The Profile Editor: Designing a direct manipulative tool for assembling profiles

Patrick Baudisch
Institute for Integrated Information and Publication Systems IPSI
German National Research Center for Information Technology GMD
64293 Darmstadt, Germany
+49-6151-869-854
baudisch@gmd.de

ABSTRACT

Information filtering systems retrieve documents from document streams according to their users' long-term information interests represented by so-called profiles. The *Profile Editor* proposed in this article allows the interactive, direct manipulative construction of profiles. It takes a set of ranked queries and compiles them into a single profile by cropping and re-ranking the queries' results. The approach of manual profile generation is expected to lead to two advantages: a) Profile generation is expected to be much faster than feedback-based automatic profile generation and b) users' confidence in their profiles should be higher because they are in control of their profiles. The *Profile Editor* is currently being implemented in the context of an Internet TV program guide, in which it will be evaluated during the next months.

Keywords

information filtering, profile, histograms, sliders, direct manipulation, user interfaces, Java

INTRODUCTION

The goal of information filtering systems is to keep users from being flooded with information. Filtering systems remove all items from an incoming information stream that are judged to be non-relevant to users – only those items in the stream that correspond to the long term informational need described in the users' so-called profiles are passed through. See [5] for a comparison between information filtering (or selective dissemination of information, SDI [10]) and information retrieval. Among others information filtering systems have been applied to personal mail and Usenet news [7,8], web sites [2, 13], internet advertising [4].

Profile creation is (not only) an iterative process

Figure 1 shows the model of information filtering as proposed by Belkin and Croft [5]. In this model there are three paths that lead to the *Profiles* node: Creation (top right), outer refinement cycle (thick and dotted boxes) and inner refinement cycle (thick boxes only).

The best explored path of the three is the inner refinement cycle that leads to incremental changes of the profile. The cycle contains three actions ▭ and two documents 🗋. The three actions in the refinement cycle are a) *Comparison or Filtering:* Items from the incoming stream are compared with the *profile*. Non-matching items are removed. The remaining items (*retrieved documents*) are presented to the user. These items may or may not contain additional rating/ranking information. b) In the second step (*use and/or evaluation*) the profile system gathers user feedback. In automatic profile generation systems (see below) users are allowed to correct the relevance and/or rating of each item suggested by the filtering system. c) In the third step (*modification*) the *profile* is modified automatically according to the received feedback. The mechanism represented by the inner cycle is also referred to as 'relevance feedback'.

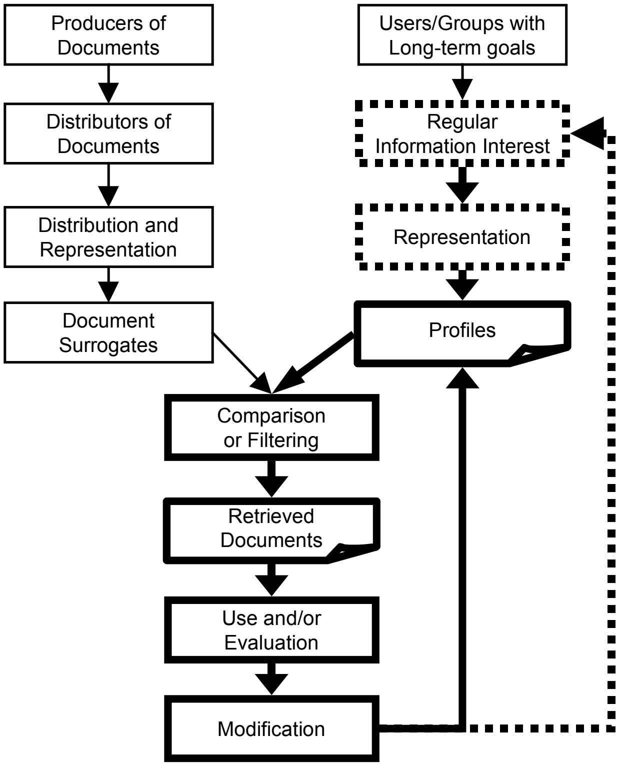


Figure 1: A general model of information filtering according to Belkin and Croft [5]. Three paths lead to the profiles: *Creation* (upper left four boxes), *outer refinement cycle* (thick boxes plus dotted boxes) and *inner refinement cycle* (thick boxes).

