Introducing Lateral Thinking in Search Engines

Yann LANDRIN-SCHWEITZER[1], Pierre COLLET, Evelyne LUTTON

COMPLEX Team - INRIA Rocquencourt, B.P. 105, 78153 Le Chesnay cedex, France
Yann.Landrin-schweitzer@inria.fr, Evelyne.Lutton@inria.fr
Tel : +33 (0)1 39 63 55 23 — Fax : +33 (0)1 39 63 59 95
http://www-rocq.inria.fr/fractales

Laboratoire d'Informatique du Littoral, ULCO BP719, 62100 Calais cedex, France
Pierre.Collet@univ-littoral.fr
Tel : +33 (0)3 21 46 57 53 — Fax : +33 (0)3 21 46 57 51
http://lil.univ-littoral.fr

Abstract. *Too much information kills information.* This common statement applies to huge databases, where state of the art search engines may retrieve hundreds of very similar documents for a precise query.
In fact, this is becoming so problematic that Novartis Pharma, one of the leaders of the pharmaceutical industry, has come up with the somewhat odd request to *decrease* the precision of their search engine, in order to keep some diversity in the retrieved documents.
Rather than decreasing precision by introducing random noise, this paper describes ELISE, an Evolutionary Learning Interactive Search Engine that interactively evolves rewriting modules and rules (some kind of elaborated user profile) along a Parisian Approach[12].
Additional documents are therefore retrieved that are related both to the domains of interest of the user and to the original query, with results that suggest of lateral thinking capabilities.

1 Introduction

The ever-growing size of document collections available over the internet or within intranets of large companies makes any complete human indexation impossible. Retrieving information from such a mass of documents can almost exclusively be done by automated search systems.

These systems need to browse over a "document landscape" in order to find relevant information. As such, they need to be provided with a goal (what must be found) and a basic topology of the search space (how to go get it).

The analysis of the goal definition (the query) must be accurate and precise in order to produce usable results. To simplify interpretation, query formulation often suppresses

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

grammatical structure, replacing it with boolean boolean operators: "*this* notion AND NOT *that* one". This improves accuracy, at the cost of limited expressivity.

Unfortunately, due to user and language specificities, interpretation is not unambiguous. The same words can be associated with distinct meanings in some jargons[2].

However, state of the art search engines make wonders and manage to find documents matching the request with impressive precision. Unfortunately, information in documents sets has a generally high level of redundancy, meaning that many similar papers are retrieved.

ELISE tries to circumvent this problem, by evolving information treatment strategies specific to a sub-task of the search, on a per-user basis. These strategies are the result of an on-line evolutionary process, using input from the user to design query-rewriting routines adapted to a topic or a document subset.

This paper presents the specifications of the problem that motivated this research (section 3), the factors constraining its solutions (section 4), the explored solutions (section 5) and the description of their technical implementation (section 7). An analysis of the results obtained on a public domain benchmark (the CFD test set) gives the basis of a reflexion on the form several important aspects of GP take in this experiment (section 8). This leads to a conclusion on the unusual and new aspects of this application, paving the way to future developments (section 9).

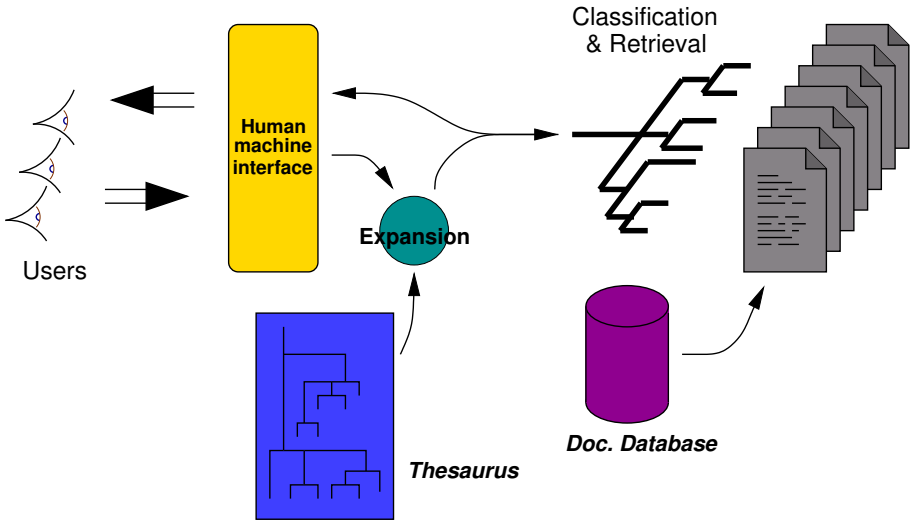

Fig. 1. Functional interaction in modern search systems: query expansion is based on a thesaurus.

[2] In this context, **jargon** means a semantic subset shared by a group of users. Note that some words may only exist in specific jargons, conveying distinct meanings, and not in any "mainstream" language.

2 Conventional search systems

Boolean search engines simply retrieve texts that correspond exactly to the query. Looking for "text AND mining" will exclusively retrieve documents that contain both words. Such a query may therefore come up with texts on *ore mining* as well as texts on *data mining*, providing that the word "text" appears in the document.

However, texts related to "text mining" that do not contain both words will not be selected, even though the user might find interest in them. In order to minimise the adverse effects of this way of exploiting queries, it is possible to perform what is called a *query expansion*: instead of using all words of the query at their "face value," advanced search engines replace them by a list of candidate synonyms, in order to collect all the possible senses of each word and broaden the search. Synonyms are obtained from a *thesaurus* [17]: a human-compiled dictionary of synonyms. The next step is a very classical document retrieval process, using word indexes or database requests, followed by the ordering and display of retrieved documents. This results in a three-stages architecture, on the model of figure 1.

This method generally improves results over the direct processing of a user-formulated boolean query, even though it does not even attempt to associate some sense to a set of words.

With such a strategy, it is very probable that all concerned documents will be found, but at the cost of "diluting" relevant references in hundreds of pieces of irrelevant information. In technical terms, the *recall rate* is high, at the expense of *precision*, as shown in figure 2.

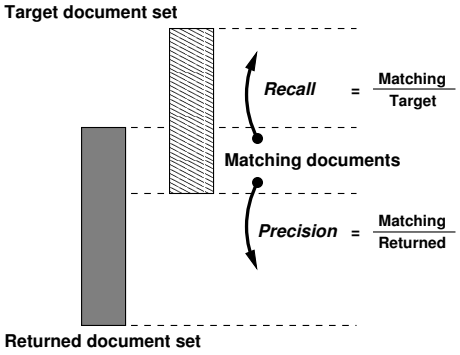

Fig. 2. Recall rate and Precision : if the desired set of returned documents, i.e. the target, is known, the recall rate is the proportion of returned documents that match the target with respect to the total size of the target. The precision is the proportion of relevant documents in the document set returned by the system.

Selecting a particular sense for each word of the query would return more specific documents. This can be done for instance by selecting a particular set of synonyms to be used for this word. However, there is no automatic way of doing this. Some attempts

at interactive query refining, using this method (see [55]), have proved their efficiency, but remain of rather tedious use.

2.1 Architecture and capabilities of modern search engines

Most modern search engines feature advanced capabilities in semantic expansion. This is a highly desirable feature in any search engine, and one that is well mastered by the users. But, to be of any use, it requires adapted thesauri, produced over several years by teams of experts and needing constant updates.

These search engines often double as document database systems, capable of unifying different databases, building and updating extensive document indexes quickly and efficiently, and tracking document sources for updates. These characteristics are well accepted and supported by the administrators of knowledge management solutions.

Most of them, through the use of heuristic mecanisms honed down to macth the specificities of the document sets, reach wery high levels of measurable performance. Mainstream systems will reach a precision of 90%, though recall rates are much more variable, depending on the data: between 20% and 80%, the later on well-categorized and formatted document collections. Tailored search system will top these performances on a restricted set of documents, as seen in some of the TREC evaluations ([52] for instance), while testing for robust retrieval sees this rates plumetting, with around 80% recall rate and 10% precision, as seen in [53].

