Table A.5 : Matrice $A^TA - \lambda I$

Afin d'obtenir les valeurs propres λ, il nous suffit de calculer les racines du déterminant de la matrice en dessus à savoir det($A^TA - \lambda I$). Puisque c'est une matrice (13*13), donc on aura 13 valeurs λ.

Les valeurs propres qu'on a trouvées sont :

$\lambda 1 = 8.47$
$\lambda 2 = 3.062$
$\lambda 3 = 0.399$
$\lambda 4 = 0.638$
$\lambda 5 = 1.253$
$\lambda 6 = 2.179$

$\lambda 7 = 8.103$
$\lambda 8 = 3.911$
$\lambda 9 = 0.114$
$\lambda 10 = 0.459$
$\lambda 11 = 2.413$
$\lambda 12 = 1$
$\lambda 13 = 1$

De même pour la matrice AA^T-λI, en calculant les racines de son déterminant on trouvera 16 valeurs λ car c'est une matrice (17*17) donc au maximum 17 racines:

A*A^T-λI=																
2-λ	2	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
2	6-λ	0	0	1	1	2	0	0	0	0	0	1	1	1	1	0
0	0	1-λ	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	3-λ	0	0	0	0	0	0	1	0	0	0	0	0	0
0	1	0	0	1-λ	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1-λ	1	0	0	0	0	0	0	0	0	0	0
0	2	0	0	0	1	2-λ	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1-λ	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	4-λ	3	1	2	0	0	0	0	1
0	0	0	0	0	0	0	1	3	3-λ	0	1	0	0	0	0	1
0	0	0	1	0	0	0	0	1	0	2-λ	1	0	0	0	0	0
0	0	0	0	0	0	0	0	2	1	1	2-λ	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	1-λ	1	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	1	1-λ	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	1-λ	1	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1-λ	0
0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1-λ

Table A.6 : Matrice AA^T-λI

Les valeurs propres qu'on a trouvées sont :

$\lambda 1 = 8.47$
$\lambda 2 = 3.062$
$\lambda 3 = 2.179$
$\lambda 4 = 1.253$
$\lambda 5 = 0.638$
$\lambda 6 = 0.399$
$\lambda 7 = \lambda 8 = 0$

$\lambda 9 = 8.103$
$\lambda 10 = 3.911$
$\lambda 11 = 2.413$
$\lambda 12 = 0$
$\lambda 13 = 0.459$
$\lambda 14 = 0.114$
$\lambda 15 = \lambda 16 = 1$

Etape 3 : le calcule des valeurs singulières s $si = \sqrt{\lambda i}$

Pour la matrice A^TA :

$s1 = 2.910$
$s2 = 1.749$
$s3 = 0.631$
$s4 = 0.798$
$s5 = 1.119$
$s6 = 1.476$

$s7 = 2.846$
$s8 = 1.977$
$s9 = 0.337$
$s10 = 0.677$
$s11 = 1.553$
$s12 = 1$
$s13 = 1$

S1:												
2,910	0	0	0	0	0	0	0	0	0	0	0	0
0	2,846	0	0	0	0	0	0	0	0	0	0	0
0	0	1,977	0	0	0	0	0	0	0	0	0	0
0	0	0	1,749	0	0	0	0	0	0	0	0	0
0	0	0	0	1,553	0	0	0	0	0	0	0	0
0	0	0	0	0	1,476	0	0	0	0	0	0	0
0	0	0	0	0	0	1,119	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0,798	0	0	0
0	0	0	0	0	0	0	0	0	0	0,677	0	0
0	0	0	0	0	0	0	0	0	0	0	0,631	0
0	0	0	0	0	0	0	0	0	0	0	0	0,337

Table A.7 : Matrice des valeurs singulières S1

Pour la matrice AA^T :

$s1 = 2.910$
$s2 = 1.749$
$s3 = 1.476$
$s4 = 1.119$
$s5 = 0.798$
$s6 = 0.631$
$s7 = 0$
$s8 = 0$

$s9 = 2.846$
$s10 = 1.977$
$s11 = 1.553$
$s12 = 0$
$s13 = 0.677$
$s14 = 0.337$
$s15 = 1$
$s16 = 1$

S2:																
2,910	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	2,846	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1,977	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1,749	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1,553	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1,476	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1,119	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0,798	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0,677	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0,631	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0,337	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table A.8 : Matrice des valeurs singulières S2

On remarque que S=S1=S2 sinon ce serait une contradiction, car la SVD suppose qu'il n'y a qu'une seule décomposition possible pour A.

La diagonale de la matrice S de dimension (13 x 13) contiendra les valeurs singulières déjà calculé. La valeur de ces valeurs singulières mesure l'importance de la **dimension sémantique** correspondante.
Ex ; s1=8.47 ⇔ la première dimension sémantique est la plus importante.
On va utiliser cette propriété pour réduire les dimensions.

Etape 4 : calculer les vecteurs propres v de A^TA qui sont la base de la matrice V.

1- Remplacer les différents valeurs singuliers λ un à un dans la matrice $A^TA-\lambda I$.
2- Retrouver le système d'équation linéaire depuis $(A^TA-\lambda_i I)*v_i$.
3- Normaliser les différents vecteurs propres v_i par sa norme $\|v_i\|$.
4- Concaténer les différents vecteurs propres v_i afin de trouver la matrice V.

V(13 x 13):												
-0,393	0	-0,324	-0,434	0	0	0	0	-0,383	0	-0,515	-0,375	0
0	-0,014	0	0	-0,499	-0,074	-0,559	-0,013	0	-0,429	0	0	-0,499
0	0,551	0	0	-0,146	0,642	-0,054	0,41	0	0,267	0	0	-0,146
-0,246	0	0,098	0,519	0	0	0	0	0,349	0	-0,709	0,19	0
0	0,418	0	0	0,364	-0,072	-0,125	0,245	0	-0,692	0	0	0,364
0,014	0	-0,129	0,42	0	0	0	0	0,227	0	0,168	-0,852	0
-0,117	0	-0,893	0,238	0	0	0	0	0,048	0	0,196	0,302	0
0	-0,141	0	0	-0,612	0,141	-0,141	0	0	0	0	0	0,752
0	0,516	0	0	-0,444	-0,516	0,515	0	0	0	0	0	-0,071
0,866	0	-0,252	-0,115	0	0	0	0	0,065	0	-0,405	-0,067	0
0	0,478	0	0	0,146	-0,174	-0,507	-0,564	-0,001	0,353	0	0	0,145
0,145	0	0,085	0,544	0	0	0	0	-0,821	0	-0,036	0,032	0
0	0,087	0	0	-0,069	0,511	0,355	-0,674	0	-0,376	0	0	-0,069

Table A.9 : Matrice des vecteurs de phrases V

La matrice V est une matrice orthogonale de dimension (13 x 13).
Les lignes de cette matrice représenteront les vecteurs de phrases (colonnes de la matrice V^T), alors que les colonnes sont les dimensions sémantiques (lignes de la matrice V^T). On peut considérer que ces dimensions sont les différents thèmes existants dans le corpus.

Chaque élément v_{ij} indique combien chaque phrase i est reliée à la dimension j.

Etape 5: de même calculer les vecteurs propres v de AA^T qui sont la base de la matrice U.

1- Remplacer les différents valeurs singuliers λ un à un dans la matrice AA^T-λI.
2- Retrouver le système d'équation linéaire depuis $(AA^T-\lambda_i I)*v_i$.
3- Normaliser les différents vecteurs propres v_i par sa norme $||v_i||$.
4- Concaténer les différents vecteurs propres v_i afin de trouver la matrice U.

U (17 x 13):													
	-0,312	0	0	-0,545	0	0,123	-0,071	0	0	-0,577	0	-0,172	0
	-0,833	0	0	0,115	0	-0,103	0,202	0	0	-0,115	0	0,081	0
	0	-0,005	0,278	0	-0,269	0	0	-0,241	0,476	0	0,706	0	-0,257
	0	-0,036	0,811	0	-0,38	0	0	0	0	0	-0,382	0	0,228
	-0,111	0	0	0,056	0	-0,087	0,798	0	0	0,316	0	-0,134	0
	-0,149	0	0	0,297	0	0,285	-0,213	0	0	0,144	0	-0,862	0
	-0,28	0	0	0,496	0	0,438	-0,256	0	0	0,062	0	0,437	0
	0	-0,175	-0,074	0	-0,235	0	0	0,751	0,084	0	0,215	0	0,203
	0	-0,698	-0,04	0	0,057	0	0	0	0	0	0,202	0	0,467
	0	-0,547	-0,175	0	-0,388	0	0	0	0	0	-0,318	0	-0,647
	0	-0,177	0,46	0	0,492	0	0	0,243	-0,476	0	0,264	0	-0,4
	0	-0,347	0,108	0	0,526	0	0	-0,243	0,476	0	-0,227	0	0,06
	-0,177	0	0	-0,405	0	0,114	-0,175	0	0	0,508	0	0,057	0
	-0,177	0	0	-0,405	0	0,114	-0,175	0	0	0,508	0	0,057	0
	-0,129	0	0	0,109	0	-0,578	-0,27	0	0	0,084	0	-0,05	0
	-0,129	0	0	0,109	0	-0,578	-0,27	0	0	0,084	0	-0,05	0
	0	-0,175	-0,074	0	-0,235	0	0	-0,508	-0,56	0	0,214	0	0,203

Table A.10 : Matrice des vecteurs de mots U

La matrice U est une matrice orthogonale de dimension (17 x 13).
Les lignes de cette matrice représenteront les vecteurs de mots, alors que les colonnes sont les dimensions sémantiques. On peut considérer que ces dimensions sont les différents thèmes existants dans le corpus.
Chaque élément u_{ij} indique combien chaque mot i est relié à la dimension j.

Etape 6: appliquer une réduction de dimension sur U, S, V^T afin de trouver la matrice A'

En se basant sur la propriété déjà vue de la matrice S (chaque élément de la diagonale indique l'importance de chaque dimension sémantique), on pourra réduire cette matrice en enlevant les détails non importants en ne gardant que les k plus grands valeurs singuliers et mettant les restes à zéro.
Posant k=9: (gardant seulement les valeurs singulières supérieures ou égales à 1)
La matrice S_k:

S(13 x 9):								
2,910	0	0	0	0	0	0	0	0
0	2,846	0	0	0	0	0	0	0
0	0	1,977	0	0	0	0	0	0
0	0	0	1,749	0	0	0	0	0
0	0	0	0	1,553	0	0	0	0
0	0	0	0	0	1,476	0	0	0
0	0	0	0	0	0	1,119	0	0
0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1

Table A.11 : Matrice des valeurs singulières S_k réduit

La matrice A' : $A' = US_kV^T$

A':	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13
relevée	0,811	0,228	-0,002	-0,292	0,315	-0,32	-0,153	0,037	-0,135	-0,669	-0,154	-0,654	0,238
note	0,846	-0,076	-0,134	0,71	0,046	0,007	0,347	-0,053	0,195	-2,126	-0,121	-0,241	0,037
donne	-0,607	0,255	-0,034	-0,119	-0,249	0,117	-0,375	0,192	0,185	-0,301	0,056	-0,361	0,197
diplôme	-0,386	0,257	0,019	0,341	-0,229	-0,25	-1,482	0,433	0,204	-0,299	-0,124	0,146	0,027
renseigner	0,116	-0,598	-0,063	0,114	-0,277	0,109	0,035	-0,144	0,526	-0,285	-0,342	0,004	0,157
obtenu	0,149	0,053	0,314	0,273	-0,08	0,675	0,01	0,093	-0,34	-0,399	0,088	0,202	0,087
examen	-0,16	0,091	0,443	0,703	-0,045	0,118	0,385	0,131	-0,481	-0,824	0,05	0,363	0,21
souhaite	-0,06	0,145	0,077	-0,089	-0,132	0,062	0,163	0,345	-0,1	-0,017	-0,705	-0,087	-0,529
certificat	-0,045	-0,095	-1,131	-0,105	-0,741	0,033	0,097	0,344	-1,076	-0,036	-0,914	-0,012	-0,19
scolarité	0,223	0,431	-0,738	0,119	-0,95	0,009	0,267	0,425	-0,52	0,174	-0,864	-0,022	-0,079
veux	-0,204	-0,31	-0,27	-0,204	0,078	-0,195	-0,8	-0,498	-0,59	-0,333	-0,286	0,462	-0,251
donner	-0,172	-0,401	-0,766	0,296	-0,168	0,055	-0,198	-0,346	-0,874	0,039	-0,214	-0,367	0,02
pouver	0,496	-0,077	0,227	-0,234	-0,268	-0,335	-0,097	0,051	-0,188	-0,367	0,213	-0,459	-0,136
montrer	0,496	-0,077	0,227	-0,234	-0,268	-0,335	-0,097	0,051	-0,188	-0,367	0,213	-0,459	-0,136
voudrais	0,076	0,203	-0,513	0,185	0,053	0,102	0,08	-0,078	0,285	-0,345	0,325	0,049	-0,568
savoir	0,076	0,203	-0,513	0,185	0,053	0,102	0,08	-0,078	0,285	-0,345	0,325	0,049	-0,568
aimerais	0,187	0,162	-0,439	-0,313	-0,441	-0,084	0,132	0,345	-0,1	-0,058	0,006	0,442	0,32

Table A.12 : Matrice A'

7- Considérant le nouvel espace terme-phrase présenté par la nouvelle matrice A' (qui est l'approximation optimal de la matrice A en réduisant l'espace à k=9 dimension, ce qui vient à ne garder que les k dimensions et mettez les autres à zéros[2] [19]). Dans cette espace calculons de nouveau la distance Euclidien entre le mot « montrer » et « donner », ce qui nous donne :

- Matrice A' : cos ('montrer', 'donner') = 0.00036
- Matrice U : cos ('montrer', 'donner') = 1.117

8- Après avoir effectué la transformation de la matrice A depuis l'espace des occurrences terme-phrase vers l'espace LSA, où chaque élément u_{ij} indique combien chaque mot i est relié à la phrase j (et non pas son occurrence comme dans la matrice A), on va devoir passer vers l'étape suivante qui est la classification des mots en concepts (i.e. catégorisation des phrases en catégories) en utilisant les matrices U et V respectivement.

9- La matrice U contient les vecteurs de mots exprimés en fonction des dimensions sémantiques (thèmes)

[2] Selon le théorème de Eckart-Young,

La matrice U représente les termes en fonctions des dimensions sémantiques

U (17 x 13):	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13
relevée	-0,312	0	0	-0,545	0	0,123	-0,071	0	0	-0,577	0	-0,172	0
note	-0,833	0	0	0,115	0	-0,103	0,202	0	0	-0,115	0	0,081	0
donne	0	-0,005	0,278	0	-0,269	0	0	-0,243	0,476	0	0,706	0	-0,257
diplôme	0	-0,036	0,811	0	-0,38	0	0	0	0	0	-0,382	0	0,228
renseigner	-0,111	0	0	0,056	0	-0,087	0,798	0	0	0,316	0	-0,134	0
obtenu	-0,149	0	0	0,297	0	0,285	-0,213	0	0	0,144	0	-0,862	0
examen	-0,28	0	0	0,496	0	0,438	-0,256	0	0	0,062	0	0,437	0
souhaite	0	-0,175	-0,074	0	-0,235	0	0	0,751	0,084	0	0,215	0	0,203
certificat	0	-0,698	-0,04	0	0,057	0	0	0	0	0	0,202	0	0,467
scolarité	0	-0,547	-0,175	0	-0,388	0	0	0	0	0	-0,318	0	-0,647
veux	0	-0,177	0,46	0	0,492	0	0	0,243	-0,476	0	0,264	0	-0,4
donner	0	-0,347	0,108	0	0,526	0	0	-0,243	0,476	0	-0,227	0	0,06
pouver	-0,177	0	0	-0,405	0	0,114	-0,175	0	0	0,508	0	0,057	0
montrer	-0,177	0	0	-0,405	0	0,114	-0,175	0	0	0,508	0	0,057	0
voudrais	-0,129	0	0	0,109	0	-0,578	-0,27	0	0	0,084	0	-0,05	0
savoir	-0,129	0	0	0,109	0	-0,578	-0,27	0	0	0,084	0	-0,05	0
aimerais	0	-0,175	-0,074	0	-0,235	0	0	-0,508	-0,56	0	0,214	0	0,203

Table A.13 : Matrice des termes-thèmes

10- La matrice V contient les vecteurs de phrases exprimés en fonction des mêmes thèmes traités par le corpus d'apprentissage.

La matrice V représente les phrases en fonctions des dimensions sémantiques

V(13 x 13):	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13
S1	-0,393	0	-0,324	-0,434	0	0	0	0	-0,383	0	-0,515	-0,375	0
S2	0	-0,014	0	0	-0,499	-0,074	-0,559	-0,013	0	-0,429	0	0	-0,499
S3	0	0,551	0	0	-0,146	0,642	-0,054	0,41	0	0,267	0	0	-0,146
S4	-0,246	0	0,098	0,519	0	0	0	0	0,349	0	-0,709	0,19	0
S5	0	0,418	0	0	0,364	-0,072	-0,125	0,245	0	-0,692	0	0	0,364
S6	0,014	0	-0,129	0,42	0	0	0	0	0,227	0	0,168	-0,852	0
S7	-0,117	0	-0,893	0,238	0	0	0	0	0,048	0	0,196	0,302	0
S8	0	-0,141	0	0	-0,612	0,141	-0,141	0	0	0	0	0	0,752
S9	0	0,516	0	0	-0,444	-0,516	0,515	0	0	0	0	0	-0,071
S10	0,866	0	-0,252	-0,115	0	0	0	0	0,065	0	-0,405	-0,067	0
S11	0	0,478	0	0	0,146	-0,174	-0,507	-0,564	-0,001	0,353	0	0	0,145
S12	0,145	0	0,085	0,544	0	0	0	0	-0,821	0	-0,036	0,032	0
S13	0	0,087	0	0	-0,069	0,511	0,355	-0,674	0	-0,376	0	0	-0,069

Table A.14 : Matrice des phrases-thèmes

11- Donc la LSA :

- Prendra une matrice d'occurrence terme-phrase.
- Traitera la matrice bruitée en la projetant dans un nouvel espace réduit.
- Extraire le sens caché (latente) depuis les phrases et les mots (connus) qui est le thème.
- Générera une matrice U qui représente les termes en fonction des thèmes.
- Générera une matrice S qui contient la force de chaque thème.
- Générera une matrice V qui représente les phrases en fonction des thèmes.

Annexe B : Classificateur k-moyennes

En ce qui suit, on va présenter les étapes du classificateur k-moyennes à partir de la matrice V des phrases.

Le classificateur k-moyennes est un classificateur non supervisé. Mais tous de même pour qu'il fonctionne correctement on est obligé de spécifier le nombre de classe k en avance (mais ceci ne le rend pas **supervisé**, car on ne sait toujours pas les constituants des classes).

Ce choix est spécifier en soit :

- On sait dés le départ le nombre de classes, du fait qu'on maitrise le domaine d'application.
- On n'a aucune idée sur les échantillons à classer, donc on doit préciser la valeur k pratiquement en prenant un intervalle de valeurs de 1 à 10 et garder celui donnant la meilleure classification.

La classification se fera en trois étapes :

1- Déterminer les k centroïdes initiale.
2- Calculer la distance entre chaque échantillon et les k centroïdes.
3- Grouper les échantillons en se basant sur le critère de la distance minimale.

Rappelons la matrice V qu'on va considérer.

La matrice V représente les phrases en fonctions des dimensions sémantiques

V(13 x 13):	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13
S1	-0,393	0	-0,324	-0,434	0	0	0	0	-0,383	0	-0,515	-0,375	0
S2	0	-0,014	0	0	-0,499	-0,074	-0,559	-0,013	0	-0,429	0	0	-0,499
S3	0	0,551	0	0	-0,146	0,642	-0,054	0,41	0	0,267	0	0	-0,146
S4	-0,246	0	0,098	0,519	0	0	0	0	0,349	0	-0,709	0,19	0
S5	0	0,418	0	0	0,364	-0,072	-0,125	0,245	0	-0,692	0	0	0,364
S6	0,014	0	-0,129	0,42	0	0	0	0	0,227	0	0,168	-0,852	0
S7	-0,117	0	-0,893	0,238	0	0	0	0	0,048	0	0,196	0,302	0
S8	0	-0,141	0	0	-0,612	0,141	-0,141	0	0	0	0	0	0,752
S9	0	0,516	0	0	-0,444	-0,516	0,515	0	0	0	0	0	-0,071
S10	0,866	0	-0,252	-0,115	0	0	0	0	0,065	0	-0,405	-0,067	0
S11	0	0,478	0	0	0,146	-0,174	-0,507	-0,564	-0,001	0,353	0	0	0,145
S12	0,145	0	0,085	0,544	0	0	0	0	-0,821	0	-0,036	0,032	0
S13	0	0,087	0	0	-0,069	0,511	0,355	-0,674	0	-0,376	0	0	-0,069

Table B.1 : Matrice des vecteurs de phrases V

Itération1 :

Phase 1 : étape 1

Puisque on traite dans ce mémoire une application de Consultation Scolaire et qu'on s'est limité aux trois services qui sont Note, Certificat de Scolarité et Diplôme ; donc notre choix de la valeur k sera k= 3.

