Université de Montréal

QUERI: Un système de question–réponse collaboratif et interactif

Par
BADIS MERDAOUI

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures en vue de l'obtention du grade de M.Sc. en informatique

Août 2005

Université de Montréal
Faculté des études supérieures

Ce mémoire intitule :

QUERI: Un système de question–réponse collaboratif et interactif

présenté par :
BADIS MERDAOUI

a été évalué(e) par un jury composé des personnes suivantes :

Sylvie Hamel
président-rapporteur

Claude Frasson
directeur de recherche

Yann-Gaël Guéhéneuc
membre de jury

mémoire accepté le : 12 janvier 2006

Résumé

Un système de question–réponse est une application qui cherche dans un corpus de document, une réponse exacte à une question posée en langue naturelle. Les systèmes de question–réponse ont montré leur efficacité en utilisant le Web comme corpus de données avec leurs propres bases de données. Le résultat des systèmes de recherche d'information tels que les moteurs de recherche est une liste de documents à consulter, d'où l'importance aux systèmes de question–réponse dans le cas d'un usager qui cherche une réponse précise à une question donnée.

Des outils, tels que les systèmes de question–réponse, les hypermédias, les quiz adaptés pour l'auto-évaluation, etc. sont indispensables aux apprenants. Dans cette optique, notre système de question–réponse QUERI est développé pour être adaptable et intégrable dans n'importe quel système d'apprentissage. De même QUERI devrait pouvoir justifier la réponse et évaluer l'adéquation de la réponse à la question; assurer l'interaction avec l'usager, ce qui aide à lever l'ambiguïté des questions; prendre en compte le contexte général de la question et répondre en temps réel à la question de l'usager.

Afin d'atteindre ces objectifs, nous utilisons plusieurs techniques, parmi lesquelles : l'approche multi-agents, les patrons d'extractions, la recherche d'information, le traitement automatique de la langue naturelle (TALN) et le raisonnement à base de cas.

Mots clés: question–réponse, outils d'aides aux apprenants, aide intelligente, représentation et gestion des connaissances, traitement de la langue naturelle, recherche d'information, raisonnement à base de cas.

Abstract

Systems of question-answer are evolving towards a new generation of search engines. The "user-machine" interaction is going to be changed. Systems of question-answer showed their efficiency by exploiting the great majority of data existing in their data bases with information extracted from the Web. Intelligent agents, Case-Based Reasoning techniques, and pattern extraction are used in the present system which allows a more efficient interaction with the user.

Keywords: question-answering, cognitive models and agents, formalizing and managing knowledge in organizations, Case-Based Reasoning.

TABLES DES MATIÈRES

TABLES DES FIGURES

Remerciements

Je tiens à remercier M. Claude Frasson, professeur à l'UdM et mon directeur de recherche pour m'avoir proposé un projet intéressant et sans qui ce projet n'aurait jamais vu le jour, je le remercie également pour toute son aide, son assistance et ses précieux conseils durant toute la période de ma formation.

Je tiens à remercier tous les membres du laboratoire HERON ainsi que toutes les personnes ayant contribué de loin ou de près à l'élaboration de ce travail.

Ce projet a été rendu possible par la contribution de Valorisation Recherche Québec (VRQ) et fait partie du projet DIVA : 2200-106.

CHAPITRE 1

1. INTRODUCTION

Les systèmes de question–réponse doivent répondre à des questions plus précises que les systèmes de recherche d'information pour satisfaire au mieux les besoins des usagers.

Un système de question–réponse est une application qui cherche dans un corpus de document, une réponse exacte à une question posée en langue naturelle.

En effet, face à une question telle que "*Les étoiles se déplacent-elles dans le ciel ?*", les moteurs de recherche traditionnels renvoient une liste de documents jugés pertinents par rapport à la question, tandis qu'un système de question–réponse, retourne à l'usager une réponse spécifique, telle que "oui".

Les systèmes de question–réponse qui existent actuellement ne sont pas orientés vers un domaine spécifique comme celui de l'apprentissage. Nous avons constaté que la majorité de ces derniers sont de nature indépendante et non intégrable aux systèmes d'apprentissage. D'une part, il n'y a pas de systèmes qui ont été développés dans l'esprit d'être réutilisés, modifiés, configurés selon les besoins des systèmes d'apprentissage. Par exemple, l'intégration ou le changement du corpus local de données demande le changement dans le code source du système pour tenir compte des nouvelles données.

D'autre part, l'exactitude, la précision et la justification de la réponse sont des buts loin d'être réalisés. Le manque d'interaction avec l'usager et celui d'un corpus local de données sont des facteurs qui diminuent l'exactitude de la réponse.

Actuellement, il n'existe, dans l'industrie du logiciel, aucun système de question–réponse qui soit orienté vers le domaine d'apprentissage.

Le sujet de notre mémoire est à l'intersection de deux domaines principaux, l'apprentissage et la recherche d'information. L'objectif de notre recherche est de

concevoir et de réaliser un système de question–réponse comme outil d'aide aux apprenants, qui sont intégré dans un système d'apprentissage.

Dans le domaine de l'apprentissage, des outils tels que les systèmes de question–réponse, les hypermédias, les quiz adaptés pour l'auto-évaluation, etc. sont indispensables aux apprenants puisqu'ils permettent de :

- compléter la matière du cours ;
- avoir de l'information supplémentaire;
- s'évaluer d'une façon correcte.

Dans l'axe d'apprentissage, une des difficultés auxquelles les développeurs font face lors de la réalisation des systèmes d'apprentissage est l'adaptation ou l'intégration d'un outil d'aide aux apprenants. Dans le cas d'un outil simple comme un hypermédia ou une FAQ (*"Frequently Asked Questions"*), l'outil doit être conçu dès le début pour le système principal et il est rare de pouvoir l'adapter ou l'intégrer dans d'autres systèmes. De même, pour un système indépendant comme celui des question–réponse, l'adaptation ou l'intégration est une tâche difficile, voir impossible si le système n'est pas conçu dès le début pour être adaptable et intégrable.

D'autres problèmes peuvent rendre le sujet difficile. La problématique du domaine de question–réponse se situe à l'intersection de plusieurs autres domaines, dont la recherche d'information et le traitement de la langue naturelle.

Malgré les résultats, relativement positifs des nouveaux systèmes de question–réponse, les chercheurs posent, toujours les questions suivantes:

1. Comment peut-on répondre en temps réel à la question, justifier la réponse et savoir évaluer l'adéquation de la réponse à la question ?
2. Comment peut-on déterminer les documents les plus pertinents à la question de l'usager et choisir celui qui contient la réponse ?
3. Comment choisir parmi plusieurs réponses candidates (qui ont le même degré de similarité) celle qui sera la plus pertinente ?

Pour faire face à ces problèmes, plusieurs approches peuvent êtres utilisées.

Le premier problème peut être résolu en utilisant des techniques d'intelligence artificielle, en particulier "d'agents intelligents" [Chu-Carroll *et al.*, 2003 et 2004]. Le deuxième peut être facilité par l'approche de raisonnement à base de cas [Gresse *et al.*, 2001]. En ce qui concerne le troisième problème, il peut être résolu par l'interaction directe avec l'usager [Nyberg *et al.*, 2002].

Dans cette optique, notre objectif est de développer un système adaptable et intégrable dans n'importe quel système d'apprentissage et qui devrait pouvoir :

- justifier la réponse et évaluer l'adéquation de la réponse à la question ;
- assurer l'interaction avec l'usager, ce qui aide à lever l'ambiguïté des questions ;
- prendre en compte le contexte général de la question ;
- répondre en temps réel à la question de l'usager.

Par notre travail, nous mettons à la disposition des chercheurs et des développeurs dans le domaine de l'apprentissage, un outil d'aide aux apprenants que nous jugeons important, voir indispensable, pour accéder à l'information supplémentaire. C'est un système de question–réponse collaboratif et interactif orienté vers le domaine d'apprentissage, il prend en entrée une question posée en langue naturelle et retourne en sortie une réponse spécifique. Ce système est adaptable et intégrable sans aucune intervention au niveau du code.

De même nous mettons à la disposition des chercheurs un ensemble d'exigences et de problématiques relatives aux systèmes de question–réponse ainsi qu'un nombre de solutions. Nous avons veillé également à fournir un résumé de tous les concepts essentiels pour élaborer un système qui répond aux exigences de l'industrie d'apprentissage et de formation.

Des scénarios d'utilisation de notre système de question–réponse –QUERI– et une validation des résultats (Publications de conférence et Validation avec des utilisateurs potentiels) sont présentés au chapitre 4.

Le deuxième chapitre présente l'état de l'art relatif aux systèmes de question–réponse. Après un bref aperçu historique, il mentionne les problèmes spécifiques au

développement de ces systèmes et décrit ensuite leurs différents types. Un inventaire des différentes phases d'analyse et un résumé des approches les plus populaires dans le développement de ces systèmes sont présentés dans ce chapitre avec une vue critique guidée par quelques critères.

Le chapitre 3 est consacré au développement du système QUERI. Nous allons passer rapidement à travers l'ensemble des technologies utilisées et présentons leurs concepts clés. Nous présentons ensuite les solutions que nous proposons pour pallier aux problèmes cités. Nous expliquons le principe d'une architecture orientée multi-agents et basée sur l'interaction système–usager et nous également justifions le choix de cette architecture.

Dans le chapitre 4, nous voyons les détails d'implémentation de l'architecture QUERI. Nous abordons, premièrement, les outils et l'infrastructure utilisés pour le développement du système. Nous présentons une évaluation des résultats obtenus ainsi que des perspectives de développement futur.

Nous terminons ce mémoire par une conclusion qui résume le travail présenté et fait ressortir nos recommandations pour le développement des systèmes de question–réponse dans le domaine d'apprentissage.

CHAPITRE 2

2. État de l'art

2.1 Introduction

Plusieurs systèmes de question–réponse ont été développés dans le milieu universitaire et industriel et sont présentés notamment dans les conférences TREC (Text REtrieval Conference). TREC est une conférence internationale initiée au début des années 90 par le "*National Institute of Standards and Technology*" (NIST) aux Etats-Unis en collaboration avec DARPA/ITO "*Defense Advanced Research Projects Agency - Information Technology Office*". Cette conférence est divisée en plusieurs domaines parmi lesquels celui de la question–réponse qui a été introduit lors de la TREC-8 (1999) afin de répondre à des questions posées en langue naturelle à partir d'un corpus textuel.

Dans ce chapitre nous allons :

- fournir un rapide aperçu historique des systèmes sus mentionnés ;
- mentionner les problèmes spécifiques au développement de ces systèmes ;
- décrire les différents types de systèmes question–réponse existants ;
- présenter les différentes phases d'analyse ;
- décrire les approches les plus populaires dans le développement de ces systèmes, notamment celles adoptés par les spécialistes de TREC ;
- décrire les problèmes relatifs à l'interaction et à la flexibilité rencontrés dans leur utilisation.

Dans la section 2.2, nous décrivons brièvement l'historique et l'état de l'art des recherches sur ces systèmes. La section 2.3 est consacrée à décrire leurs différents types. Nous montrons ensuite dans la section 2.4 les éléments d'analyse tels que : l'analyse de la question, l'analyse des entités nommées, la

mise en relation des entités et les ressources. Nous montrons également les systèmes TREC dans la section 2.5.

Finalement, dans la section 2.6 nous discutons les problèmes relatifs au manque d'interaction et de flexibilité. Nous concluons ce chapitre par la clarification des besoins en système de question–réponse qui mettent en œuvre des fonctionnalités nouvelles telles que l'évaluation de l'existence d'une réponse, la justification de la réponse, la synthèse des réponses, le dialogue d'aide à la formulation de requête ou encore la capacité de compréhension de texte.

2.2 Historique des systèmes de question–réponse

De très nombreux spécialistes continuent de développer des systèmes de question–réponse notamment dans le cadre des conférences TREC.

Beaucoup de recherches antécédentes ont exploré le domaine de question–réponse, en dehors de la conférence TREC.

Les systèmes de question–réponse ont émergé comme un domaine de recherche dans le début des années 60. Les ancêtres ont été les systèmes d'interrogation de bases de données en langage naturel. Le système BASEBALL [Green et *al.*, 1961] permettait l'interrogation de résultats de baseball, le système LUNAR l'analyse de roche lunaire [Woods et *al.*, 1972] et le système LIFER [Hendrix, 1977] l'analyse de statistiques sur des employés [Poibeau *et al.*, 2003]. Les systèmes devaient être capables de répondre à des questions telles que:

Who did the Red Sox lose on July 5? (BASEBALL)

What is the average concentration of aluminium in high alkali rocks? (LUNAR)

What is the average salary of math department secretaries? (LIFER)

Le principe était de traduire la question posée en langue naturelle en une requête de base de données. Ces systèmes étaient consacrés à un domaine bien spécifique et il était difficile de les lier à d'autres domaines [Poibeau *et al.*, 2003].

Durant les années 70 et 80, de nombreux domaines ont adopté les systèmes de question–réponse. Dans le domaine de la robotique le système SHRDLU [Winograd, 1972], dont le but est la compréhension de la langue naturelle,

permet une interaction entre un humain et un robot en utilisant un dialogue simple au sujet d'un petit monde d'objets (blocs) présentés sur un écran de visualisation. Ce dialogue est sous forme de questions et d'ordres donnés au robot en langue naturelle. Ces recherches ont été faites pour définir la notion d'intelligence et la façon de l'implanter dans une machine (robot, ordinateur).

Le système QUALM [Lehnert, 1977 et 1981] était consacré à la réservation dans les restaurants [Schank et Abelson, 1977].

Le système UNIX CONSULTANT [Wilensky, 1982] a adopté une approche similaire à celle des systèmes précédents pour répondre à des questions à propos du système Unix [Poibeau *et al.*, 2003].

À la fin des années 80, la recherche dans le domaine de question–réponse était principalement réduite aux universités et aux centres de recherche. Au milieu des années 90, l'apparition du *World Wide Web* (WWW) a fondamentalement renouvelé les approches [Poibeau *et al.*, 2003].. À partir de ce moment, les utilisateurs étaient capables d'interroger les systèmes de question–réponse et les moteurs de recherche pour exploiter la masse de données électroniques. L'énorme accroissement des masses de documents électroniques a rendu le Web comme une grande masse de connaissances à interroger. Avant la popularisation du Web, les systèmes développés avaient une popularité limitée et leur fonction était restreinte à la consultation de leurs propres bases de données locales déjà préétablies. Cela limitait leur marge de recherche et par voie de conséquence leur performance.

Un nouvel intérêt pour la tâche de question–réponse est apparu grâce à la conférence TREC initié au début des années 90. La recherche sur la question–réponse a gagné effectivement son expansion comme la principale tâche de la TREC.

2.3 Quelques systèmes de question–réponse

Le plus célèbre des systèmes développés est ELIZA [Weizenbaum, 1966] qui simulait une séance de thérapie chez un psychiatre[1].

On présente ci-dessous un dialogue avec ELIZA extrait de [Jurafsky *et al.*, 2000]:

- **Utilisateur: Men are all alike**
- **ELIZA: IN WHAT WAY?**
- **Utilisateur: They're always bugging us about something or other**
- **ELIZA: CAN YOU THINK OF A SPECIFIC EXAMPLE ?**
- **Utilisateur: My boyfriend says I'm depressed all the time**
- **ELIZA: I AM SORRY TO HEAR YOU ARE DEPRESSED**

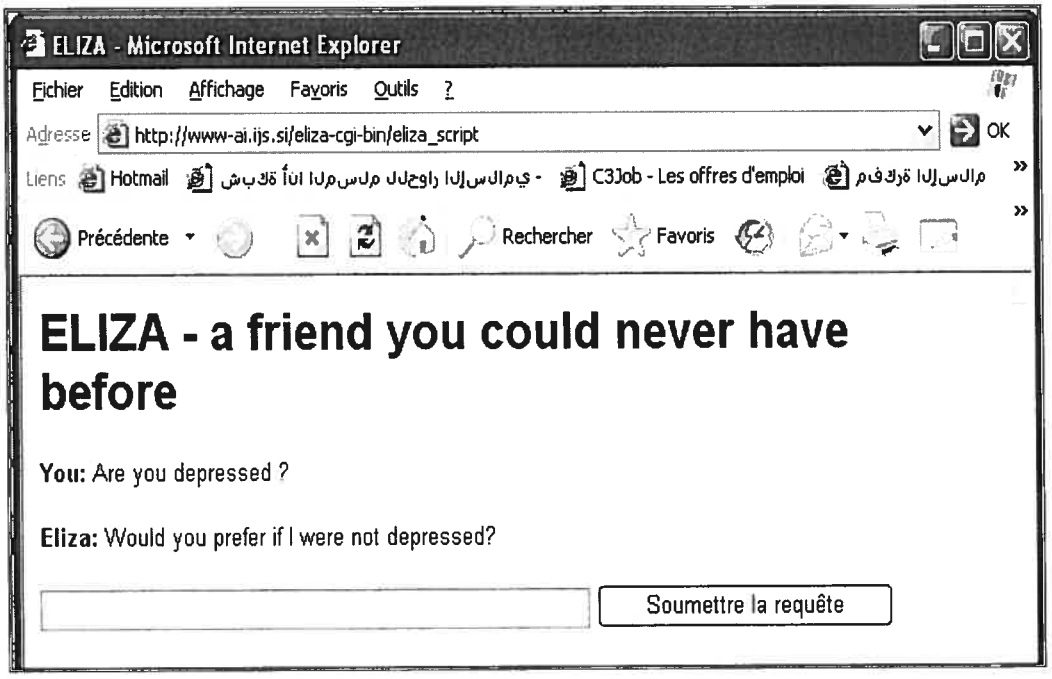


Figure 2.1 : Le système ELIZA

Récemment des projets comme QACTIS [Schone *et al.*, 2004], QALC [Chalendar *et al.*, 2002], PIQUANT [Chu-Carroll *et al.*, 2003 et 2004], QUANTUM [Plamondon, 2002] et JAVELIN [Nyberg *et al.*, 2002], se sont intéressés à la réponse automatique en utilisant des ressources disponibles sur le Web. Chacun de ses système a développé sa propre approche afin d'améliorer la qualité des réponses offertes aux usagers.

[1] http://www-ai.ijs.si/eliza/eliza.html

Nous avons aussi les systèmes de la famille Ask[1] (Ask Jeeves, Ask Ted et Ask Ellen). Ask, tente de répondre à une question formulée en langue naturelle. Il commence la recherche dans sa propre base de données en associant la question à des requêtes similaires qui ont déjà été répondues. Il affiche la réponse si la question de l'usager convient à l'une des requêtes similaires, sinon il affiche un petit ensemble d'URL susceptibles de contenir la réponse.

Cette approche se base sur le fait que 80% des questions des usagers du Web tournent autour de 20% des questions déjà existantes sur le Web.

En stockant les réponses des 20% des questions qui figurent sur le Web, la grande majorité des questions des usagers peuvent être répondues avec une grande précision très rapidement [Kosseim *et al.*, 2000].

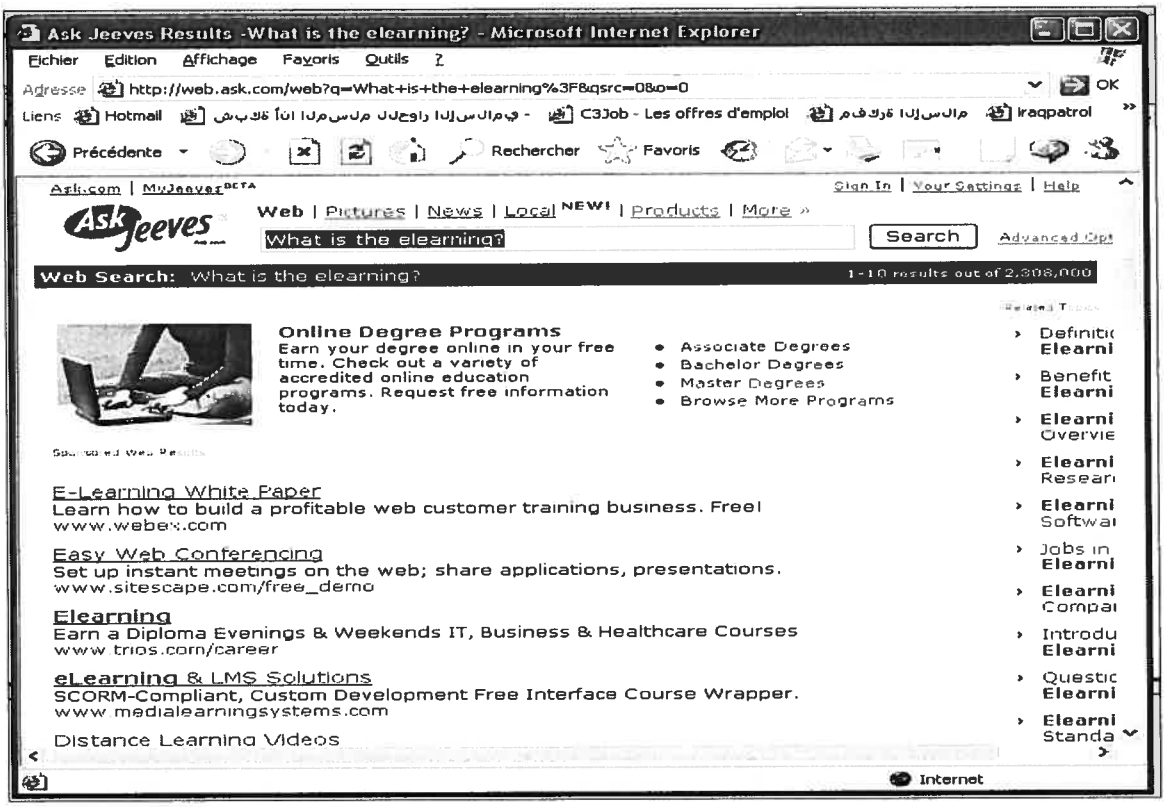


Figure 2.2: Résultats de *Ask Jeeves: what is the elearning?*

[1] http://www.ask.com/

2.4 Éléments d'analyse de la question

L'analyse de la question est un élément très important pour pouvoir trouver la bonne réponse. Nous détaillons dans cette section les différents composants d'un système d'analyse de question classique, à savoir la caractérisation du type de la question, l'identification des entités et des relations entre entités de la question.

Un système de question–réponse recherche généralement dans un corpus de documents une réponse précise. Par exemple, si un usager pose la question "Les étoiles se déplacent-elles dans le ciel ?", le système devra répondre par une réponse bien spécifique "Oui" et non par une liste de liens hypertextes.

