EVALUATING AND ANALYSING DYNAMIC PLAYLIST GENERATION HEURISTICS USING RADIO LOGS AND FUZZY SET THEORY

Klaas Bosteels
Ghent University, Gent, Belgium
klaas.bosteels@ugent.be

Elias Pampalk
Last.fm Ltd., London, UK
elias@last.fm

Etienne E. Kerre
Ghent University, Gent, Belgium
etienne.kerre@ugent.be

ABSTRACT

In this paper, we analyse and evaluate several heuristics for adding songs to a dynamically generated playlist. We explain how radio logs can be used for evaluating such heuristics, and show that formalizing the heuristics using fuzzy set theory simplifies the analysis. More concretely, we verify previous results by means of a large scale evaluation based on 1.26 million listening patterns extracted from radio logs, and explain why some heuristics perform better than others by analysing their formal definitions and conducting additional evaluations.

1. INTRODUCTION

In January 2009, Arbitron and Edison Research measured the popularity of digital music platforms by means of a survey of 1,858 American people aged 12+ . [1] They estimated that 42 million Americans tune to online radio on a weekly basis, which is more than twice their number from 2005, and claim that the number of 12+ year old Americans owning a digital music player increased from 14% in 2005 to 42% in 2009. They also found that the vast majority of these people own an Apple iPod or iPhone. Evidently, the Apple products dominate their market, which is commonly attributed to their innovating design and user interfaces. The recent "Genius" feature is a nice example of such innovation. Using this feature, users can automatically create coherent playlists by selecting a *seed song*, i.e., an example of a song of interest, and pressing a single button. Many of the popular online radio stations are similar in concept. The user supplies one or more seeds, and the system generates a corresponding list of tracks that is turned into a custom radio station. Hence, automatic playlist generation can be seen as a technology that is, to some extent, responsible for the recent growth established by certain digital music platforms, and its commercial importance is likely to increase further in the near future.

This paper is about simple heuristics for automatically generating playlists. More precisely, we will discuss simple rules of thumb for choosing the song to be played next, given a set of candidate songs. This set of candidates can consist of all available tracks, but usually it is restricted to a limited subset. In order to avoid repetition, for instance, the set of candidates has to be restricted to the songs that have not been played yet. A realistic scenario is to select the candidates using some other method, effectively turning the heuristic into an enhancement rather than a playlist generation method on its own.

A very simple way to improve upon random selection, is to repeatedly choose the candidate that is most similar to a given seed song [1]. This *playlist generation heuristic* is said to be static because the song sequence is completely determined from the seed, without taking any additional user input into account. Dynamic heuristics, on the other hand, rely on user feedback to dynamically improve the selection process [2]. For example, the aforementioned static heuristic can be made dynamic by letting it pick the song that is most similar to any of the accepted songs, where the set of accepted songs consists of the seed song as well as all tracks that were not skipped [3]. When there is no given seed, the set of accepted songs can initially be empty and the next track can be chosen at random until there is at least one accepted song. This latter heuristic could easily be added to any system that returns multiple candidate songs for being played next.

Putting it in one sentence, we discuss simple dynamic playlist generation heuristics in this paper. In comparison with alternative techniques, such heuristics are interesting because they (i) are simple and thus easy to compute and implement, and (ii) can easily be added as an enhancement to many existing playlist generation systems.

2. RELATED WORK

Dynamic playlist generation can be seen as a special case of the well-known *relevance feedback* paradigm from information retrieval [4]. In this paradigm, the user is asked to give explicit feedback by labeling results as either relevant or irrelevant, which leads to additional information that can be used by the system to refine the search strategy and generate a better list of results. Several rounds of feedback can be conducted, each bringing the results closer to the user's implicit target concept. Hence, dynamic playlist generation is basically relevance feedback with the returned set of results restricted to one item. In case of this paper, the feedback taken into account is also implicit rather than explicit, but there is no reason to as-

[1] http://www.arbitron.com/study/digital_radio_study.asp

sume that dynamic playlist generation heuristics could not be based on more explicit feedback like "thumbs up/down" buttons instead of skipping behavior.

Over the past few years, relevance feedback has received quite a lot of research attention. In particular, some relevance feedback techniques have already been applied to music information retrieval, including training a decision tree [5], a vector quantizer [6], and an SVM [7]. Since the number of examples is very low when the returned set of results is restricted to one item [2], using these machine learning techniques for dynamic playlist generation might be problematic, however. For the custom-tailored heuristics described in this paper, this is less of a problem. Moreover, their simplicity can be considered an additional advantage from a computational and implementation point of view. Furthermore, as we already mentioned in the introductory section, the described heuristics can also be thought of as a refinement that can be added to a more complex relevance feedback system.