Choisissant maintenant trois phrases[1] pour qu'ils soient nos centroides initiales. Par exemple prenons les phrases S1, S2 et S5 de coordonnées :

S1 (-0.393, 0, -0.324, -0.434, 0, 0, 0, 0, -0.383, 0, -0.515, -0.375, 0)

S2 (0, -0.014, 0, 0, -0.499, -0.074, -0.559, -0.013, 0, -0.429, 0, 0, -0.499)

S5 (0, 0.418, 0, 0, 0.364, -0.072, -0.125, 0.245, 0, -0.692, 0, 0, 0.364)

Phase 2: étape 2

Calculer la distance entre chaque phrase et ces trois phrases.

La distance qu'on considère pour ce classificateur est la distance Euclidienne.

1 ére itération: matrice de distance					1 ére itération: Matrice d'appartenance			
	S1	S2	S5			S1	S2	S5
S1	0	1,41445573	1,41411951		S1	1	0	0
S2	1,41445573	0	1,41430372		S2	0	1	0
S3	1,41457485	1,41402581	1,41408345		S3	0	1	0
S4	1,41442108	1,41436488	1,41402864		S4	0	0	1
S5	1,41411951	1,41430372	0		S5	0	0	1
S6	1,41389816	1,41395156	1,41361522		S6	0	0	1
S7	1,41461373	1,41423866	1,4139024		S7	0	0	1
S8	1,41421745	1,41388048	1,41425422		S8	0	1	0
S9	1,41429629	1,41426978	1,41326501		S9	0	0	1
S10	1,41410537	1,41413896	1,41380267		S10	0	0	1
S11	1,41435851	1,41444512	1,41489222		S11	1	0	0
S12	1,41426412	1,41453597	1,41419977		S12	0	0	1
S13	1,41428745	1,41327704	1,4139466		S13	0	1	0

Table B.2 : 1ére itération Matrice des distances et d'appartenance

Phase 3 : étape 3

Agrégation (regroupement) des phrases en catégories en considérant la phrase avec la distance minimale.

G1: S1, S11

G2: S2, S3, S8, S13

G3: S4, S5, S6, S7, S9, S10, S12

Itération 2 :

Phase 4: étape1

Redéfinir les nouveaux centroïdes, en calculant la moyenne de chaque groupe

C1= (S1+S11)/2 = (-0.1965, 0.239, -0.162, -0.217, 0.073, -0.087, -0.2535, -0.282, -0.192, 0.1765, -0.2575, -0.1875, 0.0725)

C2= (S2+S3+S8+S13)/2 = (0, 0.12075, 0, 0, -0.3315, 0.305, -0.09975, -0.06925, 0, -0.1345, 0, 0, 0.0095)

C3= (S4+S5+S6+S7+S9+S10+S12)/2 = (0.094571429, 0.133428571, -0.155857143, 0.229428571, -0.011428571, -0.084, 0.055714286, 0.035, -0.018857143, -0.098857143,

[1] Un choix justidieux serait de considérer une phrase traitant la Note, une autre pour la Certificat et enfin une pour le Diplôme.

-0.112285714, -0.056428571, 0.041857143)

Phase 5: étape 2

Calculons de nouveaux la distance Euclidien entre les phrases et les nouveaux centroïdes C1, C2 et C3

2 éme itération: matrice de distance					2 éme itération: Matrice d'appartenance			
	C1	C2	C3			C1	C2	C3
S1	0,70717926	1,11836664	1,14324559		S1	1	0	0
S2	1,22497653	0,86558185	1,14328002		S2	0	1	0
S3	1,22540381	0,86616998	1,14337588		S3	0	1	0
S4	1,22515285	1,11825173	0,85724145		S4	0	0	1
S5	1,22504061	1,11806705	0,85644416		S5	0	0	1
S6	1,22456257	1,11772892	0,85631388		S6	0	0	1
S7	1,22506837	1,1180921	0,85704016		S7	0	0	1
S8	1,22512591	0,86601555	1,14263973		S8	0	1	0
S9	1,22482305	1,11812497	0,85680131		S9	0	0	1
S10	1,22472425	1,11796598	0,85664559		S10	0	0	1
S11	0,70717926	1,11880402	1,14385873		S11	1	0	0
S12	1,22468343	1,11846812	0,85742631		S12	0	0	1
S13	1,22489122	0,86572856	1,14286602		S13	0	1	0

Table B.3 : 2^{éme} itération Matrice des distances et d'appartenance

Phase 6 : étape 3

Les nouveaux groupes obtenus après agrégation de nouvelles phrases selon le critère de la distance minimal sont :

G1: S1, S11

G2: S2, S3, S8, S13

G3: S4, S5, S6, S7, S9, S10, S12

On remarque que les groupes formés sont les mêmes, donc fin de classification.

Taux de reconnaissance :

Afin de valider les résultats trouvés, une mesure de taux de reconnaissance est nécessaire. On va opter pour la mesure F1 (voir §2.1.3 du chapitre 3).

Le résultat obtenus est égale à : F1= 2*(7/13)*(6/13)/(13/13) avec :

Précision = 7/13 = 53.84%

Rappel = 6/13 = 46.15%

F1= 49.70%

Enfin, on peut conclure que l'algorithme k-moyennes peut avoir tant d'avantages qu'inconvénients, et on peut voir ça à travers les résultats déjà trouvés.

- Le classificateur k-moyennes est un classificateur non-supervisé.
- La rapidité de calcul.
- Pas de chevauchement entre les différentes classes.
- On aura toujours k classes en finale.
- Impossibilité de comparer les résultats pour déférents valeurs de k.
- Le choix du nombre de centroïdes initiales k n'est pas aussi évident, c'est pourquoi l'orientation vers l'utilisation de plusieurs valeurs et en garder celui donnant le meilleur taux de reconnaissance. [20]

Annexe C : Classificateur Bayésien Naïf Multinomiale

On va présenter dans ce qui suit les étapes du classificateur Bayésien Naïf Multinomiale à travers l'application de scolarité.

Rappelons que ce classificateur n'est considéré qu'avec la deuxième approche qui est celle basé sur la classification de phrases.

Cette technique [Wang, Hodges, & Tang, 2003] est classique pour la catégorisation de textes [Plantié, 2006]. Elle combine l'utilisation de la loi de Bayes bien connue en probabilités et la loi multinomiale. Nous avons simplement précisé le calcul de la loi à priori en utilisant l'estimateur de Laplace pour éviter les biais dus à l'absence de certains mots dans un texte.

[Plantié, M., Roche, M., & Dray, 2008]

Le principe du classificateur Bayésien Naïf Multinomiale est le suivant :

1. Considérant le corpus d'apprentissage T contenant un ensemble de phrases S, où chacune d'eux a sa propre classe ou catégorie. En totale on a K classes, C_1, C_2,…, C_k. Chaque phrase S est représentée par un vecteur à N dimension, S = {W_1, W_2,…, W_N}.Où N représente le nombre de terme significatif W dans le corpus d'apprentissage et les W_i sont les termes.
2. Sachant une phrase S, le classificateur va prédire que cette phrase appartiendra à la classe ayant la plus grande probabilité à postériori, conditionné par S. Ceci dit S est prévoyait d'apparaitre à la classe C_i si et seulement si

   $$P(C_i|S) > P(C_j|S) \quad \text{pour } 1 \leq j \leq N,\ j \neq i$$

   Donc on devra trouver la classe qui maximise $P(C_i|S)$. La classe C_i où cette probabilité est maximale est appelé le maximum à postériori. Á l'aide de la règle de Bayes

   $$P(C_i|S) = \frac{P(S|C_i)P(C_i)}{P(S)}$$

3. Puisque P(S) est la même pour tous les classes (car elle ne dépend que de la phrase S), donc ceci revient à maximiser $P(S/C_i)P(C_i)$. Notons que la probabilité à priori des classes $P(C_i)$ pourra être estimé par $P(C_i) = \text{freq}(C_i, T)/|T|$ le nombre de phrases appartenant à la classe C_i sur le nombre totale de phrases.
4. Á cause de la nature du corpus T, qui est un corpus de phrases S constitué chacune d'une suite de mots W_i, le calcule de la probabilité $P(S/C_i)$ exige une grande capacité de calcule. Afin de la réduire, une hypothèse naïve d'indépendance conditionnelle de classe est considérée, ceci dit que chaque mot est indépendant à d'autre dans la même classe. Ce qui revient à que $P(S|C_i) \approx \prod_{k=1}^{n} P(W_k|C_i)$

   Où n représente le nombre de mots dans la phrase S (la taille de la phrase S).

La probabilité $P(W_k/C_i)$ sera estimé par $P(W_k|C_i) = freq(W_K, C_i)/freq(C_i, T)$

5. Afin de prédire la classe de la phrase S, $P(S/C_i)P(C_i)$ est évalué pour chaque classe C_i. Le classificateur prédit que la classe de S est C_i si et seulement si elle est la classe qui maximise $P(S/C_i)P(C_i)$.
6. L'estimateur de Laplace sera considérer afin d'éviter le biais des mots rares, ce qui induit que la probabilité $P(W_k/C_i)=0$ si $freq(W_k,C_i)=0$ sera ré-estimé par l'estimateur M-estimate (Estimateur de Laplace)

$$P(W_k|C_i) = \frac{n_k + 1}{n + m}$$

Où :

n= freq(Ci,T)

n_k= $freq(W_k,C_i)$

m= |T| la taille du corpus T

Considérant maintenant notre corpus d'apprentissage pour l'application de Consultation Scolaire contenant un ensemble de phrase étiqueté chacune avec l'une des trois classes Note, Diplôme ou Certificat.

N°	La phrase	Catégorie
1	puis-je avoir mon relevee de note .	Note
2	donne moi mon diplome .	Diplôme
3	puis-je me renseigner sur ma note .	Note
4	quel note ai-je obtenu en examen ?	Note
5	je souhaite avoir ma certificat de scolarite .	Certificat
6	je veux mon diplome .	Diplôme
7	donner moi ma certificat de scolarite .	Certificat
8	puis-je avoir mon diplome .	Diplôme
9	donne moi ma note d'examen ?	Note
10	veux tu me donner ma certificat STP ?	Certificat
11	pouver vous me montrer mon relevee de note .	Note
12	je voudrais savoir ma note ?	Note
13	j'aimerai bien avoir ma certificat de scolarite .	Certificat

Table C.1 : Corpus d'apprentissage étiqueté avec la Catégorie correspondante

Voici la liste des classes considérés, qui est au nombre de trois.

Les Classes	Sens
Note	Demande de la Note
Diplôme	Demande du Diplôme
Certificat	Demande de Certificat

Table C.2 : La liste des classes

Calculons pour chacune des classes C_i les fréquences n, n_k et m ainsi que les probabilités $P(C_i)$, $P(W_k/C_i)$.

Étape 1 : calcule des fréquences n, n_k et m

Terme	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13
relevée	2	0	0	0	0	0	0	0	0	0	2	0	0
note	6	0	6	6	0	0	0	0	6	0	6	6	0
donne	0	1	0	0	0	0	1	0	1	0	0	0	0
diplôme	0	3	0	0	0	3	0	3	0	0	0	0	0
renseigner	0	0	1	0	0	0	0	0	0	0	0	0	0
obtenu	0	0	0	1	0	0	0	0	0	0	0	0	0
examen	0	0	0	2	0	0	0	0	2	0	0	0	0
souhaite	0	0	0	0	1	0	0	0	0	0	0	0	0
certificat	0	0	0	0	4	0	4	0	0	4	0	0	4
scolarité	0	0	0	0	3	0	3	0	0	0	0	0	3
veux	0	0	0	0	0	1	0	0	0	1	0	0	0
donner	0	0	0	0	0	0	0	0	0	1	0	0	0
pouvez	0	0	0	0	0	0	0	0	0	0	1	0	0
montrer	0	0	0	0	0	0	0	0	0	0	1	0	0
voudrais	0	0	0	0	0	0	0	0	0	0	0	1	0
savoir	0	0	0	0	0	0	0	0	0	0	0	1	0
aimerais	0	0	0	0	0	0	0	0	0	0	0	0	1
Classe	C1	C2	C1	C1	C3	C2	C3	C2	C1	C3	C1	C1	C3

Table C.3 : Les fréquences d'apparitions n_k (terme, phrase, classe)

C1 : Note

C2 : Diplôme

C3 : Certificat

n : La fréquence de chaque classe

Classe	n
Note	6
Diplôme	3
Certificat	4

Table C.4 : La fréquence n des classes

m : La taille du corpus

m=|T|=13 phrases

n_k : La fréquence d'apparition de chaque terme i dans la classe j

Terme\Classe	C1	C2	C3
relevée	2	0	0
note	6	0	0
donne	1	1	1
diplôme	0	3	0
renseigner	1	0	0
obtenu	1	0	0
examen	2	0	0
souhaite	0	0	1
certificat	0	0	4
scolarité	0	0	3
veux	0	1	1
donner	0	0	1
pouvez	1	0	0
montrer	1	0	0
voudrais	1	0	0
savoir	1	0	0
aimerais	0	0	1

Table C.5 : Les fréquences d'apparition n_k (terme, classe)

Étape 2 : Construction du modèle Naïf de Bayes

Ceci consiste à calculer les probabilités à priori des classes C_i ; $P(C_i)$ ainsi que les probabilités conditionnelles des termes W_i sachant les classes C_j ; $P(W_i/C_j)$.

Où :

m = |T| : la taille du corpus ; m=13 phrases

$P(C_i)$ = la probabilité à priori de chaque classe C_i

= nombre de phrases dans une classe / nombre totale des phrases (m)

P('Note') = 6 phrases / 13 phrases = 0,4615

P('Diplôme') = 3 / 13 = 0,2307

P('Certificat') = 4 / 13 = 0,3076

$P(W_i/C_j)$ = la probabilité conditionnelle du mot W_i sachant la classe C_j estimé par l'estimateur de Laplace

$$= \frac{la\ fréquence\ du\ terme\ W_i\ dans\ la\ classe\ C_j + 1}{la\ fréquence\ de\ la\ classe\ C_j dans\ le\ corpus\ T + m}$$

Exemple :

P('relevée'/'Note') = (2+1)/(6+13) = 3/19 = 0.1578

P('relevée'/'Diplôme') = (0+1)/(3+13) = 1/16 = 0.0625

P('relevée'/'Certificat') = (0+1)/(4+13) = 1/17 = 0.0588

m=|T|	13		
C	Note	Diplôme	Certificat
P(C_i)	0,461538462	0,230769231	0,307692308
P("relevée"/C_i)	0,157894737	0,0625	0,058823529
P("note"/C_i)	0,368421053	0,0625	0,058823529
P("donne"/C_i)	0,105263158	0,125	0,117647059
P("diplôme"/C_i)	0,052631579	0,25	0,058823529
P("renseigner"/C_i)	0,105263158	0,0625	0,058823529
P("obtenu"/C_i)	0,105263158	0,0625	0,058823529
P("examen"/C_i)	0,157894737	0,0625	0,058823529
P("souhaite"/C_i)	0,052631579	0,0625	0,117647059
P("certificat"/C_i)	0,052631579	0,0625	0,294117647
P("scolarité"/C_i)	0,052631579	0,0625	0,235294118
P("veux"/C_i)	0,052631579	0,125	0,117647059
P("donner"/C_i)	0,052631579	0,0625	0,117647059
P("pouvez"/C_i)	0,105263158	0,0625	0,058823529
P("montrer"/C_i)	0,105263158	0,0625	0,058823529
P("voudrais"/C_i)	0,105263158	0,0625	0,058823529
P("savoir"/C_i)	0,105263158	0,0625	0,058823529
P("aimerais"/C_i)	0,052631579	0,0625	0,117647059

Table C.6 : Le modèle Naïf de Bayes

Etape 3 : Classification d'une phrase de test

Afin de classer une phrase S, on devra calculer la probabilité à postériori P(C_j/S) pour chaque classe C_j .
Où :

$$P(C_j|S) = P(C_j) \times \prod_{k=1}^{v} (W_k|C_j)$$

Considérant la phrase suivante : S = **' combien j'ai eu en cet examen'**

1- Récupérer les termes après segmentation et filtrage, 'combien', 'examen'
2- Calculons les probabilités à postériori

P('Note'/S) = P('Note') x P('combien'/'Note') x P('examen'/'Note')
= 0.46153 x ((0+1)/(6+13)) x 0.15789
= 0.46153 x 0.0526 x 0.15789
= 0,00383301311142

P(‘Diplôme’/S) = P(‘Diplôme’) x P(‘combien’/’Diplôme’) x P(‘examen’/’Diplôme’)
= 0.23076 x ((0+1)/(3+13)) x 0.0625
= 0.23076 x 0.0625 x 0.0625
= 0,00090140625

P(‘Certificat’/S) = P(‘Certificat’) x P(‘combien’/’Certificat’) x P(‘examen’/’Certificat’)
= 0.30769 x ((0+1)/(4+13)) x 0.05882
= 0.30769 x 0.05882 x 0.05882
= 0,001064543523556

Puisque P*(‘Note’/S)* a la probabilité la plus grande, donc la phrase S est classé avec la classe ‘Note’. Ceci est logique car la phrase contient le mot ‘examen’ qui est reliée à la Note.

Problème de soupassement :

Ceci est due en générale à la taille du corpus, plus le corpus est grand plus les probabilités tendent vers zéros ainsi générer un problème de soupassement (Underflow) qui en résulte une difficulté pour le CPU. C’est pourquoi la considération d’un opérateur afin d’éviter ce problème, qui est le logarithme :

P(‘Note’/S) = log (0.46153 x 0.0526 x 0.15789)
= log (0.46153) + log (0.0526) + log (0.15789)
= -0,3358 -1,2790 -0,8016
= -2.4164
P(‘Diplôme’/S) = -3,0450
P(‘Certificat’/S) = -2,9728

Encore une fois la probabilité P(‘Note’/S) a la plus grande valeur, donc la classe n’a pas changé est les valeurs des probabilités ne cause plus de problème de soupassement.

Remarque :
La propriété la plus intéressante du classificateur Naïf de Bayes est qu’il ne recours pas à un grand corpus d’apprentissage.
Tous de même, si la taille du corpus d’apprentissage est trop grande la rapidité de calcule ainsi que les performances du classificateur ne sont pas affectés.

Annexe D : PLSA la dérivée de LSA

La PLSA, une méthode de traitement automatique des langues inspirée de l'analyse sémantique latente LSA, supposera toujours que la relation entre les termes et les phrases pourra être exprimé selon une relation phrase-thème et thème-terme en utilisant en plus un modèle probabiliste tel que la PLSA:

- Prendra une matrice d'occurrences terme-phrases A.
- Modélise les probabilités conjoints P(W,S) comme un mélange de distributions multinomiales indépendantes conditionnellement :
  $P(W,S) = \sum P(S \mid Z)P(Z)P(W \mid Z) = P(S) \sum P(Z \mid S)P(W \mid Z)$
- Peut exprimer ce mélange de distributions comme la factorisation d'une matrice, comme le montre cette figure.

