Ref.: 98/ I/ I /

Soutenu à la session de Juin 1998

Université de Tunis II

Ecole Nationale des Sciences de l'Informatique

Rapport
de Projet de Fin d'Etudes

Présenté en vue de l'obtention du titre
d'**Ingénieur Informaticien**

par IMED CHIHI

Hannibal – Un Moteur de Recherche

Avec Plug-in en Arabe

Encadré par

MOHAMED SAÏD OUERGHI
Maître Assistant, ENSI

MOEZ SOUABNI
Directeur, MRS

.

مُلَخَّص

إِنَّ النُّموَّ المُتَواصِلَ للويب يَجعَلُ مِنهُ كُتلَة عظِيمَة مِن المَعلُومَاتِ، وَ رَغمَ أَنَّ العَالَمَ العَرَبيّ لَا يَزَالُ غَيرَ مَعنيٍّ مُبَاشَرَة بِمَشَاكِلِ هَذَا التضخُّمِ، يَجدُرُ التَّعجِيلُ بِالتّفكِيرِ فِي تَقنيَاتٍ فَعَّالَةٍ قَادِرَةٍ علَى تَمكِينِ المُستَعمِلِ العَرَبيّ مِن البَحثِ عَن الوَثَائِقِ الّتي يبغِيهَا دَاخل الويب العَرَبيّ. فَضلًا عَن المَشَاكِلِ المُتَعَلِّقَةِ بِتَحديدِ المَعلُومَةِ المَقصُودَةِ، فَاللُّغةُ العربيّةُ لَا تَزَالُ غَيرَ مَدعُومَةٍ بِالشّكلِ المرضي فِي تَطبِيقَاتِ إِنتَرنَت (و غيرهَا)، و أَغلَبُ المُنتَجَاتِ المُسَوَّقَة لَا تُقَدِّمُ حلًّا جذريًّا.

يَرمِي هَذَا المَشرُوعُ إِلَى تَقديمِ وَسيلَةِ بَحثٍ عَلَى الويب مَعَ بَعضِ الإِضَافَاتِ الّتِي مِن شَأنِهَا أَن تَعرِضَ الوَثَائِقَ العَرَبيّةَ ضِمنَ مُتَصَفِّحِ الويب.

اَلكلمَات المفَاتِيح إِنترنت، ويب، محرّكَات بحث، اَلعَالم العربيّ، اَللّغة العَرَبيَّة، حنّبعل، هَارفست، آليَّات الويب، متصفّح الويب.

Abstract

The increasing growth of the Web is turning it into an enormous mass of information, and though the Arab World is still not dangerously concerned with these overloads, it should be wise to start considering some efficient techniques that might allow the Arabian user to find the document he needs within the expanding Arab World Wide Web. Besides the common problems of targetting the desired information, the Arabic language is not yet well supported by Internet applications (among others) and the existing products are mostly kinds of workarounds. This project aims to provide a search facility with some extensions that would allow arabic documents viewing from within a popular browser.

Keywords Internet, World-Wide Web, Search Engines, Arab World, Arabic Language, Hannibal, Harvest, Web Robots, Web Browser.

Résumé

La croissance du Web est entrain d'en faire une énorme masse d'information universelle. Le Monde Arabe n'a pas encore suffisament contribué à cette explosion. Malgré cela, il serait raisonnable de penser à des techniques efficaces qui permettraient à l'utilisateur Arabe de trouver les documents qu'il cherche dans le Web du Monde Arabe.

En plus des problèmes habituels de ciblage de l'information souhaitée, la langue Arabe n'est pas encore bien supportée par les applications du type Internet (entre autres) et les produits existants sont plutôt des solutions "dépannage".

Ce projet a pour but de fournir un service de recherche avec quelques extensions qui permettraient l'affichage de documents en Arabe dans un navigateur populaire.

Mots Clés Internet, Web, Moteurs de Recherche, Monde Arabe, Langue Arabe, Hannibal, Harvest, Robots du Web, Navigateurs.

Table des matières

Liste des figures

Liste des tableaux

Introduction

MRS est une société de services en informatique basée à la Marsa dont l'activité majeure est la conception des sites Web et des CD-ROMs multimédia. J'ai été en stage de mémoire de fin d'études dans cette entreprise et plus précisement dans son département de développement. Il m'a été confié la charge de la réalisation d'une extension pour le navigateur Netscape Navigator pour pouvoir afficher correctement des pages en Arabe.

Les spécifications mentionnaient que l'extension doit être petite et facilement installable. Les premières études de faisabilité ont montré que l'implémentation d'une telle extension est impossible sans la réécriture totale de l'analyseur syntaxique HTML (*HTML parser and renderer*). La direction de l'entreprise m'a alors demandé de concentrer mes efforts sur un autre projet: La réalisation d'un moteur de recherche pour le Monde Arabe, il s'agit de *Hannibal*. Ainsi, le travail présenté dans ce document concerne deux aspects:

- la construction d'un système de recherche et d'indexation pour les documents disponibles sur le Web du Monde Arabe avec une interface d'interrogation *via* HTTP, et
- la proposition d'une solution pour rendre possible l'affichage de documents en Arabe dans le navigateur Netscape Navigator.

Dans le premier chapitre, je vais expliciter les problèmes que j'ai rencontré avec l'ajout d'un support pour la langue Arabe, un aperçu des standards et technologies actuels et la solution proposée. Le second chapitre décrit l'extension réalisée. Le troisième chapitre introduit les techniques de recherche et d'indexation et le fonctionnement de Hannibal. Finalement, je décrit les extensions éventuelles et les améliorations possibles de mon travail.

Chapitre 1

Support de la Langue Arabe dans les Applications Internet

1.1 Introduction

La manipulation de l'Arabe dans les applications informatiques actuelles pose énormément de problèmes. Ces problèmes peuvent être casés dans trois grandes catégories [TAS90]:

- représentation des caractères Arabes,
- inconsistence résultant de la saisie de textes dans des langues différentes,
- choix ou définition d'un jeu de caractères bilingue approprié qui satisfasse les standards internationaux et les besoins spécifiés.

Tout comme pour la plupart des applications informatiques, les applications Internet se sont restreintes à l'utilisation du jeu de caractères US-ASCII (ISO 646). Ces restrictions étaient telles que certains vieux nœuds de transit de messagerie SMTP ont pris l'habitude de supprimer le huitième bit du corps des messages.

Le jeu de caractères US-ASCII est un standard ANSI qui représente les caractères sur 7 bits; son identification officielle IANA est `ANSI_X3.4-1968`. Une extension à ce jeu de caractères est le ISO 8859-1 (Latin-1) qui est un encodage sur 8 bits compatible avec US-ASCII. Le standard ISO 8859 est en fait une famille de jeu de caractères incluant:

```
ISO 8859-1 Latin 1
ISO 8859-2 Europe Centrale
ISO 8859-5 Cyrillique
ISO 8859-6 Arabe
ISO 8859-7 Grec
ISO 8859-8 Hebreu
ISO 8859-9 Turque
```

ISO 8859-1 englobe, en plus des caractères US-ASCII, la plupart des caractères spéciaux Européens.

ISO 8859-6 est un jeu de caractères identique à ASMO-708 qui a été publié par la *Arabic Standards and Metrology Organization (ASMO)*. Ce jeu a été adopté par Apple Computers dans leurs applications en Arabe sous Macintosh. Mais, Microsoft Corporation n'a pas adopté ce standard créant son propre encodage CP-1256 des caractères Arabes pour ses versions Arabes de Windows. ISO 8859-6 est enregistré à l'IANA en tant que `ISO-8859-6: 1987`.

De même, la *Institute of Standards and Industrial Research of Iran (ISIRI)* a conçu un standard pour un jeu de caractères persiques: ISIRI-3342 qui ne définit que des glyphes Arabes. Ce jeu de caractères n'est pas enregistré chez l'IANA.

1.2 Caractéristiques de la Langue Arabe

Pour le novice, la langue Arabe présente plusieurs caractéristiques inhabituelles. L'alphabet de base est constitué d'un jeu de 28 lettres représentant les consonnes et quelques voyelles longues. Ce jeu peut être étendu à 90 par d'autres formes et marquages. Les lettres Arabes, écrites dans des formes élégantes, ne différencient pas entre minuscules et majuscules.

Les applications et les plateformes qui manipulent l'Arabe n'ont pas été conçues pour faire autant. Les applications actuelles traitent du texte Latin et certaines ne traitent que le US-ASCII. À cause des différences fondamentales entre le Latin et l'Arabe, il est toujours difficile de manipuler proprement des textes Arabes dans un environnement Latin. Ces problèmes sont encore plus durs quand il s'agit de traiter à la fois l'Arabe et le Latin, ce qui est souvent le cas. En effet, les documents techniques Arabes mentionnent très souvent des mots Latins.

1.2.1 Méthodes de Saisie non Standardisées

Jusqu'à 1990, il n'y avait aucun standard bien établi pour la saisie des textes Arabes [TAS90]. Actuellement, un *mapping* Arabe sur les touches du clavier du PC commence à s'imposer. La plupart des applications destinées au traitement de l'Arabe dans des environnements Latins utilisent soit des *translitérations* soit des *transcriptions* [TAS90].

Une translitération est un mapping entre les caractères d'une langue et un jeu de symboles. Ainsi, toutes les représentations des caractères sur un ordinateur utilisent des translitérations entre des caractères d'une langue et des nombres selon le standard ISO/R 223-1961. La saisie des caractères Arabes fait recourt parfois à des translitérations entre les caractères Arabes et des caractères Latins. Le package ArabTEX [1], par exemple, ainsi que plusieurs autres outils linguistiques utilisent une translitération à base de US-ASCII [KlL93]. Le tableau 1.1 montre quelques exemples de translitérations.

Par opposition à la translitération la transcription consiste en une description phonétique de la prononciation d'une langue par des caractères dans une autre langue. Cette technique n'est utilisée que dans des applications linguistiques avancées. Le tableau 1.2 montre quelques exemples de transcriptions.

[1] ArabTEX est un package pour LATEX développé par Professeur Klaus Lagally de l'Université de Stuttgart, Allemagne [KlL93].

Translitérations	Texte Arabe
drs	درس
darrs	دَرّس
dirAsT	دِرَاسة
tadrIs	تَدرِيس

Tableau 1.1: Exemple de translitérations.

Transcriptions en Anglais	Texte Arabe
tamronn	تمر
tooness	تُونِس
thakiraton	ذَاكِرة

Tableau 1.2: Exemple de transcriptions.

1.2.2 Morphologie Complexe

Les langues Sémitiques (type Arabe et Hebreu) présentent des difficultés inhabituelles pour des analyses morphologiques automatiques. Alors que la plupart des langues construisent les mots par concaténations comme dans *im+poli+ment*, un mot Arabe comme (دَرَس) est considéré comme une racine de trois lettres (د ، ر ، س) utilisée dans un modèle *CaCaC* où *C* est une consonne (ou radical) et *a* une voyelle. De la même manière on obtient (تَبَرَّج) de (ب ، ر ، ج) selon *CaCaCCaC* [KRB].

Il y a près de 5000 racines et quelques 400 modèles. Chaque racine ne peut être combinée qu'avec un petit sous-ensemble de modèles (17 en moyenne).

1.2.3 Physiologie Complexe

Par opposition aux langues Latines, où les caractères sont dessinés sur une même ligne de base (*baseline*) plate, l'Arabe est dessiné avec des courbes curvlignes élégantes, ce qui fait que la création de glyphes Arabes est plus complexe.

L'Arabe utilise aussi un grand nombre de marques diacritiques (اَلتَّشكِيل), qui ressemblent en quelque sorte aux accents utilisés par certaines langues Européennes. De tels symboles sont utilisés pour marquer les voyelles courtes et pour accentuer ou réduire le son d'une lettre. Ces symboles peuvent être combinés et dessinés en dessous ou au dessus des lettres pour produire des effets phonétiques divers.

Les lettres de la langue Arabe, qui a été écrite à la main pendant plusieurs siècles, dépendent de leur contexte. En effet, la forme d'une lettre dépend de sa position dans le mot et des caractères adjacents.

Les ligatures qui sont déjà un problème dans les langues Latines, sont encore plus difficiles à gérer en Arabe. Même dans les langues Latines, peu d'outils de traitement de textes intégrent des tables de ligatures. En Latin, il "suffit" d'insérer un tiret entre deux syllabes et revenir à la ligne. En Arabe, les lettres se combinent pour produire de nouvelles formes (figure 1.1).

Son	Lettre Arabe avec diactitiques
s	سْ
sa	سَ
su	سُ
si	سِ
ss	سّ
ssa	سَّ
ssu	سُّ
ssi	سِّ

Tableau 1.3: Symboles diacritiques.

غَدَر
رَاغِبٌ
لُغمٌ
لَدَغَ
بَالِغٌ

Tableau 1.4: Différentes formes d'un caractère, cas du (غ).

لَا $\Longrightarrow$ ل+آ

ضِمن $\Longrightarrow$ ضِـمن

Figure 1.1: Exemple de ligatures.

Caractère en entrée	Chaîne à afficher
(initialement)	اَلمَسَافَةُ تُسَاوِي ـ
1	اَلمَسَافَةُ تُسَاوِي ـ١
2	اَلمَسَافَةُ تُسَاوِي ـ١٢
,	اَلمَسَافَةُ تُسَاوِي ـ١٢,
5	اَلمَسَافَةُ تُسَاوِي ـ١٢,٥
4	اَلمَسَافَةُ تُسَاوِي ـ١٢,٥٤
m	اَلمَسَافَةُ تُسَاوِي ١٢,٥٤م ـ

Tableau 1.5: Problèmes de bidirectionnalité.

1.2.4 Bidirectionnalité

L'Arabe est écrit de droite à gauche, mais les nombres sont, dans beaucoup de pays, écrits et lus de gauche à droite. Ceci peut causer des ennuis dans la saisie, comme le montre le tableau 1.5 où "_" désigne la position courante du curseur.