Systems like for example the news filters Gnus [7] and GroupLens [8] implement such an inner refinement cycle. During the evaluation phase users give feedback about presented items. The systems use this feedback to modify profiles automatically. Users only deal with documents; the profiles never become directly apparent. As a consequence users do not know about the contents of their profiles – they might not even be aware of their existence. Thus there is no profile creation that could represent a regular information interest as stated in Figure 1. And there is no outer refinement cycle that could allow users to communicate how they changed their information interests.

On the one hand the approach of hiding profiles has the advantage of being easy to use. Since the profiles themselves never become apparent, users are not bothered with additional user interfaces or the profile's internal representation. On the other hand these feedback based profile builders suffer from two limitations. The first limitation is speed. When a profile is created it is either initialized to some stereotype picked by the user or it is even completely empty. In the latter case *all* profile content has to be gathered during the inner refinement cycle. This process takes a lot of time and does not provide a useful profile for quite a long time. The second limitation is the users' confidence in the profile. If the internal state of the learned profile is not accessible, users can never be sure about the current learning state. This lack of transparency can limit the users' confidence which in turn reduces the profiles' applicability in autonomous tasks. Finally the two goals, learning speed and user confidence, seem to exclude each other: Either the learning rate is low and training takes very long, or the learning rate is high and system reactions might be perceived as misunderstandings.

The *Profile Editor* attempts to overcome these limitations by giving users direct access to their profiles. It allows the direct manipulative *creation and modification* of profiles. Its goal is to reduce the number of necessary refinement cycles and to heighten the users' confidence in their profiles.

A PROFILE EDITOR DEMO SESSION

Before going into detail, let's take a look at an application example. The following example session shows a possible interaction sequence from the *TV-Online* system [3], a system that assists users in compiling their personal TV schedules.

Andrea assembles her personal TV schedule. She is interested in sports, especially in basketball, where she does not want to miss a single program. She wants to be up-to-date about current information without spending too much time on it. Finally, for recreation, she wants to include some good action movies.

The first thing she does is to select the four genres Basketball, Information, Sports and Action as her favorite genres. In the TV-Online system this is simply done using toggle buttons associated with each genre as shown in Figure 2[1].

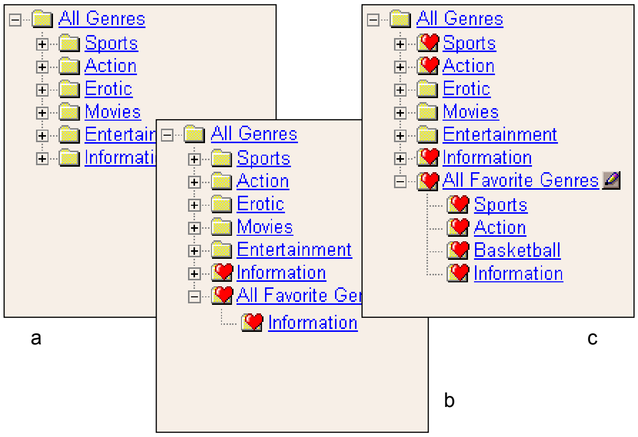
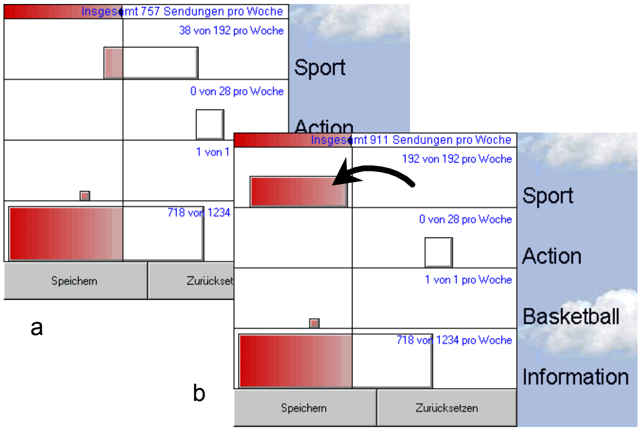