Overall, high retrieval capabilities are an integral part of modern text-retrieval systems, but do not endow state of the art search engines with imagination and adaptability.

Indeed, implementing and evaluating techniques that target document "value" for the user is, at best, tricky. Documents that satisfies most of the systematic relevancy criterions used in mainstream search engines, by being strongly correlated with search terms, will also most probably contain little new information. Conversely, documents that present radically new concept associations will not be identified as relevant, since no convergence will be detected in their semantic components. Those search systems are thus very good at presenting the user with what he/she already knows.

3 Aim of this research

Rather than asking the user to provide a precise sense definition for all words, the principle behind ELISE is to develop some kind of user profiling to specialise the engine for each user. This lets the system expand the set of retrieved documents to texts that are of interest to each particular user. Moreover, this makes available filtering criteria that can not be defined by a textual formulation, such as document structure, source or density. Some of these criteria were previously made available by search engine through specialized operators, making for more complex query syntaxes.

ELISE aims at providing these refinements over a simple boolean text query in an effortless and adaptative way, by analysing users' behaviour along the whole query process.

This research, albeit exploratory, takes place in an industrial context, where technical imperatives (practicality and efficiency on a large scale, as well as distributed access and the need for server-side processing) need to be kept in mind:

Description of the databases The described search system is intended to operate on a set of medical databases that contains several millions of documents for a volume of several tera-bytes [55].

Different domains are covered, meaning that the search engine cannot afford to be too specialised.

Typical user Users are typically scientific engineers, researchers or practitioners in many different specialised fields such as chemistry, biology, genetics, etc, and managers or financial advisors deciding policies in these areas.

Testing and evaluation This algorithm is not designed to replace, but to be grafted onto state of the art conventional search engines, meaning that in terms of recall rates, the presented system will be at least as good as the best available search engines.

The original goal of this research is therefore not to increase recall rate or precision, but to provide user-dependent text-retrieval capabilities that would promote "creativity," in the sense that the enhanced system should be capable of retrieving documents that may only be remotely related to the query, *but that are yet of interest to the particular user that produced the query*.

Such documents may or may not be retrieved by classical engines, as they do not really correspond to the specifications of conventional search engines.

The goal of this research is therefore somehow orthogonal to what usual engines have been trying to do up to now (maximising recall rates and precision). It therefore makes sense to consider the presented system as an addition to conventional search engines, as the added information provided by the system is not of the same type as the information provided by those engines.

Such a goal amounts to trying to maximise *user satisfaction* upon usage of the "enhanced" search engine. This quantity is obviously user-dependent, and no benchmark can exist to rate it. ELISE can only fulfill its goal if it is versatile enough to be able to maximise nearly any requirements of the user.

Unfortunately, standard text-mining benchmarks, such as the CFD [7] or the TREC [51] are only capable of telling which documents among the database contained in the benchmark should be retrieved by a specific query, therefore only allowing to compute the recall rate and precision of the query result.

Automatic tests performed with these benchmarks can therefore only provide information about the capacity of ELISE to adapt to its context, mesured by its recall rate, compared to a purely boolean search engine, as detailed in section 8.

4 Designing an adaptive user-dependent query processing system

Such a system must integrate with document database systems, interact with users over a client-server model, make use of remote, asynchronous resources (databases, thesauri), ... All of these constraints tend to promote a modular, data-flow-oriented architecture. According to this, the algorithm will be described in terms of information flow and processing steps.

4.1 Tailoring search engines to users specificities

The first element of information being available in a transaction (i.e. all the steps from a user's query until the reply) is the query itself, that can therefore be considered as the starting point of the processing chain.

A simple idea to achieve user-dependent processing while retaining capabilities of underlying search engines, consists simply in transforming the original query into another, maybe more complex, and pass directly this processed query to the search engine.

However, producing user-tailored transformations, without any prior information is impossible. A user profile must be elaborated from the user's behaviour. This implies being able to observe the reactions of the user to the results that are presented to him after a query.

Such intelligence could be gathered by asking the user to rate the pertinence of retrieved documents on a predetermined scale. A more elegant solution consists in spying the behaviour of the user: the main purpose of this algorithm being to retrieve documents, determining which ones are actually viewed by the user can be considered as a reasonably reliable indication of the user's interest in those proposed. A rating of the pertinence of the proposed documents can therefore be determined transparently, just by keeping track of which documents were selected by the user.

This leads to a 3 steps information processing loop:

1. Using a user-profile to process an incoming query.
2. Performing a search in the document database with the new query and presenting the result list to the user.
3. Integrating documents consultations from this list to update the user profile.

In practice, the second step is handled by the underlying search engine (boolean search engine for tests, state of the art search engine for the real-world system).

4.2 Methods to build user profiles

The simple method proposed above can be extended by replacing the query expansion phase of classical search engines by a *query processing* phase, with processing rules learnt from user input.

The resulting structure is essentially close to that of conventional search systems, therefore allowing to reuse most of the existing software tools and database formats. In addition, deciding which sense to give to a particular word in the query can be done on a per-user basis, provided that the system keeps track of the idendity of the user. This leads to figure 3.

5 Evolving processing rules with Interactive Parisian EAs

Methods allowing a system to learn how to process a given query into another one, without any kind of *a-priori* information, are very few. Evolutionary Algorithms (among which Genetic Programming) are good candidates, as they only need a fitness function that can simply be provided by user satisfaction.

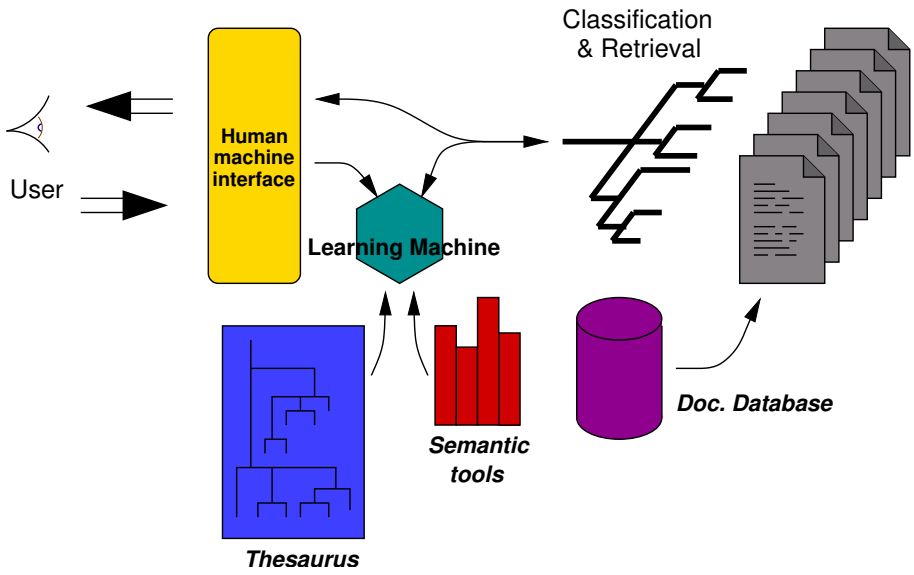

Fig. 3. Functional interaction in a learning search system: the machine learning stage performs an adaptation of the system to a specific user.

An Evolutionary Algorithm can be used to evolve a profile, made of a set of transformation rules. These would be used to process queries, extracting requests to be passed to a classical search engine from the original user query.

More precisely, the proposed solution is based on an "Interactive Parisian Evolutionary algorithm" paradigm [12], which would acts as a user-specific adaptive learning component on top of a classical query processing system.

