FDF-SEARCH

un outil de recherche de texte libre dans les annotations de séquences.

Rapport de stage

Javier Iglesias [1]
sous la direction de Cornelis Victor Jongeneel [2]

Institut Suisse de Bioinformatique
Université de Lausanne

29 octobre 2001

1. javier.iglesias@iismail.unil.ch
2. victor.jongeneel@isrec.unil.ch

Table des matières

Chapitre 1

Introduction

Ce rapport conclut le travail de stage du DEA de bioinformatique[1] 2000-2001, dirigé par Victor Jongeneel dans son groupe lausannois[2] de l'Institut Suisse de Bioinformatique[3].

Il s'agissait de fournir une interface web pour la recherche de texte libre dans les annotations de séquences à l'aide d'une machine dédiée et optmisée au niveau matériel. La principale difficulté étant de rendre l'utilisation du langage de requête spécifique parfaitement transparent pour l'utilisateur, tout en parvenant à exploiter ces spécificités.

Le résultat est accessible sur le web à l'adresse :
`http://www.isrec.isb-sib.ch/cgi-bin/dea/jiglesia/fdf/fdf-search`

1.1 problème

Les systèmes de recherche comme `SRS`[4] ou `Entrez`[5] font un travail important. Tous deux fonctionnent sur la base d'une indexation. Les recherches se faisant sur les index, elles sont extrêmement rapides, mais le mécanisme souffre de quelques problèmes :

temps il faut indexer presque chaque mot des bases de données, ce qui demande un temps colossal, qui augmente (*back of the envelop estimation*) selon un ordre $n.log(n)$ avec la quantité de données à indexer.

index l'indexation se fait sur presque chaque mot... justement : *presque*. Les recherches exécutables se limitent donc aux mots de l'index, ce qui peut poser des problèmes. Ainsi, si l'on est intéressé par toutes les entrées contenant le mot *interleukin-2* dans le champ *description* de SWISSPROT, `SRS` pourra faire un bout du travail. Il est par contre impossible de rechercher les entrées qui mentionneraient *interleukin 2* ou *interleukin II* dans n'importe quel champ.

1. `http://www.isb-sib.ch/DEA`
2. `http://www.isrec.isb-sib.ch`
3. `http://www.isb-sib.ch`
4. `http://www.expasy.ch/srs5`
5. `http://www.ncbi.nlm.nih.gov/Entrez`

inconsistence étant donné le temps nécessaire pour l'indexation, celle-ci n'est lancée que périodiquement. Entre chaque indexation, les nouvelles informations ajoutées ne sont donc pas prises en compte pour les recherches.

La combinaison d'une machine optimisée (FDF) et d'un langage spécialisé (PSL) permet d'exécuter des recherches dans les bases de données en utilisant du texte libre, c'est-à-dire n'importe quelle suite de caractères, sans nécessairement qu'elle appartienne à un dictionnaire pré-établi. Le contre-coup étant qu'il faut remachouiller toutes les données à chaque recherche.

1.2 FDF

Le Fast Data Finder (FDF) est un système de recherche de texte optimisé au niveau matériel. Cette machine a été développée et vendue par la firme californienne Paracel[6], dans le but de rechercher, filtrer et catégoriser des quantités massives de texte libre.

Ces fonctions de recherche sont accomplies sur une architecture massivement parallèle, implémentée par une technologie d'*Application-Specific Integrated Circuits* (ASIC). Des milliers de processeurs travaillent de concert pour trouver des requêtes et des patterns dans des flots de données ou des archives.

Les bases de données à traiter sont dans un premier temps chargées sur la machine, qui les formate à sa façon de manière à optimiser les recherches. C'est alors que l'on utilise le *Pattern Specification Language* (PSL) pour décrire les requêtes à exécuter.

La machine spécifiquement utilisée lors de ce stage est une version assez ancienne du GeneMatcher[7], plus connu dans l'Institut Suisse de Bioinformatique pour ses performances exceptionnelles[8] lors de l'exécution des algorithmes d'alignement de séquence comme `Fasta` ou `Blast`.

1.3 PSL

Il s'agit de l'acronyme de *Pattern Specification Language*, le nom du language de spécification des requêtes utilisé par le FDF. Il offre les fonctions communes à la plupart des languages :

expressions booléennes "contient les mots 'insuline' et 'récepteur'".

listes "contient au moins deux mots dans la liste 'insuline', 'récepteur', 'binding', 'foie'".

plus un certain nombre d'autres, rares, voire uniques :

proximité "contient les mots 'insuline' et 'récepteur' dans la même phrase".

tolérance aux erreurs et aux orthographes alternatives. "contient le mot 'insuline', en acceptant une faute d'orthographe, une lettre en plus, ou une lettre en moins".

macros "définir l'expression 'interleukin2' comme étant le mot 'interleukin-2', ou 'interleukine-2', ou 'IL-2'".