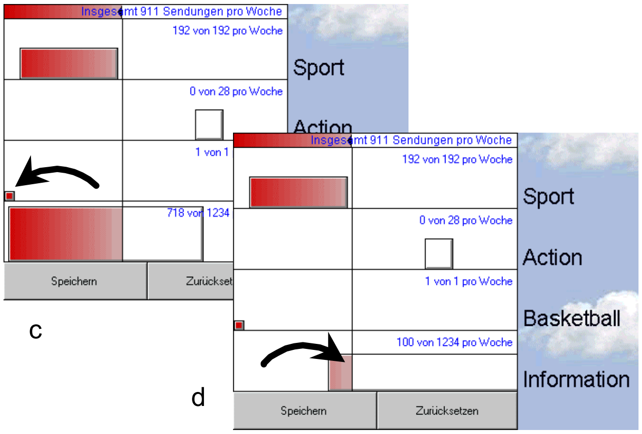
Figure 2: Andrea opens a tree-like menu that contains the hierarchy of all available genres (a) She marks her favorite genre 'Information' by toggling the heart icon in front of it. With the selection of the first favorite genre the folder 'All Favorite Genres' that holds her new favorite genre appears automatically (b). Finally, she selects the other three favorite genres. The original basketball genre is not visible here – it is hidden inside its parent genre sports. (c)

She now has created a personal profile that consists of four genres. She could already query it by selecting 'All Favorite Genres' and starting the query process. This would return the union of all programs from the selected genres. See section 'Initialization' for details on what Andrea would get and under which ranking. Instead she decides to specify her profile in more detail using the Profile Editor. She invokes it by clicking on the edit button ✎.

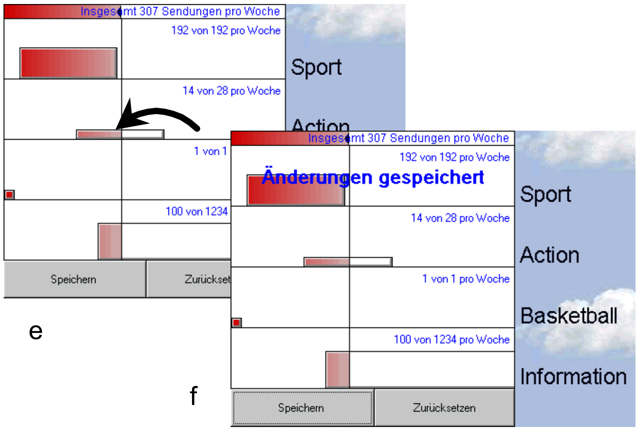


[1] In this example the selection of input queries is the creation of the profile, which is the first path to the profile in Figure 1. This mechanism is not understood as being part of the Profile Editor. Different filtering systems might employ different mechanisms of input queries construction or selection. In a filtering system based on a Web search engine the process of choosing queries might be to 'bookmark' them.

When the Profile Editor is loaded it displays Andrea's four favorite genres[2] (a). To include all sports programs in her personal schedule Andrea moves the corresponding box completely to the left of the vertical line (cropping boundary) (b).



Then she makes sure not to miss any basketball events by dragging the corresponding box to the utter left. All basketball programs will now be output with a maximum rating (c). Next she reduces the number of selected information programs by cropping them at the vertical line. The remaining hundred programs per week will only get low ratings (d).



Now she moves the better half[3] of the action movies into her selection. She stretches the box horizontally to assign higher ratings to the best action movies. (e) Finally she saves the changes (f). As she can tell from the small text in the containers she now has selected an overall number of 307 broadcasts per week (out of approximately 10,000 on German cable TV).

As she now queries her new profile to get her personal schedule for the current week, the broadcasts returned by her favorite genres are output ranked in the order: All basketball broadcasts, then the top half of all sports programs, then all other sports programs mixed with the better action movies and the top information programs (Figure 3).[3]

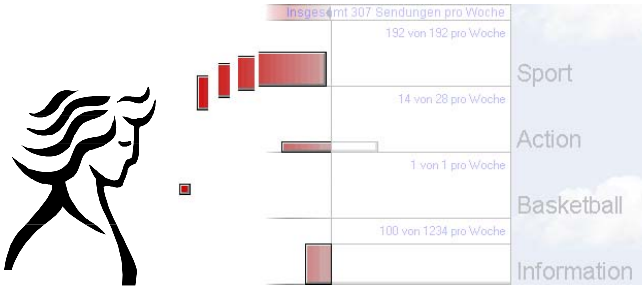


Figure 3: When Andrea queries her profile all items left of the cropping boundary are output ordered from left to right.

