Mercure at trec8 : Adhoc, Web, CLIR and Filtering tasks

M. Boughanem, C. Julien, J. Mothe & C. Soule-Dupuy

IRIT/SIG
Campus Univ. Toulouse III
118, Route de Narbonne
F-31062 Toulouse Cedex 4
Email : trec@irit.fr

1 Summary

The tests performed for TREC8 were focused on automatic Adhoc, Web, Clir and Filtering (batch and routing) tasks. All the submitted runs were based on the Mercure system.

Automatic adhoc : Four runs were submitted. All these runs were based on automatic relevance back-propagation used in the previous TREC, with a slight change for one of these runs (Mer8Adtd3). A strategy based on predicting the relevance of documents using the past relevant documents was tested for this run. More precisely, instead of using the same relevance value for all top retrieved documents, some of them are selected and have their relevance value boosted.

Web : Four runs were submitted in this track:

1. content based only using Mercure simple search
2. content+ilink, according to Mercure architecture, we consider that document nodes are linked each other by weighted links. The top selected documents resulting from the initial search spread their signals towards the other document nodes. The documents were then sorted according to their activations, the top 1000 documents were submitted.
3. (2) + pseudo-relevance back-propagation method.
4. reranking of the 40 top documents using their links between each others.

Cross-language : Three runs were submitted for our first participation in this track. All these runs were based on query translation using an online machine translation . Two of these runs are a comparison between query translation from English to other languages and from French to other languages.

Filtering - batch and routing : The profiles were learned using three different strategies : Relevance Back-propagation (RB) and Gradient Back-propagation (GB) used in the

previous TREC and a new strategy based on Genetic Algorithm (GA). Four runs were submitted, two batch runs based on RB+GB and two routing runs, one based on RB+GB and the other one based on GA.

2 Mercure model

Mercure is an information retrieval system based on a connectionist approach and modelled by a multi-layered network. The network is composed of a query layer (set of query terms), a term layer representing the indexing terms and a document layer [3],[1].

Mercure includes the implementation of a retrieval process based on spreading activation forward and backward through the weighted links. Queries and documents can be used either as inputs or outputs. The links between two layers are symmetric and their weights are based on the tf.idf measure inspired from the OKAPI and SMART term weightings.

- the query-term (at stage s) links are weighted as follows:

$$q^s_{ui} = \frac{(1+log(tf_{ui}))*(log(N/n_i)}{\sqrt{\sum_{i=1}^{T}(1+log(tf_{ui}))*(log(N/n_i)^2}} \text{ if } tf_{ui} \neq 0$$
$$q_{ui} = 0 \text{ otherwise}$$

- the term-document link weights are expressed by:

$$w_{ij} = \frac{(1+log(tf_{ij}))*(h_1+h_2*log(\frac{N}{n_i}))}{h_3+h_4*\frac{doclen_j}{avg\_doclen}}$$

The query evaluation is based on spreading activation. Each node computes an input and spreads an output signal. The query modification is based on relevance back-propagation. It consists in spreading backward the document relevance from the document layer to the query layer [1].

2.1 Query evaluation

A query is evaluated using the spreading activation process described as follows :

1. The query u is the input of the network. Each node from the term layer computes an input value from this initial query: $In(t_i) = q^s_{ui}$ and then an activation value : $Out(t_i) = g(In(t_i))$ where g is the identity function.
2. These signals are propagated forwards through the network from the term layer to the document layer. Each document node computes an input : $In(d_j) = \sum_{i=1}^{T} Out(t_i) * w_{ij}$ and then an activation , $Out(d_j) = g(In(d_j))$.

   The set of retrieved documents, $Output_u(Out(d_1), Out(d_2), ..., Out(d_N))$ is then ranked in a decreasing order of the activation value.

2.1.1 Query modification based on relevance back-propagation

The top retrieved documents are judged and a *relevance value* is assigned to each document (positive for relevant documents, negative for non-relevant documents and nil for non-judged documents). These values are used to compute the *DesiredOutput* vector.