5.1 Evolutionary Algorithms for document retrieval

Information retrieval, document retrieval, web-mining and information filtering are closely related research fields. In these fields, EAs have been considered as attractive optimisation tools: GP have been used for information filtering [64], and EAs have been used in several ways [28, 37]:

- Search and retrieval, [28] for web-document retrieval (to learn the weights of HTML tags).
- Query optimisation, queries modification [62],
- Document representation, document indexing [22, 58, 57], ranking [28], tuning of the weights of retrieved keywords [24], dependence modelling.
- Classification, on which a lot of research has been conducted, [6] and feature selection for document classification [54, 63] (to find optimal feature subsets).

EAs are preferably used off-line, due to their generally high computational cost. They are mainly used as tools for data pre- or post-processing [19, 20]. Only few ap-

proaches are based on interactive implementations, like [15] which examines the nature of interactive querying and retrieval and uses an EA in Query-by-Browsing.

Personalising On-Line Information Retrieval systems or interfaces has been also considered as an important feature. For example [59] considers the problem of extracting pertinent informations: users have difficulties in specifying their interest, which therefore changes in the course of time, leading to subjective ratings representing their state of mind. The solution proposed is based on an EA with a monitoring module, that records all activities of the user. The behaviour of the user is modelled by way of a classifier system so that future actions can be predicted on the basis of past experience.

ELISE uses a more "pragmatic" approach, using the cost-efficient Parisian Approach [2] and interactive EAs to specialise existing search engines with evolved user profiles. With one generation per query, time spent in the evolutionary algorithm is not apparent for the human user. High efficiency is obtained through the Parisian paradigm, where the whole population, made of short processing rules, is the solution, i.e the user profile.

5.2 The Parisian EA paradigm

Interactive Evolution is now an attractive research topic, with many applications in various domains (an overview of this vast topic can be found in [47]). Actually, interaction with humans raise several problems, mainly linked to the "user bottleneck" [39], i.e. the human fatigue. Solutions have to be found in order to avoid systematic and boring interactions [39, 47, 1], for example by reducing the size of the population and the number of generations, by choosing specific models to constrain the research in *a-priori* "interesting" areas of the search space, or performing automatic learning (based on a limited number of characteristic quantities) in order to assist the user and only present to him the most interesting individuals of the population, with respect to previous votes of the user.

The Parisian approach is another way to tackle this "user bottleneck" as it has been designed to reduce computational costs while maintaining exploration capabilities and genetic diversity. This approach has been designed relatively recently [12] and is actually a generalisation of the Michigan classifier systems approaches [21]. It is based on the capability of an EA not only to push its best individual towards the global optimum, but also to drive its whole population in attractive areas of the search space. The idea is then to design a fitness landscape where the solution to the problem is given by the whole population or at least by a set of individuals and not anymore by a single individual. Individuals do not encode a complete solution but a part of a solution, the solution to the problem being then built from several "collaborating" individuals.

A Parisian population is a "society" which builds in common the solution that is searched for. Of course, design of such algorithms becomes more complex than for a direct conventional EA approach and the diversity of the population becomes a crucial factor. Moreover, splitting the problem into interconnected subproblems is not always possible.

However, when it is possible to do so, the benefit is great: a Parisian approach limits the computational waste that occurs in classical EA implementations, when at the end of the evolution, the whole final population is dumped except the best individual only.

Experiences and theoretical developments have proved that the EA gains more information about its environment than the only knowkledge of the position of the global optimum. The Parisian approach tries to use this important feature of EAs.

A Parisian EA may have all the usual components of an EA, plus the following additional ones:

- *two* fitness functions : a "global" one that is calculated on the whole population or on a major portion of it (after a clusterisation process or elimination of the very bad individuals, for example) and a "local" one for each individual, that measures how much this individual contributes to the global solution.
- a distribution process at each generation that shares the global fitness (which represents the output of the system or the so-called interaction with the environment) on the individuals that contributed to the solution (that is a part or even the whole "local" fitness function),
- a diversity control scheme, in order to avoid trivial solutions where all individuals are concentrated on the same area of the search space.

Developing a Parisian EA for interactive search tools is based on the fact that search cannot be reduced only to an optimisation process: often users do not have precisely in mind what they search for. The aim may fluctuate, users sometimes gradually build their queries from an exploration. Innovation (diversity) is important, "user satisfaction" is a very peculiar quantity, very difficult to measure and moreover to embed in a fitness function.

6 Sense extraction: different tools for distinct contexts

Semantic analysis is essentially a tool to take educated decisions on sense choices. Therefore, the way semantic tools are used in this algorithm depends primarily on the type and amount of information that can be extracted from the context.

From now on, the term *sense* will refer to the consensual signification of a word and *meaning* to the signification intended by the user. A set of synonyms can be associated to each sense (from now on abbreviated as *synset*) that can be used equivalently to convey the sense in a text. Obviously, a synset can contain several words and conversely a word can bear several senses.

Moreover, a single word can point to several *lexical forms*, as for instance **leaves**, that can either be the plural of **leaf** *(n.)*, or the 3rd person of **leave** *(vb.)*.

To simplify things, lexical forms will be considered as bearing senses. Dedicated dictionaries of synonyms (*thesauri*) can provide, for each lexical form, a list of related synsets.

6.1 On the use of fuzzy thesauri in query processing

A query is meant to convey, in a very small number of words (typically 2 to 10), the description of a search topic. This puts the emphasis on individual senses and their relations. A thorough analysis of queries is therefore needed, implying a detailed sense

extraction (so that statistically rare senses are not overlooked) to pick out the interpretation most compatible with the search history.

The second step in query analysis would be to determine the relations between query terms and to group them accordingly into topics. A local analysis of the query is needed to obtain such information: the limited size of human short term memory makes it unlikely to find a long distance between connected terms.

A basic approach to merge these two steps is to use synset correlation between close terms. This very simple method works surprisingly well on classical queries. Generally, a strong degree of redundancy appears, even on plain text. For instance, tests realized with the WordNet [60, 61] English Language Thesarus and simple English text, shows a 15-20% redundancy between name and adjective synsets in the same sentence or in neighbour ones.

"*Fuzzy*" thesauri, associating a weight to each word in a synset, can be built recursively from classical thesauri, so as to provide a high level of connectivity. Moreover, the weights —a measure of the specificity of a word to the sense designed by the synset— actually improve the significance of the used correlations.

Different thesauri for different fields The use of specialised thesauri, often available in technical and scientific domains, generally improves the search results when used for related queries, since they are more precise and comprehensive on the field they are specialised in.

Specific thesauri can be used at two levels: in rewriting rules, or directly in genetic operators. A rewriting rule either uses the generic capabilities of a particular thesaurus (that is, replacing all terms in a query by their synset according to this thesaurus) or modifies a particular term in a way coded in this rule. Genetic operators modify the query, mutating sub-rules or explicit terms, replacing one of these by another, from the synset of the original term.

6.2 User interaction and information feedback

The fact that human users are variable and interested in different fields makes it difficult for a rewriting algorithm to be robust, as a typical user will interleave queries on diverse topics.

Fortunately, splitting the analysis task between multiple rules (as implied by the evolutionary framework described below), does not require a particular scheme to handle the full complexity of the decision process, but merely to be able to determine whether it could apply on the current query or not.

In the proposed architecture, user feedback is the controlling stage. It decides of the pace and direction of evolution, through fitness calculation. As such, fitness analysis and handling is a crucial point for the system to be efficient.

7 ELISE: an Evolutionary Learning Interactive Search Engine

ELISE is based on an interactive EA, that evolves a "user profile" made of the whole population of individuals (cf: Parisian approach). Interactions with the user are analysed to improve the whole population (the profile) at each generation (see fig. 4).

The system 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.

ELISE can be used as an improving mechanism on top of any kind of search engine. The one used for experiments (see section 8) is SWISH++, [46], a basic public-domain boolean search engine, without any semantic expansion nor stemming tool.