BASIC ELEMENTS: HISTOGRAMS AND SLIDERS

Before exactly defining the Profile Editor we will take a look at the basic techniques used. We will start by taking a closer look at sliders and histograms to find out that the draggable boxes demonstrated in the example above represent abstract histograms of query results.

Figure 4 shows a dialog used in a commercial image processor. The dialog allows the conversion of gray scale images into black and white images. The conversion method is very simple in that all brighter pixels are turned to white and all darker pixels are turned to black. The dialog contains a slider that allows the definition of a so-called threshold value, i.e. the luminance value of the darkest color that is converted to white. To assist users in finding an appropriate threshold value the slider is accompanied by a histogram that represents the luminance distribution of an image. Good threshold values might for example be found at local minima around the median of the histogram.

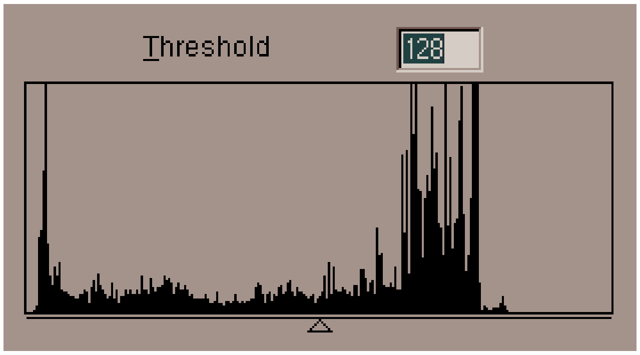


Figure 4: The "threshold" dialog in Adobe Photoshop [1]. The histogram represents the luminance distribution. The little triangle at the bottom is a slider that can be dragged by the user to select a luminance value. The histogram helps in finding useful values to be selection using the slider.

[2] Actually the initial state of the Profile Editor would already be much more appropriate. The shown state was chosen to show all possible interactions. It is a kind of 'worst case' initialization. See section 'initialization' for the actual initialization.

[3] In the TV-Online example ratings are generated on the basis of other viewers schedules (collaborative filtering [13])

Application to information retrieval

The relation between grayscale and black and white images complements the relation between *rating* and *relevance* in information retrieval. Assuming that items are rated perfectly then there is a boundary that determines which items are still relevant and which ones are not. Like in image processing a histogram slider can be used to select this boundary or threshold. Figure 5 suggests such a user interface component for the information retrieval system Inquery [6] and its graphical user interface Xinquery. Both the original and the proposed widgets visualize ratings of documents sorted by rank. While the bar chart in the original interface represents only twelve documents, the suggested histogram represents about two thousand on the same display area (assuming that each white pixel represents one document). The triangular cursor under the histogram marks the currently selected document and displays its rating and rank. Being able to display the whole range at once provides a quick overview about the amount of returned documents and their rating distribution. Notice the different lengths of the two scroll bars.

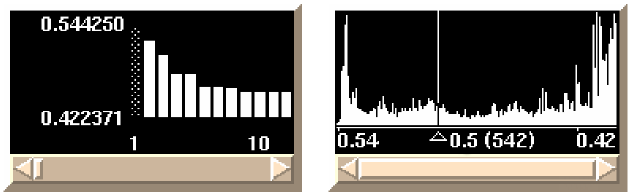


Figure 5: The original Xinquery rating bar chart (left) compared to a widget using a combination of histogram and slider (right). The histogram can represents far more documents than the bar chart. See [15] for more interesting discussion on the Xinquery user interface

To emphasize the relevance property histograms can be colored gradually according to the ratings represented by the individual horizontal positions. The leftmost parts that represent high ratings could for example be rendered red, symbolizing 'hot'. Parts with only average ratings directly left of the threshold document could be rendered in a pale rose. Parts right of the threshold could be filled with background color to underline that they are not selected.

Application to Information filtering

Applying the combination of histograms and sliders to information *filtering* leads to a number of conceptual changes. In information *retrieval* different informational needs can be processed sequentially. Each informational need is represented by a query which is modified and repeated until the right documents are found (stepwise refinement). When one informational need is satisfied the next one is processed. This approach is not feasible in information *filtering*. Here the data base to search is supposed to be dynamic. Informational needs are expected to be long term interests that exist at least for several sessions. It becomes necessary to hold and maintain several queries at the same time in a so-called profile. Figure 6 illustrates the inclusion hierarchy of profiles and queries.