3. FORMALIZATION

The definition of playlist generation heuristics can be formalized using fuzzy set theory [8]. In this section, we explain this formalization in detail, since we rely on it extensively in the subsequent sections.

3.1 Fuzzy Sets

Let U denote a universe, i.e., a (crisp) set of considered objects. A *fuzzy set* F in U is a $U \to [0,1]$ mapping that associates a degree of membership $F(u)$ with each element u from U [9]. The higher $F(u)$, the more u is a member of F. In particular, u fully belongs to F when $F(u) = 1$, and $F(u) = 0$ implies that u is not at all an element of F. We use the notation $\mathcal{F}(U)$ for the class of fuzzy sets in U, which can be regarded a superclass of $\mathcal{P}(U)$, the class of crisp sets in U. A *(binary) fuzzy relation* L in U is a fuzzy set in $U \times U$, i.e., $L \in \mathcal{F}(U \times U)$ [9].

The fuzzy set Sim_X, with X a crisp set in the (finite) universe C of songs to be explored, is the main stepping stone towards the fuzzy formalization. It is given by

$$Sim_X(u) = \max_{x \in X} M(u,x) \tag{1}$$

for all $u \in U$, where U is the subset of C consisting of all candidate songs. In this definition, M is a fuzzy relation in C such that each relationship degree $M(c,d)$, with $(c,d) \in C^2$, corresponds to the degree to which c is similar to d. Putting it in words, Sim_X is a fuzzy set in U such that $Sim_X(u)$ can be interpreted as the degree to which u is similar to any song in X.

In order to obtain a crisp set of tracks from a fuzzy song set, we rely on the following formal operator:

$$X \rceil F = \left\{ x \in X \mid F(x) = \max_{y \in X} F(y) \right\} \tag{2}$$

for all $X \in \mathcal{P}(C)$ and $F \in \mathcal{F}(C)$, i.e., $X \rceil F$ is the crisp set consisting of the elements from X with the greatest membership degree in F. Using this operator, we can formally define the dynamic heuristic discussed in the introductory section of this paper as $U \rceil Sim_A$, with A the set of all accepted songs. In practice, the set $U \rceil Sim_A$ will be a singleton most of the time, but theoretically speaking it can contain up to $|U|$ elements. We can choose one element at random when $|U \rceil Sim_A| > 1$, however, since each song from the set can be considered equally suitable for being played next. In the remainder of this paper, we silently assume that this procedure is followed for all introduced heuristics, i.e., we will define the heuristics as crisp sets and assume that one element is chosen at random when this set has several members.

3.2 Operations on Fuzzy Sets

The set-theoretic operations complement, intersection, and union can be generalized to fuzzy sets as follows:

$$(\mathrm{co}_{\mathcal{N}} F)(u) = \mathcal{N}(F(u)) \tag{3}$$

$$(F \cap_{\mathcal{T}} G)(u) = \mathcal{T}(F(u), G(u)) \tag{4}$$

$$(F \cup_{\mathcal{S}} G)(u) = \mathcal{S}(F(u), G(u)) \tag{5}$$

for each $u \in U$, with $F, G \in \mathcal{F}(U)$, $\mathcal{N}$ a *negator*, $\mathcal{T}$ a *t-norm*, and $\mathcal{S}$ a *t-conorm*. We restrict the sheer number of possibilities by only considering the widely-used standard negator N_{S} given by $N_{\mathrm{S}}(x) = 1 - x$ for all $x \in [0,1]$, the three prototypical t-norms [10] given by

$$T_{\mathrm{M}}(x,y) = \min(x,y) \tag{6}$$

$$T_{\mathrm{P}}(x,y) = x \cdot y \tag{7}$$

$$T_{\mathrm{L}}(x,y) = \max(x + y - 1, 0) \tag{8}$$

for all $x, y \in [0,1]$, and their duals

$$S_{\mathrm{M}}(x,y) = \max(x,y) \tag{9}$$

$$S_{\mathrm{P}}(x,y) = x + y - x \cdot y \tag{10}$$

$$S_{\mathrm{L}}(x,y) = \min(x + y, 1) \tag{11}$$

for all $x, y \in [0,1]$. In the remainder, we will abbreviate $\mathrm{co}_{N_{\mathrm{S}}}$ by co since N_{S} is the only negator we consider.