1.3 Problématique de l'Internationalisation des Applications

Les premières applications sur Internet n'ont pas été conçues avec un soucis d'internalionalisation (ou i18n[2]). Il s'agissait d'un Internet pour les Américains. Avec les développements glorieux de l'informatique, les industriels du logiciel et du matériel, qui vendent leurs produits sur le marché international, se sont confrontés aux problèmes de diversité des langues et des cultures. Apple Computers, par exemple, a distribué un MacOS pour les Américains, un pour les Chinois, un pour les Arabes, un autre pour les Israëliens, etc. Mais il y a là des problèmes d'interopérabilité, où un utilisateur (supposons un Espagnol) a à faire plusieurs configurations pour pouvoir lire un document écrit dans une langue étrangère (soit le Coréen).

1.3.1 Tendances ISO

Le standard ISO/IEC[3] 10646-1 définit un jeu de caractères multi-octets appelé *Universal Character Set (UCS)* qui englobe la presque totalité des caractères du monde [FrY98]. Il s'agit en fait de deux standards: UCS-4 à quatre octets par caractère et UCS-2 à deux octets par caractère.

Malgré qu'il soit un standard, ISO/IEC 10646-1 n'a pas connu le succès qu'il mérite à cause de difficultés d'implémentation. Cependant, il est à noter la création par un consortium de constructeurs informatiques, de l'organisme Unicode[4] qui s'est chargé de définir un standard basé sur UCS-2 avec des propriétés et des détails supplémentaires

[2] Un *i* suivi de 18 lettres et un *n*.
[3] http: //www.iso.org
[4] http: //www.unicode.org

pour les développeurs d'applications. Ce standard Unicode a connu, depuis, un succès fulgurant et a été adopté par ISO.

Le problème avec UCS est qu'il est incompatible avec les applications et les protocoles en cours d'utilisation, de ce fait il est apparue la nécessité de définir des encodages d'UCS pour permettre son stockage et sa transmission sans grandes difficultés et incompatibilités. L'*UCS Transformation Format (UTF)* est un standard de représentation des caractères Unicode. UTF-8 [FrY98] est la variante la plus élaborée d'UTF.

Remarquons enfin que, pour exprimer la bidirectionnalité, la norme ISO/IEC 10646-1 réserve cinq caractères (202A à 202E) pour contrôler la direction du texte.

1.3.2 Tendances IETF/W3C

Le World Wide Web Consortium[5] (W3C) est une entité qui travaille, sous l'authorité de l'IETF[6], sur la standardisation du Web (HTML et HTTP) [HAL96]. L'*Internet Engineering Task Force* (*IETF*) et le W3C ont commencé à considérer très sérieusement les problèmes d'internationalisation des application depuis quelques années. Ainsi, trouve-t-on dans [FrY97] des extensions pour HTML 2.0 [TBL95] qui suppriment la restriction à ISO-8859-1. Le protocole HTTP, quant à lui, a évolué pour supporter la spécification du type de contenu à la manière de MIME. Le type MIME text/html possède, dans le contexte qui nous intéresse, un attribut "charset" (qui vaut ISO-8859-1 par défaul [TBL95]).

Pour les données textuelles (celles qui sont les plus susceptibles de causer des problèmes), [FrY97] exige que tous les protocoles mentionnent le jeu de caractères utilisé, et qu'ils puissent "comprendre" UTF-8 comme format de transfert pour Unicode. Ainsi, un agent utilisateur conforme aux spécifications de l'IETF ne doit plus ignorer le paramètre "charset", doit pouvoir reconnaître toutes les références aux caractères définis dans ISO-10646-1 et pouvoir utiliser l'algorithme de présentation BIDI pour les textes bidirectionnels. De plus, un attribut DIR est ajouté à toutes les balises renfermant des blocs de texte pour résoudre le problème de bidirectionnalité (qui est est supposé être traité selon les spécifications de Unicode). Par exemple,

```
<SPAN DIR=LTR> AB <SPAN DIR=RTL> xy <SPAN DIR=LTR> CD
</SPAN> wz </SPAN> EF </SPAN>
```

doit produire,

AB yx CD zw EF

SPAN étant une balise renfermant un texte (du type <TITLE>, par exemple).

Le problème de bidirectonnalité du texte dans la messagerie SMTP a été traité par H. NUSSBACHER dans [HNu93] où il définit les types de média (types MIME) avec les spécifications de direction pour l'Arabe et l'Hebreu:

```
text/plain;  charset=ISO-8859-6
text/plain;  charset=ISO-8859-6-e
text/plain;  charset=ISO-8859-6-i
text/plain;  charset=ISO-8859-8
```

[5] http: //www.w3.org
[6] http: //www.ietf.org

```
text/plain;  charset=ISO-8859-8-e
text/plain;  charset=ISO-8859-8-i
```

Ceci définit trois manières pour spécifier la directionnalité:

- implicite (suffixe i): la directionnalité implicite est une méthode de présentation où la direction est déterminée par un algorithme complexe qui reconnaît la direction par le type des caractères. Par exemple, un agent utilisateur peut considérer que les caractères dont le code est inférieur à 127 sont des caractères Latins et doivent donc s'écrire de gauche à droite alors que les caractères dont le code est supérieur à 127 sont des caractères Arabes/Hebreux et doivent s'écrire de droite à gauche,
- explicite (suffixe e): la directionnalité est spécifiée par des séquences de contrôle à l'interieur du texte,
- visuel (par défaut): c'est la directionnalité la plus simple, les agents utilisateur ne font aucun traitement et affichent le texte sans considérations de directionnalité (comme s'il était unidirectionnel). La résolution des problèmes de direction est laissée à la charge des outils auteurs.

1.4 Les Agents Utilisateur

Malgré la disponibilité des standards, peu d'applications les supportent. Dans le cas d'Unicode, qui se prête à devenir incontournable dans le futur, seuls les systèmes d'exploitation modernes offrent un support plus-ou-moins complet. Le pire est que les applications qui tournent sur ces plate-formes n'en tirent même pas profit. De plus, les polices Unicode ne sont pas encore à la portée du grand public, certains grands éditeurs n'offrent que des versions expérimentales[7].

En effet, les navigateurs n'affichent pas correctement les documents Arabes parce que, primo: ils ne savent pas afficher de droite à gauche, et secondo: ils ne peuvent pas faire la jointure des caractères.

1.4.1 SakhrSoft Sindbad

Sakhr[8] est une compagnie du Groupe Al-Alamyah qui travaille sur les applications informatiques (matériel et logiciel) en Arabe depuis une quinzaine d'années. Sakhr à écrit Sindbad[9] qui est un navigateur capable de présenter des documents HTML en Arabe, il supporte les encodages ISO 8859-6, CP 1256 et UTF-8 (Unicode).

Sindbad est une refonte de Navigator, il ne s'agit pas de plug-in, mais d'une modification de certaines DLLs pour supporter les spécificités de la langue Arabe.

Ainsi, Sindbad est plutôt un navigateur à part entière basé sur l'interface de Netscape Navigator. C'est une sorte de remanipulation de certaines composantes de Navigator qui ne fonctionne que sous Windows 95.

[7] C'est le cas de Bitstream qui offre CyberBit, une police Unicode TrueType à www.bitstream.com/cyberbit.html

[8] http: //www.sakhr.com

[9] Sindbad 3 et 4 sont disponibles gratuitement depuis la page d'acueil de Sakhr.

J'ai testé Sindbad 4 (archive de 1.4Mb) sur Windows 95 et j'ai constaté qu'il fonctionne bien et produit une présentation adéquate des documents. A remarquer, seulement, que Sindbad ne peut être aussi stable et solide que Navigator et qu'il lui faudrait être porté sur d'autres plate-formes que Windows.

1.4.2 Alis Tango

Alis Technologies[10] est une compagnie Canadienne spécialisée dans les applications informatiques multilingues (matériel et logiciel).

Tango[11] est un navigateur multilingue basé sur la version Mosaic de Spyglass et produit pas Alis Technologies. Il est disponible uniquement sous Windows 3.1/95 et supporte près de 85 langues dont l'Arabe dans plusieurs encodages.

J'ai testé une version de Tango sur Windows 95, et j'ai constaté qu'elle gère assez bien les documents multiligues même bidirectionnels. Cependent, son interface n'est pas soignée et s'avère austère pour un habitué de Netscape Navigator.

1.4.3 AccentSoft Multilingual Mosaic

Accent Software[12] est une compagnie Européenne spécialisée dans les applications multilingues, surtout les traitements de textes et qui a entrepris la réécriture de quelques modules de NCSA Mosaic[13] pour le rendre multilingue. Multilingual Mosaic ne tourne que sur Windows, je n'ai pas eu l'occasion de le tester à cause de problèmes avec les serveurs FTP de Accent.

1.4.4 Netscape Communicator avec support Unicode

Il s'agit sûrement de la meilleure approche pour afficher correctement des documents en Arabe. Cependent, le problème reste entier à cause de la non disponibilité, jusqu'à la date d'écriture de ces lignes, des polices Unicode sur diverses plateformes.

J'ai téléchargé une version expérimentale d'une police TrueType distribuée par Bitstream[14] (Bitstream Cyberbit) pour l'essayer avec Netscape Communicator. Sous Windows 95, Cyberbit affiche les glyphes Arabes mais dans la mauvaise direction et sans joindre les caractères, c'est à dire qu'elle ne fait pas mieux que les polices ISO 8859 classiques. Le problème est sûrement dû à l'implémentation Unicode de Windows 95 et je suppose que sous Windows NT, Cyberbit s'affichera correctement.

1.4.5 AraMosaic et PMosaic

Persian Mosaic ou PMosaic est une refonte de NCSA Mosaic pour le support des caratères Arabes, il s'agit du travail d'étudiants Iranians de Stanford. PMosaic est conçu pour afficher le jeu de caractères ISIRI 3342.

[10] http: //www.alis.com
[11] Disponible en version d'évaluation à http: //www.alis.com/P_NET/P_MCP/P_MCP.EN.HTML
[12] http: //www.accentsoft.com
[13] http: //www.ncsa.uiuc.edu
[14] http: //www.bitstream.com

AraMosaic est développé par LangBox International[15], une compagnie du Groupe Spartacus[16]. Tout comme PMosaic, AraMosaic est une modification de NCSA Mosaic mais avec plus de possibilités que PMosaic. AraMosaic sait afficher les documents dans les deux directions, gérer des textes Latins dans un document Arabe et interpréter plusieurs jeux de caractères: ISIRI 3342, CP 1256 et ISO 8859-6.

PMosaic et AraMosaic sont disponibles sous la plupart des systèmes Unix gratuitement. J'ai eu l'occasion de les tester, ils offrent des présentations assez correctes des documents Arabes, seulement ils sont peu stables.

[15] http: //www.spartacus.com/langbox
[16] http: //www.spartacus.com

Chapitre 2

Le Plug-in Hannibal Viewer

Cette première partie s'intéresse à l'implémentation d'une extension pour un navigateur pour qu'il puisse afficher correctement des documents Arabes.

Une première approche consisterait à reprendre le code source d'un navigateur existant (libre de droits) et d'ajouter les fonctions nécessaires. Les navigateurs qui se prêtent à de telles pratiques sont W3C Amaya, NCSA[1] Mosaic et Netscape Navigator, mais:

- W3C amaya est un produit expérimental et non déstiné au grand public, il fût développé pour implémenter les spécifications du W3C pour HTML et HTTP,
- Ce travail a été déjà fait pour NCSA Mosaic (voir plus haut): PMosaic et AraMosaic sont des modifications de Mosaic qui incluent un support pour la langue Arabe,
- Netscape Navigator est sûrement le meilleur client mais Netscape n'a mis son source à la disposition du public[2] que tout dernièrement (1^{er} avril, 1998), et comme l'archive compressée du source de Netscape fait une dizaine de Mb, il aurait été aberrant de tenter ce travail en l'espace d'un mois.

Netscape à mis au point, depuis la version 2 de Mozilla[3], une technologie pour l'extension du navigateur par des modules logiciels développés par des tiers et appelés *plug-ins*. Cette technologie s'avère dans, notre cas, la plus convenable (voir dicussions plus loin). Ainsi, dans ce qui suit, je donne une introduction à la technologie des plug-ins ainsi que l'implémentation de l'extension en question que j'ai appelé *Hannibal Viewer*.

2.1 Technologie des Plug-ins

Pour des raisons de disponibilité de documentation, j'ai choisi de me concentrer sur un produit particulier qui est Netscape Navigator. De plus Netscape Navigator était et est toujours le navigateur le plus populaire sur pratiquement toutes les plate-formes.

[1] National Center for Supercomputer Applications, Université d'Illinois.
[2] Voir dans http: //www.mozilla.org
[3] Le navigateur de Netscape s'appelle, en fait, *Mozilla* mais il est écrit *"Netscape"*.

L'architecture des plug-ins de Netscape Navigator est basée sur des modules de code à chargement dynamique[4]. Ces modules résident dans un répertoire ou sous-dossier[5] nommé PLUGINS que Navigator lit durant son démarrage. Chaque module dispose d'une ressource qui détermine le type de média (type MIME) qu'il sait gérer, lorsque Navigator rencontre ce type MIME intégré dans une page Web à travers HTML ou en tant que fichier séparé, il charge le module approprié.

2.1.1 Introduction à MIME

MIME est initialement créé, vers 1992, pour gérer des types de média divers dans la messagerie électronique. Avec le développement du World Wide Web, il a été rapidement adopté par les serveurs et les clients HTTP [BoF93].

L'extension d'un fichier est un paramètre important dans la détermination du type MIME de son contenu. Par exemple, un fichier contenant une séquence audio au format Microsoft WAV et nommé `dialog.wav` ne renferme pas d'informations relatives au type MIME de son contenu. La seule information est alors son extension (wav) qui signifie ici le type MIME audio/x-wav [ZaO].

Lorsqu'un nouveau type de média se crée, les développeurs peuvent créer de nouveaux types MIME. Ainsi, si une compagnie invente un nouveau format vidéo FastVideo par exemple, elle peut lui associer un sous-type MIME `video/x-fastvideo`. Le nouveau nom du sous-type doit être déposé à l'*Internet Assigned Numbers Authority (IANA)*.

Un nouveau type, comme `text` et `video`, requiert la plublication d'un RFC et doit donc passer par le processus de standardisation habituel dans d'Internet.

