Stéfane Fermigier
UFR de mathématiques de l'université de Paris VII, membre de l'Institut de mathématiques de Jussieu

Le moteur de recherche *FermiVista!*

FermiVista! est un moteur de recherche spécialisé dans les documents scientifiques. Sa spécificité, par rapport aux moteurs de recherche généralistes comme AltaVista, Excite, Infoseek ou Google, est de ne prendre en compte que les documents aux formats PostScript et DVI (et plus récemment PDF), et originaires de sites Web et FTP connus pour abriter des centres de recherche ou d'enseignement en mathématiques, en physique et en informatique. Aucun autre système actuel ne permet de recherche directe sur ce type de documents.

Moteur de recherche

En 1995, découvrant d'une part le World Wide Web et d'autre part le langage de programmation Python, j'entreprends à titre d'exercice de programmation de réaliser un petit moteur de recherche, dérivé du programme de vérification de liens Webchecker inclus comme exemple dans la distribution de Python.
Cette première version est déployée sur le serveur des élèves de l'ENS rue d'Ulm et indexe alors une dizaine de milliers de pages html. Il apparait cependant que, compte-tenu des moyens informatiques dont je dispose, il n'est pas possible de pousser le projet plus avant, et en particulier d'entrer en compétition avec les systèmes existants (AltaVista, Lycos et Infoseek étaient à l'époque les moteurs de recherche les plus riches, avec déjà plusieurs millions voire dizaines de millions de pages indexées).

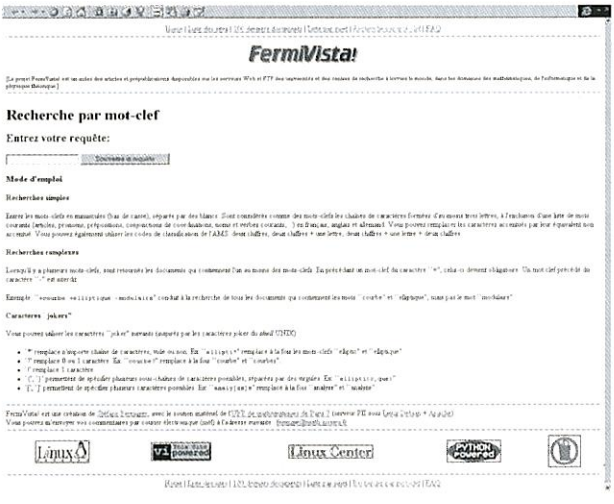
Début 1997, Raphaël Rouquier me suggère de spécialiser *FermiVista!* en un moteur de recherche sur les documents mathématiques. C'est ce que j'entreprends pendant les grandes vacances de 1997, réalisant une refonte complète du système. Cette version est développée sur mon PC personnel (sous Linux, qui n'est à l'époque pas relié à l'Internet, toute l'expérimentation se faisant donc en local), puis déployée sur le serveur (SUN) de l'Institut de mathématiques de Jussieu.

Une première liste de sites d'institutions d'enseignement ou de recherche en mathématique est constituée à partir de listes disponibles sur le Web : la cellule Mathdoc en France [http://www-mathdoc.ujf-grenoble.fr/], la Mathematics WWW Virtual Library [http://euclid.math.fsu.edu/Science/

math.html], le serveur Web de l'université de Pennsylvanie [http://www.math.psu.edu/MathLists/Contents.html] et quelques autres.

À l'automne 1997, je m'aperçois qu'il est infiniment plus efficace de commencer par indexer les sites FTP anonymes plutôt que les sites WWW. En effet, pour un site FTP, on a la possibilité, grâce à la commande ls-R, de connaître la liste complète des fichiers du site, et donc de repérer facilement ceux qui ont un suffixe intéressant (.ps, .ps.gz, .dvi, .dvi.gz, etc.). Une liste de sites FTP anonymes intéressants pour le projet est constituée à partir de la FAQ sur les sites FTP anonymes [ftp://ftp.iaehv.nl/pub/users/perry/ftp-list/ftp-list.zip] et du résultat de recherches sur le serveur norvégien FTPsearch (racheté maintenant par Lycos : http://ftpsearch.lycos.com/). Il apparaît par ailleurs que, compte tenu de la frontière floue qui existe entre les mathématiques, l'informatique et la physique théorique, de nombreux documents issus de ces deux derniers domaines ont fait leur chemin dans la base de données. Je décide donc d'élargir le champ du projet et de rajouter des sites (Web et FTP) qui correspondent à ces domaines.