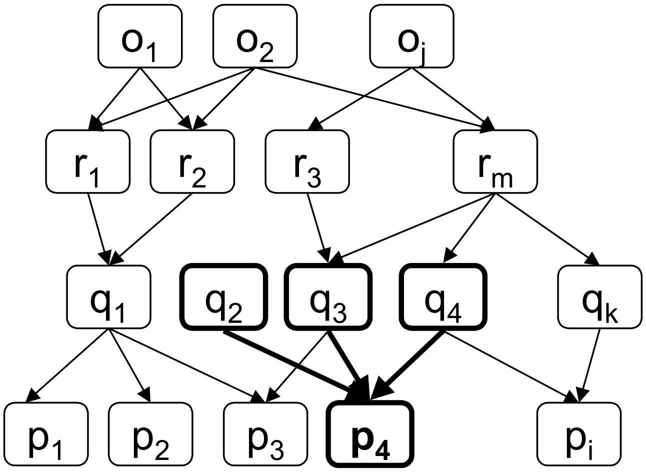


Figure 6: Inference network for information filtering according to Belkin and Croft [5]. O_j are the nodes associated with incoming objects, r_m's are concept nodes, q_k's are query nodes and p_i's represent the profiles. Profiles are collections of queries, The profile p_4 for example includes q_2, q_3 and q_4.

Since a profile consists of several queries, an adapted interface has to contain several histogram sliders, one for every query (Figure 7a). To integrate the results of all these queries into a single output, the ratings of the documents returned by the individual queries have to be mapped to a common domain. To visualize that in the interface, all histograms are inserted into a container that represents this common rating domain (Figure 7b).

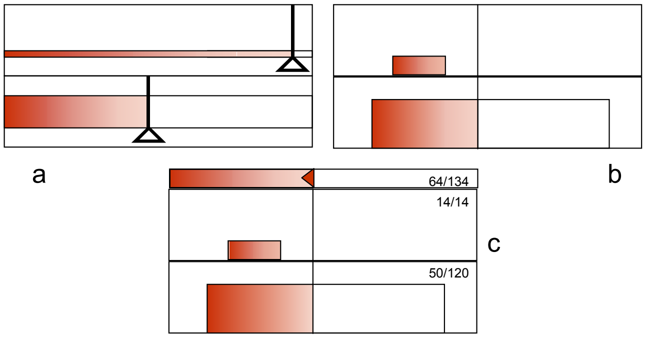


Figure 7: Application of the histogram sliders to information filtering. A profile consists of several queries, each one represented by a histogram slider (a). To integrate the ratings of the different queries into a common space sliders are replaced by a single vertical line called *cropping boundary*. Histograms are moved now instead of sliders (b). The version already presented in the demo session has an extra handle for the cropping boundary and textual information about the number of selected items (c).

Output ratings are now represented by horizontal positions of the surrounding container. The set of threshold sliders now becomes a single vertical line that crosses the whole container. Since this line defines which parts of the result sets will be cut off, we call the line *cropping boundary*. Like the slider, the cropping boundary separates histograms in two subsets: The subset of items that will be returned to the user and the one that will be filtered out. The cropping boundary has only one degree of freedom but has to represent the n degrees of freedom represented by the n sliders before. To accomplish that, *histograms* now have to be dragged instead. The cropping boundary usually stays fixed. Allowing it to be moved as well provides an additional degree of freedom that can be used for influencing the cropping of all queries at once. The next step in adapting histogram sliders to information filtering is to abstract the histograms. The rating distributions visualized by histograms change over time – they might never be the same for two individual runs of the Profile Editor. Therefore no specific histogram can represent all future states of the profile. To avoid misleading information in histograms we use abstract shapes instead. Abstract shapes represent any possible state of a histogram, although not precisely[4]. Of course all advantages related to the display of the concrete rating distribution get lost during abstraction. Different levels of abstraction are possible for displaying and manipulating rating histograms. Figure 8 gives examples. The *abstract* histogram type is very useful in the case that the general type of distribution is known and about constant over time. The *ranked* display is a catch-all: It matches all possible distributions if the rating distribution is replaced by a ranking. Since much rating information gets lost during ranking, abstract histograms should be used instead of normalized histograms whenever possible. Finally, distributions of known type can be given any arbitrary shape by transforming ratings using a continual function. Using this approach any distribution can for example be represented by a rectangle, as we used it for most examples in this article.