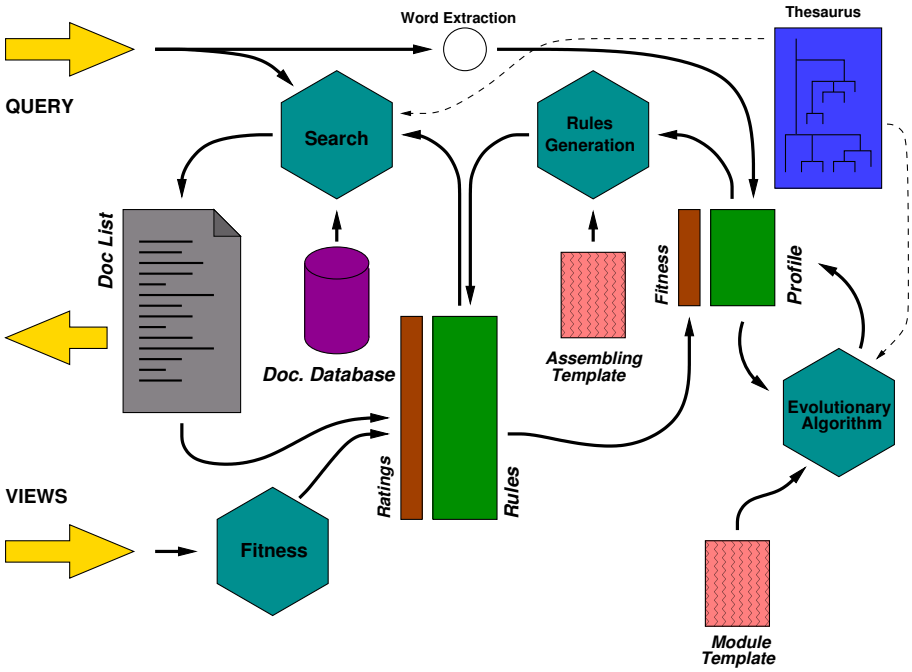

Fig. 4. ELISE: A new query of the user is rewritten using rules generated from the user profile (evolved by the EA). The rules produce a set of queries that are presented to the data base and return results. The interaction with the user (viewed documents) is used to rate the rules and to update the fitness of the modules of the profile. A new generation is then produced by the EA.

7.1 Encoding a profile in an EA population: genome and structure

As EA performance strongly depends on the adequation between the genotype and the optimisation task to be performed —or, when applicable, the topology of the search space— the instruction set must be finely tailored to the context of query rewriting,

calling for specific semantic tools, designed for an *in-depth* analysis of a short sequence of words.

The profile, i.e. the EA population, is used to modify the query of the user. Several possibilities have been considered as basic population components (Parisian approach):

1. evolving user-specific semantic networks (or thesauri),
2. evolving queries themselves,
3. evolving rewriting rules,
4. evolving smaller components of rewriting rules.

The first two choices would have implied some amount of off-line processing (limiting real time interaction).

Experiments with earlier versions of ELISE have shown that the third item was not a good choice: finding the right modification to apply to the query is too difficult a task without hundreds of trials. The tests showed that the search space was too large to be correctly sampled, resulting in at best random and at worst empty result sets.

Methods naturally used by human users when results are not good enough are generally the same, whatever the subject of the query. Users replacing terms by synonyms, excluding uninteresting terms that appear too often in the result list, etc. This means that even if a query is not related to the previous one, the refining methods evolved by the user still apply.

The current ELISE system therefore tries to do the same: evolve methods that will turn an original query into a better one, by evolving modules that will for instance expand the first word into its synonyms, then intersect the result with synonyms of the second word. A population of such small rewriting modules is evolved with the aim of maximising the number of modules that correspond to rewriting techniques the user likes most.

These small modules are then assembled into more complex rewriting rules in order to modify a complete query.

This structure is typical of a Parisian approach in the sense that the very difficult original problem (rewriting a complete query) is decomposed into interconnected subproblems that are much easier to solve.

Evolving modules The way small modules are coded has a strong impact on the efficency of the whole system. They need to inter-operate and their structure must allow them to be recombined and mutated so that they can be optimised efficiently.

Constraints can be described as follows:

- A boolean query is essentially a tree, where terminals are query terms, and nodes boolean operators.
- There is no theoretical limit to the complexity of such trees (i.e. depth) and as common web search-engine practice reveals, no meaningful empirical limit either.

This implies that it must be possible to process arbitrarily large sets of data, meaning that modelisations equivalent to finite automata (like tree-based codings used in GP) are not general enough.

Theoretically, RABIN automata (essentially, automata able to recognize infinite words described by ω-rational expressions) should be enough, since the set of words recognized by the underlying search engine is finite [31, 30, 32]. But the size of the alphabets involved would make it impractical.

Additionally, the requirements of genetic operators (mutation / crossover) must be satisfied, as they need to operate on these program representations. In ELISE, this essentially means having a very lax coding structure, since it will be needed to cut and merge program sections or change elements and still obtain valid codings.

Finally, a RPL (Reverse Polish Lisp) structure —a particular case of stack-based languages— satisfies the constraint. It 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 to make it possible to reuse an important part of the theoretical and practical tools developed throughout the history of genetic programming.

7.2 OKit: A toolbox working on RPL structures

As the generality and description capacity of program codings is a common problem in GP, a generic set of tools called *OKit* was developed to handle RPL structures, independently of the text-retrieval context [36]. In this representation, programs are lists, containing either terminals or sub-lists. Two types of terminals are available:

1. Instructions, that take arguments and put back results on a stack.
2. Character strings, that are simply pushed onto the stack when the program is run.

In this representation, there is no difference between data (e.g. queries to be transformed) and program parts.

The drawback of a lax syntax is that, depending on the set of instructions, some programs can not be guaranteed to run. However, evolutionary optimization has a wide range of tools to deal with invalid individuals and allowing for such transgressions in many cases actually benefits to the efficiency of optimisation algorithms.

Instruction set If the representation is general enough to be usable in most genetic programming cases, the necessary adaptation to the specificities of the problem lies mainly in the creation of an appropriate set of instructions. While the basic stack operations need to remain present in most applications (duplication or deletion of an element, list operations...), a set of instructions able to operate on the particular data that is dealt with is needed. 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 diverse flavours of term expansion that can be based on a semantic network.
Tree operations consist of:

- conditionals, based on a particular value of the root node,
- splitting instructions, that can break a tree into subtrees,
- merging instructions, that join two subtrees into a single one with a given root node value.

Moreover, it is possible to take advantage of the similar structure shared by data and program code in this representation, by introducing mapping instructions, that will apply a list (i.e. program) on every node or every terminal, of a query. As this type of instruction enlarges considerably the description space of the coding, which is also the search space of the ELISE learning machine, mapping instructions should be used with care.

7.3 Modules initialisation

Initialisation templates are used to create a minimal proportion of reasonable modules (30%), the rest of the initial population being made of randomly generated modules.

Then, of course, artificial evolution takes over and the modules of the best rules are rewarded through the fitness function.

7.4 Variation operators on modules

To produce a new module from two parents modules, mutations are performed on both parents, after which a crossover is performed between them. A "migration" operator (that tries to repair a possibly unfeasible module) is then applied to the result of the crossover.

As explained earlier, a module is a program part. Even with the lax syntax that is adopted, some of the modules fail to run. While keeping unfeasible individuals may be seen as a reservoir of unexpressed genes, the occurrence of such individuals should remain low.

Consequently, genetic operators must be designed so as to produce valid structures most of the time. This means essentially taking into account instructions arity and input types when doing mutations or crossover, as well as "repairing" discrepancies when needed, by introducing extra arguments or ignoring others, as required to keep valid signatures of input and output mutated or recombined instructions.

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

Local, intra-class mutation that 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 prototype (i.e. the same arity and type signature).

Local, inter-class mutation that:
- changes an instruction into another, with a different prototype, repairing the resulting module as above,
- or turns an terminal into another terminal of a different type altogether (i.e. changing an instruction into a character string).