Le schéma de fonctionnement général de la plupart des systèmes de question–réponse actuels est le suivant :

- analyser la question en langue naturelle ;
- extraire les mots-clés pour former une requête ;
- rechercher les documents pertinents par interrogation d'un moteur de recherche traditionnel ;
- parcourir les documents retournés par le moteur de recherche pour extraire les réponses candidates et à partir des informations de la question, la réponse qui présente le plus grand score de similarité est retournée.

2.4.1 Architecture générale d'un système question–réponse

Les systèmes de question–réponse reposent généralement sur des architectures faites d'un ensemble de modules tels que : un analyseur de la question, un moteur de recherche traditionnel, un module d'extraction de passages (paragraphes susceptibles de contenir la bonne réponse) et un module d'extraction de réponse. La figure suivante présente l'architecture générale d'un système question–réponse.

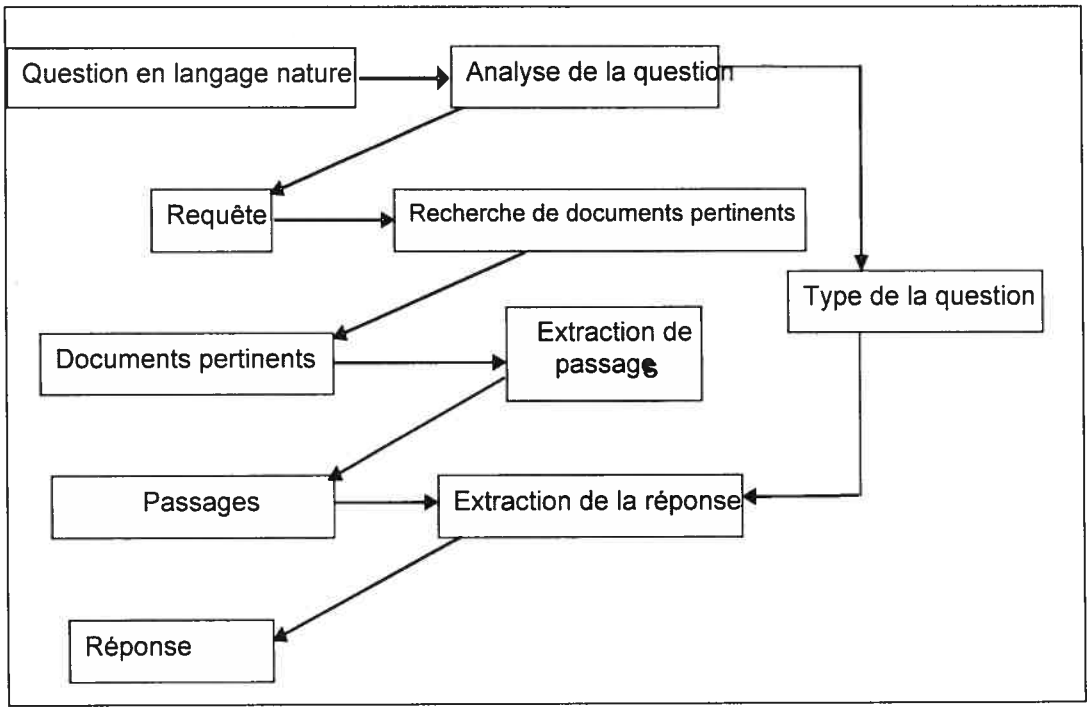


Figure 2.3: Architecture d'un système question–réponse

Les tâches d'indexation de la question et de l'interrogation d'un moteur de recherche se basent sur des techniques de recherche d'information, tandis que l'analyse de la question et l'extraction de la réponse sont des tâches de traitement automatique de la langue naturelle (TALN) [Nyberg *et al.*, 2002].

2.4.2 Détermination du type de la question

La première phase dans le module d'analyse de question est de déterminer le type de la question parmi les deux types principaux de questions, celles portant sur les entités nommées et les questions complexes.

Une entité nommée est un ensemble de noms de personnes, d'entreprises, de lieux, etc. Les questions portant sur des entités sont plus faciles que les questions portant sur des définitions et des caractérisations d'objets.

Les questions suivantes demandent des entités nommées comme réponses:

Qui est le ministre d'éducation du Québec ? (nom propre : personne)

Où se trouve Alger ? (nom propre : lieu)

Tandis que les questions suivantes demandent des réponses plus complexes: définition, méthode...

C'est quoi le elearning ?

Quelles sont les différentes étapes d'un projet ?

D'autres recherches ont établi des classifications plus détaillées pour résoudre le problème d'extraction de réponse. Par exemple, des questions qui demandent des comparaisons, des définitions, des quantifications, des conséquences, des causes, etc. Parmi ces classifications nous avons celles de [Harabagiu, 2000] et de [Graesser *et al.*, 1992].

En générale, la méthode adoptée par les systèmes de question–réponse est de faire correspondre des types de question` à des patrons. Un patron est un modèle qui est déjà prédéfini avec lequel on compare un autre (réponse candidate).

En quelle année → Date

Qui → Nom de personne

Où → Nom de lieu

L'identification des questions « complexes » ne pose pas obligatoirement de gros problèmes en soi (en anglais, les questions introduites par *why* et *how* ou *what* sont plus difficiles que des questions introduites par *who* ou *when*). C'est la recherche d'une réponse juste et pertinente qui pose davantage de problème dans ce cas [Poibeau *et al.*, 2003].

Au-delà de l'emploi de patrons certains systèmes effectuent une analyse syntaxique des questions, généralement à l'aide d'analyseurs robustes. Par exemple, l'équipe du LIMSI [Chalendar *et al.*, 2002] combine les analyses produites par plusieurs analyseurs syntaxiques robustes pour construire une analyse pour des questions de TREC [Cui *et al.*, 2004].

2.4.3 Détection d'entités nommées

Dans le cas d'une question de type entité nommée, l'entité nommée la plus proche des mots de la question ne répond pas toujours avec précision à celle-ci. Des connaissances sémantiques ne pourront pas non plus permettre de rejeter la mauvaise réponse, car les noms propres sont peu présents dans les bases de données sémantiques et ils ont souvent les mêmes caractéristiques sémantiques. Une analyse syntaxique des relations de la phrase permettra de trouver la bonne réponse [Tan *et al.*, 2004].

2.4.4 Analyse syntaxique

Un analyseur syntaxique permet de donner, par exemple le type de la question, le "focus", le type attendu de la réponse, le type sémantique général, et les relations de dépendance de la question. Cette analyse est rendue possible par la forme syntaxique relativement figée des questions. D'un autre coté, les phrases candidates sont analysées par un étiqueteur morpho-syntaxique, qui ne retourne qu'une analyse partielle de ces phrases [Tan *et al.*, 2004].

2.4.5 Création de la requête

L'analyse de la question est une tâche principale d'un système de question–réponse. C'est à partir de cette phase de traitement qu'est générée la requête adressée au corpus de données. Pour créer ou générer une requête on passe par :

- l'analyse lexicale et la génération des jetons (*Tokenization*) ;
- l'extraction des termes appropriés (indexation) ;
- la correction d'orthographe ;
- la normalisation (par exemple, prendre la racine pour un verbe régulier).

Plusieurs cas peuvent se présenter lors de l'analyse : dans le cas d'une analyse dite « complète », le système détermine le type de question, ainsi que ses éléments clés. Ces éléments seront utilisés pour former une requête à poser au moteur de recherche pour déterminer l'ensemble des documents pertinents.

Dans le cas d'une analyse dite « incomplète », seul le type de l'entité recherchée est déterminé. Ce cas se présente après l'échec de l'analyse en profondeur de la question et par conséquent, une absence de mots clés extraits. Ainsi, la requête est générée en utilisant l'analyse syntaxique. La présence des noms propres dans la question constitue une contrainte dans la construction de la requête, car ceux-ci représentent généralement le focus de la question.

2.4.6 WordNet : pour l'extraction des réponses

Wordnet est une ontologie[1] lexicographique pour la langue anglaise. Elle a été développée en 1985 par le *Cognitive Science Laboratory* à *Princeton University* sous la direction du Professeur George A. Miller. Cette ontologie est sous la forme d'un réseau lexical ou sémantique où les nœuds sont des ensembles de synonymes (synsets). Chaque synset correspond au sens d'un mot, qui est défini par les relations qu'il entretient avec les sens voisins [Fellbaum *et al.*, 1998]. Les relations privilégiées dépendent de la partie du discours considérée : l'hyponymie pour les noms, l'antonymie pour les adjectifs et les adverbes.

Afin de pouvoir extraire une réponse, il est utile de disposer d'un thesaurus pour répondre à des questions du type «*Qui est le plus grand animal ?*». Pour valider les réponses proposées, il est préférable d'être capable de vérifier qu'elles désignent bien des animaux ou non. Un filtre sur le type de réponse mis en parallèle avec le type demandé dans la question est alors fondamental.

---

1 Une organisation hiérarchique de la connaissance sur un ensemble d'objets par leur regroupement en sous-catégories suivant leurs caractéristiques essentielles.

2.5 Les conférences TREC "Text REtrieval Conference"

L'objectif de cette conférence est d'encourager les recherches dans le domaine de la recherche d'information. Les résultats présentés à TREC sont chaque année résumés dans des actes édités par le NIST. Ces résultats sont également organisés par groupe et par tâche et décrivent les différentes techniques et systèmes expérimentés.

Il existe trois types de questions pour la tâche question–réponse :

- des questions qui demandent des entités (ou "factoid questions"). Par exemple, *"what is the population of Iceland?"* ;
- des questions qui demandent des listes (ou "list questions"). Par exemple, *"What are the different steps of a project?"* ;
- des questions qui demandent des définitions comme *"what is eLearning?"*.

Nous allons détailler chacune des tâches de la conférence TREC dans ce qui suit.

2.5.1 Tâche principale : Ad hoc

La principale tâche ad hoc dans TREC est d'évaluer les performances des systèmes de recherche d'informations. Sur des ensembles statiques de documents, seuls les requêtes changent. Cette tâche est similaire à une recherche dans une bibliothèque par exemple, où la collection est connue mais les questions susceptibles d'être posées ne le sont pas.

Pour cette tâche, les participants disposent d'environ 2 giga-octets de documents et d'un ensemble de 50 "sujets" en langage naturel. Pour chaque "sujet", les participants créent une requête et soumettent les 1000 "meilleurs" documents pour ce "sujet". Les participants sont tenus de décrire la façon dont ils ont construit leurs requêtes à partir de "sujets" : entièrement automatiquement ou manuellement.

2.5.2 Tâches spécifiques

Ces tâches ont été introduites progressivement dans TREC (depuis TREC4 en 1996) afin de permettre l'évaluation de problèmes spécifiques en recherche d'information telles que le filtrage, le croisement de langues, la recherche dans de très large corpus (25 giga-octets et plus), les modèles d'interaction ...

2.5.2.1 Filtrage d'informations

Dans cette tâche, à l'inverse de la tâche principale ad hoc, les "sujets" sont stables alors que le flot de documents varie à chaque fois. Pour chaque document, le système doit prendre une décision binaire, relativement à la pertinence du document (au lieu de fournir une liste ordonnée).

2.5.2.2 Recherche d'information multilingues

Une tâche ad-hoc dans laquelle les documents sont en anglais, allemand, français, ou italien, et les sujets sont fournis dans chaque langue. Le centre d'intérêt de cette tâche est la recherche de documents qui concernent le sujet indépendamment de la langue.

2.5.2.3 Modèles d'interaction

L'objectif de cette tâche est d'étudier les modèles d'interaction en recherche d'information.

2.5.2.4 Construction de requêtes

Cette tâche a pour objectif de comparer les méthodes de construction de requêtes et l'influence de ces méthodes sur les performances des systèmes.

2.5.2.5 Tâche question–réponse

L'objectif de cette tâche est de travailler sur des parties au lieu du document tout entier. Au lieu de rechercher la pertinence d'un document dans sa totalité, on va rechercher la pertinence de passages (ou extraits) de documents. Pour chacune des 200 questions, les systèmes doivent restituer des extraits de documents qui sont pertinents pour ces questions. Plusieurs niveaux de réponses doivent

être testés. Ces niveaux diffèrent par la longueur maximale de l'extrait (en octets) : de phrases courtes (2 à 3 mots) au document en entier (1000 mots). On peut assimiler ces types d'extraits à des fenêtres de 50, 200 , 1000, ... octets.

2.5.2.6 Recherche d'informations dans des documents sonores

Cette tâche ad hoc étudie la capacité des systèmes de recherche d'information à retrouver des documents sonores (textes parlés). Il s'agit de comparer l'efficacité de leurs systèmes sur des transcriptions de journaux télévisés.

2.5.2.7 La tâche Web

Une nouvelle tâche ad-hoc dans laquelle les documents sont un ensemble représentatif de documents issus du World Wide Web.

2.5.3 Quelques systèmes TREC

Dans cette section, nous présentons quelques méthodes qui ont eu un apport à la problématique visée. Nous présentons notamment les systèmes : PIQUANT d'IBM TREC13 [Chu-Carroll *et al.*, 2004], QACTIS TREC13 [Schone *et al.*, 2004], LIMSI, QALC TREC12 [Chalendar *et al.*, 2002] et Modèles d'extraction sous forme d'expressions régulières TREC10 [Soubbotin *et al.*, 2003]. D'autres travaux sont aussi intéressants mais ne seront pas présentés ici. Par exemple, JAVELIN TREC11 [Nyberg *et al.*, 2002], POWERANSWER TREC11 [Molovan et al. 2003] et QUANTUM TREC-X [Soubbotin *et al.*, 2001].

2.5.3.1 PIQUANT II d'IBM TREC13

PIQUANT II a été réalisé pour créer une plateforme efficace de système de question–réponse. Les buts primaires sont la réutilisation des composants et l'amélioration du temps de réponse du système PIQUANT original. La nouvelle architecture continue à soutenir l'approche multi-agents où différentes stratégies sont utilisées pour répondre aux différents types de

questions. Cette approche a été testée pour la première fois dans la TREC11 avec un grand succès. Grâce à cette technique PIQUANT a passé de la 11^{e} à la 5^{e} place. L'une des améliorations principales par rapport au système précédent est le développement de l'agent de profil, qui est un agent intelligent à base de modèles d'extraction. Cet agent a fait une amélioration de 61% de réponses exactes. Le schéma suivant montre un diagramme du PIQUANT II dans TREC 2004.

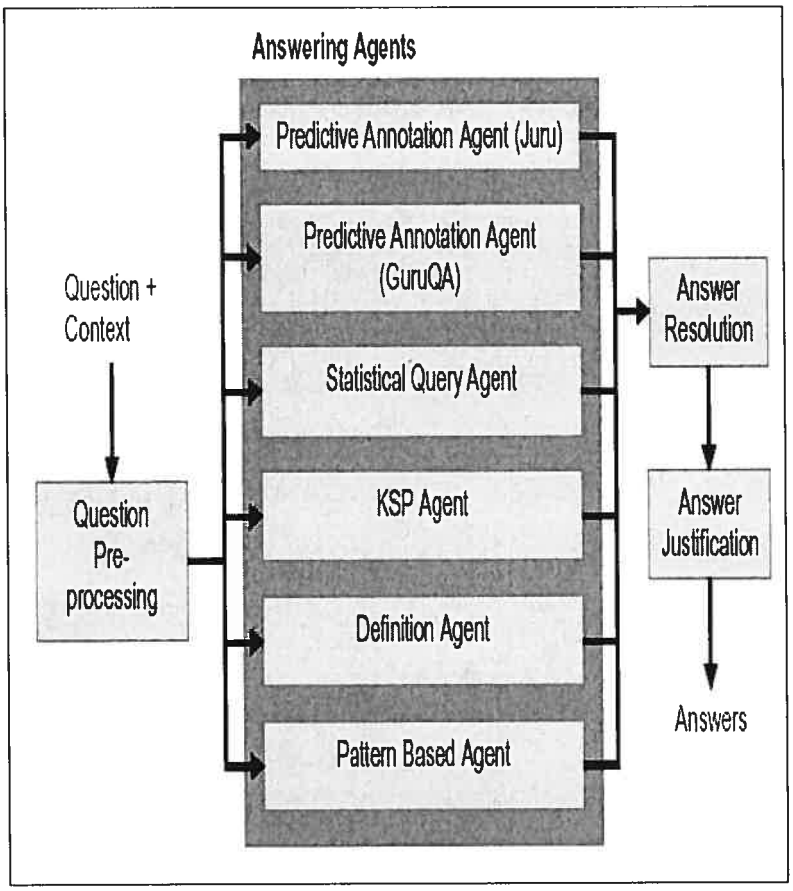


Figure 2.4: Architecture du système PIQUANT II

L'analyse de question produit un QFRAME qui contient le type de réponse exigé, le type de question, des mots-clés de la question et une forme sémantique simple. Dans l'étape suivante le système passe le QFRAME au générateur QPLAN qui est un programme général qui dirige le traitement. Il contient la liste des agents disponibles qui vont être utilisés. Les résultats

sont passés au module "*Answer Resolution*". Ce dernier fait le choix de la réponse la plus pertinente selon le type de la question.

Le composant "*Answering Agents*" est le cœur du système PIQUANT II. PIQUANT II fournit les mécanismes pour exécuter ces différents agents de réponse en parallèle et pour accumuler et envoyer leurs résultats au composant "*Answer Resolution*". Ce dernier combine les réponses candidates avec les scores les plus élevés de chaque agent et produit un ensemble de réponses candidates. Dans le TREC 2004, le composant "*Answer Resolution*" adopte la probabilité a priori d'égale en additionnant les points de confiance de toutes les réponses sémantiquement équivalentes afin de produire la réponse ayant le plus haut score.

En termes d'amélioration, le nouveau système s'exécute mieux que l'ancien (TREC 2003) sur chacune des trois tâches secondaires (questions sur les entités nommées, questions complexes et autres questions), cette amélioration s'étend largement de 5% à 160%. Néanmoins ce dernier reste limité à cause de manque d'interaction avec l'usager, ce qui ne lui permet pas de prendre une décision en cas d'ambiguïté.

2.5.3.2 QACTIS TREC13

Le sujet de question–réponse représente un secteur d'intérêt pour le programme AQUAINT conduit par le gouvernement des États-Unis pendant trois ans (2001-2004). QACTIS est un système question–réponse pour texte, image et discours, il est développé par le Département de la Défense. L'objectif final pour ce projet est de développer un prototype dans lequel des utilisateurs peuvent poser des questions dans plus d'une langue (par exemple, anglais et espagnol), interpréter ces questions indépendamment de la langue et renvoyer des réponses. La spécificité principale de ce projet est qu'il permet de répondre à des questions en utilisant la voix. Ceci a permet d'enrichir pratiquement le domaine d'étude. Pour les buts de la conférence TREC, QACTIS se base sur le texte anglais dans l'espoir d'introduire d'autres langues et d'autres types de supports. En termes de ses capacités de

questions et réponses, ce système se compose de trois composants principaux. Les deux premiers composants sont utilisés pour interpréter les questions et postuler les réponses possibles. Le troisième permet de valider les réponses proposées. Le schéma suivant montre l'architecture du système QACTIS.

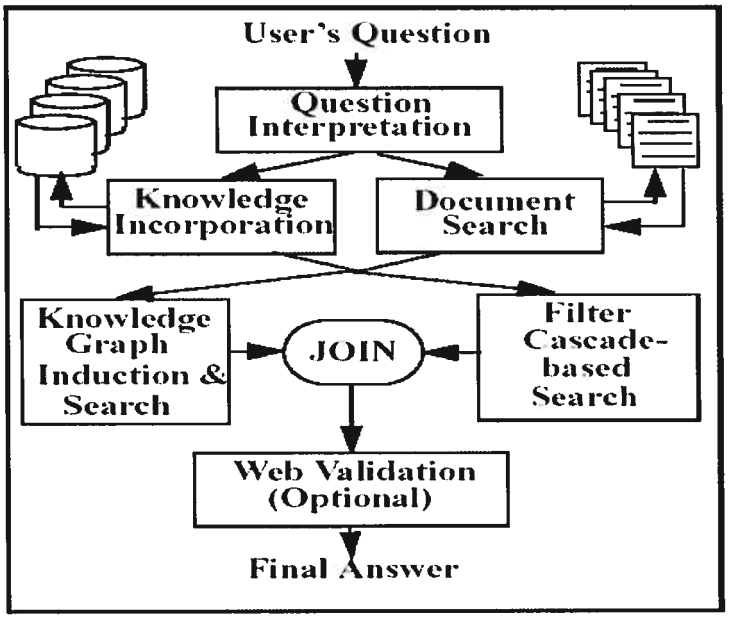


Figure 2.5: Architecture du système QACTIS

Dans ce qui suit, nous donnons une brève description de chacun des composants du système :

1. **Le composant d'analyse de la question *"Question Interpretation"*** fait l'analyse et le classement de la question sous l'un des six types adoptés par les auteurs (*When, Where, What, Who, How, Why*).
2. **Le copmposant d'incorporation de la connaissance *"Knowledge Incorporation"*** utilise *WordNet* et d'autres dictionnaires électroniques pour déterminer l'ontologie de la question, juger les significations des mots, déterminer les synonymes (synset) et déterminer l'information taxonomique et ontologique (utiliser des synonymes, hyponymes, etc.; dans le système, chaque mot de la question est associé à un concept intégré dans l'ontologie).

3. **Composant de recherche de documents *"Document Search"***

   Pour la recherche de documents pertinents à la question de l'usager, QACTIS utilise le moteur de recherche Lemur. QACTIS utilise Lemur comme une boîte noire où l'entrée est la question de l'usager est la sortie est une liste de documents.

4. **Composant d'induction de Connaissance graphique *"Knowledge-Graph Induction"*** pour convertir l'ensemble des documents retourné par Lemur en forme graphique simple pour trouver les sous graphes reliés a la question.

5. **Le composant Cascade De Filtres " Filters Cascade "** sert à identifier des réponses de potentiel et à éliminer également les autres. Les réponses laissées à l'extrémité de tous les filtres sont considérées comme une réponse appropriée.

6- **Composant de validation Web *"Web Validation"*** utilise le Web pour valider les réponses candidates, il calcule le score de pertinence de chaque réponse en utilisant la fréquence des éléments repérés (utilisation de la redondance du Web).

Le problème principal de QACTIS est le fait qui fournit des réponses inexactes; le système se limite à chercher seulement si le bon genre de réponse est identifié. L'évaluation de QACTIS dans le cadre de la conférence TREC a été basée sur le facteur "précision de réponse".

La condition de précision de réponse a eu une conséquence extrêmement pénalisante pour le système. QACTIS a produit seulement 47 réponses correctes. Le système a fournit 12 réponses qui ont été considérées inexactes, ce qui est influencé sur le classement final de ce dernier.