For this paper, however, we mainly need a generalized set-theoretic difference, which can be obtained by defining

$$(F \setminus_{\mathcal{I}} G)(u) = N_{\mathrm{S}}(\mathcal{I}(F(u), G(u))) \tag{12}$$

for every u from U, with $F, G \in \mathcal{F}(U)$ and $\mathcal{I}$ an *implicator*. We consider two ways of generating implicators in this paper, namely, *S-implicators* and *R-implicators*. The S-implicator induced by a t-conorm $\mathcal{S}$ and the standard negator N_{S} is the $[0,1]^2 \to [0,1]$ mapping $\mathcal{I}_{\mathcal{S}}$ defined as $\mathcal{I}_{\mathcal{S}}(x,y) = \mathcal{S}(N_{\mathrm{S}}(x), y)$, for all $x, y \in [0,1]$, and the R-implicator induced by a t-norm $\mathcal{T}$ is the $[0,1]^2 \to [0,1]$ mapping $\mathcal{I}_{\mathcal{T}}$ given by, for all $x, y \in [0,1]$, $\mathcal{I}_{\mathcal{T}}(x,y) = \sup\{\gamma \in [0,1] \mid \mathcal{T}(x,\gamma) \leq y\}$. For the above-mentioned prototypical t-norms and the corresponding t-conorms, this leads to the following implicators:

$$I_{S_{\mathrm{M}}}(x,y) = \max(1 - x, y) \tag{13}$$

$$I_{S_{\mathrm{P}}}(x,y) = 1 - x + x \cdot y \tag{14}$$

$$I_{S_{\mathrm{L}}}(x,y) = \min(1 - x + y, 1) \tag{15}$$

$$I_{T_{\mathrm{M}}}(x,y) = \begin{cases} 1 & \text{if } x \leq y \\ y & \text{otherwise} \end{cases} \tag{16}$$

$$I_{T_{\mathrm{P}}}(x,y) = \begin{cases} 1 & \text{if } x \leq y \\ \frac{y}{x} & \text{otherwise} \end{cases} \tag{17}$$

$$I_{T_{\mathrm{L}}}(x,y) = I_{S_{\mathrm{L}},N_{\mathrm{S}}}(x,y) \tag{18}$$

for all $x, y \in [0,1]$.

3.3 Formal Heuristics

Having the operations on fuzzy sets at our disposal, we can incorporate the set R of all rejected songs by replacing Sim_A in $U \rceil Sim_A$ with a set-theoretic expression in terms of both Sim_A and Sim_R. The heuristic defined as $U \rceil (Sim_A \setminus_{\mathcal{I}} Sim_R)$, for instance, selects the songs that are similar to an accepted song but not similar to any rejected ones, as illustrated by Fig. 4(a). By taking into account the fact that $(U \rceil F) \rceil F = U \rceil F$ for each $F \in \mathcal{F}(U)$, we can easily define slightly more fine-grained heuristics, however. Instead of replacing Sim_A in $U \rceil Sim_A$, we can first rewrite $U \rceil Sim_A$ as $(U \rceil Sim_A) \rceil Sim_A$ and then replace only the first occurrence of Sim_A, which effectively leads to heuristics of the form $(U \rceil P) \rceil Sim_A$, where P is a set-theoretic expression in Sim_A and Sim_R. We call this expression P the *preselection expression*, since it implements a preselection step that precedes further filtering by Sim_A. As values for P, we consider the set-theoretic expressions illustrated by Fig. 4(a), Fig. 4(b), Fig. 4(c), which leads to the following heuristics:

$$H_{\mathrm{a}}^{\mathcal{I}} = (U \rceil (Sim_A \setminus_{\mathcal{I}} Sim_R)) \rceil Sim_A \tag{19}$$

$$H_{\mathrm{b}} = U \rceil Sim_A \tag{20}$$

$$H_{\mathrm{c}}^{\mathcal{I}} = (U \rceil \mathrm{co}(Sim_R \setminus_{\mathcal{I}} Sim_A)) \rceil Sim_A \tag{21}$$

For the value of the parameter $\mathcal{I}$ in $H_{\mathrm{a}}^{\mathcal{I}}$ and $H_{\mathrm{c}}^{\mathcal{I}}$, we will consider the five different implicators discussed in the previous subsection, namely, $I_{S_{\mathrm{M}}}$, $I_{S_{\mathrm{P}}}$, $I_{S_{\mathrm{L}}}$, $I_{T_{\mathrm{M}}}$, and $I_{T_{\mathrm{P}}}$.

The contour plots in Fig. 1 show how the implemented preselection strategy varies for the considered implicators. For each preselection expression P, there exists a corresponding $[0,1]^2 \to [0,1]$ mapping p such that $P(u) = p(Sim_A(u), Sim_R(u))$ for all $u \in U$. Table 1 lists these mappings for all considered (non-trivial) preselection expressions, and the plots in Fig. 1 each illustrate one of these mappings. Essentially, these plots provide a top view of the three-dimensional plots for the $[0,1]^2 \to [0,1]$ mappings. More precisely, the lines connect points for which the illustrated mapping yields the same value, leading to a partitioning of the $[0,1]^2$ square into different areas. The darker the area, the higher the values returned by the mapping in this area. Hence, songs u for which the point $(Sim_A(u), Sim_R(u))$ is in a dark area are given preference by the preselection strategy in question.