Global (structural) mutation that changes
- the structure itself of the module, by replacing an terminal by a list,
- or the opposite.

Following the same idea, two types of crossover are used:

A local version will not descend into sublists leaving them unchanged,
A global version will also apply a crossover in sublists when possible.

These crossovers rely on the same mechanisms as those used for bit strings (remember that modules are lists, too): 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 a low proportion of invalid individuals, a *migration* operator is added to the usual mutation and crossover operators, that actually repairs the most obvious incompatibilities in the module, thus bringing it closer to the set of feasible individuals. This is generally enough to ensure that the individual will be usable in most cases.

7.5 Examples

Example of modules :

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

Initially, the original user query is placed on top of the stack. In Reverse Polish Notation, the first module gets interpreted as follows :

- `VOCGEN`: generalises all the terms of the query.
- `"metabolic disorder"`: Pushes the term "metabolic disorder" on top of the stack.
- `VOCSYN`: finds synonyms for what is on top of the stack, i.e. "metabolic disorder."
- `#AND`: restricts the generalisation of the terms of the query to those that also are synonyms of "metabolic disorder", meaning that only documents containing synonyms of "metabolic disorder" will be retrieved.

As for the second one:

- `[` : grouping operator meaning we are entering a subroutine.
- `SPLIT`: cuts the original query in two parts, separating the two operands of the higher boolean operator. The stack now contains two subqueries that were the operands of the top operator that has been removed.
- `"acetic acid"`: Pushes the term "acetic acid" on the stack.
- `VOCGEN`: Generalises the item on top of the stack (i.e. "acetic acid" in this case).
- `#AND`: Restricts the scope of the second operand of the operator removed by the `SPLIT` to documents containing generalisations of "acetic acid."
- `#OR`: Group the new modified second operand with the orginal first operand using an `OR`.
- `]` : Closes the subroutine, that will now appear on top of the stack.
- `IFOR` : Applies the above subroutine to the query, if and only if its top operator is an `OR`.

Examples of mutations 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" terminal.

Another mutation, on the second module, has given:

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

This is a result of replacing instructions inside their class, as well as removing the AND operator.

In this case, the scope restriction introduced by the AND operator has been replaced by a scope generalisation, meaning that documents containing "acetic acid" or any of its generalizations will also be retrieved.

Then, the two operands that have been split are not grouped any more, since an operator has disappeared. When the group is exited with the], two subqueries are on the stack: the second operand that has been modified, and the unmodified first operand of the original query.

Finally, this new mutated module will now operate only on queries featuring an AND as top operator.

Example of a crossover A crossover between both mutated modules could read:

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

The result is a module that will perform all the operations described above, to which synonyms of all the terms of the second operand are added before it is grouped with the unchanged first operand with an AND operator.

7.6 Assembling modules into rewriting rules for profile evaluation

The initialisation step is important in order to provide reasonable answers, even with an initial non-evolved user profile.

Therefore, the more complex rules built from the modules presented above use a set of templates that perform various plausible 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", as well as "VOCSYN", "VOCGEN" and others).

Modules are chosen with a stochastic selector (tournament) in order to favour the expression of good modules in the derived rules.

Example:

```
[ @ @ «#AND» ]
[ @ «VOCSYN» ]
```

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

In the results presented below, 50 rules are created, that retrieve 50 different sets of documents. The untouched original query is also added to these rules so as to be sure that the system is really an addition to the basic search engine.

The problem is now to efficiently use the meager pieces of information collected from the interaction with the user. The situation is typically that of a low feedback, the only information available being which and how many documents have been viewed by the user. In the Parisian Approach paradigm, there are two fitness functions: a global one, corresponding to the global evaluation (number of viewed documents, for instance), and a local fitness function (fitness of each module).

The current proposed solution is to associate two counters ($C_{retrieved}$ and C_{count}) to each module of the population. The results presented below use the CFD benchmark (see section 8 below), which allows to determine the precision and recall rate for each query. These quantities are translated into a bonus that is worth *Recall_rate* $+\alpha*$ *Precision*.

As the algorithm keeps track of which module is used in which rule that retrieved which document the calculated bonus is 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 whenever they have been used to construct a rule.

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

7.7 Engine parameters

As is often the case in the Parisian Approach paradigm, elitism is quite massively used, in the sense that a significant part of the population (the 70% top ranked modules) is transmitted unchanged to the next generation. These 70% best modules represent the core of the user profile that must not change quickly.

Mutation and crossover rates used for the generation of the 30% remaining modules are respectively set at 30% (per gene) and 80% (per individual). The typical module population size is 50.

8 Tests and Results

Tests presented in this section have been performed automatically on the Cystic Fibrosis Database (CFD)[7], using a basic boolean search engine (SWISH++ [46]). The semantic operators are based on a medical oriented thesaurus: MeSH [33].

Automatically testing such an interactive system is of course extremely difficult (especially with respect to the "user satisfaction" quantity). However, an automatic test remains interesting in the sense that it allows to gather statistics rather easily. The CFD benchmark consists of 1239 documents published from 1974 to 1979 discussing a specific topic (cystic fibrosis aspects). A set of 100 queries exists, that come along with the relevant documents that should be retrieved by each query.

Therefore, the global recall rate and precision of the system can be precisely measured by comparing the returned document set to the documents that should have been retrieved.

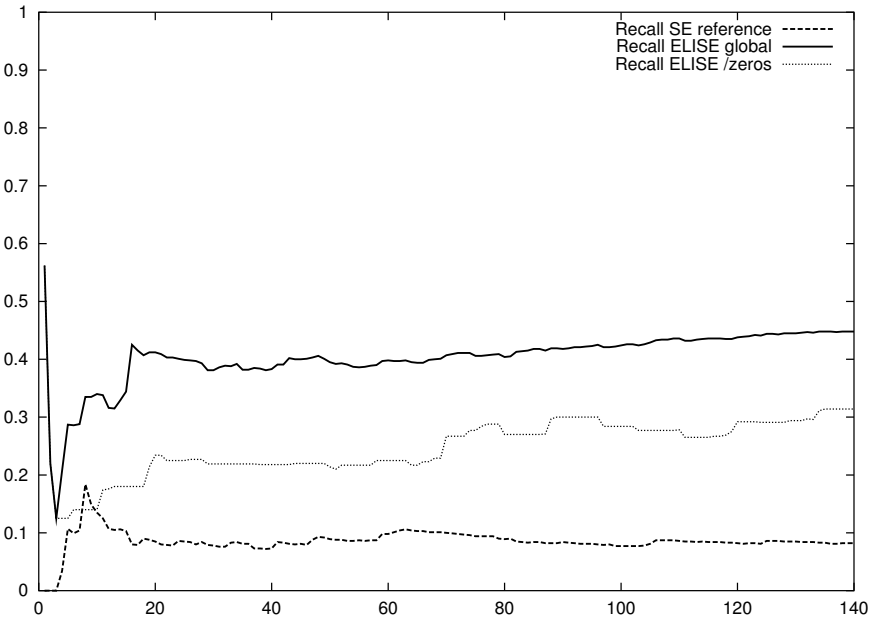

Fig. 5. Cumulated mean recall rates of the underlying boolean search engine (dashes), of ELISE (solid line), and of ELISE for queries for which the underlying search engine returns no answer (dots).

8.1 Search performance

Figure 5 presents three curves: The bottom one is the cumulated mean recall rate of the basic search engine (SWISH++) on the original query, showing that in average, searches made on the words of the query only retrieve 10% of the documents that should have been found.

The top curve is the cumulated mean recall rate of ELISE. After some hesitations corresponding to the initial learning stage, ELISE is capable, after 18 queries only, of finding around 40% of the interesting documents. This percentage keeps rising up to 45% at the end of the test of 140 queries, (i.e. 140 generations of the Parisian EA).

The most interesting curve is however the middle one, that shows the recall rate of ELISE on the queries on which SWISH++ retrieved no relevant document.

