Introducing Lateral Thinking in Search Engines with Interactive Evolutionary Algorithms

Yann LANDRIN-SCHWEITZER* , Pierre COLLET, Evelyne LUTTON, Thierry PROST
Projet Fractales — INRIA Rocquencourt, B.P. 105, 78153 Le Chesnay Cedex, France
Yann.Landrin-schweitzer@inria.fr, Pierre.Collet@inria.fr, Evelyne.Lutton@inria.fr
`http://www-rocq.inria.fr/fractales`

ABSTRACT

Nowadays, large medical databases consist of a collection of smaller databases, each on possibly different fields and using different formats, making it increasingly difficult to retrieve valuable information among the thousands of documents retrieved by a simple query. A new Evolutionary Learning Interactive Search Engine (ELISE) feeds on previous user requests to retrieve "alternative" documents that may not be returned by more conventional search engines, in a way that may recall "lateral thinking." Tests on the "Cystic Fibrosis Database" benchmark [1] prove that, while suggesting original documents by adaptation of its internal rules to the context of the user, ELISE is able to improve its recall rate.

1. INTRODUCTION

Databases of leading pharmaceutical companies grow not only linearly but also in a discrete fashion each time another company is absorbed, along with its database [12].

In the end, huge databases of millions of documents and several tera-bytes of data are in fact constituted of several subdatabases, each on their own domain field with their own specific structures, languages, query systems, etc.

State of the art search engines spanning these sub-databases are used to retrieve documents with a surprisingly high recall rate and precision, considering the sheer mass of possibly multilingual data explored.

The success of such relatively deterministic engines is now so high that for each query, the user is confronted with too many documents containing words that do match the query !

To help the user find the data that might interest him among all the retrieved documents, the new frontier ([5, 6, 10]) is now:

- to try to present the user with fewer documents, albeit those he is interested in,
- to introduce notions of creativity in the search, that we will call hereafter *lateral thinking*.

After a short introduction to search engines and evolutionary algorithms, section 2 describes the developed ELISE search system. Tests and analysis are discussed in section 3, followed by a conclusion and a list of future developments.

1.1 Brief description of search engines and elaboration of a solution

Simple and basic search engines are of the "boolean" kind, meaning that they try to find documents containing only the words of the query. With such engines, looking for `text mining` will retrieve all documents containing the word `mining` as well as the word `text`, (possibly including documents on ore mining that also contain the word "text") but not documents on *data extraction* that do not contain the exact words of the query.

To take an example, in a database where there are 100 documents related to the general field of *data extraction*, a query on "text mining" may retrieve 400 documents. If only 40 of them are about *data extraction*, the *recall rate* of the tested engine will be of 40%, since the database contains 100 documents on *data extraction*.

As only 40 documents among 400 were matching the request of the user, the *precision rate* of the engine on this test will be of 10% only (see figure 1).

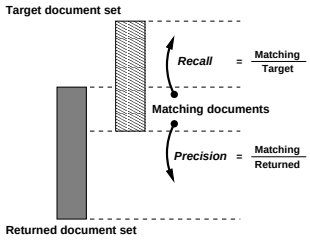

Figure 1: Recall rate and Precision: if the desired set of returned documents (i.e. the target) is known, the recall rate is the proportion of returned documents that match the target with respect to the total size of the target. The precision is the proportion of relevant documents in the document set returned by the system.

In order to maximise recall rates, more elaborated engines *expand* terms of the original query by using synonyms to broaden the search. Such engines will also find documents on `data extraction` provided that their thesauri contain

*This research is partly funded by Novartis-Pharma (IK@N/KE)

`data` as synonym for `text` and `extraction` as synonym for `mining`, but maybe not documents containing `knowledge extraction`, as `knowledge` is hardly a synonym of `text`. A lot of research is currently done on man-made thesauri to make sure that all documents that could be matching a query are actually found [4].

Other engines perform *stemming* on words of the query, meaning that words will be expanded into their derivatives, rather than their synonyms. A search for `organ` will have the engine search for `organs`, `organism`, but also `organiser`.

Of course, engines use at least both techniques plus a couple of others, therefore maximising recall rate and lowering precision. In front of the great number of retrieved documents, some ranking is attempted by the search engine, mostly depending on the number of occurrences of the words of the query in the document, but this is not satisfactory: users are not interested in documents containing words of the request, but in documents that match the meaning they attached to their query.