6. `http://www.paracel.com`
7. `http://www.paracel.com/products/genematcher.html`
8. `http://www.paracel.com/html/gmfeatures.html`

requêtes emboîtées "recherche, dans les résultats de la recherche précédente, le mot 'insuline'".

codages non-ascii les codages kanji, ou cyrillic par exemple.

Ces singularités permettent de composer des requêtes qui sont difficilement imaginables, voire impossible à exprimer, au moyen de langages pourtant puissants comme les *regexps*.

Comme les données, la requête doit être compilée avant d'être exécutée.

1.4 prérequis

Voici la liste du matériel et autres logiciels nécessaires au bon fonctionnement de `FDF-SEARCH`. Ces différents outils se combinent (Fig. 1.4) pour permettre à un utilisateur connecté au web d'interroger le `FDF` comme s'il était assis devant la machine hôte.

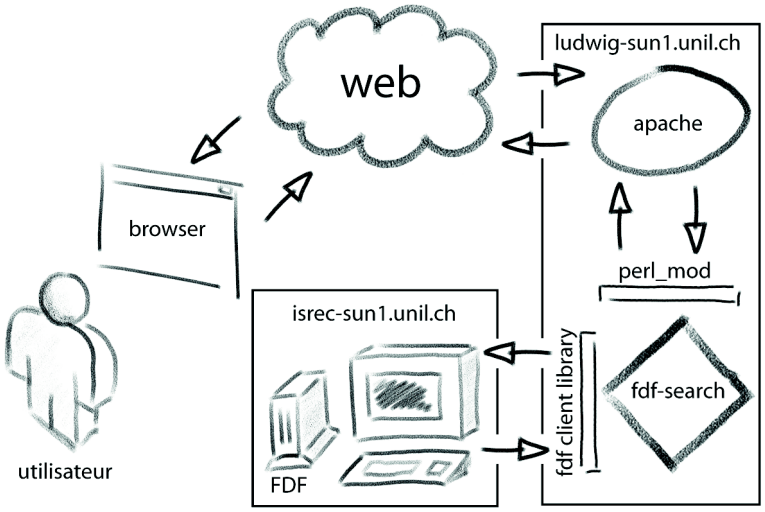

FIG. 1.1 – *La manière de laquelle les différents programmes se combinent.*

1.4.1 `FDF` hardware

Installé, configuré et en marche.

C'est le plus difficile à obtenir : c'est cher, et rare !

Il se connecte au bus SCSI d'une machine hôte quelconque.

Peu importe la machine, tant que le fichier `FDF-SEARCH.CONFIG` est modifié en conséquence.

1.4.2 `fdf` client library

Installée, configurée et accessible sur la machine exécutant le serveur web.

Il s'agit du paquet logiciel fourni avec le hardware.

Il contient les programmes permettant de charger les bases de données dans le FDF, de compiler les requêtes, d'exécuter les requêtes, de browser les résultats, etc...

La version 1.200 de la librairie est nécessaire pour pouvoir se connecter au vieux FDF, alors que la version 1.400 est utilisée avec le GENEMATCHER.

Peut-être placé n'importe où dans le système de fichier, tant que le fichier FDF-SEARCH.CONFIG est modifié en conséquence.

1.4.3 serveur web

Installé, configuré avec le support CGI et allumé.

Indispensable pour servir les pages HTML.

Le serveur web le plus utilisé est apache[9], et c'est précisément celui qui équipe l'Institut Suisse de Bioinformatique.

1.4.4 client web

Installé et configuré.

Indispensable pour visualiser les pages HTML que le script génère.

Tous les browsers connus (Navigator, Opera, HotJava, IExplorer, ...) devraient fonctionner. Il faudrait vraiment aller chercher une version de 1992 pour voir apparaître un problème! Cependant, il n'est pas interdit de trouver que je n'ai pas bon goût pour associer les formes et les couleurs.

1.4.5 perl et perl_mod

Installés et configurés.

perl[10] est un langage de script interprété, utilisé dans le monde un*x pour automatiser les tâches courantes d'administration. C'est un langage apprécié sur le web pour la réalisation de scripts CGI. De manière à pouvoir exécuter des scripts CGI écrits en perl au travers d'apache, il faut en outre installer le module perl_mod.