2.5.3.3 LIMSI, QALC TREC12

Ce système a comme principe l'utilisation de plusieurs sources de données. Il utilise en effet deux agents de recherche indépendants, l'un pour Internet et l'autre pour le corpus TREC, ce qui permet au système d'obtenir des scores élevés de certitude pour des réponses données.

Le système est constitué des modules suivants : analyse de question, sélection de document, identification d'entité-nommée et extraction de réponse.

Vu la grande quantité de documents disponibles sur le Web et aussi leur haute redondance, le système augmente ses chances d'avoir la réponse exacte [Clarke et *al.*, 2001]. Le but du système est d'obtenir des scores de certitude élevés en utilisant différentes sources de données. Cependant, l'évaluation de réponse est grandement pénalisée par l'absence d'interaction avec l'usager ainsi que l'utilisation de la même technique pour répondre à n'importe quel type de question. La figure suivante montre l'architecture du système QALC.

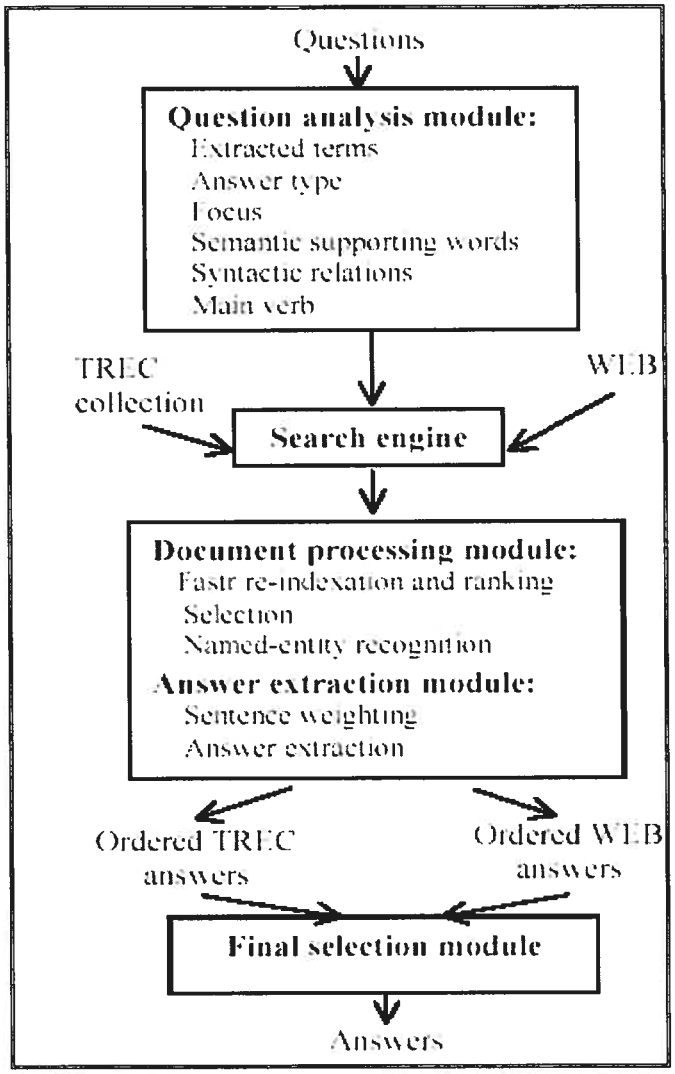


Figure 2.6: Architecture du système QALC

2.5.3.4 Modèles d'extraction sous forme d'expressions régulières TREC10

Ce système a apporté une nouvelle méthode pour l'extraction de la réponse. À la TREC10, le système a utilisé des modèles d'extraction sous forme d'expressions régulières pour classer les réponses possibles. Cette méthode met fin à la méthode classique où le classement des réponses consistait généralement à trouver des entités nommées dans des blocs contenant les mots-clés de la question, puis à regarder les fréquences des réponses.

Malgré ses résultats relativement positifs, le système n'assure pas l'interaction avec l'usager et n'utilise pas l'approche multi-agent pour bien évaluer la réponse choisie.

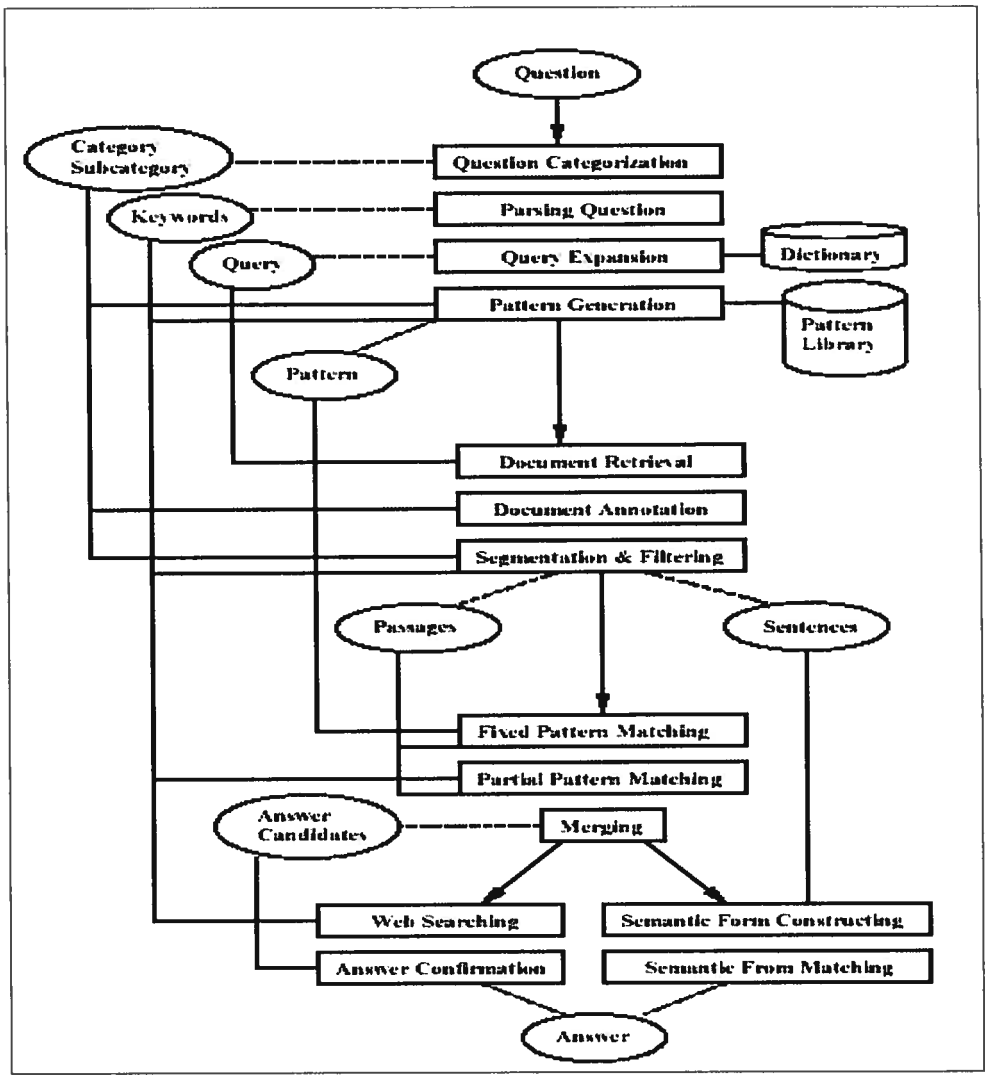


Figure 2.7: Architecture du système ILQUA

2.6 Évaluation et discussions

Nous avons présenté dans ce chapitre quelques architectures de systèmes question–réponse, bien que loin d'être exhaustives, elles sont représentatives. Nous avons évoqué également quelques caractéristiques principales jugées comme fondamentales par la communauté des chercheurs. Nous allons dans cette section discuter les différentes plateformes et présenter les aspects qui nous intéressent dans notre architecture. Nous allons également exposer les lacunes et insuffisances dans ces approches de développement.

Nous avons vu que le processus d'opération d'un système de question–réponse contient généralement : l'analyse de la question, l'indexation de la question, l'interrogation d'un moteur de recherche pour chercher les documents pertinents et l'extraction et la validation de la réponse.

Actuellement les trois premières étapes du processus ne posent pas de grandes difficultés du point de vue des techniques disponibles en recherche. Cependant, le dernier processus qui s'occupe de l'extraction et la validation de la réponse, reste le vrai défi pour le développement de ces systèmes. Dans le cas des systèmes qui ont été déjà présentés, et tous les systèmes de la TREC, le problème était toujours l'exactitude, la précision et la validation de la réponse. Malgré les efforts donnés par les spécialistes le taux d'exactitude des réponses reste loin d'être satisfaisant.

Les systèmes présentés tout au long du chapitre, possèdent tous des architectures sans module de spécification de la réponse. Ils fournissent aux usagers des réponses avec un taux d'erreur élevé et dans la plupart des cas une déclaration d'absence de réponse. À part le système Ask Jeeves, le reste des systèmes souffrent de l'absence d'un module qui commence la recherche en comparant la question à des requêtes similaires qui ont déjà été répondues sur une base de données locales.

Nous croyons que le manque de ces deux modules peut poser les problèmes suivants :

1. Perte de temps et manque de précision en cherchant directement la réponse sur le corpus sans passer par le modèle local de question–réponse.
2. Taux d'erreur et d'ambiguïté élevé à cause du manque d'un module d'interaction avec l'usager.

Par conséquent notre contribution, sera surtout basée sur les points suivants :

1. L'adoption d'une approche multi-agents qui va nous donner la possibilité de choisir la technique convenable à chaque type de question ce qui nous permet d'atteindre les performances souhaitées telles que l'obtention de la réponse en temps réel. Un agent est affecté à chaque type de question selon la classification des questions.
2. L'intégration d'un corpus local de données, du moment que 20% des questions existantes sur le Web comptent pour 80% des questions des usagers. Ce corpus va nous permettre d'augmenter le taux d'exactitude et d'avoir des réponses en temps réel.
3. Garantir une interaction minimum et suffisante avec l'usager. Cette interaction permettra de limiter le nombre de documents à parcourir ainsi que le nombre de passages à consulter et d'augmenter le taux de validation de la réponse et surtout de diminuer le taux d'erreur.
4. Nous jugeons qu'un module "Répertoriage automatique des questions sans réponse" qui classe automatiquement les questions des usagers qui n'ont pas de réponses, est indispensable pour enrichir la base de données locales de question–réponse.
5. Utiliser le raisonnement à base de cas pour calculer la similarité entre la question de l'usager et chaque document du corpus (Web et base de données locales).

Les systèmes de question–réponse ont montré leurs efficacités en exploitant la grande masse de données existante dans leurs bases de données et celle trouvée sur le Web comme démontré dans les exemples précédemment cités.

Parmi les domaines d'applications de ces systèmes nous voyons le commerce électronique (consultation d'une liste de produits), la documentation en ligne (interrogation de la documentation d'une bibliothèque), le Web sémantique, les systèmes tutoriels intelligents (eLearning). Dans ces derniers, les étudiants ont toujours besoin d'informations supplémentaires, pour compléter la matière du cours, ces informations supplémentaires peuvent être fournies grâce à un système de question–réponse.

Un système de question–réponse efficace et accessible aux clients sur le site Web de l'entreprise permettrait de réduire le volume de courriels.

Nous allons voir dans le prochain chapitre notre système QUERI. Nous ferons quelques rappels des concepts technologiques que nous allons utilisés dans notre système tels que le "Case-Based Reasoning"[1], les agents intelligents, les parseurs (analyseurs), WordNet, l'acquisition de connaissances, la recherche d'information, l'extraction d'information.

[1] Un système de question–réponse est considéré comme un CBR car il contient la description d'un problème (la question) et la description d'une solution (la réponse).

CHAPITRE 3

3. QUERI

3.1. Introduction

QUERI est un système de question–réponse qui entre dans le cadre des systèmes d'aide aux apprenants. Nous avons introduit dans le chapitre précédent quelques systèmes de question–réponse issus du domaine industriel et universitaire notamment celle des conférences TREC. Nous avons soulevé certaines insuffisances et limites des systèmes existants relativement aux exigences des systèmes d'aide aux apprenants.

Nous présentons les solutions que nous proposons pour pallier les problèmes précédemment cités. Nous avons opté pour une architecture orientée multi-agents et basée sur l'interaction système-usager.

La prochaine section du chapitre introduit le concept de notre architecture. Nous allons justifier comment une telle architecture peut répondre aux besoins spécifiés des systèmes d'aide aux apprenants. Nous allons montrer que le paradigme des "Services Web" est utilisé dans cette architecture, notamment pour implanter le module de recherche sur le Web. Nous allons passer rapidement à travers l'ensemble des technologies utilisées et présenter leurs concepts clés. Nous présentons l'architecture de QUERI. Ensuite nous passerons au concept d'aide aux apprenants et la nécessité de répondre à des besoins spécifiques des utilisateurs pour lesquels QUERI sera développé. Nous allons montrer que l'approche adoptée permettra de répondre à ces besoins.

3.2 Principe de conception

Peu de systèmes de question–réponse ont été développés dans le but satisfaire un domaine spécifique. Nous avons vu dans les conférences TREC que les systèmes ne sont pas limités à un seul domaine.

Dans le domaine des STI (Systèmes Tutoriels Intelligents) les étudiants ont toujours besoin d'informations supplémentaires pour compléter la matière du cours. D'où l'importance de tels systèmes de question–réponse comme un outil d'aide aux apprenants.

Les STI ont leur propre contexte d'utilisation. Généralement, ils sont orientés vers un domaine spécifique. Les systèmes de question–réponse, comme des outils d'aide aux apprenants, peuvent profiter de ces caractéristiques. Par conséquent l'interaction avec l'usager et la limitation du contexte général sont indispensables pour ces derniers.

QUERI a été conçue dans l'objectif de :

- créer un outil efficace d'aide à l'apprentissage sous forme d'un système de question–réponse orienté à un contexte (domaine) bien spécifié ;
- augmenter la précision (taux d'exactitude) des réponses produites et améliorer le temps de réponse du système en se limitant à un seul domaine (Exemple : un cours d'agents intelligents) par l'intégration d'un corpus local de données ;
- Adopter une approche multi-agents qui offre la possibilité de choisir la technique convenable à chaque type de question ;
- limiter –en cas de recherche sur le Web– le nombre de documents à parcourir, limiter le nombre de passages à consulter et augmenter le taux de validation de la réponse avec une simple interaction avec l'usager ;
- traiter les questions sans réponse par un module "Répertoriage automatique des questions sans réponse" qui classe automatiquement les questions des usagers qui n'ont pas de réponses pour enrichir le corpus local de questions-réponses, un expert ou un administrateur consulte les questions et donne des réponses à celles qui sont importantes ;
- utiliser le raisonnement à base de cas pour calculer la similarité entre la question de l'usager et chaque document du corpus.

L'exploitation de l'expertise des autres chercheurs et industriels ainsi que la conformité aux standards et aux spécifications sont des principes fondamentaux qui ont été grandement employés lors de la conception de QUERI.

Dans la section suivante, nous allons présenter les différentes technologies utilisées lors de la conception et l'implémentation de QUERI. Nous allons commencer d'abord par l'explication du concept des technologies multi-agents et "Case-Based Reasoning", les patrons d'extraction, la recherche d'information et les moteurs de recherche, puis nous allons voir XML et XSLT, les deux formalismes de base utilisés pour la réalisation de notre interface interactive. Le concept des services Web a été utilisé en ASP.NET comme module de recherche sur le Web.
Un peu plus loin, nous présentons les réseaux sémantiques tels que WordNet et leurs utilisations pour ajouter la sémantique aux systèmes.

3.3 Présentation des technologies utilisées

Dans cette section du chapitre, nous allons passer à travers les différentes technologies utilisées dans l'architecture de QUERI.

3.3.1 Raisonnement à base de cas

3.3.1.1 Introduction

Le raisonnement à base de cas (CBR) est une méthode de résolution de problèmes qui utilise des expériences passées pour résoudre de nouveaux problèmes [Gresse *et al.*, 2001]. L'ensemble des expériences passées forme une banque de cas.

La technique de raisonnement à base de cas peut être expliquée de la façon suivante : dans le but de trouver une solution à un problème courant, il suffit de chercher un problème similaire dans une base de cas, de prendre la solution du problème passé et de l'utiliser comme point de départ pour trouver la solution au problème actuel.

Le processus CBR proposé par Aamodt et Plaza est un processus cyclique qui peut se décomposer en quatre phases :

1. *La recherche* : cette phase permet de rechercher les cas de la banque les plus similaires avec le nouveau problème à résoudre.

2. *L'adaptation* : cette phase consiste à reprendre la solution du cas sélectionné dans la phase précédente et de l'adapter si nécessaire au nouveau problème à résoudre. On obtient une solution possible pour résoudre le problème.
3. *La révision* : la solution proposée est testée dans l'environnement du problème initial et évaluée. Il s'agit de vérifier si la solution proposée peut résoudre le problème initial ou si cela nécessite de nouvelles modifications.
4. *La maintenance* : cette nouvelle expérience est ensuite mémorisée dans la base de cas en vue d'être utilisée dans de futures résolutions de problèmes [Aamodt *et al.*, 1994]

Le cœur de la technique de raisonnement à base de cas est le principe de similarité, utilisé lors de la première phase du cycle. En effet, la définition d'une mesure de similarité est très importante pour retrouver les cas les plus adéquats permettant de résoudre le nouveau problème. La mesure de similarité dépend beaucoup de l'application et du but que l'on cherche à atteindre.

3.3.1.2 Raisonnement à base de cas et systèmes de question–réponse

Un système de question–réponse est considéré comme système de raisonnement à base de cas car il contient la description d'un problème (la question) et la description d'une solution (la réponse) [Gresse *et al.*, 2001].
La solution est construite en calculant la similarité entre le nouveau cas et ceux de la base de cas. Pour cela on utilise une fonction de similarité dont le but est de déterminer la similarité de deux cas dans la base en se basant sur leurs attributs. On peut donc, à partir de la description d'un cas, obtenir une liste ordonnée (en fonction de la valeur donnée par la fonction de similarité) des cas similaires.

3.3.1.3 La fonction de similarité

La métrique de similarité est une fonction qui prend deux entités (phrases en langue naturelle, image, parole, ...) en entrée et qui retourne une valeur

reflétant la similarité entre ces deux entités selon un certain but. Généralement, les chercheurs définissent la similarité entre deux cas comme une combinaison des similarités entre les différents attributs des cas.

Pour comparer deux cas en se basant sur leurs attributs, la fonction de similarité globale sera une fonction pondérée par les différentes similarités des attributs (les mots d'une phrase par exemple). La similarité entre deux cas ***q*** et ***c*** est : $S(q , c) = \sum w_i S_i(q_i, c_i)$

Où $q_1, \ldots q_n$ sont les attributs du cas ***q*** et $c_1,\ldots, c_n$ les attributs du cas ***c***. S_i sont les différentes similarités des attributs et w_i sont leurs poids associés.

Pour définir la fonction de similarité globale, nous devons d'abord définir les différentes similarités des attributs et leurs poids respectifs.

Une fois que les similarités entre le nouveau cas et ceux de la base de cas sont calculés, un ensemble des cas les plus similaires est choisi. Il y a deux méthodes différentes pour les sélectionner :

- sélectionner les cas dont la similarité dépasse un certain seuil ;
- sélectionner les **n** cas les plus similaires.

Dans le cas d'un système de question–réponse, nous utilisons la première méthode.

3.3.2 Approche multi-agents

3.3.2.1 Introduction

Les agentsintelligents sont considérés comme un nouveau modèle informatique des systèmes de calcul complexes, ouverts et hétérogènes [Ferber *et al.*, 1995]. Dans un système multi-agents, les composants logiciels agissent plutôt comme des entités individuelles indépendantes, au lieu d'être uniquement des composantes du système. Les systèmes multi-agents comprennent des idées et techniques venant de plusieurs disciplines : intelligence artificielle, sociologie, sciences de la gestion, etc.

Dans ce qui suit, nous allons donner quelques notions sur les agents et systèmes multi-agents.

3.3.2.2 Agents intelligents

Il est difficile de donner une seule définition pour les agents intelligents. Nous allons par contre en présenter quelques-unes que nous avons jugées plus importantes et qui correspondent à notre vision du système QUERI.

- un agent est une entité qui perçoit son environnement et agit sur celui-ci [Russell *et al.*, 1997] ;
- les agents intelligents sont des entités logicielles qui réalisent des opérations à la place d'un utilisateur ou d'un autre programme, avec une sorte d'indépendance ou d'autonomie, et pour faire cela ils utilisent une sorte de connaissance ou de représentation des buts ou des désirs de l'utilisateur [Crabtree, 1997] ;
- un agent est une entité autonome, réelle ou abstraite, qui est capable d'agir sur elle-même et sur son environnement et qui, dans un univers multi-agents, peut communiquer avec d'autres agents et dont le comportement est la conséquence de ses observations, de ses connaissances et des interactions avec les autres agents [Ferber *et al.*, 95].

Dans le cadre de notre travail, nous gardons la définition de [Ferber *et al.*, 95], car elle mentionne explicitement la notion d'agent interactif.

3.3.2.3 Les caractéristiques d'un agent intelligent

Un agent intelligent peut être caractérisé par [Wooldridge *et al.*, 1995] :

- **situé :** l'agent est capable d'agir sur son environnement à partir des entrées sensorielles qu'il reçoit de cet environnement ;
- **autonome :** l'agent est capable d'agir sans l'intervention d'un tiers (humain ou agent) et contrôle ses propres actions ainsi que son état interne ;
- **proactif :** l'agent exhibe un comportement proactif et opportuniste, en étant capable de prendre l'initiative au bon moment ;

• **capable de répondre à temps :** l'agent doit être capable de percevoir son environnement et d'élaborer une réponse dans le temps requis.

• **social :** l'agent doit être capable d'interagir avec des autres agents (logiciels ou humains) afin d'accomplir des tâches ou aider ces agents à accomplir les leurs.

3.3.2.4 Systèmes multi-agents

Les systèmes multi-agents produisent une solution à un problème donné, en révélant la possibilité de représenter les individus, leurs comportements et leurs interactions. La simulation multi-agents est fondée sur l'idée qu'il est possible de représenter sous forme informatique des comportements individuels des entités dont les interactions font apparaître des phénomènes nouveaux.

Nous citons comme exemple l'architecture des agents cognitifs [Frasson *et al.* 1996].