Unfortunately, many senses may be attached to a set of words, and the number of occurrences of words in a document does not help there.

However, properties can be found to direct the search towards document interesting a particular user whose domain of research and naming conventions, for instance, are quite personal.

Rather than simply storing the n previous queries and trying to find intersections between the result of query expansion and some history of previous requests, a solution could be to evolve a user profile thanks to genetic programming: an evolutionary algorithm could be used to evolve *modules* of a rewriting language, that would be used to reformulate user queries.

This proposed solution replaces the *query expansion* phase by a *query processing* phase, where evolved *modules* (i.e. modules that were found to give good results for the user) are applied to the query with two major results:

1. Rewritten queries will preferably retrieve documents that match fields of interest of the user.
2. Other documents related to previous and present queries will be retrieved, therefore bringing some "lateral thinking" abilities to the search engine.

Alternative documents presented to the user have a "lateral thinking" character because, rather than originating from word expansion out of blind thesauri, they might point out unsuspected relationships between the formulated request subject and other domains of interest of the user.

1.2 Evolutionary optimisation and application to knowledge extraction

Evolutionary algorithms are based on the Darwinian *survival of the fittest* theory. A set of *individuals* constitutes a population of potential solutions to a problem. An evolution process is then simulated the following way: At each generation, some individuals are created by mutation and recombination of selected "parent" individuals. This selection is designed in order to favour better individuals in terms of the problem at hand (via a so-called "evaluation" or "fitness" function). Such a stochastic process has been proved to converge theoretically, and many successful applications are based on this approach.

Schematically, recombination allows individuals to be generated using "genes" of their parents, therefore exploring the potential of the genetic pool of the population, while mutation allows "new genes" to appear, thus allowing to explore the search space in a stochastic fashion.

Evolutionary algorithms have been considered as attractive optimisation tools in the framework of information retrieval, document retrieval, web-mining and information filtering, and have been used several ways [9], but mainly in off-line implementation, for data pre- or post-processing [5].

For the present system, evolutionary algorithms are used interactively, in order to evolve a "user profile" at each new query. This profile is a set of "modules", that can perform basic rewriting tasks on words of the query.

The evaluation step is extremely simple: A list of documents corresponding to the processed query is presented to the user. The documents actually viewed by the users are considered as interesting, and the modules that constituted the query that retrieved the document are rewarded accordingly. Modules that rarely or never contribute to the retrieval of "interesting" documents are simply discarded and replaced by newly generated modules.

This algorithm allows to evolve a profile that maximises the "satisfaction" of the user in term of viewed documents, whatever this "satisfaction" means.

2. ELISE: AN EVOLUTIONARY LEARNING INTERACTIVE SEARCH ENGINE.

ELISE is based on an interactive Parisian EA, i.e. an EA whose whole population contributes to the solution (on the contrary to "classical" implementation, where a solution to the problem is encoded in one individual)[3]. More precisely, the population is made of parts of rewriting rules, called "modules." This population of modules encodes informations about the user that have been evolved so far by the system, i.e. the "user profile."

ELISE has been designed to be transparent to the user. User queries are rewritten with the help of the user profile, then the database is searched with the set of rewritten queries and presented to the user as a list of documents, in the same way as any usual search engine. Information about "user satisfaction" is collected as the number of documents actually read by the user. This information is internally used by the EA as a fitness function (see figure 2).

ELISE can be used as an improving mechanism on top of any kind of boolean search engine. The one used for experiments (see section 3) is SWISH++, [11], a basic public-domain boolean search engine, without any semantic expansion nor stemming tool.

2.1 Genome and structure

Experiences and earlier tests of the ELISE system have led us to an encoding scheme where genomes are small parts of complete rewriting rules: the *modules*, see [2]. The user profile is thus made of small rules which are aimed at performing the favourite semantic implicit transformations that are specific to a user (i.e. related to the usual "meaning" that the user gives to frequently used terms, like a subset of synomyms, a restriction to terms in connexion with usual interests, etc ...).