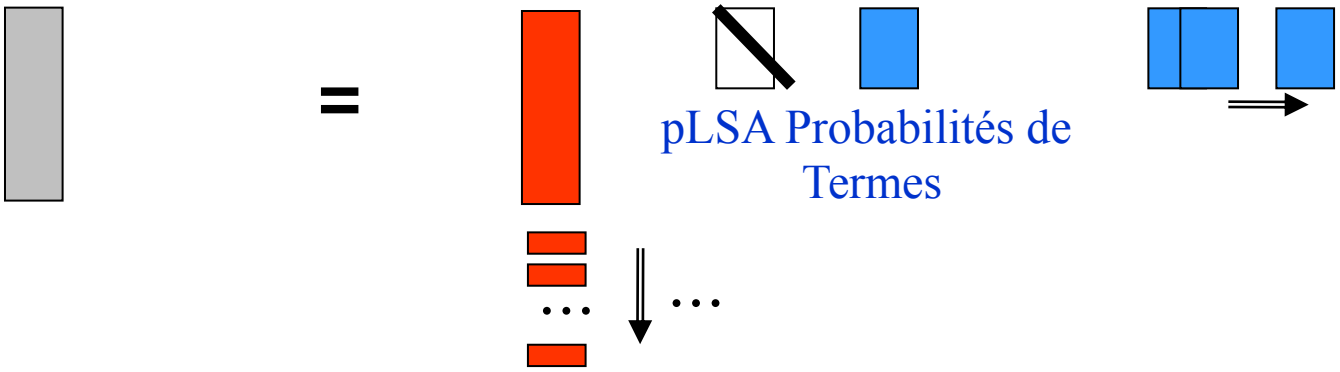


Figure D.1 : Factorisation de la matrice des d'occurrences A en trois matrices de probabilités

- Est basé sur l'algorithme EM (Expectation Maximum) afin d'estimer la probabilité maximum de vraisemblance.
- Alterne deux étapes :
  - Etape E : une étape d'espérance où les probabilités à postériori de la classe variable Z (thème) est calculé.

$$P(z|s,w) = \frac{P(z)P(s|z)P(w|z)}{\sum_{z' \in Z} P(z')P(s|z')P(w|z')}$$

  - Etape M : une étape de maximisation où les paramètres P(Z), P(S|Z) et P(W|Z) sont réestimés.

$$P(w|z) \propto \sum_{s \in S} n(s,w)P(z|s,w)$$

$$P(s|z) \propto \sum_{w \in W} n(s,w)P(z|S,w)$$

$$P(z) \propto \sum_{s \in S} \sum_{w \in W} n(s,w)P(z|S,w)$$

Considérant la matrice d'occurrence A suivante :

Terme	Phrase												
	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13
relevée	1	0	0	0	0	0	0	0	0	0	1	0	0
note	1	0	1	1	0	0	0	0	1	0	1	1	0
donne	0	1	0	0	0	0	0	0	0	0	0	0	0
diplôme	0	1	0	0	0	1	0	1	0	0	0	0	0
renseigner	0	0	1	0	0	0	0	0	0	0	0	0	0
obtenu	0	0	0	1	0	0	0	0	0	0	0	0	0
examen	0	0	0	1	0	0	0	0	1	0	0	0	0
souhaite	0	0	0	0	1	0	0	0	0	0	0	0	0
certificat	0	0	0	0	1	0	1	0	0	1	0	0	1
scolarité	0	0	0	0	1	0	1	0	0	0	0	0	1
veux	0	0	0	0	0	1	0	0	0	1	0	0	0
donner	0	0	0	0	0	0	1	0	0	1	0	0	0
pouvez	0	0	0	0	0	0	0	0	0	0	1	0	0
montrer	0	0	0	0	0	0	0	0	0	0	1	0	0
voudrais	0	0	0	0	0	0	0	0	0	0	0	1	0
savoir	0	0	0	0	0	0	0	0	0	0	0	1	0
aimerais	0	0	0	0	0	0	0	0	0	0	0	0	1

Table D.1 : Matrice d'occurrences terme-Phrases A

Et considérant qu'on a trois thèmes ; nz=3 (ceci n'est considérer que pour l'approche basé sur la catégorisation des phrases).

Phase 1 : initialiser les paramètres du modèle EM
P(Z) = 1/3 car : nz=3
P(S/Z) = 1/13 car : ns=13
P(W/Z) = 1/17 car : nw= 17
Ou bien générer des valeurs aléatoires
Donc :
Etape E :
P(Z/S,W) = 1/3 * 1/13 * 1/17 = 1/663
Etape M : réestimation des paramètres jusqu'à convergence

En fin d'itération on a obtenus trois matrices : la matrice S des Thèmes, la matrice U des Termes-Thèmes et la matrice V des Phrases-Thèmes.

La matrice S : elle contient les probabilités P(z)

	Théme 1	Théme 2	Théme 3
pz	0.60606060606060608	0.060606060606060608	0.33333333333333331

Table D.2 : Matrice des probabilités P(z)

La matrice U : elle contient les probabilités P(W/Z)

pwz	Théme 1	Théme 2	Théme 3
relevée	1.0	1.0	0.0
note	2.0	0.0	4.0
donne	1.0	0.0	0.0
diplôme	2.0	0.0	1.0
renseigner	1.0	0.0	0.0
obtenu	0.0	0.0	1.0
examen	1.0	0.0	1.0
souhaite	1.0	0.0	0.0
certificat	4.0	0.0	0.0
scolarité	3.0	0.0	0.0
veux	0.0	0.0	2.0
donner	2.0	0.0	0.0
pouvez	1.0	0.0	0.0
montrer	1.0	0.0	0.0
voudrais	0.0	0.0	1.0
savoir	0.0	0.0	1.0
aimerais	0.0	1.0	0.0

Table D.3 : Matrice des probabilités P(w/z)

La matrice V : elle contient les probabilités P(S/Z)

psz	Théme 1	Théme 2	Théme 3
puis-je avoir mon relevee de note .	0.0	1.0	1.0
donne moi mon diplome .	2.0	0.0	0.0
puis-je me renseigner sur ma note .	0.5	0.0	0.5
quel note ai-je obtenu en examen ?	0.0	0.0	3.0
je souhaite avoir ma certificat de scolarite .	3.0	0.0	0.0
je veux mon diplome .	0.0	0.0	2.0
donner moi ma certificat de scolarite .	3.0	0.0	0.0
puis-je avoir mon diplome .	1.0	0.0	0.0
donne moi ma note d'examen ?	2.0	0.0	0.0
veux tu me donner ma certificat STP ?	2.0	0.0	1.0
pouver vous me montrer mon relevee de note .	4.0	0.0	0.0
je voudrais savoir ma note ?	0.0	0.0	3.0
j'aimerai bien avoir ma certificat de scolarite .	2.0	1.0	0.0

Table D.4: Matrice des probabilités P(s/z)

En résumé la PLSA prendra en entrée une matrice d'occurrences terme-phrase et générera en sortie trois matrices :

- Une matrice S contenant les probabilités P(z)
- Une matrice U contenant les probabilités d'appartenance de chaque Terme W au Thème Z ; P(W/Z)
- Une matrice V contenant les probabilités d'appartenance de chaque Phrase S au Thème Z ; P(S/Z)

Annexe E : Différents Algorithme utilisé par le SyCE basé sur la méthode d'Extraction de Concepts

II.2.1.4. Usage de Punkt Tokenizer

Dans cette partie, on va présenter un exemple d'un tokeniseur qui est le tokeniseur de NLTK. [1][4]

Le PunktSentenceTokenizer divise un texte en une liste de phrases, en utilisant un algorithme non supervisé pour construire un modèle pour les mots abrégés, collocations, et les mots qui commencent les phrases. Il doit être entrainé sur une grande collection de texte dans la langue désirée avant qu'il puisse être utilisé. L'algorithme de cette tokeniseur est décrit dans Kiss & Strunk (2006). [58]

Exemple d'utilisation pour notre étude :

```
>>> import nltk.data
>>> text = """
... Punkt sait que les périodes de M. Smith et Johann S. Bach
... ne marque pas les limites de la phrase. Et parfois, des phrases
... peut commencer par des mots non capitalisés. i est une bonne variable
... nom.
... "" "
>>> tokenizer = nltk.data.load('tokenizers/punkt/english.pickle')
>>> print '\n-----\n'.join(tokenizer.tokenize(text.strip()))
Punkt sait que les périodes de M. Smith et Johann S. Bach
ne marque pas les limites de la phrase.
----------
Et parfois, des phrases
peut commencer par des mots non capitalisés.
----------
i est une bonne variable
nom.
```

Et encore, on peut utiliser une autre fonction dans NLTK qui est « Split » ; comme ceci :

[1] Natural Language Toolkit (NLTK) est une bibliothèque logicielle en Python permettant un traitement automatique des langues. En plus de la bibliothèque, NLTK fournit des démonstrations graphiques, des données-échantillon, des tutoriels, ainsi que la documentation de l'interface de programmation (API),

```
>>> import nltk
>>> text=open('corpus2.txt').read() #lire le contenu du corpus
>>> text # Visualiser le corpus
"puis-je avoir mon relevée de note .\ndonne moi mon diplôme .\npuis-je me renseigner sur ma note  .\nquel note ai-je obtenu en examen  ?\nje souhaite avoir ma certificat de scolarité .\nje veux mon diplôme .\ndonner moi ma certificat de scolarité  .\npuis-je avoir mon diplôme .\ndonne moi ma note d'examen  ?\nveux tu me donner ma certificat STP  ?\npouver vous me montrer mon relevée de note .\nje voudrais savoir ma note  ?\nj'aimerai bien avoir ma certificat de scolarité.»
>>> for d in text.split('\n'): # Segmentation en phrase
        d  #  Visualiser les Phrases
'puis-je avoir mon relevée de note.'
'donne moi mon diplôme.'
'puis-je me renseigner sur ma note.'
'quel note ai-je obtenu en examen?'
'je souhaite avoir ma certificat de scolarité.'
'je veux mon diplôme.'
'donner moi ma certificat de scolarité.'
'puis-je avoir mon diplôme.'
"donne moi ma note d'examen?"
'veux tu me donner ma certificat STP?'
'pouver vous me montrer mon relevée de note.'
'je voudrais savoir ma note?'
"j'aimerai bien avoir ma certificat de scolarité."
```

D.1 Phase d'apprentissage

D.1.1 Modèle de représentation vectoriel LSA en Python

```
from __future__ import division # import division module
import sys, string, math, nltk
from nltk import bigrams
from nltk import NgramModel
from nltk.probability import FreqDist
from scipy import array
from numpy import zeros
from scipy.linalg import svd
#following needed for TFIDF
from math import log
from numpy import asarray, sum

#////////////PrEtRaItEmEnT/////////////////////
```

```
# importer le corpus dans la variable text
text=open('corpus2Test.txt').read()
# déviser le text en phrases
docs=text.split('\n')
# importer le fichier de filtrage "stoplist"
stopwords = open('french.stoplist.txt').read()
#////////////////////////////////////////////////

#***************Tokenizer*************************
token = nltk.WordTokenizer().tokenize(text) # create tokens file
Token=[w for w in token
       if (w not in stopwords and len(w)>3)]
Token_list=open('tokenList.txt','w')
for i,w  in enumerate(set(Token)):
  Token_list.write('%s-%s\n' % (i+1,w))
Token_list.close()
#************************************************

#//////////////////TF-IDF Transformation/////////////////////////////
def freq(word, document):
  #return document.replace("'"," ").split().count(word)
  return document.split().count(word)

def wordCount(document):
  return len(document.split(None))

def numDocsContaining(word,documentList):
  count = 0
  for document in documentList:
    x=document.replace("'"," ")
    if freq(word,x) > 0:
      count += 1
  return count

def tf(word, document):
  return (freq(word,document) / float(wordCount(document)))

def idf(word, documentList):
  return math.log(len(documentList) / numDocsContaining(word,documentList))

def tfidf(word, document, documentList):
  return (tf(word,document) * idf(word,documentList))
#//////////////////////////TF-IDF Result/////////////////////////////////
##j=1
##for document in docs:
##  out.write('\tD%s' %j)
##  j+=1
##out.write('\n')
##for w in set(Token):
##  out.write(w)
##  for document in docs:
##          tf(w, document)
##          idf(w, docs)
##          float(tf(w,document) * idf(w,docs))
##          out.write('\t%s'%str(tf(w,document) * idf(w,docs)))
##  out.write('\n')
##out.close()
#//////////////////////////////

#/////////////////////Matrice d'occurence TF-IDF////////////
```

```
mat='['
A = zeros([len(set(Token)),len(docs)])
for i,w in enumerate(set(Token)):
    mat+='['
    for j,document in enumerate(docs):
          x=tf(w, document)
         # print x
          y=idf(w, docs)
         # print y
          z=float(x*y)
          #print z
          A[i,j]+=z
          if (j<(len(docs)-1)):
            mat+= "%+0.2f, "%(z)
          else:
            mat+= "%+0.2f "%(z)

    if (i<(len(set(Token))-1)):
        mat += "],\n"
    else:
        mat += "]]"
#print mat
print array(A)[:,0:3]

#////////////////////////////////////////////////////////

#////////////////////Matrice d'occurence booléen////////////
B = zeros([len(set(Token)),len(docs)])
for i,w in enumerate(set(Token)):
    for j,document in enumerate(docs):
         if w in document:
           B[i,j]+=1

#////////////////////////////////////////////////////////

#//////////Sauvegarder les différents matrices en fichiers textes/////////

#http://bytes.com/topic/python/answers/450735-help-saving-output-onto-text-
file
import sys
former, sys.stdout = sys.stdout,open('matrixBooléen.txt','w')
try:
    print B
finally:
    results, sys.stdout = sys.stdout, former
    results.close()

#http://bytes.com/topic/python/answers/450735-help-saving-output-onto-text-
file
import sys
former, sys.stdout = sys.stdout,open('matrixTFIDFW.txt','w')
try:
    print mat
finally:
    results, sys.stdout = sys.stdout, former
    results.close()

#Save_as_.dat
    #http://bytes.com/topic/python/answers/450735-help-saving-output-onto-
text-file
import sys
```

```
former, sys.stdout = sys.stdout,open('word_tfidf.dat','w')
try:
    print array(A[:,0:3])
finally:
    results, sys.stdout = sys.stdout, former
    results.close()

#////////////////////////////////////////////////////

#//////////////////LSA Transformation//////////////////////////

U, S, Vt = svd(A)

#////////////////////////////////////////////////////////////

#//////////////////LSA Result//////////////////////////////////

#http://bytes.com/topic/python/answers/450735-help-saving-output-onto-text-
file
import sys
#Save_matrixS
formerS, sys.stdout = sys.stdout,open('matrixS.txt','w')
try:
    print S
finally:
    resultS, sys.stdout = sys.stdout, formerS
    resultS.close()
#Save_matrixU
    formerU, sys.stdout = sys.stdout,open('matrixU.txt','w')
try:
    print -1*U[:, 0:3]
finally:
    resultU, sys.stdout = sys.stdout, formerU
    resultU.close()
#Save_matrixVt
    formerVt, sys.stdout = sys.stdout,open('matrixVt.txt','w')
try:
    print -1*Vt[0:3, :]
finally:
    resultVt, sys.stdout = sys.stdout, formerVt
    resultVt.close()

#////////////////////////////////////////////////////////////
```

D.1.2 Classificateur k-moyennes en Matlab

- **Fonction kMeansCluster**

```
function y=kMeansCluster(m,k,isRand)
%%%%%%%%%%%%%%%%
%
% kMeansCluster - Simple k means clustering algorithm
% Author: Kardi Teknomo, Ph.D.
%
% Purpose: classify the objects in data matrix based on the attributes
% Criteria: minimize Euclidean distance between centroids and object points
% For more explanation of the algorithm, see
http://people.revoledu.com/kardi/tutorial/kMean/index.html
```

```
% Output: matrix data plus an additional column represent the group of each
object
%
% Example: m = [ 1 1; 2 1; 4 3; 5 4]  or in a nice form
%          m = [ 1 1;
%                2 1;
%                4 3;
%                5 4]
%          k = 2
% kMeansCluster(m,k) produces m = [ 1 1 1;
%                                   2 1 1;
%                                   4 3 2;
%                                   5 4 2]
% Input:
%   m      - required, matrix data: objects in rows and attributes in
columns
%   k      - optional, number of groups (default = 1)
%   isRand - optional, if using random initialization isRand=1, otherwise
input any number (default)
%            it will assign the first k data as initial centroids
%
% Local Variables
%   f      - row number of data that belong to group i
%   c      - centroid coordinate size (1:k, 1:maxCol)
%   g      - current iteration group matrix size (1:maxRow)
%   i      - scalar iterator
%   maxCol - scalar number of rows in the data matrix m = number of
attributes
%   maxRow - scalar number of columns in the data matrix m = number of
objects
%   temp   - previous iteration group matrix size (1:maxRow)
%   z      - minimum value (not needed)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if nargin<3,        isRand=0;   end
if nargin<2,        k=1;        end

[maxRow, maxCol]=size(m)
if maxRow<=k,
    y=[m, 1:maxRow]
else

      % initial value of centroid
    if isRand,
        p = randperm(size(m,1));      % random initialization
        for i=1:k
            c(i,:)=m(p(i),:)
      end
    else
        for i=1:k
           c(i,:)=m(i,:)        % sequential initialization
      end
    end

      temp=zeros(maxRow,1);   % initialize as zero vector

      while 1,
        %d=DistMatrix(m,c);  % calculate objcets-centroid distances
        d=distmat(m,c)
        [z,g]=min(d,[],2);  % find group matrix g
        if g==temp,
```

```
            break;              % stop the iteration
        else
            temp=g;             % copy group matrix to temporary variable
        end
        for i=1:k
            f=find(g==i);
            if f                % only compute centroid if f is not empty
                c(i,:)=mean(m(find(g==i),:),1);
            end
        end
    end

    y=[m,g];

end
```

- **Fonction de Distance entre Matrice**

```
function d=DistMatrix(A,B)
%%%%%%%%%%%%%%%%%%%%%%%%%%
% DISTMATRIX return distance matrix between points in A=[x1 y1 ... w1] and
in B=[x2 y2 ... w2]
% Copyright (c) 2005 by Kardi Teknomo,  http://people.revoledu.com/kardi/
%
% Numbers of rows (represent points) in A and B are not necessarily the
same.
% It can be use for distance-in-a-slice (Spacing) or distance-between-slice
(Headway),
%
% A and B must contain the same number of columns (represent variables of n
dimensions),
% first column is the X coordinates, second column is the Y coordinates,
and so on.
% The distance matrix is distance between points in A as rows
% and points in B as columns.
% example: Spacing= dist(A,A)
% Headway = dist(A,B), with hA ~= hB or hA=hB
%          A=[1 2 3; 4 5 6; 2 4 6; 1 2 3]; B=[4 5 1; 6 2 0]
%          dist(A,B)= [ 4.69   5.83;
%                       5.00   7.00;
%                       5.48   7.48;
%                       4.69   5.83]
%
%          dist(B,A)= [ 4.69   5.00     5.48    4.69;
%                       5.83   7.00     7.48    5.83]
%%%%%%%%%%%%%%%%%%%%%%%%%%
[hA,wA]=size(A);
[hB,wB]=size(B);
if wA ~= wB,
   error(' second dimension of A and B must be the same');
end
```

```
for k=1:wA
   C{k}= repmat(A(:,k),1,hB);
   D{k}= repmat(B(:,k),1,hA);
end
S=zeros(hA,hB);
for k=1:wA
   S=S+(C{k}-D{k}').^2;
end
d=sqrt(S);
```

- **Classificateur k-moyennes en interface graphique**

```
function Clustring_FinalView
clc
clear
fh = figure('name','Data Clustring','numbertitle','off');

bh1 = uicontrol(fh,'Position',[10 300 60 30],...
                'String','DataPlot',...
                'Callback',@Data_plot);


bh2 = uicontrol(fh,'Position',[10 200 60 30],...
                'String','Kmeans',...
                'Callback',@Kmeans_plot);

bh1 = uicontrol(fh,'Position',[10 100 60 30],...
                 'String','Hcluster',...
                 'Callback',@Hcluster_plot);
% ah1 = axes('Parent',fh,'units','pixels',...
%            'Position',[120 220 300 170]);
%
% ah2 = axes('Parent',fh,'units','pixels',...
%            'Position',[120 30 300 170]);
%----------------Kmeans-----------
% m = [ 1 1; 2 1; 4 3; 5 4];%Data
k=4;
ESS = uiimport;
vars = fieldnames(ESS);
m = ESS.(vars{1});
t=kMeans(m,k);
 %Result
   %-----------------------------------
%    %-------------------------------------------------
%    function Data_Import(hObject,eventdata)
%       k=3;
%       ESS = uiimport;
```

```
%       vars = fieldnames(ESS);
%       m = ESS.(vars{1});
%
%    end
   %-------------------------------------------------
   %-------------------------------------------------
   function Data_plot(hObject,eventdata)
      for i=1:length(m)
%           subplot(3,1,1)
          plot(m(i,2),m(i,3),'*r');
%           s = sprintf('\nm%d',i);
%           text(m(i,2),m(i,3),s)
%           hold on
%           subplot(3,1,2)
%           plot(m(i,1),m(i,2),'*b');
%           hold on
%           subplot(3,1,3)
%           plot(m(i,1),m(i,3),'*g');
          hold on
      end

      title('Data Representation')
   end
   %-------------------------------------------------
   %-------------------------------------------------
   function Kmeans_plot(hObject,eventdata)
      figure('name','Kmeans Clustring','numbertitle','off');
      for i=1:length(m)
          hb=plot(m(i,2),m(i,3),'*r');
          x=[];
          y=[];
          z=[];
          hold on
          switch t(i,4)
              case 1
                  hp=plot(t(i,2),t(i,3),'om');
                  hold on
              case 2
                  hm=plot(t(i,2),t(i,3),'sb');
                  hold on
              case 3
                  hn=plot(t(i,2),t(i,3),'+g');
                  hold on
              case 4
                  ho=plot(t(i,2),t(i,3),'dc');
                  hold on
          end
      end
      hold off
      title('Kmeans Cluster')
      legend([hb,hp,hm,hn,ho],'Visualisation de données','Groupe 1','Groupe
2',...
          'Groupe 3','Groupe 4');
      xlabel('Dimension 2');
      ylabel('Dimension 3');
    end

%   %-------------------------------------------------
%-------------------------------------------------
    function  Hcluster_plot(hObject,eventdata)
       figure('name','Hierarchical Clustring','numbertitle','off');
```

```
        %# cluster the subset data
        D = pdist(m, 'euclid');
        T = linkage(D, 'ward');
        CUTOFF = 0.6*max(T(:,3));      %# CUTOFF = 5;
        C = cluster(T, 'criterion','distance', 'cutoff',CUTOFF);
        K = length( unique(C) );       %# number of clusters found

        %# visualize the hierarchy of clusters

        h = dendrogram(T, 0, 'colorthreshold',CUTOFF);
        set(h, 'LineWidth',2)
        set(gca, 'XTickLabel',[], 'XTick',[])
        title('Hierarchical Cluster')
    end
 %------------------------------------------------
end
```