$DesiredOutput = (rel_1, rel_2, ..., rel_N)$, $rel_j = \frac{Coef\_rel}{Nb\_rel} for\ relevant\ document$
and $rel_j = \frac{Coef\_rel}{Nb\_Nrel} for\ nonrelevant\ document$

1. This output is in fact considered as an "input" in the back-spreading process and is presented to the document layer. Each document node computes an input value, $In(d_j) = rel_j$ and then an activation signal, $Out(d_j) = g(In(d_j))$.
2. This activation is back-spread to the term layer. Each term node computes an input value, $In(t_i) = \sum_{j=1}^{N}(w_{ji} * Out(d_j))$ and then an output signal, $Out(t_i) = g(In(t_i))$.
3. Finally, the new query-term links corresponding to the new query are computed as follows: $Q_u^{(s+1)} = (q_{u1}^{(s+1)}, q_{u2}^{(s+1)}, ..., q_{uT}^{(s+1)})$ with $q_{ui}^{(s+1)} = M_a * q_{ui}^{(s)} + M_b * Out(t_i)$

   Notations :
   T: the total number of indexing terms,
   N: the total number of documents,
   $q_{ui}^{(s)}$: the weight of the term t_i in the query u at the stage s,
   t_i: the term t_i,
   d_j: the document d_j,
   w_{ij}: the weight of the link between the term t_i and the document d_j,
   $doclen_j$: document length in words (without stop words),
   avg_doclen: average document length, tf_{ij}: the term frequency of t_i in the document D_j,
   n_i: the number of documents containing term t_i,
   M_a and M_b: tuned and determined by a series of experiments and set to $M_a = 2$ and $M_b = 0.75$.

3 Adhoc experiment and results

3.1 Adhoc methodology: Query modification based on past relevant documents

The main experiment undertaken this year concerns the use of past known relevant documents. The hypothesis that has been taken is : documents that have been judged as relevant for past queries could give a good approximation of relevance in pseudo-relevance feedback context.

Usually, the values used in pseudo-relevance back-propagation are : coef_rel=1 for the top 12 documents assumed as relevant and -.75 for documents from 501-1000 assumed as non relevant.

The strategy tested this year consists of selecting some retrieved documents to be boosted instead of using the same relevance value for all the top retrieved documents. These documents are selected as follows:

1. Let us consider D_{ret}, the top 12 documents, $qrels_i$ the relevant documents for a given query q_i, i varying from 1 through 400 (past TREC qrels).

2. Build $comm_i = D_{ret} \cap qrels_i$, each $comm_i$ contains a set of documents from the top retrieved
3. Select the $comm_p$ containing the greatest number of documents and at least two documents.
4. Boost the coefficient value of the documents in $comm_p$. The coefficient value that has been used is 1.75.

3.2 Adhoc results and discussion

Four automatic runs have been submitted : Mer8Adtd1,Mer8Adtd2,Mer8Adtd4 (title+description) and Mer8Adtnd3 (long topic). These runs were based on a completely automatic processing of TREC queries and automatic query expansion based on pseudo feedback : Mer8Adtd4 is based on the query modification based on past relevant documents; Mer8Adtd1,Mer8Adtd2 and Mer8Adtnd3 are based on the classical relevance back-propagation using or not non-relevant documents.

TREC results			
Run	Best	≥median	$< median$
Mer8Adtd1	0	26	24
Mer8Adtd2	0	26	24
Mer8Adtnd3	0	28	22
Mer8Adtd4	1	27	23

Table 1: Comparative automatic adhoc results at average precision

Table 1 compares our runs against the published median runs. As mentioned in the previous TREC by [5], it is hard to tell much about relative performance since all automatic runs (title only, title+description and long topics) are in the same pool. Our results are in the average, with one best query.