Oh! inutile de demander la version nécessaire... Je suppose qu'une version 5.0 ou supérieure devrait marcher.

perl peut-être placé n'importe où dans le système de fichier, tant que la première ligne de FDF-SEARCH est modifiée en conséquence. Quant à perl_mod, se référer à son manuel.

9. http://www.apache.org
10. http://www.perl.com

Chapitre 2

Description

2.1 architecture de `fdf-search`

FDF-SEARCH est un script CGI écrit en `perl` qui génère un formulaire, que l'utilisateur remplit pour décrire la requête qu'il souhaite effectuer. Une fois la recherche lancée, le script renvoie le formulaire – dont les champs contiennent encore les valeurs entrées précédemment – ainsi qu'un petit message demandant à l'utilisateur clémence et patience, le temps d'effectuer la recherche...

Afin de rendre plus claire la maintenance du script (?), et sachant que son temps d'exécution n'est pas le facteur limitant, les différentes ressources qui lui sont nécessaires ont été réparties dans plusieurs fichiers, décrits ci-après.

Vous trouverez des pointeurs sur le code source des différents fichiers dans l'annexe D, p.31.

2.1.1 `fdf-search`

Le script CGI écrit en `perl`, à proprement parler.

Répond aux requêtes des utilisateurs, compose et exécute (si nécessaire) la recherche en tenant compte des informations qui lui sont fournies.

Nécessite d'être placé dans un répertoire depuis lequel le serveur web peut l'exécuter en CGI, avec les droits d'exécution nécessaires.

2.1.2 `fdf-search.config`

Code `perl` déclarant et initialisant les variables requises par FDF-SEARCH, comme par exemple le nom de la machine hôte du FDF, ou les noms des bases de données.

Le contenu de ce fichier aurait tout aussi bien pu être inclu dans FDF-SEARCH, mais cela me semblait moins clair... non?

Doit être placé dans le même dossier que FDF-SEARCH, sans pour autant nécessiter les droits d'exécution.

2.1.3 `fdf-search.macros`

Fichier texte contenant une série de définitions en PSL qui sont compilées en même temps que toute requête.

Si nécessaire, les commentaires – compris entre `/*` et `*/` – pourraient être retirés... mais le fichier serait bien plus difficile à comprendre ! Sachant que le temps de compilation est ridiculement court comparé au temps d'exécution, ça n'apporte rien de les retirer.

Doit être placé dans le même répertoire que FDF-SEARCH.

2.1.4 fdf-search.format

Script `perl` qui formate les entrées SWISSPROT en HTML (`yaspf`).

Ce bout de code a été séparé de FDF-SEARCH pour faciliter son exécution depuis le script CGI, et pour rendre les modifications du formatage plus aisées.

Doit avoir les droits d'exécution pour le serveur web, et peut être placé n'importe où, tant que FDF-SEARCH.CONFIG est modifié en conséquence.

2.1.5 fdf-search.skel

Fichier HTML servant de gabarit à FDF-SEARCH pour générer les pages.

Peut être placé n'importe où, tant que FDF-SEARCH.CONFIG est modifié en conséquence.

2.1.6 RecallableWidgets.pm

Module `perl` utilsé par FDF-SEARCH pour initialiser et remplir le formulaire avec des données sensées (?) entre deux générations du formulaire pour le même utilisateur.

L'idée est d'aider l'utilisateur à trouver la bonne requête en copiant les valeurs précédentes pour formuler la suivante. De cette manière, on évite le bouton `back` et les erreurs par copie à la main.

Peut être placé n'importe où, tant que la directive `use RecallableWidget;` reste capable de le retrouver.

2.2 formulaire

Au premier appel à FDF-SEARCH[1], celui-ci renvoie un formulaire (Fig. 2.2) permettant à l'utilisateur de décrire la recherche qu'il souhaite exécuter. Le design du formulaire est basé sur celui de SRS, tout en laissant transparaître les spécificités du PSL comme la recherche de mots partiels ou complets, ainsi que la tolérance.