En janvier 1998, l'UFR de mathématiques de Paris VII met à la disposition du projet une machine dédiée, puissante pour l'époque (Pentium II cadencé à 233 MHz avec 192 Mo de RAM, sous Linux). Le portage de Solaris à Linux se déroule en quelques heures. Pendant l'été 1998, je modifie le système pour utiliser une base de données SQL (MySQL). En mars 1999, *FermiVista!* contient les références à environ 220 000 documents scientifiques obtenus après exploration de plus d'un million d'URL. En un peu plus d'un an, le système a répondu à environ 200 000 requêtes.



Utilisation

Le langage de requêtes de *FermiVista!* est dérivé de celui d'Altavista : l'utilisateur rentre une liste de mots (de plus de trois lettres, sans chiffres et sans caractères spéciaux) et le système renvoie, par paquet de cinquante, une liste de documents qui contiennent ces mots. Les pages qui arrivent en premier sont celles qui contiennent le plus de mots de la liste. Pour chaque document, le système indique le score obtenu pour la requête, la date de dernière modification et une liste d'URL où le document est disponible.

Il est possible de réaliser des requêtes avancées, en utilisant les modificateurs « + » (qui impose la présence d'un mot) et « - » (qui exclut les documents qui contiennent un mot). Il est aussi possible d'utiliser des caractères joker : « ? » qui remplace zéro ou une lettre, « ! » qui remplace une et une seule lettre et « * » qui remplace un nombre quelconque de lettres.

Le langage de requêtes ne prend pas en compte les opérateurs *ou*, *et* et *non* : l'opérateur *ou* est implicite et les opérateurs *et* et *non* peuvent être essentiellement simulés à l'aide des modificateurs « + » et « - ». Il n'est cependant pas possible, à l'heure actuelle, de construire des requêtes complexes avec des parenthèses.

Architecture

Les composantes principales du système sont : la base de données, qui contient les informations sur les documents visités, l'arpenteur (crawler) qui explore le Web et alimente la base, l'indexeur, qui construit à partir de la base un index inversé, structure de données qui facilite la recherche par mots-clefs, le script d'interrogations, qui répond aux requêtes des utilisateurs, et enfin un petit nombre d'utilitaires annexes.

La base de données

La base de données contient deux tables : une table d'informations sur les URL et une sur les sites explorés. La table des URL contient toutes les URL dont le système est conscient, qu'elles aient été déjà visitées ou qu'elles soient destinées à l'être un jour. Pour chaque URL, elle contient à la fois les informations qui seront publiées dans les réponses aux requêtes - URL, résumé (les 200 premiers caractères, actuellement), liste de mots-clefs, taille du fichier - et les données nécessaires au fonctionnement interne du système, date de dernière visite et statut de la page. Il faut noter que pour des contraintes évidentes de place, la base de données ne contient pas le contenu intégral des fichiers indexés.

La table des sites contient des informations internes au système, principalement une copie du fichier « /robots.txt » (pour les serveurs HTTP), la date de dernière visite sur le site et surtout un dra-

peau qui indique (pour les serveurs HTTP) si l'on peut explorer systématiquement le site. Le système de gestion de bases de données (SGBD) retenu après plusieurs essais est le logiciel libre MySQL [http://www.mysql.com/], un serveur SQL léger, rapide et fiable. Un système de base de données objets (SGBDO) serait peut-être plus à même de représenter efficacement la structure arborescente du Web, mais, faute d'un système à la fois librement disponible, de qualité industrielle et facilement interfaçable avec Python, aucune expérience n'a pu être tentée à ce jour dans ce domaine.

L'arpenteur

L'arpenteur est le programme qui va chercher les documents *via* l'Internet (protocoles HTTP ou FTP) et en analyse le contenu. Ses fonctions sont les suivantes :