2.1.2 La Balise EMBED

Pour les documents intégrés dans des pages Web, la balise EMBED de HTML informe Navigator sur la taille de la fenêtre à réserver au plug-in à l'intérieur de la page. Cette fenêtre est créée par Navigator pour le compte du plug-in. Ensuite Navigator passe au plug-in un *handle* sur cette fenêtre pour dessiner et gérer les événements. La balise EMBED doit ressembler à quelque chose du genre:

<EMBED TYPE=video/mpeg SRC="dialog.mpg" WIDTH=300 HEIGHT=134>

Ceci permet au navigateur de "comprendre" qu'il doit ouvrir une fenêtre de dimensions 300x134 pour y afficher une vidéo contenue dans le fichier dialog.mpg. Le type MIME spécifié explicitement par l'attribut TYPE informe donc le navigateur qu'il doit lancer le plug-in approprié.

2.1.3 Architecture d'un Plug-in

Un plug-in est un module de code partagé qui communique avec le navigateur selon le schéma simple explicité dans la figure 2.1 [ZaO]. Les deux parties définissent des interfaces standards (voir annexe D):

- le navigateur définit des méthodes dont les noms commencent par NPN_ c'est à dire une méthode d'interfaçage des Plug-ins Netscape définie du côté Navigator,

[4] Il s'agit de librairies DLL sous Windows et de librairies partagées *shared object code (.so)* sous Unix.
[5] L'appellation diffère selon les plateformes

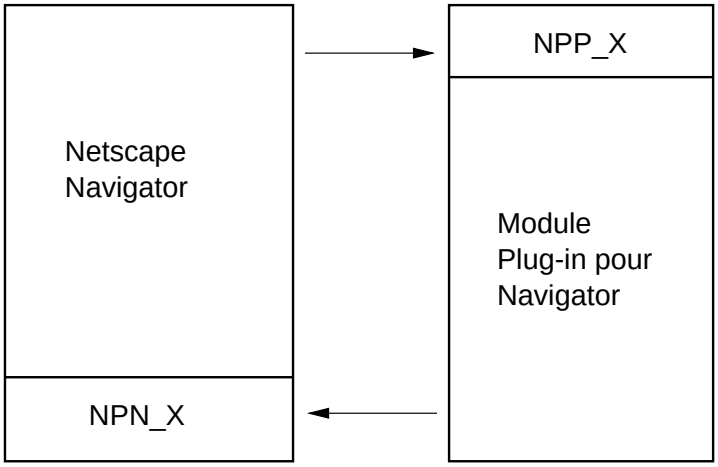

Figure 2.1: Modèle simple des plug-ins.

- le plug-in doit, à son tour, définir des méthodes commençant par NPP_

Avec Navigator 3.0, Netscape a amélioré son API et a introduit LiveConnect qui est une technologie pour intéger des plug-ins avec des appliquettes Java et des scripts JavaScript. Avec ces extensions, le modèle des plug-ins devient plus riche et plus complexe (figure 2.2).

Lors de son démarrage, Navigator consulte la liste des plug-ins installés et les interroge pour savoir les types de média qu'ils gèrent. Un plug-in n'est chargé que si besoin est.

La fenêtre du plug-in reçoit les événements générés par l'utilisateur (clicks, touches, etc.) et par le gestionnaire de fenêtres (redimentionnement, exposition, etc.). Le plug-in peut traiter les événements pour personnaliser son comportement et relayer à Navigator ceux qui subiront le traitement par défaut [ZaO]. Le modèle de gestion des événements est explicité dans la figure 2.3.

2.1.4 Alternatives aux Plug-ins

En fait, il n'y a pas que les plug-ins pour gérer les types de médias non supportés par le navigateur. Une première approche est l'utilisation d'applications externes dites *helpers*, une seconde est le recourt aux appliquettes Java.

Un helper est une application normale qui sait gérer et éventuellement éditer un type de média donné. Par exemple, il arrive souvent d'associer le type MIME `application/postscript` à une application helper externe. De telles applications peuvent recevoir le document en question directement depuis le serveur HTTP *via* des sockets ou bien depuis le navigateur qui télécharge le document et leur passe son nom en paramètre. L'application helper la plus connue est certainement Real Audio Player de Progressive Networking qui est capable de jouer un flux audio en temps réel sur Internet. Le problème avec les applications helper est qu'elles peuvent parfois avoir un aspect désagréable par rapport aux plug-ins qui, eux, s'affichent dans la même fenêtre que le document HTML [ZaO]. En effet, avec un helper la notion de document hypermédia intégré n'est plus conservée.

L'approche qui coonsiste à recourir à des appliquettes Java pour gérer les objets envoyés par le serveur a l'avantage de ne pas faire de suppositions sur la partie cliente.

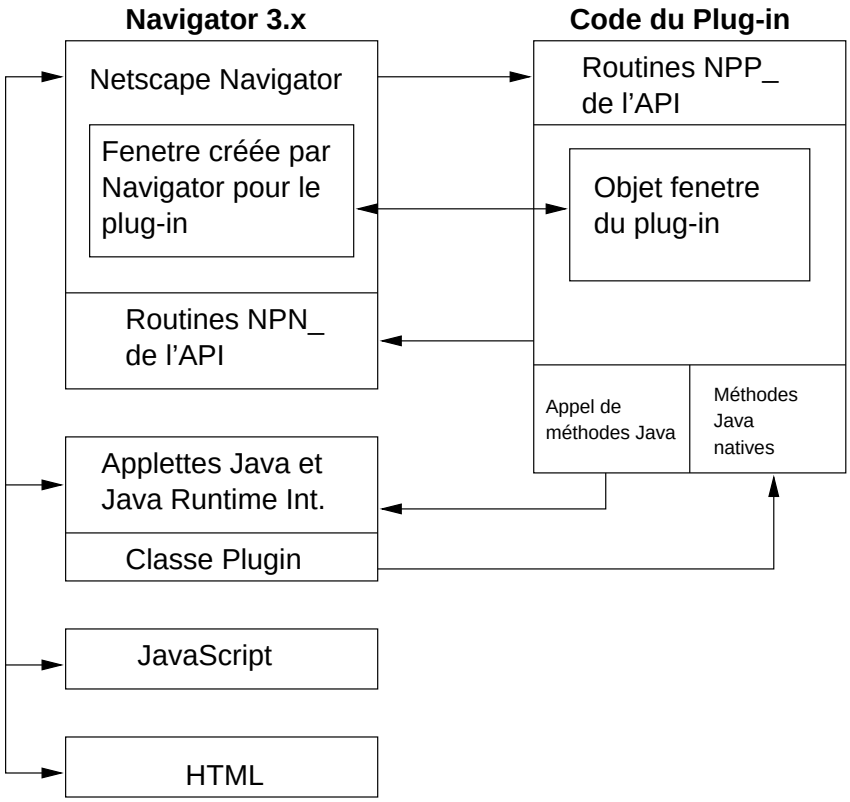

Figure 2.2: Les extensions LiveConnect.

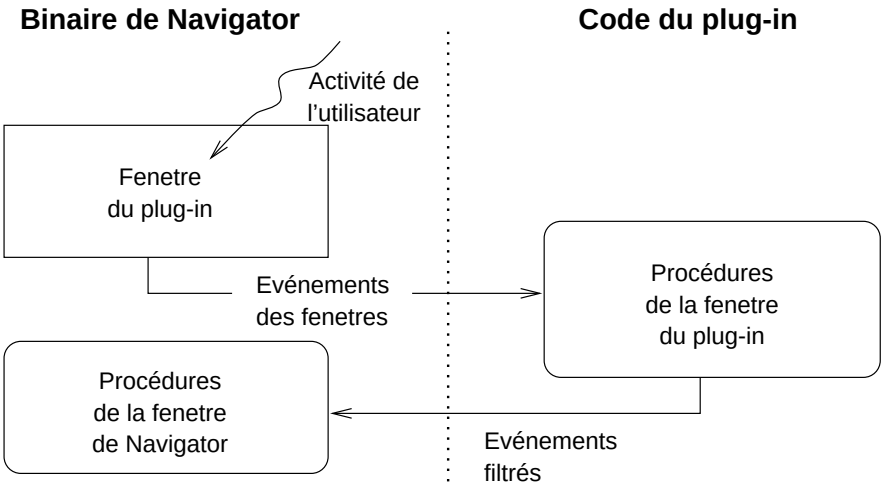

Figure 2.3: Gestion des événements.

	Plug-in	Helper	Appliquette
Facilité de mise en œuvre	Bonne	Mauvaise	Moyenne
Vitesse d'exécution	Bonne	Moyenne	Mauvaise
Aspect visuel	Bon	Mauvais	Mauvais
Portabilité	Moyenne	Mauvaise	Bonne

Tableau 2.1: Les trois approches comparées.

En effet, cette technique ne suppose pas que le client dispose d'une application donnée ou d'un plug-in, et de ce fait,

- une appliquette est généralement assez lourde à charger, surtout s'il s'agit de manipulations graphiques intenses,
- il faut la recharger chaque fois qu'on a envie de visualiser le document, et,
- elle est lourde à s'exécuter à cause de l'interprétation.

2.1.5 Comparatif

Le tableau 2.1 présente un comparatif sommaire des trois approches: appliquette, helper et plug-in.

2.2 Spécifications et Problèmes de Conception de Hannibal Viewer

Le plug-in à réaliser doit pouvoir afficher des documents en Arabe au sein du navigateur (Netscape Navigator précisement). Cette spécification peut être appréhendée de plusieurs manières, il faut donc se décider sur les points suivants:

- nature des documents présentés: simple texte (*plain text*) ou du texte formaté (HTML)?
- jeu de caractères: quels sont les jeux de caractères supportés?
- police: faut-il utiliser une police par jeu de caractère ou les mapper tous dans le jeu de caractères d'une police unique?
- type de média: quel sera le type MIME géré?

2.2.1 Le Type de Média Géré par Hannibal Viewer

Netscape Navigator supporte plusieurs encodages dont la plupart de la famille ISO 8859 et les encodages Chinois/Japonais/Koréens. L'encodage User-defined est laissé pour les usages particuliers des utilisateurs, mais l'utilisation de cet encodage paramétrable ne convient pas à l'Arabe car Navigator n'inclut pas un support pour l'écriture de droite à gauche ou la jointure des caractères successifs.

En principe, dans notre problème, il convient d'implémenter un plug-in qui gère le type MIME "text/html; charset=iso-8859-6". Le problème avec cette approche est qu'il faudra écrire un analyseur syntaxique HTML (parser) qui sache faire l'affichage de toutes les balises (rendering), il s'agit en fait d'un navigateur à part entière. Je n'ai pas écrit un tel plug-in pour deux raisons:

- cette solution existe déjà et je ne voulait pas proposer, dans le cadre de ce travail, une solution existante,
- la réalisation d'un tel plug-in aurait été très complexe et trop longue,

Ecrire un plug-in qui gère le type de média text/html est en quelque sorte réécrire un navigateur. En effet, l'objectif de ce travail est d'*ajouter* au navigateur la capacité d'afficher des documents Arabes, ce qui revient à ajouter un support pour l'écriture de droite à gauche et pour la jointure des caractères. Mais, le modèle des plug-ins ne permet que la prise en charge de tout un type de média, ainsi il n'est pas possible de ne gérer que les textes Arabes dans une page HTML.

L'écriture d'un parser HTML n'est pas facile à cause des technologies propriétaires des constructeurs. D'ailleurs aucun des deux navigateurs les plus populaires (Netscape Navigator et Microsoft Internet Explorer) ne supporte toutes les balises de HTML 3.2. Les difficultés majeures dans l'écriture d'un tel outil sont:

- la gestion d'une multitude d'éléments de l'interface graphique (*Graphical User Interface, GUI*) tels les fontes, les couleurs, les bitmaps, les curseurs, les barres de défilement, etc.,
- la gestion des formulaires et les problèmes de la saisie multilingue et de la configuration des claviers qui sont derrière,
- la gestion des tâches parallèles (comme les animations, l'affichage en cours de téléchargement, etc) par des threads,
- la gestion de tous les formats graphiques standards (GIF, JPEG, XPM, etc) et donc l'implémentation des algorithmes de décompression et d'affichage,
- l'exécution des appliquettes Java sous la machine virtuelle Java disponible sous Navigator,

La quasi totalité de ce travail est déjà faite par Netscape! , et en utilisant un tel plug-in l'utilisateur aura deux navigateurs qui tournent sur sa machine, et comme le binaire de Navigator fait déjà près de 7Mb, la machine cliente aura des problèmes de surcharge.

En plus de ces problèmes, j'ai eu beaucoup de peine à trouver de la documentation sur les plug-ins. En effet, le kit de développement des plug-ins de Netscape ne contient qu'un descriptif des fonctions de l'API.

Pour réduire la taille du problème, j'ai décidé de ne traiter dans Hannibal Viewer que le type de média text/plain. Ce type de média peut être paramétré par l'attribut "charset", qui vaut par défaut US-ASCII [BoF93].

2.2.2 Le Jeu de Caractères

Le jeu de caractères le plus standard est sans doute ISO-8859-6. Microsoft CP-1256 est propriétaire et ASMO 449 et ISIRI 3342 ne définissent pas de glyphes Latins.

La technologie des plug-ins de Netscape permet à un plug-in de gérer plusieurs types MIME. En effet, lors de la notification d'un nouveau flux, Navigator signale le type MIME dans le paramètre `type` de `NPP_NewStream()`. La police à utiliser pour l'affichage a un encodage donné, et pour utiliser plusieurs encodages il faudra soit utiliser plusieurs polices (une par encodage) soit utiliser une unique police et faire des conversions vers un encodage particulier.

J'ai choisi de gérer les types de média suivant [IAN]:

- `text/plain; charset=ISO-8859-6` (alias `ASMO-708`)
- `text/plain; charset=ISO-8859-i`.

Des versions prochaines de Hannibal Viewer supporteront:

- `text/plain; charset=ISIRI-3342`,
- `text/plain; charset=CP-1256`
- `text/plain; charset=ASMO_449`.

2.3 Réalisation

2.3.1 Plate-forme

Dans un but de portabilité et d'ouverture de l'outil réalisé, j'ai opté pour une plate-forme de développement X Window [AlJ] sur UNIX sachant que le navigateur utilisé est Netscape Navigator. Le plug-in est écrit en C, sous Linux, et compilé avec le compilateur C/C++ de GNU version 2.7.2.1.

