Mercure and MercureFiltre applied for Web and Filtering tasks at TREC-10

M. Boughanem, C. Chrisment, 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 TREC-10 focus on the Filtering (adaptive, batch and routing) tracks and web tracks. The runs are based on Mercure system for web, routing and batch tracks, and MercureFiltre for adaptive track.

2 Mercure model

Mercure is an information retrieval system based on a connexionist 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 [2] [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 [5] 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:

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

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, h_4 and h_5: 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 back-propagation. It consists in spreading backward the document relevance from the document layer to the query layer [2].

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, $Output_u\,(Out\,(d_1)\,, Out\,(d_2)\,, \ldots, Out\,(d_N))$ 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* corresponding to the user preference is assigned to each document (positive for relevant documents, negative for non-relevant documents and null for non-judged documents). These values are used to compute the *DesiredOutput* vector.

$$DesiredOutput = (rel_1, rel_2, \ldots, rel_N) \begin{cases} rel_j = \frac{Coef\_rel}{Nb\_rel}\; if\ D_j\ is\ relevant \\ rel_j = \frac{Coef\_Nrel}{Nb\_Nrel} otherwise \end{cases} \tag{3}$$

1. This output is considered as an "input" in the back-spreading process and is presented to the document layer. Each document node computes an input value, $In\left(d_j\right) = rel_j$ and then an activation signal, $Out\left(d_j\right) = g\left(In\left(d_j\right)\right)$.

2. This activation is back-spread to the term layer. Each term node computes an input value, $In\left(t_i\right) = \sum_{j=1}^{N}\left(d_{ji} * Out\left(d_j\right)\right)$ and then an output signal, $Out\left(t_i\right) = g\left(In\left(t_i\right)\right)$.

3. Finally, the new query-term links corresponding to the new query are computed as follows: $Q_u^{(s+1)} = \left(q_{u1}^{(s+1)}, q_{u2}^{(s+1)}, \ldots, q_{uT}^{(s+1)}\right)$ with $q_{ui}^{(s+1)} = M_a * q_{ui}^{(s)} + M_b * Out\left(t_i\right)$

   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),
   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 negative value for non relevant document.

3 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 back-propagation. The relevance value assigned to each document was used as user judgement. It corresponds to $Coef\_rel$ in the relevance back-propagation algorithm.
The filtering algorithm starts with an initial query, built from all the topic parts, and its relevance judgements corresponding to all documents with items down to 26150. A pool of queries based on the learning method was then selected. All the remaining documents (with items up to and including 26150) were used as test data.

3.1 Batch Filtering

The profiles in the batch task are learned using the relevance back-propagation method.
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 [4]. 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.
We build profiles using relevance back-propagation 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 10000, let $result_0$ be the obtained document ranked list
3. $i = 1$
4. repeat
   (a) build a new profile P_i by relevance back-propagation
   (b) evaluate P_i on the training dataset
   (c) select top 10000, let $result_i$ be the obtained document ranked list
   (d) inc i

   until $i = max\_iteration$
5. for each $r \in \{1 \ldots 10000\}$, $i \in \{0, 1, \ldots max\_iteration\}$
   (a) $result_{ir}$ contains top r documents from $result_i$
   (b) evaluate $result_{ir}$ using $T9U$ utility
6. select profile P_i such as $\exists r \in \{0, 1 \ldots 10000\}$ where $result_{ir}$ gives the best (max) $T9U$
7. apply P_i on the test dataset, let $test\_result_i$ be the obtained document ranked list
8. select documents in $test\_result_i$ having their rsv at least equal to the rsv of the r^{th} document
9. submit this list

In the experiments, we carried out relevance twice. We found that this number of iterations was enough to learn the profile.
We computed the utility on the top 10000 documents only as this set is likely to contain most of the relevant documents.

3.1.1 Batch filtering results

Table 1 lists the comparative batch results.

TREC batch filtering				
Evaluation measure	$= max$	$\geq median$	$< median$	Avg
F-Beta	1	24	59	0.392
SetPrecision	10	36	38	0.631
SetRecall	1	20	63	0.204
T10SU	0	28	56	0.181

Table 1: Comparative batch filtering results

3.2 Routing track

3.2.1 Description of the experiment

We experiment routing using a similar method then in the batch filtering track, the queries having the best average precision in the training dataset were selected as routing queries, and applied on the test documents.

3.2.2 Routing results

Table 2 shows the routing results at average uninterpolated precision.

TREC Routing				
$= max$	$\geq median$	$< median$	$AvgP$	Precision at 1000
2	47	35	0.092	0.5594

Table 2: Comparative routing results at average uninterpolated precision

The average number of relevant documents per profile is largely great than 1000. The evaluation function used this year is the average uninterpolated precision, it consists on computing the precision at each relevant document at its position in the ranked list, adding these numbers up and divide by the total number of relevant documents. Relevant documents which do not appear in the top 1000 receive a precision score of zero. Non relevant documents appearing in the top of the ranked have a large influence on the average uninterpolated precision.