Modules are essentially trees, and since it will be needed to cut and merge program sections or change elements and still obtain valid codings, an RPL (Reverse Polish Lisp) structure —a particular case of stack-based languages— was cho-

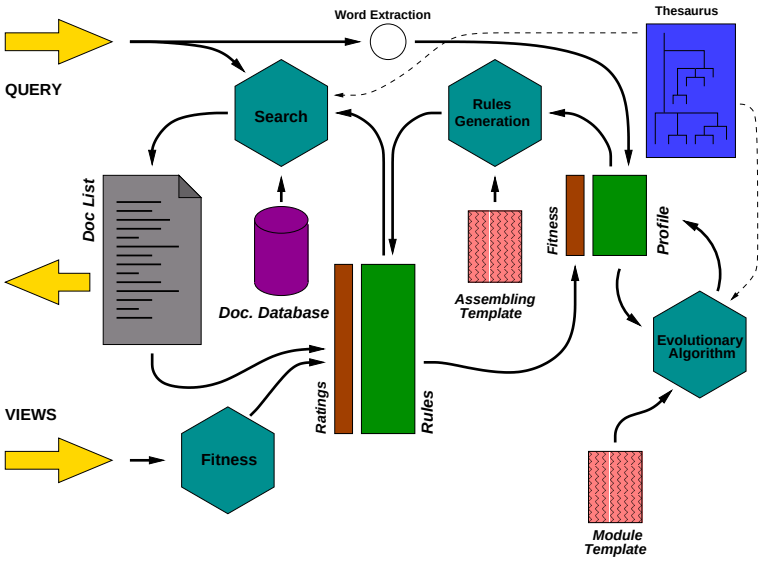

Figure 2: ELISE evolves a set of rewriting modules, the "user profile." Queries are rewritten by rules derived from this profile and returned documents are displayed to the user. Fitness functions of modules are then updated and activate a new generation of the EA.

sen. RPL encoding offers both the processing capabilities and a lax enough structure to suit the need of ELISE. It is also close enough to classical tree representations that it is possible to reuse an important part of the theoretical and practical tools developed throughout the history of genetic programming.

The implementation of this part of ELISE is based on a set of generic tools [8], where programs are lists, containing either atoms (i.e. instructions and character strings) or sublists. However, depending on the set of instructions, some programs can not be guaranteed to run. It has been thus necessary to deal with invalid individuals in the evaluation and genetic operators (which is not a major problem for EA techniques).

2.1.1 *Instruction sets of the RPL encoding*

While the basic stack operations are the same in most applications (duplication or deletion of an element, list operations...), a set of instructions is needed to operate on the particular data that is dealt with. In ELISE, this mainly consists in semantic operations (sense or concept lookups in a semantic network, returning sets of terms) and tree operations (since queries have a tree structure).

Semantic instructions actually reflect the different flavours of term expansion that can be based on a semantic network [4]. Tree operations are: conditionals, splitting (break a tree into subtrees), or grouping (join two subtrees) instructions. Mapping instructions are also available, that will apply a list (i.e. program) on every node or every leaf of a query.

2.1.2 *Derived rewriting rules*

In order to be sure to evaluate independently each module of the population, modules are directly copied in the rewriting rule set. Then, a set of templates is used to build other complex rewriting rules from the modules of the current population. These templates perform various combinations of two or three modules via the set of boolean operators available in the underlying search engine (for example "AND," "OR," "NO" and maybe "NEAR" or "LIKE"). The idea is to build randomly plausible combinations of evolved modules, with no *a priori* preference about the way they are combined. Modules are aimed to containing user-specific preferences (the "memory"), while templates represent more casual combinations of terms (the "innovation").

Example: [@ @ «#AND»]
[@ «VOCSYN»]

@ stands for a selected module. The first template groups two modules whith the "AND" operator, while the second one commands a synonym expansion of a module.

2.2 Genetic operators

The paradigm of genetic algorithms was originally developed to deal with bit-string individuals, on which the definition of genetic operators —mutation and crossover— is straightforward. However, when working on more complex genome structures, an important part of the EA performance is linked to the accurate tailoring of these operators.

Genetic operators must be designed in order to produce valid structures most of the time. This means essentially taking into account instruction arity and input type when doing mutations or crossover, "repairing" discrepancies when needed by introducing extra arguments or ignoring some, as required to keep the overall input and output signature of mutated or recombined instructions.