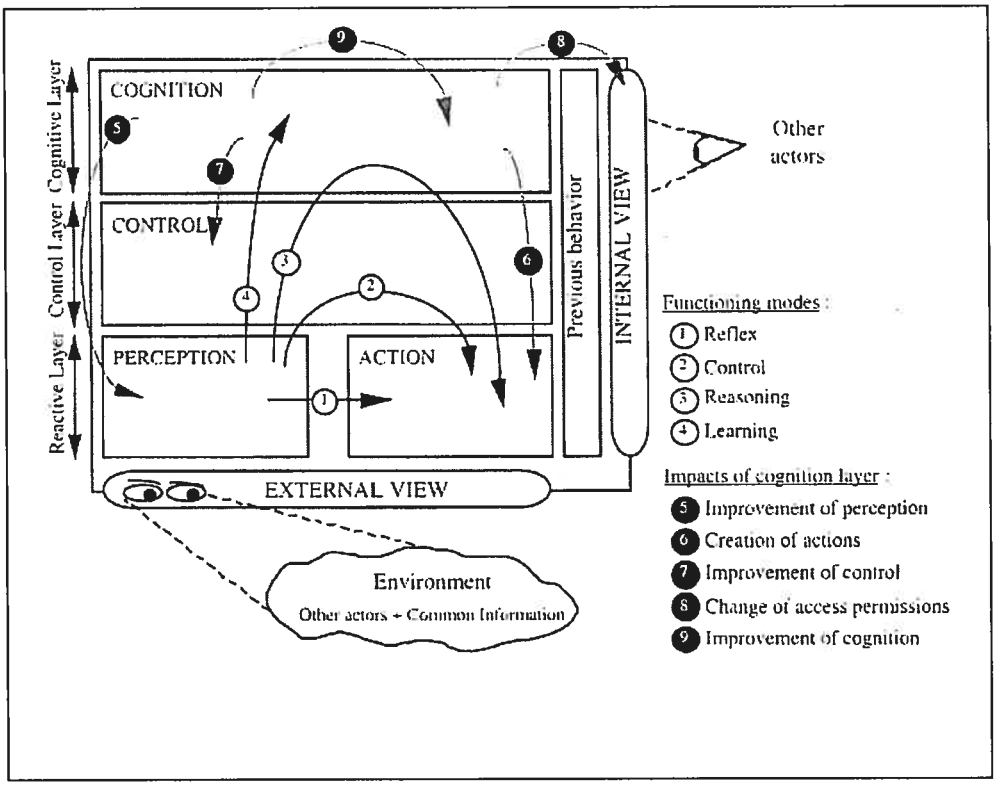


Figure 3.1 : architecture des agents cognitifs.

Les agents cognitifs sont des agents intelligents qui peuvent tenir compte de leur passé, qui ont en général une représentation précise de leur environnement et qui ont un mode "social" de planification et d'engagement. De plus ces agents sont dits intentionnels car ils ont des buts [Ferber *et al.*, 1995].

L'autre catégorie des agents est celle des agents réactifs. Ils n'ont pas de représentation de leur environnement, de leur monde si ce n'est que très partiellement. Les actions consistent à réagir à des informations leur parvenant. L'interaction de plusieurs agents permet l'émergence d'une intelligence [Bonabeau *et al.*, 1994]. Dans notre approche, nous utilisons le principe de collaboration entre agents réactifs pour élaborer notre extracteur de réponse que nous détaillons dans le chapitre suivant.

3.3.3 Patrons d'extraction

La méthode des patrons d'extraction a été utilisée pour la première fois dans la TREC10 par le système ExactAnswer "Modèles d'extraction sous forme d'expressions régulières" [Soubbotin *et al.*, 2001]. Cette méthode est basée sur des informations regroupées dans un patron d'extraction qui est formé d'un ensemble de règles. Ces règles comportent des patrons syntaxiques et des relations sémantiques. Le rôle des patrons est de déterminer la réponse dans une phrase candidate tandis que celui des relations sémantiques est de valider la réponse trouvée. Les contraintes sémantiques sont analysées par WordNet.

Nous donnons ci-dessous un exemple d'un patron d'extraction.

- question : Quelle est la population du Canada ?
- type de la question : qNombre
- patron d'extraction de la réponse : rNombre
  - o patrons syntaxiques :

    <Réponse>, <Pays> < population | nombre>

    < Réponse > (<Pays> < population>)
  - o axes sémantiques :

    Population (type de la réponse) , Canada (entité nommée)

3.3.4 Recherche d'information et moteurs de recherche

Les systèmes de question–réponse utilisent des techniques de recherche d'information.

Les moteurs de recherche constituent un cas pratique des systèmes de recherche d'information.

Un moteur de recherche (par exemple, *Google*[1], *Yahoo*[2], *Altavista*[3]) est un système qui permet de trouver des documents pertinents par rapport à une requête d'un utilisateur, à partir d'une base de documents volumineuse.

Pour trouver les documents pertinents à la requête de l'usager, nous avons deux approches principales sont :

- parcourir les documents d'une façon séquentielle, en les comparant avec la requête ;
- construire une structure d'index qui permet de trouver très rapidement les documents incluant des mots demandés, ou chaque mot est mis en correspondance avec les documents qui le contiennent : `Mot → {Doc1, Doc2, Doc3, …}`

Le modèle d'espace vectoriel "*Vector Space Modele*" (VSM) reste l'outil le plus utilisé. Dans ce modèle, un espace à N dimensions est utilisé où N est le nombre d'attribut de la requête.

La première étape consiste à créer un index inversé qui contient pour chaque terme du langage les documents qui l'utilisent.

Une fois l'index construit, le modèle calcule pour chaque terme :

- son **DF :** fréquence dans les documents ;
- son **IDF :** fréquence dans les documents inversée
  IDF = log(nb_total_de_documents/DF) ;
- pour chaque document, la **TF** : fréquence de ce terme dans ce document.

---

[1] http://www.google.com/
[2] http://www.yahoo.com/
[3] http://www.altavista.com/

Une fois cela effectué, un vecteur est calculé pour chaque document de la manière suivante :

- pour i allant de 0 au nombre de termes dans le langage : **dimension i = TF (terme i dans ce document) * IDF(terme i)** ;
- calcule **n** la norme du vecteur de ce document ;
- divise toutes les dimensions du document par **n.**

Ainsi pour chaque document il y a un vecteur, chacune des dimensions contenant un score pour le terme correspondant. La même méthode est appliquée pour la requête de l'utilisateur.

Pour comparer un document à une requête, la méthode des cosinus est appliquée pour mesurer la similarité. On calcule un produit scalaire entre les deux vecteurs, ce qui donne un score entre 0 (pas du tout reliés0) et 1 (textes semblables).

Pour un vecteur Q (question de l'usager), et un vecteur D (corpus de document), tel que $\overrightarrow{\mathbf{Q}} = (q1, \ldots, qn)$ et $\overrightarrow{\mathbf{D}} = (d1, \ldots, dn)$, la similarité est :

$$sim\ (\overrightarrow{Q}, \overrightarrow{D}) = \frac{\sum_{i=1}^{n} q_i d_i}{\sqrt{\sum_{i=1}^{n} q_n^2 \sum_{i=1}^{n} d_n^2}}$$

Un moteur de recherche est un élément de base dans l'architecture générale des systèmes de question–réponse. Parmi les moteurs de recherche les plus utilisés par ces derniers, nous avons Okapi, Google et Lemur.

3.3.5 Les Services Web, XML et XSLT

Un Service Web est une application Web sans interface HTML. Elle contient un ensemble de méthodes publiées sur l'internet. L'interface est décrite en WSDL et les méthodes sont accessibles en utilisant des messages XML (SOAP).

Les services Web sont des fonctionnalités de traitement disponibles sur un serveur Web. Les services Web établissent un paradigme simple et dimensionnel pour les traitements locaux et distants avec des implications compréhensives dans le développement du logiciel [Mignerat *et al.*, 2002].

Un ensemble de technologies standardisées forment le concept du service Web.

3.3.5.1 XML (eXtensible Markup Language)

Avec l'apparition de la notion du Web sémantique et la standardisation du Web par l'ISO (*International Organization for Standardization*), XML s'est imposé comme l'outil le plus important.

XML est un méta-langage : un ensemble de règles qui permettent de créer d'autres langages qui ont pour but de créer des documents « universels » respectant la même structure et pouvant ainsi être facilement échangeables.

Un document XML est un document structuré contenant un élément racine délimité par des balises début et fin. Chaque élément peut contenir des éléments fils ou des attributs.

3.3.5.2 XSLT (Extensible Style Language Transformations)

XSLT est un langage destiné à transformer un document XML en autre format, comme HTML, du texte pur, ou du Rich Text Format...

XSLT est un langage déclaratif et non procédural, il ne spécifie pas les algorithmes, il déclare les balises à remplacer, par exemple : toute balise <para> présente dans le XML source est à remplacer dans le HTML cible par une balise <p>.

A côté de sa syntaxe propre, XSLT fait aussi appel à un second langage, déclaratif lui aussi : XPath. XPath sert à spécifier des chemins de localisation à l'intérieur d'un arbre XML.

3.4 Approche proposée

Le système que nous proposons se scinde sur les modules suivants :

- l'interface graphique qui assure l'interaction avec l'usager ;
- le module d'analyse des questions ;
- le processus de traitement ;
- le moteur de recherche (extraction de documents pertinents et passages) ;
- l'extracteur de réponse ;
- le corpus de données (base de données de questions-réponses et Web).

3.4.1 Architecture QUERI

La figure suivante illustre la disposition des différents composants du système. L'usager communique sa question en langue naturelle via l'interface interactive. La première étape est l'analyse de la question. Après avoir transformé la question de l'usager sous forme de requête, une seconde étape consiste à la recherche de la réponse dans la base de données locale. Si le système ne trouve pas de réponses dans la base de données locale, un processus de recherche se déclenche afin de trouver la réponse pertinente sur le Web. La troisième étape est l'extraction de la réponse. Selon le type de question le système peut affecter un agent intelligent qui correspond à la méthode d'extraction la plus convenable telle que : la recherche sémantique, la recherche statistique, la recherche par mots clés, …

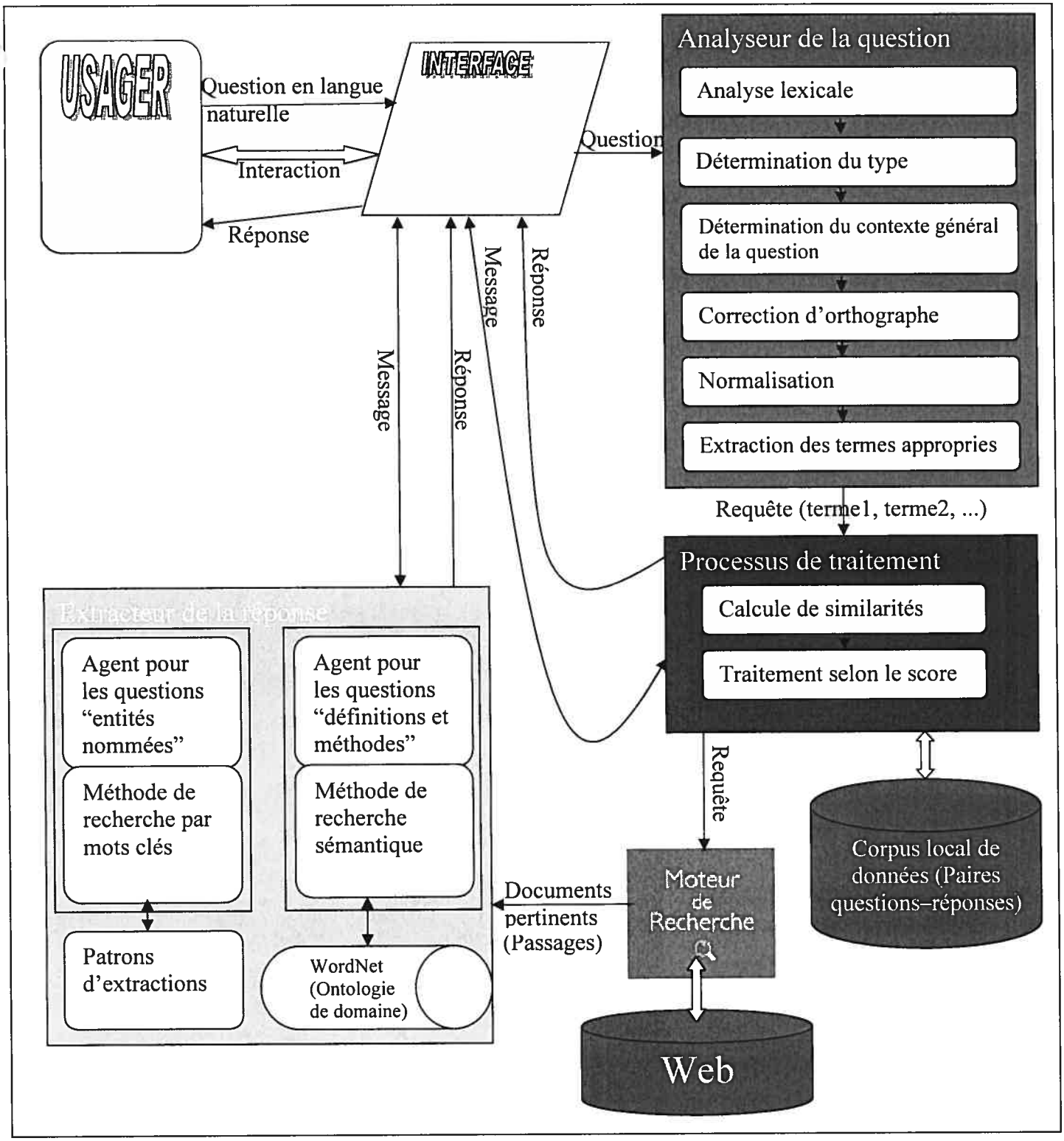


Figure 3.2: Architecture de QUERI

3.4.2 Interface interactive

L'interface interactive (interaction usager–système) constitue un élément de base de notre système. À travers cette interface, l'usager peut clarifier sa question, dans le cas d'ambiguïté ou si cette dernière n'atteint pas le seuil minimal de certitude pour qu'elle soit validée par le système.

Nous croyons qu'une structure XML des menus permet une manipulation efficace et facile (multi langues, manipulation automatique des composants de l'interface) de l'interaction système-usager. Cette structure nous offre une interface multi-langue très fiable et nous permet de manipuler de façon automatique ses composants, grâce aux feuilles de style XSLT.

Nous avons défini notre propre format d'interface graphique (voir la figure suivante).

```
<field type="buttons" visible="true">
    <button name="confirmei" method="confirm()" >
        <langstring xml:lang="en">Yes</langstring>
        <langstring xml:lang="fr">Oui</langstring>
    </button>
    <button name="Noi" method="ref()" >
        <langstring xml:lang="en">No</langstring>
        <langstring xml:lang="fr">Non</langstring>
    </button>
</field>


<field type="radiobox" name="choice" visible="false" mode="single"
    <choices xml:lang="en">
        <choice id="a1">01</choice>
        <choice id="a2">02</choice>
        <choice id="a3">03</choice>
    </choices>
    <choices xml:lang="fr">
        <choice id="r1">01</choice>
        <choice id="r2">02</choice>
        <choice id="r3">03</choice>
    </choices>

    <comments>
        <langstring xml:lang="en" />
        <langstring xml:lang="fr" />
    </comments>
</field>
```

Figure 3.3: Un exemple de description XML

Nous allons détailler dans le chapitre suivant, l'implémentation de notre interface et les pages Web caractérisées par les descriptions XML et les feuilles de style XSLT qui permettent de générer les présentations correspondantes.

3.4.3 Analyseur de question

L'usager communique sa question en langue naturelle via l'interface interactive. La première étape est l'analyse de la question. Celle-ci consiste à :

- segmenter la question de l'usager en unités lexicales ;
- déterminer le type de la question. Deux types sont possibles :
  - questions qui demandent des entités nommées ou des réponses simples (oui, non) ;
  - questions qui demandent des définitions, des méthodes …

  Pour classer les questions, on se base sur le mot question (*How, What, Where* …), et la classification de Moldovan [Moldovan *et al.*, 1999] ;
- déterminer le contexte général de la question : c'est une étape préliminaire de l'analyse sémantique, laquelle est basée sur une hiérarchie que nous avons préalablement établie. Dans cette hiérarchie : *sujet* ➔ *contexte* ➔ *mot-clé* nous comparons le contexte général du corpus local avec celui de la question. La figure suivante, montre un exemple de cette hiérarchie

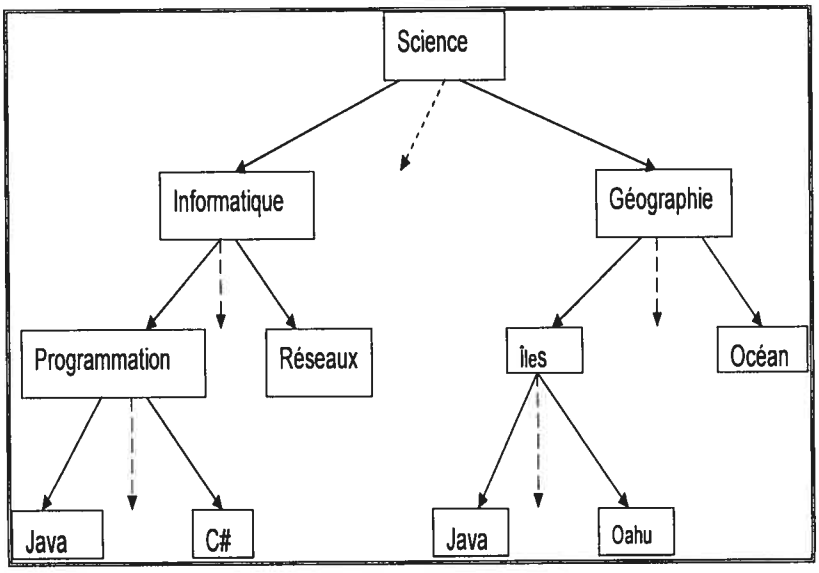


Figure 3.4: Exemple du concept « Hiérarchie de sujets »

- vérifier et corriger l'orthographe ;
- normaliser (par exemple en prenant juste la racine pour un verbe régulier) ;
- extraire les termes appropriés (éliminer les termes inutiles pour la construction de la requête, comme : le, de, *the, an,...)* ;
- générer une requête à partir de la question.

3.4.4 Processus de traitement

Dans notre approche un modèle conversationnel de raisonnement à base de cas est utilisé. Ce modèle définit le problème à résoudre en interrogeant l'usager du système (d'où le terme de « conversation »). Ainsi, le système sélectionne les solutions les plus appropriées.

Le schéma de résolution de problème pour le modèle conversationnel de raisonnement à base de cas est basé sur le principe suivant : il s'agit de connaître progressivement les aspects et caractéristiques du problème à résoudre par interaction avec l'usager. Ainsi, les attributs permettant la recherche sont trouvés en collaboration avec l'usager et permettent de trouver les solutions appropriées.

Par conséquent, notre processus de traitement est basé sur le calcul de similarité entre la question de l'usager et chaque question du corpus local des paires (question, réponse). Pour calculer la similarité nous utilisons la méthode des cosinus déjà présentée, où la question de l'usager et chaque question du corpus sont représentées par des vecteurs.

En premier ordre on calcule la similarité du type, en se basant sur le mot question (*What, Why, How, ..*) et le contexte général (corpus et question de l'usager).

➢ **Similarité du type**

$$SimType(user\_q, q_k) = \begin{cases} 1.0 & \text{Si type user\_q = type } q_k \\ 0 & \text{Sinon} \end{cases}$$

user_q : Question de l'usager

q_k : Question numéro **k,** de la base de données

Le processus de traitement calcule ensuite, la similarité locale entre chaque terme de la question de l'usager et les termes de chaque question du corpus.

➢ **Similarité locale**

$$SimLoc(user\_q_i , q_k) = \mathbf{max}\ SimLoc_J(user\_q_i , q_{kJ}) \qquad 0 <= J <= m$$

user_q$_i$: index (terme) numéro **i** de la question de l'usager

q_{kJ} : index numéro **J** de la question q_k

m : Est le nombre total des index (termes appropriés) de la question q_k

La similarité locale est la valeur maximale entre quatre valeurs déjà prédéfinies.

Ces valeurs et les seuils de l'algorithme (scores de similarités), ont été choisit selon nos tests d'une part et des travaux précédents [von Wangenheim *et al.*, 2001], [Tan *et al.*, 2004], [Sutcliffe *et al.*, 2004] d'autre part.

$$SimLoc_J (user\_q_i , q_{kJ}) = \begin{cases} \mathbf{1.0} & \text{Si user\_q}_i \text{ est identique à } q_{kJ} \\ \mathbf{0.9} & \text{Si ((user\_q}_i \text{ Suffixe/Préfixe de } q_{kJ} \text{) Ou (} q_{kJ} \text{ S/P de user\_q}_i \text{))} \\ & \quad \text{et (|user\_q}_i - q_{kJ} \text{ | <=2)} \\ \mathbf{0.6} & \text{Si ((user\_q}_i \text{ Contient } q_{kJ} \text{)} \\ & \text{Ou } (q_{kJ} \text{ Contient user\_qi)) et (|user\_q}_i - q_{kJ} \text{ | <=2)} \\ \mathbf{0} & \text{Sinon} \end{cases}$$

➢ **Similarité globale**

Une fois que la similarité locale et celle de type calculées, on calcule la similarité globale entre la question de l'usager et chaque question du corpus. Cette dernière est la somme des similarités locales (similarités entre chaque terme de la question de l'usager et les termes de la question **k** du corpus local) avec la similarité du type sur le nombre de termes appropries de la question **k** plus **1** [von Wangenheim *et al.*, 2001].

$$\text{SimGlob}(\text{user\_q}, q_k) = \frac{\left[\sum_{(i=1)}^{\mathbf{n}} \text{simLoc}(\text{user\_q}_i, q_k)\right] + \text{simType}(\text{user\_q}, q_k)}{n+1},$$

où **n** est le nombre de termes de la question **k.**

➢ **Similarité détaillée**

Lorsque plusieurs questions du corpus possèdent le même score de similarité avec la question de l'usager, un autre type de similarité est calculé en considérant le nombre de termes de la question de l'usager [von Wangenheim *et al.*, 2001].

$$\text{SimDéta}(\text{user\_q}, q_k) = \frac{\left[\sum_{(i=1)}^{\mathbf{n}} \text{simLoc}(\text{user\_q}_i, q_k)\right] + \text{simType}(\text{user\_q}, q_k)}{\frac{n+m}{2} + 1},$$

où **m** est le nombre de termes de la question de l'usager.