D.1.3 Modèle PLSA en python

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import math,random

class plsa:

    def __init__(self,n,nz=3): # 3 catégories de phrases
        self.n = n
        self.nz = nz

        # num of the sentence
        self.ns = len(n[0])
        # num of the words
        self.nw = len(n)

        # initialize the probability
        self.pwsz = self.arr([ self.nw, self.ns, self.nz ]) # P(z|s,w)
        self.pwz  = self.arr([ self.nw, self.nz ])          # P(w|z)
        self.psz  = self.arr([ self.ns, self.nz ])          # P(s|z)
        self.pz   = self.arr([ self.nz ])                   # P(z)
        self.pws  = self.arr([ self.nw, self.ns ])          # P(s,w)


    def train(self,k=1000):
        tmp = 0
        for i in range(k):
            self.e_step()
            self.m_step()
            L = self.likelihood()
            if abs(L - tmp) < 1.0e-10:
                break
            else:
                tmp = L
            """

    @staticmethod
    def normalized(list):
```

```
        total = sum(list)
        if total == 0:
            return [1.0/len(list)] * len(list)
        return map(lambda a: a/total, list)

    @staticmethod
    def maximized(list):
        maximum = max(list)
        result = [0.0] * len(list)
        for i in range(len(list)):
            if list[i] == maximum:
                result[i] = 1.0
                break
        return result

    @staticmethod
    def arr(list):
        if len(list) > 1:
            return [ plsa.arr(list[1:]) for i in range(list[0]) ]
        else:
            return plsa.normalized([ random.random() for i in
range(list[0]) ])

    def likelihood(self):
        """ log-liklihood """
        # P(s,w)= Σz(pz * pzw * pzs)
        for w in range(self.nw):
            for s in range(self.ns):
                self.pws[w][s] = sum([
self.pz[z]*self.psz[s][z]*self.pwz[w][z] for z in range(self.nz) ])
        # Σ n(s,w) log P(s,w)
        return sum([ self.n[w][s]*plsa.mylog(self.pws[w][s]) for w in
range(self.nw) for s in range(self.ns) ])

    @staticmethod
    def mylog(x):
        if x == 0:
            return -float("inf")
        return math.log(x)

    def e_step(self):
        """ E-step """
        # P(z|s,w)= (pz * pzw * prs)/Σz' (pz' * pz'w * pz's)
        for w in range(self.nw):
            for s in range(self.ns):
                for z in range(self.nz):
                    self.pwsz[w][s][z] =
self.pz[z]*self.psz[s][z]*self.pwz[w][z]
                self.pwsz[w][s] = self.maximized(self.pwsz[w][s])


    def m_step(self):
        """ M-step """
        # P(w|z)= Σs n(s,w) * log P(s,w)/Σz' (pz' * pz'w * pz's)
        for w in range(self.nw):
            for z in range(self.nz):
                self.pwz[w][z] = sum([ self.n[w][s]*self.pwsz[w][s][z] for
s in range(self.ns) ])
            self.pwz[z] = self.normalized(self.pwz[z])

        # P(s|z)= Σw n(s,w) * log P(s,w)/Σz' (pz' * pz'w * pz's)
```

```
        for s in range(self.ns):
            for z in range(self.nz):
                self.psz[s][z] = sum([ self.n[w][s]*self.pwsz[w][s][z] for
w in range(self.nw) ])
            self.psz[z] = self.normalized(self.psz[z])

        # P(z)= Σsw n(s,w) * log P(s,w)/Σz' (pz' * pz'w * pz's)
        for z in range(self.nz):
            self.pz[z] = sum([ self.n[w][s]*self.pwsz[w][s][z] for w in
range(self.nw) for s in range(self.ns) ])
        self.pz = self.normalized(self.pz)

if __name__ == '__main__':
    n = [[1, 0,  0,        0,         0,         0,         0,
0,        0,         0,         1,         0,         0],
         [1,     0,    1,    1,    0,    0,    0,    0,    1,    0,    1,
      1,    0],
         [0,     1,    0,    0,    0,    0,    0,    0,    0,    0,    0,
      0,    0],
         [0,     1,    0,    0,    0,    1,    0,    1,    0,    0,    0,
      0,    0],
         [0,     0,    1,    0,    0,    0,    0,    0,    0,    0,    0,
      0,    0],
         [0,     0,    0,    1,    0,    0,    0,    0,    0,    0,    0,
      0,    0],
         [0,     0,    0,    1,    0,    0,    0,    0,    1,    0,    0,
      0,    0],
         [0,     0,    0,    0,    1,    0,    0,    0,    0,    0,    0,
      0,    0],
         [0,     0,    0,    0,    1,    0,    1,    0,    0,    1,    0,
      0,    1],
         [0,     0,    0,    0,    1,    0,    1,    0,    0,    0,    0,
      0,    1],
         [0,     0,    0,    0,    0,    1,    0,    0,    0,    1,    0,
      0,    0],
         [0,     0,    0,    0,    0,    0,    1,    0,    0,    1,    0,
      0,    0],
         [0,     0,    0,    0,    0,    0,    0,    0,    0,    0,    1,
      0,    0],
         [0,     0,    0,    0,    0,    0,    0,    0,    0,    0,    1,
      0,    0],
         [0,     0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
      1,    0],
         [0,     0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
      1,    0],
         [0,     0,    0,    0,    0,    0,    0,    0,      0,        0,
        0,          0,         1]]
    p = plsa(n)
    p.train()
    print 'pz', p.pz
    print 'psz', p.psz
    print 'pwz', p.pwz
    print 'pwsz', p.pwsz
```

Sortie du code PLSA

```
IDLE 2.6       ==== No Subprocess ====
>>>
```

```
pz [0.45454545454545453, 0.060606060606060608, 0.48484848484848486]
psz [[0.0, 0.0, 2.0], [1.0, 1.0, 0.0], [1.0, 0.0, 0.0], [3.0, 0.0,
0.0], [1.0, 0.0, 2.0], [2.0, 0.0, 0.0], [0.0, 1.0, 2.0], [1.0, 0.0,
0.0], [2.0, 0.0, 0.0], [1.0, 0.0, 2.0], [0.0, 0.0, 4.0], [0.0, 0.0,
3.0], [2.0, 0.0, 1.0]]
pwz [[0.0, 0.0, 2.0], [3.0, 0.0, 3.0], [0.0, 1.0, 0.0], [3.0, 0.0,
0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [2.0, 0.0, 0.0], [0.0, 0.0,
1.0], [3.0, 1.0, 0.0], [0.0, 0.0, 3.0], [1.0, 0.0, 1.0], [0.0, 0.0,
2.0], [0.0, 0.0, 1.0], [0.0, 0.0, 1.0], [0.0, 0.0, 1.0], [0.0, 0.0,
1.0], [1.0, 0.0, 0.0]]
pwsz [[[0.0, 0.0, 1.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0,
0.0], [0.0, 0.0, 1.0], [1.0, 0.0, 0.0], [0.0, 0.0, 1.0], [1.0, 0.0,
0.0], [1.0, 0.0, 0.0], [0.0, 0.0, 1.0], [0.0, 0.0, 1.0], [0.0, 0.0,
1.0], [0.0, 0.0, 1.0]], [[0.0, 0.0, 1.0], [1.0, 0.0, 0.0], [1.0,
0.0, 0.0], [1.0, 0.0, 0.0], [0.0, 0.0, 1.0], [1.0, 0.0, 0.0], [0.0,
0.0, 1.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [0.0, 0.0, 1.0], [0.0,
0.0, 1.0], [0.0, 0.0, 1.0], [1.0, 0.0, 0.0]], [[1.0, 0.0, 0.0],
[0.0, 1.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0],
[1.0, 0.0, 0.0], [0.0, 1.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0],
[1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0]],
[[1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0],
[1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0],
[1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0],
[1.0, 0.0, 0.0]], [[1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0,
0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0,
0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0,
0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0]], [[1.0, 0.0, 0.0], [1.0,
0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0,
0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0,
0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0]], [[1.0, 0.0, 0.0],
[1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0],
[1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0],
[1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0],
[1.0, 0.0, 0.0]], [[0.0, 0.0, 1.0], [1.0, 0.0, 0.0], [1.0, 0.0,
0.0], [1.0, 0.0, 0.0], [0.0, 0.0, 1.0], [1.0, 0.0, 0.0], [0.0, 0.0,
1.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [0.0, 0.0, 1.0], [0.0, 0.0,
1.0], [0.0, 0.0, 1.0], [0.0, 0.0, 1.0]], [[1.0, 0.0, 0.0], [1.0,
0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [0.0, 0.0, 1.0], [1.0,
0.0, 0.0], [0.0, 0.0, 1.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [0.0,
0.0, 1.0], [0.0, 0.0, 1.0], [0.0, 0.0, 1.0], [0.0, 0.0, 1.0]],
[[0.0, 0.0, 1.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0],
[0.0, 0.0, 1.0], [1.0, 0.0, 0.0], [0.0, 0.0, 1.0], [1.0, 0.0, 0.0],
[1.0, 0.0, 0.0], [0.0, 0.0, 1.0], [0.0, 0.0, 1.0], [0.0, 0.0, 1.0],
[0.0, 0.0, 1.0]], [[0.0, 0.0, 1.0], [1.0, 0.0, 0.0], [1.0, 0.0,
0.0], [1.0, 0.0, 0.0], [0.0, 0.0, 1.0], [1.0, 0.0, 0.0], [0.0, 0.0,
1.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [0.0, 0.0, 1.0], [0.0, 0.0,
1.0], [0.0, 0.0, 1.0], [1.0, 0.0, 0.0]], [[0.0, 0.0, 1.0], [1.0,
0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [0.0, 0.0, 1.0], [1.0,
0.0, 0.0], [0.0, 0.0, 1.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [0.0,
0.0, 1.0], [0.0, 0.0, 1.0], [0.0, 0.0, 1.0], [0.0, 0.0, 1.0]],
[[0.0, 0.0, 1.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0],
[0.0, 0.0, 1.0], [1.0, 0.0, 0.0], [0.0, 0.0, 1.0], [1.0, 0.0, 0.0],
[1.0, 0.0, 0.0], [0.0, 0.0, 1.0], [0.0, 0.0, 1.0], [0.0, 0.0, 1.0],
[0.0, 0.0, 1.0]], [[0.0, 0.0, 1.0], [1.0, 0.0, 0.0], [1.0, 0.0,
0.0], [1.0, 0.0, 0.0], [0.0, 0.0, 1.0], [1.0, 0.0, 0.0], [0.0, 0.0,
1.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [0.0, 0.0, 1.0], [0.0, 0.0,
1.0], [0.0, 0.0, 1.0], [1.0, 0.0, 0.0]], [[0.0, 0.0, 1.0], [1.0,
0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [0.0, 0.0, 1.0], [1.0,
0.0, 0.0], [0.0, 0.0, 1.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [0.0,
0.0, 1.0], [0.0, 0.0, 1.0], [0.0, 0.0, 1.0], [0.0, 0.0, 1.0]],
[[0.0, 0.0, 1.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0],
```

```
      [0.0, 0.0, 1.0], [1.0, 0.0, 0.0], [0.0, 0.0, 1.0], [1.0, 0.0, 0.0],
      [1.0, 0.0, 0.0], [0.0, 0.0, 1.0], [0.0, 0.0, 1.0], [0.0, 0.0, 1.0],
      [0.0, 0.0, 1.0]], [[1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0,
      0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0,
      0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0], [1.0, 0.0,
      0.0], [1.0, 0.0, 0.0], [1.0, 0.0, 0.0]]]
>>>
```

D.2 Phase de Test

D.2.1 Base de données de Scolarité

Du fait que l'application de Scolarité est une application d'interrogation d'une base de données, alors la première étape à effectuer est de créer cette base (à des fins expérimentales seulement, les données ne sont pas réelles tandis que les spécialités, option et module le sont).

```
from pysqlite2 import dbapi2 as sqlite
# Create user database
con = sqlite.connect('sample.db')
cur = con.cursor()
ex = cur.execute
# créer la table Etudiants
ex('create table IF NOT EXISTS Etudiants(inscription,nom,date_de_naissance,
code_degre,annee_obtention)')
# créer la table Degre
ex('create table IF NOT EXISTS Degre(code_degre,designation)')
# créer la table Module
ex('create table IF NOT EXISTS Module(code_module,designation)')
# créer la table Etudier
ex('create table IF NOT EXISTS
Etudier(inscription,code_sp,code_module,note)')
# créer la table specilaite
ex('create table IF NOT EXISTS Specialite(code_sp,designation)')
# créer la table Annee
ex('create table IF NOT EXISTS Annee(annee_scolaire)')
##con.commit()
##con.close()
##
####
##########################################################################
##########################################################################
################Vérifier si les tables ont bien été créer#################
##ex("select name from sqlite_master where type = 'table'").fetchall()#
####[(u'Etudiants',), (u'Degre',), (u'Module',), (u'Etudier',),##########
####(u'Specialite',), (u'Annee',)]#######################################
##########################################################################
##########################################################################
con = sqlite.connect('sample.db')
cur = con.cursor()
```

```
ex = cur.execute
### Charger la table Etudiants
##for m in [('16001','Mohamed LICHOURI', '7/02/1985','006','2007'),
##          ('16002','Wahib DJAOUTI','12/08/1978','005','2007'),
##          ('16003','Chakib BERZALI','2/12/1974','001','2009/2010'),
##          ('16004','Rabie SBAA','2/12/1977','001','2010/2011'),
##          ('16005','Abdelhakim KOBI','2/12/1988','002','2011/2012')]:
##    ex('insert into Etudiants values(?,?,?,?,?)', m)
##
#################################################################################
###################################
#################################################################################
###################################
### Charger la table Degre
##for n in [('001','Ingenieur Electronique'),('002','Magistere
Electronique'),('003','Doctorat Electronique'),
##          ('004','Troisieme annee Electronique'),('005','Quatrieme annee
Electronique'),
##          ('006','Cinquieme annee Electronique')]:
##    ex('insert into Degre values(?,?)', n)
##
#################################################################################
###################################
#################################################################################
###################################
### Charger la table Module
##for t in [('001','electronique generale'),('002','electronique
numerique'),('003','traitement de signale'),
##
('004','electricite'),('005','mesure'),('006','champ'),('007','antenne'),('
008','capteur'),
##          ('009','reseaux'),('010','fibre
optique'),('012','television'),('013','systeme de
telecommunication'),('014','micro onde')]:
##    ex('insert into Module values(?,?)', t)
##
#################################################################################
###################################
#################################################################################
###################################
### Charger la table Etudier
##for s in
[('16001','001','012','12'),('16001','001','006','10'),('16001','001','014'
,'08.5'),
##
('16001','001','010','14'),('16001','001','013','09.5'),('16002','001','001
','12'),('16002','001','003','10'),
##
('16002','001','006','08.5'),('16002','001','007','14'),('16002','001','002
','09.5')]:
##    ex('insert into Etudier values(?,?,?,?)', s)
##
#################################################################################
###################################
#################################################################################
###################################
### Charger la table Specialite
##for q in
[('001','Communication'),('002','Controle'),('003','Instrumentation')]:
##    ex('insert into Specialite values(?,?)', q)
##
```

```
#########################################################################
##################################
#########################################################################
##################################
### Charger la table Annee
##
##
#########################################################################
##################################
#########################################################################
##################################
##con.commit()
##con.close()
```

D.2.2 Phase de compréhension

❖ Etiquetage des phrases

```
##Définir les différents concepts:
##Concept:
##Ordre;Demande;Date;Objet
Ordre_concept='montrer,indiquer,sélectionner,trouver,donner,afficher,présen
ter,prendre,passer,chercher,souhaite,faut,désire,désirerais,peux,pourrais,v
eux,voudrais,possible,aimerais,souhaiterais'
Information_concept='note,certificat,diplome,scolarite'
Valeur_concept='module,examen,test'
Module_concept=['electronique generale','electronique
numerique','traitement de
signale','electricite','mesure','champ','antenne','capteur','reseaux','fibr
e optique','television','systeme de telecommunication']
stopword=open('french.stoplist.txt').read()


def Make_Concept(doc):
    out=open('concept.txt','w')
    token=[word for word in doc.lower().split()
           if (word not in stopword.split() and len(word)>3)]
    #print '***************Extraction de Concept************'
    for t in token:
        if t in Ordre_concept.split(','):
            z='Ordre'
        elif t in Information_concept.split(','):
            z='Information'
        elif t in Valeur_concept.split(','):
            z='Valeur'
        elif t in [w.lower() for w in Module_concept]:
            z='Module'
        else:
            z=t
        #print t,
        out.write('%s\t'%z)
    out.close()
    Concept2Sens('concept.txt')
```

❖ Moteur d'inférence

Sert à construire les requêtes génériques.

```
def Concept2Sens(filename):
    Q=text.get()
    #ReOpen_the_concept_file
    #filename in generale is 'concept.txt'
    concept=open(filename).read().replace('\t',' ').split()
    outt=open('sens.txt','w')
    #print '\n **************Concept2Sens************'
    if ' '.join(concept)=='Information':
        #print 'select Information from usersTable where code'
        outt.write('select Information from usersTable where code')
        Sens2Sql('Ordre Information')
    elif ' '.join(concept)=='Ordre Information':
        #print 'select Information from usersTable where code'
        outt.write('select Information from usersTable where code')
        Sens2Sql('Ordre Information')
    elif ' '.join(concept)=='Ordre Information Valeur':
        #print 'select Note from Etudier where code'
        outt.write('select Note from Etudier where code')
        Sens2Sql('Ordre Information Valeur')
    elif ' '.join(concept)=='Ordre Information Valeur Module':
        #print 'select Note from Etudier where code&module'
        outt.write('select Note from Etudier where code&module')
        Sens2Sql('Ordre Information Valeur Module')
    elif ' '.join(concept)=='Ordre Valeur Module':
        #print 'select Note from Etudier where code&module'
        outt.write('select Note from Etudier where code&module')
        Sens2Sql('Ordre Information Valeur Module')
    elif ' '.join(concept)=='Module':
        #print 'select Information from usersTable where code'
        outt.write('select Information from usersTable where code')
        Sens2Sql('Ordre Information Valeur Module')
    elif ' '.join(concept)=='Ordre Information Module':
        #print 'select Information from usersTable where code'
        outt.write('select Information from usersTable where code')
        Sens2Sql('Ordre Information Module')
    else:
        Unknown_Solver(' '.join(concept))

    outt.close()
```

❖ Génération des requêtes SQL

```
def Sens2Sql(concept):
    Q=text.get()
    #print '**************Sens2SQL************'
    if concept=='Ordre Information':
        #print 'select Information from usersTable where code'
        #s=raw_input("Veuiller indiquer votre numero d'inscription: ")
        for w in Q.split():
             #if w in Information_concept:
             if w=='note':
                  #t=raw_input( "Vous voulez votre Note! ")
                  vbs_Create('Vous voulez votre Note ')
                  os.startfile("file2.vbs")
                  if tkMessageBox.askyesno(END,"IM: Vous voulez votre Note!
"):
                       listbox.insert(END,"[%s] <IM>: Veuiller indiquer
votre numero d'inscription: " % now)
                       num = tkSimpleDialog.askinteger (title="Numéro
d'inscription",
```