Pour soumettre une requête effective, il faut procéder aux étapes suivantes :

1. sélectionner le `field` de l'entrée où le texte doit se trouver,
2. décider si le texte est un `complete word` ou un `partial word`,
3. taper le texte à rechercher dans le champ `search for`,
4. décider si le texte doit être exact ou si une `tolerance` s'applique,
5. oublier le mode `expert` (vous n'êtes pas encore prêt),
6. sélectionner la base de données dans laquelle faire la recherche (`search in`),

1. `http://www.isrec.isb-sib.ch/cgi-bin/dea/jiglesia/fdf/fdf-search`

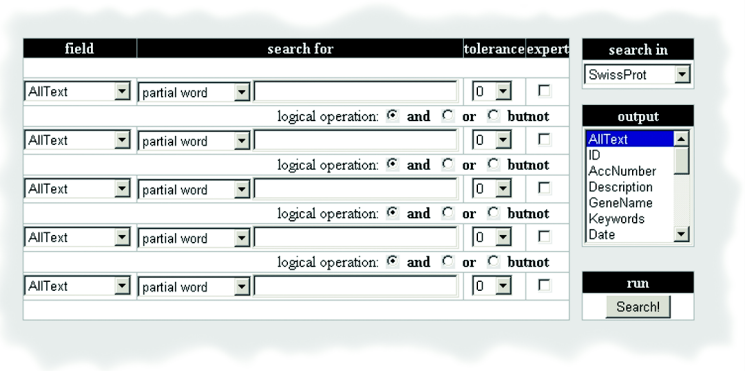

FIG. 2.1 – *Capture d'écran partielle [Windows 2000, Netscape 4.78] du formulaire généré par* `FDF-SEARCH`.

7. choisir les champs souhaités dans l'`output`,
8. presser le bouton `Search!`,
9. patienter deux minutes, le temps que les résultats formatés apparaissent.

La signification de la plupart des champs devraient se comprendre naturellement. Au cas où ce ne serait pas le cas, les détails des champs se trouvent en annexe.

2.3 script

Lorsque l'utilisateur appuie sur le bouton `search!` du formulaire (Fig. 2.2), le script `FDF-SEARCH` reçoit toutes les informations nécessaires pour composer une requête qui soit syntaxiquement correcte en `PSL`. Le script CGI renvoie le formulaire rempli à l'utilisateur, puis ouvre un `pipe` faisant appel à la librairie cliente du `FDF` au travers de la commande en ligne `fdf`.

La ligne de commande complète est composée de plusieurs programmes à la queueleuleu (Fig. 2.3) :

`fdf psl ...` compile la requête écrite en `PSL` ainsi que les macros définies dans `FDF-SEARCH.MACROS`.

`fdf search ...` prend une requête compilée en entrée pour rendre une liste de localisation des matchs.

`fdf browse ...` prend une liste de localisation de matchs en entrée et retourne les entrées complètes qui y sont mentionnées.

`fdf-search.format` prend des entrées au format SWISSPROT en entrée, les parse, et les formate en HTML, ne gardant que les lignes qui lui sont demandées en paramètre.

Vous trouverez plus de détails dans l'annexe C.4, p.28.

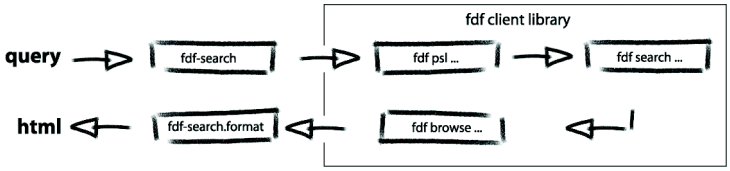

FIG. 2.2 – FDF-SEARCH *et* FDF-SEARCH.FORMAT *enrobent la* `fdf` *client library pour la rendre accessible sur le web.*

2.4 résultats des recherches

Le résultat net des recherches fructueuses par FDF-SEARCH est une liste d'entrées formatées (Fig. 2.4) ne présentant que les lignes requises, et dont l'ID est un lien hypertexte vers l'entrée complète de SWISSPROT au travers d'EXPASY [2].

FIG. 2.3 – *Capture d'écran partielle [Windows 2000, Netscape 4.78] d'une recherche fructueuse par* FDF-SEARCH*, formatée par* FDF-SEARCH.FORMAT.

2. http://www.expasy.ch

Chapitre 3

Problèmes rencontrés

Dans l'ensemble, la difficulté de ce projet n'était pas très élevée. Les choses les plus complexes n'étaient pas à réaliser, mais à comprendre...

3.1 apprentissage du PSL

Le PSL est un langage teigneux... il regorge de petites subtilités. Ses concepteurs ont débordé d'imagination au moment de donner aux caractères spéciaux des significations qui diffèrent de celles que l'on trouve dans la plupart des autres langages (que je connaisse).