This curve somehow represents the added value of the ELISE system over a boolean search engine. The relevant documents that have been found were retrieved only thanks to the modules and rewriting rules of the ELISE system.

The fact that this curve is rising shows that positive evolution takes place in the modules. If the modules had evolved randomly without learning a user profile (interested in CFD, in this case), the curve would have looked flat, like the bottom curve of the SWISH++ boolean search engine.

Figure 6 presents the same set of cumulated mean curves showing precision.

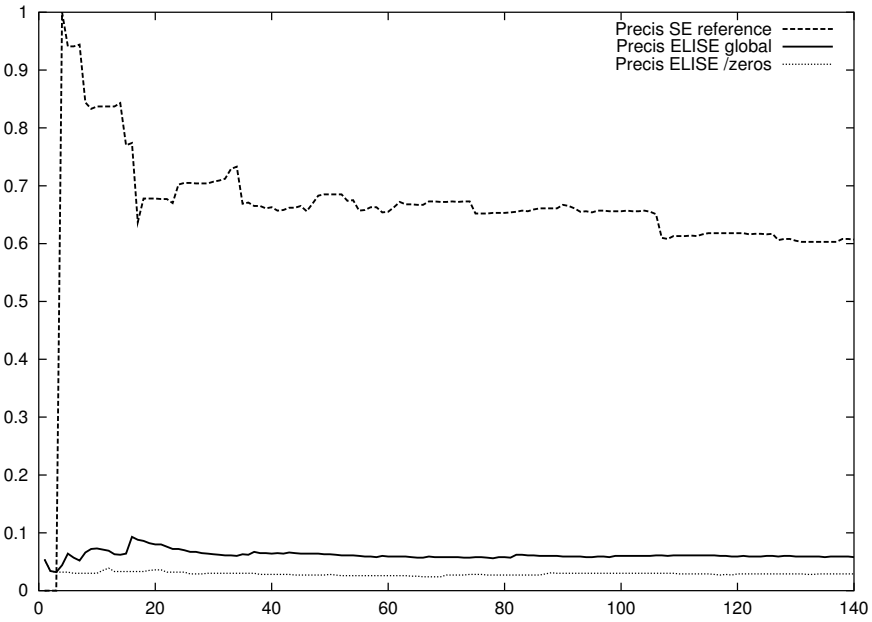

Fig. 6. Cumulated mean precision of the underlying Boolean Search Engine (dashes), of ELISE (solid line), and of ELISE for queries for which the BSE returns no answer (dots).

The top curve is now that of the boolean search engine. On the contrary of recall rate (that was very poor as the BSE only retrieves documents that contain all words of the query) precision is quite high as it is unlikely that documents that are totally unrelated to the query will contain all the words of the query.

The middle curve is that of ELISE, which provides a much lower precision than the basic seach engine. This behaviour was expected, since queries have been rewritten specifically in order to decrease precision in favour of diversity in the retrieved documents.

The bottom curve now corresponds to the precision of the ELISE system for queries on which the boolean search engine returned no answer (i.e. on which the query was so badly formulated that no interesting document contained all words of the query). The fact that this curve is lower than that for which the query is better formulated shows that ELISE does not return documents at random based solely on its evolved modules. Better formulating a query does indeed give a better precision, even with the complex ELISE rewriting system.

Figure 7 shows the evolution of the best and mean fitness of the modules. Interestingly enough, the mean fitness does not increase until generation 40. Some of the best modules of the final population (after 140 generations) are detailed and explained below.

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

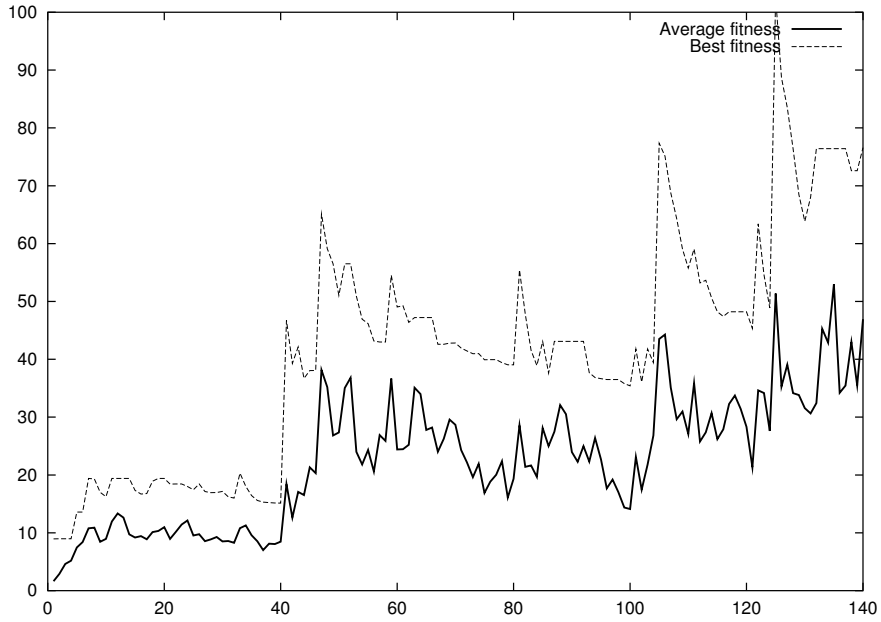

Fig. 7. Evolution of the fitness of the modules : best individual (dashes) and mean fitness for the whole population(solid line).