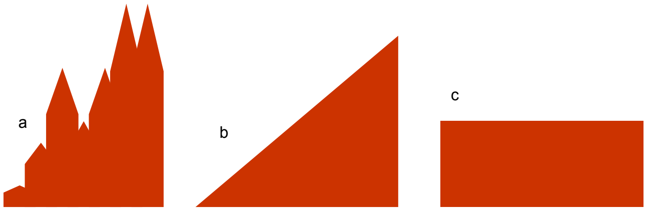


Figure 8: Examples for different abstraction levels of the histogram representation: realistic (a), abstract (b), ranked (c).

Histogram areas have the important function of visualizing the number of presented and selected items. For an example see any figure of the demo session and compare the areas of basketball and information. Based on the area information users are able to estimate how many items they are dealing with and how much effort it is going to take to process the results. Therefore the limited space within the containers makes perfect sense: The overall space left of the cropping boundary represents users' input capacities. By dragging the cropping boundary this space can be customized within the limits of the container[5].

If some queries return very many items while others return only very few the area dynamics may exceed the displayable range. In this case histogram surfaces can be scaled non-proportionally to make sure that even the smallest and the biggest histograms can be easily recognized and manipulated by users. Scaling can for example be done using the following formula:

$$s = s_{\min}\left(\frac{n}{n_{\min}}\right)^{e} \quad \text{with} \quad e = \frac{\log\left(\frac{s_{\max}}{s_{\min}}\right)}{\log\left(\frac{n_{\max}}{n_{\min}}\right)}$$

with s, $s_{\min}$ and $s_{\max}$ being the current, minimum and maximum surfaces respectively, n, $n_{\min}$ and $n_{\max}$ being the current, the minimum and the maximum number of query result items that should be displayed entirely.

PROFILE EDITOR USER INTERACTIONS

The Profile Editor, as implemented in TV-Online, is completely mouse-driven. It supports the following drag and drop interactions:

1. Dragging histograms in the horizontal direction shifts them within the container. Moving histograms to the left increases the ratings of all represented items; moving them to the right decreases ratings. Moving histograms or parts of histograms into the area left of the cropping boundary increases the number of selected items, the opposite decreases the number of selected items. To provide more space for non-selected histogram parts boxes are allowed to stick out to the right.
2. Dragging histograms vertically modifies their aspect ratio. Dragging downwards makes histograms flat and wide, dragging upwards makes them high and narrow. Flat and wide histograms assign a wide spectrum of ratings to the represented items, high and narrow histograms assign similar ratings to represented items. Since histogram deformation can be confusing for novice users, the deforming feature might be omitted in a simplified version. In this case all histograms have fixed aspect ratios. But the additional degree of freedom provided by the change-aspect-ratio feature proved to be quite useful. It allows users to assign arbitrary ratings to the best items while making use of the cropping feature at the same time.

[4] To visualize the fact that the histograms in the Profile Editor are not concrete, it might be interesting to give them a less determined shape. Good ideas about so-called non-photo realistic line drawings can be found in [13].

[5] In the TV-Online example there is no such surface restriction: Histograms can be deformed arbitrarily so they can stick out at the top. This is necessary to support the multi-select feature (see section 'Initialization').

3. Dragging the cropping boundary is a shortcut to modify all queries at once. Moving the cropping boundary to the left decreases the overall number of selected items; moving it to the right decreases it.

The fact that no handles are needed to manipulate histograms makes the interface easy to use. To facilitate the picking and dragging of small histograms any mouse-down event in the whole container (beside those that initiate dragging the cropping boundary) can be used to start a histogram drag interaction.