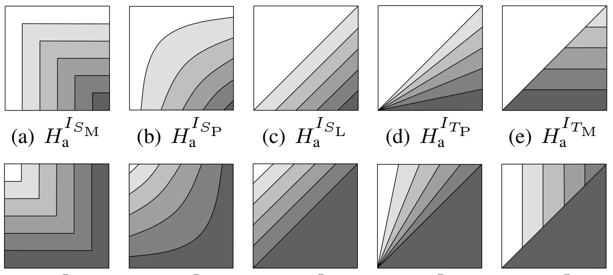
All previously-introduced playlist generation heuristics can be formalized in this way [8]. In particular, the well-performing heuristic defined as

> For each candidate song, let d_a be the distance to the nearest accepted, and let d_s be the distance to the nearest skipped. If $d_a < d_s$, then add the candidate to the set S. From S play the song with smallest d_a. If S is empty, then play the candidate song which has the best (i.e. the lowest) d_a/d_s ratio.

in [2], is equivalent to $H_{\mathrm{c}}^{I_{T_{\mathrm{P}}}}$. In addition to being more concise and precise, the formal definition of this heuristic was also obtained more systematically and is easier to analyse, as we will demonstrate later on in this paper.

preselection expression	$[0,1]^2 \to [0,1]$ mapping
$Sim_A \setminus_{I_{S_{\mathrm{M}}}} Sim_R$	$\min(x, 1-y)$
$Sim_A \setminus_{I_{S_{\mathrm{P}}}} Sim_R$	$x - x \cdot y$
$Sim_A \setminus_{I_{S_{\mathrm{L}}}} Sim_R$	$\max(x-y, 0)$
$Sim_A \setminus_{I_{T_{\mathrm{M}}}} Sim_R$	$\begin{cases} 0 & \text{if } x \leq y \\ 1-y & \text{otherwise} \end{cases}$
$Sim_A \setminus_{I_{T_{\mathrm{P}}}} Sim_R$	$\begin{cases} 0 & \text{if } x \leq y \\ 1-\frac{y}{x} & \text{otherwise} \end{cases}$
$\mathrm{co}(Sim_R \setminus_{I_{S_{\mathrm{M}}}} Sim_A)$	$\max(1-y, x)$
$\mathrm{co}(Sim_R \setminus_{I_{S_{\mathrm{P}}}} Sim_A)$	$1 - y + y \cdot x$
$\mathrm{co}(Sim_R \setminus_{I_{S_{\mathrm{L}}}} Sim_A)$	$\min(1-y+x, 1)$
$\mathrm{co}(Sim_R \setminus_{I_{T_{\mathrm{M}}}} Sim_A)$	$\begin{cases} 1 & \text{if } y \leq x \\ x & \text{otherwise} \end{cases}$
$\mathrm{co}(Sim_R \setminus_{I_{T_{\mathrm{P}}}} Sim_A)$	$\begin{cases} 1 & \text{if } y \leq x \\ \frac{x}{y} & \text{otherwise} \end{cases}$

Table 1. Corresponding $[0,1]^2 \to [0,1]$ mappings for the considered (non-trivial) preselection expressions.


(a) $H_{\mathrm{a}}^{I_{S_{\mathrm{M}}}}$ (b) $H_{\mathrm{a}}^{I_{S_{\mathrm{P}}}}$ (c) $H_{\mathrm{a}}^{I_{S_{\mathrm{L}}}}$ (d) $H_{\mathrm{a}}^{I_{T_{\mathrm{P}}}}$ (e) $H_{\mathrm{a}}^{I_{T_{\mathrm{M}}}}$

(f) $H_{\mathrm{c}}^{I_{S_{\mathrm{M}}}}$ (g) $H_{\mathrm{c}}^{I_{S_{\mathrm{P}}}}$ (h) $H_{\mathrm{c}}^{I_{S_{\mathrm{L}}}}$ (i) $H_{\mathrm{c}}^{I_{T_{\mathrm{M}}}}$ (j) $H_{\mathrm{c}}^{I_{T_{\mathrm{P}}}}$

Figure 1. Contour plots illustrating the preselection strategies of the considered instances of $H_{\mathrm{a}}^{\mathcal{I}}$ and $H_{\mathrm{c}}^{\mathcal{I}}$. Every candidate song u corresponds to a point $(Sim_A(u), Sim_R(u))$ in each of these plots, and the points in the darker areas are given preference by the strategy in question.