C'est ma fois le seul point où je ne puisse pas faire de tutorial, howto, ou autre : il faut se plonger dedans... [1]

3.2 manipulation du FDF

Les plus gros problèmes rencontrés ont tourné autour des connections avec le FDF. C'est la raison pour laquelle ce document est plutôt technique, et regroupe un grand nombre de marches à suivre en annexes:

- chargement des bases de données (annexe B, p.25),
- configuration du shell (annexe C.1, p.27),
- utilisation de la commande `fdf`(annexes C.2 et C.3, p.28),
- exécution d'une requête (annexe C.4, p.28).

Que le programme FDF-SEARCH soit effectivement utilisé à l'avenir ou non, ce document devrait au moins permettre à quiconque de se connecter sur le FDF et d'y exécuter des recherches grâce à des indications, me semble-t-il, plus claires et plus précises que celles du manuel. Elles ont au moins le mérite d'être en français :).

3.3 module RecallableWidgets.pm

Je l'admets... la partie la plus complexe du script n'est pas le script, mais le module `perl` qui l'accompagne et qui permet de recopier les données tapées par l'utilisateur dans le formulaire retourné par FDF-SEARCH.

C'est sur cette librairie que j'ai passé le plus clair de mon temps, parce qu'il me semblait essentiel, du point de vue de l'utilisateur, que l'interface soit la plus facile d'utilisation possible. Faire réapparaître le formulaire rempli à chaque appel me semblait donc essentiel.

3.4 machine à café

J'ai eu pas mal de peine à me familiariser avec cet engin venu de l'espace pour parvenir à en extraire cette liqueur des dieux.

Chapitre 4

Problèmes non-résolus

Certains problèmes ne sont pas résolus à la fin de ce stage :

4.1 synchronisation

Pour être réellement utilisable, les bases de données du `FDF` devraient être synchronisées avec les autres. Les scripts nécessaires sont disponibles à l'Institut Suisse de Bioinformatique, ce n'est donc pas un problème insoluble.

Une question reste cependant ouverte : les 16 minutes et plus nécessaires au chargement de SWISSPROT sont-elles réellement justifiées ?

4.2 temps de recherche

Les recherches s'exécutent séquentiellement.

Une recherche dans SWISSPROT (septembre 2001, 300Mo) prend au minimum deux minutes pour s'accomplir.

1. Il est impossible d'empêcher un utilisateur impatient de presser le bouton `stop` avant de tenter de renvoyer sa requête... ignorant que sa première requête est toujours en train de s'exécuter, et qu'il faudra attendre qu'elle soit finie avant d'exécuter la suivante, le pénalisant d'une minute encore... Il y a probablement de quoi rendre fou l'utilisateur moyen (et forcément pressé) !
2. Il n'est pas exclu que le serveur et/ou le client web considèrent comme un *time out* et décident de clore une connection ouverte deux minutes durant, sans qu'aucune donnée ne passe au travers.
   Il semblerait que des solutions transférant des bits à temps régulier aient déjà été développées pour l'Institut de Bioinformatique par Laurent Falquet. C'est la raison pour laquelle je n'ai pas implémenté une solution propre.

4.3 sécurité ?

Un novice écrivant un script CGI avec `perl` peut créer d'énormes trous de sécurité sur la machine hôte... je suggère donc avec insistance que mon code

soit *reviewé* avant d'être mis en production !

4.4 machine à café

C'est toujours la même machine qui martyrise les stagiaires.

Chapitre 5

Conclusion

FDF-SEARCH est accessible sur le web, et donne les résultats escomptés... mais à quel prix?

Le web a contribué cette dernière décennie à nous habituer à obtenir tout et vite. QUI est encore prêt aujourd'hui à attendre deux minutes pour obtenir le résultat d'une recherche dans SWISSPROT? Peut-être n'ai-je pas complètement conscience des enjeux qui se cachent derrière FDF-SEARCH, mais j'entends déjà s'écrier l'utilisateur d'un ton irrité "que c'est lent..." au mieux, et au pire "mais ça marche pas leur !§&$* de machin !".

Afin de réduire la frustation de l'*Homo sapiens celer*, je suggérerais d'utiliser FDF-SEARCH d'une manière légèrement différente.

Le formulaire offre actuellement une seconde base de données, SH2, qui est une vieille version de SWISSPROT dont on a extrait les protéines dans lesquelles le domaine SH2 a été détecté. Les recherches dans ce sous-ensemble ne prennent qu'une quarantaine de secondes. Peut-être serait-il judicieux de découper SWISSPROT en sous-ensembles avant de les charger dans le FDF? Une fois encore, c'est une question de prix! Sans compter que les critères de découpage ne sont pas forcément évidents... pourquoi pas au coup par coup, et réserver cet outil aux membres de l'Institut Suisse de Bioinformatique, qui pourraient y charger ponctuellement une portion d'intérêt de la base de données, et libérer l'espace une fois les recherches terminées?