2.3.2 Implémentation

Après une étude approfondie de l'API des plug-ins de Netscape (voir annexe D), j'ai construit une squelette de plug-in que j'ai enrichie au fur et à mesure de mes lectures.

Au lancement, le plug-in s'enregistre chez Navigator pour gérer les types MIME cités plus haut. À l'arrivée d'un nouveau flux, le plug-in retourne dans stype la valeur `NP_FILEONLY` pour déclarer qu'il ne gèrera pas le flux "au vol". Navigator devra donc enregistrer le document en entier dans son cache et passer un descripteur vers le plug-in. La figure 2.4 montre les appels échangés entre Navigator et Hannibal Viewer pour traiter le document.

À chaque demande de mise à jour de la fenêtre de Navigator (sur redimentionnement de la fenêtre, traitement d'un nouveau flux, etc) le plug-in alloue un block mémoire avec `NPN_MemAlloc()` pour traiter le document ainsi qu'une ressource de type police de caractères. La police utilisée dans le plug-in est une police *Naskh* (نسخ), c'est la même police fournie avec AraMosaic et conçue par LangBox, Int.

Les fonctions principales de Hannibal Viewer sont:

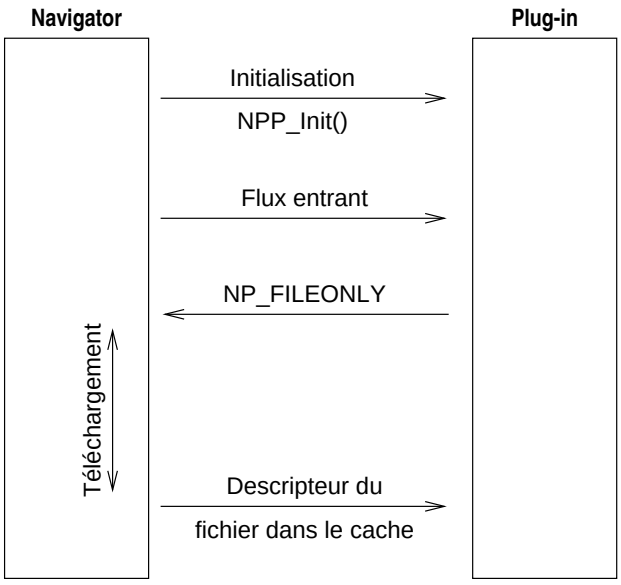

Figure 2.4: Schéma de communication plug-in-Navigator

- `HAN_GetChar()`: lit un caractère dans le tampon à l'index spécifié et le retourne,
- `HAN_MakeString()`: construit une chaine qui ne dépasse pas les bords de la fenêtre,
- `HAN_RevertString()`: inverse une chaine de caractères, n'est appelée que lorsqu'il s'agit de type text/plain; charset=ISO-8859-6-i,
- `HAN_JoinChars()`: applique les règles d'écriture de la langue Arabe pour joindre les caractères selon leur position relative,

Le source est très largement commenté et peut ainsi être étendu sans difficultés moyennant la compréhension de la technologie des plug-ins. L'algorithme de traitement du document est le suivant:

```
Allocation de ressources
Tant que non Terminé
  ContructionChaîne()
  Si Document fini, quitter
  Joindre les caractères de la chaîne
  Inverser la chaîne
  Afficher la chaîne
Fin

Procédure ConstructionChaîne()
  Si chaîne vide, quitter
  Pour les caractères de 0 à long(chaîne)
    Si le car. courant se fusionne à LAM et car. préc. est LAM
      car. préc. = ISO_MADDA_LAM
      car. courant = FormeAvecHAMZA(car. courant)
    Sinon
      Si (car. cour. modifie son prédecesseur) et (car. préc. ne se fusionne pas à LAM)
```

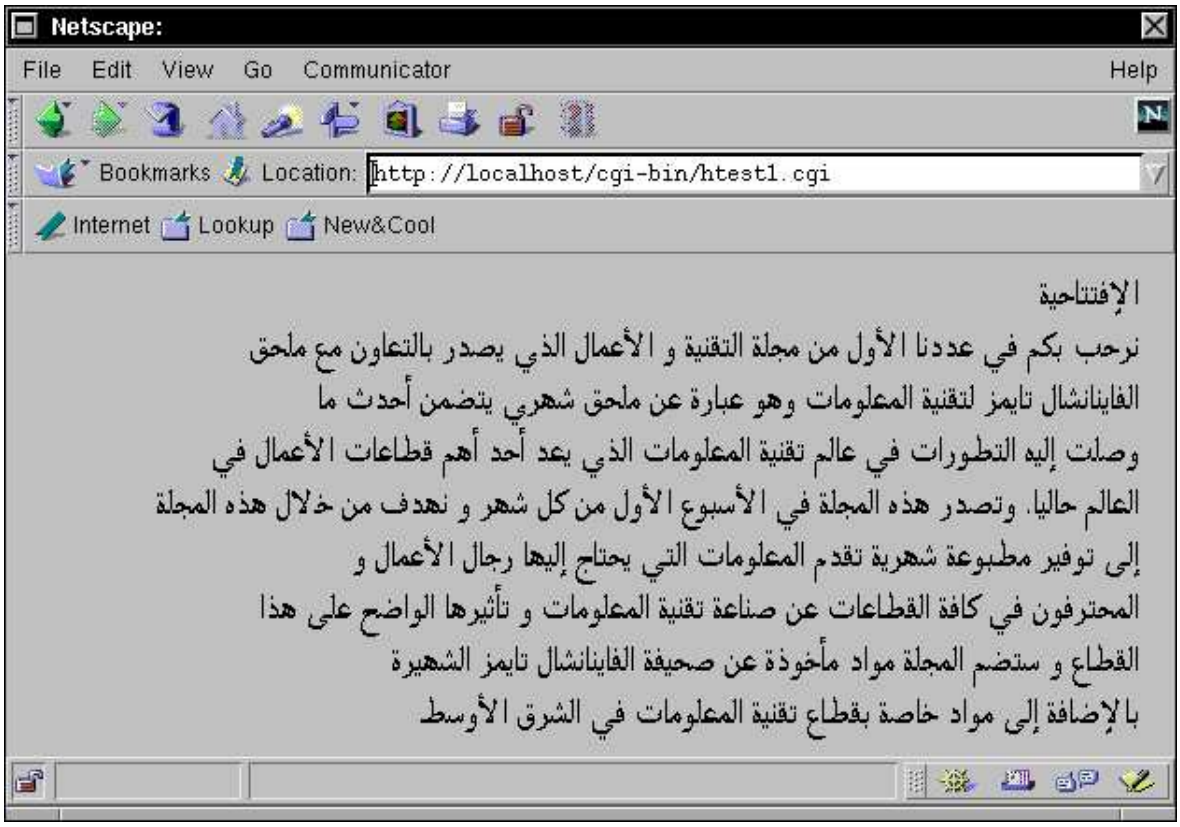

Figure 2.5: Recopie d'ecran de Hannibal Viewer

```
      car. préc. = FormeMilieu(car. courant)
    Fin Si
    car. courant = FormeFinale(car. courant)
  Fin Si
  car. préc. = car. courant
 Fin Pour
Fin Procédure
```

La figure 2.5 montre l'aspect de l'affichage du plug-in.

2.3.3 Communiquer le Type de Média

Pour que Navigator puisse lancer le plug-in, il doit être informé sur le type de média du document traité. Dans HTML, il est possible de spécifier la META balise HTTP-EQUIV pour indiquer le type MIME comme dans:

<META HTTP-EQUIV="Content-Type: " CONTENT="text/html; charset=iso-8859-6">

Il est aussi possible de passer le paramètre jeu de caractères dans une balise META ordinaire:

<META NAME="charset" CONTENT="iso-8859-6">

La balise HREF de HTML peut spécifier l'encodage du document sur lequel pointe le lien, l'ajout de l'attribut CHARSET force l'encodage du document référencé par le lien à la valeur spécifiée.

<a CHARSET="iso-8859-6" href="http: //www.hannibal.tn/docs/FAQ.ar.html"> Hannibal FAQs

Ces techniques ne sont utilisables qu'avec des documents HTML et ne peuvent pas servir dans notre cas. La meilleure manière de spécifier le type de média est de le communiquer directement par HTTP (*in-the-wire*), le paramètre `Content-type` de l'entête HTTP spécifie le type de média tout comme dans MIME. Le problème avec cette solution est que seul le serveur HTTP peut écrire l'entête, c'est à dire qu'il n'est pas possible de passer l'information sur le type de média dans le document.

Pour accéder à l'entête HTTP, il est possible d'utiliser des scripts CGI qui renvoient toute la réponse HTTP (entête+document). MIME sait gérer des correspondances entre une extension de fichier et le type de MEDIA, ces correspondances sont stockées dans un fichier `mime.types` accessible au serveur. En se basant sur ce fichier, le serveur HTTP peut revoyer le champ `Content-Type` correctement.

Chapitre 3

Chercher sur le Web

3.1 Motivations et Etat de l'Art

La seconde partie de ce travail s'intéresse à un service de recherche dans le Monde Arabe. Ce travail est motivé par plusieurs facteurs. En effet, les outils du genre Gopher et WWW ont rendu facile la publication et la consultation de documents sur Internet. Ceci a entraîné une croissance exponentielle dans la taille des informations et dans leur diversité. Le problème auquel est confronté l'utilisateur est l'accès à l'information pertinente. Pire encore, des prévisions du groupe Gartner affirment que le volume croissant d'informations menace de paralyser les entreprises d'ici l'an 1999 [KAM97].

Pour faire face à cette sur-information, certaines compagnies ont créé des services de recherche de documents communement connus comme des *moteurs de recherche*. Un moteur de recherche dans le sens le plus générique est un programme qui tourne sur un site connu et qui maintient une liste de références de documents, l'interrogation se fait en général *via* HTTP.

3.1.1 Les Moteurs de Recherche Comparés

L'appelation moteur de recherche couvre, par abus, les vrais moteurs de recherche et les répertoires. La différence réside surtout dans la manière dont les listes sont construites [AlL98].

- Moteurs de recherche: les moteurs de recherche, tels HotBot[1] et AltaVista[2] , créent leurs listes automatiquement. Ils parcourent le Web et les utilisateurs cherchent dans ce qui a été trouvé. Si les pages Web changent, les moteurs de recherche s'en apperçoivent, après un délai, et mettent à jour leurs indexes. Avec ce genre d'outils, les titres des pages, les occurrences des mots et d'autres éléments influent sur la manière dont le document est listé.

- Répertoires: les répertoires, comme Yahoo! [3] , dépendent des humains. Les administrateurs fournissent, à la main, des descriptions de leurs sites, les recherches se font sur les descriptions fournies. La modification ou même la disparition du site n'influe pas sur son référencement dans le répertoire.

- Moteurs de recherche hybrides: il s'agit d'une combinaison des deux approches. Les administrateurs fournissent leurs URLs pour la révision que ce soit par des

humains ou par des outils logiciels comme ceux des moteurs de recherche. Par opposition aux répertoires, il n'y a pas de garantie d'être listé.

3.1.2 Complexité des Moteurs de Recherche

Dans un site ordinaire, les serveurs HTTP ne font que délivrer des documents HTML et exécuter de temps en temps quelques scripts CGI. Les surcharges, qui sont dues à trop de connexions simultanées, sont très rares.

Un site hébergeant un moteur de recherche est beaucoup plus sollicité qu'un site ordinaire. Ceci est dû à plusieurs facteurs:

- le site est un lieu de référence: ce genre de sites accepte des millions de connexions par jour, puisqu'il est un site prévilégié par des utilisateurs de tout profil,
- le site exécute du code: chaque requête d'interrogation émise par un utilisateur est traitée par un processus lancé du côté serveur. Ceci soumet les machines à des charges très grandes, surtout que les connexions simultanées sont fréquentes et la plupart des programmes de recherche sont des scripts interprétés (Perl le plus souvent) donc assez lourds au démarrage.
- le site maintient un volume gigantesque de données: les bases de données des listes et les indexes de tout le Web.

Un site comme AltaVista est un exemple typique des moteurs de recherche. Il s'agit d'un projet lancé par Digital, Inc. et devenu accessible au grand public en 1995. Ce moteur de recherche tourne sur des serveurs Digital AlphaServer de haute performance et est relié à Internet par 10 connexions OC-48 à 2.48 Gbps, le site est aussi relié aux grands réseaux d'accès, tels UUNET et GTE Internetworking, par des connections à 100Mbps. Les AlphaServer sont assez puissants pour exécuter les requêtes en moins de 0.5 seconde sachant que l'index contient plus de 50 millions d'URL.

3.1.3 Considérations Economiques

Internet a pris un tournant "tragique" dès la création du domaine de niveau superieur ".com". Les sites commerciaux constituent une très grande partie du Web, et les entreprises accordent une importance capitale à leur classement dans les résultats des moteurs de recherche (*ranking*). Un client, qui cherche une information sur un produit donné ne fouine pas, le plus souvent, au delà de la deuxième page de résultats.

Les concepteurs des sites Web ont recours au *spamming* pour "forcer" les moteurs de recherche à les classer en premier. En effet, la manière la plus naturelle pour classer des sites indexés est le taux d'occurrence du mot clé dans l'index. Le spamming consiste alors à "fourrer" une page par des mots clés du contenu du site de sorte que le moteur de recherche la liste dans des positions avancées quand la requête porte sur les mots clés en question.

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

3.2 Les Robots du Web

Un robot est un client HTTP qui traverse automatiquement la structure hypertexte du Web en rapatriant un document et, recursivement, tous ceux qu'il référence. Les agents utilisateurs (clients HTTP) ne sont pas des robots, car ils sont commandés par des humains.

Plusieurs appellations synonymes sont utilisées dans la littérature comme *spider*, *worm*, *crawler* et *wonderer* pour désigner les robots du Web. Dans le jargon de l'Intelligence Artificielle, il arrive de parler d'Agents Mobiles sur Internet pour les désigner. Ces appellations sont parfois sources de confusion, car elles donnent l'impression qu'un code exécutable se "déplace" sur le réseau.

Dans l'annexe B on trouve une déscription assez détaillée du système Harvest qui fait office d'un système de gestion d'information complexe et qui peut servir comme un robot. Dans la mise en place du moteur de recherche Hannibal Crawler, j'ai utilisé Harvest. Ce choix est édicté par plusieurs facteurs (voir plus loin) dont la disponibilité du code source.