- rapatrier les documents *via* l'Internet, de préférence de façon multiplexée pour ne pas être pénalisé par la lenteur de certaines connections. Plusieurs implémentations ont été essayées, notamment le *multithreading* et l'utilisation d'une boucle *select*, mais force est de constater que dans un univers où les connections ne sont pas fiables (au niveau des DNS ou pendant les transferts de fichiers), les librairies usuelles ne sont pas suffisantes, et qu'il faut consacrer un effort de programmation important pour tenir compte de tous les imprévus. L'approche actuelle consiste donc plutôt à utiliser tout simplement plusieurs instances du programme *wget* en parallèle (actuellement 8 au maximum). *Wget* est un utilitaire qui sait rapatrier des fichiers selon les protocoles HTTP et FTP, il a été testé par de nombreux utilisateurs, et l'utilisation de processus indépendants permet d'éviter les blocages dus aux serveurs DNS. Les pénalités en termes de performances dues à l'utilisation de processus externes sont totalement négligeables dans le cadre de ce projet :
- pour les documents HTML, les analyser pour en extraire les URL et ajouter dans la base de données les URL qui ne sont pas encore connues du système et qui sont situées sur des sites « explorables » ;
- pour les documents PostScript, DVI ou PDF, les faire passer dans des filtres pour en extraire du texte ISO-latin (les langues non latines ne sont actuellement pas gérées par le système). Pour DVI, j'utilise actuellement « dvi2tty » puis un filtre en *sed*. Pour PostScript, j'utilise soit « ps2ascii » (qui fait partie de la distribution de GhostScript), soit « pstotext » lorsque *ps2ascii* n'arrive pas à interpréter correctement le fichier (*pstotext* est plus robuste mais plus lent que *ps2ascii*). Les deux programmes nécessitent un interpréteur PostScript (ghostscript, dans mon cas). Enfin les

documents PDF sont transformés en PostScript à l'aide du programme « pdftops » et on se ramène au cas précédent. C'est actuellement la composante du système la moins satisfaisante, car le rendu « à plat » (en tant que suite de caractère ISO-latin) d'un texte à la mise en page complexe ou présentant des formules mathématiques ou des diagrammes peut s'avérer très approximatif. Certains documents PostScript conduisent même à des erreurs de l'intepréteur GhostScript ;
- à partir du texte ISO-latin des documents à indexer, en extraire un résumé (en l'occurrence, les 200 premières lettres) et une liste d'au plus 200 mots-clefs. Sont évidemment exclus de la liste des mots-clefs les mots les plus courants et jugés non significatifs pour les requêtes dans les langages les plus courants (anglais, français, allemand, néerlandais...).

Quelques remarques :
- l'arpenteur s'astreint, comme c'est l'usage en pareille circonstance, à respecter un temps d'attente entre deux accès à un même site. Ce temps est actuellement fixé à dix minutes ;
- le facteur limitant de l'efficacité de l'arpenteur n'est pas la bande passante réseau mais le temps CPU nécessaire au décodage des pages PostScript.

L'indexeur

Le rôle de l'indexeur est de parcourir l'ensemble de la base de données et de constuire un index inversé, c'est-à-dire un dictionnaire (une table de hachage, dans l'implémentation actuelle) qui, pour chaque mot rencontré au moins une fois dans l'ensemble des documents, retourne la liste des documents qui le contiennent. Chaque document est en fait identifié par un numéro (sur 32 bits) et une deuxième table permet de trouver toute l'information utile (URL, résumé, date de modification) pour un document de numéro donné. Le calcul de l'index inversé est une opération assez longue, qui prend une dizaine d'heures et plus d'une centaine de Mo de RAM.

La structure de données particulièrement simple utilisée dans l'implémentation actuelle est néanmoins suffisante compte tenu des contraintes. On peut trouver des structures de données et des algorithmes beaucoup plus efficaces (mais plus complexes à implémenter) dans *Managing Gigabytes: Compressing and Indexing Documents and Images* de Ian H. Witten, Alistair Moffat, Timothy C. Bell (Van Nostrand Reinhold, 1994) et dans des travaux de recherches plus récents, en particulier ceux de l'équipe de Stanford qui a développé Google [http://google.stanford.edu/google-

papers.html]. Une dernière fonction de l'indexeur est de créer une série de pages statiques (références par site et par catégorie MS pour l'instant), dont le rôle principal est de faire en sorte que le site soit lui-même indexé par les moteurs de recherche traditionnels (html).

Le script d'interrogation

Le script d'interrogation répond aux requêtes des utilisateurs rentrées à l'aide d'un formulaire Web. Ses fonctions sont les suivantes:

- analyse de la requête, pour tenir compte en particulier des modificateurs « + » et « - », puis expansion des caractères « joker »;
- construction de la liste des documents qui répondent à la requête, par suite d'opération *ou*, *et* ou *non*. Compte tenu du nombre de documents qui peuvent contenir un mot donné (plus de 40 000 par exemple pour le mot « theorem »), cette partie du système a été écrite en C, et c'est d'ailleurs la seule;
- mise en page de la réponse sous forme d'un document HTML.