Il serait également imaginable de demander à FDF-SEARCH de faire courir la recherche hors-ligne, et d'envoyer les résultats, une fois obtenus, par e-mail. Je ne pense pas qu'un outil de recherche qui n'est pas au moins un peu interactif ait la moindre chance d'être utilisé. Une fois encore, il se pourrait que je sois à côté de la plaque!

Chapitre 6

Remerciements

Tout d'abord, merci à *Victor Jongeneel* pour m'avoir proposé de suivre le DEA, mais également pour m'avoir donné la possibilité de le suivre en dehors des chemins battus... hum...

A *Patricia Palagi* – la "maman" des étudiants du DEA – à qui j'ai donné un nombre incalculable de soucis, complications, maux de tête et ulcères, à force de battre les chemins qui ne l'étaient pas encore...

A *Marco Pagni* pour son aide inestimable lors de mes premiers pas avec le `FDF`, ainsi que pour les détails inextricables de `perl` et des regular expressions.

Aux Professeurs *Alessandro Villa* et *Marco Tomassini* pour m'avoir laissé suivre le DEA durant mon travail de diplôme et de thèse respectivement.

A *Laurent Bucher*, *Mathieu Perrenoud* et *Denise Hayward* pour leurs encouragements, relectures, corrections et commentaires.

A mes *deux réveils-matin*... sans lesquels je n'aurais jamais réussi à prendre le train tous les matins à 6h30 pour me rendre à Genève, alors que je me couchais à 1h00 pour parvenir à finir mon travail de diplôme.

A mes *parents* pour tout le reste !

Annexe A

détails des champs du formulaire

Voici quelques détails complémentaires sur les divers champs du formulaire généré par `FDF-SEARCH`.

`field` type de ligne SWISSPROT où le texte doit être trouvé.
Les noms proviennent de la définition des entrées SWISSPROT.
Exemples :
- sélectionnez `Authors` si le texte cherché se trouve sur une ligne commençant par `RA` dans l'entrée.
- selectionnez `AllText` si le texte doit juste être quelque part dans l'entrée.

`partial word/complete word` définit le type de recherche à exécuter.
Exemples :
- une recherche `partial word` du mot 'cat' matchera 'cat', 'cats', et 'concatenation'.
- une recherche `complete word` du mot 'cat' ne matchera que 'cat', ni 'cats', ni 'concatenation'.

`search for` texte à rechercher `;)` Les détails exacts de la recherche sont définis par les autres champs.
Exemple :
- rechercher le mot 'alkaloid' trouvera toutes les occurrences de la chaîne 'alkaloid' dans la base de données.

`tolerance` Le `FDF` permet de matcher des chaînes qui *ressemblent* au texte recherché. Il s'agit d'une fonctionnalité unique, très utile pour tenir compte des fautes d'orthographe et des orthographes alternatives. Une recherche avec une tolérance de deux matchera le texte exact, ainsi que le texte ayant au maximum deux positions fausses, absentes ou ajoutées.
Les caractères ajoutés ne pourront l'être qu'après un début de match, jamais avant. Ainsi, une recherche de 'cat' avec une tolérance de 1 matchera 'cats', 'chat', 'hat', mais pas 'scat'.
Exemples :
- 'cat' avec une tolérance de 1 matchera :
  'cat' exact

'c at' caractère supplémentaire

'ct' caractère absent

'cmt' faute d'orthographe

'c t' faute d'orthographe

`expert` Pour éviter les frustrations, les requêtes sont normallement contrôlées avant d'être compilées.

Si vous vous sentez chanceux, ou que vous possédez les connaissances suffisantes, vous pouvez débrancher ces contrôles en sélectionnant le mode `expert`. Le `PSL` est un langage assez particulier... préparez-vous à composer des requêtes qui ne se compileront pas !

Il est cependant nécessaire d'utiliser le mode `expert` pour faire des recherches *case sensitives*.

- rechercher une lettre minuscule (`a`) matchera à la fois les lettres minuscules et majuscules (`a` et `A`).
- rechercher une lettre majuscule (`A`) ne matchera que les lettres majuscules (`A`).
- pour ne rechercher que les lettres minuscules, il faut les "échapper" (`\a`).

Le mode non-`expert` ne fait que des recherches *case insensitive*.

Exemples :

- Si vous recherchez 'Cat' écrit de cette manière, il vous faut brancher le mode `expert` et taper la chaîne 'C\a\t' dans le champ `search for`.
- Pour rechercher à la fois 'Cat' et 'CaT', passer en mode `expert`, et tapez 'C\at'.