3.2.1 Usages

Les robots peuvent être utilisés pour plusieurs raisons [MaK95]:

- indexation: création d'indexes de documents divers pour offrir un service de recherche par mots clés. Notons là que les documents indexés peuvent contenir des informations textuelles aussi bien que du son ou de l'image fixe,

- validation de code HTML: analyse syntaxique du code HTML d'un site pour le certifier conforme aux recommentations W3C ou pas. Le problème de code HTML invalide commence à s'accentuer à cause des technologies propriétaires diverses, heureusement les navigateurs ignorent le code non reconnu. En principe, cette tâche incombe aux systèmes auteurs HTML.

- maintenance des hyperliens: à cause de l'architecture fortement distribuée du Web et de l'autonomie des sites, il arrive parfois qu'un lien devienne invalide (*dead link*) à cause d'un changement du nom du document ou même sa destruction. *MOMspider* est un exemple de robot qui effectue cette vérification de validité des liens,

- *mirroring*: les institutions qui maintiennent des grosses archives FTP de documents ou de programmes et qui sont fortement sollicitées ont recourt à dupliquer ces archives dans plusieurs zones géorgaphiques pour réduire le trafic et distribuer la charge sur les serveurs. Pour garder une copie exacte du site d'origine il faut faire des mises à jour périodiques et automatiques par des robots. Le mirroring de sites FTP est une pratique très en vogue, le mirroring de sites HTTP est beaucoup plus complexe. En effet, il est plus difficile de connaître la date de mise à jour d'un document, les références doivent être changées pour pointer sur les copies au lieu des originaux, les références vers des documents non copiés ne doivent pas changer,

- découverte de ressources: c'est sans doute l'application la plus utile des robots. Un site de recherche utilise un robot pour parcourir une large partie du Web (ou le Web en entier s'il a les moyens!) et créer un index accessible par les utilisateurs pour des recherches par mots clés. Les liens invalides sont automatiquement purgés grâce aux mécanismes de maintenance d'index.

3.2.2 Dangers et Problèmes des Robots

Les robots restent un sujet encore contreversé car, bien que très utiles pour l'automatisation de tâches à grande échelle, des robots mal conçus ou mal configurés menacent d'écrouler les performances du réseau. Le protocole HTTP fonctionne en mode transactionnel asymétrique: le client émet des requêtes très courtes et les serveurs renvoient des documents beaucoup plus volumineux ce qui cause des rafales dans le trafic. Entre deux requêtes un humain met un temps pour lire les documents, mais un robot ne laisse pas ce gap soumettant, ainsi, le réseau à de fortes contraintes **continuellement**. Les principaux dangers sont [MaK95]:

- Demande élevée en bande passante: pour accélérer les transferts certains robots ouvrent des connexions parallèles, et comme TCP équilibre la charge par connexion et non pas par protocole, les autres utilisateurs souffriront d'un manque de bande passante. Ceci est parmi les raisons de la remise en cause de la gratuité d'Internet. Les serveurs peuvent être soumis à de fortes charges si le robot lance des requêtes nombreuses en un intervalle de temps court (*rapid fire*), ceci peut dans des cas extrêmes causer le crash du serveur HTTP,

- Coût des mises à jour: la gestion des mises à jour implique un estampillage (*time stamping*) des documents de l'index qui s'avère parfois difficile à gérer. Pour cette raison, certains robots n'implémentent pas corretement ces mécanismes ce qui entraîne une croissance inévitable du trafic inutile,

- Conséquences des mauvaises implémentations: un robot peut visiter un site plusieurs fois s'il ne considère pas l'équivalence sémantique entre plusieurs aspects syntaxiques d'un URL. Par exemple, plusieurs alias DNS peuvent référencer la même machine et si le robot ne fait la distinction, il aura à visiter le même site plusieurs fois. Certains robots font des transferts de documents qu'ils ne savent pas exploiter. Par exemple, il serait maladroit de transférer les images et les appliquettes Java si on est entrain de construire un index pour un service de recherche.

Il faut donc prendre beaucoup de soins lors de la configuration d'un robot pour ne pas trop contraindre les ressources du Web: communications et serveurs. En outre, d'autres complexités font que les développeurs de robots sont confrontés à des problèmes divers:

- Le volume de la matière est énorme: la taille du Web augmente sans cesse et les changements sont très fréquents. Il faut alors penser à inclure les nouvelles resources qui se créent et à mettre à jour les ressources perimées le plus vite possible, il s'agit du problème de réduction du temps d'absolescence des documents qui sera présenté plus loin.

- Savoir quoi inclure/exclure: un robot est généralement incapable de décider de l'utilité d'un document. L'intêret d'un document depend de l'audience qui n'a pas, le plus souvent, un profil clairement définit ce qui amène le robot à inclure pratiquement tout document rencontré. Pour spécifier (explicitement) aux robots les parties d'un site à indexer, un standard Internet pour l'exclusion des robots, qui est encore à l'étape de draft, a été définit. Ce standard permet aux administrateurs des sites Web (*webmasters*) de spécifier les zones de l'arborescence qui ne se prêtent pas à une indexation (voir annexe C.2),

- Indexation des documents: la création d'un résumé à partir d'un texte est encore un sujet de recherche, les robots actuels utilisent des algorithmes très simples pour extraire des mots clés d'un document. Par exemple, l'indexation d'un document peut consister à en extraire le premier paragraphe et la première ligne de chaque paragraphe qui suit.

- Classification des documents: la recherche thématique est une facilité très recherchée par les utilisateurs des systèmes de recherche, et c'est l'un des atouts majeurs des répertoires sur les moteurs de recherche. Les balises META de HTML permettent d'inclure des informations supplémentaires et peuvent être utilisées pour communiquer la classe du document. Mais, les systèmes de classification classiques utilisés dans les librairies ne sont pas unifiés, il sera donc très difficile d'imposer un standard de classification sur le Web.

Pour exploiter un robot tout en ayant un comportement "idéal" et *network-friendly*, un administrateur doit:

- n'utiliser un robot que s'il en a vraiment besoin,

- garantir que les serveurs puissent reconnaître le robot et que son administrateur puisse être contacté,

- éviter les rafales et le transfert de documents inutiles,

- se conformer au protocole d'exclusion de robots,

- surveiller l'activité du robot en consultant les historiques (*log files*),

3.3 Le Moteur de Recherche Hannibal Crawler

Hannibal est un moteur de recherche basé sur Harvest. Le projet Hannibal est une initiative de MRS, il a pour objectif d'offrir un service de recherche dans tout le Monde Arabe.

Le déploiement de Hannibal dans le Monde Arabe dépendra beaucoup de la coopération entre les fournisseurs de services Internet (*Internet Service Providers, ISP*) dans le Monde Arabe et de la publicité dont il bénéficiera.

	Moteurs de recherche	Répertoires
Qualité de l'index	Moyenne/mauvaise	Bonne
Sollicitation de l'utilis.	Faible	Forte
Absolescence	Peu probable	Élevéee

Tableau 3.1: Les moteurs de recherche et les répertoires comparés

3.3.1 Pouquoi un Robot?

La valeur d'un moteur de recherche réside surtout dans son taux de couverture, c'est à dire le nombre de sites qu'il indexe. Un répertoire est beaucoup plus facile à mettre en place qu'un moteur de recherche, en effet beaucoup de scripts Perl qui fournissent un service de répertoire sont distribués gratuitement sur Internet.

Le problème majeur d'un répertoire est qu'il sollicite beaucoup l'utilisateur, en effet pour ajouter son URL à un répertoire il faut remplir un long formulaire et faire une indexation manuelle du site. Le tableau 3.1 montre les principales différences entre les moteurs de recherche et les répertoires.

L'utilisation d'un robot facilitera la pénétration du moteur de recherche, ce qui revient à lui garantir un succès.

3.3.2 Fonctions de Hannibal

Hannibal offre un service de recherche simple et un service de recherche avancée. La recopie d'ecran reproduite sur la figure 3.1 montre l'aspect de Hannibal Crawler.

Le service de recherche avancée permet:

- activer/désactiver l'affichage des lignes où il y a une occurrence du mot recherché,
- activer/désactiver l'affichage des informations sur l'objet stocké dans l'index,
- activer/désactiver l'affichage des liens vers les informations d'indexation collectées par les gatherers,
- spécifier le nombre maximal des objets retournés,
- spécifier le nombre maximal des lignes d'occurrence par objet,
- spécifier le nombre maximal des résultats (objets+lignes d'occurrences),

La déclaration de l'URL se fait d'une manière très simple et ne nécessite que l'URL sans informations additionnelles, le script qui gère la liste des URLs s'assure que l'URL n'est pas déjà déclaré. La mise à jour de l'index se fait périodiquement.

Le déploiement de Hannibal se fera par étapes et se concentrera sur les fournisseurs d'accès Internet car ils offrent, exclusivement, les services d'hébergement. Tout d'abord, il sera installé sur une machine chez Planet Tunisie qui est un partenaire de MRS.

La distribution du processus de collecte (*gathering*) économisera beaucoup de trafic sur le réseau. Il serait donc très intéressant d'installer des gatherers chez ATI et 3S Global Net en premier lieu, puis chercher à coopérer avec d'autres fournisseurs de services dans le Monde Arabe.

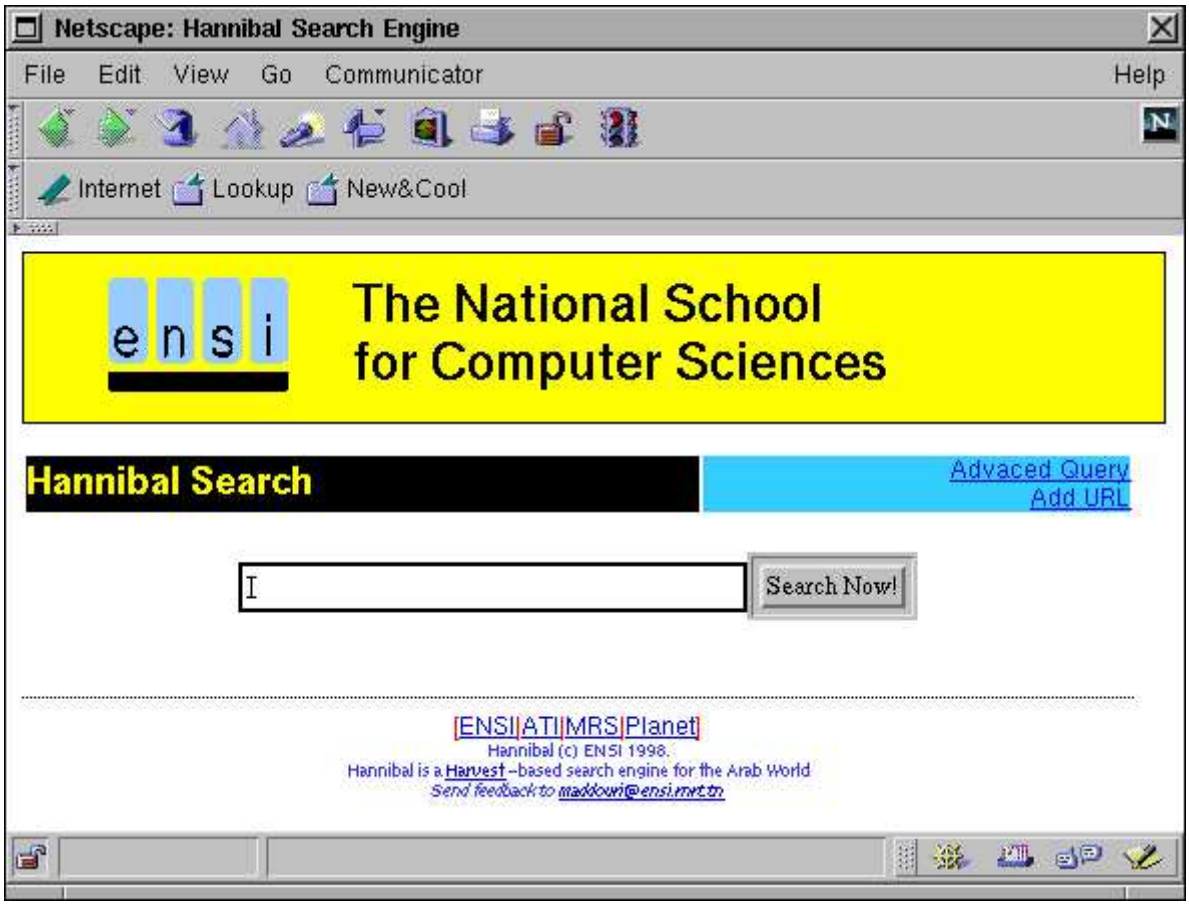

Figure 3.1: Recopie d'écran de la page d'accueil de Hannibal Crawler.

3.3.3 Langage de Requêtes

L'indexeur utilisé dans Hannibal Crawler est Glimpse, il offre plusieurs possibilités avancées:

- requêtes avec/sans respect de la casse (*case-sensitive* ou *case-insensitive*),
- recherche de parties de mot (*compute* retourne les correspondances pour *computer*, *computing*, *computation*, etc.) ou de plusieurs mots ("*informatique industrielle*"),
- combinaisons de mots avec des opérateurs booléens,
- tolérance d'erreurs (*acces* retourne les correspondances pour *accès*),
- affichage des lignes contenant l'expression cherchée,
- spécification du nombre maximum des correspondances retournées,
- une forme simplifiée d'expressions régulières.

Des exemples sont donnés pour illustrer ces possibilités:

Un seul mot clé: `informatique`

Retourne tous les objets contenant le mot clé *informatique.*

Combinaison booléenne: `informatique AND industrielle`

Retourne tous les objets contenant à la fois le mot *informatique* et le mot *industrielle.*

Phrase: `''informatique industrielle''`

Retourne tous les objets qui contiennent les mots consécutifs *informatique industrielle*.

Requêtes structurées: `Title: protocoles`

Retourne tous les objets dont l'attribut *Title* contient le mot *protocoles*.

