







$$\begin{aligned} \text{simError}_f(q, c_i, c_j) &= \text{sim}_f(q, c_i) - \text{sim}_f(q, c_j) \\ \text{rankError}(c_i, c_j) &= \text{rank}(c_i) - \text{rank}(c_j). \end{aligned}$$

A pseudo-code description of the algorithm is provided in Figure 6 and Figure 7. The learning rate  $\alpha$  is a small positive value that regulates the changes made to the weights. Usually the learning rate is adjusted empirically by conducting trials on a representative domain case base. We found in our experiments that a small value should be selected to prevent the algorithm from bouncing back and forth over a local optimal set of weights. A scaling factor, between 0 and 1, is used to decrease the learning rate when weight updates fails to reduce the error rate.

## Learning from Relevance Feedback

We describe in this section how to adjust the weights of the CBR system when a user provides some feedback on the relevance of some cases.

### Relevance Feedback

The idea of relevance feedback is to get the user to indicate that some cases should either be present or not present in the retrieval results. In general, relevance should indicate if a case meets some user's information needs. However, in practice, a case might be judged irrelevant for one of the following reasons:

- The value of one important feature is not acceptable.
- Some combinations of feature values do not go well together.
- The solution of a case is not useful.

With respect to weight optimization, the reasons a) and b) indicate that more weight should be assigned to the features used to determine the relevance of a case. However reason c) is more complex as it relates to the utility of a case. While we expect case utility to be proportional to case similarity, it cannot be fully guaranteed in practice. Hence, relevance feedback should work well if the feedback given by user is solely based on the features of the case problems.

### Rocchio Update

The field of Information Retrieval (IR) [Manning et al. 2009] has studied for many years how to cope with relevance in retrieval systems. Their main intuition was that the keywords of a query should be assigned a higher weight if they appear in relevant documents. Conversely, they should be penalized if they appear frequently in non-relevant documents.

We adapted this intuition to case-based reasoning as follows: features assigning higher similarity values to relevant cases should see their weight increased while

weights of other features should be decreased. Extending the Rocchio equation proposed in the IR literature, we propose the following update function to adjust weights based on relevance feedback:

$$w'_f = w_f + \frac{\beta}{|cases_{REL}|} \sum_{c_i \in cases_{REL}} \text{sim}_f(q, c_i) - \frac{\gamma}{|cases_{SNR}|} \sum_{c_j \in cases_{SNR}} \text{sim}_f(q, c_j)$$

where  $cases_{REL}$  is the group of relevant cases and  $cases_{SNR}$  is the group of non-relevant cases. This relevance feedback update function has the effect of modifying the feature weights so that the query problem  $q$  is moved closer to the centroid of the relevant cases and moved away from the centroid of the non-relevant cases.

The parameters  $\beta$  and  $\gamma$  determine the relative important of relevant cases to irrelevant ones. In practice, values such as  $\beta = 0.75$  and  $\gamma = 0.25$  are frequently used. But we recommend determining them empirically on a representative data set.

It is important to mention that this approach does not require an iterative algorithm as the learning algorithm presented for coping with ranking feedback. Given some retrieval results and a relevance feedback, the update function is applied only once to each feature weight.

### Pseudo-ranking Approach

A relevance feedback suggests some observations on the desired ranking of the cases. First, we would expect the group of relevant cases to be returned at the top of the case ranking. For instance, in our example presented in Figure 8, relevant cases  $c_1$  and  $c_5$  should be the most highly ranked cases. Even if ranked first, it is impossible to determine a strict total ordering among the relevant cases solely based on their relevance value. Hence we considered that they should all have the same rank (rank 1).

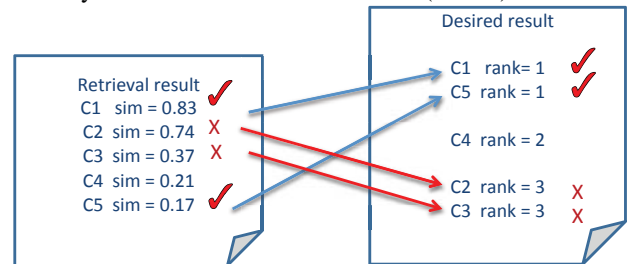


Figure 8: Relevance feedback as a pseudo-ranking

In contrast, non-relevant cases should have lower similarities than those of the relevant group and be located at the bottom of the case ranking. For instance,  $c_2$  and  $c_3$  should not precede  $c_1$  and  $c_5$ .

This leaves the remaining group of cases for which we do not have a relevance feedback ( $c_4$  in our example). One option would be to ignore these “unknown” cases during the learning process. However we believe that some

valuable information resides in the fact that these cases are neither relevant nor non-relevant. We assume that the similarity of the “*unknown*” cases should be lower than those of the relevant group and higher than those of the non-relevant. This implies that they should be ranked as an intermediate group in-between relevant and irrelevant case.