Therefore, three types of mutation operators are used, with distinct probabilities:

- local, intra-class mutation (replaces a character string by another term connected to it in the semantic network that is used, or an instruction by another with the same arity),
- local, inter-class mutation (changes an instruction into another, with a different prototype, repairing the resulting genome, or turns an atom into another atom

of a different type, for example, an instruction into a character string),

- global mutation (modifies the structure itself of the genome, by replacing an atom by a list, or the opposite).

All mutations try to repair modules as much as possible.

Following the same idea, two types of crossover are used: a local one that will leave sublists unchanged, while the global one will apply on sublists when possible. These crossovers rely on the same mechanism: cut points are positioned randomly along the two parents lists and list sections are exchanged in alternance. One of the modified lists is then taken as result of the crossover.

In order to keep the proportion of invalid individuals low, a *migration* operator is finally used, that repairs the most obvious incompatibilities in the genome.

Example: Two typical modules are:

```
[ «VOCGEN» "metabolic disorder" «VOCSYN» «#AND» ]
[ [ «SPLIT» "acetic acid" «VOCGEN» «#AND» «#OR» ] «IFOR» ]
```

The first one generalises the query terms and adds to the original query the additional condition to recover only documents containing synonyms of "metabolic disorder." The second one restricts the set of documents retrieved by the second part of an "or" query to those having to do with generalisations of "acetic acid."

A mutation of the first module is for instance:

```
[ «VOCGEN» "paramyxoviridae infections" «VOCSYN» «#AND» ]
```

This particular result was obtained with a semantic operation on the "metabolic disorder" atom.

Another mutation, on the second module, looks like:

```
[ [ «SPLIT» "acetic acid" «VOCGEN» «#OR» «#OR» ] «IFAND» ]
```

This is a result of replacing instructions inside their class.

A crossover between both mutated modules reads:

```
[ [ «SPLIT» "acetic acid" «VOCGEN» «#OR» «#OR» ]
  «IFAND» «VOCSYN» «#AND» ]
```

2.3 Fitness functions and user interaction

The fitness function must embed a "user satisfaction" measurement and is collected from the interaction with the user. The only information available being the number of documents viewed by the user from the set of returned documents by the system (i.e. a "recall" rate). The efficiency of each module of the population has to be derived from this "global" evaluation (number of viewed documents).

Two counters $C_{retrieved}$ and C_{count} are associated to each module of the population. For each query produced by ELISE (translations of the original query), the associated set of documents can be evaluated with respect to "user satisfaction," (i.e. if this document is viewed or not by the user) these quantities are translated into a bonus (in the results presented in the next section, the bonus is $Recall\_rate + \alpha * Precision$, as the recall rate and precision can automatically be evaluated from the CFD benchmark —see section 3).

As the algorithm keeps track of which module is used in which rule and of which rule produced which translated query, the computed bonus is directly added to the $C_{retrieved}$ counter of the modules that participated in the production of interesting documents, while their corresponding C_{count} counter is incremented by 1.

The fitness of each module is therefore $C_{retrieved}/C_{count}$, measuring the mean efficiency of the module when it is used.

2.4 Initialisation and parameters

In order to provide reasonable answers even with an initial non-evolved user-profile at the beginning, an initialisation template is used to create a minimal proportion of sensible modules (30%) and the rest of the initial population is made of randomly generated modules. Of course, an "identity" rewriting rule is always present, in order to be sure to always keep the original query.

As the fitness function is very noisy (a correct evaluation of a rewriting rule must be made on a minimal set of queries), a "slow" evolution process has been preferred. A significant part of the population (the 70% top ranked modules, a sort of memory of the best ones) is transmitted unchanged to the next generation. Mutation and crossover rates used for the generation of the rest of the population are respectively set at 30% (per gene) and 80% (per individual). Typical module population size is 50 and 50 complex rules are derived at each generation.

3. EXPERIMENTS AND ANALYSIS

Tests presented below have been performed automatically on the Cystic Fibrosis Database (CFD)[1], using a basic boolean search engine Swish++ [11]. The semantic operators are based on a medical oriented thesaurus: MeSH [7].

Of course automatically testing such an interactive system is extremely difficult (especially with respect to the "satisfaction of the user" quantity), however an automatic test remains interesting in the sense that it allows to gather some statistics rather easily. The CFD test consists of 1239 documents published from 1974 to 1979 discussing a specific topic (Cystic Fibrosis Aspects), and a set of 100 queries with the respective relevant documents as answers. For each query, the global recall rate and precision of the system can thus be precisely measured in comparing the returned document set to the relevant registered answers (informations about the innovation or about the rating of a particular document are reduced to the strict minimum, contrarily to interactive tests).