4 Adaptive track

We experiment this year an adaptive filtering process [1] using MercureFiltre. MercureFiltre is inspired from Mercure, It includes three elements: the profile, the document and the dissemination threshold. The profile and the document are represented by a set of weighted terms. The adaptation concerns the profile and the dissemination threshold.

4.1 System initialisation

The user profile is represented as follows:

$$p^{(0)} = \left(\left(tp_1, w_1^{(0)}\right), \left(tp_2, w_2^{(0)}\right) \ldots \left(tp_n, w_n^{(0)}\right)\right) \quad (4)$$

where tp_i is a term and $w_i^{(0)}$ is its weight in the initial profile (at $t = 0$), the term-profile weight is noted $w_i^{(t)}$ where t represents the instant when the system makes the last profile update. Initially, the term-profile weight is computed as follows:

$$w_i^{(0)} = \frac{tfp_i}{max_j\left(tfp_j\right)} \quad (5)$$

The formula 5 seems to be abusively simplistic, but at the beginning of the filtering process, no information is known but a set of terms and their occurrences in the initial user profile. However, this weight will be adjusted by learning.

4.1.1 Filtering incoming documents

Each term belonging to a document arriving at time t is weighted as follows:

$$d_i^{(t)} = \frac{tf_i^{(t)}}{h_3 + h_4 * \frac{dl^{(t)}}{\Delta l^{(t)}} + tf_i^{(t)}} * log\left(\frac{N^{(t)}}{n_i^{(t)}} + 1\right) \quad (6)$$

Notations:
$tf_i^{(t)}$: appearance frequency of the term t_i in the document $d^{(t)}$
h_3, h_4: constant parameters, for the experiments $h_3 = 0.2$ and $h_4 = 0.7$
$dl^{(t)}$: document length of $d^{(t)}$ (number of terms)
$\Delta l^{(t)}$: average document length
$N^{(t)}$: number of incoming documents until time t
$n_i^{(t)}$: number of incoming documents containing the term t_i

The weighting formula 6 is a form of Okapi term-profile weight used in Mercure.
$N^{(t)}$, $n_i^{(t)}$ and $\Delta l^{(t)}$ are collection based parameters that are computed on the cumulative documents filtered at t.

A relevance value noted $rsv\left(d^{(t)}, p^{(t)}\right)$ is then computed corresponding to the document and the profile, as follows:

$$rsv\left(d^{(t)}, p^{(t)}\right) = \sum_{t_i \in d^{(t)}, tp_j \in p^{(t)} \ and \ t_i = tp_j} d_i^{(t)} * w_j^{(t)} \quad (7)$$

The binary decision rule used for document filtering is the following:

$$\begin{cases} if\ rsv\left(d^{(t)}, p^{(t)}\right) \geq threshold^{(t)}\ accept\ the\ document \\ otherwise\ reject\ the\ document \end{cases}$$

4.1.2 Adaptive learning

The learning process is processed at each selected relevant document. Learning consists on adapting the user representation: update term-profile weight, add new terms and remove some non significant terms.
We assume that an interesting term must appear frequently in relevant documents, and appear a little in non relevant documents. When a relevant document is selected, each term-profile weight increases the more:

- it appears frequently in this document
- it appears in many relevant documents ($\frac{r_i^{(t)}}{R^{(t)}}$ is near 1)
- it appears in a few non relevant documents ($\frac{s_i^{(t)}}{N^{(t)}}$ is near 0)

Notations:
$r_i^{(t)}$: the number of relevant documents containing the term t_i until the time t
$R^{(t)}$: the total number of relevant documents until the time t
$s_i^{(t)}$ the number of non relevant documents containing the term t_i until the time t.

The learning process we adopted is a kind of relevance reinforcement process. The problem is assimilated to a linear equation resolution. The equation to resolve is to force obtaining a rsv value β to a relevant document. Each solution of this system is a set of weights affected to the document terms. The constraints are that the weight of each term in the profile divided by its importance according to the same profile is constant. The system is then the following:

$$\begin{cases} \sum_{t_i \in d^{(t)}, tp_i \in p^{(t)} \ and \ t_i = tp_j} d_i^{(t)} * w_j^{(t)} = \beta \\ \frac{w_i^{(t)}}{f\left(d_i^{(t)}, r_i^{(t)}, s_i^{(t)}\right)} = \frac{w_j^{(t)}}{f\left(d_j^{(t)}, r_j^{(t)}, s_j^{(t)}\right)} \ \forall \left(t_i, t_j\right) \in {d^{(t)}}^2 \end{cases} \tag{8}$$

However, the rsv depends on the document length. Let α be the rsv value wished for a relevant document with length dl_m, α and dl_m are constants. When $dl^{(t)}$ increases β increases but $\frac{\beta}{dl^{(t)}}$ decreases, so β is logistically proportional to the document length:

$$\beta = \frac{\alpha}{log\left(dl_m\right)} * log\left(dl^{(t)}\right) \tag{9}$$