DEFINITION OF PROFILE AND RATINGS

A profile generated by the *Profile Editor* consists of the position of the cropping boundary b and a set of queries[6] q with their current rating transformation f.

$$profile := ((q, f)^n, b)$$

Each rating transformation maps its query's input ratings r_{in} to global output ratings r_{out}. Output ratings are defined as

$$r_{\text{out}} = f(r_{in}) = r_{in} w + i$$

with w being the horizontal scaling of a histogram and i being the indentation measured from the right. Assuming that both input and output rating are ranged 0 to 1, $w = 1$ assigns the full container width to a histogram. $i = 0$ results in the histogram to be right aligned with its container, $i = 1 - w$ to be left aligned. Inserting this into the profile definition given above this leads to

$$profile := ((q, w, i)^n, b)$$

If an item is returned by more than one input query, the output rating is calculated as the maximum over all r_{out}. Other functions like weighted sums were tested but cannot be discussed here due to space limitations.

The cropping boundary b determines the minimum output rating for items to be returned to the user. The function of the cropping boundary is to remove non-relevant items. The cropping boundary defines the minimum rating for items to be returned to the user. The Boolean variable *output* that determines whether an item is output to the user

$$output := \begin{cases} true & if \ r_{out} > b \\ false & if \ r_{out} \le b \end{cases}$$

with b being the position of the cropping boundary measured from the right of the container. This definition reduces the value range of r_{out} to [b,1]. For many visualizations it will be useful to stretch the output domain to the full range [0,1] by replacing the definition of r_{out} with

$$r_{\text{out}} = f(r_{in}) = (r_{in} w + i - b) / (1 - b)$$

To support the abstract histogram visualization all queries have to be provided with the average number of returned items and the shape of the typical distribution.

INITIALIZATION

In information retrieval descriptors with low inverted document frequencies are considered more relevant (Law of Zipf, [14, p. 60]). This notion is used to initialize profiles. Queries returning fewer items are expected to deliver more relevant items and are therefore initialized to higher ratings and histograms are placed more to the left.

Additional constraints might be imposed by the application. In the TV-Online system users can create and use profiles without fine tuning, i.e. without using the Profile Editor (see Figure 2), which makes the profile work a kind of multi select[7]. Therefore, all queries have to be initialized as being fully inside the selected range, i.e. left of the cropping boundary. With these initializations users will profit from the indentation created based on the triviality notion even without fine tuning their profile (Figure 9).

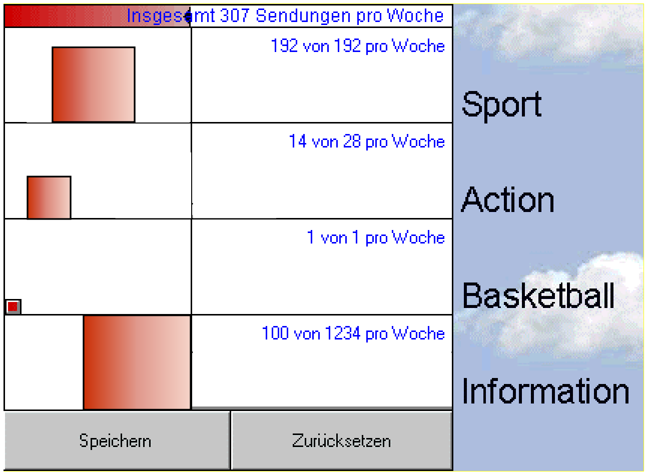


Figure 9: Example of an initialization of newly added queries. Smaller histograms are placed more to the left. In TV-Online all histograms are placed left of the cropping boundary.

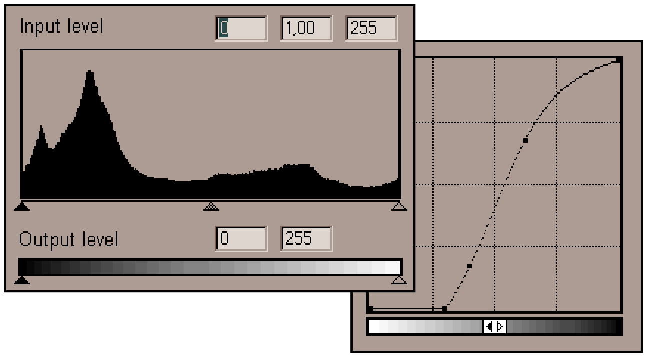
FURTHER RESEARCH

1. Use the Profile Editor on top of Web search engines. The existing service *The Informant* [16] notifies users about newly found pages. The Profile Editor could be used to rank individual queries and to define minimum ratings.
2. Program and test the proposed the histogram user interface component for retrieval systems (Figure 5)

[6] The Profile Editor supports only the definition of the cropping boundary and the transformations — as already mentioned the query set q is expected to be provided by the surrounding system.