4. BASIC EVALUATION

The evaluations described in [2] and [8] are all based on the fairly simplistic assumption that a song is a good addition to a playlist when it is from the same genre as the seed. For this paper, however, we evaluated the considered heuristics using patterns extracted from Last.fm radio logs. More precisely, we looked for sequences of 22 tracks for which the last two tracks did not both get accepted or rejected, i.e., one of them got accepted while the other got rejected. Tracks were considered accepted when the user listened to more than 50% of them. In order to make sure that the extracted patterns represent genuine user interactions, we imposed two additional restrictions: (i) at least 5 and at most 15 of the first 20 tracks got accepted, and (ii) the last song of the sequence was not the last song of a listening session. In this way, we avoid problems like, e.g., the user falling asleep or getting distracted while listening to the radio station, or the last song being considered a skip whereas the user really just turned off the radio while this song was playing.

All of the patterns used for our evaluation were extracted from log files produced by Last.fm "playlist" radio stations, which basically shuffle randomly through user-generated lists of tracks. Last.fm provides its users the ability to create and share playlists, and subscribers can listen to these playlists in random shuffle mode when they contain at least 45 playable tracks by 15 different artists. We considered 1,260,271 patterns extracted from log files generated by such stations, involving 53,768 unique listeners and 516,261 different tracks from 70,306 artists.

The similarity values used for our evaluation were derived from tag data using the well-known cosine similarity measure [4], i.e., songs to which Last.fm users applied the same tags were considered similar to each other. Since the values from $[0, 1]$ obtained in this way can directly be interpreted as membership degrees, we did not have to apply any normalization procedures in order to obtain the fuzzy relation M on which the definition of Sim_X is based.

For each considered pattern, we made every heuristic choose between the last two tracks based on the acceptance history for the 20 previous tracks, and counted how many times they picked the wrong one. More formally, each pattern corresponds to a (A, R, r, w) tuple, where A and R are the sets of accepted and rejected songs, respectively, and r and w are the right and the wrong choice. The *failure rate* for a given heuristic is then obtained by putting $U = \{r, w\}$ for each pattern and counting how many times w is returned by the heuristic.

Fig. 2 shows the results of our basic evaluation. The circles mark the failure rates, and the lines through them represent the 95% binomial confidence intervals computed by approximating the binomial distribution with a normal distribution. These results roughly confirm the findings obtained in [8]. Again, $H_c^{I_{S_L}}$ and $H_c^{I_{T_P}}$ perform significantly better than the other heuristics, although the difference between $H_c^{I_{T_P}}$ and $H_c^{I_{S_P}}$ is just barely significant in this case. It still remains unclear why exactly these two heuristics perform best, however, which is precisely the motivation for the subsequent sections of this paper.

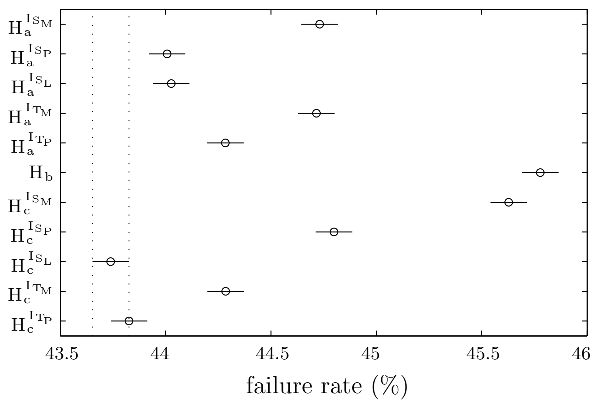


Figure 2. Results of the basic evaluation. The circles mark the failure rates, and the lines represent the 95% binomial confidence intervals.

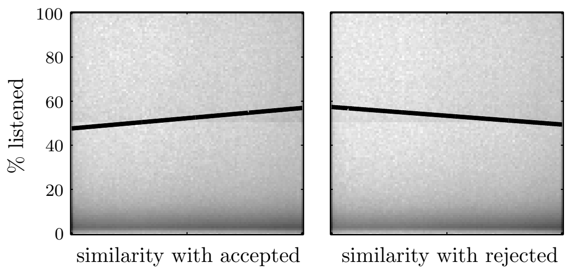
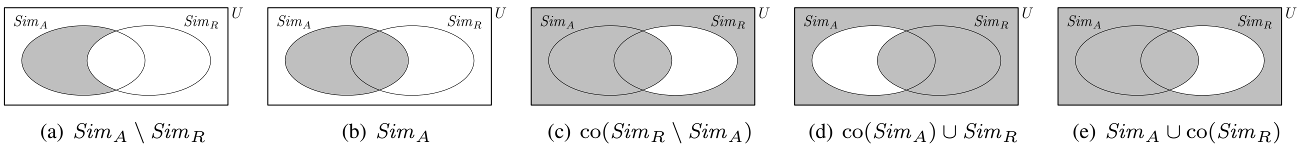


Figure 3. Two-dimensional histograms for the (similarity with accepted tracks, % listened) and (similarity with rejected tracks, % listened) pairs corresponding to the considered patterns. Darker regions contain more pairs, and the thick black lines were obtained using linear regression.