Expressions régulières:

Glimpse supporte seulement quelques expressions régulières, la recherche par expressions régulières est plus coûteuse en temps.
`^école` *école* en début de ligne,
`école$` *école* en fin de ligne,
`[a-ho-z]` caractères de *a* à *h* et de *o* à *z*,
`.` tout caractère sauf retour à la ligne,
`c*` une ou plusieurs occurrences du caractère "c" (*Kleene star*),
`\*` le caractère "*", le \ (*backslash*) déspécialise les caractères.

Remerciments

Je tient à exprimer ma gratitude et mes sentiments de respect profond envers DR. MOHAMED SAÏD OUERGHI. En effet, son experience et son savoir faire m'ont fait économiser énormement d'efforts et en dehors des obligations professionnelles il est une personne de qualités rarissimes.

Conclusion

Ce travail a traité deux problèmes, à savoir: le support de la langue Arabe dans les navigateurs du Web et la mise en place d'un service de recherche sur des serveurs bilingues (latin ou arabe). Nous avons donc développé un plug-in Hannibal Viewer pour résoudre le premier problème et nous avons transformer le système Harvest pour décrire le moteur de recherche Hannibal Crawler.

Bien que le plug-in développé n'apporte pas la solution adéquate au problème de la langue Arabe sur le Web, il peut être considéré comme une contribution utile. La solution ultime pour ce problème est sans doute Unicode, les nouveaux navigateurs commencent à offrir un support complet pour Unicode et l'encodage UTF-8.

La version actuelle du plug-in pour X Window doit se compiler sans problèmes sur la plupart des plate-formes UNIX modernes. Son portage sous Windows doit se faire sans grande difficulté car le noyau de ce système est compatible POSIX et parce qu'il existe par ailleurs des émulateurs X Window. En effet, les routines graphiques utilisées sont de bas niveau (uniquement Xlib) et peuvent être facilement mappées vers la plupart des APIs des interfaces graphiques modernes.

Les extensions du plug-in qui pourraient être apportées couvriront l'aspect présentation et encodage. Concernant l'aspect présentation, il serait, par exemple, intéressant d'ajouter un analyseur syntaxique réduit de HTML qui ne reconnaît qu'un sous-ensemble de balises qui améliorent l'aspect érgonomique des documents et permettent de gérer les hyperliens. Pour l'ajout d'un autre encodage, il suffirait d'ajouter une police par encodage ou de mapper les nouveaux encodages vers un seul.

Le moteur Hannibal Crawler permet d'effectuer de simples recherches par mots clés mais également selon des expressions régulières ou booléennes (une certaine syntaxe est prévue dans ce cadre). Dans son mode *Advanced Query*, il est possible d'intégrer une certaine recherche "sensitive" acceptant des erreurs dans les mots clés. Les résultats de la recherche peuvent donner lieu à du texte en Arabe. Ceci est possible dans la mesure où les mots clés pointent sur des documents dont le contenu est en Arabe. Que faut-il de plus pour construire un vrai moteur de recherche intégrant complètement la langue Arabe?

La réponse à cette question dépend énormément de deux choses: la première est la capacité du système Harvest d'indexer des documents d'encodage quelconque, la deuxième est le temps nécessaire pour modifier le noyau de ce système (dans sa version actuelle l'archive est de quelque 1.5 Mb à décortiquer après décompression!).

Remarquons pour conclure, que ce travail n'est sans doute pas inutile, car il nous a permis de mettre en exercice les problèmes relatifs à l'internationalisation des applications (et notamment du Web). Cependant, il existe pour des solutions d'accès Web en Arabe livrables tout de suite à savoir le navigateur Sindbad de Sakhr et le moteur Ibhath de Ayna[4].

[4] http: //www.ayna.com

Glossaire

CGI *Common Gateway Interface*. Un standard pour interfacer un serveur Web avec des applications tournant sur le site serveur. Ceci permettra aux clients HTTP de communiquer avec des applications de tout genre.

HTTP *HyperText Transfer Protocol*. Le protocole de transfert de documents sur le Web, c'est un protocole très simple et très leger pour un mode transactinnel.

HTML *HyperText Markup Language*. Le langage de description de document le plus utilisé sur le Web, la quasi totalité des documents transmis par HTTP sont des documents HTML.

MIME *Multipurpose Internet Mail Extensions*. Des spécifications pour étendre les possibilités de la messagerie sur Internet aux transmissions de documents multimedias.

IETF *Internet Engineering Task Force*. Un des organismes chargés d'éditer les documents de standardisation sur Internet.

Unicode Un rassemblement de constructeurs du monde informatique qui travaille sur l'établissement d'un standard pour l'internationalisation des applications.

Plug-in Un module logiciel qui s'ajoute à une application pour étendre ses possibilités. Par opposition aux modules logiciels modernes (tels Java Beans), un plug-in dépond de la plateforme et de l'application.

Appliquette Une application Java qui tourne dans la machine virtuelle Java implémentée par le navigateur avec certaines restrictions. Les appliquettes sont un Byte Code Java qui est transmit depuis le serveur HTTP vers le client.

Glyphe L'aspect typographique visible d'un caractère.

API *Application Programming Interface*. Un ensemble de fonctions standards qui servent de services pour les applications. Ainsi, l'interface des appels systèmes de UNIX est l'API du système d'exploitation.

Helper Une application externe qui rend des services à une application donnée en augmentant ces possibilités.

Robot Dans le jargon des systèmes d'information, un client qui accède automatiquement aux serveurs pour effectuer une tâche donnée.

Annexe A

Configurer Hannibal

A.1 Compiler et Installer le Plug-in

Si une ressource de type text/plain; charset=ISO-8859(-i) est intégrée dans une page Web avec la balise EMBED, l'installation du plug-in peut être déclenchée automatiquement. En effet, l'un des attributs de cette balise est PLUGINSPAGE qui est un URL que le navigateur ouvrira automatiquement si l'utilisateur accède à une ressource sans qu'il ait le plug-in adéquat.

Le plug-in Hannibal Viewer est écrit pour X Window sur les plate-formes UNIX. Le plug-in est distribué sous forme de source avec un Makefile et se compile sans problèmes sur HP hp-ux, SGI Irix, Sun Solaris/SunOs, Linux, Digital OSF/1. Un portage vers Windows est éminent. Pour compiler le plug-in, il faut décompresser l'archive:

```
gzip -d np_hannibal_viewer-src-1.0.tar.gz
tar xf np_hannibal_viewer-src-1.0.tar
```

Ceci crée un répertoire npav. Pour compiler il faut se placer dans npav/unix et lancer:

```
make -f [fichier make]
```

Le paramètre envoyé à `make` est le fichier Makefile spécifique à l'architecture cible. L'architecture utilisée peut être connue avec la commande `uname -a`. Cette compilation génère un binaire `npav.so` qu'il faudra copier dans `~/.netscape/plugins/`, si le navigateur utilisé est Netscape Navigator 3, il faut le redémarrer pour qu'il relise la liste de ses plug-ins. Notons là que si Navigator 3 n'arrive pas à résoudre certains symbôls dans le code du plug-in, il ne l'acceptera pas et ne sera donc pas enregistré sans aucune notification de l'utilisateur. Navigator 4 notifie l'utilisateur si des symbôls sont non résolus.

Il faudra aussi installer la police de caractères utilisant l'encodage ISO-8859-6. Si l'utilisateur a les droits d'accès, il peut la copier dans un repertoire de recherche de polices du serveur X (/usr/X11/lib/fonts/misc) et lancer `mkfontdir` . Un utilisateur ordinaire peut quant à lui utiliser `xset`:

```
xset +fp /home/foulen/npav/fonts/
```

A.2 Compiler et Installer Harvest

Harvest est distribué au format source ou binaire pour quelques plateformes, à l'heure de l'écriture de ces lignes l'IRTF-RD distribue la version 1.5. Pour installer Harvest, il suffit de lancer ces commandes:

```
% configure
% make all
% make install
```

Par défaut, Harvest est installé dans `/usr/local/harvest`. Pour choisir un autre lieu, il suffit de changer le paramètre `prefix` dans l'entête des fichiers Makefile.

Le script `RunHarvest` offre une interface interactive pour lancer le gatherer et le broker et faire les paramètrisations souhaitées.

A.3 Configurer le Serveur Web

Pour exploiter l'interface Web dans les requêtes de recherche et d'administration, il faut configurer le serveur Web pour exécuter les scripts CGI et accéder à quelques documents sous le répertoire d'installation de Harvest.

A.3.1 Serveurs du type NCSA

Il s'agit des serveurs NCSA httpd et Apache. Il faut ajouter une entrée de type Alias à `conf/srm.conf` comme suit:

```
ScriptAlias /Harvest/cgi-bin/ usr/local/harvest/cgi-bin/
Alias /Harvest/ /usr/local/harvest/
```

Ceci suppose que Harvest est installé sous `/usr/local/harvest`. La ligne `ScriptAlias` sert à autoriser l'exécution de scripts sous le répertoire spécifié, elle peut être remplacée par:

```
AddType application/x-httpd-cgi .cgi
```

qui autorise l'exécution de scripts n'importe où dans le système de fichiers moyennant qu'ils portent l'extension `.cgi`. Pour forcer le seveur à relire son fichier de configuration, il faut lui envoyer le signal SIGHUP (1).

A.3.2 Serveur CERN httpd

Il faut ajouter une entrée Exec et Pass à httpd.conf comme suit:

```
Exec /Harvest/cgi-bin/* /usr/local/harvest/cgi-bin/*
Pass /Harvest/* /usr/local/harvest/*
```

Pour forcer le seveur à relire son fichier de configuration, il faut lui envoyer le signal SIGHUP (1).

A.3.3 Serveur Netscape

Depuis l'interface d'administration, choisir "URL mapping | Map a URL to a local directory" et mapper /Harvest à /usr/local/harvest/.

Sous "CGI and Server Parsed HTML | Activate CGI as a file type...", choisir dans "Browse Files" la racine d'installation de Harvest et sélectionner "I'd like to active CGI as a file type".

Annexe B

Le Système Harvest

Harvest est un système conçu par le groupe de *Resource Discovery* à l'*Internet Research Task Force (IRTF-RD)* dont la plupart des membres sont de l'Université de Colorado. C'est un ensemble d'outils pour rassembler, extraire, chercher, cacher et dupliquer de l'information pertinante à travers Internet d'une manière distribuée [BOW95].

Le but de Harvest est d'offrir un outil pour créer des indexes en faisant un bon usage des serveurs sur Internet, des liens de communication et de l'espace disque. Des mesures effectuées par les auteurs de ce système ont montré des performances surprenantes par rapport à d'autres outils du même genre.

B.1 Description du Système

La figure B.1 présente l'architecture simplifiée de Harvest. Le *gatherer* collecte les informations utiles à l'indexation (mots clés, auteur, titre, etc) depuis les ressources disponibles chez les fournisseurs (serveurs FTP et HTTP). Le *broker* collecte les informations d'indexation depuis un ou plusieurs gatherers, supprime les duplicatas, indexe l'information et fournit une interface d'interrogation par HTTP. Le *replicateur* duplique les brokers dans plusieurs serveurs d'Internet pour distribuer la charge et le trafic réseau. Le *cache* sert, comme son nom l'indique, à maintenir un cache des objets transférés depuis les serveurs d'Internet pour réduire le trafic réseau.

B.2 Le Gatherer

Le gatherer collecte les ressources en utilisant plusieurs méthodes d'accès: HTTP, FTP, Gopher, NNTP et fichiers locaux [HSW96]. Selon le type du document, le gatherer extrait des informations d'indexation structurées et les met à la disposition des brokers pour la constitution d'indexes. Par exemple, si le gatherer accède à un document technique au format LaTeX, il en extrait le titre, le nom de l'auteur et le résumé dans une structure. Cette structure est une liste de paires attribut-valeur conforme au *Summary Object Interchange Format (SOIF).* Le *daemon gatherd* sert une base de donnée locale des objets collectés, ce serveur peut être interrogé à la main (depuis la ligne de commande) par un client *gather* ou par un broker. Dans la suite on utilisera le terme objet pour référencer une entité structurée stockable et transférable et qui contient un extrait d'une ressource sachant qu'une ressource est tout document textuel ou non textuel.

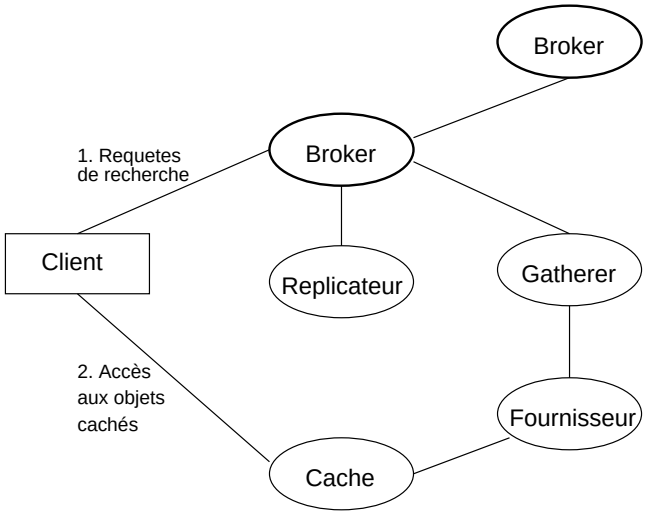

Figure B.1: Architecture simplifiée de Harvest.

B.2.1 Le Fichier de Configuration

Lors de son lancement, le gatherer lit un fichier de configuration qui énumère les sites des fournisseurs ainsi que quelques données diverses. Le fichier de configuration du gatherer doit ressembler à ceci:

```
#
# exemple.cf - Fichier de confi guration d'un gatherer type
#
Gatherer-Name:  Gatherer de test typique
Top-Directory:  /usr/local/harvest/gatherers/typique

# URLs des fournisseurs
<RootNodes>
http: //www.ensi.rnrt.tn
http: //www.enit.rnrt.tn
</RootNodes>

<LeafNodes>
http: //www.netcom.com.tn/docs/ATM/switches.html
http: //www.tmi.com.tn/os/solaris/sparc/latest.html
</LeafNodes>
```

Les URLs listés sont soit des nœuds feuilles (LeafNodes) soit des nœuds racines (RootNodes). Les nœuds feuilles spécifient des documents qui seront transférés et traités par le gatherer. Les nœuds racines sont des racines d'arborescences que le gatherer doit énumérer récursivement pour avoir aucun, un ou plusieurs nœuds feuilles. Dans le contexte de FTP et Gopher, l'énumération se fait en parcourant récursivement le contenu des répertoires. Pour HTTP, l'énumération se fait en suivant récursivement les liens dans les documents. Dans les news, l'énumération retourne tous les messages dans un groupe de discussions.