Il est également nécessaire d'utiliser le mode `expert` pour pouvoir utiliser les *wildcards*.

- '*' n'importe quel nombre de caractères alphanumériques (*regexp* '\w*')
- ' ' n'importe quel nombre de caractères d'espacement, tabulation, ou saut de ligne (*regexp* '\s*')
- '.' un seul caractère quelconque (*regexp* '.')
- '?' un seul caractère alphanumériques (*regexp* '\w')
- '_' un seul caractère d'espacement (*regexp* '\s')
-

En mode non-`expert`, tous les *wildcards* sont "échappés".

`opérations logiques` combinent les différentes lignes de manière logique.

La précédence est définie par l'ordre des lignes de la requête.

En français, une requête complète pourrait se lire :

```
recherche les entrées de "swissprot" qui ont "human" comme
"organisme" ET "récepteur" dans les "commentaires" OU "shah"
dans les "auteurs"
```

et s'écrirait

```
(("human" in "OS") and ("receptor" in "CC"))
or ("shah" in "RA")
```

`search in` définit la base de données dans laquelle il faut chercher.

Une seule base peut être cherchée à la fois.

`output` liste les champs qui doivent apparaître dans la sortie formatée.

Sélectionnez `AllText` pour obtenir l'entrée complète (sans la séquence).

Sélectionnez `none` pour n'obtenir que la liste des `ID`.

`Search!` lance la recherche.

Soyez patient ! Chaque recherche doit traverser la base de données au complet, ce qui peut prendre près de deux minutes.

Annexe B

mini-howto : mise à jour de swissprot

Cette section liste les commandes à appeler pour mettre à jour la base de données SWISSPROT sur le `FDF`, en utilisant les ressources du site lausannois de l'Institut Suisse de Bioinformatique.

1. se loguer sur `ludwig-sun1.unil.ch`
   ```
   %> ssh ludwig-sun1.unil.ch -l <username>
   ```
2. retirer la version précédente de la base de données
   ```
   %> /usr/local/fdf/bin/fdf -h isrec-sun1.unil.ch \
   /fdf/swiss.ref
   %> /usr/local/fdf/bin/fdf -h isrec-sun1.unil.ch \
   /fdf/swiss.seq
   ```
3. charger la dernière version en date à l'aide de `btkload`
   ```
   %> /usr/local/fdf/bin/btkload /db/swiss-prot/swiss.dat \
   isrec-sun1.unil.ch:/fdf/swiss -frame
   ```
4. attendre...
5. attendre...
6. attendre... le 27 septembre 2001, il m'a fallu 16 minutes pour charger les près de 300Mo du fichier `swiss.dat`
7. le flag `-frame` crée un fichier `/fdf/swiss.frame` qui peut être simplement supprimé. La commande échouait dans un `core dump` si je ne mettais pas le `-frame`, alors...
   ```
   %> /usr/local/fdf/bin/fdf -h isrec-sun1.unil.ch \
   /fdf/swiss.frame
   ```
8. départ pour un café, parce que c'est fini !

Lorsqu'une base de données est chargée dans le `FDF`, elle est partagée en deux fichiers. L'un d'entre eux contient les séquences, le second les annotations. Ainsi, charger la base de données `/fdf/x` créera les deux fichiers `/fdf/x.ref` et `/fdf/x.seq`.

Annexe C

utilisation de la commande en ligne `fdf`

Parmi les outils du package `fdf` installé sur le réseau des groupes lausannois de l'Institut de Bioinformatique se trouve le programme `fdf` qui permet de se connecter directement au `FDF`.

Vous trouverez plus bas quelques détails sur la configuration nécessaire pour faire marcher le programme `fdf` convenablement, ainsi que quelques informations sur les commandes les plus utiles pour gérer le système de fichiers et les bases de données.

C.1 configuration

Avant de pouvoir utiliser les commandes présentées plus loin, il est nécessaire de suivre un certain nombre de pas, décrits ici :

1. se loguer sur `ludwig-sun1.unil.ch`
   `%> ssh ludwig-sun1.unil.ch -l <username>`
2. par défaut, les shells sont configurés pour se connecter au GeneMatcher, Deux variables d'environnement – `$PATH` et `$FDF_SERVERHOST` – doivent être modifiées pour s'adresser à la bonne machine, et avec la bonne version des librairies. Vous pouvez toujours demander plus de détails sur les variables d'environnement à votre guru `un*x`[1] préféré .