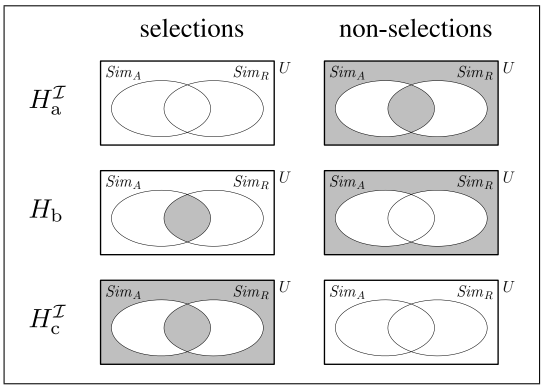
5. INCONSISTENT USER PREFERENCES

With each pattern considered for our basic evaluation, we can associate two pairs of the form (similarity with accepted tracks, % listened), one for the right choice and another for the wrong one. Similarly, we can also associate two pairs of the form (similarity with rejected tracks, % listened) with each pattern. Fig. 3 shows the distribution of these pairs for the considered patterns. The two thick black lines in this figure are the linear regression lines, i.e., the best-fitting straight lines through all of the points in terms of least squares. As illustrated by these regression lines, users apparently tend to avoid songs that are similar to the skipped tracks in favor of the ones similar to the tracks that were not skipped, which is the main assumption behind the dynamic heuristics discussed in this paper. However, the regression lines are only slightly tilted, suggesting that the user preferences are often driven by reasons unrelated to the (computed) similarity with the accepted or rejected tracks. We say such preferences are *inconsistent*, and distinguish the resulting *inconsistent skipping behavior* into two categories: (i) an *inconsistent accept* occurs when an accepted song is either similar to a rejected track, or not similar to any of the accepted ones, and (ii) an *inconsistent reject* occurs when a rejected song is similar to an accepted track or not similar to any rejected tracks. In the context of a radio station, for instance, an eclectic user might not mind when a song is not similar to any of the already accepted songs, leading to an inconsistent accept. On the other hand, the user might reject a particular track because she happens to dislike the corresponding artist for certain (unmeasurable) reasons, even though the track is very similar to the already accepted songs, resulting in an inconsistent reject which might in turn lead to inconsistent accepts, since the user is likely to accept songs that are similar to



Sim_A
Sim_R
U
(a) $Sim_A \setminus Sim_R$
(b) Sim_A
(c) $\mathrm{co}(Sim_R \setminus Sim_A)$
(d) $\mathrm{co}(Sim_A) \cup Sim_R$
(e) $Sim_A \cup \mathrm{co}(Sim_R)$

Figure 4. The dark areas in these Venn diagrams depict the main set-theoretic expressions considered in this paper.



selections
non-selections
$H^{\mathcal{I}}_{\mathrm{a}}$
H_{b}
$H^{\mathcal{I}}_{\mathrm{c}}$
Sim_A
Sim_R
U

Figure 5. The inconsistent selections and non-selections area for all considered heuristics.

the inconsistently rejected track.

Now, by thinking of the fuzzy sets as if they were crisp sets, we can intuitively determine how well the preselection expressions from the heuristics comply with inconsistent user preferences. A song u from U selected by a crisp preselection expression P can lead to an inconsistent accept when either $u \notin Sim_A$ or $u \in Sim_R$. Hence, the area corresponding to potential inconsistent accepts for a preselection expression P is the intersection of P with the set-theoretic expression shown by Fig. 4(d). We call this area the *inconsistent selections area*. Similarly, we can define the *inconsistent non-selections area* as the intersection of $\mathrm{co}(P)$ and the expression shown by Fig. 4(e). The larger the inconsistent selections area, the better the preselection expression complies with inconsistent accepts, and the larger the inconsistent non-selections area, the better it complies with inconsistent rejects.

Fig. 5 shows the inconsistent selections and non-selections area for all considered heuristics. Judging from this figure, $H^{\mathcal{I}}_{\mathrm{a}}$ should perform best when inconsistent rejects occur more frequently than inconsistent accepts, $H^{\mathcal{I}}_{\mathrm{c}}$ is expected to perform best when inconsistent accepts are more common, and $H^{\mathcal{I}}_{\mathrm{b}}$ should perform similarly under both circumstances. In order to verify these theoretical insights, we conducted some additional evaluations.