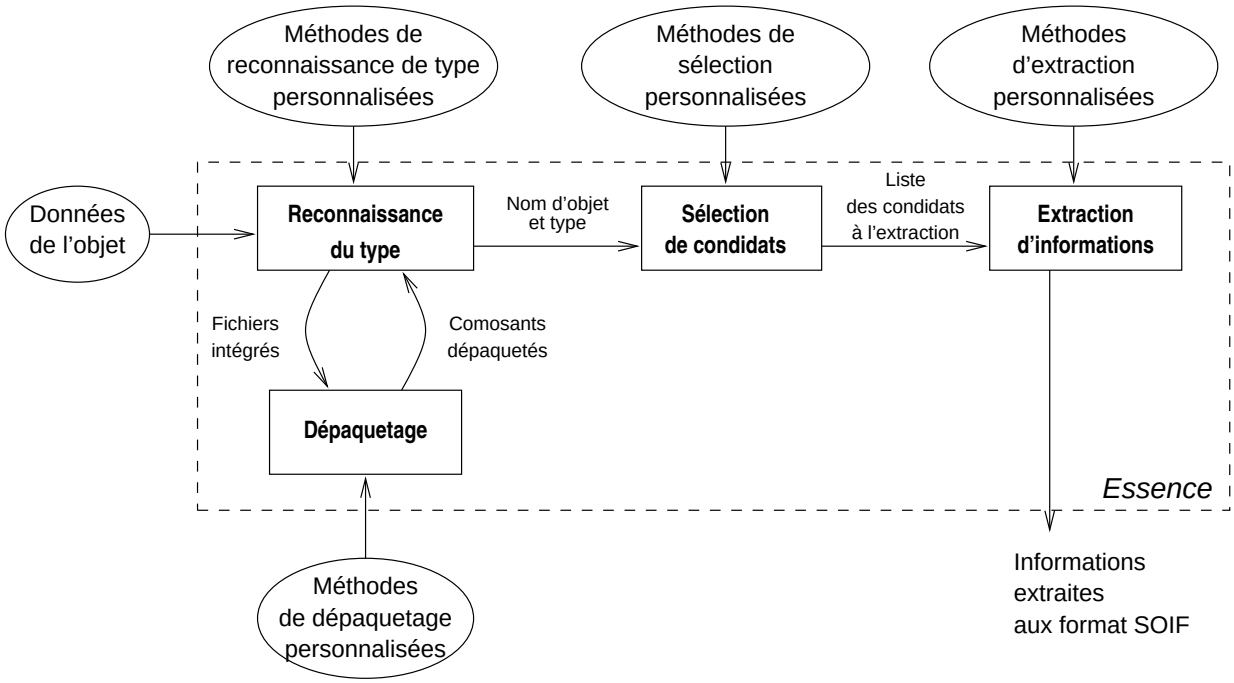

Figure B.2: Architecture de Essence.

Même que Harvest n'est pas conçu pour être un robot, il peut être utilisé pour remplir les mêmes fonctions. Pour éviter les problèmes discutés plus haut, un nœud racine n'est pas étendu à plus que 250 nœuds feuilles et le gatherer ne suit les liens que s'ils sont à l'interieur du site spécifié par le nœud racine. Bien sûr ces paramètres sont ajustables dans le fichier de configuration.

B.2.2 Extraction de Données avec Essence

Essence est un composant du gatherer qui fait la plupart des traitements d'extraction des informations d'indexation, c'est le noyau du gatherer. Le schéma de fonctionnement de Essence est détaillé dans la figure B.2 [HSC95]. En effet, Essence:

- peut reconnaître le type du document transféré en exploitation les reseignements données explicitement par le protocole (comme le champs Content-type dans HTTP) ou en appliquant certaines heuristiques comme celles utilisées par la commande `file`,
- extrait (*unnest*) les documents intégrés dans des archives ou compressés par uuencode, tar, zip et gzip,
- choisit les données à indexer, par exemple il rejette les documents audio parce qu'ils ne seront pas indexés,
- extrait les données d'indexation depuis les documents collectés (*summarizing*).

Un *summarizer* est un script spécifique à un type de document qui sait en extraire les informations utiles à l'indexation. Le tableau B.1 [HSC95] liste les summarizers livrés dans la distribution de Harvest et donne une idée sur ce que c'est "information utile".

Type	Données extraites
Audio	Nom du fichier
Bibliographique	Auteur et titres
Binaire	Champs de texte significatifs
C, Entête C (.h)	Noms des fonctions et commentaires
DVI	Traite le texte sans formatage
FAQ, ReadMe	Tous les mots
FrameMaker	Converti en SGML
Police	Commentaires
HTML	Liens hypertextes et quelques champs particuliers
LaTeX	Auteur, titre, résumé, etc.
Courrier	Quelques champs d'entête
Makefile	Commentaires et noms des cibles
ManPage	Synopsis, auteur, titre, etc
News	Quelques champs d'entête
Code objet	Table des symbôls
Patch	Noms des fichiers à patcher
Script Perl	Noms des procédures et commentaires
PostScript	Les texte du document sans formatage
RCS, SCCS	Renseignements du contrôle de version
RTF	Converti en SGML
SGML	Champs nommés dans les tables d'extraction
Script Shell	Commentaires
Distribution de source	Fichier README et commentaires des Makefiles
Lien symbolique	Nom du fichier, propriétaire et date de création
TeX	Le texte du document sans formatage
Texte	Les 100 premières lignes et la première phrase des para. suiv.
TROFF	Auteur, titre, etc.
non reconnu	Nom du fichier, propriétaire et date de création

Tableau B.1: Type de documents reconnus et traités par Essence.

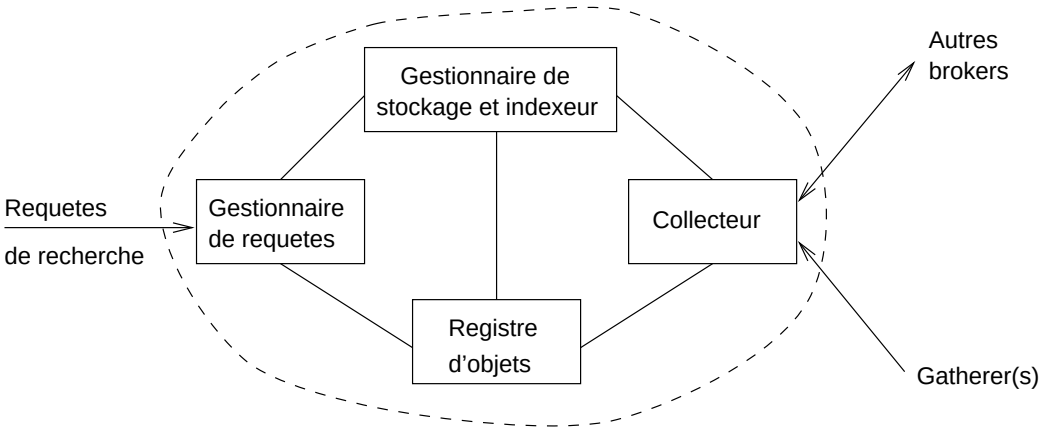

Figure B.3: Composantes du broker.

B.3 Le Broker

La fonction principale du broker est le maintient de l'index. La figure B.3 montre les détails des composantes du broker. Les objets dans la base du broker sont au format SOIF, ils sont rassemblés depuis les gatherers et utilisés pour construire l'index.

B.3.1 Le gestionnaire de stockage

Le gestionnaire de stockage (*storage manager*) maintient la base de données des objets dans le broker. Cette composante effectue 3 opérations: ajouter, supprimer et renvoyer. Pour accélérer les accès aux objets, le gestionnaire de stackage utilise un cache, et rassemble les objets selon les deux chiffres de poids le plus faible du BUID comme explicité dans la figure B.4. Un nom d'objet est formé de la chaîne "OBJ" concaténée au BUID.

B.3.2 Le Registre

Pour identifier les objets et éviter les duplications, le broker utilise un identificateur par objet: l'OID. Cet identificateur est construit d'une manière unique à partir de l'URL de l'objet lui même et le nom, le hôte et la version du gatherer. L'OID est unique sur Internet mais occupe trop d'espace pour servir d'identificateur dans une base de données, le broker utilise pour son fonctionnement interne un *Broker Unique Identifier (BUID)* qui est plus réduit en taille mais ne sert qu'à l'interieur du broker.

Le registre (*registry*) est une composante du broker qui maintient la liste des OID et BUID et gère les accès aux renseignements sur les objets avec un mécanisme de caching. Les objets, dont la durée de validité a expiré, sont supprimés du registre et du gestionnaire de stockage.

B.3.3 L'indexeur

Harvest spécifie une interface vers l'indexeur mais ne spécifie pas un indexeur particulier. En effet, dans la conception d'un indexeur il y a un énorme compromis entre espace disque et rapidité d'accès, l'approche de Harvest permet de personnaliser les

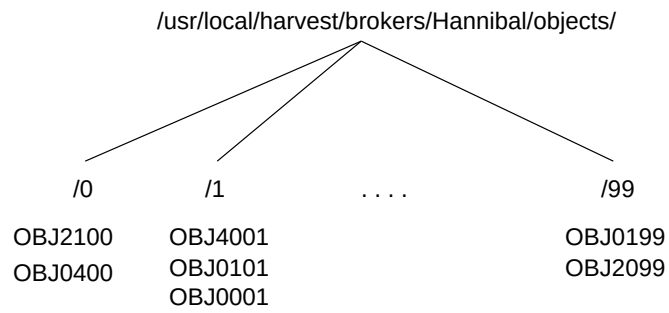

Figure B.4: Organisation des objets du gestionnaire de stockage.

indexeurs selon les besoins. La distribution de Harvest est livrée avec les indexeurs Glimpse et Nebula.

B.3.4 Le collecteur

Le collecteur (*collector*) rassemble les objets depuis des gatherers ou d'autres brokers et se charge des mises à jour des objets dans le broker. Le collecteur peut demander la compression des objets tranférés pour réduire la bande passante consommée, il est aussi possible de demander le transfert des objets plus récents qu'une date donnée. L'ordonnancement des collections et les sites de collection peuvent être configurés par l'administrateur.

B.3.5 Le gestionnaire de requêtes

Le gestionnaire de requêtes (*query manager*) s'occupe des requêtes de recherche émises par l'utilisateur et des requêtes d'administration et de collecte émises par l'administrateur ou les autres brokers, mais, sa fonction principale est le traitement des requêtes de recherche.

Comme le broker peut utiliser plusieurs indexeurs, il est nécessaire de convertir les requêtes vers une syntaxe indépendante de l'indexeur. Une fois la requête traitée, le gestionnaire de requêtes renvoi une description des objets correspondants.

Annexe C

Recommendations pour les Concepteurs de Sites Web

Pour que le projet Hannibal puisse rendre des services de bonne qualité, il a besoin de la coopération des concepteurs des sites Web. Même que Harvest est conçu pour fonctionner sans la coopération des utilisateurs, il est toujours possible d'améliorer la qualité des résultats de recherche en diffusant des documents qui se prêtent à une bonne indexation. Dans le monde des documentistes on dit qu'un document mal indexé est un document perdu, l'éditeur d'un document a donc interêt à ce que son document soit bien indexé. Les sections suivantes présentes quelques astuces pour améliorer la qualité de l'indexation et donc celle des résultats de recherche.

C.1 Déclaration de l'URL

Bien sûr, il faut tout d'abord déclarer son site chez Hannibal. Le projet Hannibal prévoit un formulaire HTML pour déclarer les URLs des sites. Ce formulaire est accessible en cliquant l'icône "Add URL" dans la page d'accueil de Hannibal à http: //www.hannibal.tn.

Il faut garder en tête que le gatherer de Hannibal ne suit que les liens à l'interieur du site, c'est à dire que si les pages du site Web se trouvent sur plus qu'une machine il faudra déclarer les deux parties. Imaginons une société qui a un site Web à www.societe.tn dont certaines pages renvoient à des divisions de la société sur d'autres machines comme www.societe-medical.jo et www.societe-mil.eg, il faut dans un cas pareil déclarer les trois URLs.

Les documents non référencés par des liens ne seront jamais accédés même s'ils sont disponibles sur le même serveur. L'exemple typique est celui des pages personnelles, la page www.societe.tn/~mouldi ne sera jamais accédée si elle ne fait pas partie des nœuds de l'arborescence hypertexte de racine www.societe.tn/index.html.

Le temps entre la déclaration du site et l'inclusion dans l'index ne doit pas dépasser quelques jours. En effet, ceci dépend complètement de l'administrateur du moteur de recherche qui a la liberté de décider des heures des collectes.

C.2 Le Protocole d'Exclusion des Robots

Le travail sur ce protocole [MaK97] est en cours. Il a été initié par Martijn Koster de Nexor Inc., une extension aux agents mobiles a été étudiée par F. GUIDICI et A. SAPIA dans [**?**]. Ce protocole espère éviter des effets indisérables causés par les robots comme l'indexation d'un site non encore annoncé et donc qui n'est pas censé être accédé pas les utilisateurs, le parcours de parties du site Web qui demandent trop de ressources au serveur (des scripts très lourds, des documents fortement liés, etc), le parcours récursif d'un espace énorme d'URLs, etc. Ce protocole peut servir aux administrateurs pour interdire explicitement l'accès à des parties du site Web par robot.

Un robot conforme à ce standard doit procéder à la lecture du fichier /robots.txt, ce fichier doit être accessible par HTTP et de type de média text/plain. Si ce fichier n'est pas diponible sur le site Web, le robot doit considérer qu'aucune restriction n'est applicable.

