Mercure at trec9: Web and Filtering tasks

M. Abchiche, M. Boughanem, T. Dkaki, J. Mothe, C. Soule Dupuy, M. Tmar

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 TREC9 focus on the Web and Filtering (batch and routing) tracks. The submitted runs are based on the Mercure system. For one of the filtering routing runs, we combine Mercure with mining text functionnalities from our system Tétralogie. We also performed some experiments taking hyperlinks into account to evaluate their influence on the retrieval effectiveness, but no runs were sent.

Web: We submit four runs in this track. Two elements were tested: a modified Mercure term weighting scheme and the notion of the user preference on the retrieved document were tested.

Filtering (batch and routing): our main investigation this year concerns the notion of non-relevant profile in a filtering system. The filtering consists on, first filtering the documents using a relevant profile learned from relevant documents, second re-filtering the selected documents using non-relevant profile learned from non-relevant documents so that non-relevant documents accepted by the relevant profile are rejected. This notion of non-relevant profile was introduced by Hoashi [6] in an adaptive system whereas we use this technique for a batch system.

2 Mercure model

Mercure is an information retrieval system based on a connectionist approach and modeled 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 [4],[3].

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:

$$q_{ui}^{(s)} = \begin{cases} \frac{nq_u * qtf_{ui}}{nq_u - qtf_{ui}} \; if \; (nq_u > qtf_{ui}) \\ qtf_{ui} \; otherwise \end{cases} \tag{1}$$

Notations:
$q_{ui}^{(s)}$: the weight of the term t_i in the query u at the stage s,
qtf_{ui}: the query term frequency of t_i in q_u,
nq_u: query length (number of terms) of q_u,

- the term-document link weights are expressed by:

$$d_{ij} = \frac{tf_{ij} * (h_1 + h_2 * log(\frac{N}{n_i}))}{h_3 + h_4 * \frac{dl_j}{\Delta d} + h_5 * tf_{ij}} \tag{2}$$

Notations:
d_{ij}: term-document weight of term t_i and document d_j
tf_{ij}: the term frequency of t_i in the document D_j,
N: the total number of documents,
n_i: the number of documents containing term t_i,
h_1, h_2, h_3 and h_4: constant parameters,
Δ_d: average document length.

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 backpropagation. It consists in spreading backward the document relevance from the document layer to the query layer [3].

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: $In(t_i) = q_{ui}^{(}s)$ 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: $In(d_j) = \sum_{i=1}^{T} Out(t_i) * d_{ij}$ 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 backpropagation

The top retrieved documents are judged and a *relevance value* corresponding to the user preference 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 $rel_j = \frac{Coef\_rel}{Nb\_Nrel} for\ non-relevant\ document$

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, $In(t_i) = \sum_{j=1}^{N}(d_{ji} * Out(d_j))$ 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,
   d_{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,
   qtf: query term frequency,
   nq: query length (number of terms),

   M_a and M_b: tuned and determined by a series of experiments and set to $M_a = 2$ and $M_b = 0.75$.

   $Coef\_rel$ ($Coef\_Nrel$): user preference positive value for relevant document and relevant value for negative document.

3 Web Track Experiment

3.1 Web methodology

We tested the relevance backpropagation strategy using a user preference. In fact, we consider that the relevance of the documents are not all the same, but depends on the user satisfaction. In Mercure system, the user satisfaction is represented by the $coef\_rel$ parameter assigned by the user. Because the user does not judge the document, in the pseudo relevance back-propagation, the top retrieved documents are assumed as relevant. The "user" preference is then assigned to a document according to its rank in the retrieved set. For the trec9 experiment, the top 12 documents were assumed to be relevant. The user preference is computed as follows :

- $coef\_rel = 1$ for the documents ranked from 1 to 5

- $coef\_rel = .75$ for the documents ranked from 6 to 10

- $coef\_rel = .5$ for the documents ranked from 11 to 12

3.2 Results

Four runs based on content only were submitted:

1. Mer9Wt0: simple search using the title field
2. Mer9Wt1: title field + pseudo-relevance feedback based on Mercure relevance backpropagation. The top 12 retrieved documents are assumed to be relevant and the $coef\_rel = 1$
3. Mer9WtMr: title field + pseudo-relevance feedback based on Mercure relevance backpropagation. The top 12 retrieved documents are assumed to be relevant and the $coef\_rel$ are computed using the user preferences as described above.
4. Mer9Wtnd: simple search using the title+Description+Narrative fields.