Type	Run	average precision	Exact precision
title+description	basic td search	.2231	.2786
"	Mer8Adtd1 : 12 rel, 0 nrel	.2231 (0%)	.2786 (0%)
"	Mer8Adtd2 : 12 rel, 500 nrel	.2231 (0%)	.2786 (0%)
"	Mer8Adtd4 : 12 rel + past relevance	.2247 (1%)	.2733 (-2%)
Long Topic	basic tnd	.2327	.2928
"	Mer8Adtnd3	.2327 (0%)	.2931 (0%)

Table 2: Adhoc component results - 50 queries

Table 2 shows that the results obtained from all the tested strategies are quite the same. No strategy has improved the results. However, the results obtained with long topic seems to be slightly better than with description+title only. The results obtained this year are quite the same than those obtained in our previous participation in TREC.

What did we gain from the strategy based on past queries (run Mer8Adtd4)? Note that : when the condition of boosting the relevance value in Mer8Adtd4 is not satisfied for a given query, the result of this query is the same than Mer8Adtd1. The difference between these runs can be seen only when some documents from the top selected are boosted.

topics	AvgP basic search	AvgP Mer8Adtd1	AvgP Mer8Adtd4
407	0.2120	0.3012 (42.08%)	0.2876 (35.66%)
411	0.1834	0.2497 (36.15%)	0.2497 (36.15%)
412	0.1565	0.2493 (59.30%)	0.3280 (109.58%)
416	0.2595	0.2540 (-2.12%)	0.2744 (5.74%)
422	0.2503	0.2896 (15.70%)	0.2896 (15.70%)
424	0.2279	0.0899 (-60.55%)	0.0955 (-58.10%)
427	0.1866	0.1776 (-4.82%)	0.1710 (-8.36%)
429	0.4561	0.2007 (-56%)	0.2621 (-42.53%)
431	0.4994	0.3061 (-38.71%)	0.3007 (-39.79%)
436	0.0313	0.0531 (69.65%)	0.0531 (69.65%)
438	0.2157	0.3007 (39.41%)	0.3119 (44.60%)
443	0.0912	0.0941 (3.18%)	0.0960 (5.26%)
446	0.1436	0.2548 (77.44%)	0.2548 (77.44%)

Table 3: Adhoc comparison between using the relevance of past documents and no

Table 3 compares the results between Mer8Adtd1 and Mer8Adtd4, query by query. It can be seen that 13 queries were concerned with the past relevance strategy. The average precisions of 7 of them were improved, 4 were unchanged and 3 were dropped.

4 Web Track

4.1 Web methodology

Mercure has been extended to take into account the links between document nodes. The document nodes in Mercure architecture are linked each others. The weight of the link between two nodes is computed as follows :

$$d_{ij} = \frac{\|link(d_i) \cap link(d_j)\|}{min(\|link(d_i)\|, \|link(d_j)\|)}$$

where $link(d_i)$ is the list of documents (ilink or olink) linked to the document d_i. Only ilinks have been used in our experiments. Four runs using the title and the description fields, were submitted :

1. Mer8Wctd : content based only + pseudo-relevance feedback based on Mercure relevance back-propagation.

2. Mer8Wci1 : content+ilink. According to the spreading activation process of Mercure, the top 12 documents (resulting from the initial search) spread their signals through the document-document links towards the document nodes. Each document node computes an activation value as follows :

   $$Out^{(new)}(d_j) = Out^{(old)}(d_j) + g(\sum_{k=1}^{N} d_{kj} * Out^{(old)}(d_k))$$

   .

   The documents are then sorted according to their new activation. The top 1000 documents were then submitted.

3. Mer8Wci2 : (2) + pseudo-relevance back-propagation method. The top 12 retrieved documents in Mer8Wci1 were then used for relevance back-propagation and the top 1000 retrieved documents resulting from the new query were submitted.