6. ADDITIONAL EVALUATIONS

By disregarding some of the extracted patterns, we can control the level of inconsistent accepts and rejects. As illustrated by Fig. 6(a), for example, the relative number of inconsistent accepts can be increased by ignoring all patterns for which either $s_{\mathrm{A}} - l_{\mathrm{A}} > 0.6$ or $1 - s_{\mathrm{R}} - l_{\mathrm{R}} > 0.6$ holds, with $(s_{\mathrm{A}}, l_{\mathrm{A}})$ and $(s_{\mathrm{R}}, l_{\mathrm{R}})$ a (similarity with accepted tracks, % listened) and a (similarity with rejected tracks, % listened) pair, respectively, corresponding to the pattern in question. Disregarding patterns in this way actually removes inconsistent rejects, but this effectively leads to a higher percentage of inconsistent accepts in the obtained dataset. Increasing the level of inconsistent rejects can be done analogously. By considering several cut-off values, we generated 9 different datasets that gradually move from a high level of inconsistent accepts to a high level of inconsistent rejects, as illustrated by Fig. 6.

We conducted the basic evaluation for every generated dataset, which led to the plots shown in Fig. 7. In accordance with Fig. 5, the dash-dotted line is below the dashed one in the left part of each of these plots, whereas it is always above the dashed one in the right part. The solid line, on the other hand, is roughly symmetrical along the dotted vertical divider, which also complies nicely with Fig. 5. Although their magnitudes vary a lot depending on the used value for the implicator $\mathcal{I}$, the differences in performance are clearly visible in each subfigure, confirming the insights we obtained by analysing the formal definitions of the heuristics.

Now that we linked the performance of the heuristics to inconsistent user preferences, we can finally explain why the failure rate for the best performing instance of $H^{\mathcal{I}}_{\mathrm{c}}$ is significantly smaller than those for all instances of $H^{\mathcal{I}}_{\mathrm{a}}$ in Fig. 2. The reason for this is simply that the full collection of extracted listening patterns contains more inconsistent accepts than inconsistent rejects, which can easily be demonstrated by reducing the granularity of the two-dimensional histograms from Fig. 3 and summing up the counts for certain bins. For instance, we can get a rough idea of the number of inconsistent accepts by considering merely four bins and summing up the counts for the bins highlighted in Fig. 8(a). Similarly, we can roughly determine the number of inconsistent rejects by summing up the counts for the bins highlighted in Fig. 8(b). The following numbers were obtained in this way: 1,222,094 inconsistent accepts and 1,186,155 inconsistent rejects. Moreover, the dataset illustrated by Fig. 6(a) consists of 554,614 patterns, while the one corresponding to Fig. 6(e) is made up of only 440,171 patterns. Hence, the original dataset indeed seems to contain more inconsistent accepts than inconsistent rejects. The difference is not that large, however, which explains why there is only a very small gap between the performance of $H^{I_{S_{\mathrm{L}}}}_{\mathrm{c}}$ and $H^{I_{S_{\mathrm{P}}}}_{\mathrm{a}}$ in Fig. 2.

Note that Fig. 7 also illustrates that $I_{S_{\mathrm{L}}}$ can be seen as a balanced compromise between the extremes $I_{S_{\mathrm{M}}}$ and $I_{T_{\mathrm{M}}}$. For the other implicators, the measured performance tends to vary a lot for different heuristics, but $I_{S_{\mathrm{L}}}$ rarely leads to significantly worse performance than any of the other considered implicators. In Fig. 2 as well in as all empirical

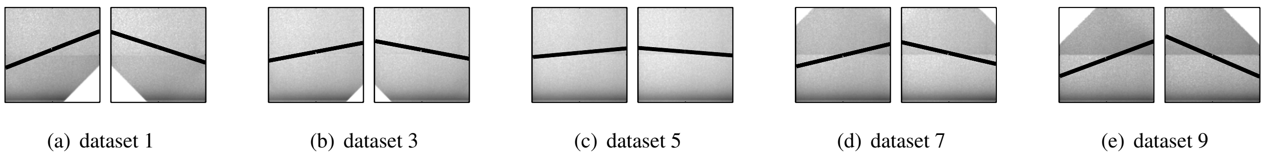


Figure 6. Two-dimensional histograms that illustrate how the 9 generated datasets gradually move from a high level of inconsistent accepts to a high level of inconsistent rejects.