Unfortunately, the official results were wrong because of a mistake in our indexing script. This explains in part the bad results obtained this year by Mercure comparing to those obtained in the previous TREC. This will be modified and the corrected results will be integrated in the final paper. Consequently to the problem encountered with the indexing process, the experiments carried out by taking hyperlinks into account could not be done in time and will be included in the final paper too.

Type	Run	average precision	Exact precision
Mer9Wt0	title simple search	0.0996	0.1274
Mer9Wt1	title +simple pseudo-rb	0.0114	0.0242
Mer9WtMr	title+simple pseudo-rb basen user preference	0.0154	0.0307
Mer9WtMr	TDN fields +simple serach	0.0140	0.0295

Table 1: Web component results - 50 queries

4 Batch and Routing Experiments

The batch and routing experiments were performed using Mercure system. The profiles were learned using the same learning algorithm as before: the relevance backpropagation. The relevance value assigned to each document was used as user preference (2 and 1). It corresponds to $Coef\_rel$ in the relevance backpropagation algorithm.
The filtering algorithm starts with an initial query, built from all the topic parts, and its OHSU87 relevance judgements. A pool of queries based on the learning method was then selected. The OHSU88-91 documents were used as test data.

4.1 Batch Filtering

The profiles in the batch task are learned using the relevance backpropagation method. Two techniques are tested to be compared. The first one corresponds to filtering documents using learned (positive, relevant) profiles. The second represents filtering documents using relevant and non-relevant (negative) profiles. These methods are detailed below. We use the phrase *"positive profiles"* for relevant learned profiles, and *"negative profiles"* for non-relevant profiles.
The TREC standard output file of each query was analyzed to build an output file containing:

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

In this section, we detail both the batch filtering methods and the results.

4.1.1 Batch filtering using positive profiles

As it has been done in [7]. The document activation weights which maximizes the utility function were found and selected as thresholds. Then the queries corresponding to these thresholds were selected and tested on a set of test documents. The documents weighted by values higher than the threshold were selected for the corresponding query.
In the first run (mer9b1), we build profiles by learning using relevance backpropagation on the training dataset, then we apply them on the test dataset.
The following algorithm is used:

For each profile P

1. evaluate P on the training dataset
2. select top 1000, let $result_0$ be the obtained document ranked list
3. $i = 1$
4. repeat
   (a) build a new profile P_i by relevance backpropagation
   (b) evaluate P_i on the training dataset
   (c) select top 1000, let $result_i$ be the obtained document ranked list
   (d) inc i

until $i = max\_iteration$

5. for each $r \in \{1 \ldots 1000\}$, $i \in \{0, 1, \ldots max\_iteration\}$
   (a) $result_{ir}$ contains top r documents from $result_i$
   (b) evaluate $result_{ir}$ on $utility\_[T9U]$
6. select profile P_i such as $\exists r \in \{0, 1 \ldots max\_iteration\}$ where $result_{ir}$ gives the best $utility\_[T9U]$ value
7. apply P_i on the test dataset, $test\_result_i$ is the obtained document ranked list
8. *submitted* contains documents in $test\_result_i$ having a rsv at least equal to the rsv of the r^{th} document
9. submit the selected list *submitted*

In the experiments, we carried out relevance backpropagation twice. We found that this number of iterations was enough to learn the profile.
We computed the utility on the top 1000 documents only as this set is likely to contain most of the relevant documents.