4. Mer8Wci3 : rerank the 40 top documents using the links between documents. Instead of sorting the top 40 documents according to their activation value these documents are sorted according to another RSV (Retrieval Status Value) computed as follows : each document from the 40 top retrieved at the initial search computes an RSV :

   $$RSV(d_j) = \sum_{d_k \in top\ 40\ doc.} d_{kj}$$

   The 40 top documents are then sorted according to their RSV. The rank of the remaining documents 41-1000 is still unchanged.

Table 4 compares our runs against the published median results. Table (5) shows that using content only + relevance back-propagation (Mer8Wctd) gives better results than all the runs using the links. The strategy we used to take into account the links gives no improvment.

TREC results			
Run	Best	$\geq$median	$< median$
Mer8Wctd	0	23	27
Mer8Wci1	0	22	28
Mer8Wci2	5	19	31
Mer8Wci3	1	23	27

Table 4: Comparative automatic small web results at average precision

Type	Run	average precision	Exact precision
Title-Description	basic search	.1480	.1810
"	Mer8Wctd	.1638 (11%)	.1957 (8%)
"	Mer8Wci1	.1401 (-6%)	.1666 (-8%)
"	Mer8Wci2	.1488 (1%)	.1771 (-2%)
"	Mer8Wci3	.1435 (-3%)	.1741 (-2%)

Table 5: Small web component results - 50 queries

4.2 Comparison between Adhoc and Web runs

Table (6) compares Adhoc run (Mer8Adtd1) and Web run (Mer8Wctd). Both runs were based on Mercure pseudo-relevance back-propagation of the top 12 documents resulting from the initial Mercure search. The queries are built from the title and the description items. The results obtained using the Adhoc database are much better than those obtained using the Web database. A comparison with other participant runs will give some explanations for this difference.

Type	Run	average precision	Exact precision
T+D	Mer8Wctd : 12 rel, 0 nrel	.1638	.1957
T+D	Mer8Adtd1 : 12 rel, 0 nrel	.2231	.2786

Table 6: Comparison between adhoc and small web runs - 50 queries

5 Clir Experiment

This year is our first participation in cross-language track. It consists of three runs: two runs use English topics and retrieve documents from the pool of documents in all four languages (German, French, Italian and English), the other one uses French topics and retrieves documents from the pool of documents in all four languages. Our approach to the CLIR task is to translate the query from the source language into other languages using an online Machine Translation (MT). The runs were performed by doing individual runs for pair languages and merging the results to form the final ranked list. Before merging, the pseudo-relevance back-propagation method can be used. In this case, the top (12 in this experiment) documents resulting from the initial search by pair languages are assumed as relevant and thanks to the relevance backpropagation method, a new query is built for each language and applied to the index in the same language. The list of documents resulting from these searchs are then merged to form the final ranked list (shows figure 1).The merging consists of sorting the documents resulting from the pair language search in decreasing order of their activation values. Four index files were created, one index by language. English words are stemmed using Porter, French words using the truncature (8 first characters), no stemming for Italian and German words.