[7] As the evaluations showed many users did not want to spend additional work on fine tuning their profiles. In this case it was very important that the profiles worked without the extra effort.

3. Explore and compare different versions of the Profile Editor: Cropping boundary draggable or not, with additional display of number of selected items or not, with extra container for cropping boundary or not.
4. Apply the Profile Editor to image processing. While the Profile Editor maps input ratings to output ratings, gray image filters like the threshold dialog (Figure 4) map input luminance to output luminance. Figure 10 shows two more examples. Can the Profile Editor user interface be used to manipulate multi channel images?



Input level 0 1,00 255

Output level 0 255

Figure 10: Two dialogs from an image processor that map input luminance to output luminance (Adobe Photoshop4.0 [1]).

CONCLUSION

We introduced the concept of direct profile manipulation to fasten profile creation and to increase the users' confidence in their profiles. At the beginning of this article it was presented as an alternative to automatic profile generation as used in systems like Group Lens. But actually the concept of direct profile manipulation is not necessarily opposed to feedback based learning. It seems useful to combine both approaches: Provide a Profile Editor for bigger changes in the outer refinement cycle and to give users more insight into their profile. Use the more convenient feedback learning for incremental changes in the inner refinement cycle. This combination will be the next concept to implement and test.

ACKNOWLEDGMENTS

I thank Dieter Böcker and Ulrich Thiel for their support. Special thanks to Gerrit Voss for the Java implementation of the Profile Editor.

REFERENCES

1. Adobe Photoshop, online at http://www.adobe.com/prodindex/photoshop/main.html
2. Balabanovic, M and Shoham, Y. Fab: Content-Based, Collarorative Recommendation, *Commun. ACM 39,* 6, p. 66–72 (recommend Web sites)
3. Baudisch, P. Designing an Evolving Internet TV Program Guide, *Proceedings of the HCIC '97 workshop,* 19.-23.2.1997, Snow Mountain Ranch, CO. Available Online at www.darmstadt.gmd.de/~baudisch/Publications
4. Baudisch, P., Leopold, D. User-configurable advertising profiles applied to Web page banners, To appear in *Proceedings of the first Berlin Economics Workshop,* 24-25 October 1997, Berlin
5. Belkin, Nicholas J., and Croft, W. Bruce Information Filtering and Information Retrieval: Two Sides of the Same Coin? *Commun. ACM 35*, 12
6. Callan, J.P., W.B. Croft, and S.M. Harding: 1992, 'The INQUERY Retrieval System'. In: *Proceedings of the 3rd International Conference on Database and Expert System Applications*. Berlin and New York: Springer, pp. 78-83
7. Gnus news reader, online at http://www.aston.ac.uk/lis/as/manuals/xemacs/gnus/
8. Konstan J.A. et al Grouplens: Applying Collaborative Filtering to Usenet News, *Commun. ACM 39,* 6, p77–87, The Grouplens homepage is http://www.cs.umn.edu/Research/GroupLens
9. Netnanny, online at http://www.netnanny.com
10. Parker, K.H. and Soergel, D. The importance of sdi for current awareness in fields with severe scatter of information. *J. Am. Soc. Inf. Sci. 30, 3* (1979), 125–135
11. Pollock, S. A rule-based message filtering system. *ACM Transactions on Office Information Systems* 6, 3 (July 1988), 232–54
12. Robertson, S.E. The probability ranking principle in IR. J. Doc. 33, 4 (Dec 1977), 294–304
13. Schumann, J., Strothotte, Th., Raab, A., Laser, S. (1996), "Assessing the Effect of Non-Photorealistic Images in Computer-Aided Design", *ACM Human Factors in Computing Systems, SIGCHI '96*, Vancouver, April 13-15, 1996, pp. 35-41
14. Salton, G., McGill, M.J., Introduction to Modern Information Retrieval, New York: McGraw-Hill, 1983
15. Shneiderman, B. A User-Interface Framework for Text Searches, D-Lib Magazine, Jan 1997, available online at http://www.dlib.org/dlib/january97/retrieval/01shneiderman.html
16. The Informant, online http://informant.dartmouth.edu