```
                                        prompt="Entrer votre matricule")
                        z=num
                        #print ex('select nom from Etudiants where
inscription="%s"' %(s)).fetchall()
                        listbox.insert(END,ex('select nom from Etudiants
where inscription="%s"' %(z)).fetchall())
                        #print ex('select note from Etudier where
inscription="%s"' %(s)).fetchall()
                        listbox.insert(END,ex('select note from Etudier where
inscription="%s"' %(z)).fetchall())

                elif w=='certificat':
                        #t=raw_input( "Vous voulez votre Certificat de Scolarite!
")
                        #listbox.insert(END, "IM: Vous voulez votre Certificat de
Scolarite! ")
                        vbs_Create('Vous voulez votre Certificat de Scolarite')
                        os.startfile("file2.vbs")
                        if tkMessageBox.askyesno(END,"IM: Vous voulez votre
Certificat de Scolarite! "):
                                listbox.insert(END,"[%s] <IM>: Veuiller indiquer
votre numero d'inscription: " % now)
                                num = tkSimpleDialog.askinteger (title="Numéro
d'inscription",
                                        prompt="Entrer votre matricule")
                                z=num
                                nom=ex('select nom from Etudiants where
inscription="%s"' %(z)).fetchall()
                                dateNaissance=ex('select date_de_naissance from
Etudiants where inscription="%s"' %(z)).fetchall()
                                niveau=ex('select designation from Degre where
code_degre =(select code_degre from Etudiants where inscription="%s")'
%(z)).fetchall()
                                certificat_format(nom, dateNaissance, niveau)

                elif w=='diplome':
                        #t=raw_input("Vous voulez votre diplome! ")
                        vbs_Create('Vous voulez votre diplome')
                        os.startfile("file2.vbs")
                        if tkMessageBox.askyesno(END,"IM: Vous voulez votre
diplome! "):
                                listbox.insert(END,"[%s] <IM>: Veuiller indiquer
votre numero d'inscription: " % now)
                                num = tkSimpleDialog.askinteger (title="Numéro
d'inscription",
                                        prompt="Entrer votre matricule")
                                z=num
                                nom=ex('select nom from Etudiants where
inscription="%s"' %(z)).fetchall()
                                dateNaissance=ex('select date_de_naissance from
Etudiants where inscription="%s"' %(z)).fetchall()
                                niveau=ex('select designation from Degre where
code_degre =(select code_degre from Etudiants where inscription="%s"
)'%(z)).fetchall()
                                diplome_format(nom, dateNaissance, niveau)
        elif concept=='Ordre Information Valeur':
                #print 'select note from Etudier where code_module'
                #t=raw_input( "Vous voulez votre Note! ")
                vbs_Create('Vous voulez votre Note')
                os.startfile("file2.vbs")
                if tkMessageBox.askyesno(END,"IM: Vous voulez votre Note! "):
```

```
                #s=raw_input("Veuiller indiquer votre numero
d'inscription: ")
                #print ex('select nom from Etudiants where
inscription="%s"' %(s)).fetchall()
                listbox.insert(END,"[%s] <IM>: Veuiller indiquer votre
numero d'inscription: " % now)
                num = tkSimpleDialog.askinteger (title="Numéro
d'inscription",
                                    prompt="Entrer votre matricule")
                z=num
                listbox.insert(END,ex('select nom from Etudiants where
inscription="%s"' %(z)).fetchall())
                #print ex('select note from Etudier where
inscription="%s"' %(s)).fetchall()
                listbox.insert(END,ex('select note from Etudier where
inscription="%s"' %(z)).fetchall())
    elif concept=='Ordre Information Valeur Module':
        #print 'select note from Etudier where inscription&module'
        #t=raw_input( "Vous voulez votre Note! ")
        vbs_Create('Vous voulez votre Note')
        os.startfile("file2.vbs")
        if tkMessageBox.askyesno(END,"IM: Vous voulez votre Note! "):
                #s=raw_input("Veuiller indiquer votre numero d'inscription:
")
                listbox.insert(END,"[%s] <IM>: Veuiller indiquer votre
numero d'inscription: " % now)
                num = tkSimpleDialog.askinteger (title="Numéro
d'inscription",
                                    prompt="Entrer votre matricule")
                z=num
                for w in [w.lower() for w in Module_concept]:
                        if w in Q:
                                #print ex('select nom from Etudiants where
inscription="%s"' %(s)).fetchall()
                                listbox.insert(END,ex('select nom from
Etudiants where inscription="%s"' %(z)).fetchall())
                                t=ex('select code_module from Module where
designation="%s"'%(w)).fetchall()
                                # Pour convertir la liste en caractére
                                x=str(" ".join(["%s" % el for el in t]))
                                #print ('La note du module de #%s#'%w),'est
egale à ',ex('select note from Etudier where inscription="%s" and
code_module="%s"' %(s,x)).fetchall()
                                listbox.insert(END,('IM: La note du module
de #%s#'%w),'est egale à ',ex('select note from Etudier where
inscription="%s" and code_module="%s"' %(z,x)).fetchall())

    elif concept=='Ordre Information Module':
        #print 'select Information from usersTable where code'
        #s=raw_input("Veuiller indiquer votre numero d'inscription: ")

        for w in Q.split():
                if w=='note':
                     #t=raw_input( "Vous voulez votre Note! ")
                     vbs_Create('Vous voulez votre Note')
                     os.startfile("file2.vbs")
                     if tkMessageBox.askyesno(END,"IM: Vous voulez votre
Note! "):
                          listbox.insert(END,"[%s] <IM>: Veuiller indiquer
votre numero d'inscription: " % now)
```

```
                            num = tkSimpleDialog.askinteger (title="Numéro
d'inscription",
                                  prompt="Entrer votre matricule")
                            z=num
                            for d in Q.split():
                                 if d in [w.lower() for w in Module_concept]:
                                      #print ex('select nom from Etudiants
where inscription="%s"' %(s)).fetchall()
                                      listbox.insert(END,ex('select nom from
Etudiants where inscription="%s"' %(z)).fetchall())
                                      t=ex('select code_module from Module
where designation="%s"'%(d)).fetchall()
                                      # Pour convertir la liste en caractére
                                      x=str(" ".join(["%s" % el for el in
t]))
                                      v=ex('select note from Etudier where
inscription="%s" and code_module="%s"' %(z,x)).fetchall()
                                      #print ('La note du module de
#%s#'%d),('est egale à %s'%v)
                                      listbox.insert(END,('IM: La note du
module de #%s#'%d),('est egale à %s'%v))

                elif w=='certificat':
                     #t=raw_input( "Vous voulez votre Certificat de
Scolarite! ")
                     vbs_Create('Vous voulez votre Certificat de
Scolarite')
                     os.startfile("file2.vbs")
                     if tkMessageBox.askyesno(END,"IM: Vous voulez votre
Certificat de Scolarite! "):
                            listbox.insert(END,"[%s] <IM>: Veuiller indiquer
votre numero d'inscription: " % now)
                            num = tkSimpleDialog.askinteger (title="Numéro
d'inscription",
                                  prompt="Entrer votre matricule")
                            z=num
                            nom=ex('select nom from Etudiants where
inscription="%s"' %(z)).fetchall()
                            dateNaissance=ex('select date_de_naissance from
Etudiants where inscription="%s"' %(z)).fetchall()
                            niveau=ex('select designation from Degre where
code_degre =(select code_degre from Etudiants where inscription="%s")'
%(z)).fetchall()
                            certificat_format(nom, dateNaissance, niveau)

                elif w=='diplome':
                     #t=raw_input("Vous voulez votre diplome! ")
                     vbs_Create('Vous voulez votre diplome')
                     os.startfile("file2.vbs")
                     if tkMessageBox.askyesno(END,"IM: Vous voulez votre
diplome! "):
                            listbox.insert(END,"[%s] <IM>: Veuiller indiquer
votre numero d'inscription: " % now)
                            num = tkSimpleDialog.askinteger (title="Numéro
d'inscription",
                                  prompt="Entrer votre matricule")
                            z=num
                            nom=ex('select nom from Etudiants where
inscription="%s"' %(z)).fetchall()
                            dateNaissance=ex('select date_de_naissance from
Etudiants where inscription="%s"' %(z)).fetchall()
```

```
                            niveau=ex('select designation from Degre where
code_degre =(select code_degre from Etudiants where inscription="%s"
)'%(z)).fetchall()
                            diplome_format(nom, dateNaissance, niveau)
```

❖ Gestionnaire de mots hors vocabulaire

```
def Unknown_Solver(doc):
    a=re.search('Information',doc)
    b=re.search('Valeur',doc)
    c=re.search('Module',doc)
    if a is not None:
        if b is not None:
            if c is not None: # "j'espere avoir ma note du module de champ"
                Sens2Sql('Ordre Information Valeur Module')
            else: # "j'espere avoir ma note d'examen "
                Sens2Sql('Ordre Information Valeur')
        else:
            if c is not None: # " j'espere avoir ma note du champ"
                Sens2Sql('Ordre Information Module')
            else: # " j'espere avoir ma note "
                Sens2Sql('Ordre Information')
    else:
         if b is not None:
             if c is not None: # " combien j'ai eu en module de champ"
                Sens2Sql('Ordre Information Valeur Module')
             else: # "Combien j'ai eu en examen"
                Sens2Sql('Ordre Information Valeur')
         else:
             if c is not None: # "Combien j'ai eu en champ"
                Sens2Sql('Ordre Information Valeur Module')
             else:
                 Erreur()
```

❖ Interface Graphique

```
#////////////////////////////////////Interface
Graphique//////////////////////
now = time.strftime('%H:%M:%S')
os.startfile('speack.vbs')
"""This is the GUI"""
root = Tk()
root.geometry("600x400")
root.title("Système de Consultation Scolaire V1.5")

def Enter():
    #if type(text.get())<> int:
    Q = text.get()
    listbox.insert(END,"[%s] <User>: %s"%(now,Q))
    Make_Concept(Q)
    #Concept2Sens('concept.txt')
    text.delete(0,END)

def ReturnInsert(event):
            Enter()

def CopyToText(event):
            text.delete(0, END)
            current_note = listbox.get(ANCHOR)
            text.insert(0, current_note)
```

```
def CloseMsg():
    #print " Merci d'avoir choisit notre systeme. Esperant vous avoir
bientot"
    os.startfile('fileClose.vbs')
    root.destroy()

def Erreur():
    out=open('concept.txt','w')
    vbs_Create('Est-ce que vous étes satisfait ')
    os.startfile("file2.vbs")
    if tkMessageBox.askyesno(END,"IM: Est-ce que vous étes satisfait
Mr/Mme? "):
        vbs_Create('On vous souhaite une bonne journée ')
        os.startfile("file2.vbs")
        listbox.insert(END, "[%s] <IM> : On vous souhaite une bonne journée
Mr/Mme" % now)
    else:
        vbs_Create('Veuillez préciser votre requète ')
        os.startfile("file2.vbs")
        listbox.insert(END, "[%s] <IM> : Veuillez Mr/Mme préciser votre
requète" % now)


textframe = Frame(root)
listframe = Frame(root)

enter_button = Button(textframe, text="Entrer", command = Enter)

text = Entry(textframe)

scrollbar = Scrollbar(listframe, orient=VERTICAL)
listbox = Listbox(listframe, yscrollcommand=scrollbar.set,
selectmode=EXTENDED)
scrollbar.configure(command=listbox.yview)

listbox.insert(END, "[%s] <IM> : Que voulez-vous Mr/Mme?" % now)



text.bind("<Return>", ReturnInsert)
listbox.bind("<Double-Button-1>", CopyToText)

text.pack(side=LEFT, fill=X, expand=1)
enter_button.pack(side=LEFT)
listbox.pack(side=LEFT,fill=BOTH, expand=1)
scrollbar.pack(side=RIGHT, fill=Y)

textframe.pack(fill=X)
listframe.pack(fill=BOTH, expand=1)



#////////////////////////////FIN////////////////////////////////////////////
//
```

❖ Interface Vocale

```
###/////////////////////////////////////////
```

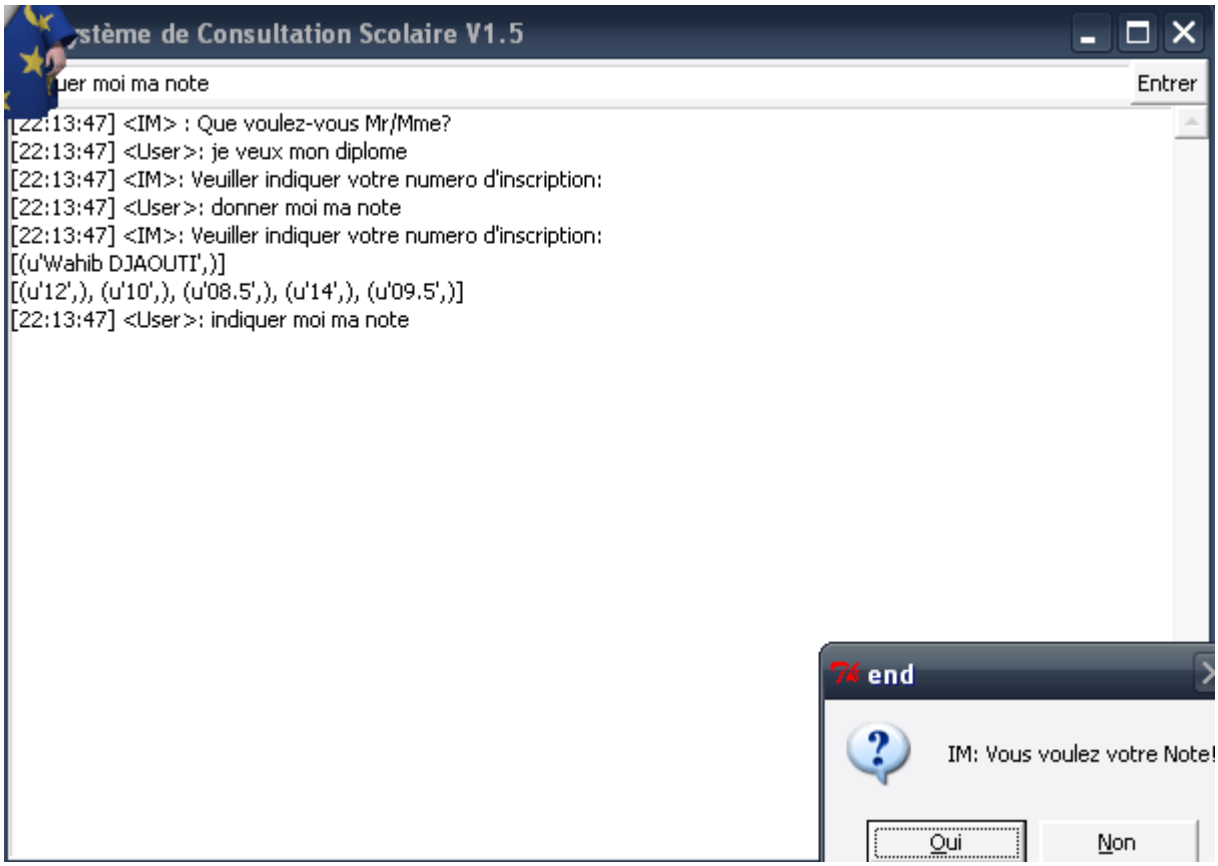
```
#////////////////////Interface Vocale/////////////////////////////
def vbs_Create(text):
    s=open('file2.vbs','w')
    s.write('message="%s" \n'%text)
    s.write('agFl = "C:\Windows\Msagent\Chars\merlin.acs" \n')
    s.write('Set objAgent = CreateObject("Agent.Control.2") \n')
    s.write('objAgent.Connected = TRUE \n')
    s.write('objAgent.Characters.Load "Merlin", agFl \n')
    s.write('objAgent.Characters("Merlin").LanguageID = &H40C \n')
    s.write('Set objCharacter = objAgent.Characters.Character("Merlin")
\n')
    s.write('objCharacter.Show \n')
    s.write('objCharacter.Speak message \n')
    s.write('objCharacter.Hide \n')
    s.write('Do While objCharacter.Visible = TRUE \n')
    s.write(' \t\t Wscript.Sleep 250 \n')
    s.write('Loop')
    s.close()
##
###////////////////////////////////////////////////
```

❖ Modèle de sortie

```
#////////////////////////////////////////////////////////////////
#////////////////////////////////////////////////////////////////
# Ouvrir une connection à la base de données
con = sqlite.connect('sample.db')
cur = con.cursor()
ex = cur.execute

#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

def certificat_format(nom, dateNaissance, niveau):
  crtf=open('attestation.txt','w+')
  crtf.write('\t\t\t\t CERTIFICAT DE SCOLARITE \n\n\n\n')
  crtf.write('\t\t Je soussigné, Monsieur le directeur du departement du
physique,atteste que:\n')
  crtf.write('\t\t\t\t\t\t\t\t\t Monsieur %s\n'%nom)
  crtf.write('\t\t\t\t\t\t\t\t\t Ne le %s\n'%dateNaissance)
  crtf.write("\t\t Etudiant en %s , pour l'annee universitaire
2010/2011"%niveau)
  crtf.write("\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t fait aujourd'hui")
  crtf.close()
  os.startfile('attestation.txt')

def diplome_format(nom, dateNaissance, niveau):
  dplm=open('diplome.txt','w+')
  dplm.write('\t\t\t\t Diplome de fin de formation \n\n\n\n')
  dplm.write('\t\t Je soussigné, Monsieur le directeur du departement du
physique,atteste que:\n')
  dplm.write('\t\t\t\t\t\t\t\t\t Monsieur %s\n'%nom)
  dplm.write('\t\t\t\t\t\t\t\t\t Ne le %s\n'%dateNaissance)
  dplm.write("\t\t A complete avec succe le programme de formation en %s"%
niveau)
  dplm.write("\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t fait aujourd'hui")
  dplm.close()
  os.startfile('diplome.txt')
```

```
#////////////////////////////////////////////#////////////////////////////////
////////
#////////////////////////////////////////////#////////////////////////////////
////////
#////////////////////////////////////////////#////////////////////////////////
////////
#////////////////////////////////////////////#////////////////////////////////
////////
```

D.2.3 Exemple de traitement des requêtes par l'application de scolarité



N.B :

Pour cette méthode d'Extraction de Concept, on ne considérera que l'application de Consultation Scolaire du fait qu'on ne peut l'appliquer avec celle du Diagnostique Médicale.

Annexe F : Différents Algorithme utilisé par le SyCE basé sur la méthode de la Classification de Phrase

E.1 Phase d'apprentissage

E.1.1 Modèle LSA

Ceci est le même que celui du premier système à la différence qu'ici on va travailler avec la matrice Vt plutôt qu'avec U pour la classification.

E.1.2 Classification de Phrases à l'aide du classificateur k-moyenne

C'est toujours le même que celui précédent à la différence qu'au fait qu'on travail avec la matrice Vt, on va modifier un petit peu l'algorithme des k-moyenne en travaillant avec m' (transposé) plutôt qu'avec m.

E.1.3 Modèle PLSA

La différence se situe à ce niveau seulement

```
    def get_pdz(self):
        return [self.normalized([self.pzw[j][i] * self.pz[j] for j in
range(self.nz)]) for i in range(self.nd)]
```

E.2 Phase de Teste

E.2.1 Application de Scolarité

- **Base de données**

C'est la même base que celle de la première méthode (Extraction de Concept).

- **Classificateur Bayesienne**

```
# Clear the previous debug
import logging
logging.root.setLevel(logging.INFO) # will suppress DEBUG level events
# Import the needed bibliotheque
import os
import re
import math
import cPickle
from pysqlite2 import dbapi2 as sqlite

def getwords(doc):
  splitter=re.compile('\\W*')
  words=[s.lower() for s in splitter.split(doc)
```

```
          if len(s)>2 and len(s)<20]

  # Return the unique set of words only
  return dict([(w,1) for w in words])

#def entryfeatures(entry):


def sampletrain(cl):
  cl.train('puis-je avoir mon relevee de note.','note')
  cl.train('donne moi mon diplome.','diplome')
  cl.train('je souhaite avoir ma certificat de scolarite.','certificat')
  cl.train('puis-je avoir mon diplome.','diplome')
  cl.train('pouver vous me montrer mon relevee de note.','note')
  cl.train("j'aimerai bien avoir ma certificat de scolarite.",'certificat')
  cl.train('je veux bien avoir la moyenne du module de math','note')
  cl.train("je veux savoir le resultat d'examen du physique",'note')
  cl.train("combien j'ai eu en chimie",'note')
  cl.train("j'aimerais bien avoir ma note du math",'note')
  cl.train('veuiller me donner la note du module de chimie','note')
  cl.train('donner moi les resultats du module de math et informatique','note')
  cl.train("resultat d'examen du module d'informatique",'note')

class classifier:
  def __init__(self,getfeatures):
    self.fc={}
    self.cc={}
    self.getfeatures=getfeatures

  def setdb(self,dbfile):
    self.con=sqlite.connect(dbfile)
    self.con.execute('create table if not exists fc(feature,category,count)') #
feature: ex: 'je veux ma note'===veux,note
    self.con.execute('create table if not exists cc(category,count)')

  def incf(self,f,cat):
    count=self.fcount(f,cat)
    if count==0:
      self.con.execute("insert into fc values ('%s','%s',1)"
                       % (f,cat))
    else:
      self.con.execute(
        "update fc set count=%d where feature='%s' and category='%s'"
        % (count+1,f,cat))

  def fcount(self,f,cat):
    res=self.con.execute(
      'select count from fc where feature="%s" and category="%s"'
      %(f,cat)).fetchone()
    if res==None: return 0
    else: return float(res[0])

  def incc(self,cat):
    count=self.catcount(cat)
    if count==0:
      self.con.execute("insert into cc values ('%s',1)" % (cat))
    else:
      self.con.execute("update cc set count=%d where category='%s'"
                       % (count+1,cat))

  def catcount(self,cat):
```

```
    res=self.con.execute('select count from cc where category="%s"'
                         %(cat)).fetchone()
    if res==None: return 0.0
    else: return float(res[0])

  def categories(self):
    cur=self.con.execute('select category from cc');
    return [d[0] for d in cur]

  def totalcount(self):
    res=self.con.execute('select sum(count) from cc').fetchone();
    if res==None: return 0
    return res[0]