Les utilitaires

Un petit nombre d'utilitaires permettent d'extraire des informations statistiques de la base de données, de modifier le statut d'un serveur ou d'ajouter des URL. D'une importance toute particulière est le programme qui explore les sites FTP connus du système (à l'aide de « ls » récursifs) pour en extraire les URL de documents avec suffixes « .ps », « .dvi » et variantes qui seront ensuite explorés par l'arpenteur.

Expériences retenues

Le langage Python est particulièrement bien adapté au développement rapide d'applications pour l'Internet. La seule partie qui posait un problème réel de performances a été réécrite très facilement en C, à l'aide de l'utilitaire SWIG. Force est de constater cependant que l'absence de vérification statique de typage est une source de perte de temps, de plus en plus importante à mesure que le projet devient plus complexe.

L'Internet est un monde plein de surprises, et cela demande pas mal d'expérimentation avant de découvrir les plus ennuyeuses. Un seul exemple : il est facile, à l'aide d'un lien symbolique d'un répertoire sur un de ses parents, de créer une infinité (en pratique, un peu moins quand même) d'URL qui renvoient toutes au même document.

Heureusement, les producteurs de textes scientifiques n'en sont pas encore à « spammer » (utiliser des astuces pour que leurs textes

soient référencés favorablement) *FermiVista!*, comme c'est le cas avec les moteurs de recherche grand public.
La nature exacte d'un moteur de recherche reste mystérieuse pour certains utilisateurs qui s'imaginent par exemple qu'on peut leur faxer le contenu d'un article. D'autres s'inquiètent qu'on puisse publier des informations « confidentielles », qu'ils ont eux-mêmes rendues librement disponibles sur leurs serveurs FTP anonymes ! D'autres encore, et notamment au sein de l'éducation nationale, sont tellement traumatisés par les diverses mises en gardes juridiques qu'ils se sentent obligés de demander la permission avant de référencer le site dans leurs propres pages.

Développements futurs

FermiVista!, bien que pleinement utilisable depuis plus d'un an, est un travail en cours avec de nombreux points non encore implémentés ou à améliorer. En voici, une liste, non exhaustive :

- le mécanisme de classification des pages, basé au départ sur les catégories de l'AMS (codes MSC), est malheureusement très incomplet. J'envisage un mécanisme de classification lexicale basé sur la similitude avec des documents dont le code MSC est connu. L'une des difficultés de la mise en œuvre de ce procédé sera l'existence de documents dans plusieurs langues différentes ;
- un effort particulier devrait être consacré pour améliorer la qualité du décodage des fichiers PS et DVI. Le problème est relativement facile à résoudre avec DVI, qui est un format de très bas niveau, donc facile à décoder, et qui contient par ailleurs des informations précises sur les polices de caractères employées. Pour PostScript, et en particulier pour le PostScript craché par dvips, c'est infiniment plus dur, tout simplement parce que PostScript est un langage de programmation à part entière. La diversité des encodages de caractères utilisés par les différents outils qui produisent du PostScript (dvips, Word, FrameMaker, etc.) et l'impossibilité de retrouver à partir d'un fichier PostScript les tables de caractères employées (qui varient de toute façon au sein d'un même document, au moins dans le cas des textes mathématiques produits par TeX) rendent l'exercice très difficile ;
- comme il reste encore plusieurs centaines de milliers de pages intéressantes qui n'ont pas encore été visitées, le mécanisme de vidage hors de la base de données des pages qui ont disparues entre temps (cela arrive encore souvent pour les preprints, lorsque leurs auteurs se sentent obligés de se soumettre aux règles imposées par les éditeurs de revues qui imposent une cession totale du copyright sans aucune contrepartie) n'a pas encore été implémenté proprement ;

- un nettoyage manuel ou semi-automatique de la base de données pour éliminer les sites qui n'ont en fait pas de pages PS ou DVI, les sites qui n'ont pas de rapport avec les sujets scientifiques traités ou les parties de sites qui sont en fait des miroirs d'autres sites devra un jour être mis en œuvre.
- une limitation actuelle du langage de requêtes est l'impossibilité de tenir compte de la proximité des mots-clef. Pour répondre à la demande (légitime) de nombreux utilisateurs d'ajouter cette fonctionalité, la structure de donnée de l'index inversé devra être complètement revue, et en fait il faudra faire repasser l'arpenteur par tous les documents, ce qui représente une année au moins de temps CPU.

Stéfane Fermigier
fermigier@fermigier.com
http://fermivista.math.jussieu.fr/