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

```
[ «#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 shown to Thierry Prost, medical practicioner. His 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.*

This analysis seems to show that the ELISE system hosts unexpected keywords in its modules, that might potentially provide lateral thinking capabilities to conventional search engines.

8.2 Evolutionary behaviour

Using the Parisian evolutionary paradigm allows ELISE to be efficient with a very simple architecture. Emphasis has been made on tailoring genetic operators for the dedicated task of evolving programs. In particular, shielding emergent structures from disruptive evolutionary effects, exploring variations around performant modules have been prevalent in the conception of these operators.

However, the genetic engine itself has been kept very simple, and does not make use of the numerous schemes elaborated to counter some drawbacks of genetic programming. Analysing the evolutionary behaviour of ELISE during the experiments on the CFD (cf. below) show that most of these do not appear in this context.

Diversity The notion of distance between individuals is very difficult to implement, particularly so in interactive evolution. The design of the ELISE prototype has therefore let aside any consideration of diversity management, that can be implemented using techniques such as niching or sharing.

In most EA applications, evolution will tend to duplicate (with insignificant variations) the best individuals to fill the population. When mutation remains a "local" operator (i.e. mutated offspring remain close from their ascendants) this can slow down considerably, or even freeze, the evolution.

This effect can be limited easily in the Parisian paradigm. The local fitness can be easily tailored to include a proximity penality.

If this proximity information originates in a genotypic comparison, it provides an easy implementation of sharing techniques in the Parisian context. However, a phenotypic comparison can also be used. In ELISE, this would lead to comparing returned document sets at each query, and integrating the distance between these sets for each module into its fitness.

Comparing the semantic content of modules and queries in the CFD set shows that modules are specialized to handle subsets of queries, related to particular topics. For a given query, modules that are not specific will receive a very small reward at this generation. As a consequence, it is likely that they will not be selected to create children, even if there is a good chance they are kept in the next generation.

This can be seen practically by analysing the respective age of the best individuals, and the length of their bloodlines. High fitness individual are generally young, and have short bloodlines. High fitness is thus very temporary, but medium fitness individuals, that remain longer in the populations, mecanically receive lower rewards. This ensures they do not fill the population, allowing creative individuals to emerge.

This variability limits the need for diversity management techniques in the evolutionary engine.

8.3 Code growth and redundancy

Bloat is often considered as a plague of variable-length genome approaches. An alternate interpretation in GP presents introns as necessary buffers to protect emergent structures from evolutionary damage.

In any case, bloat and introns are often subject to a careful analysis in GP applications. However, tests on ELISE show that bloat does not appear, whatever evolutionary parameters were chosen. In fact, an analysis of the largest individuals whenever they appeared showed that most of them where "infeasible" individuals, that would not run on most of the queries they had to process.

In ELISE, combining the natural penality against infeasible individuals and global operators thus leads to an implicit penality against large individuals. While not forbidding explicitely arbitrarily large individuals, this has the effect of limiting code growth.

9 Conclusion and future work

This papers shows that it is possible to use evolutionary techniques to refine search engines in the framework of scientific publications related to medecine, biology and bio-

chemistry. The tests presented in this paper above a basic boolean search engine show that the system is able to adapt (improve its recall rate) and discover new associations.

This work has been driven by tests on relevant benchmarks. However, due to the eminent user-related nature of the fitness function, only real-world testing will shed light on parts of the system that should be modified or upgraded.

Real-size experiments will be conducted soon on real databases with a more efficient underlying search system at Novartis-Pharma (Ulix search engine, [55]). Conditions will therefore be quite different from test benches, and hopefully user feedback will be of good enough quality to guide future developments. The present system must therefore be open and versatile enough so that modifications do not require drastic changes in the structure of the algorithm.

Many things remain to be done, even though ELISE is already a functional prototype:

Diversity control A major component of Parisian approaches, which once again appears as crucial in the test presented in this paper, is the diversity control of the population of modules. It is based on a sensible measurement of the distance between individuals with respect to the problem and keeping in mind the computational cost.

This point has been temporarily simulated in the current implementation by artifically increasing probabilities of operators that usually increase genetic diversity. However this problem has to be considered carefully for the real-life experiments to come.

Level of abstraction Until now, methods have been suggested to improve information processing with EAs under the assumption that this information was only of a quantitative nature, directly convertible into fitness. However, since the system deals with texts and senses, document and query analysis provides essentially semantic information.

This "higher level" intelligence could also be used to steer the evolution toward regions of interest, as determined by the analysis of the document-to-query relation. Several experiments using symbolic information to steer the evolution have proved highly successful (for instance [40, 13, 34, 41, 45, 13, 40]). Most of these use this information to alter the behaviour of genetic operators, putting forward particular genetic constructs.

Thesauri To enlarge the semantic scope of genetic operators and instructions, multiple thesauri should be used, like MeSH, but also EmTree[16] and WordNet [61] or even dynamically built user specific and fuzzy thesauri.

For instance, a dynamic thesaurus, generated from the analysis of term relations in queries and visited documents, could be used to determine term mutations in the rules, favouring recently extracted terms over coordinated terms produced by reference thesauri. To be exploited to full potential, this method will need much fine-tuning.

Ranking and aggregation of results sets Right now, ELISE produces multiple queries out of the single query written by the user. Therefore, several sets of documents are retrieved by the underlying search engine for each query.

Results aggregation is therefore necessary, with the important task of producing a relevant ordering for the final set that is presented to the user. Ordering is a de-

terminant part of user interaction [49], since users will seldom consult documents beyond the first twenty.
Despite its importance, the current prototype of ELISE does not yet attempt to solve this problem, as any test until now could only be conducted with machine-readable benchmarks, where ordering is not taken into account.
Ordered sets resulting from the different queries passed to the search engine are therefore simply interleaved, so that the best matches for each sets are presented to the user.

Acknowledgements

The authors are grateful to Thierry Prost, MD, PhD, for his educated analysis of the innards of ELISE.

References

1. W. Banzhaf, "Interactive Evolution," in *Handbook of Evolutionary Computation*, 1997, Oxford University Press.
2. A. Boumaza and J. Louchet, "Dynamic Flies: Using Real-Time Parisian Evolution in Robotics," In *EVOIASP 2001*, Lake Como, Italy, 2001.
3. R. J. Brachman "What's in a Concept: Structural Foundations for Semantic Networks," *International Journal of Human-Computer Studies* 9, 1977
4. C. Brouard "Construction et exploitation de réseaux sémantiques flous pour l'extraction d'information pertinente: le système RELIEFS," PhD. Thesis, LIP6, Université Paris-6, 2000
5. E. Cantu-Paz and C. Kamath, "On the use of evolutionary algorithms in data mining," In Abbass, H., Sarker, R. and Newton, C. (Eds.) *Data Mining: a Heuristic Approach*, pp. 48-71. Hershey, PA: IDEA Group Publishing, 2002.
6. E. Cantu-Paz, C. Kamath, "On the use of Evolutionary Algorithms in Data Mining," in *Data Mining: a Heuristic Approach*, H. A. Abbass, R. A. Sarker and C. S. Newton (Eds), Idea Group Publishing, 2001.
7. Cystic Fibrosis Reference Collection, `http://www.sims.berkeley.edu/ hearst/irbook/cfc.html`
8. J. Chapuis, E. Lutton "ArtiE-Fract: Interactive Evolution of Fractals," *GA '01, Generative Art Conference*, Milano, Italy, 2001
9. H. Chen, "Machine learning for information retrieval: neural networks, symbolic learning and genetic algorithms," JASIS vol. 46 (3), *Journal of the American Society for Information Science and Technology*, April 1995.