  """
  def incf(self,f,cat):
    self.fc.setdefault(f,{})
    self.fc[f].setdefault(cat,0)
    self.fc[f][cat]+=1

  def incc(self,cat):
    self.cc.setdefault(cat,0)
    self.cc[cat]+=1

  def fcount(self,f,cat):
    if f in self.fc and cat in self.fc[f]:
      return float(self.fc[f][cat])
    return 0.0

  def catcount(self,cat):
    if cat in self.cc:
      return float(self.cc[cat])
    return 0

  def totalcount(self):
    return sum(self.cc.values())

  def categories(self):
    return self.cc.keys()
  """


  def train(self,item,cat):
    features=self.getfeatures(item)
    for f in features:
      self.incf(f,cat)
    self.incc(cat)
    self.con.commit()

  def fprob(self,f,cat):
    if self.catcount(cat)==0: return 0
    return self.fcount(f,cat)/self.catcount(cat)

  def setfilename(self,filename):
    self.filename=filename
    self.restoredata()

  def restoredata(self):
    try: f=file(self.filename,'rb')
    except: return
```

```
    self.fc=cPickle.load(f)
    self.cc=cPickle.load(f)
    f.close()

  def savedata(self):
    f=file(self.filename,'wb')
    cPickle.dump(self.fc,f,True)
    cPickle.dump(self.cc,f,True)
    f.close()
  def weightedprob(self,f,cat,prf,weight=1.0,ap=0.5):
    basicprob=prf(f,cat)
    totals=sum([self.fcount(f,c) for c in self.categories()])
    bp=((weight*ap)+(totals*basicprob))/(weight+totals)
    return bp



class naivebayes(classifier):
  def __init__(self,getfeatures):
    classifier.__init__(self,getfeatures)
    self.thresholds={}

  def setthreshold(self,cat,t):
    self.thresholds[cat]=t

  def getthreshold(self,cat):
    if cat not in self.thresholds: return 1.0
    return self.thresholds[cat]

  def classify(self,item,default=None):
    probs={}
    max=0.0
    for cat in self.categories():
      probs[cat]=self.prob(item,cat)
      if probs[cat]>max:
        max=probs[cat]
        best=cat
    for cat in probs:
      if cat==best: continue
      if probs[cat]*self.getthreshold(best)>probs[best]: return default
    return best

  def docprob(self,item,cat):
    features=self.getfeatures(item)
    p=1
    for f in features: p*=self.weightedprob(f,cat,self.fprob)
    return p


  def prob(self,item,cat):
    catprob=self.catcount(cat)/self.totalcount()
    docprob=self.docprob(item,cat)
    return docprob*catprob
```

❖ La phase de Compréhension

```
#////////////////////////////////////////////////////////////
```

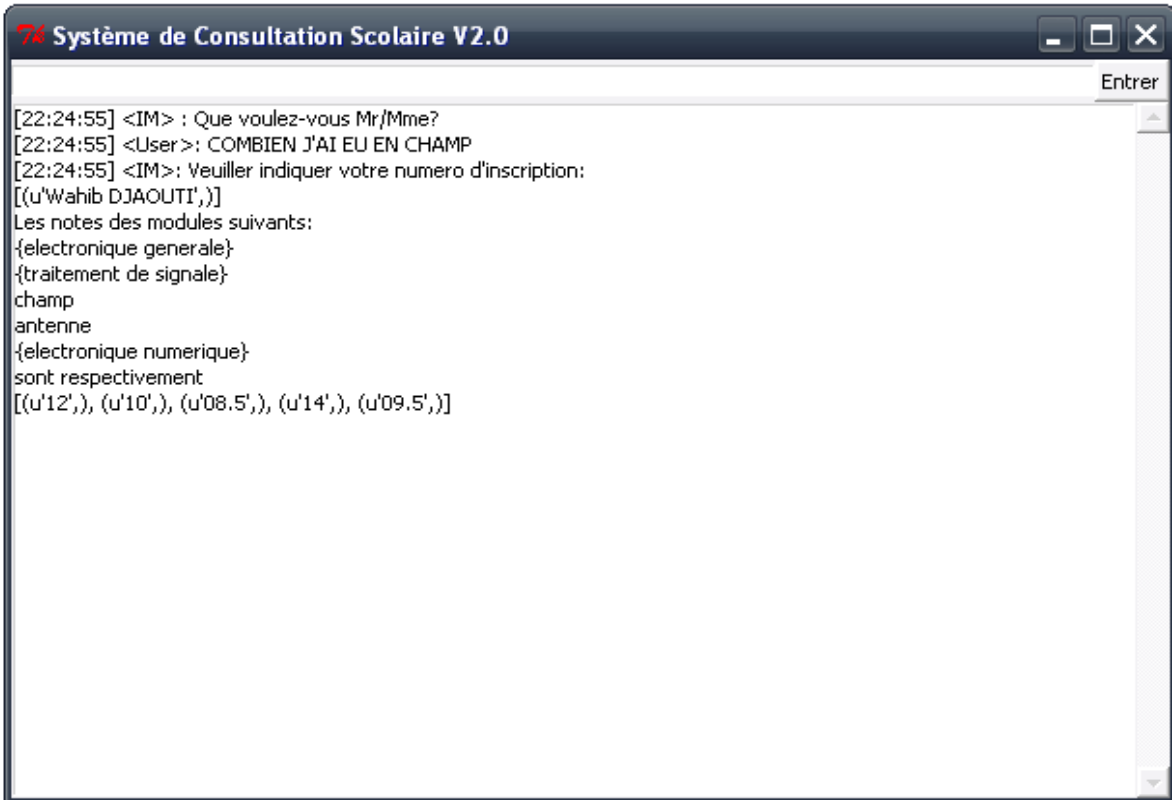
```
#////////////////////////////////////////////////////////////////
# connect to the database
con = sqlite.connect('sample.db')
cur = con.cursor()
ex = cur.execute

#////////////////////////////////
getwords("donne moi ma note d'examen?")
cl=naivebayes(getwords)# utiliser le classifieur bayésien
cl.setdb('test.db')
sampletrain(cl)
##cl.classify('pouver vous me montrer mon relevee de note.')
##cl.classify('donner moi ma certificat de scolarite.')
##cl.classify('je veux mon diplome.')
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Module_concept=['electronique generale','electronique numerique','traitement de
signale','electricite','mesure','champ','antenne','capteur','reseaux','fibre
optique','television','systeme de telecommunication']

def certificat_format(nom, dateNaissance, niveau):
  crtf=open('attestation.txt','w+')
  crtf.write('\t\t\t\t CERTIFICAT DE SCOLARITE \n\n\n\n')
  crtf.write("\t\t Je soussigné, Monsieur le directeur du departement
d'Electronique,atteste que:\n")
  crtf.write('\t\t\t\t\t\t\t\t Monsieur %s\n'%nom)
  crtf.write('\t\t\t\t\t\t\t\t Ne le %s\n'%dateNaissance)
  crtf.write("\t\t Etudiant en %s , pour l'annee universitaire
2010/2011"%niveau)
  crtf.write("\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t fait aujourd'hui")
  crtf.close()
  os.startfile('attestation.txt')

def diplome_format(nom, dateNaissance, niveau):
  dplm=open('diplome.txt','w+')
  dplm.write('\t\t\t\t Diplome de fin de formation \n\n\n\n')
  dplm.write("\t\t Je soussigné, Monsieur le directeur du departement
d'Electronique,atteste que:\n")
  dplm.write('\t\t\t\t\t\t\t\t Monsieur %s\n'%nom)
  dplm.write('\t\t\t\t\t\t\t\t Ne le %s\n'%dateNaissance)
  dplm.write("\t\t A complete avec succe le programme de formation en
%s"%niveau)
  dplm.write("\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t fait aujourd'hui")
  dplm.close()
  os.startfile('diplome.txt')

def take_action(doc):
##    e=re.search('code',doc)
##    f=re.search('inscription',doc)
##    g=re.search('numero',doc)
  if cl.classify(doc)=='diplome':
    #t=raw_input("Vous voulez votre diplome! ")
    vbs_Create('Vous voulez votre diplome')
    os.startfile("file2.vbs")
    if tkMessageBox.askyesno(END,"IM: Vous voulez votre diplome! "):
      vbs_Create("Veuiller indiquer votre numero d'inscription")
      os.startfile("file2.vbs")
      #s=raw_input("veuiller bien  indique votre numero d'inscription: ")
      listbox.insert(END,"[%s] <IM>: Veuiller indiquer votre numero
d'inscription: " % now)
      num = tkSimpleDialog.askinteger (title="Numéro d'inscription",
```

```
                                prompt="Entrer votre matricule")
      s=num
      #s=raw_input("veuiller bien  indique votre numero d'inscription: ")
      nom=ex('select nom from Etudiants where inscription="%s"' %(s)).fetchall()
      dateNaissance=ex('select date_de_naissance from Etudiants where
inscription="%s"' %(s)).fetchall()
      code =ex('select code_degre from Etudiants where
inscription="%s"'%(s)).fetchall()
      x=str(" ".join(["%s" % el for el in code]))
      niveau=ex('select designation from Degre where
code_degre="%s"'%(x)).fetchall()
      if x in ['001','002','003']:
        diplome_format(nom, dateNaissance, niveau)
        vbs_Create('Félicitation pour votre réussite! On vous souhaite de même
dans la vie privé.')
        os.startfile("file2.vbs")
      else:
        #print ("Vous n'avez pas ecore compléter votre cursus!")
        listbox.insert(END, "[%s] <IM>: Vous n'avez pas ecore compléter votre
cursus!" % now)
    else:
      #print ("Désolé, de ne pas pouvoir donner reponse à votre requete")
      listbox.insert(END," [%s] <IM>: Désolé, de ne pas pouvoir donner reponse à
votre requete" % now)
      listbox.insert(END, "[%s] <IM> : Que voulez-vous Mr/Mme?" % now)

  elif cl.classify(doc)=='certificat':
    #t=raw_input("Vous voulez votre certificat de scolarité! ")
    vbs_Create("Vous voulez votre certificat de scolarité")
    os.startfile("file2.vbs")
    if tkMessageBox.askyesno(END,"IM: Vous voulez votre certificat de scolarité!
"):
      vbs_Create("Veuiller indiquer votre numero d'inscription")
      os.startfile("file2.vbs")
      #s=raw_input("veuiller bien  indique votre numero d'inscription: ")
      listbox.insert(END,"[%s] <IM>: Veuiller indiquer votre numero
d'inscription: " % now)
      num = tkSimpleDialog.askinteger (title="Numéro d'inscription",
                                prompt="Entrer votre matricule")
      s=num

      #s=raw_input("veuiller bien  indique votre numero d'inscription: ")
      nom=ex('select nom from Etudiants where inscription="%s"' %(s)).fetchall()
      dateNaissance=ex('select date_de_naissance from Etudiants where
inscription="%s"' %(s)).fetchall()
      code =ex('select code_degre from Etudiants where
inscription="%s"'%(s)).fetchall()
      x=str(" ".join(["%s" % el for el in code]))
      niveau=ex('select designation from Degre where
code_degre="%s"'%(x)).fetchall()
      if x in ['001','002','003']:
        #print ("Vous n'avez pas le droit d'aquérir une autre Certificat de
Scolarité!")
        listbox.insert(END, "[%s] <IM>: Vous n'avez pas le droit d'aquérir une
autre Certificat de Scolarité!" % now)
      else:
        certificat_format(nom, dateNaissance, niveau)
        vbs_Create('On vous souhaite une bonne année scolaire!')
        os.startfile("file2.vbs")
    else:
```

```
      listbox.insert(END," [%s] <IM>: Désolé, de ne pas pouvoir donner reponse à
votre requete" % now)
      listbox.insert(END, "[%s] <IM> : Que voulez-vous Mr/Mme?" % now)

  elif cl.classify(doc)=='note':
    #t=raw_input("Vous voulez votre note! ")
    vbs_Create("Vous voulez votre Note")
    os.startfile("file2.vbs")
    if tkMessageBox.askyesno(END,"IM: Vous voulez votre Note! "):
      #s=raw_input("veuiller bien  indique votre numero d'inscription: ")
      vbs_Create("Veuiller indiquer votre numero d'inscription")
      os.startfile("file2.vbs")
      listbox.insert(END,"[%s] <IM>: Veuiller indiquer votre numero
d'inscription: " % now)
      num = tkSimpleDialog.askinteger (title="Numéro d'inscription",
                                       prompt="Entrer votre matricule")
      z=num
##      for w in Module_concept:
##        r=re.search(w,doc)
      w="\n".join( [str(x) for x in set(doc.split()).intersection(
set(Module_concept) ) ])
      if w is not '':
          #print ex('select nom from Etudiants where inscription="%s"'
%(s)).fetchall()
          listbox.insert(END,ex('select nom from Etudiants where
inscription="%s"' %(z)).fetchall())
          t=ex('select code_module from Module where
designation="%s"'%(w)).fetchall()
          # Pour convertir la liste en caractére
          x=str(" ".join(["%s" % el for el in t]))
          #print ('La note du module de #%s#'%w),'est egale à ',ex('select note
from Etudier where inscription="%s" and code_module="%s"' %(s,x)).fetchall()
          #listbox.insert(END,('IM: La note du module de #%s#'%w),'est egale à
',ex('select note from Etudier where inscription="%s" and code_module="%s"'
%(z,x)).fetchall())
          f=ex('select note from Etudier where inscription="%s" and
code_module="%s"' %(z,x)).fetchall()
          listbox.insert(END,('IM: La note du module de #%s#'%w),('est egale à:
%s' %(f)))
          if (f > 10):
               vbs_Create("Bravo! Bon travail.")
               os.startfile("file2.vbs")
          elif (f>5):
               vbs_Create("Pas mal tous de mème. Veuillez faire mieux.")
               os.startfile("file2.vbs")
          else:
               vbs_Create("Malheureusement! On vous souhaite bon chance pour la
suite.")
               os.startfile("file2.vbs")


      else:
          #print ex('select nom from Etudiants where inscription="%s"'
%(s)).fetchall()
          listbox.insert(END,ex('select nom from Etudiants where
inscription="%s"' %(z)).fetchall())
          #print ex('select note from Etudier where inscription="%s"'
%(s)).fetchall()
          listbox.insert(END,'Les notes des modules suivants: ')
          d=[]
```

```
            for column in ex('select code_module from Etudier where
inscription="%s"'%(z)):
                d.append(column)
            for a in d:
                listbox.insert(END,ex("select designation from Module where
code_module='%s'"%(a)).fetchone()),
            listbox.insert(END,'sont respectivement ')
            listbox.insert(END,ex('select note from Etudier where
inscription="%s"' %(z)).fetchall())
        else:
          listbox.insert(END," [%s] <IM>: Désolé, de ne pas pouvoir donner reponse à
votre requete" % now)
          listbox.insert(END, "[%s] <IM> : Que voulez-vous Mr/Mme?" % now)

    else:
        listbox.insert(END," [%s] <IM>: Désolé, de ne pas pouvoir donner reponse à
votre requete" % now)
        listbox.insert(END, "[%s] <IM> : Que voulez-vous Mr/Mme?" % now)
```

E.2.2 Exemple de traitement des requêtes par l'application de scolarité



E.2.3 Application de Diagnostique Médicale à Distance

❖ **Classificateur Bayesienne**

Du fait, qu'on vise la réalisation d'un Système de Compréhension Automatique de l'Enonce, on essayera de garder les mêmes algorithmes ainsi que le code sauf pour des cas précis.
Pour cette raison, on va garder le même classificateur Bayesienne.

❖ La phase de Compréhension

Les seules modifications qu'on va appliquer ont des liens avec le corpus d'apprentissage ainsi que le module de Compréhension qui est basé sur un moteur d'inférence.

Corpus d'apprentissage :

```
def sampletrain(cl):
  cl.train("j'ai eu des difficultés respiratoire, à parler ainsi qu'à avaler
",'allergie')
  cl.train("j'ai une légère fiévre, des maux de téte, des maux de gorge et une
voix rauque ",'bronchite')
  cl.train("j'ai de la fièvre, des vomissements et des douleurs
abdominales",'diarrhee')
  cl.train("j'ai des selles fréquentes et liquides avec perte importante de
liquide ",'diarrhee')
  cl.train("j'ai des diminution de l'énérgie en général ",'fatigue')
  cl.train("je me sent souvent fatigué même en ayant bien dormi et on a de la
peine à se lever le matin ",'fatigue')
  cl.train("j'ai une fièvre èlevè, des maux de tête importants, une fatigue
extrême et des douleurs dans les articulations",'grippe')
  cl.train("j'ai de forts maux de tête",'grippe')
  cl.train("j'ai des troubles digestifs, des ulcères, des nausées et
vomissements ",'stress')
  cl.train("je suis fatigue,pâleur de la peau et des muqueuses, des palpitations
ainsi que des essoufflement",'anemie')
  cl.train("j'ai des Écoulement nasal, éternuement, larmoiement des
yeux",'grippe')
  cl.train("j'ai divers troubles du sommeil ",'stress')
  cl.train("j'ai des difficulté à se concentrer",'fatigue')
  cl.train("j'ai de la fièvre à un niveau élevé, et également avoir des frissons
à cause de cette fièvre",'grippe')
  cl.train("j'ai des troubles psychiques : nervosité, colère excessive et
inhabituelle, crise d'angoisse, peurs excessives (phobies),
dépression",'stress')
  cl.train("j'ai  des maux de tête, des vertiges et des bourdonnements
d'oreille",'anemie')
  cl.train("j'ai des problèmes de concentration",'stress')
  cl.train("j'ai des douleurs a l'estomac , vomissement et de la
diarree",'allergie')
  cl.train("je me sens faible , des acceleartions du rythme cardiaque , de
l'anxiete , detresse et perte de conscience",'allergie')
  cl.train("je me sens de la fievre , des maux de tete , des maux de gorge , une
voix rauque ainsi des complications respiratoires",'bronchite')
  cl.train("j'ai de la fievre , des vomissements et des douleurs
abdominales",'diarrhee')
  cl.train("j'ai une toux seche , des maux de gorge , un rhume ainsi que des
nausees",'grippe')
  cl.train("j'ai des douleurs au niveau des articulations",'grippe')
  cl.train("j'ai des courbatures",'grippe')
  cl.train("j'ai une perte d'appetit , fievre , maux de tete et des irritation
de la gorge",'grippe')
  cl.train("je m'enerve rapidement",'stress')
  cl.train("j'ai des problemes cardiaques ( acceleration du rythme cardiaque
)",'stress')
  cl.train("j'ai un probleme de chute des cheveux",'stress')
```

Moteur D'inférence :

```
#////////////////////////////////////////////////////////////
#////////////////////////////////////////////////////////////
#//////////////////////////////////
getwords("J'ai mal à la tête, je crois que j'ai de la fièvre ")
cl=naivebayes(getwords)# utiliser le classifieur bayésien
cl.setdb('test.db')
sampletrain(cl)
##cl.classify('pouver vous me montrer mon relevee de note.')
##cl.classify('donner moi ma certificat de scolarite.')
##cl.classify('je veux mon diplome.')
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
#-----------------------------
# Finding todays date

today = datetime.date.today()
#-------------------------------
#http://pleac.sourceforge.net/pleac_python/datesandtimes.html
#////////////////////////////////////////////////////////////

def Ordon_Med(NOM, PRENOM, AGE,DIAGNOSTIQUE):
  crtf=open('Certificat-Medicale.txt','w+')
  crtf.write('\t\t\t\t CABINET DE DIAGNOSTIQUE MEDICALE A DISTANCE  \n')
  crtf.write("\t\t\t\t\t\t\t Docteur xx yy \n")
  crtf.write("\t\t\t\t\t\t\t\t MEDECINE GENERALE \n")
  crtf.write("\t\t\t\t\t\t\t\t\t ---oOo--- \n")
  crtf.write("\t\t\t\t\t\t\t\t\t TEL:0xx.xx.xx.xx\n")
  crtf.write("\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t Alger le %s \n"%today)
  crtf.write("-------------------------------------------------------\n\n")
  crtf.write("NOM: %s                    PRENOM:%s                    AGE: %s
\n\n\n"%(NOM,PRENOM,AGE))
  crtf.write("\t\t\t\t\t CERTIFICAT MEDICALE \n\n\n")
  crtf.write("\t\t %s \n\n\n"%DIAGNOSTIQUE)
  crtf.close()
  os.startfile('Certificat-Medicale.txt')

def take_action(doc):

  #NOM=raw_input('Veuillez indiquer votre nom: ')
  #PRENOM=raw_input(u'Veuillez indiquer votre prénom: ')
  #AGE=raw_input(u'Veuillez préciser votre age: ')
  listbox.insert(END,"[%s] <IM> : 'Veuillez indiquer votre nom: " % now)
  vbs_Create('Veuillez indiquer votre nom')
  os.startfile("file2.vbs")
  NOM = tkSimpleDialog.askstring (title="Nom du patient",
                                  prompt="Veuillez indiquer votre nom")
  listbox.insert(END,"[%s] <IM> : 'Veuillez indiquer votre prénom: " % now)
  vbs_Create('Veuillez indiquer votre prénom ')
  os.startfile("file2.vbs")
  PRENOM = tkSimpleDialog.askstring(title="Prénom du patient",
                                  prompt="Veuillez indiquer votre prénom")
  listbox.insert(END,"[%s] <IM> : 'Veuillez indiquer votre age: " % now)
  vbs_Create('Veuillez indiquer votre age')
  os.startfile("file2.vbs")
  AGE = tkSimpleDialog.askinteger (title="Age du patient",
                                  prompt="Veuillez indiquer votre age")

  if cl.classify(doc)=='allergie':
```

```
    x='''\t\t\tLa cause de votre maladie est forte probablement une allergie...
                    Pour vous-vous soulager en cas d'une allergie légère, un
lavage nasal contribuera à évacuer
                    le mucus de votre nez. Achetez une solution saline à la
pharmacie ou préparez-la vous-même
                    (1 cuiller à thé de sel pour un demi-litre d'eau tiède).
Penchez-vous au-dessus de l'évier,
                    versez un peu de solution dans votre main et aspirez-la dans
une narine, puis laissez-la sortir
                    et mouchez-vous délicatement. Faites la même chose avec
l'autre narine.