For these experiments, $\alpha = 20$ and $dl_m = 100$.
The solution of the equation system 8 is a set of "provisional" weights, for each term appearing in the document, the provisional weight solution of the system 8 is the following:

$$\forall i, \ pw_i^{(t)} = \frac{\beta f\left(d_i^{(t)}, r_i^{(t)}, s_i^{(t)}\right)}{\sum_j f\left(d_j^{(t)}, r_j^{(t)}, s_j^{(t)}\right) * d_j^{(t)}} \tag{10}$$

The function f is proportional to the term importance. The function f used for these experiments is the following:

$$\forall i, f\left(d_i^{(t)}, r_i^{(t)}, s_i^{(t)}\right) = d_i^{(t)} * \frac{exp\left(\gamma * \frac{r_i^{(t)}}{R^{(t)}}\right) * \left(1 - exp\left(\gamma\left(\frac{s_i^{(t)}}{S^{(t)}} - 1\right)\right)\right)}{exp\left(\gamma\right)} \tag{11}$$

where $R^{(t)}$ is the number of relevant documents and $S^{(t)}$ is the number of non relevant documents until the time t. $gamma$ is set to 3.

The "provisional" weight $pw_i^{(t)}$ contributes in learning the term-profile weight corresponding to the term t_i, we use the following gradient propagation formula:

$$w_i^{(t+1)} = w_i^{(t)} + log\left(1 + pw_i^{(t)}\right) \tag{12}$$

We add 1 to $pw_i^{(t)}$ to avoid adding negative value to the term-profile weight.

4.1.3 Threshold setting and updating

For each topic, the dissemination threshold should be set and updated [6]. To set the intial threshold, we consider for this experiment that first 40 incoming documents are non relevant, the threshold is initialized with the average of the top 10 rsv values computed for these documents.

Threshold updating is made periodically at each 10 selected documents as follows:

$$\begin{aligned}
Threshold^{(t+1)} &= Threshold^{(t)} * A_1 * A_2 * A_3 && (13)\\
A_1 &= -exp\left(-\alpha_c \frac{S^{(t+1)}}{N^{(t+1)}}\right) + 2 && (14)\\
A_2 &= \frac{1}{\beta_c \frac{R^{(t+1)}}{R^{(t+1)} + S^{(t+1)}}} + 1 && (15)\\
A_3 &= exp\left(-\frac{NT^{(t+1)}}{\gamma_c}\right) + 1 && (16)
\end{aligned}$$

Notations:
α_c, β_c and γ_c are corrector parameters,
$R^{(t+1)}$ is the number of selected relevant documents between instants t and $t+1$,
$S^{(t+1)}$ is the number of selected non relevant documents between instants t and $t+1$,
$N^{(t+1)}$ is the number of incoming documents between instants t and $t+1$,
$NT^{(t+1)}$ is the number of all incoming documents until instant $t+1$.

A_1 enables to increase the threshold if the proportion of selected non relevant documents ($\frac{S}{N}$) is high, A_2 enables to decrease the threshold if the proportion of relevant rejected documents is assumed to be relatively high ($\frac{R}{R+S}$), A_3 enables to increase the threshold less quikly when the number of evaluated documents (N) is assumed to be enough for learning.

4.1.4 Adaptive results

Table 3 lists the adaptive results.

TREC adaptive filtering	
Evaluation	Avg
F-Beta	0.0192
SetPrecision	0.3865
SetRcall	0.0196
T10SU	0.0297

Table 3: Adaptive filtering results

Unfortunately, we have had no time to submit our results.

5 Web Track Experiment

This year experiment is based on Mercure sample search. Two runs were submitted to be integrated on the pool:

1. Merxtd: simple search using the title and the descirption fields
2. Merxt: simple search using the title field

Table 4 shows the comparative web results:

Type	Run	average precision	=max	$\geq$ median	$<$ median
Merxtd	title + description simple search	0.1729	2	18	30
Merxt	title simple search	0.1438	0	26	24

Table 4: Web results - 50 queries

References

[1] N. J. Belkin, W. B. Croft, *Information retrieval and information filtering: two sides of the same coin?*, Communications of the ACM 35(12), pages 29-38, 1992.

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

[3] M. Boughanem, T. Dkaki, J. Mothe & C. Soule-Dupuy, *Mercure at TREC-7*, Proceedings of the 7th International Conference on Text REtrieval TREC-7, E. M. Voorhees and D.K. Harman (Ed.), NIST SP 500-242, pages 413-418, November 1998.

[4] S. Walker, S. E. Robertson, M. Boughanem, G. J. F. Jones, K. Sparck Jones, *Okapi at TREC-6 automatic and ad hoc, VLC, routing, filtering and QSDR*, Proceedings of the 6th International Conference on Text REtrieval TREC-6, D.K. Harman (Ed.), NIST SP 500-240, pages 125-136, November 1997.

[5] S. E. Robertson, S. Walker *Okapi/Keenbow at TREC-8* In Proceedings of the TREC-8 Conference, National Institute of Standards and Technology, pages 151-161, November 2000.

[6] S. E. Robertson, S. Walker, *Threshold setting in adaptive filtering*, Journal of documentation 56, pages 312-331, 2000.