Après le calcul des similarités, les étapes suivantes sont à réaliser :

- Mettre les similarités (globales et détaillées) dans deux tableaux différents.
- Faire le tri des tableaux en ordre décroissant avec la méthode **mergesSort**.
- **Si** le tableau des similarités globales contient une **et une seule** valeur maximale :
  - **Si** (valeur maximale >= 70%) **alors** on affiche la réponse adéquate
  - **Sinon Si** (valeur maximale >= 50%) **alors** demande la confirmation de l'usager

    Sinon demande à l'usager de mieux formuler sa question
- **Sinon Si** le tableau des similarités détaillées contient une **et une seule** valeur maximale :

  (Faire comme dans le cas des similarités globales)

  Sinon Propose les trois premières questions du tableau des similarités détaillées.

Cette démarche, basée sur un modèle conversationnel de raisonnement à base de cas et sur le calcul de similarité avec la méthode des cosinus permet d'obtenir d'excellents résultats. Une description détaillée de ces résultats sera présentée dans le chapitre suivant.

3.4.5 Base de données relationnelle, serveur et service Web

Dans QUERI, comme dans tout autre logiciel, le stockage de l'information se fait dans des bases de données. Nous avons opté pour une base de données relationnelle. Les bases de données relationnelles ont prouvé leur efficacité, fiabilité et performance durant les dernières années.

QUERI sera utilisé par un nombre d'utilisateurs qu'on ne peut prévoir à l'avance et qui peuvent accéder de façon concurrente aux données (couples question/réponse).

La base de données que nous avons utilisée est de type *Access*[1].

---

1 http://www.bcschools.net/staff/AccessHelp.htm

Comme QUERI est un outil d'aide aux apprenants, il doit supporter Le passage à l'échelle *(scalability)*. Il doit avoir la capacité de supporter une augmentation de ses contraintes dans un domaine particulier. Par exemple, en terme de nombre d'utilisateurs, ses performances de fonctionnement doivent être aussi bonnes avec une dizaine d'utilisateurs connectés qu'avec une centaine. De même, il doit supporter une base de données gigantesque.

Ce passage à l'échelle, doit être supportée par un minimum d'infrastructure de développement et de déploiement. Notamment le serveur Web qui héberge, le service Web du moteur de recherche (*Google*) et la base de données.

IIS *(Internet Information Services)* muni de *ASP.NET* de Microsoft a été utilisé pour le déploiement de l'application.

Voici les critères utilisés pour évaluer le stockage de données et une meilleure qualité d'accès.

- dimensionnel (la mise en échelle) ;
- temps de réponse aux requêtes ;
- sécurité.

Nous allons détailler dans le chapitre suivant l'implémentation du service Web du moteur de recherche (*Google*), la construction et la conception de la base de données relationnelles et les caractéristiques du serveur IIS.

3.4.6 Extracteur de réponse

Le principe de ce module est la collaboration entre agents. Actuellement, il y a deux approches multi-agents pour les systèmes de question–réponse collaboratifs. La première utilise différents agents, chacun pouvant répondre à un type de question [Chu-Carroll *et al.*, 2003 et 2004]. La deuxième utilise deux agents de recherche indépendants, l'un destiné à rechercher des réponses sur le Web et l'autre sur le corpus local de documents [Chalendar *et al.*, 2002].

Dans la présente étude, nous avons adopté la première approche. Cette approche va permettre d'utiliser des outils qui vont aider à l'extraction de réponses comme WordNet, patrons d'extraction (pour classer les réponses possibles) et autres. La possibilité de choisir la technique convenable à chaque type de question permet d'atteindre les performances souhaitées telles que l'obtention de la réponse en temps réel. Selon la classification des questions, l'extracteur de réponse affecte un agent à chaque type de question. Les agents que nous avons utilisés sont les agents réactifs. Les agents réactifs sont souvent qualifiés de ne pas être "intelligents" par eux-mêmes. Ils sont des composantes très simples qui perçoivent l'environnement et sont capables d'agir sur celui-ci. Ils n'ont pas une représentation symbolique de l'environnement ou des connaissances et ils ne possèdent pas de croyances ni de mécanismes d'envoi de messages. Leurs capacités répondent uniquement au mode stimulus/action qui peut être considéré comme une forme de communication. Un système multi-agents constitué d'agents réactifs présente un comportement global intelligent.

Les agents réactifs sont considérés intelligents au niveau du groupe, du système. En conséquence, l'intelligence est distribuée entre les agents réactifs et le comportement intelligent devrait émerger de l'interaction entre ces agents réactifs et l'environnement [Wooldridge *et al.*, 1995].

Le noyau du système multi-agents se compose de deux agents principaux, le premier réagit lorsqu'il s'agit d'une question qui demande une entité nommée comme réponse. Cet agent utilise la méthode de modèles d'extraction pour l'extraction de la réponse. Il fait correspondre des types à des patrons :

À quel endroit → Lieu

Qui → Nom propre

Nous avons utilisé la technique ***collocation*** du TALN, pour construire les patrons utilisés par l'agent. Plus précisément, nous avons adopté le classement de Clas (1994) qui classe les collocations lexicales en divers groupes basés sur une fonction syntagmatique qui intègre les six catégories suivantes :

1. Verbe et nom.
2. Nom et adjectif.
3. Adverbe et adjectif.
4. Verbe et adverbe.
5. Nom (sujet) et verbe.
6. Marquage de la quantité (unité ou collectif) du nom.

Nous citons quelques exemples des patrons d'extractions que nous avons construit :

<NOM> est né à<RÉPONSE>

<NOM> est né en <RÉPONSE>

<NOM> est né le <RÉPONSE >

<RÉPONSE> a été inventé par <NOM>

<RÉPONSE> a inventé <NOM>

<RÉPONSE> découvert < NOM>

< NOM > a été découvert par<RÉPONSE>

Le <NOM> dans <RÉPONSE>

<NOM> dans <RÉPONSE> Près

Le deuxième agent réactif s'occupe des questions qui demandent des définitions, des méthodes ou des listes. Il utilise comme outil d'extraction une ontologie de domaines ou le thésaurus WordNet.

Il est important de garder une trace des lexicalisations des concepts de la question pour pouvoir les réutiliser dans la construction (extraction) de réponses et aussi utiliser des synonymes, hyponymes, etc. Ainsi, dans le système, chaque mot de la question est associé à un concept intégré dans l'ontologie.

Dans le cas de WordNet, la relation sémantique de base entre les mots est la synonymie.

Les synsets sont liés par les relations telles que est-un (is-a) partie-tout (part-whole).

Parmi les relations sémantiques supportées dans WordNet on retrouve :

- relation synonymie : le synset représente un ensemble de mots qui sont interchangeables dans un seul contexte. Par exemple un ordinateur est une machine.
- relation méronymie : le nom d'une partie constituante, "substance de" ou "membre" d'une autre classe. X est un méronyme de Y si X est une partie de Y. exemple : {ordinateur} Has_meronym {{clavier}, {écran}} ; {clavier} Has_meronym {{touches}, {câble} [Fellbaum *et al.*, 1998]

Considérant un utilisateur qui demande la définition d'un IHM. Si la recherche est faite juste par mots clés et si on suppose que le mot IHM n'existe ni dans le corpus local ni sur le WEB, le système n'affichera pas de réponse. Tandis qu'avec WordNet ou avec une ontologie construite spécialement pour un domaine spécifique, le système génère une autre requête avec un synset de IHM qui est *Interface Homme Machine*, et l'usager aura la bonne réponse.

L'agent peut utiliser le thésaurus comme une base de connaissances. Pour la question : Qui est Ridley Scott? Et grâce au lien *est-un,* l'agent affiche que Ridley Scott est un réalisateur.

L'agent apporte une aide pour filtrer les réponses possibles. Pour la question : Qui est le réalisateur de film *Kingdom of Heaven*? Si le système hésite entre deux réponses possibles, Orlando Bloom et Ridley Scott, l'agent peut détecter grâce au thesaurus que le premier est un acteur (réponse non valide) alors que le deuxième est bien un réalisateur (réponse valide).

3.5 Conclusion

Dans ce chapitre, nous avons présenté l'architecture de QUERI, un système de question–réponse d'aide aux apprenants. L'objectif principal de notre approche est de réaliser un système qui peut répondre à des questions posées en langage naturel à partir d'un corpus de données. Notre système devrait être réutilisable, c'est à dire qu'on peut l'intégrer dans différents systèmes tutoriels en respectant le contexte général de chaque système. De même, il doit avoir la capacité de supporter une augmentation de ses contraintes dans un domaine particulier (nombre d'utilisateurs, taille de corpus de données, etc.)

Le système devrait aussi pouvoir :

- justifier la réponse et évaluer l'adéquation de la réponse à la question ;
- assurer l'interaction avec l'usager, ce qui aide à lever l'ambiguïté des questions ;
- prendre en compte le contexte général de la question ;
- extraire les bonnes réponses ;
- reformuler la réponse dans le cas de fusion de documents ;
- répondre en temps réel à la question de l'usager.

Afin d'atteindre les objectifs mentionnés précédemment, nous avons eu recours à diverses techniques pour améliorer l'ensemble du processus.

L'approche multi-agents, les patrons d'extractions et les ontologies pour l'extraction de réponse. La recherche d'information, particulièrement le service Web *"Google"* pour la recherche des documents (passages) pertinents. Le Traitement Automatique de la Langue Naturelle (TALN) est présent par l'analyse lexical et la détermination du type de la question. Le Raisonnement à base de cas "Case-Based Reasoning" dans sa forme conversationnelle a été utilisé avec la méthode des cosinus pour le calcul de similarité.

Avec ce troisième chapitre, nous arrivons à la fin de notre approche et nos solutions logicielles pour la problématique des systèmes de question–réponse dans le cadre des systèmes d'apprentissage. Nous allons présenter dans le prochain chapitre l'implémentation des solutions, discuter l'architecture et proposer les futures perspectives de QUERI.

CHAPITRE 4

4. IMPLEMENTATION ET RESULTATS

4.1 Introduction

Dans les précédents chapitres, nous avons exploré le domaine de systèmes de question–réponse ainsi que celui de leurs applications, comme outils d'aide aux apprenants. Nous avons proposé une architecture multi-agents d'une plateforme collaborative destinée au domaine d'apprentissage ("*scalability*", réutilisation, conversation,...). Ce chapitre est consacré aux technologies utilisées dans l'implémentation de notre système. Pour illustrer notre système, nous présentons quelques scénarios d'exécution avec des prises d'écrans correspondants.

Nous finirons ce chapitre par une description des perspectives envisagées et une conclusion.

4.2 Validation des résultats

QUERI porte sur la dimension de recherche qui concerne la contribution scientifique dans les domaines de systèmes de question–réponse et d'outils d'aide aux apprenants par de nouvelles idées qui doivent être validées par la communauté des chercheurs. Il peut avoir un aspect commercial comme un outil supplémentaire sur le site d'une entreprise qui va minimiser d'une façon considérable le volume de courriels. La validation selon les normes du génie logiciel peut être réalisée par des audites et des présentations aux utilisateurs finaux.

La section 4.2 concerne donc la présentation des contributions du QUERI dans le domaine de recherche.

4.2.1 Publications de conférence

Nous avons écrit deux articles de recherche qui ont été acceptés dans des conférences internationales. Le premier a été accepté dans la colloque international, intitulée TICE 2004 "Technologies de l'Information et de la Connaissance dans l'Enseignement Supérieur et l'Industrie" à l'Université de Technologie de Compiègne en France[1].

Notre article intitulé, «QUERI : Un système de question–réponse collaboratif et interactif»[2], présente notre démarche pour renforcer l'efficacité des systèmes de questions-réponse et améliorer leurs performances en introduisant des techniques d'agents intelligents, de raisonnement à base de cas et de modèles d'extraction de réponses.

L'autre article a été accepté dans la 4e Colloque annuel DIVA'2005, "Développement, intégration et évaluation des technologies de formation et d'apprentissage" à l'école Polytechnique de Montréal[3].

L'article intitulé, «Un système de question–réponse collaboratif et interactif dans le cadre d'outils d'aide aux apprenants»[4], montre l'application des systèmes de question–réponse dans le domaine d'apprentissage et les techniques qui peuvent être utilisées pour améliorer cette application.

4.2.2 Validation avec des utilisateurs potentiels (Étudiants)

L'omniprésence des étudiants (utilisateurs finaux) dans toutes les phases du développement était l'un des facteurs les plus importants qui ont fait le succès du système QUERI (phase de teste). Les étudiants ont participé à l'élaboration de la spécification et à la phase de tests et de déploiement.

Nous allons nous concentrer sur l'évaluation du produit finale par la communauté étudiante. Nous avons invité une centaine d'étudiants de différentes universités (UdM, UQAM) à tester notre système afin de vérifier ses performances.

---

[1] http://www.utc.fr/tice2004/index2.htm
[2] http://archive-edutice.ccsd.cnrs.fr/docs/00/02/75/32/PDF/Merdaoui_Frasson.pdf
[3] http://www.polymtl.ca/carrefour/article.php?no=1940
[4] http://doe.concordia.ca/cslp/Downloads/PDF/programme-preliminaire-pres.pdf

Après les tests, les étudiants répondent à un questionnaire pour donner leur appréciation du système.

Le questionnaire contient deux sections principales : la première section interroge les étudiants sur leurs connaissances à propos des systèmes de question–réponse, leurs utilités et l'utilité d'un système de question–réponse avec les caractéristiques de QUERI tandis que dans la deuxième section les utilisateurs évaluent les performances de QUERI tel que l'adéquation de la réponse retournée à la question posée et la vitesse de réponse.

4.2.2.1 Évaluation générale

Nous avons posé des questions par rapport à la connaissance, l'utilisation et l'utilité des systèmes de question–réponse. Les résultats montrent que 31% de la communauté interrogée ont déjà entendu parler de systèmes de question–réponse. Donc, la majorité de 69%, n'a aucune idée sur ce genre de systèmes. Cela pourrait s'expliquer par l'indisponibilité des systèmes de question–réponse d'un coté (par rapport au moteurs de recherches) et la qualité des systèmes trouvés sur le Web d'un autre coté. Après une simple description des systèmes de question–réponse à notre communauté de test, 87% des étudiants croient qu'un système de question–réponse efficace et destiné à un domaine bien spécifique est un outil très important.

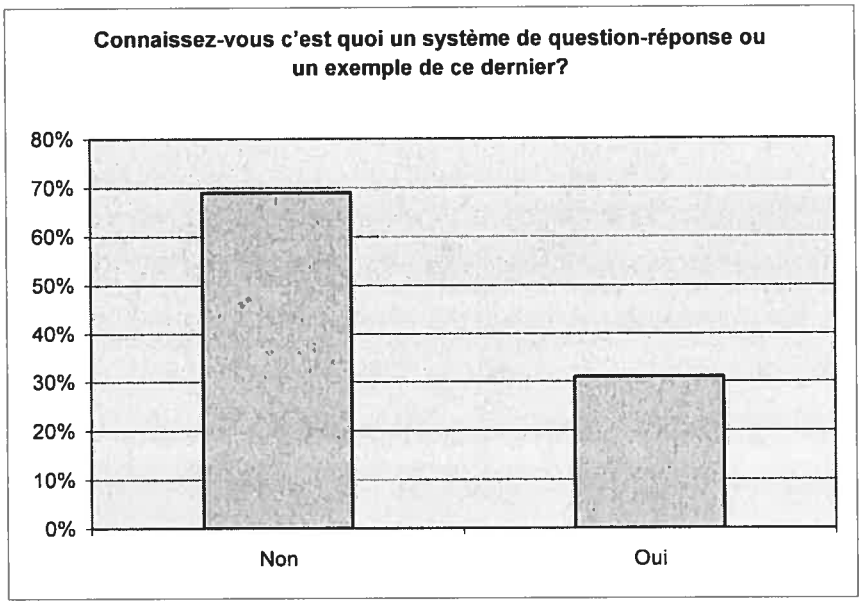


Figure 4.2.1 : Popularité des systèmes question–réponse

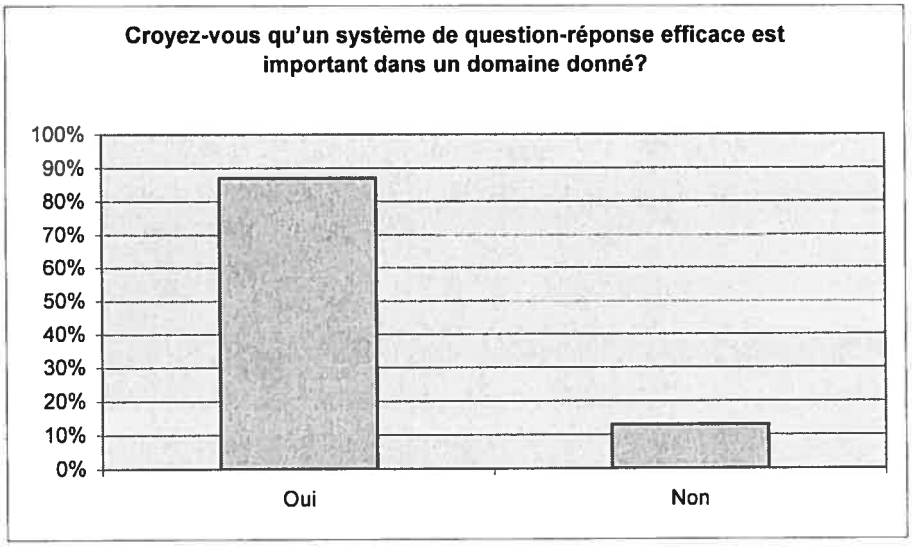


Figure 4.2.2 : Importance des systèmes question–réponse

Parmi les étudiants qui connaissent les systèmes de question–réponse, il n'y' en a aucun (0%) qui les utilise souvent, et un très faible pourcentage (7%) qui les utilise habituellement à cause de l'inefficacité des systèmes disponibles sur le Web.

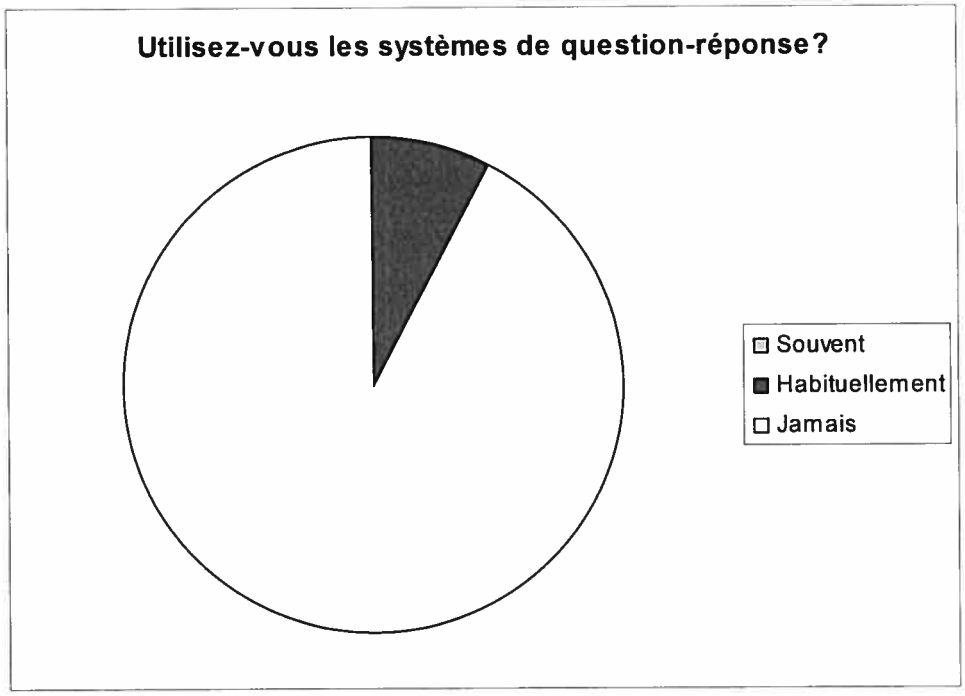


Figure 4.2.3 : Utilisation des systèmes de question

Après cette première phase de questions, nous avons constaté que la popularité des systèmes de question–réponse reste juste entre chercheurs. Plusieurs facteurs participent à cette limite, parmi lesquels nous pouvons citer l'indisponibilité des systèmes efficaces sur le Web, car la plupart des systèmes ne sont pas gratuits, ou ne sont pas disponible à cause de leur limite au domaine de recherche.

4.2.2.2 Évaluation des performances de QUERI

L'évaluation des performances du système compte sur les points suivants :

- l'utilité du système dans le domaine d'apprentissage ;
- l'exactitude de la réponse fournie à l'usager ;
- l'aspect d'interaction système usager ;
- le temps de réponse du système ;
- la convivialité de l'interface interactive.

Pour cette évaluation, nous avons demandé aux utilisateurs de noter entre 0 et 10 les énoncés qui expriment les points mentionnés ci-dessus. La note 10 représente une satisfaction totale tandis que la note 0 représente un désaccord total.

Un pourcentage de 84% des utilisateurs croit à l'utilité de système comme outil d'aide aux apprenants et que plus de 78% des étudiants ont eu des réponses exactes à leurs questions après que le contexte de recherche fût définit.

69% des étudiants pensent que l'interaction avec le système a beaucoup aidé pour trouver la bonne réponse et que l'interface est conviviale.

À propos du temps de réponse, plus de 96% des utilisateurs ont donné une note supérieure ou égale à 8 pour ce dernier.

Les résultats de tests sont prometteurs et le système continue à être enrichis pour répondre mieux aux attentes des utilisateurs finaux.

4.3 Implémentation de l'architecture QUERI

4.3.1. Langage de programmation

Le langage de programmation le plus souvent utilisé pour les services Web de la plateforme *.NET* est le langage *C#*. Le langage est basé sur une machine virtuelle, que *Microsoft* appelle CLR (*Common Language Runtime*).

Comme nous allons implémenter le module "Moteur de recherché" sous forme de service Web, nous avons choisit le langage C# pour la réalisation d'une deuxième Version de QUERI. Une première version a été implémentée avec le langage Java, Servlet, JSP, serveur Tomcat.

Les services Web sont implémentés en utilisant la technologie *ASP.NET*[1], la composante Web de la technologie *.NET*.

La connexion avec la base de données a été faite avec le serveur Jet de Microsoft et le driver OLEDB. L'application a été réalisée sur un environnement Windows XP.

4.3.2 Bibliothèques utilitaires et implémentation de l'interface

La disponibilité du code source pour certaines fonctionnalités de notre système, justifie la présence de tels codes dans nos développements. Par exemple, dans le cas de l'implémentation de notre interface, la librairie *MSXML 4.0* a été utilisée pour la manipulation des documents *XML* et *XSLT*. Cette librairie est inclue avec l'environnement de développement *Visual Studio .NET*.

QUERI utilise le composant *RAD Menu*[2], qui permet de générer la présentation *ASP* à partir d'une description XML. L'interface utilisateur est constituée d'un ensemble d'informations :

- la liste des pages Web ;
- le contenu de chaque page Web ;
- les menus qui permettent d'accéder à ces pages.