10. H. Chen "Machine Learning for Information Retrieval: Neural Networks, Symbolic Learning and Genetic Algorithms," *Journal of the American Society for Information Science*, 1995
11. H. Chen, G. Shankaranarayanan, L. She and A. Iyer, "A Machine Learning Approach to Inductive Query by Examples: An Experiment Using Relevance Feedback, ID3, Genetic Algorithms and Simulated Annealing," JASIS vol. 49 (8), *Journal of the American Society for Information Science and Technology*, June 1998.
12. 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*, Volume 1, Issue 4, pp. 339-361, October, 2000.
13. L. Davis "Adapting operator probabilities in Genetic Algorithms," *ICGA '89, International Conference on Genetic Algorithms and their Application*, 1989.

14. S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, R. Harshman "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, 1990.
15. A. Dix, "Interactive Querying, locating and dicovering information," *Second Workshop on Information Retrieval and Human Computer Interaction*, Glasgow, 11th September 1998, http://www.hiraeth.com/alan/topics/QbB/
16. EMBASE, the Excerpta Medica database Elsevier Science, Secondary Publishing Division, New York
17. D. J. Foskett. "Thesaurus," *Readings in Information Retrieval*, K. S. Jones, P. Willet, M. Kaufmann Publishers, San Fransisco, 1997
18. A. Freitas, "A survey of evolutionary algorithms for data mining and knowledge discovery," to appear in: Ghosh, A.; Tsutsui, S. (Eds.) *Advances in evolutionary computation*, Springer-Verlag, 2001, citeseer.nj.nec.com/freitas01survey.html
19. 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.
20. A.A. Freitas, "A survey of evolutionary algorithms for data mining and knowledge discovery," To appear in: A. Ghosh and S. Tsutsui. (Eds.) Advances in Evolutionary Computation. Springer-Verlag, 2002. http://www.ppgia.pucpr.br/ alex
21. D. A. Goldberg "Genetic Algorithms in Search, Optimization and Machine Learning," *Addison-Wesley Publishing*, 1989
22. M. D. Gordon, "Probabilistic and Genetic Algrithms for Document Retrieval," *Communications of the ACM* 31, pp 1208-1218, 1988.
23. T. Hofmann "Probabilistic Latent Semantic Indexing," *SIGIR '99, International ACM SIGIR Conference on Research and Development in Information Retrieval*, Berkeley, USA, 1999
24. J.-T. Horng and C.-C. Yeh, "Applying Genetic Algorithms to QUery Optimisation in Document Retrieval," *Information Processing and Management* 36, pp 737-759, 2000
25. S. Kamohara, H. Takagi and T. Takeda, "Control Rule Acquisition for an Arm Wrestling Robot," in *IEEE Int. Conf. on System, Man and Cybernetics* (SMC'97), vol 5, Orlando, FL, USA, pp 4227-4231, 1997.
26. 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.
27. S. Kim and B.-T. Zhang, "Evolutionary Learning of Web-Document Structure for Information Retrieval," Proceedings of the *2001 Congress on Evolutionary Computation* (CEC2001), vol. 2, pp. 1253-1260, 2001.
28. S. Kim, B.-T. Zhang, "Evolutionary Learning of Web-Document Structure for information Retrieval," *Proceedings of the 2001 Congress on Evolutionary Computation* (CEC2001), vol. 2, pp. 1253-1260, 2001.
29. M. L. Wong, K. S. Leung and J. C.Y. Cheng, "Discovering Knowledge from Noisy Databases Using Genetic Programming" JASIS vol. 51 (9), *Journal of the American Society for Information Science and Technology*, July 2000.
30. R. McNaughton, "Algebraic decision procedures for local testability," *Math. Systems Theory*, vol 8, 1974, number 1, pp 60–76.
31. R. McNaughton, "Testing and generating infinite sequences by a finite automaton," *Information and Control*, vol 9, 1966, pp 521–530.
32. , R. McNaughton and S. Papert, "Counter-free automata," With an appendix by W. Henneman, *M.I.T. Research Monograph*, No. 65, The M.I.T. Press, Cambridge, Mass.-London, 1971.
33. "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
34. C. Meyer, J.-G. Ganascia, J.-D. Zucker "Learning Strategies in Games by Anticipation," *IJCAI '97, International Joint Conference on Artificial Intelligence*, Nagoya, Japan, 1997

35. N. Monmarche, G. Nocent, G. Venturini and P. Santini. "On Generating HTML Style Sheets with an Interactive Genetic Algorithm Based on Gene Frequencies," *Artificial Evolution*, European Conference, AE 99, Dunkerque, France, November 1999, Selected papers, Springer Verlag, LNCS 1829, 1999, C. Fonlupt and J. K. Hao and E. Lutton and E. Ronald and M. Schoenauer (Eds).
36. Y. Landrin-Schweitzer, OKit, a Virtual Machine and Compiler for Concatenative Languages. `http://varkhan.free.fr/Software/OKit`
37. S. Pal, V. Talwar and P. Mitra, "Web Mining in Soft Computing Framework: Relevance, State of the Art and Future Directions," IEEE Transactions on Neural Networks, 2002, citeseer.nj.nec.com/pal02web.html
38. R.S. Parpinelli, H.S. Lopes and A.A. Freitas, "Data Mining with an Ant Colony Optimization Algorithm," To appear in IEEE Trans. on Evolutionary Computation, special issue on Ant Colony algorithms. 2002.
39. R. Poli and S. Cagnoni, "Genetic Programming with User-Driven Selection : Experiments on the Evolution of Algorithms for Image Enhancement," in *2nd Annual Conf. on Genetic Programming*, pp 269-277, 1997.
40. J. D. Schaffer, A Morishima "An Adaptive Crossover Distribution Mechanism for Genetic Algorithms," *ICGA '87, International Conference on Genetic Algorithms*, 1987.
41. M. Sebag, C. Ravisé, M. Schoenauer "Controlling Evolution by Means of Machine Learning," *EP 1996, 5th Annual Conf. on Evolutionary Programming* (EP'96), March 1996, San Diego, USA.
42. D.-H. Shin , Y.-H. Kim, S. Kim, J.-H. Eom, H.-J. Shin and B.-T. Zhang, "SCAI TREC-8 Experiments," Proceedings of the *Eighth Text Retrieval Conference* (TREC-8), pp. 511-518, 1999.
43. K. Sims, "Interactive evolution of dynamical systems," in *First European Conference on Artificial Life*, pages 171–178, 1991. Paris, December.
44. K. Sims, "Artificial Evolution for Computer Graphics," *Computer Graphics*, 25(4):319–328, July 1991.
45. W. M. Spears "Adapting crossover in a Genetic Algortithm," *ICGA '91, International Conference on Genetic Algorithms*, 1991.
46. SWISH++, Simple Web Indexing System for Humans: C++ version, `http://homepage.mac.com/pauljlucas/software/swish/`
47. H. Takagi, " Interactive Evolutionary Computation : System Optimisation Based on Human Subjective Evaluation," *IEEE Int. Conf. on Intelligent Engineering Systems* (INES'98), Vienna, Austria, pp 1-6, Sept 17-19, 1998.
48. H. Takagi, M. Ohsaki, "IEC-based Hearing Aids Fitting," *IEEE Int. Conf. on System, Man and Cybernetics (SMC'99)*, Tokyo, Japan, vol 3, pp 657-662, Oct. 12-15, 1999.
49. "The BEST Search Engines", UC Berkeley - Teaching Library Internet Workshops `http://www.lib.berkeley.edu/TeachingLib/Guides/Internet/SearchEngines.html`
50. S.J.P. Todd and W. Latham, "Evolutionary Art and Computers,", *Academic Press*, 1992.
51. The Text REtrieval Conference (TREC) homepage `http://trec.nist.gov/`
52. P. Kantor, E.M. Voorhees, "TREC-5 Confusion Track", Proceedings of the Fifth Text REtrieval Conference (TREC-5), Gaithersburg, Maryland, Nov. 20-22, 1996.
53. E.M. Voorhees, "TREC-2003 Robust Retrieval Track", Proceedings of the Twelfth Text REtrieval Conference (TREC 2003) Gaithersburg, Maryland, November 18-21, 2003
54. L.Y. Tseng and S. B. Yang, "Genetic Algrithms for Clustering. Feature selection and Classification." *International Conference on Neural Networks*, Vol 3, pp 1612-1616, 1997.
55. T. Vachon, N. Grandjean, P. Parisot, "Interactive Exploration of Patent Data for Competitive Intelligence: Applications in Ulix (Novartis Knowledge Miner)," *International Chemical Information Conference and Exhibition*, Nîmes, France, 21-24 October 2001. `http://www.infonortics.com/chemical/ch01/01chempro.html`

56. M. J. Martin-Bautista, M.-A. Vila and H. L. Larsen, "A Fuzzy Genetic Algorithm Approach to an Adaptive Information Retrieval Agent," JASIS vol 50 (9), *Journal of the American Society for Information Science and Technology*, July 1999
57. D. Vrajitoru, "Genetic Algorithms in Information Retrieval," *AIDRI97*, Learning; From Natural Principles to Artificial Methods , Genève, June 1997.
58. D. Vrajitoru, "Large Population or Many Generations for Genetic Algorithms ? Implications in Information Retrieval", In F. Crestani, G. Pasi (eds.): *Soft Computing in Information Retrieval. Techniques and Applications*, Physica-Verlag, Heidelberg, pp 199-222, 2000.
59. W. Winiwarter, "PEA-A Personal E-mail Assistant with Evolutionary Adaptation," *International Journal of Information Technology*, 1999. `http://citeseer.nj.nec.com/winiwarter99pea.html`
60. C. Fellbaum, G. A. Miller, "WordNet: An Electronic Lexical Database," The MIT Press, May 1998
61. G. A. Miller, C. Fellbaum, R. Tengi, P. Wakefield, "WordNet, an Electronic Lexical Database for the English language: `http://www.cogsci.princeton.edu/ wn/`" Cognitive Science Laboratory, Princeton University.
62. J. Yang, R. R. Korfhage and E. Rasmussen, "Query Improvement in Information Retrieval using Genetic Algorithms: A Report on the Experiments of the TREC project", The first Text Retrieval Conference (TREC-1), 1993.
63. J. Yang and V. Honavar, "Feature extraction, Construction and Selection - A Data Mining Perspective," *Kluver Academics Publishes*, pp 117-136, 1998.
64. B.-T. Zhang, J.-H. Kwak, C.-H. Lee, "Building Software agents for information fitlering on the internet: A Genetic Programing approach," *Genetic Programing Conference*, 1996.