Given that a relevance feedback can be interpreted as a pseudo-ranking of three groups of cases (*relevant*, *unknown* and *irrelevant*), it becomes possible to apply a gradient descent algorithm to learn weights from relevance feedback. However, to do so, we need to refine the notion of error. A *pseudo-ranking error* occurs if:

- Two cases do not have the same relevance feedback;
- And the case with lower relevance has a higher similarity value than the other.

### Learning with Combined Feedback

To learn weights when both types of feedback are provided by the user, we simply adopt a cascade of learners where:

- Cases are ranked first based on the ranking feedback. Gradient descent is applied to establish this ranking.
- The resulting weights are then updated using the modified Rocchio update function to take into account the relevance of the cases.

This first step of this scheme ensures that a suitable set of weights is selected to respect the relative ranking desired by the user. This approach also offers the advantage that a pseudo-ranking optimization can be conducted if the user only provides some relevance feedback.

The second step, where a relevance feedback update is applied to the ranking weights, aims to reduce further the global similarity of the non-relevant cases so that it would leave some opportunity to insert new cases in the retrieval results during successive search over the entire case base.

### Empirical Study

To study the behavior of the proposed weight-learning scheme, we conducted experiments in an interactive setting to evaluate how user feedback were translated into feature weights. The training sessions made use of the travel case base<sup>1</sup>, a data set containing an interesting variety of feature types (numerical, categorical, ontological...). The training scheme and the learning algorithms described in the paper were implemented in Java. We also made use of jCOLIBRI [Recio-Garcia et al. 2008] as a CBR framework to perform case retrieval and to obtain the retrieval results needed by the weight learning algorithms.

<sup>1</sup> The travel case base contains 1024 cases defined with 7 problem features (holiday type, number of persons, region, transportation mean, duration, season, accommodation) and 2 solution features (hotel & price). This case base is available at <http://www.cs.auckland.ac.nz/~ian/GBR/>.

Each of our training sessions followed this procedure:

1. The human trainer selects some features to determine the relevance and ranking of the cases.
2. The trainer submits a query to the CBR system and analyzes the retrieval results. In our experiments, the 5 top-ranked cases were presented.
3. The trainer formulates relevance and ranking feedback based on the features selected in step 1.
4. The learning scheme updates the weights and stored them in the similarity configuration to be reused for other training episodes. In our experiments, we used a learning rate of 0.1, a scaling factor of 0.5,  $\beta = 0.8$  and  $\gamma = 0.1$ .
5. Steps 2-4 are repeated with new queries until the ranking and relevance of the results are deemed satisfactory by the trainer.

The results presented in this section are the average values obtained over 5 different training sessions.

In our first experiment, we use a single feature (*holiday type*) to determine the relevance of a case. To be relevant, a case must have the same feature value as the query. We also use another feature (*season*) to establish the relative ranking of the cases. A ranking feedback corresponds to the cases ranked in decreasing order of similarity for this feature. The training results, presented in Figure 9, clearly indicates that the learning scheme recognizes, within 2 or 3 user queries, the two features used to formulate the feedback. It is interesting to observe that most of the weight is assigned to the feature associated to relevance (*holiday type*). We also notice that the feature assigned to ranking feedback (*season*), is assigned sufficient weight by the learning scheme to act as a soft constraint.

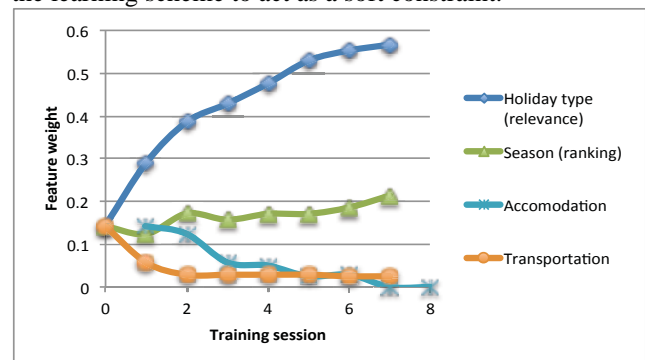


Figure 9: Weight learning results with relevance and ranking feedback determined using one feature each.

We repeated the same experiment using two features to determine the relevance of a case (*holiday type* and *season*) and one feature to establish case ranking (*accommodation*). Again the features used to establish case relevance were quickly recognized through learning and are assigned most of the global weight (see Figure 10). In practice, most of the recommendations become relevant after a few training

episodes. We also observed that the learning scheme assigns sufficient weight to the ranking feature (*accommodation*) to get a consistent case ordering in most situations. However we noticed during our training sessions that the gradient learner sometimes had difficulty to perform its optimization satisfactorily.