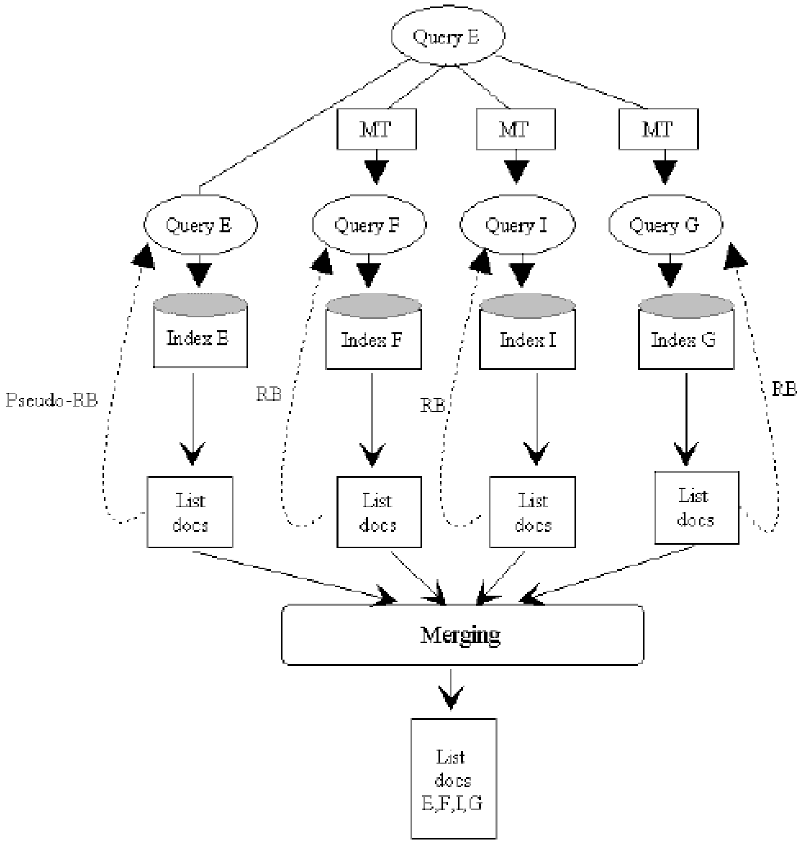

Figure 1: The cross-language methodology

Three runs were submitted.

- Mer8Can2x : query source in English + pseudo Relevance Backpropagation (RB)
- Mer8Cfr2x : query source in French + pseudo RB
- Mer8Can2x0 : query source in English

TREC results					
Run	Best	$\geq$median	$< median$	AvgP	R-prec
Mer8Can2x	4	10	18	0.2051	0.2386
Mer8Cfr2x	2	12	16	0.1620	0.2031
Mer8Can2x0	1	14	14	0.2059	0.2529

Table 7: Comparative automatic CLIR at average precision

Table 7 compares our runs against the published median runs. It can be seen that the run Mer8Can2x (English query source+RB) are better than Mer8Cfr2x (French query source+RB). The translation from English into the other languages seems to work better than from French. However, it can be seen that the pseudo relevance back-propagation does not bring anything (comparison between Mer8Can2x and Mer8Can2x0), the average precision even dropped.

6 Batch and Routing Experiments

The batch and routing experiments were performed using Mercure system. Three different learning techniques were tested to build the batch and the routing queries : the Relevance Backpropagation, the Gradient Backpropagation process presented in trec7 [3] and a new technique based on Genetic Algorithm (GA).

The FT92 documents were used as training data. The filtering algorithm starts with an initial query, built from all the parts of the topic, and its FT92 relevance judgments (positive and negative). The profiles were learned using the three techniques listed above. A pool of queries based on the three methods was then selected.

The GA we used to learn the profiles is not a classical GA. The GA operators we defined are based on some relevant information issued from the IR domain. The details of these operators can be found in [2].

6.1 Representation of IR for GA

The genetic algorithm attempts to make evolve, generation by generation, a population of queries towards those improving the outcome of the system. The competition starts between the various representations of the potential solutions the arbitrator of the competition is the fitness function.

6.1.1 Query individual and query population

The potential solution in our GA is a query. The initial set of queries (initial population) ($Pop^{(0)}$), contains the initial query and N given training documents. A subset of these documents can generate a different query. The query population is renewed at each iteration of the GA by applying selection, crossover and mutation operators.

6.1.2 Fitness function

A *fitness* is assigned to each query. This fitness represents the query effectiveness during the retrieving stage. It is computed according to the relevance of the retrieved documents. The formula used is:

$$Fitness(Q_u^{(s)}) = 1 - \frac{\sum_{i=1}^{R} log(rank_i) - \sum_{i=1}^{N} log(i)}{log(\frac{(R!)}{(N-R)!R!})} \qquad (1)$$

where $Q_u^{(s)}$ is the query u at the generation (s) of the GA, $\|D_r\|$ is the number of relevant documents retrieved through all generations, $\|D_{nr}\|$ the number of nonrelevant documents retrieved through all generations.