4.1.2 Batch filtering using positive and negative profiles

The second run (mer9b2) is based on negative profile building. We built -in addition to the positive profile- a negative profile and applied both positive and negative profiles on the test dataset as detailed below.
A negative profile is a profile aiming at excluding non-relevant documents from the top ranked ones. A document is filtered when it is potentially relevant compared to the positive profile, and non-relevant compared to the negative profile. Generally, negative profile is not provided by the user, but the system can build it by learning. It is built starting from the non-relevant selected documents.
In our experiments, negative profiles are built by relevance backpropagation. We select the top 50 non-relevant documents from the filtered ones resulting from training documents evaluation. These documents are used as relevant documents in a relevance backpropagation process, the resulting learned profile is considered as the negative profile. As there is no initial negative profile, we use different values of Ma and Mb: we assign 0 to Ma and 1 to Mb and thus, the weight assigned to each term in the negative profile is its output activation done by this relevance backpropagation process.
Two values are then learned using the training data set, a and b, used in filtering documents using positive and negative profiles. a is a multiplicative value of the relevance threshold, it allows to select more documents, so it is less than 1. b allows to select documents which [positive profile rsv/negative profile rsv] ratio is higher than b, so it defines [negative rsv/positive rsv] ratio threshold. Batch filtering document process using negative profiles is described as follows:

For each document D_i

1. compute an rsv value for the positive profile: $rsv_p(D_i)$

2. if $rsv_p(D_i) > a * threshold$

   (a) compute an rsv value to the negative profile: $rsv_n(D_i)$

   (b) if $rsv_p(D_i) > b * rsv_n(D_i)$ filter document D_i else reject document D_i

   else reject document D_i

We varied a in an interval raised by 1 and b in an interval undervalued by 1, and retained the values giving the best utility-[$T9U$] value. a should be less than 1 in order to select more documents, and b is more than 1 to filter documents which are most likely to be relevant to the positive profile and non-relevant to the negative profile. This algorithm was processed on the test documents resulting from mer9b1 run. Note that if $a = 1$ and $b = 0$, $mer9b1$ and $mer9b2$ are the same.
In our experiments, we selected as positive profiles the profiles applied to $mer9b1$ run, and carried out relevance backpropagation using top 50 non-relevant documents from the document ranked list resulting from evaluating these profiles on the training dataset. a and b values were then learned on the dataset. Finally, positive, negative profiles, rsv threshold, a and b were applied on the test dataset.

4.1.3 Batch filtering results

Table 2 lists the comparative batch results on utility-[$T9U$].

TREC batch filtering			
Run	$\geq median$	$= max$	$score = 0$
mer9b1	38	10	5
mer9b2	22	5	0

Table 2: Comparative batch filtering results

It can be seen that in both mer9b1 and mer9b2 runs the results are quite good, 13 best queries in the first run and 7 in the second run. For 5 queries, no documents were retrieved using positive profiles only, whereas documents are retrieved for these queries using positive and negative profiles.
Batch experiments have been performed to show the effectiveness of using negative profiles. Despite better results using positive profiles only, the first results using negative profiles are promising. Using negative profiles on the training documents improve results for many values of a and b. The best values were chosen and applied on the test documents. Future work will be devoted to taking into account other parameters in order to obtain significant results when using negative profiles.

4.2 Routing task

We experiment two methods in the routing track:

- using a similar method then in the batch filtering track

- using the results of a factorial correspondence analysis applied to a sub-collection

In the first method, the queries having the best average precision in the training dataset were selected as routing queries, and are applied on the test documents.
The second method we experiment expands each query using the training document set before it is sent to Mercure. To expand the query, we first performed a search on the training document set and selected the top ranked documents. Then a list of the most representative terms -either single terms or phrases- is extracted from this sub-collections. Once selected, the list of terms is used to build a crossing matrix that crosses the terms and the documents: $M = [g_i * l_{ij}]$ where l_{ij} is a local weight that reflect the importance of term i within document j and g_i is a global weight that reflect the importance of term i within the sub-collection of documents. A Correspondence Factorial Analysis (CFA) [1][2] is then performed on the matrix and the relevant factorial axes are selected. A factorial axis is considered as relevant if it contributes to relevant documents. The terms are then re-ranked according to their contribution to the selected factorial axes. The top ranked terms are used to expand the query after altering their weight using the cosines of the angles between them and the relevant documents within the factorial space.
The main purpose of this method is to evaluate the use of the CFA to find out the dimensions -parts- of factorial space that are characterized by relevant documents and then to find what terms contribute most to those dimensions.