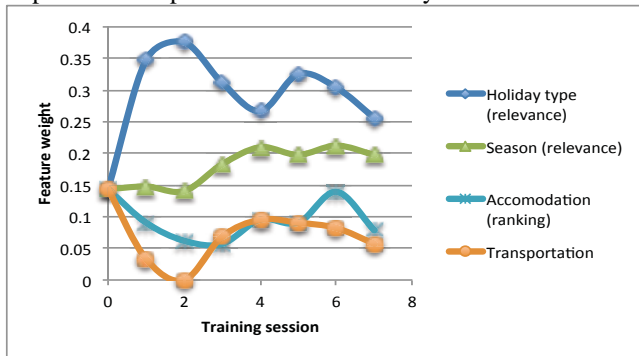


Figure 10: Weight learning results for relevance and ranking feedback established from 3 features.

Part of this problem is due to the error estimation function. If two cases have inverted ranks but also have the same global similarity score, the training scheme estimates that there is no error as the difference of similarity is null. In these conditions, weight learning does not get applied. This relates to the fact that ranking feedback corresponds to non-strict partial rankings. Other criteria to force optimization in these situations should be studied.

Another problem occurs when some features get assigned a weight of 0. These features have no more influence on the ranking of the cases. It then becomes difficult in the following training episodes to reassign them some weight. As fewer features contribute to similarity estimation, the likelihood of having multiple cases with the same similarity score is increased - which brings us back to the problem with the error estimation function.

## Related Work

Early work in this research direction was dedicated to the modification of feature weights when CBR is applied to classification tasks. [Wettschereck & Aha 1997] compared various learning algorithms based on a leave-one out evaluation of the cases to estimate the classification accuracy of a CBR retrieval component. In [Bonzano et al. 1997], learning is used to update feature weights in order to optimize the problem-solving performance of a CBR system. [Branting 2003] proposed an approach to determine feature weights by looking at selections from a set of items. Feature weights are modified by adding/subtracting fixed values so that selected items are moved higher in the recommendation list. Our presentation of the gradient descent approach to learn weights from ranking feedback is inspired from the work of Armin Stahl

[Stahl 2001]. The topic of user preferences has also been studied for recommender systems [Bridge et al. 2005] and conversational CBR [Aha et al., 2005]. Our research effort differs from previous work as it incorporates both relevance and ranking feedback to learn CBR feature weights. We also make use a modified formulation of Rocchio relevance feedback to update CBR feature weights.

## Conclusion

In this paper, we explored how CBR weight learning can be conducted for both ranking and relevance feedback. We proposed to combine techniques to consider both types of feedback simultaneously. Experiments conducted to evaluate the algorithms clearly indicate that the learning scheme can recognize within a few sessions the features used to formulate both types of feedback.

As future work, we recommend to experiment with more complex learning schemes to estimate their benefits. For instance, stochastic gradient descent seems to be an interesting candidate. Other error estimation functions should be studied to force weight optimization when multiple cases have the same global similarity. Finally, to improve the robustness of the solutions, we would like to integrate past feedback into the learning scheme using some decay functions for preferences change over time.

## References

- Aha, D.W. ; McSherry, D. ; & Yang, Q. 2005. Advances in conversational case-based reasoning. *Knowledge Engineering Review*, 20(3), pp. 247-254.
- Bonzano, A.; Cunningham, P.; and Smyth, B. 1997. Using Introspective Learning to Improve Retrieval in CBR: A Case Study in Air Traffic Control, *In Proceedings of the Second International Conference on Case-Based Reasoning Research and Development*, Springer, pp. 291-302.
- Branting, K. 2004. Learning Feature Weights from Customer Return-Set Selections, *Knowledge and Information Systems*, vol. 6, issue 2, Springer, pp. 188-202.
- Bridge, D.G. ; Göker, M.H. ; McGinty, L. ; Smyth, B. 2005. Case-based recommender systems. *Knowledge Engineering Review*, 20(3), pp. 315-320.
- Manning, C.; Raghavan, P.; and Shutze, H. 2009. *Introduction to Information Retrieval*, Cambridge University Press.
- Recio-Garcia, J. A.; Diaz-Agudo, B.; and Gonzalez-Calero, P. 2008. *jCOLIBRI: Tutorial*, Technical report, Facultad de Informatica, Universidad Complutense de Madrid, Spain.
- Stahl, A. 2001. Learning Feature Weights from Case Order Feedback, *Case-Based Reasoning Research and Development*, Springer. pp. 502-516.
- Wettschereck, D.; Aha, D.; and Mohri, T. 1997. A Review and Empirical Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms. *Artificial Intelligence Review*, 11 (1-5), pp. 273-314.