Le fichier robots.txt contient des blocs qui, pour un ensemble d'agents utilisateurs, spécifient les parties autorisées et les parties interdites. Dans l'exemple qui suit, il est interdit aux robots des moteurs de recherche les plus connus (*scooter* est celui d'AltaVista) de parcourir les documents sous /tmp sauf /tmp/ready.html. Le champs User-agent peut accepter un joker "*" pour désigner tous les robots.

```
User-agent:  webcrawler
User-agent:  infoseek
User-agent:  scooter
Allow:  /tmp/ready.html
Disallow:  /tmp
Disallow:  /user/foolene
```

Si des mécanismes de caching sont utilisés, la durée de validité par défaut est de 7 jours.
[MaK97] donne la grammaire complète du fichier robots.txt et renvoie aux définitions de `token` et `path` dans RFC 1945 et RFC 1808 respectivement.

Il faut veiller à ne pas exposer trop de ressources dans robots.txt, en effet n'importe qui peut lire ce fichier et peut avoir une idée sur les documents et les applications dans le site. Des informations pareilles sont parfois très utiles aux pirates! .

C.3 Les META Tags de HTML

Les balises META de HTML sont des balises de la section HEAD qui servent à inclure dans le document des renseignements supplémentaires que d'autres balises ne peuvent pas exprimer. Le protocole d'exclusion de robots basé sur le fichier robots.txt nécessite l'autorisation de l'administrateur du site, une autre approche qui n'est pas bien reconnue par les robots actuels consiste à utiliser la balise META. Dans la balise META robots, l'utilisateur ordinaire peut indiquer si la page peut être indexée ou pas et si les liens dans la page doivent être suivis ou pas. Par exemple,

<META NAME="robots" CONTENT="noindex, follow">

indique qu'il ne faut pas indexer cette page mais qu'il faudra passer, récursivement, aux pages qu'elle référence. Le contenu de cette META est: `index`, `noindex`, `follow`, `nofollow`, `all=index+follow` ou `any=noindex+nofollow`.

La META balise CHARSET est aussi d'une grande utilité pour connaître le jeu de caractères utilisé et parsuite permettre à l'agent utilisateur de la présenter correctement. Il faut donc essayer d'inclure cette META balise à toute page éditée.

Les informations d'indexation extraites par les gatherers de Harvest comprennent le plus souvent l'auteur et le titre du document. Alors, il faut veiller à choisir des titres qui reflètent le contenu du document.

Annexe D

Netscape Plug-ins API

Lors de l'initialisation du plug-in, Navigator fournit une structure de pointeurs sur les méthodes implémentées de son côté [NCI].

```
typedef struct _NPNetscapeFuncs {
   uint16 size;
   uint16 version;
   NPN_GetURLUPP geturl;
   NPN_PostURLUPP posturl;
   NPN_RequestReadUPP requestread;
   NPN_NewStreamUPP newstream;
   NPN_WriteUPP write;
   NPN_DestroyStreamUPP destroystream;
   NPN_StatusUPP status;
   NPN_UserAgentUPP uagent;
   NPN_MemAllocUPP memalloc;
   NPN_MemFreeUPP memfree;
   NPN_MemFlushUPP memflush;
   NPN_ReloadPluginsUPP reloadplugins;
   NPN_GetJavaEnvUPP getJavaEnv;
   NPN_GetJavaPeerUPP getJavaPeer;
   NPN_GetURLNotifyUPP geturlnotify;
   NPN_PostURLNotifyUPP posturlnotify;
#ifdef XP_UNIX
   NPN_GetValueUPP getvalue;
#endif /* XP_UNIX */
} NPNetscapeFuncs;
```

Ainsi, un appel à `NPN_MemAlloc()` est en fait un appel à `NPNetscapeFuncs.memalloc()`. Du côté de Navigator, l'API définit les méthodes suivantes [NCI].

D.1 API: Services de Navigator

`NPN_DestroyStream` Ferme et détruit le flux donnée dans l'instance donnée. Le flux en question aurait pu être créé par Navigator (appel à `NPP_NewStream()`)

ou par une demande explicite du plug-in (`NPN_NewStream()`). La cause de la déstruction du flux est donnée dans le paramètre `reason`.

`NPN_GetJavaEnv` Retourne l'environnement d'exécution de Java, nécessaire pour appeler des méthodes Java.

`NPN_GetJavaPeer` Retourne l'objet Java associé avec l'instance du plug-in.

`NPN_GetURL` Demande à Navigator d'ouvrir un nouveau flux depuis l'URL spécifié et retourne immédiatement. Le flux peut être envoyé au plug-in ou bien à Navigator pour être traité sans l'intervention du plug-in.

`NPN_GetURLNotify` Tout comme `NPN_GetURL()` mais avec notification. Après l'appel, Navigator informe le plug-in sur le succès ou l'echec de la tentative d'ouverture de l'URL.

`NPN_MemAlloc` Alloue de la mémoire dans l'espace mémoire de Navigator, utiliser cette fonction à l'avantage de bénéficier des possibilités de *chaching* intégrées dans Navigator.

`NPN_MemFlush` Fonction de libération de la mémoire du *cache* disponible uniquement sur Macintosh. En effet, sur Macintosh la gestion du *cache* fait qu'il faut explicitement vider le cache pour pouvoir allouer de la mémoire avec `NPN_MemAlloc()`.

`NPN_MemFree` Libère un bloc mémoire alloué par `NPN_MemAlloc()`.

`NPN_NewStream` Crée un flux généré par le plug-in et consommé par le Navigator.

`NPN_PostURL` Envoie des données contenues dans un fichier ou un buffer en mémoire vers un serveur distant. Cette méthode fonctionne avec pratiquement tous les protocoles: news, HTTP, SMTP et FTP.

`NPN_PostURLNotify` Tout comme `NPN_PostURL()` mais avec notification du résultat.

`NPN_RequestRead` Demande à Navigator la lecture de plages précises d'octets à l'interieur d'un flux.

`NPN_Status` Affiche un méssage dans la barre d'état de Navigator. Cette méthode peut s'avérer utile pour reproduire le comportement de Navigator qui affiche les URLs des liens dans cette zone.

`NPN_UserAgent` Retourne la chaîne que Navigator utilise pour s'identifier chez les serveurs HTTP. Il s'agit exactement de la valeur du champ "User Agent" de l'entête HTTP.

`NPN_Version` Retourne la version de l'API des plug-ins supportée par Navigator. Les valeurs `plugin_major` et `plugin_minor` spécifient la version et la sous-version de l'API.

`NPN_Write` Ecrit des données vers un flux déstiné à la consommation par Navigator.

Du côté de Navigator, l'API définit les fonctions suivantes:

D.2 API: Services du Plug-in

`NPP_Destroy` Détruit une instance d'un plug-in, cette méthode est appelée lorsque l'utilisateur quitte la page qui contient l'instance ou lorsqu'il ferme la fenêtre.

`NPP_DestroyStream` Indique la fermeture et la déstruction d'un flux. Le paramètre `reason` donne la raison de la fermeture, le plus souvent il s'agit d'une fin normale (`NPRES_DONE`) ou d'un arrêt sur demande de l'utilisateur (`NPRES_USER_BREAK`).

`NPP_GetJavaClass` Retourne la classe Java associée au plug-in.

`NPP_HandleEvent` Gestionnaire d'événements spécifique au Macintosh. Cette méthode n'est appelée que sur les plateformes Macintosh, les événements concernant la gestion de la fenêtre (mouvements de la souris à l'interieur de la zone, touches du clavier..) son passés à ce gestionnaire qui peut soit les traiter et retourner `TRUE` ou les ignorer et retourner `FALSE`.

`NPP_Initialize` Effectue des initialisations globales. Cette méthode est appelée une seule fois lors du chargement du plug-in et avant la création de toute instance. Les ressources allouées au niveau de cette méthodes sont, en principe, libérées par `NPP_Shutdown()`.

`NPP_New` Crée une nouvelle instance d'un plug-in et passe un paramètre mode qui informe sur l'aspect su plug-in: `NP_EMBED` pour un affichage dans une sous-fenêtre de la page HTML ou `NP_FULL` pour un affichage pleine fenêtre.

`NPP_NewStream` Informe une instance de l'arrivée d'un nouveau flux et fournit son type MIME. Un poiteur sur un entier `stype` est passé pour permettre au plug-in de choisir la manière dont it aime traiter le flux. Un plug-in peut traiter un flux en entrée par une série d'appels à `NPP_WriteReady()` et `NPP_Write()` (`NP_NORMAL`) ou laisser Navigator télécharger la totalité du flux et le traiter comme fichier ordinaire (`NP_ASFILEONLY`).

`NPP_Print` Gère les requêtes d'impressions du contenu de la fenêtre du plug-in. Le plug-in peut explicitement demander la prise en compte de l'impression en pleine page ou de sa fenêtre seulement.

`NPP_SetWindow` Appelée pour informer le plug-in de la création d'une fenêtre en fournissant un pointeur `NPWindow`. Un pointeur `NPWindow` nul signifie que la ressource fenêtre associée est détruite. Cette méthode est appelée, aussi, chaque fois que la fenêtre nécessite d'être repeinte, comme dans le cas d'un redimentionnement par exemple.

`NPP_Shutdown` Appelée après la déstruction de la dernière instance du plug-in. Les ressources partagées par toutes les instances et allouées dans `NP_Initialize()` sont libérées à ce niveau.

`NPP_StreamAsFile` Si le plug-in répond `NP_ASFILEONLY` dans l'appel à `NPP_NewStream()`, Navigator appelle cette méthode en lui passant le nom du fichier téléchargé. Le plug-in peut alors commencer à traiter le document.

`NPP_URLNotify` Notification du plug-in sur la terminaison d'une requête d'ouverture de flux avec `NPN_GetURL()`, cette méthode fournit l'URL en question et la raison de la terminaison: normale (`NPRES_DONE`), sur interruption par l'utilisateur (`NPRES_USER_BREAK`) ou sur un problème de réseau (`NPRES_NETWORK_ERR`).

`NPP_Write` Dans le cas ou le plug-in choisit un mode NP_NORMAL, Navigator commence à lui délivrer le flux par bloc en appelant cette méthode.

`NPP_WriteReady` Appelée par Navigator pour interroger le plug-in sur le facteur de blocage des transferts, c'est à dire la taille des bloc de données qu'il est prêt à gérer.

Bibliographie

[HSW96] DARREN HARDY, MICHAEL SCHWARTZ, DUANE WESSELS, "*Harvest User's Manual*", University of Colorado at Boulder, 31 Janvier 1996.

[HaS94] DARREN HARDY, MICHAEL SCHWARTZ, "*Customized Information Extraction as a Basis for Resource Discovery*", Department of Computer Science, University of Colorado, Mars 1994.

[FrY98] F. YERGEAU, "*UTF-8, a transformation format for ISO 10646*", RFC 2279, Alis Technologies, January 1998.

[MaK97] M. KOSTER, "*A Method for Web Robots Control*" (travail en cours), Internet Draft, WebCrawler, 4 Juin 1997, http: //info.webcrawler.com/mak/projects/robots/exclusion.html.

[GSA97] F. GUIDICI, A. SAPPIA, "*An Extension to the Web Robots Control Method*" (travail en cours), Internet Draft, Université de Genoa, Italie, 22 Février 1997.

[AlL98] ALEX LANGE, "*Sorting Through Search Engines*", Web Techniques Magazine, Volume 2, Numéro 6. Mars, 1998.

[HAL96] H. ALVESTRAND, "*IETF Policy on Character Sets and Languages*", UNINETT, RFC 2277, Janvier 1998.

[KRB] K. R. BEESLEY, "*Arabic Morphological Analysis on the Internet*", Xerox Research Centre Europe, http: //www.xrce.xerox.com.

[KRB] K. R. BEESLEY, "*Arabic Finite-State Morphological Analysis and Generation*", Xerox Research Centre Europe, http: //www.xrce.xerox.com.

[BOW95] C. BOWMAN, P. DANZIG, D. HARDY, U. MANBER, M. SCHWARTZ, "*The Harvest Information Dicovery and Access System*", Transarc, Inc, University of Southern California, University of Colorado at Boulder, University of Arizona.

[HSC95] DARREN HARDY, MICHAEL SHWARTZ, "*Customized Information Extraction as a Basis for Resource Discovery*", Department of Computer Science, University of Colorado. Février, 1995.

[TAS90] MURAT TAYLI, ABDULLAH I. AL-SALAMAH, "*Building Bilingual Microcomputer Systems*", Communications of the ACM, 33, 5. Mai 1990.

[ZaO] ZAN OLIPHANT, "*Programming Netscape Plug-Ins*", http: //leaf.stpn.soft.net, ISBN 1-57521-098-3, Sams.net Publishing.

[FrY97] F. YERGEAU, G. NICOL, G. ADAMS, M. DUERST, "*Internationalization of the Hypertext Markup Language*", RFC 2070 (Standards Track), Janvier 1997.

[TBL95] T. BERNERS-LEE, D. CONNOLLY, "*Hypertext Markup Language - 2.0*", MIT/W3C, RFC 1866, Novembre 1995.

[NCI] NETSCAPE COMMUNICATIONS, INC., "*Plug-ins SDK Reference Manual*".

[KAM97] ERIC GOODWIN, "*Intelligent Information Retrieval*", Knowledge Asset Media, 1997.

[AlJ] ALBERT JANSSEN, "*Système XWindow, la bible du programmeur Xlib*", Eyrolles, pp. 317-370.

[HNu93] H. NUSSBACHER, "*Handling of Bi-directionnal Texts in MIME*", RFC 1556, Israeli Inter-University Computer Center, Decembre 1993.

[IAN] IANA, "*Character Sets*", Internet Assigned Numbers Authority- Character Sets Registry, Juin 1998.

[KlL93] KLAUS LAGALLY, "*ArabTEX - a System for Typesetting Arabic, User Manual Version 3.00*", Universität Stuttgart, Fakultät Informatik, Novembre 1993.

[BoF93] N. BORENSTEIN, N. FREED, "*MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies*", RFC 1521, Bellcore, Innosoft, Septembre 1993, pp. 1-29.

[MaK95] MARTIJN KOSTER, "*Robots in the Web: threat or treat? *", NEXOR, Avril 1995.

[WGC94] WILLIAM G. CAMARGO, "*The Harvest Broker*", The Pennsylvania State University, Décembre 1994.

[IsC96] ISAAC COHEN, "*CGI/Perl et Javascript*", Eyrolles 1996.

[ShG96] SHISHIR GUNDAVARAM, "*Programmation CGI pour le Web*", O'REILLY International 1996, pp. 1-102.

Index