4.2.1 Extraction of the sub-collection

CFA is about orthogonal factorization and singular value decomposition of a square matrix which dimension is the collection document number. Almost all the algorithms used to achieve these factorization and decomposition are iterative and very time consuming. The overall time they take mainly depends on the matrix dimension. For this reason we chose to perform the analysis on a sub-collection rather than on the entire training collection. The sub-collection contains top ranked documents returned by a first search using the *Vigie*[5] system. This system includes the stop word removal and the Porter stemming.

4.2.2 Items extraction

In order to select terms, we use the term weight presented in the OKAPI system [8]
$w_{ij} = \frac{tf*log(\frac{N-n+0.5}{n+0.5})}{2*(0.25+0.75*(\frac{dl}{avdl})+tf}$
where:
N is the collection document number
n is the number of documents containing the term i
tf is the term frequency within the document j
dl is the document length -size-
avdl is the average document size
the weights are calculated for each pair of term/document and each term is associated to its second highest associated weight so that we can rank the terms and select the top of them. This means that the selected terms are representative of at least two documents.

4.2.3 Terms re-ranking strategy and selection

The principal goal of the method is to expand the topic -query- by adding new terms and modifying the weights of the existing ones. The terms added are the ones that contribute the most to the factorial axes related to relevant documents. In other words the axes to which the relevant documents contribute highly. The top ranked terms are used to expand the query. Using this method, a term can be added more than once. The new query is constituted of new terms and already existing terms.
The weight of new terms is calculated as bellow: $w_i = K * p_i * Max_j cos(T_i, D_j)$
The weight of the already existing is calculated as bellow:
$w_i = w_{i,old} * K * p_i * Max_j cos(T_i, D_j)$
K is a constant that depends on the number of terms and documents,
p_i is the number of time term T_i is selected to added to the topic,
$w_{i,old}$ is the initial weight of term T_i,
(T_i, D_j) is the angle formed by relevant document D_j and term T_i within the factorial space.

4.2.4 Routing results

Table 3 shows the routing results at average precision.

TREC Routing			
run	$= max$	$\geq median$	AvgP
mer9r1	6	38	0.235
mer9r2	0	20	0.185

Table 3: Comparative routing results at average precision

In average, mer9r1 obtained better precision results than mer9r2.

The average precision obtained in mer9r1 is slightly lower than the average precision obtained in TREC8. It can be seen that using the relevance values as user preferences improves routing results. Thus results could probably be improved more by assigning different relevance values depending on documents.
The results of 10 of the queries that obtained equal or better results than the average using the second method (run mer9r2) are higher than the one of run mer9r1. The combination of the two methods has to be explored further.

References

[1] J. P. Benzécri *Correspondence Analysis Handbook*, Marcel Dekker Ed., New York, 1992.

[2] M. W. Berry, Z. Drmac, E. R. Jessup *Matrices, Vector Sparces, and Information Retrieval* In SIAM Review Vol. 41. No 2, pp 335-362, 1999.

[3] M. Boughanem, C. Chrisment & C. Soule-Dupuy, *Query modification based on relevance backpropagation in Adhoc environment*, Information Processing and Management, 35(1999)121-139 April 1999.

[4] 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, November 1998.

[5] T. Dkaki, B. Dousset, J. Mothe *Analyse d'informations issues du Web avec Tétralogie* In Proceedings of the Veille Stratégique scientifique et technologique Conférence, pp 159-170, 1998.

[6] K. Hoashi, K. Matsumoto, N. Inoue, K. Hashimoto *Document filtering method using non-relevant information profile* Proceedings of the ACM/SIGIR, Athens,Greece July 2000

[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, November 1997.

[8] S. E. Robertson, S. Walker *Okapi/Keenbow at TREC-8* In Proceedings of the TREC 8 Conference, National Institute of Standards and Technology, 1999.