---

[1] www.asp.net

[2] www.telerik.com/Default.aspx

Une page Web est caractérisée par une description XML de son contenu ainsi qu'une feuille de style XSLT qui permet, lorsque invoquée par le moteur XSLT, de générer la présentation correspondante.

La figure 4.3.1 contient la description XML du contenu de la page alors que la figure 4.3.2 représente la feuille de style correspondante.

```
<?xml version="1.0" encoding="utf-8" ?>
<content usertable="user" sessiontable="session" authentification="au
    <title>
        <langstring xml:lang="en">Left frame </langstring>
        <langstring xml:lang="fr">Frame gauche</langstring>
    </title>

    <!-- Account Information -->

  <field type="buttons">
        <button name="welcome" method="welcome()">
            <langstring xml:lang="en">Welcome page</langstring>
            <langstring xml:lang="fr">Accueil</langstring>
        </button>
        <button name="Ask" method="ask()">
            <langstring xml:lang="en">Ask directly</langstring>
            <langstring xml:lang="fr">Pose ta question</langstring>
        </button>
        <button name="interest" method="interest()">
            <langstring xml:lang="en">Fields of interest</langstring>
            <langstring xml:lang="fr">Champs d'intérêt</langstring>
        </button>
        <button name="database" method="database()">
            <langstring xml:lang="en">Edit of database</langstring>
            <langstring xml:lang="fr">Base de données</langstring>
        </button>
        <button name="faq" method="faq()">
            <langstring xml:lang="en">FAQ</langstring>
            <langstring xml:lang="fr">FAQ</langstring>
        </button>
        <button name="send" method="send()">
            <langstring xml:lang="en">Send email</langstring>
            <langstring xml:lang="fr">Message</langstring>
```

Figure 4.3.1 : Une description XML d'une page

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:asp="remove">
    <xsl:param name="lang"></xsl:param>
    <xsl:output method="xml" indent="yes" encoding="utf-8" omit-xml-declaration="yes" />
    <xsl:template match="/">
        <html>
            <body link="none" vlink="none" alink="none">
                <table  width="97%" id="table1" border="0" cellSpacing="1" cellPadding="0">
                <xsl:for-each select="//field">
                        <xsl:if test="@type='buttons'">
                            <tr vAlign="top">
                                <td colspan="1" height="66px" valign="top" Color="blue">
                                    <xsl:for-each select="button">
                                        <asp:Button id="{@name}" CssClass="button_left" runat="serve
                                    </xsl:for-each>
                                </td>
                            </tr>
                        </xsl:if>
                    </xsl:for-each>
                </table>
            </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

Figure 4.3.2 : La feuille de style XSLT

Avec les dernières versions des browsers (IE, Mozilla, ...) les transformations XSLT s'effectuent au niveau de ces derniers. Autrement dit c'est au niveau client (browser) que la transformation avec la feuille de style se réalise, d'où l'énorme avantage de soulager les serveurs en reportant le traitement sur le poste client.

4.3.3 Implémentation de service Web "Moteur de recherche"

La technologie des services Web a été utilisée pour le développement et le déploiement de service Web Google, utilisé comme module de base dans notre application. Nous avons constaté l'existence de plusieurs produits qui assistent le développement des services Web. Les deux grandes familles d'industriels qui

offrent ces outils sont réparties entre le monde *Java* et *Windows*. Dans nos implémentations, nous avons opté pour la technologie *.NET* de *Microsoft*.

Microsoft Visual Studio .NET 2003 offre beaucoup de facilités d'implémentation et de déboguage des services. Le choix de la technologie .NET peut se résumer dans les points suivants :

- la richesse de l'environnement de développement ;
- la performance et la fiabilité des services Web déployés ;
- la familiarité avec l'outil ;
- support industriel d'autres outils.

D'un autre coté, plusieurs systèmes de question–réponse utilisent Google comme module de recherche de documents et de passages pertinents à la question de l'usager. Connu par sa simplicité et son efficacité, Google reste le premier moteur de recherche mondial pour réussir et affiner des recherches sur Internet. De plus, Google nous offre la possibilité d'implémenter l'API Google sur notre application sous forme d'un Web service afin de fournir un service de recherche rapide, puissant et personnalisé.

Pour utiliser le kit APIs Web de Google comme service Web au niveau de notre système, nous avons besoin de le télécharger et obtenir une clé de licence par une simple inscription au site ***http://www.google.com/apis/***.

Pour ajouter l'API Google comme une référence Web .NET offre cette possibilité en suivant les étapes suivantes :

1. Dans l'explorateur de solution de notre application (QUERI), on clique avec le bouton droit de la souris sur la rubrique "Web References"
2. On sélectionne "Ajoutez une référence Web"
3. On fait entrer l'adresse de la description du service Web (http://api.google.com/GoogleSearch.wsdl) dans la zone de texte Adresse.
4. On clique sur le bouton Ajouter une référence pour importer la définition du service Web.

Une fois toutes ces étapes effectuées, nous ajoutons une classe qui contient les informations de base, comme la clé de licence, les mots à rechercher, les pages de

résultats, le nombre maximal de résultats, un booléen indiquant si le filtre de pages similaires est activé ou non, le domaine de restriction …

Après l'implémentation de cette classe, nous considérons que notre Web service Google est prêt à passer ses résultats de recherche aux autres modules du système.

4.3.4 Base de données relationnelle

4.3.4.1 Modèle conceptuel des données

Conceptuellement la base de données de (questions/réponses) contient quatre tables **(Figure 4.3.3)**. La table ***Questions*** contient toutes les questions, avec les dates d'insertion ou de modification, la table ***Answers*** contient toutes les réponses. La table ***Users***, quant à elle, identifie tous les usagers autorisés pour faire des modifications dans la base de données (**BD Q/A**), tandis que la table Automatic_index contient toutes les questions qui ont été enregistrées d'une façon automatique par le système.

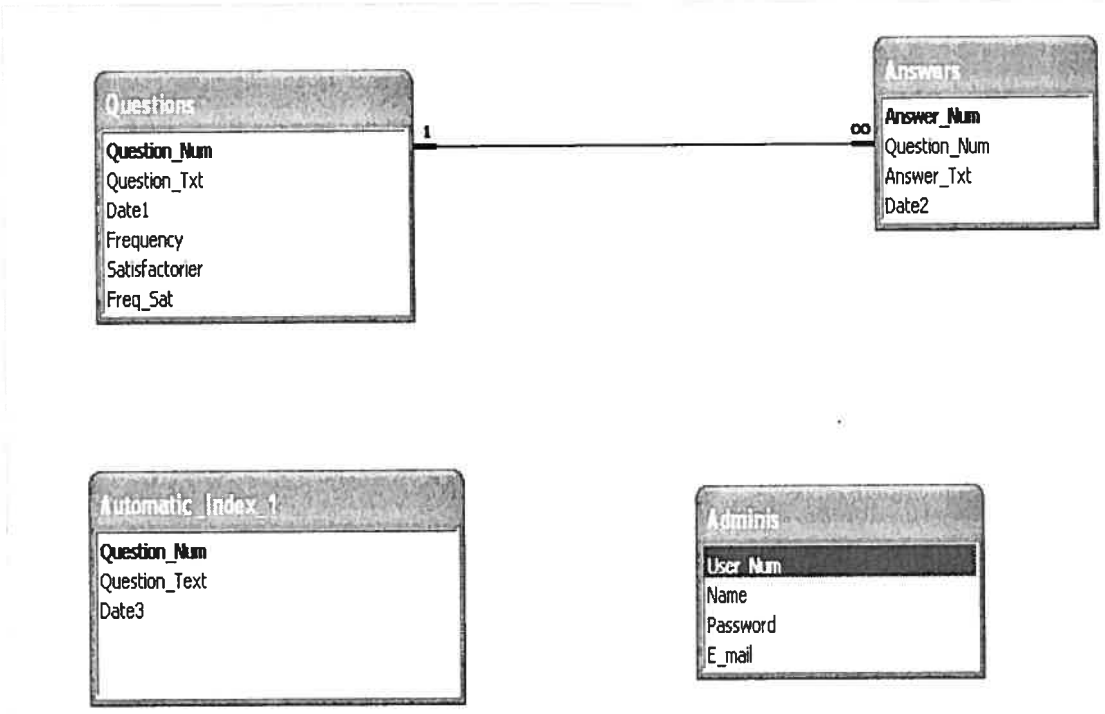


Figure 4.3.3 : Modèle conceptuel de la base de données

4.3.4.2 Exemple d'une entrée de la base de données

Dans ce qui suit, nous présentons une entrée de chaque table, avec les différents champs.

Table **Questions** :

Question_num	Question_Txt	Date1
3	**What do you mean by intelligent e-Learning ?**	22/07/2003 4:24:53 PM

Table **Answers** :

Answer_num	Question_num	Answer_Txt	Date2
3	3	**Intelligent e-Learning consists in providing e-Learning adapted and personalized to the learner (personal profile,rythm, pedagogical style,...)**	22/07/2003 4:24:53 PM

Table **Users:**

Num_User	Name	Password
1	Ali	*******

Table **Automatic_index** :

Question_num	Question_Txt	Date3
3	**What is the elearn?**	22/07/2003 4:24:53 PM

4.3.5 Implémentation de l'extracteur de réponse

Le module d'extraction de réponses se base surtout sur la coopération entre agents réactifs. L'exécution d'un agent réactif est directement liée à ses perceptions par une fonction de réflexe (Stimulus/Réponse), il réagit aux états du monde. La figure ci-dessous montre le concept d'un agent interactif.

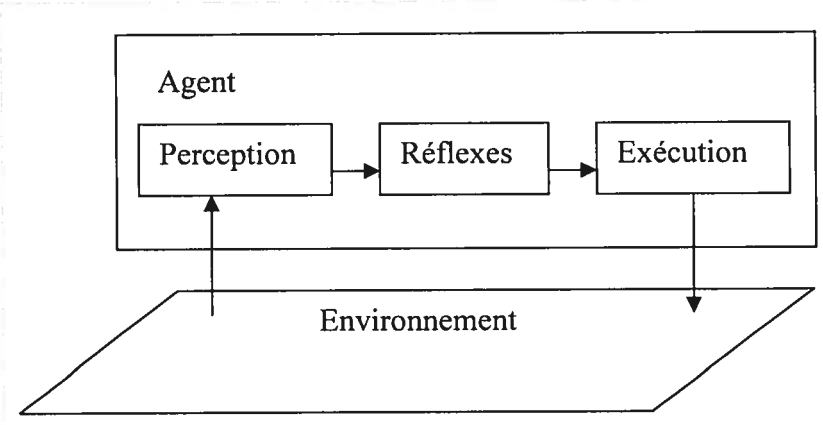


Figure 4.3.4 : Agent interactif

Dans notre cas, plusieurs facteurs (règles) peuvent déclencher l'exécution de l'un de nos agents, ce qui constitue l'environnement d'exécution. Les facteurs qui constituent cet environnement sont :

- le type de la question ;
- le contexte de la question ;
- la présence (l'absence) de la question dans le corpus local de données.

Selon son environnement, l'agent se déclenche automatiquement en appliquant la méthode convenable. Coté implémentation, chaque agent est une entité complètement déterminé par la fonction d'agent qui, étant donnée une séquence de perceptions, renvoie une action. La figure suivante montre la méthode "agent réactif"

```
méthode AGENT-RÉACTIF( perception, action)
{ static: règles, ensembles de règles condition-action
  état = INTERPRETATION-ENTRÉE( perception)
  règle = RÈGLE-SELECT(état, règles)
  action = RÈGLE-ACTION[règle]
  retourner action
}
```

Figure 4.3.5 : Méthode agent interactif

L'entrée est l'ensemble des règles qui peuvent déclencher l'intervention de l'agent et l'ensemble d'actions à appliquer.

Ensuite, on a trois procédures. La première est une interprétation de l'entrée, où on détermine si les facteurs nécessaires pour déclencher l'événement sont tous présents. Le rôle de la deuxième procédure est de sélectionner la règle convenable selon l'état et l'ensemble de règles. La troisième procédure quant à elle, détermine l'action à exécuter.

4.4 Scénarios d'utilisation

Nous allons présenter dans cette section quelques scénarios d'utilisation de QUERI. Une suite d'écrans permettra de démontrer les points forts de l'application et les différentes formes d'interaction usager-système.

La figure suivante illustre l'écran d'accueil de l'application :

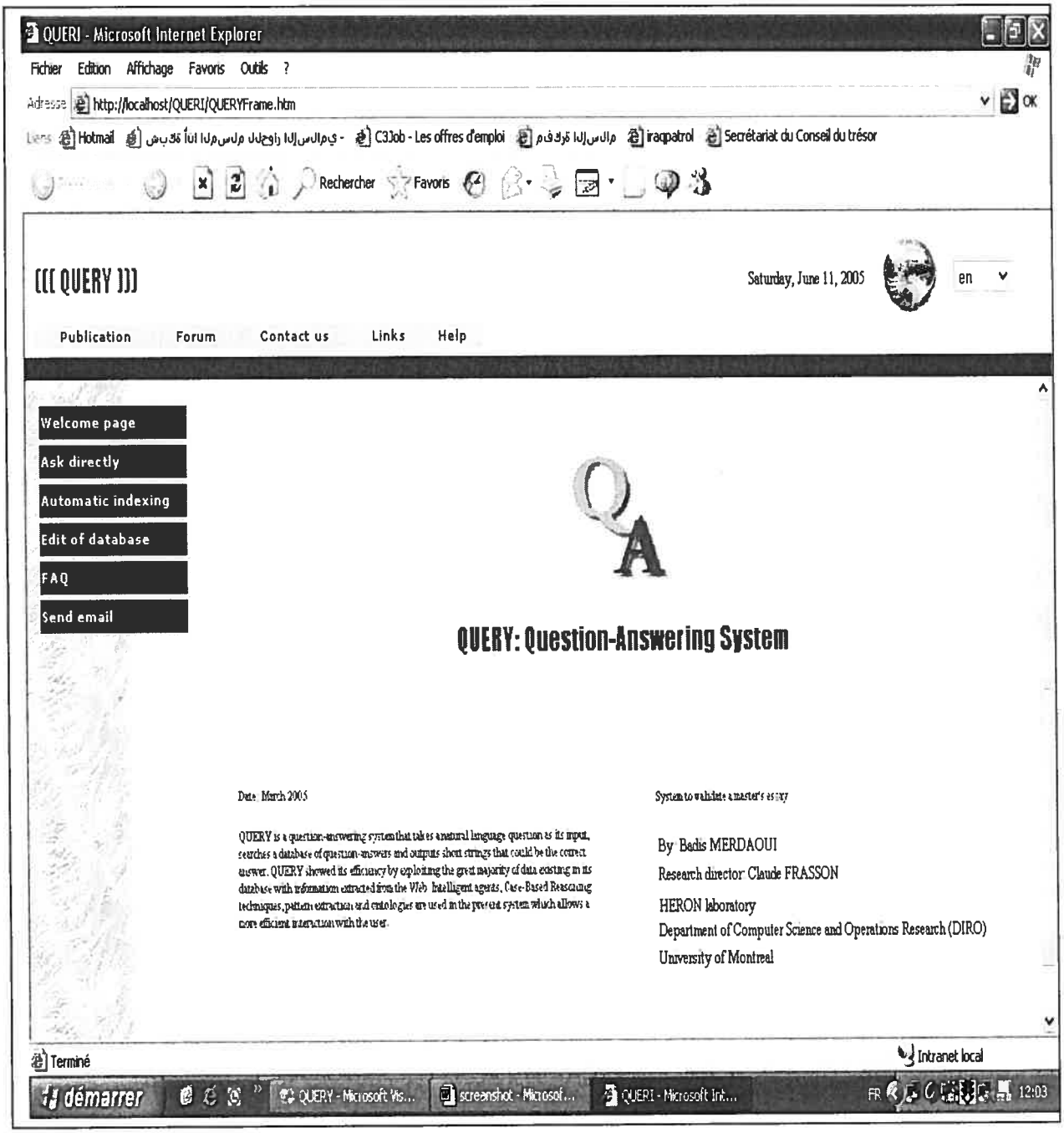


Figure 4.4.1 : Page d'accueil de QUERI

L'interaction avec l'usager peut prendre plusieurs formes, telle que : la demande de confirmation de la réponse en cas de doute, la demande clarification de cette dernière en cas d'ambiguïté, la proposition des réponses candidates qui ont des scores de similarités plus élevés, la proposition des liens hypertexte qui ont un rapport avec le contexte de la question, la proposition d'envoyer la question à un expert.

Ce premier exemple, représente le cas ou le système à besoin d'une clarification de la part de l'usager pour trouver la réponse convenable.

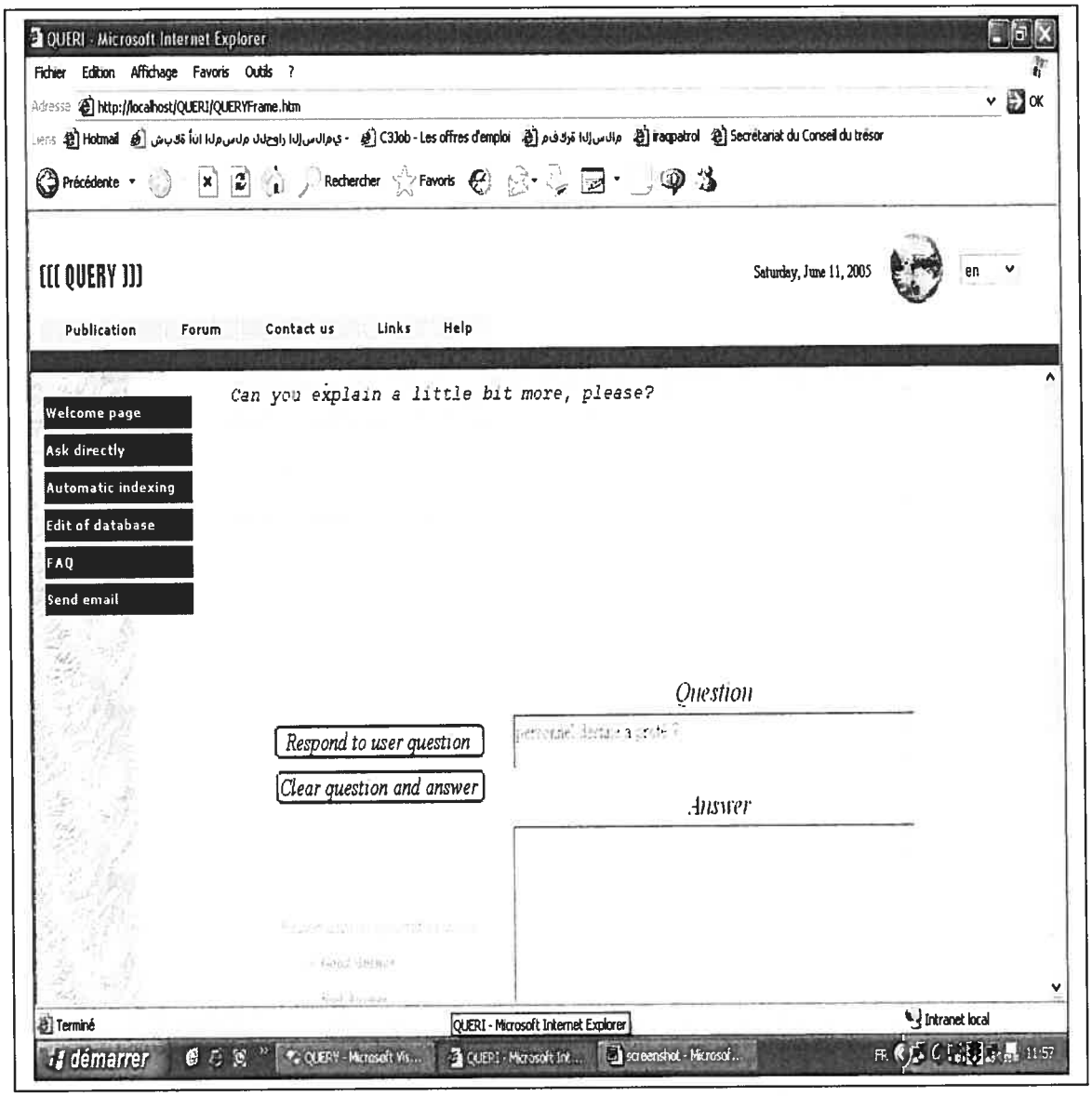


Figure 4.4.2 : Interaction système-usager

Dans le cas où la réponse candidate du système n'atteint pas le seuil minimum d'affichage, le système à besoin de confirmer avec l'usager.

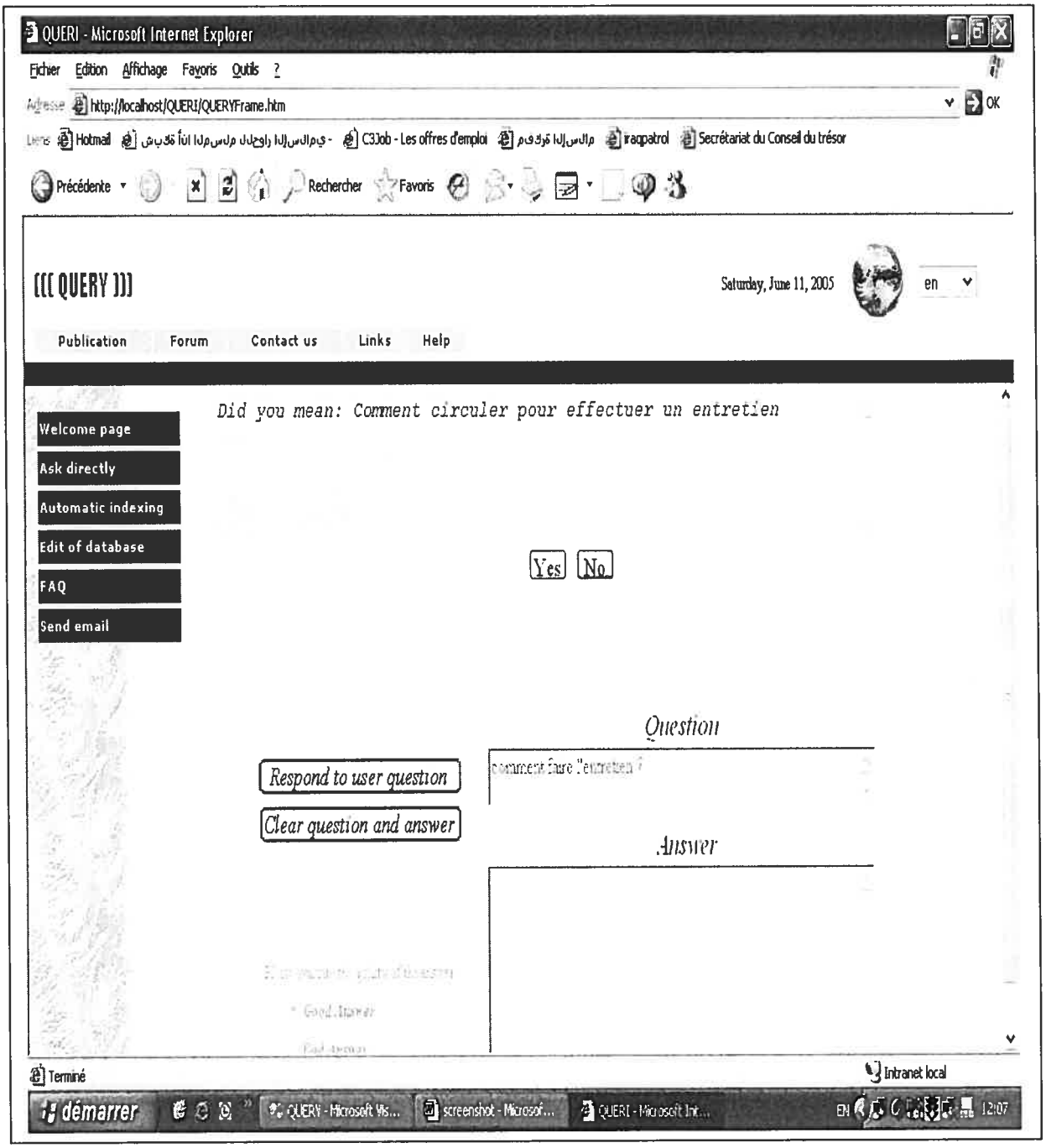


Figure 4.4.3 : Interaction système-usager

Dans le cas où on a le même score de similarité pour plusieurs réponses candidates, on propose à l'usager de choisir entre trois réponses candidates.