   – Pour un `csh`-like shell :
   `%> setenv PATH /usr/local/fdf/bin:$PATH`
   `%> setenv FDF_SERVERHOST isrec-sun1.unil.ch`
   – Pour un `sh`-like shell :
   `%> PATH=/usr/local/fdf/bin:$PATH`
   `%> FDF_SERVERHOST=isrec-sun1.unil.ch`
   `%> export PATH FDF_SERVERHOST`

   Il suffit alors d'appeler le programme en écrivant simplement :
   `%> fdf ...`

1. `http://www.ugu.com`

3. Si on ne souhaite pas modifier – même ponctuellement – son environnement, il est également possible de donner à chaque fois les détails complets de la commande avec :
   `%> /usr/local/fdf/bin/fdf -h isrec-sun1.unil.ch ...`

C.2 commandes courantes du système

Le programme `fdf` est conçu pour simuler le comportement des programmes standard des shells `un*x` :

`fdf help [<command>]` donne la liste des commandes disponibles, ou les détails sur l'utilisation de la `command`.

`fdf ls [-l] /fdf` liste le contenu du système de fichiers du `FDF`.

`fdf df` donne le taux d'utilisation du fichier système du `FDF`.

`fdf cp <from> <to>` copie un fichier de `from` à `to`.

`fdf mv <from> <to>` déplace un fichier de `from` à `to`.

`fdf rm /fdf/<name>` supprime le fichier `name`.

`fdf mkdir <directory>` crée un répertoire nommé `directory`.

`fdf rmdir <directory>` supprime le répertoire nommé `directory`.

`fdf chown <owner> <filename>` redéfinit le propriétaire du fichier `filename`, à la façon d'`un*x`.

`fdf chgrp <group> <filename>` redéfinit le groupe du fichier `filename`, à la façon d'`un*x`.

`fdf chmod <mode> <filename>` redéfinit le mode du fichier `filename`, à la façon d'`un*x`.

C.3 commandes courantes d'administration des bases de données

`fdf read /fdf/<database> <document>` imprime à l'écran le contenu du `document` de la base de données `database`.

`fdf dbstat /fdf/<database>` imprime à l'écran des informations précises concernant la `database`. Les informations les plus intéressantes sont : le propriétaire, la taille, le nombre d'entrées, les préfixes et suffixes ainsi que la date de la dernière modification.

`fdf docstat /fdf/<database> <document>` imprime à l'écran des informations précises sur un `document` particulier d'une `database`.

C.4 commandes d'exécution d'une requête

Une fois le texte de la requête défini en `PSL`, il faut exécuter une sérieuse ligne de commande faite de `pipes` pour obtenir les entrées dans le terminal :
`%> echo '<query>' | fdf psl | fdf search /fdf/<db> | fdf browse`
Les différents programmes qui participent sont :

`fdf psl` lit une requête écrite en `PSL` de `stdin`, et écrit sur son `stdout` la version compilée, prête à être exécutée.

`fdf search /fdf/<database>` lit de `stdin` une requête en `PSL` compilée, et recherche les matchs dans la `database`, dont il imprime les positions exactes sur `stdout`.
En filtrant cette sortie, il est possible de faire des recherches dans les séquences, et de sortir les annotations (et inversément), bien qu'elles soient dans des fichiers différents (`.seq` et `.ref`). Une option intéressante !

`fdf browse` lit de `stdin` une liste de positions, et imprime sur son `stdout` les entrées qui lui correspondent.

Annexe D

code source

Ce n'est pas que ce soit réellement intéressant... mais comme on dit dans le monde *opensource* : `use the force, read the source ! :)`

D.1 `fdf-search`

`http://www.isrec.isb-sib.ch/~jiglesia/cgi-bin/fdf/`

D.2 `fdf-search.config`

`http://www.isrec.isb-sib.ch/~jiglesia/cgi-bin/fdf/`

D.3 `fdf-search.macros`

`http://www.isrec.isb-sib.ch/~jiglesia/cgi-bin/fdf/`

D.4 `fdf-search.format`

`http://www.isrec.isb-sib.ch/~jiglesia/cgi-bin/fdf/`

D.5 `fdf-search.skel`

`http://www.isrec.isb-sib.ch/~jiglesia/cgi-bin/fdf/`

D.6 `RecallableWidgets.pm`

`http://www.isrec.isb-sib.ch/~jiglesia/cgi-bin/fdf/`

Bibliographie

[1] PSL Reference Manual, Paracel, Inc, 30 december 1992, Rev.A.
[2] FDF Manual, Paracel, date inconnue, Rev.0.