Figure 3 presents three curves: the cumulated mean recall rate of the basic search engine (less than 10%), the cumulated mean recall rate of ELISE (around 45% at 18 queries only, and constantly growing until the end of the test of 140 queries, i.e. 140 generations of the EA). A learning behaviour can be seen on this curve, resulting on a definite increase of the recall rate. This behaviour is even more obvious on the statistics of the recall rates of queries for which the underlying system returns no document: ELISE is able to obtain reasonable recall rates even in this extremely unfavourable case.

Figure 4 presents the same set of cumulated mean curves but for the precision measurement. ELISE provides a much lower precision than the basic seach engine, which is correct, due to the diversity policy that has been adopted (each module of the user-profile needs to be evaluated at least once: if it returns unrelated answers, this lowers the global precision of ELISE, but knowing it is a "bad" module is a crucial information to evolve the whole system).

Some of the best modules of the final population (after 140 generations) are detailled and explained below.

```
[ «_DATA_» "artery, ciliary" «#OR» ]
```

Returns documents obtained from the original query, plus documents about "ciliary arteries." This is a generalisation (extension of the scope) of the original query.

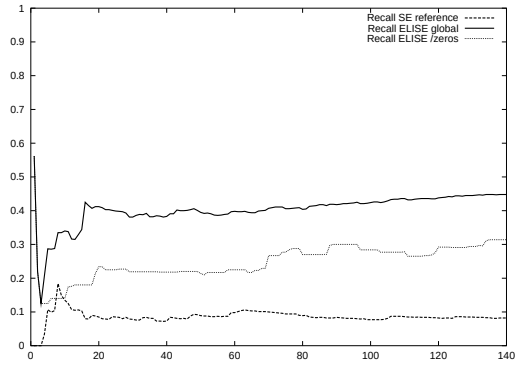

Figure 3: Cumulated mean recall rates of the underlying boolean search engine (dashes), of ELISE (solid line), and of ELISE for queries for which the underlying search engine returns no answer (dots).

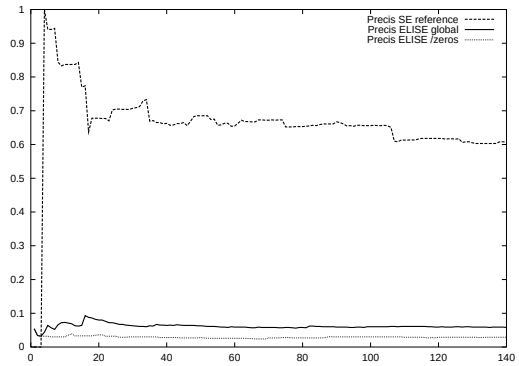

Figure 4: Cumulated mean precisions of the underlying boolean search engine (dashes), of ELISE (solid line), and of ELISE for queries for which the underlying search engine returns no answer (dots).

```
[ «#NOT» "ananase" «SPLIT» ]
```

Performs a search for documents about "ananase," while ignoring the original query (!)

```
[ «_DATA_» "arteries, ciliary" «#AND» ]
```

Among documents obtained from the original query, this module returns only those about "ciliary arteries." This is a restriction of the original query.

```
[ «_DATA_» "artery, ciliary" «VOCGEN» ]
```

Performs a search for documents having to do with "ciliary arteries" or generalisations of this concept, in the set of documents returned by the original query.

```
[ «_DATA_» "dayto anase" «#OR» ]
```

Performs a search for documents about "dayto anase," and appends them to those returned by the original query.

```
[ «SPLIT» "bromelain" «#VOID» ]
```

Uses the implicit operator of boolean search engines (that is, generally a loose "and") to restrict the scope of the right part of the original query (i.e. at the right of the root boolean operator).

Evolved terms used in these modules seem to be in connexion to the CFD topic, while not being mentioned in the 140 queries that were submitted to ELISE.