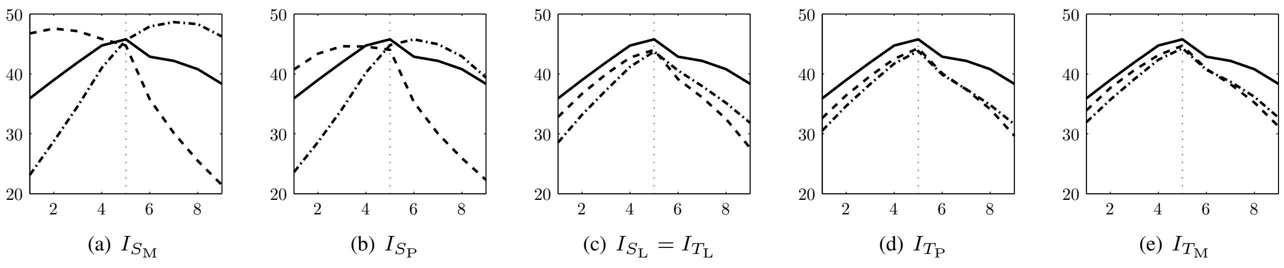


Figure 7. Results of the additional evaluations for $H_\mathrm{a}^\mathcal{I}$ (- -), H_b (–), and $H_\mathrm{c}^\mathcal{I}$ (-·-). The numbers along the horizontal axis are dataset identifiers, while the vertical axis shows failure rate percentages.

results described in [8], $H_\mathrm{a}^{I_{S_\mathrm{L}}}$ and $H_\mathrm{c}^{I_{S_\mathrm{L}}}$ perform at least as well as all other instances of $H_\mathrm{a}^\mathcal{I}$ and $H_\mathrm{c}^\mathcal{I}$, respectively.

7. CONCLUSION AND FUTURE WORK

The mathematical apparatus from the theory of fuzzy sets proves to be very convenient for defining dynamic playlist generation heuristics. Using the described fuzzy framework, we obtained definitions that are not only systematic and both concise and precise, but also intuitively clear and easy to analyse. We relied on this latter benefit to relate the performance of the considered heuristics to inconsistent user preferences. More precisely, we established that $H_\mathrm{a}^\mathcal{I}$ performs best when inconsistent rejects occur more frequently than inconsistent accepts, that $H_\mathrm{c}^\mathcal{I}$ can be expected to perform best when inconsistent accepts are more common, and that $H_\mathrm{b}^\mathcal{I}$ performs similarly under both circumstances. We clearly confirmed these theoretical insights by means of a new methodology for evaluating playlist generation heuristics based on listening patterns extracted from radio logs, which allowed us to conduct accurate experiments using massive amounts of data.

Since we mainly focussed on comparing the heuristics with each other in this paper, it still remains largely unclear to what extent they can improve the performance of a particular playlist generation system. Future work should try to measure the performance impact of the considered heuristics on specific playlist generations systems, and compare them with potential alternatives. In order to obtain a fairer comparison, the underlying fuzzy relation M could then be based on a more advanced similarity measure than simple tag-based cosine similarity.

8. ACKNOWLEDGEMENTS

The first author would like to thank the Fund for Scientific Research–Flanders (FWO) for funding this research.

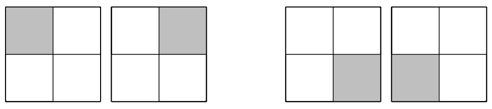


Figure 8. Categorization of certain bins from the coarse-grained two-dimensional histograms.

9. REFERENCES

[1] B. Logan. Content-based playlist generation: Exploratory experiments. In *Proc. ISMIR Intl. Conf. on Music Info. Retrieval*, 2002.

[2] E. Pampalk, T. Pohle, and G. Widmer. Dynamic playlist generation based on skipping behavior. *Proc. ISMIR Intl. Conf. on Music Info. Retrieval*, 2005.

[3] B. Logan. Music recommendation from song sets. In *Proc. ISMIR Intl. Conf. on Music Info. Retrieval*, 2004.

[4] H. Blanken, A. de Vries, H. Blok, and L. Feng, editors. *Multimedia retrieval*. Springer, 2007.

[5] S. Pauws and B. Eggen. PATS: Realization and user evaluation of an automatic playlist generator. In *Proc. ISMIR Intl. Conf. on Music Info.Retrieval*, 2002.

[6] K. Hoashi, K. Matsumoto, and N. Inoue. Personalization of user profiles for content-based music retrieval based on relevance feedback. In *Proc. ACM Intl. Conf. on Multimedia*, 2003.

[7] M. Mandel, G. Poliner, and D. Ellis. Support vector machine active learning for music retrieval. *Multimedia Systems*, 12:3–13, 2006.

[8] K. Bosteels and E. Kerre. A fuzzy framework for defining dynamic playlist generation heuristics. *Fuzzy Sets and Systems*. To appear.

[9] L. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[10] E. Klement, R. Mesiar, and E. Pap. *Triangular norms*. Kluwer Academic Publishers, 2000.