6.1.3 Genetic operators

Some genetic operators defined in this work are not pure genetic operators. They have been adapted to take advantage of some techniques developed in IR. For this reason, the defined operators are qualified as knowledge based operators.

Selection.

The selection procedure is based on the roulette wheel selection [6]. It consists in assigning to every individual of the population a number of copies in the next generation, proportional to the individual relative fitness. Then, one spin of the wheel selects a single individual each time until the population size is reached.

Crossover based on term weight

This crossover does not use a crossing point, it modifies the term weights according to their distribution in the relevant and nonrelevant documents. Let us consider $Q_u^{(s)}$ and $Q_v^{(s)}$ two individuals (queries) selected for crossover. The result is the new individual $Q_p^{(s)}$ defined as follows :

$$\left.\begin{array}{l} Q_u^{(s)}(q_{u1}^{(s)}, q_{u2}^{(s)}, ..., q_{uT}^{(s)}) \\ Q_v^{(s)}(q_{v1}^{(s)}, q_{v2}^{(s)}, ..., q_{vT}^{(s)}) \end{array}\right\} \longrightarrow Q_p^{(s+1)}(q_{p1}^{(s+1)}, q_{p2}^{(s+1)}, ..., q_{pT}^{(s+1)})$$

$$q_{pi}^{(s)} = \begin{cases} Max(q_{ui}^{(s)}, q_{vi}^{(s)}) \ if \ weight(t_i, D_r^{(s)}) \geq weight(t_i, D_{nr}^{(s)}) \\ Min(q_{ui}^{(s)}, q_{vi}^{(s)}) \ otherwise \end{cases}$$

Where the weight of term t_i in a set of documents D is defined by $weight(t_i, D) = \sum_{d_j \in D} w_{ji}$ and $D_r^{(s)}$ is the set of relevant documents retrieved by the population $Pop^{(s)}$ and $D_{nr}^{(s)}$ is set of nonrelevant documents retrieved by the $Pop^{(s)}$.

Mutation based on term relevance

The mutation operator explores the terms occurring in the relevant documents in order to adjust the corresponding gene values in the query selected for the mutation.
Let us consider $Q_u^{(s)}$ as the selected individual query and $L^{(s)}$ as the set of terms from $D_r^{(s)}$, the relevant documents retrieved in the previous generation of the GA. The mutation alters the genes of the selected individual on the basis of the $L^{(s)}$ terms and on the probability P_m. The $L^{(s)}$ terms are sorted according to a score value calculated as follows:

$$Score(t_i) = \frac{\sum_{d_j \in D_r^{(s)}} w_{ji}}{\|D_r^{(s)}\|}$$

This score is used to limit the number of terms that can be used for the mutation process. The mutation operation is done as follows :

for each term t_i *in* $L^{(s)}$
if (random(p) $< P_m$*) then* /* mutate the gene t_i
$q_{ui}^{(s)} = average(Q_i^{(s)}) - \delta$ /* modify the weight of t_i in $Q_u^{(s)}$
endif
endfor

Where P_m is the mutation probability, $random(p)$ generates a random number p in the range of [0..1]. The average function is computed as follows :

$$average(Q_u^{(s)}) = \frac{\sum_j^T q_{ui}^{(s)}}{n_{q_{ui}^{(s)}}}$$

where $n_{q_{ui}^{(s)}}$ is number of $q_{ui}^{(s)} \neq 0$ in $Q_u^{(s)}$. δ is a parameter used to control the average value (we used $\delta = 0$).