The evolved modules, along with the submitted queries, have been presented to a medical practicioner, whose analysis is the following:

The terms present in the modules of the ELISE system after 140 generations contain in particular a whole set of enzymes (ananase, dayto anase, traumanase, bromelains, etc) used in various pathologies as anti-inflammatory agents. This set draws the attention to:

1. *the use of enzymes in the cystic fibrosis-like pancreatic substitute, to compensate for the malabsorption due to the attack of this organ by the disease,*
2. *and to the therapeutic role played in the fluxing of bronchial secretions —even if in this case, they are of other enzymatic kinds.*

The absence of recent work indexed in Medline on these particular enzymes (no relevant references found after 1997) is probably due to the old age of the base having been used for the tests.

If the relationship with the term "ciliary arteries" and its derivatives is less obvious and straightforward (hypothesis: ocular arterial disease is frequent in diabetes, itself frequent in cystic fibrosis) this type of approach has the merit to give hints and tracks of reflexion to the practitioner or researcher and to draw attention to ignored aspects of the studied pathology.

Because of the sheer mass of indexed data, this step is indeed quite original and very much complementary to the traditional output of search engines which aim more at drawing up inventories of fixtures of current knowledge or recent work, rather than to open new fields of research.

Other tests have been performed on some subsets of questions of the CFD, in order to evaluate more precisely the character strings that are evolved in the user profile after a given set of questions. The questions of the CFD have been classified in several sub-classes, presented in table 1.

Two types of tests have been performed, see table 2:

- restriction of the set of questions presented to ELISE to a specific sub-class (during 150 generations, the learning set is also the test set),
- first evolve the profile during 100 generations on the whole set of questions of the CFD, except a given sub-class of questions (learning set), and then specialise the profile on this sub-class during 50 generations (test set).

Evolved terms have been sorted in 4 relevance categories (see table 2).

The analysis of the medical practicioner is the following:

Some keywords returned by this second series of tests stress nontraditional or indirect aspects of the treatment of cystic fibrosis which would not necessarily have been studied in a usual approach. Indeed, one finds relevant references relating to subjects as various as catheterization, education, tobacco exposure, or genetically modified organisms; it should be noted that these "unusual" items are in a majority compared to references, in the same time relevant and "difficult to circumvent," such as infections (for example, pseudomonas aeruginosa *and* human adenoviruses *are terms returned by the search engine).*

In addition, if the items are classified according to whether they interest more specifically clinicians or fundamental biology researchers, one finds approximately a third of terms of the field of fundamental biology into the category "undirectly related to the test set" and a half of them into the category "unrelated terms". It will therefore be interesting to test the search engine in real world conditions in order to see whether it clearly separates these two types of users, which may result in a reduction of the proportion of nonrelevant terms.

Set n°	Classification 1: CF afflictions and consequences by organ or function	Set n°	Classification 2: specialities
10	respiratory system	10	diagnosis of CF or of CF-related pathologies
11	epithelia or secretions in the respiratory tract	20	epidemiology and genetics of CF
12	infectious pathologies and immunologic mecanisms in the lung	30	clinical features of CF
13	treatment of respiratory system damage	40	biochemical and physiologic manifestations of CF
14	other respiratory afflictions	41	in the respiratory system
20	gastrointestinal and hepatic consequences of CF	42	other
21	pancreatic pathologies	50	CF treatments
22	other gastrointestinal and hepatic pathologies	51	treatment of pulmonary manifestations
30	physiologic and biochemical consequences of CF in other organs	52	treatment of digestive consequences
40	epidemiology and genetics of CF	53	other treatments
50	diagnosis of CF	60	other pathologies and manifestations

Table 1: Query sets classification key: subsets of the CFD, and sub-subsets

Test run number	Run 1	Run 2	Run 3	Run 4	Run 5
Learning set[1]	1-13	1-12	1-!12	2-50	2-!50
Number of terms in learning set	14	14	85	21	81
Test set[1]	1-13	1-12	1-12	2-50	2-50
Number of terms in test set	14	13	13	21	21
Recall rate	38.1	38.4	42.3	33.3	35.4
Precision	9.8	10.4	12.2	12.0	9.1
% of terms directly related to the test set	19.2	28.6	26.6	20.0	28.6
% of terms undirectly related to the test set	19.2	57.1	40.0	10.0	42.8
% of terms related to the CF but not the test set	11.6	0	20.0	30.0	5.7
% of unrelated terms	50.0	14.3	13.3	40.0	22.8

Note: The "N-S" notation means "Classification N, subset S". "N-!S" means "Classification N, all the CF database *except* subset S", see table 1.