                    En cas de nom amélioration de votre état aprés une journé,
une consultation médicale chez un
                    médecin générale est essentiel..,
                    Malgrés que ceci est recommendé dés l'apparition des
sympotomes de maladies.'''

    Diagnostique=str(x)


  elif cl.classify(doc)=='bronchite':

    x='''\t\t\tLa cause de votre maladie est forte probablement une bronchite
aigue...
                     Des remèdes naturels sont conseillée:
                     1- Inhalation de vapeur:
                     Prenez une douche chaude ou encore, versez de l'eau très
chaude dans un bol, penchez-vous au-dessus
                     et couvrez votre tête d'une serviette. Cette technique
contribuera à relâcher les sécrétions de vos poumons.
                     2- Méthodes naturelles pour éclaircir le mucus:
                     Pour éclaircir votre mucus et vous permettre de l'évacuer
plus facilement par la toux, buvez au moins huit
                     verres de 250 ml d'eau par jour.
                     Consommez des piments forts, de la salsa épicée ou des plats
préparés avec du piment de Cayenne. En plus de
                     faire couler le nez, les aliments piquants éclaircissent le
mucus des poumons, permettant de mieux l'évacuer
                     par la toux.
                     Prenez de la tisane de molène. Utilisé pour soigner les
affections respiratoires, ce remède populaire agit
                     aussi sur le mucus.
                     Évitez les produits laitiers. Le lait de vache renferme de
la lactalbumine, substance qui stimule la production
                     de mucus dans les voies respiratoires supérieures et
inférieures ainsi que dans les intestins.

                     En cas de nom amélioration de votre état aprés une journé,
une consultation médicale chez un
                     médecin générale est essentiel..,
                     Malgrés que ceci est recommendé dés l'apparition des
sympotomes de maladies.'''

    Diagnostique=str(x)


  elif cl.classify(doc)=='diarrhee':

     x='''\t\t\tLa cause de votre maladie est forte probablement la diarrhee...
```

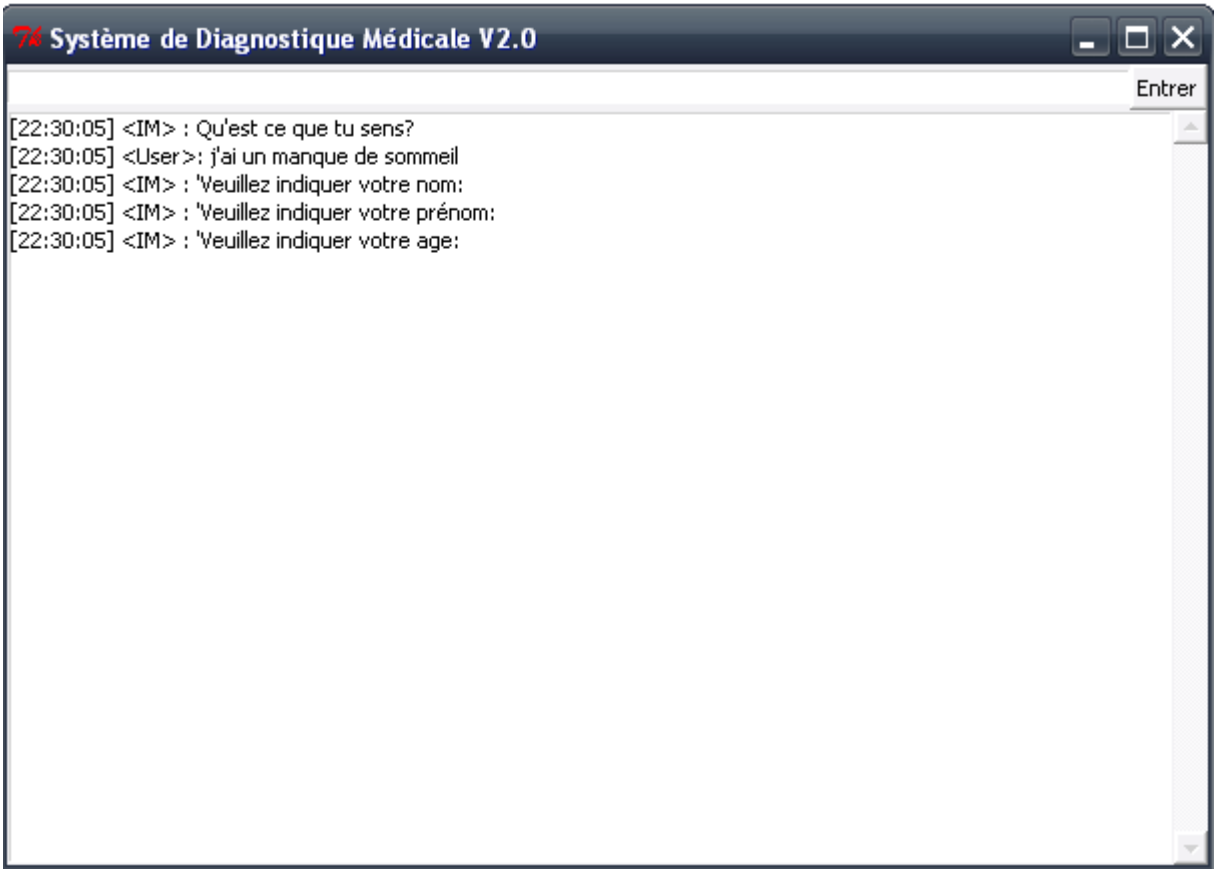
```
                    Traitement de la diarrhée:
                    1- Le plus important, quand on a la diarrhée, c'est de
remplacer les liquides qu'on a perdus.
                    Buvez beaucoup d'eau, de jus de pomme, de colas, de
boissons riches en électrolytes, tel que le Gatorade,
                    ou prenez du bouillon de poulet ou de boeuf. Évitez, par
contre, le lait et les jus de fruits acides,
                    de même que les aliments que vous digérez mal. Lorsque
votre diarrhée se sera atténuée, vous pourrez vous
                    tourner vers le régime BRATT (banane, riz, compote de
pommes, thé et pain grillé) afin de donner du volume
                    à vos selles.
                    2- Médicaments contre la diarrhée:
                    Vous pouvez aussi prendre l'un de ces antidiarrhéiques
populaires, offerts en vente libre:
                    Lopéramide (Imodium) : ralentit le transit des selles dans
les intestins.
                    Attapulgite (Kaopectate) :  absorbe les irritants qui
causent la diarrhée dans le tractus digestif.
                    Salicylate de bismuth (Pepto-Bismol) : agit en tapissant
l'intérieur de l'intestin d'une pellicule,
                    le protégeant ainsi des irritants et diminuant
l'accumulation de liquides qui peuvent entraîner un
                    épisode de diarrhée.
                    3- La diarrhée dure habituellement un jour ou deux.
Consultez un médecin si vous faites 38,8° C ou plus de
                    fièvre, si vos selles sont inhabituellement foncées ou
saignantes,  si votre diarrhée dure plus de quatre jours,
                    si vous avez des étourdissements, manquez d'énergie, avez
mal au ventre ou au rectum, ou avez des symptômes de
                    déshydratation (forte soif, sécheresse de la bouche ou de
la peau, ou fréquent besoin d'uriner).

                    En cas de nom amélioration de votre état aprés une journé,
une consultation médicale chez un
                    médecin générale est essentiel..,
                    Malgrés que ceci est recommendé dés l'apparition des
sympotomes de maladies.'''

     Diagnostique=str(x)


  elif cl.classify(doc)=='fatigue':

    x='''\t\t\tLa cause de votre maladie est forte probablement une fatigue...
                    Traitement de la fatigue:
                    Diminuez votre consommation de caféine. Cette substance crée
une dépendance qui oblige à en consommer de plus
                    en plus pour obtenir un effet stimulant.
                    Renoncez à la nicotine. Tout comme la caféine, c'est un
stimulant. On pourrait donc penser qu'elle donne de
                    l'énergie, mais c'est plutôt le contraire. Quand on fume, le
taux d'oxygène dans le sang diminue; comme les
                    muscles et les tissus ont besoin d'oxygène pour produire de
l'énergie, il en résulte de la fatigue.
                    Gardez la forme. L'exercice a pour effet de libérer dans le
cerveau des endorphines, neurotransmetteurs
                    énergisants, d'augmenter l'apport au cerveau et aux muscles
de sang enrichi en oxygène, et de multiplier le
                    nombre de cellules sanguines du corps. De plus, il favorise
le sommeil. Enfin, quand on est en forme,
```

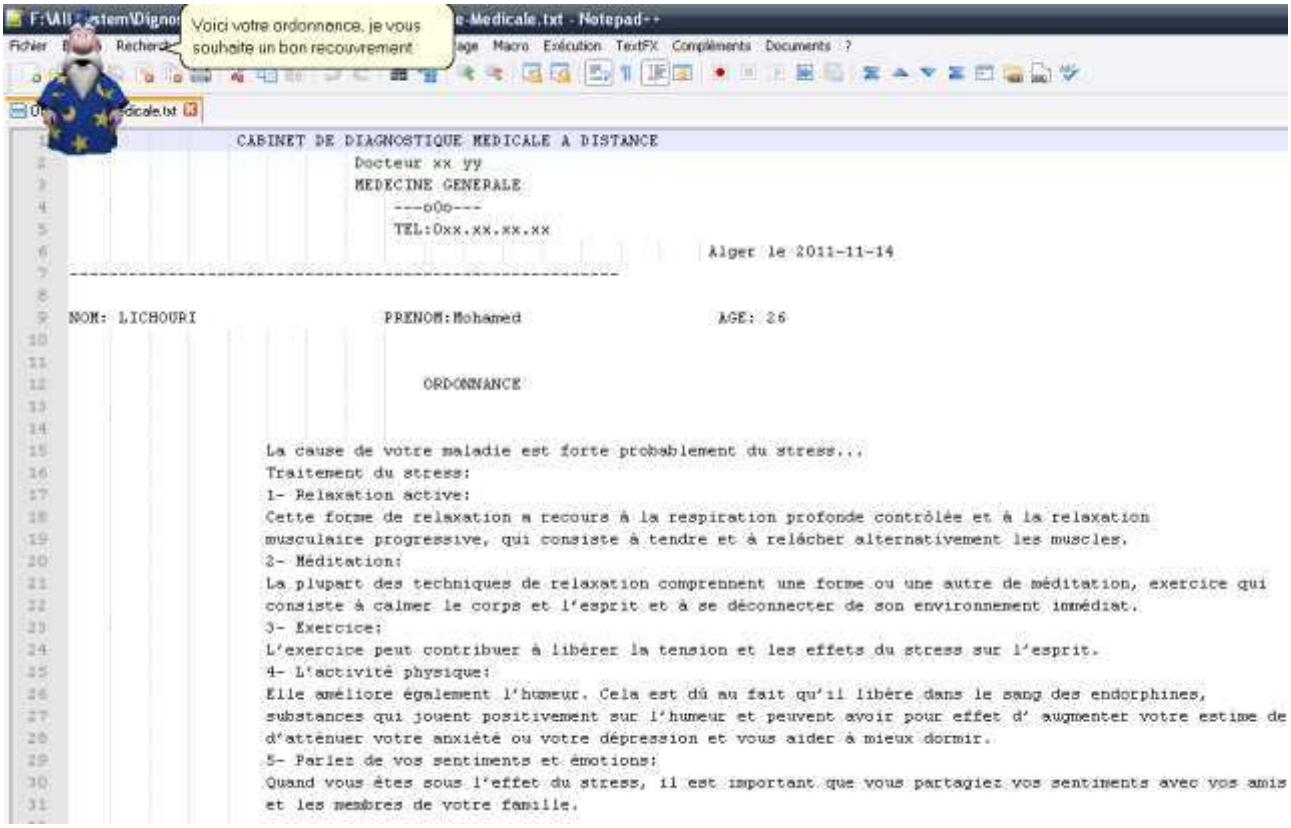
```
                    les activités quotidiennes - par exemple ces lourds sacs
d'épicerie qu'il faut porter de la voiture à
                    la maison - sont moins fatigantes.
                    Dormez suffisamment. Sur semaine, seulement 35% des gens
dorment les huit heures prescrites. Votre sommeil
                    devrait être pour vous une priorité. Gardez à l'esprit qu'on
ne peut récupérer durant la fin de semaine le
                    sommeil qu'on a perdu durant la semaine.
                    Si vous éprouvez souvent de la fatigue sans vous en
expliquer les causes, consultez votre médecin; il pourra
                    déterminer s'il y a une cause médicale sous-jacente et, le
cas échéant, vous proposer un traitement approprié.
                    Si, par contre, elle ne résulte pas d'une maladie, il vous
faudra apporter des changements à votre mode de vie,
                    par exemple en modifiant votre alimentation et en faisant de
l'exercice.

                    En cas de nom amélioration de votre état aprés une journé,
une consultation médicale chez un
                    médecin générale est essentiel..,
                    Malgrés que ceci est recommendé dés l'apparition des
sympotomes de maladies.'''

    Diagnostique=str(x)


  elif cl.classify(doc)=='grippe':

    x='''\t\t\tLa cause de votre maladie est forte probablement une grippe...
                    4 aliments pour combattre le rhume et la grippe:
                    1- Eau, thé décaféiné et jus:
                    Les virus de la grippe et du rhume se multiplient quand la
gorge et les voies nasales sont sèches.
                    La consommation de grandes quantités de liquides tout au
long de la journée a pour effet d'hydrater
                    les muqueuses, qui seront alors plus aptes à piéger les
virus.
                    Ordonnance: au moins huit verres de liquides tous les jours,
plus si vous faites de la fièvre.
                    2- Soupe au poulet:
                    La soupe chaude élève la température du nez et de la gorge,
créant un milieu inhospitaliser pour les virus,
                    qui préfèrent le froid et la sécheresse. La soupe fumante
éclaircit le mucus, qui s'évacue alors plus facilement.
                    Ordonnance: prenez un bol fumant dès que les symptômes
apparaissent.
                    3- Ail:
                    L'ail renferme de l'allicine, puissant antimicrobien qui
combat les bactéries, les virus et les champignons.
                    Ordonnance: une gousse toutes les 3 ou 4 heures. Vous pouvez
également couper la gousse en morceaux que vous
                    avalerez comme si c'était des comprimés. Ou ajoutez-les avec
des oignons à la soupe au poulet. Pour favoriser
                    la formation de ses composés thérapeutiques, hachez-le et
laissez-le reposer 10 à 15 minutes avant de l'utiliser.
                    4- Épices et condiments épicés:
                    Le piment de Cayenne, le raifort ou le wasabi, ajoutés aux
plats, pourraient contribuer à désobstruer le nez.
                    Ils ont pour effet de contracter les vaisseaux sanguins du
nez et de la gorge, soulageant temporairement la congestion.
```

```
                    Ordonnance: autant d’épices que vous êtes capable de
tolérer.

                    En cas de nom amélioration de votre état aprés une journé,
une consultation médicale chez un
                    médecin générale est essentiel..,
                    Malgrés que ceci est recommendé dés l'apparition des
sympotomes de maladies.'''

    Diagnostique=str(x)


  elif cl.classify(doc)=='stress':

    x='''\t\t\tLa cause de votre maladie est forte probablement du stress...
                    Traitement du stress:
                    1- Relaxation active:
                    Cette forme de relaxation a recours à la respiration
profonde contrôlée et à la relaxation
                    musculaire progressive, qui consiste à tendre et à relâcher
alternativement les muscles.
                    2- Méditation:
                    La plupart des techniques de relaxation comprennent une
forme ou une autre de méditation, exercice qui
                    consiste à calmer le corps et l’esprit et à se déconnecter
de son environnement immédiat.
                    3- Exercice:
                    L’exercice peut contribuer à libérer la tension et les
effets du stress sur l’esprit.
                    4- L’activité physique:
                    Elle améliore également l’humeur. Cela est dû au fait qu’il
libère dans le sang des endorphines,
                    substances qui jouent positivement sur l’humeur et peuvent
avoir pour effet d’ augmenter votre estime de vous-même,
                    d’atténuer votre anxiété ou votre dépression et vous aider à
mieux dormir.
                    5- Parlez de vos sentiments et émotions:
                    Quand vous êtes sous l’effet du stress, il est important que
vous partagiez vos sentiments avec vos amis
                    et les membres de votre famille.

                    En cas de nom amélioration de votre état aprés une journé,
une consultation médicale chez un
                    médecin générale est essentiel..,
                    Malgrés que ceci est recommendé dés l'apparition des
sympotomes de maladies.'''

    Diagnostique=str(x)


  elif cl.classify(doc)=='anemie':

    x='''\t\t\tLa cause de votre maladie est forte probablement une anémie...
                    Traitement de l’anémie ferriprive:
                    Consultez un médecin. Il pourrait :
                    Vous faire passer des analyses pour mesurer la teneur en fer
de votre sang et sa richesse en globules rouges;
                    Rechercher la cause sous-jacente, par exemple en ayant
recours à l’endoscopie ou à la radiographie au baryum pour
                    dépister d’éventuels troubles du tube digestif;
```

```
                    Si le diagnostic est confirmé, prescrire des comprimés de
fer ou des médicaments pour réguler les pertes
                    menstruelles ou encore administrer du fer sous forme
d'injections.

                    En cas de nom amélioration de votre état aprés une journé,
une consultation médicale chez un
                    médecin générale est essentiel..,
                    Malgrés que ceci est recommendé dés l'apparition des
sympotomes de maladies.'''