Learning algorithm

- Submit the initial query and do the search
- Build the initial population as follows : the top 10 relevant documents join the initial query in the set of initial individuals
- Compute the fitness of these individuals
- Apply the genetic operators to these individuals as follows :
    - Repeat
      Select two individuals, crossover then mutation
    - Until Size($Pop^{(s+1)}$)=*fixed population size* (the population is then built).
- Repeat
  For each query in the population do a search
  Compute the fitness of each query
  Apply the genetic operators
- Until a fixed number of iteration

6.2 Routing task

For the routing task, the queries having the best average precision in the training data were selected as routing queries. Two runs were submitted : Mer8R1 based on relevance backpropagation and Mer8R2 based on GA.

TREC Routing				
Run	Best	≥median	$< median$	AvgP
Mer8R1	3	14	34	0.271
Mer8R2	1	3	45	0.108

Table 8: Comparative routing results at average precision

Table 8 shows the comparative routing results at average precision. There was no gain from the GA-based run. The results obtained in TREC7 filtering database (AP collection) were much better than those obtained in the FT collection. This is probably due to this collection. In fact, a lot of queries has only few relevant documents in the part of the FT used as training data.

6.3 Batch Filtering

The profiles in the batch task are learned using the relevance backpropagation method and the gradient back-propagation. The pool of the queries contains the queries generated by both methods. The TREC standard output file of each query was analyzed to build an output file containing:

$$< topic >< func >< value >< thresh >< rank >< prec >< recall >< method >$$

As it has been done in [7]. The weights, corresponding to the document activation, which maximize the utility function were then found and selected as thresholds. Thus, the queries corresponding to these thresholds were selected and tested against the test data. The $< method >$ field correspond to the used method (relevance backpropagation or gradient backpropagation).

Two runs were submitted : Mer8BaLF1 based on utility-[$LF1$] and Mer8BaLF2 utility-[$LF2$].

TREC batch filtering					
Run	Best (with score=0)	$\geq$median	$< median$	$min = med = max = 0$	$score = 0$
Mer8BaLF1	17 (9)	38	7	5	22
Mer8BaLF2	15 (7)	33	10	7	23

Table 9: Comparative batch filtering results

Table 9 lists the comparative batch results. It can be seen that in both Mer8BaLF1 and Mer8BaLF2 runs the results are quite good, 17 best queries in the first run and 15 in the second run. It can be also noticed that 9 queries among the 17 best in the first run and 7 among the 15 best in the second run have a $score = 0$. It can be shown also that there are 22 queries with score zero (no documents have been submitted for these queries).

References

[1] M. Boughanem, C. Chrisment & C. Soule-Dupuy, *Query modification based on relevance backpropagation in Adhoc environment*, Information Processing and Managment. April 1999.

[2] M. Boughanem, C. Chrisment & L. Tamine, *Query space exploration based on GA*, Information Retrieval journal. October 1999.

[3] M. Boughanem, T. Dkaki, J. Mothe & C. Soule-Dupuy, *Mercure at trec7*, Proceedings of the 7th International Conference on Text REtrieval TREC7, E. M. Voorhees and Harman D.K. (Ed.), NIST SP 500-236, Nov. 1997.

[4] C. Buckley & al, *Query zoning : TREC'5*, Proceedings of the 5th International Conference on Text REtrieval TREC5, Harman D.K. (Ed.), NIST SP 500-236, Nov. 1996.

[5] C. Buckley & al, *SMART High Precision : TREC 7* , Proceedings of the 7th International Conference on Text REtrieval TREC 7, E. M. Voorhees and Harman D.K. (Ed.), NIST SP 500-236, Nov. 1997.

[6] D. E. Goldberg *Algorithmes génétiques. Exploration, optimisation et apprentissage automatique* Edition Addison-Wesley, France 1994.

[7] S. Robertson and al *Okapi at TREC-6*, Proceedings of the 6th International Conference on Text REtrieval TREC6, Harman D.K. (Ed.), NIST SP 500-236, Nov. 1997.