Table 2: Runs performance, and relevance classification of the vocabulary extracted from final modules

4. CONCLUSION AND FUTURE WORKS

Results seem to show that after a learning period of 20 to 30 generations (i.e. 20 to 30 queries, which is a reasonable value, even for a human user), the recall rate of ELISE rises to around 40% while the recall rate of the purely boolean engine remains quite low. This shows that some learning takes place that allows to retrieve documents beyond the scope of the boolean engine.

An *a posteriori* analysis of the evolutionary engine shows that after the last query, the contained modules have evolved accordingly to subjects explored previously. More interestingly, many terms found in the modules are not present in the previous queries given to the engine, meaning that they have been extracted from previous synonym expansions and kept because of their interesting value to the simulated user.

However, some terms that have been evaluated as interesting for the user by the evolutionary algorithm are not directly related to Cystic Fibrosis. Specialists of the domain have nevertheless found some indirect correlation of the terms to Cystic Fibrosis, suggesting that retrieval of documents containing those terms might be of some value to the user for subsequent queries.

The user may therefore see odd documents appearing in the result list, that may (or may not) be of use to the user for his future queries, as if the ELISE search engine were showing some "lateral thinking" capabilities.

Further work include finding a way to rank the presented results, that is not based on word count. A possible solution might be to interleave the best results of each derived query.

The ELISE system will soon be tested in the real world on ULIX (the Novartis-Pharma state of the art search engine). Modifications will probably be necessary as tests will be conducted by real users rather than by an automatic procedure.

Acknowledgement

The authors are grateful to Gilles ROGER M.D. PhD. for his help on relevance classification of many runs, as Cystic Fibrosis specialist.

5. REFERENCES

[1] Cystic Fibrosis Reference Collection, `http://www.sims.berkeley.edu/ hearst/irbook/cfc.html`

[2] P. Collet, Y. Landrin-Schweitzer and E. Lutton, "Interactive Parisian GP for an Evolutionary Search Engine", submitted.

[3] P. Collet, E. Lutton, F. Raynal, M. Schoenauer, "Polar IFS + Parisian Genetic Programming = Efficient IFS Inverse Problem Solving," In *Genetic Programming and Evolvable Machines Journal*, Vol. 1, Issue 4, pp. 339-361, 2000.

[4] D. J. Foskett, "Thesaurus," *Readings in Information Retrieval*, K. S. Jones, P. Willet, M. Kaufmann Publishers, San Fransisco, 1997.

[5] A. A. Freitas, "Data Mining with Evolutionary Algorithms: Research Directions," AAAI Worshop, Technical Report WS-99-06, ISBN 1-57735-090-1, The AAAI Press, 1999.

[6] Y.-H. Kim, S. Kim, J.-H. Eom and B.-T. Zhang, "SCAI Experiments on TREC-9," Proceedings of the *Ninth Text REtrieval Conference* (TREC-9), pp. 392-399, 2000.

[7] "MeSH (Medical Subject Headings), a controlled vocabulary thesaurus" `http://www.nlm.nih.gov/pubs/factsheets/mesh.html` National Library of Medicine (NLM), National Institutes of Health, Bethesda, Maryland.

[8] Y. Landrin-Schweitzer, OKit, a Virtual Machine and Compiler for Concatenative Languages, `http://varkhan.free.fr/Software/OKit`

[9] S. Pal, V. Talwar and P. Mitra, "Web Mining in Soft Computing Framework: Relevance, State of the Art and Future Directions," IEEE Trans. on Neural Networks, 2002.

[10] D.-H. Shin , Y.-H. Kim, S. Kim, J.-H. Eom, H.-J. Shin and B.-T. Zhang, "SCAI TREC-8 Experiments," Proc. of the TREC-8, pp. 511-518, 1999.

[11] SWISH++, Simple Web Indexing System for Humans: C++ version, `http://homepage.mac.com/pauljlucas/software/swish/`

[12] T. Vachon, N. Grandjean, P. Parisot, "Interactive Exploration of Patent Data for Competitive Intelligence: Applications in Ulix (Novartis Knowledge Miner)," *Int. Chemical Information Conf. and Exhibition*, Nîmes, France, 21-24 October 2001.