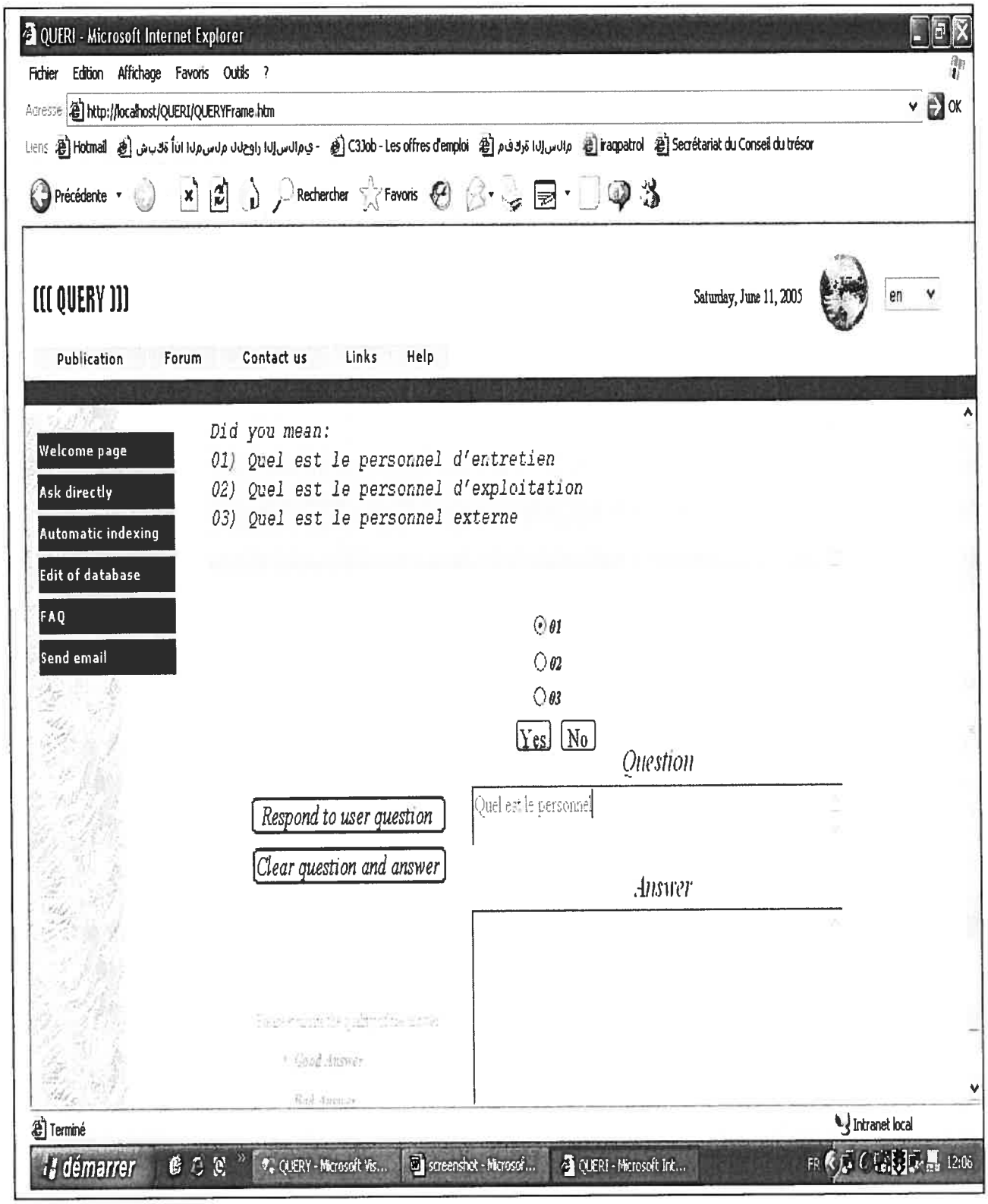


Figure 4.4.4 : Interaction système-usager (proposition des questions les plus pertinentes)

Lorsque le système ne trouve la réponse ni dans le corpus local, ni sur le Web, il affiche les liens les plus pertinents.

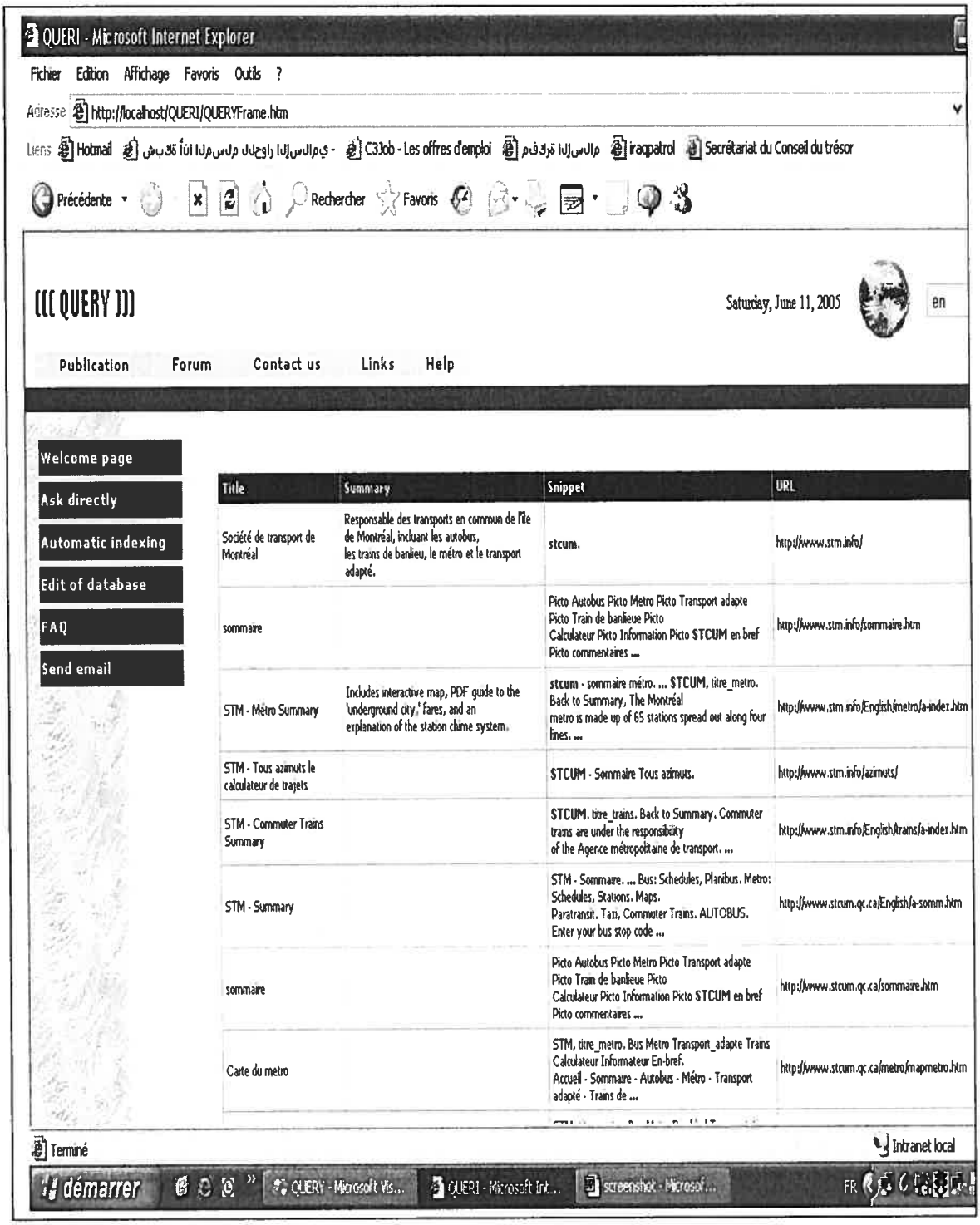


Figure 4.4.5 : Interaction système-usager (Proposition de liens les plus pertinents)

Nous présentons un échantillon des autres interfaces, tel que l'administration de la base de données locale et l'indexation automatique des questions sans réponse.

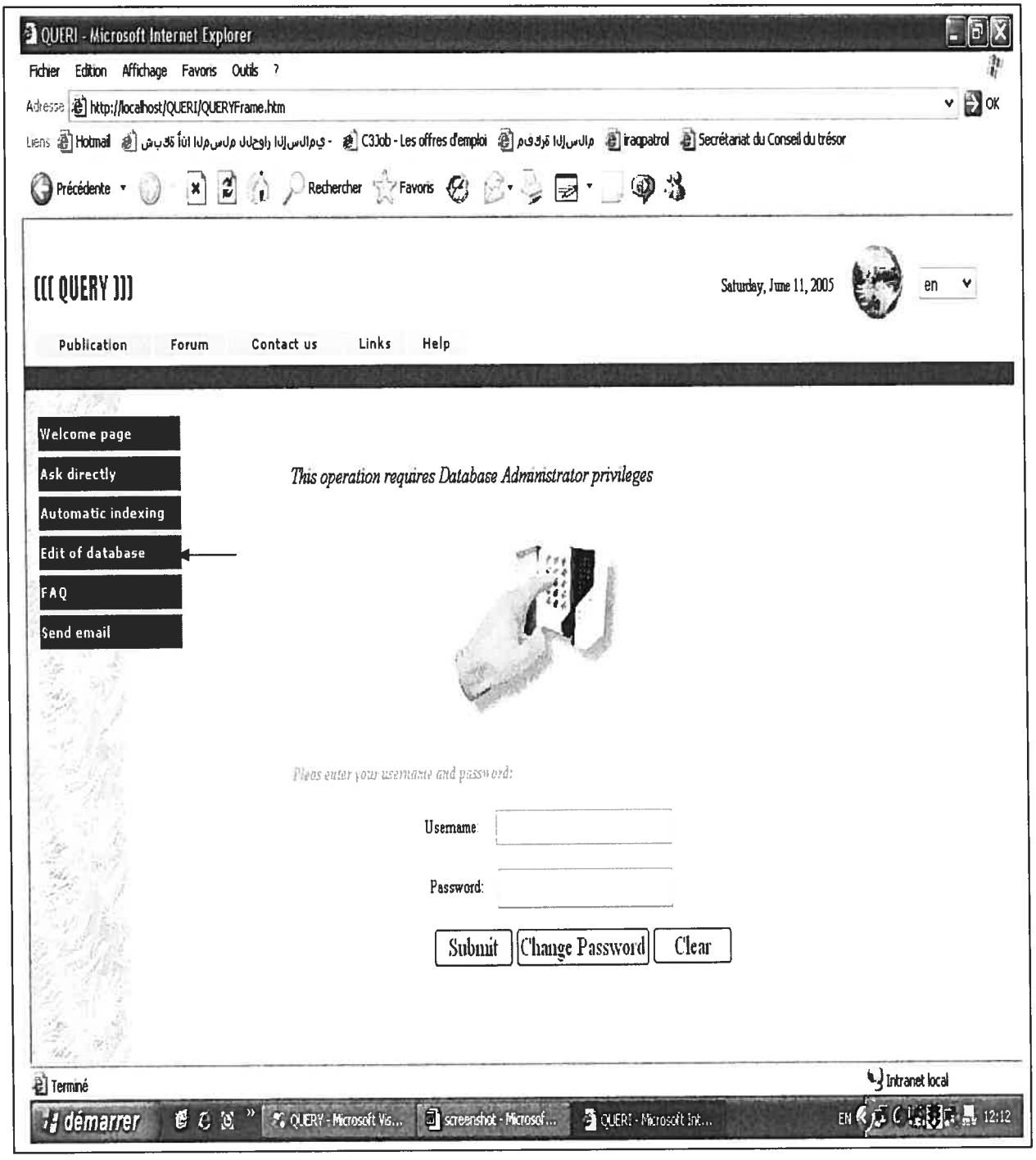


Figure 4.4.6 : Accès à la gestion de la base de données

Si l'usager a les droits d'administrateur, il aura la fenêtre suivante :

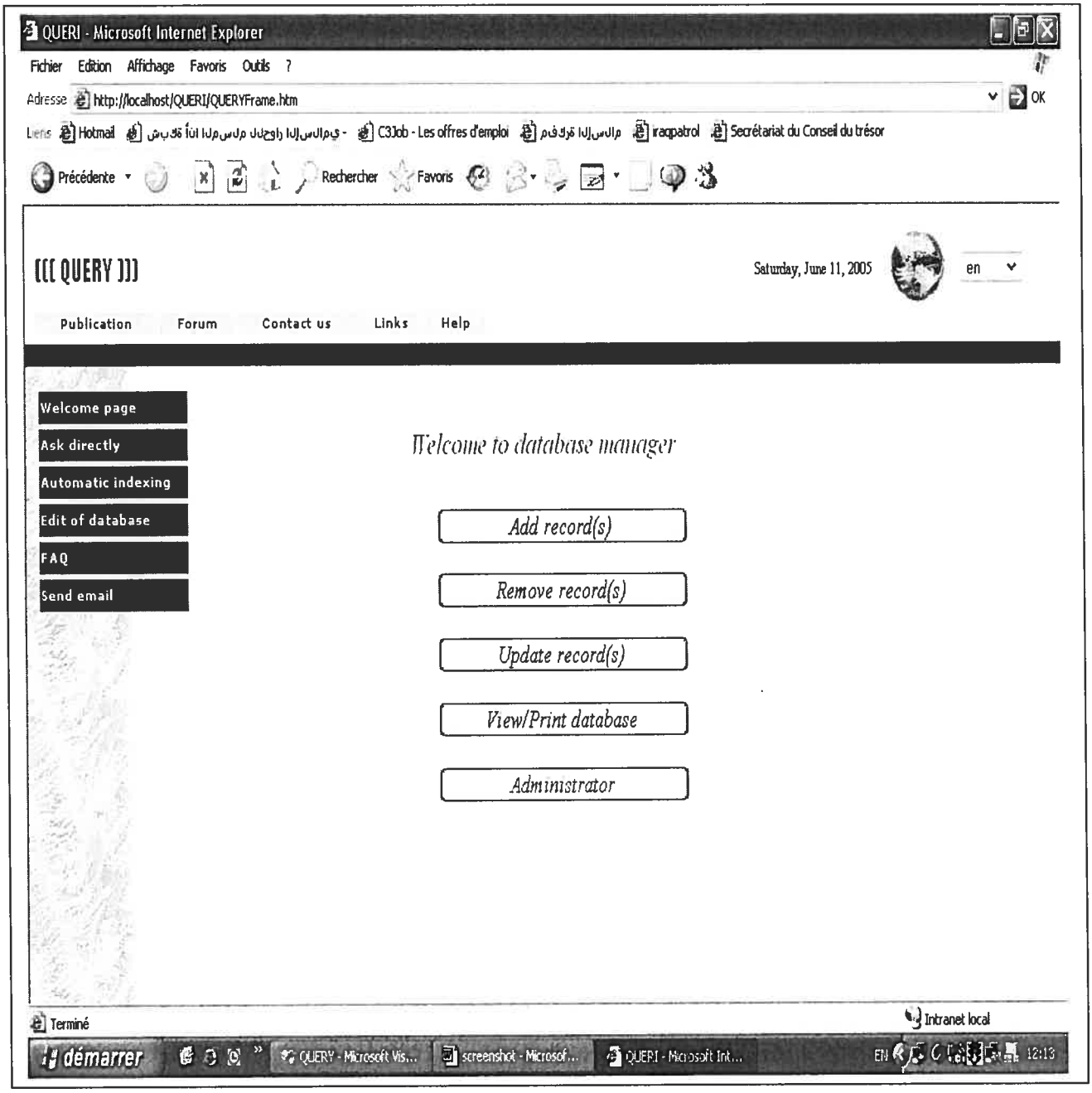


Figure 4.4.7 : Gestion de la base de données

Voici un aperçu de la fenêtre "Remove record(s)"

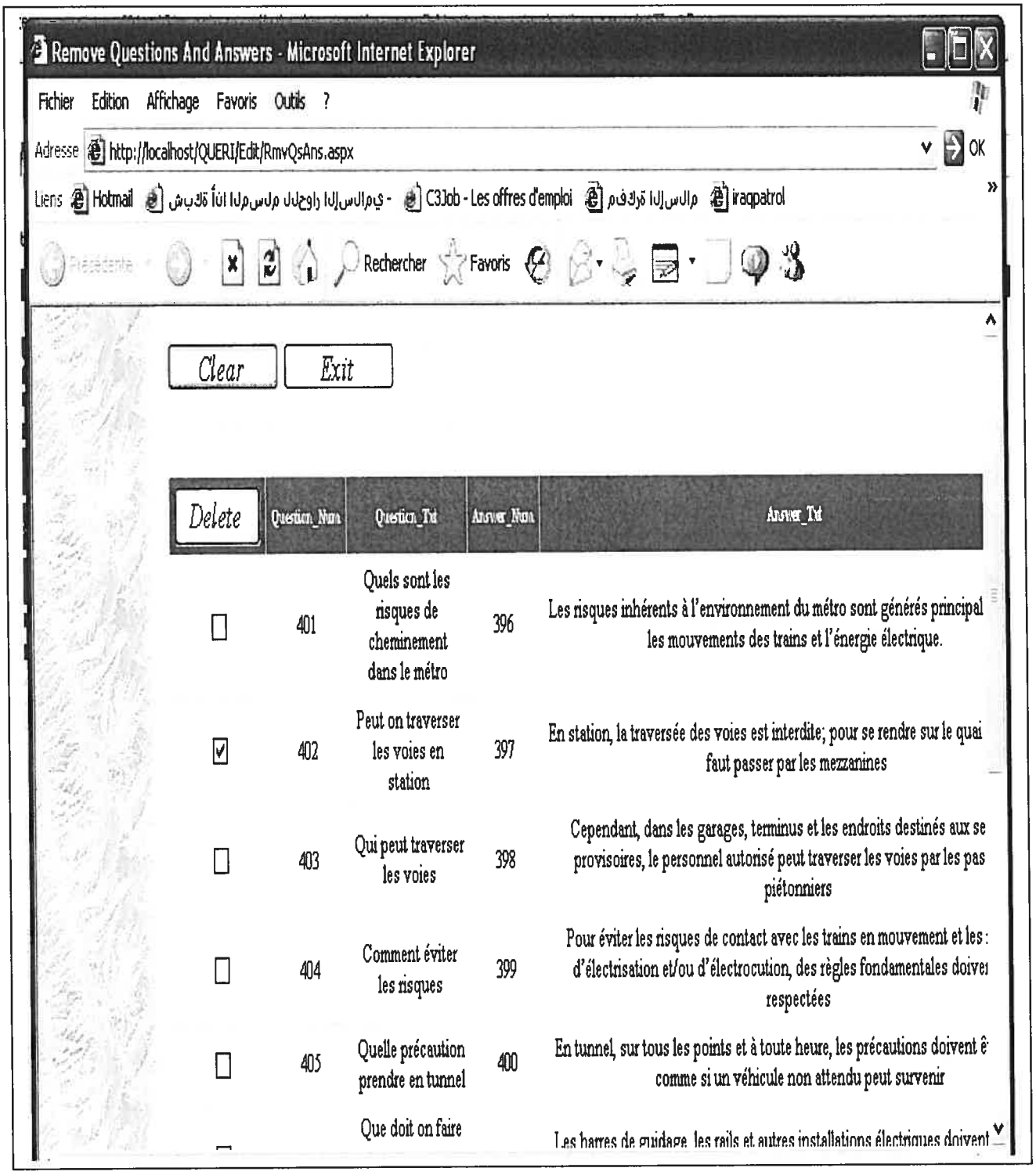
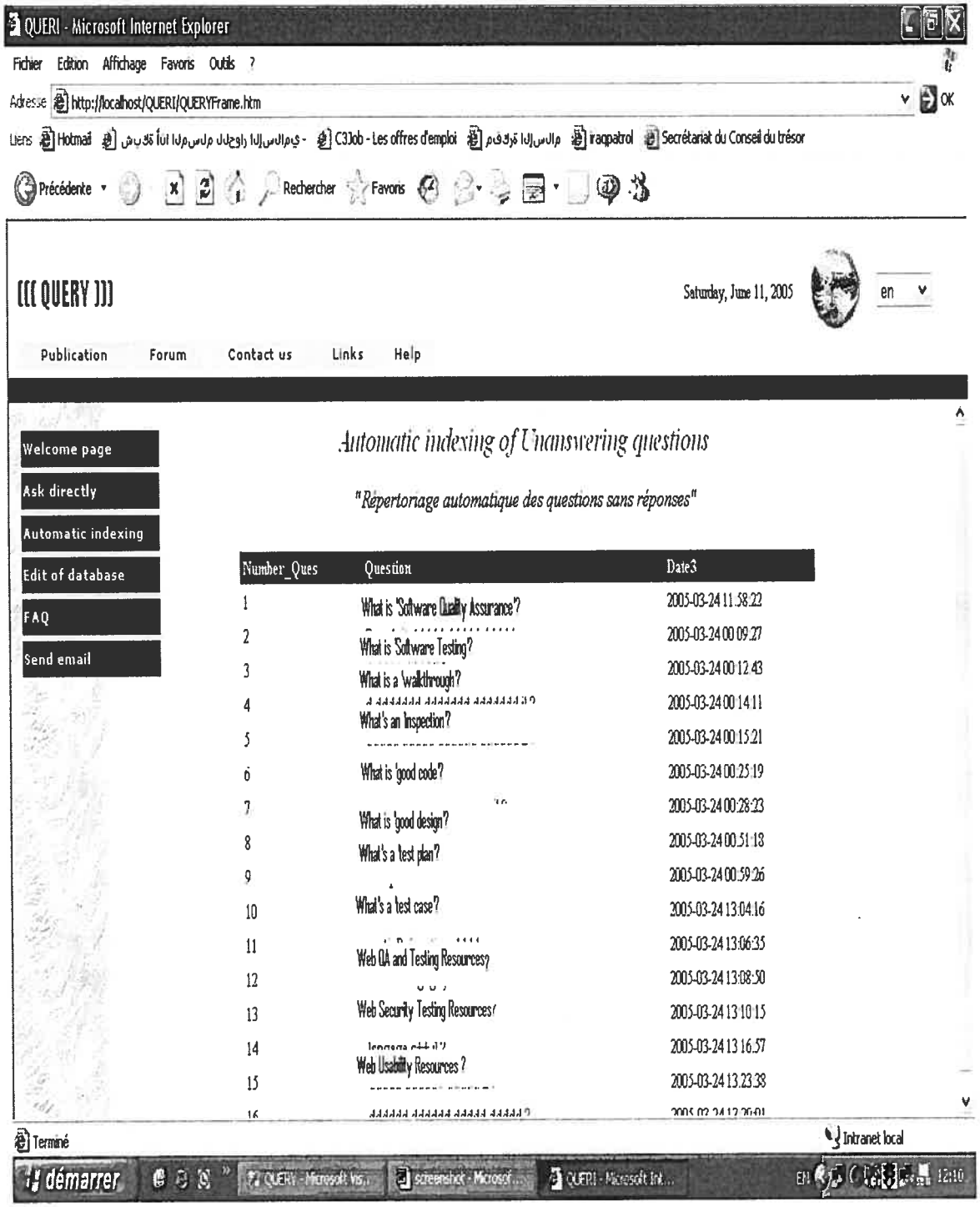


Figure 4.4.8 : Effacer un couple question–réponse de la base de données

Le système peut classer automatiquement les questions sans réponse avec leurs dates, ce qui permet d'enrichir le corpus local par un expert.