    Diagnostique=str(x)

  else:
    #print '''Désole de ne pas donner un diagnostique à votre cas,
          #   Prévoyer une consultation chez un médecin générale le plus
rapidement possible est fortement recommendé'''
    listbox.insert(END,"[%s] <IM> : Désole de ne pas donner un diagnostique à
votre cas, Prévoyer une consultation chez un médecin générale le plus rapidement
possible est fortement recommendé \n" % now)
    Diagnostique='''Désole de ne pas donner un diagnostique à votre cas,
                Prévoyer une consultation chez un médecin générale le plus
rapidement possible est fortement recommendé'''

  Ordon_Med(NOM, PRENOM, AGE,Diagnostique)
  vbs_Create('Voici votre ordonnance, je vous souhaite un bon recouvrement.')
  os.startfile("file2.vbs")
```

E.2.4 Exemple de traitement des requêtes par l'application de scolarité


Système de Diagnostique Médicale V2.0
Entrer
[22:30:05] <IM> : Qu'est ce que tu sens?
[22:30:05] <User>: j'ai un manque de sommeil
[22:30:05] <IM> : 'Veuillez indiquer votre nom:
[22:30:05] <IM> : 'Veuillez indiquer votre prénom:
[22:30:05] <IM> : 'Veuillez indiquer votre age:


Voici votre ordonnance, je vous souhaite un bon recouvrement
CABINET DE DIAGNOSTIQUE MEDICALE A DISTANCE
Docteur xx yy
MEDECINE GENERALE
---oOo---
TEL:0xx.xx.xx.xx
Alger le 2011-11-14
NOM: LICHOURI
PRENOM:Mohamed
AGE: 26
ORDONNANCE
La cause de votre maladie est forte probablement du stress...
Traitement du stress:
1- Relaxation active:
Cette forme de relaxation a recours à la respiration profonde contrôlée et à la relaxation musculaire progressive, qui consiste à tendre et à relâcher alternativement les muscles.
2- Méditation:
La plupart des techniques de relaxation comprennent une forme ou une autre de méditation, exercice qui consiste à calmer le corps et l'esprit et à se déconnecter de son environnement immédiat.
3- Exercice:
L'exercice peut contribuer à libérer la tension et les effets du stress sur l'esprit.
4- L'activité physique:
Elle améliore également l'humeur. Cela est dû au fait qu'il libère dans le sang des endorphines, substances qui jouent positivement sur l'humeur et peuvent avoir pour effet d' augmenter votre estime de d'atténuer votre anxiété ou votre dépression et vous aider à mieux dormir.
5- Parlez de vos sentiments et émotions:
Quand vous êtes sous l'effet du stress, il est important que vous partagiez vos sentiments avec vos amis et les membres de votre famille.

Bibliographie

Bibliographie

[1] Michael F McTear (2004) Spoken dialogue technology: toward the conversational user interface. Springer Verlag: London.

[2] Michael Baryla; The Science Behind an Answer ; http://www-03.ibm.com/ innov - ation/us/watson/ ;mise en ligne le 29/06/2010. Consulté le 14/12/2011

[3] PIERACCINI R., LEVIN E. & VIDAL E. (1993). Learning how to understand language. In Proceedings of the 4rd European Conference on Speech Communication and Technology, Berlin.

[4] Caroline Bousquet-Vernhettes, Nadine Vigouroux et Guy Pérennou, « Stochastic Conceptual Model for Spoken Language Understanding », Internationnal Workshop on Speech and Computer (SPECOM'99), Moscou, Russie, p. 71-74, 1999.

[5] MAYNARD H. & LEFÉVRE F. (2002). Apprentissage d'un module stochastique de compréhension de la parole. In 24èmes Journées d'étude sur la parole, Nancy.

[6] Salma Jamoussi, Kamel Smaïli, Jean Paul Haton: Understanding Speech Based on a Bayesian Concept Extraction Method. TSD 2003: 181-188

[7] Kiss, Tibor and Strunk, Jan (2006): Unsupervised Multilingual Sentence Boundary Detection. Computational Linguistics 32: 485-525.

[8] Stephanie Seneff, « Robust Parsing for Spoken Language Systems », International Conference on Acoustics, Speech and Signal Processing (ICASSP'92), San Francisco, États-Unis, Vol. 1, p. 189-192, 1992.

[9] Alex Carobus et Jyoti Paul, « A Distributed Two-tier Architecture for Multiple Domain Language Understanding: Combining Call-Routing and Domain Specific Understanding to Increase Performance, Location Independence and Scalability », 2000. http://nlp.stanford.edu/courses/cs224n/2000/jpaul2/paperoutline.html

[10] P. Baggia, A. Kellner, G. Pérennou, C. Popovici, J. Sturm et F. Wessel, « Language Modelling and Spoken Dialogue Systems - the ARISE Experience », European conference on speech communication and technology (EUROSPEECH'99), Budapest, Hongrie, Vol. 4, p. 1767-1770, 1999.

[11] Lokbani M. N. & White S, « La reconnaissance de la parole », La Recherche, n° 319, 1999, p. 82.

[12] Victor Zue, « Conversational interfaces: advances and challenges », European conference on speech communication and technology (EUROSPEECH'97), Rhodes, Grèce, p. 9-18.

[13] S. Bennacef, H. Bonneau-Maynard, J.L. Gauvain, L. Lamel, and W. Minker, « A Spoken Language System For Information RetrievalProc», ICSLP-94, September, 1994.

[14] Jean Caelen, Ho. Nguyen : « Gestion de buts de dialogue », TALN 2004, Fès, 19-21 avril 2004

[15] Frank Panaget, « D'un système générique de génération d' énoncés en contexte de dialogue oral à la formalisation logique des capacités linguistiques d' un agent rationnel dialoguant », Thèse de doctorat, Université de rennes I, 1996.

[16] C. Ponton, « Génération automatique de textes : 30 ans de réalisations », Génération Automatique de Textes (GAT' 97), Grenoble, France, 1997.

[17] « Centre icom' – Projet du Programme France de Handicap International », http://fr.creativecommons.org

[18] Roberto Pieraccini, Evelyne Tzoukermann, Zakkar Garelov, Jean-Luc Gauvain, Esther Levin, Chin-Hui Lee et Jay G. Wilpon, « A Speech Understanding System Based on Statistical Representation of Semantics », International Conference on Acoustics, Speech and Signal Processing (ICASSP'95), San Francisco, États-Unis, p. 193-196, 1992.

[19] Salma Jamoussi, « Méthodes statistiques pour la compréhension automatique de la parole », LORIA 2004.

[20] Robbe, S., Carbonell, N., Dauchy, P. (2000). Expression constraints in multimodal human-computer interaction. In H. Lieberman (Ed.), Proceedings of the ACM International Conference on Intelligent User Interfaces (IUI'2000), New Orleans, January 9-12, 2000, New York: ACM Press, pp. 225-229.

[21] Korzybski, 1951 « Une carte n'est pas le territoire: prolégomènes aux systèmes non-aristotéliciens et à la sémantique générale » 1951.

[22] Mandelbaum, David G, «Selected Writings of Edward Sapir. Selected Writings of Edward Sapir»,. Berkeley: University of California Press.

[23] Guy Pérennou, « Compréhension du dialogue oral – Le rôle du lexique dans l'approche par segments conceptuels », Lexique et communication parlée - GDR PRC, p. 169-178, 1996.

[24] Caroline Bousquet-Vernhettes, Nadine Vigouroux et Guy Pérennou, « Stochastic Conceptual Model for Spoken Language Understanding », Internationnal Workshop on Speech and Computer (SPECOM'99) , Moscou, Russie, p. 71-74, 1999.

[25] John F. Sowa, «Conceptuels structures : Information Processing in Mind and Machine » Addition-Wisley Publishing Company, 1984.

[26] Jérôme Goulian et Jean-Yves Antoine, « Compréhension automatique de la parole combinant syntaxe locale et sémantique globale pour une CHM portant sur des tâches relativement complexes », Traitement Automatique du langage Naturel (TALN'01), Tours, France, p. 203-212, 2001.

[27] Villaneau J., Antoine J.Y., Ridoux O. (2001) « Combining syntax and pragmatic knowledge for the understanding of spontaneous spoken utterance», Actes de LACL'01, pp. 279-295.

[28] Wolfgang Minker, « Compréhension automatique de la parole spontanée, L'Harmattan », 1999.

[29] Caroline Bousquet-Vernhettes, « Un environnement de compréhension de la parole pour les serveurs interactifs : l'environnement CACAO », Rencontres Jeunes Chercheurs en Interaction Homme Machine (RJC-IHM'2000), Île de Berder, France, p. 77-80, 2000.

[30] Jean-Yves Antoine, Caroline Bousquet-Vernhettes, Jérôme Goulian, Mohamed Zakaria Kurdi, Sophie Rosset, Nadine Vigouroux et Jeanne Villaneau, « Predictive and objective evaluation of speech understanding: the 'challenge' evaluation campaign of the I3 workgroup of the French CNRS », Third International Conference on Language Resources and Evaluation (LREC'02), Las Palmas, Espagne, Vol. II, p. 529-536, 2002.

[31] Claire Blanche-Benveniste, Le français parlé – Études grammaticales, CNRS éditions, 1990.

[32] Jean-Léon Bouraoui, Caroline Bousquet-Vernhettes, « Modélisation linguistique pour la compréhension d'énoncés : comparaison avec une approche basée sur les segments conceptuels », Workshop couplage de l'écrit avec l'oral, TALN 2002, Nancy, France, ATALA/ATILF/LORIA, 24-27 juin 2002

[33] Renato De Mori, «Spoken Dialogues with computers », Academic Press, 1998.

[34] Caroline Bousquet-Vernhettes, « Compréhension robuste de la parole spontanée dans le dialogue oral homme-machine – Décodage conceptuel stochastique », thèse de doctorat 2002.

[35] Harald Aust, Martin Oerder, Frank Seide et Volker Steinbiss, « The Philips automatic train timetable information system », Speech Communication, Elsevier science B.V., Vol. 17, p. 249-262, 1995.

[36] J. Fereiros, J. Colas, J. Macias-Guarasa, A. Ruiz et J.M. Pardo, « Controlling a Hifi with a continuous speech understanding system », International Conference on Spoken Language Processing (ICSLP'98), Sydney, Australie, Vol. 7, p. 2871-2874, 1998.

[37] Manuela Boros et Paul Heisterkamp, « Linguistic phrase spotting in a simple application spoken dialogue system », European conference on speech communication and technology (EUROSPEECH'99), Budapest, Hongrie, Vol. 5, p. 1983-1986, 1999.

[38] Kadri Hacioglu et Wayne Ward, « A Word Graph Interface for a Flexible Concept Based Speech Understanding Framework », European conference on speech communication and technology (EUROSPEECH'01), Aalborg, Danemark, Vol. 3, p. 1775-1778, 2001.

[39] L. Mayfield, M. Gavalda, Y-H. Seo, B. Suhm, W. Ward et A. Waibel, « Parsing real input in JANUS: A concept-based approach to spoken language translation », Theoretical and Methodical Issues in Machine Translation Conference (TMI'95), Louvain, Belgique, p. 196-204, 1995.

[40] Ye-Yi Wang, « A robust parser for spoken language understanding », European conference on speech communication and technology (EUROSPEECH'99), Budapest, Hongrie, Vol.5, p. 2055-2058, 1999.

[41] E. Vidal, R. Pieraccini, et E. Levin, 1993. Learning associations between grammars : a new approach to natural language understanding. Dans les actes de Proceedings of the European Conference on Speech Communication and Technology (Eurospeech), Berlin, Germany.

[42] S. Young (2002). "The Statistical Approach to the Design of Spoken Dialogue Systems." Tech Report CUED/F-INFENG/TR.433, Cambridge University Engineering Department.

[43] Minker, Wolfgang (1997): "Stochastically-based natural language understanding across tasks and languages", In EUROSPEECH-1997, 1423-1426.

[44] Stephanie Seneff, « TINA: A Natural Language System for Spoken Language Applications », Computational Linguistic, p. 61-86, 1992.

[45] Stephanie Seneff, « TINA: A Probabilistic Syntactic Parser for Speech Understanding Systems », DARPA Speech and Natural Language Workshop, Philadelphie, États-Unis, p. 168-178, 1989.

[46] Ward, W. (1991). Understanding spontaneous speech: The Phoenix system. In Proceedings of the International Conference on Audio, Speech and Signal Processing (ICASSP), pages 365-367.

[47] Roberto Pieraccini et Esther Levin, « A Spontaneous-Speech Understanding System for Database Query Applications », ESCA Workshop on Spoken Dialogue Systems, Vigso, Danemark, p. 85-88,1995.

[48] Wolfgang Minker, « Stochastic versus Rule-based Speech Understanding for Information Retrieval », Speech Communication, Elsevier science B.V., Vol. 25, p. 223-247, 1998.

[49] Ch. J. Fillmore, « The case for case », Universals in Linguistic Theory, p. 1-90, 1968.

[50] R. Bruce, « Cases Systems for Natural Languages », Artificial Intelligence, Vol. 6, p. 327-360, 1975.

[51] Wolfgang Minker et Samir Bennacef, « Compréhension et évaluation dans le domaine ATIS », Journées d'Études Paroles (JEP'96), Avignon, France, p. 414- 420, 1996.

[52] Schwartz, R.; Miller, S.; Stallard, D.; and Makhoul, J.1996. Language understanding using hidden understanding models. In Proceedings of the International Conference on Spoken Language Processing, volume 2, 997–1000.

[53] Didierjean Geneviève ; Dupuis Claire ; Duval Raymond ; Egret Marie-Agnès ; Kremer Daniel ; Robert Gilles ; Wenner Brigitte ; Ziegler Michèle.Petit x. Num. 44. p. 35-48. « A propos de charades dont la solution est un système d'équations à 2 inconnues ». IREM de Grenoble, Grenoble, 1997

[54] Soderland, S. 2001, «Building a Machine Learning Based Text Understanding System». In Proceedings of IJCAI Workshop on Adaptive Text Extraction and Mining, 64-70. http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.20.8882

[55] Rapsodis, Arc Inria 2008-2009, http://rapsodis.loria.fr/ ; mise en ligne le 09/11/2011. Consulté le 17/12/2011

[56] Anne Piret, Jean Nizet, E. Bourgeois 1996, « L'analyse structurale:Une méthode d'analyse de contenu pour les sciences humaines »,

[57] Structural Analysis, http://www.ischool.utexas.edu /~palmquis/courses/structural.htm Mise en ligne le 31/08/1999. Consulté le 12/11/2011

[58] H. Muller, V. Amerl, and G. Natalis. Worterkennungsverfahren als Grundlage einer Universalmethode zur automatischen Segmentierung von Texten in Sätze. Ein Verfahren zur maschinellen Satzgrenzenbestimmung im Englischen. Sprache und Datenverarbeitung, 1. 1980.

[59] Riley M. (1989). Some Applications of Tree-based Modeling, to Speech and Language Indexing.In Proc. of the DARPA Speech and Natural Language Workshop, pages: 339-352.

[60] Jeffrey C. Reynar, Adwait Ratnaparkhi — 1997 ; A Maximum Entropy Approach to Identifying Sentence Boundaries. In Proceedings of the Fifth Conference on Applied Natural Language Processing.

[61] Palmer D., Hearst M (1997). Adaptive Multilingual Sentence Boundary Disambiguation. Computational Linguistics,pages: 241-267(2).

[62] Santos D.(2001).Punctuation and Multilinguality : some reflections from a language Engineering perspective. www.oslo.sintef.no/portug/Diana/download/ponctuacao.ps

[63] Conference on Computational Natural Language Learning (CoNLL-2000). Chunking ; http://www.clips.ua.ac.be/conll2000/chunking/; mise en ligne 19/04/2011. Consulté le 02/10/2011.

[64] Lance A. Ramshaw, Mitchell P. Marcus, «Text Chunking Using Transformation-based Learning In: Proceedings of the Third ACL Workshop on Very Large Corpora» (WVLC 1995). http://www.aclweb.org/anthology-new/W/W95/W95-0107.pdf

[65] Firth, J. 1957/1968. «A synopsis of linguistic theory, 1930-55». «Studies in linguistic analysis (Special Volume of the Philological Society), 1957». « Reprinted in F. Palmer. Selected Papers of J. R. Firth 1952-59». « London and Harlow: Longmans, Green and Co., Ltd., 168-205».

[66] Mikheev, Andrei «Periods, capitalized words, etc», Computational Linguistics , 28(3):289–318.

[67] Bernhard Huber. Stoplist; http://textalyser.net /stoplist.html :

[68] Deerwester, S. Dumais, S.T, Furnas, G.W, Landauer, T.K, Harshman,R, «Indexing by Latent Semantic Analysis», Journal of the American Society For Information Science, 41(6), pp.391-407, 1990

[69] Virginie Zampa, « Utilisation de l'analyse sémantique latente pour tenter d'optimiser l'acquisition par exposition à une langue étrangère de spécialité », *Alsic*, Vol. 8, n° 2 | 2005, [En ligne], mis en ligne le 15 septembre 2005. URL : http://alsic.revues.org/index339.html ; Consulté le 01 mars 2012.

[70] Egwald Mathematics «Linear Algebra by Elmer G. Wiens» http://www.egwald.ca/linearalgebra/index.php

[71] Honkela T, Kaski S, Lagus K, and Kohonen T, «Websom - self-organizing maps of document collections.In Proc». of WSOM.

[72] S. Lawrence, C.L. Giles, S. Fong, «Natural Language Grammatical Inference with Recurrent Neural Networks», IEEE Trans. on Knowledge and Data Engineering, 12(1), 126-140, 2000. www.dfki.de/~sroa/papers/flairs_natural_language.pdf

[73] Landauer, T. K. and Dumais, S. T «A solution to Plato's problem: the Latent Semantic Analysis theory of acquisition, induction and representation of knowledge». (html) Psychological Review, 104(2) , 211-240. This paper takes a somewhat broader cognitive science perspective on dimension reduction and inductive learning.

[74] Josef Steinberger, Karel Ježek, «Using latent semantic analysis in text summarization and summary evaluation», (2004)

[75] Jure Leskovec, «Dimensionality reduction PCA, SVD, MDS, ICA, and friends», Machine Learning recitation April 27 2006

[76] Classification topologique non supervisée pour des données catégorielles, [En ligne], mise en ligne le lundi 20 juin 2011. URL: http://quetedesavoir.blogspot.com/2011/06/classification-topologique-non_20.html. Consulté le 06 Décembre 2011

[77] (Electronic Version): StatSoft, Inc. (2011). Electronic Statistics Textbook. Tulsa, OK: StatSoft. WEB: http://www.statsoft.com/textbook/. (Printed Version): Hill, T. & Lewicki, P. (2007). STATISTICS: Methods and Applications. StatSoft, Tulsa, OK. «Cluster Analysis», http://www.statsoft.com/textbook/cluster-analysis/

[78] Jean-Yves BAUDOT ; «Méthodes d'agrégation de la CAH», http://www.jybaudot.fr/Classif/agregcah.html,

[79] Philippe Preux ; Segmentation par les k-moyennes http://www.grappa.univ-lille3.fr/~ppreux/ensg/miashs/fouilleDeDonneesII/tp/k-moyennes/. mise en ligne 08/04/2010. Consulté en 10/2012

[80] Mark Sinka et David Corne, «A Large Benchmark Dataset for Web Document Clustering, Soft Computing Systems: Design, Management and Applications, Volume 87 of Frontiers in Artificial Intelligence and Applications » (2002)

[81] C. J. van Rijsbergen, Information Retrieval, London: Butterworths, 1979, [En ligne].

http://www.dcs.gla.ac.uk/Keith/Preface.html ; Consulté le 05 Janvier 2012

[82] I. Hatzilygeroudis, P. J. Vassilakos, A. Tsakalidis, « An Intelligent Medical System for Diagnosis of Bone Diseases - Published in the Proceedings of the 1st International Conference on Medical Physics and Biomedical Engineering (MPBE'94) », Nicosia, Cyprus, May 1994, Vol. I, 148-152.

[83] Antonio Ribeiro Filho, «Knowledge Engineer – SI Intelligent Systems»

MDSS, Medical Diagnosis Support System. http://www.lpa.co.uk/new_lin.htm . Mise en ligne le 06/01/2012. Consulté en 01/2012

[84] J.Tian and H.Tianfield, «A Multi-agent Approach to the Design of an E-medicine System». MATES 2003, LNAI 2831, pp. 85-94.

[85] Qiao Yang, «A Multi-agent Prototype System for Helping Medical Diagnosis» January 15th, 2008

[86] «EasyDiagnosis», http://www.easydiagnosis.com/cgi-bin/expert/quest.cgi ?accept.x=59&accept.y=11&mod=Chest+Pain&code. Mise en ligne le 20/01/2012. Consulté en 2012.

[87] Taisia Greceanu Stancovici, « On an Intelligent System for Medical Diagnosis», Using Electrographic Images 1996.

[88] Peter Szolovits, Ph.D.; Ramesh S. PATIL, Ph.D.; And William B. Schwartz, M.D «Artificial Intelligence in Medical Diagnosis», Reprinted from ANNALS OF INTERNAL MEDICINE Vol.108; No.1, pages 80-87. January 1988.
http://groups.csail.mit.edu/medg/ftp/psz/SchwartzAnnals.html

[89] Joseph Weizenbaum, ELIZA--A «Computer Program For the Study of Natural Language Communication Between Man and Machine»
http://i5.nyu.edu/~mm64/x52.9265/january1966.html [en]
http://fr.wikipedia.org/wiki/ELIZA [fr]
http://elizia.net/ [fr]
http://elizia.net/supervision.cgi « Questionnaire d'Elizia »

[90] J.Glass, et al, «The MIT ATIS System: December. 1994 Progress Report», Proc. DARPA Spoken Language Systems Technology workshop, 252-256, 1995.

[91] S. Harbeck, E. Nöth, H. Niemann, « Multilingual Speech Recognition». In SQEL, 2nd Workshop on Multi-Lingual Information Retrieval Dialogs, Plzeñ, Czech Republic, April 1997.

[92] Ying Dou, « Project Supervisor: Prof. Pascale Fung , Multilingual Speech Recognition and Spoken», Language Understanding (FP2-09).

[93] Landauer, T. K. and Dumais, S. T «A solution to Plato's problem: the Latent Semantic Analysis theory of acquisition, induction and representation of knowledge. (html) Psychological Review, 104(2), 211-240. This paper takes a somewhat broader cognitive science perspective on dimension reduction and inductive learning.

[94] Introduction to IR, LSA (and SEO) by Marie-Claire Jenkins, Consulté le 30 October 2011 http://www.justmeandmy.com/introduction-to-seo-ir-and-lsa-by-marie-claire-jenkins

[95] Thomas Hofmann, «Probabilistic latent semantic analysis», In Proceedings of the 15th Conference on Uncertainty in AI (1999).