L'usager peut envoyer sa question à l'administrateur ou l'expert.

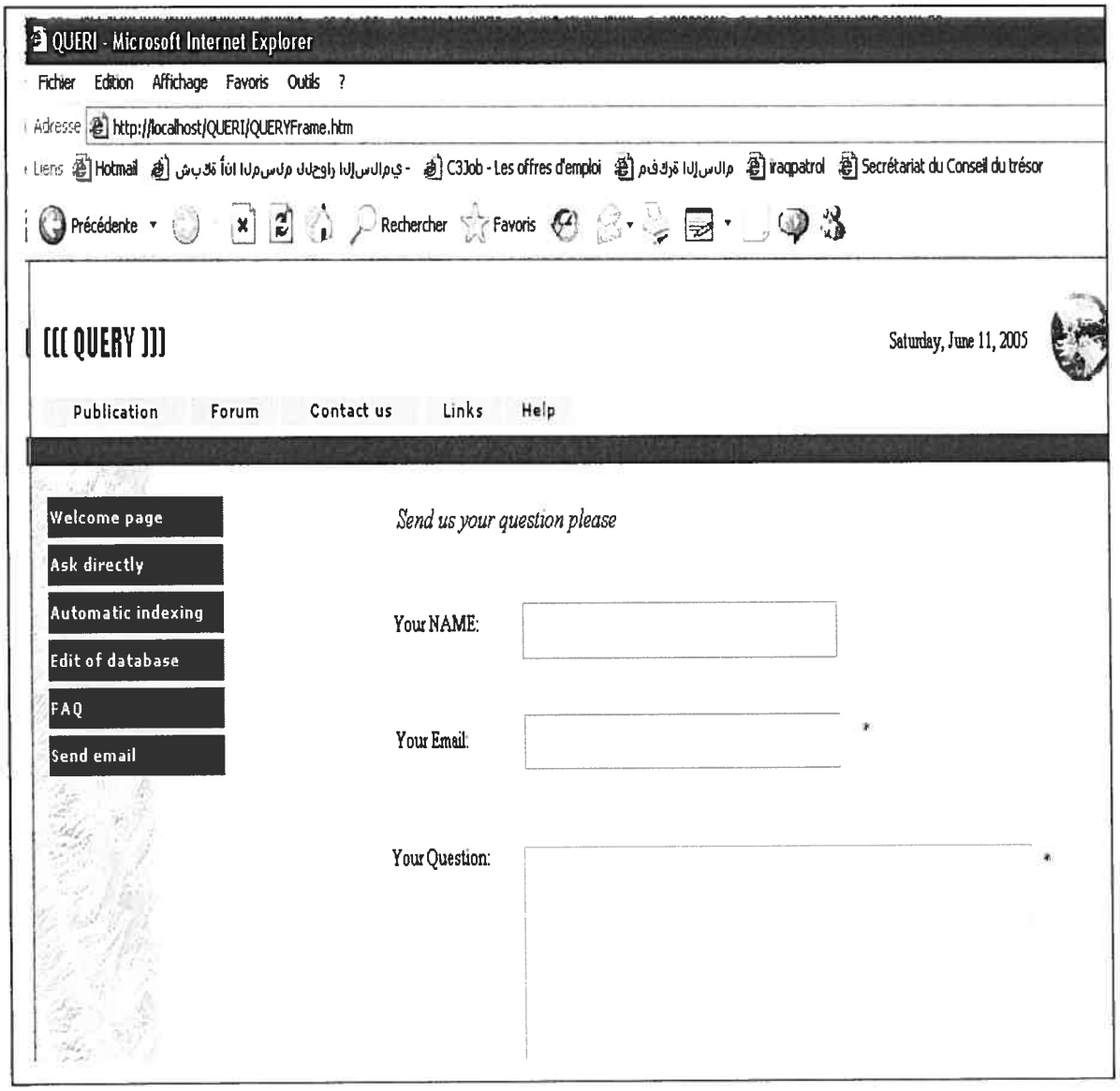


Figure 4.4.10 : Possibilité d'envoyer la question à un expert (administrateur)

4.5 Conclusion

Dans ce chapitre nous avons montré les détails d'implémentation de QUERI. Nous avons explicité les différents choix technologiques relatifs à l'implémentation ainsi que les bibliothèques logicielles disponibles. Nous avons vu les différentes composantes logicielles utilisées pour l'obtention d'un système avec un minimum de performance et de fiabilité. Pour évaluer le projet et en parallèle des articles publiés nous sommes basés sur les tests de la communauté des utilisateurs finaux (étudiants). Il y avait un accord total de l'absence d'un produit orienté vers le domaine d'apprentissage. Un système de question–réponse intégré dans un système tutorial est indispensable. Une majorité des utilisateurs (78%) ont eu des réponses exactes à leurs questions et par conséquent confirment l'efficacité de QUERI et le besoin d'élaborer des systèmes de question–réponse adaptable.

CHAPITRE 5

5. CONCLUSION

Nous avons exploré le développement des systèmes de question–réponse, un domaine de recherche très actif et qui a attiré l'attention de beaucoup de chercheurs et d'industriels. Nous nous somme concentrés sur l'application de ces derniers au domaine de l'apprentissage.

La plupart des systèmes de question–réponse tentent de fournir une réponse exacte en effectuant une recherche directe sur le Web, et sans aucune interaction avec l'usager. Ces systèmes ont tous des architectures sans module de spécification de la réponse et ils fournissent aux usagers des réponses avec un taux d'erreur élevé et dans la plupart des cas une déclaration d'absence de réponse. Ils souffrent aussi de l'absence d'un module qui commence la recherche dans une base de données locale. L'objectif principal de notre approche était de réaliser un système réutilisable et intégrable qui peut répondre à des questions posées en langage naturel à partir d'un corpus de données et qui doit avoir la capacité de supporter une augmentation de ses contraintes dans un domaine particulier.

Pour y arriver, nous avons évalué beaucoup de systèmes de question–réponse et nous avons soulevé les insuffisances relatives à l'exactitude et la justification de la réponse d'un coté et à l'adaptabilité d'un autre coté.

Pour améliorer l'ensemble du processus, nous avons utilisé plusieurs techniques :

L'approche multi-agents, les patrons d'extractions et les ontologies pour l'extraction de réponse, la recherche d'information pour la recherche des documents pertinents, le traitement automatique de la langue naturelle dans le cas de l'analyse lexical et la détermination du type de la question et le raisonnement à base de cas pour le calcul de similarité. Nous avons montré que la combinaison de ces techniques pouvait donner des résultats très satisfaisants.

Pour évaluer les performances de QUERI nous avons utilisé plusieurs critères, parmi lesquels : l'utilité du système dans le domaine d'apprentissage, l'exactitude

de la réponse fournie à l'usager, l'aspect d'interaction système usager et la convivialité de l'interface interactive.

Un pourcentage de 84% des utilisateurs croit à l'utilité de systèmes comme outil d'aide aux apprenants et plus de 78% des étudiants ont eu des réponses exactes à leurs questions après certaines interactions avec le système.

69% des étudiants croient que l'interaction avec le système a beaucoup aidé pour trouver la bonne réponse et que l'interface est conviviale.

Les résultats de tests sont prometteurs et le système continue à être enrichis pour répondre mieux aux attentes des utilisateurs finaux.

Un système de question–réponse efficace et accessible aux clients sur le site Web de l'entreprise permettrait de réduire le volume de courriels.

Les systèmes de question–réponse ont montré leur efficacité en exploitant la grande masse de données existante dans leurs bases de données et celle trouvée sur le Web. Ils continuent de s'imposer comme l'outil de recherche d'information le plus important même devant les fameux moteurs de recherche.

Malgré les résultats positifs et prometteurs atteints, nous croyons qu'il reste encore du travail à faire pour améliorer notre extracteur de réponse qui se base actuellement sur deux agents principaux seulement, ce qui limite l'utilisation des méthodes d'extraction. Des perspectives de recherche sont encore ouvertes dans le cadre du système QUERI, comme l'orientation vers des systèmes multimédias, où le système cherche des images et des paroles au lieu du texte.

BIBLIOGRAPHIE

[Aamodt *et al.*, 1994] A. Aamodt, E. Plaza Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches, AICom – Artificial Intelligence Communications, 1994.

[Ahn *et al.*, 2004] D. Ahn, V. Jijkoun, G. Mishne, K. Muller, M. de Rijke, S. Schlobach. Using Wikipedia at the TREC QA Track, Informatics Institute, University of Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands, 2004.

[Bonabeau *et al.*, 1994] E. Bonabeau, G. Theraulaz: Why Do We Need Artificial Life? Artificial Life 1(3): 303-325 (1994)

[Chalendar *et al.*, 2002] G. de Chalendar, T. Dalmas, F. Elkateb-Gara, O. Ferret, B. Grau, M. Hurault-Plantet, G. Illouz, L. Monceaux, I. Robba, A. Vilnat, LIMSI-CNRS, The Question Answering System QALC at LIMSI, Experiments in Using Web and WordNet, In Proceedings of the 11th Text REtrieval Conference (TREC-11), 2002.

[Chali *et al.*, 2004] Y. Chali, S. Dubien. University of Lethbridge's Participation in TREC-2004 QA Track, Department of Mathematics and Computer Science, University of Lethbridge 4401 University Drive, Lethbridge, Alberta, Canada, T1K 3M4, 2004

[Chen *et al.*, 2004] J. Chen, H. Ge, Y. Wu, S. Jiang. UNT at TREC 2004: Question Answering Combining Multiple Evidences, School of Library and Information Sciences, University of North Texas, 2004.

[Chu-Carroll *et al.*, 2003] J. Chu-Carroll, J. Prager, C. Welty, K. Czuba, D. Ferrucci, IBM T.J. Watson Research Center IBM's PIQUANT in TREC 2003 (TREC-12), 2003.

[Chu-Carroll *et al.*, 2004] J. Chu-Carroll, J. Prager, C. Welty, K. Czuba, D. Ferrucci, IBM T.J. Watson Research Center IBM's PIQUANT in TREC 2004 (TREC-13), 2004.

[Clarke *et al.*, 2001] Clarke, C. L. A., Cormack, G. V., Lynam, T. R., Li, C. M., and McLearn, G. L. (2001). Web Reinforced Question Answering (MultiText Experiments for TREC 2001). In

Proceedings of The Tenth Text Retrieval Conference (TREC-10), pages 673–679, Gaithersburg, Maryland.

[Crabtree, 1997] Barry Crabtree. The Practical Applications Company, Blackpool, Lancashire, UK. Held London, UK, April 21-23, 1997.

[Cui *et al.*, 2004] H. Cui, K. Li, R. Sun, T. Chua, M. Kan. National University of Singapore at the TREC-13 Question Answering Main Task, Department of Computer Science, School of Computing, National University of Singapore, 2004.

[Fellbaum *et al.*, 1998] C. Fellbaum WordNet, an electronic lexical database, The MIT Press, Cambridge, Massachusetts, 1998, 423 p.

[Ferber *et al.*, 1995] J. Ferber, Les systèmes multi-agents: Vers une intelligence collective. InterEditions, 1995.

[Gaizauskas *et al.*, 2004] R. Gaizauskas, M. A. Greenwood, M. Hepple, I. Roberts, H. Saggion. The University of Sheffield's TREC 2004 Q&A Experiments, Department of Computer Science, University of Sheffield, UK, 2004

[Graesser *et al.*, 1992] A. Graesser, N. Person, J.Huber. Mechanisms that Generate Questions. Awrence Erlbaum Associates: Hillsdale, New Jersey. 1992

[Green *et al.*, 1961] Green B., Wolf A., Chomsky C. et Laughery K. – « BASEBALL: an automatic question answerer ». In Proceedings of the Western Joint Computer Conference. 1961.

[Gresse *et al.*, 2001] C. Gresse von Wangenheim, A. Bortolon, A. von Wangenheim: A Hybrid Approach for the Management of FAQ Documents in Latin Languages. ICCBR 2001: 204-218, 2001.

[Han *et al.*, 2004] K. Han, H. Chung, S. Kim, Y. Song, J. Lee, H. Rim. Korea University Question Answering System at TREC 2004, Natural Language Processing Lab. Dept. of Computer Science and Engineering, Korea University 1, 5-ga, Anam-dong, Seongbuk-gu, Seoul 136-701, Korea, 2004.

[Harabagiu, 2000] Harabagiu, Experiments with Open Domain Textual Question Answering, in Proceedings of COLING-2000, Saarbruken Germany, pages 292-298, August 2000.

[Hendrix *et al.*, 1977] Hendrix G. – « Human engineering for applied natural language processing ». In Proceedings of the 5th International Joint Conference on Artificial Intelligence (IJCAI'1977), 1977.

[Jurafsky *et al.*, 2000] D. Jurafsky, J. H. Martin, An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, Prentice-Hall, 2000

[Kaisser *et al.*, 2004] M. Kaisser, T. Becker. Question Answering by Searching Large Corpora with Linguistic Methods, Saarland University / DFKI GmbH, 2004.

[Katz *et al.*, 2004] B. Katz, M. Bilotti, S. Felshin, A. Fernandes, W. Hildebrandt, R. Katzir, J. Lin, D. Loreto, G. Marton, F. Mora, O. Uzuner. Answering multiple questions on a topic from heterogeneous resources, MIT Computer Science and Artificial Intelligence Laboratory Cambridge, MA 02139, 2004.

[Kejselj *et al.*, 2004] V. Kejselj, A. Cox. DalTREC 2004: Question Answering using Regular Expression Rewriting, Faculty of Computer Science Dalhousie University, Halifax, Canada, 2004.

[Kolodne, 1994] Kolodner, J.L. Case-Based Learning, Kluwer Academic Publishers, Dordrecht, Netherlands, 1993.

[Kosseim *et al.*, 2000] L. Kosseim. Systèmes de réponse automatique : Etat de l'art, RALI, DIRO, Université de Montréal, août 1999, révisé en mars 2000

[Leake, 1996] Leake, D. B., CBR in Context: The Present and Future, in Case-Based Reasoning: Experiences, Lessons, and Futures Directions, In Leake, D. editor, AAAI Press/MIT Press, 1996.

[Lenhert *et al.*, 1972] Lenhert W. – « A conceptual theory of question answering ». In Proceedings of the 5th International Joint Conference on Artificial Intelligence (IJCAI'1977), 1977.

[Lenhert *et al.*, 1981] Lenhert W. – « A computational theory of question answering ». In Joshi A., webber W. et Sag I. (éds.) Elements of discourse understanding. Cambridge University Press, Cambridge. 1981.

[Mengelle *et al.*, 1996] T. Mengelle, C. Frasson (1996). A Multi-Agent Architecture for an ITS with Multiple Strategies (ITS'1996)

[Merdaoui *et al.*, 2004] B. Merdaoui, C. Frasson QUERI : Un système de question–réponse collaboratif et interactif, TICE 2004, Paris, France, 2004.

[Merdaoui *et al.*, 2005] B. Merdaoui, C. Frasson Un système de question–réponse collaboratif et interactif dans le cadre d'outils d'aide aux apprenants, DIVA 2005, Montréal, Canada, 2005.

[Mignerat *et al.*, 2002] M. Mignerat, B. A. Aubert. Panorama des Systèmes d'Intégration InterOrganisationnels : Aspects Technologiques, 2002RP-03, Montréal, 2002

[Molla *et al.*, 2004] D. Molla, M. Gardiner. AnswerFinder at TREC 2004, Centre for Language Technology, Division of Information and Communication Sciences, Macquarie University, Sydney, Australia, 2004.

[Moldovan *et al.*, 1999] D. I. Moldovan, S. M. Harabagiu, M. Pasca, R. Mihalcea, R. Goodrum, R. Girju, V. Rus: LASSO: A Tool for Surfing the Answer Net. TREC 1999

[Molovan *et al.*, 2003] D. Molovan, S. Harabagiu, R. Girju, P. Morarescu, F. Lacatusu, A. Novischi, A. Badulescu, O. Bolohan, LCC Tools for Question Answering, 2003.

[Nyberg *et al.*, 2002] E. Nyberg, T. Mitamura, J. Carbonnell, J. Callan, K. Collins-Thompson, K. Czuba, M. Duggan, L. Hiyakumoto, N. Hu, Y. Huang, J. Ko, L.V. Lita, S. Murtagh, V. Pedro, D. Svoboda, Carnegia Mellon University, The JAVELIN Question-Answering System at TREC 2002, In Proceedings of the 11 th Text REtrieval Conference (TREC-11), 2002.

[Plamondon, 2002] L. Plamondon, Le système de question–réponse QUANTUM. Mémoire de maîtrise, Université de Montréal, Montréal, Canada, mars 2002.

[Roussinov *et al.*, 2004] D. Roussinov, J. Antonio Robles-Flores, Y. Ding. Experiments with Web QA System and TREC2004 Questions, Arizona State University, 2004

[Poibeau *et al.*, 2003] T. Poibeau, P. Zweigenbaum, A. Nazarenko, Traitement Automatique des Langues pour les systèmes de Question/Réponse Document de travail élaboré dans le cadre de l'action spécifique RIP-WEB, 2003.

[Russell *et al.*, 1997] Russell, S.J. Rationality and intelligence. Artificial Intelligence, Vol. 94, p.57-77., 1997.

[Schank et Abelson, 1977] Schank, R.C. et Abelson, R. – Scripts, Plans, Goals, and Understanding. Hillsdale, NJ: Earlbaum Assoc, 1977.

[Schone *et al.*, 2004] P. Schone, G. Ciany, P. McNamee, J. Mayfield, T. Bassi, A. Kulman. Question Answering with QACTIS at TREC 2004 U.S. Department of Defense, Ft. George G. Meade, MD 20755-6000, 2004.

[Soubbotin *et al.*, 2001] M.M. Soubbotin, InsightSoft-M, Patterns of Potential Answer Expressions as Clues to the Right Answers, In Proceedings of the 10th Text REtrieval Conference (TREC-10), 2001.

[Soubbotin *et al.*, 2003] M. M. Soubbotin, S. M. Soubbotin, Use of Patterns Detection of Answer Strings: A Systematic Approach, 2003.

[Sutcliffe *et al.*, 2004] R. F. E. Sutcliffe, I. Gabbay, K. White, A. O'Gorman, M. Mulcahy. Question Answering using the DLT System at TREC 2004, Documents and Linguistic Technology Group, Department of Computer Science and Information Systems, University of Limerick, Limerick, Ireland, 2004.

[Tan *et al.*, 2004] W. Tan, Q. Chen, S. Ma. THUIR at TREC 2004: QA, State Key Lab of Intelligent Tech. & Sys., CS&T Dept, Tsinghua University, Beijing 100084, China, 2004.

[von Wangenheim *et al.*, 2001] C. G. von Wangenheim, A. Bortolon, V. Wangenheim, "A hybrid approach for the management of FAQ Document in Latin languages". In: iccbr 2001- international conference of case-based reasoning, 2001, Vancouver. International conference of case-based reasoning 2001.

[Weizenbaum, 1966] J. Weizenbaum. ELIZA--A Computer Program For the Study of Natural Language Communication Between Man and Machine,Massachusetts Institute of Technology, Department of Electrical, Engineering, Cambridge, Mass.. Communications of the ACM, Volume 9, Number 1 36-35. January 1966.

[Wilensky, 1982] Wilensky R. – Talking to Unix in English: an overview of an on-line Unic consultant. Rapport technique. Université de Californie à Berkeley. 1982.

[Winograd *et al.*, 1972] Winograd T. – Understading natural language. Academic Press, New York, 1972.

[Woods *et al.*, 1972] Woods W., Kaplan R., Nash-Webber B. – The lunar sciences natural language information system. Rapport technique. BBN. 1972.

[Wooldridge *et al.*, 1995] M.Wooldridge, et N. R. Jennings. Agent theories, architectures, and languages. Dans Wooldridge, Jennings (ed), Intelligent Agents, Springer Verlag,p.1-22.1995.

[Wu *et al.*, 2004] L. Wu, X. Huang, L. You, Z. Zhang, X. Li, Y. Zhou. FDUQA on TREC2004 QA Track, Fudan University, Shanghai, China, 2004.