A multicriteria ant colony algorithm for generating music playlists

Jose A. Mocholi, Victor Martinez, Javier Jaen, Alejandro Catala
ISSI - Department of Information Systems and Computation
Universidad Politécnica de Valencia
Camino de Vera s/n, 46022 Valencia, Spain
+34 96 387 35 69
{jmocholi, vmartinez, fjaen, acatala}@dsic.upv.es

ABSTRACT

In this paper we address the problem of music playlist generation based on the user-personalized specification of context information. We propose a generic semantic multicriteria ant colony algorithm capable of dealing with domain-specific problems by the use of ontologies. It also employs any associated metadata defined in the search space to feed its solution-building process and considers any restrictions the user may have specified. An example is given of the use of the algorithm for the problem of automatic generation of music playlists, some experimental results are presented and the behavior of the approach is explained in different situations.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval - *information filtering*. I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search - *heuristic methods*.

General Terms

Algorithms, Recommender Systems.

Keywords

Ant Colony Optimization, Music Playlist Generation, Context-awareness, Ontologies, Semantic Search.

1. Introduction

We live in the so-called *information society*, which is a natural consequence of the general advance in technology and especially of the development and growth of the Internet since its birth in the early 90s. Users are now faced with the challenges involved in learning to apply the sources of knowledge generated by the new model of society and to evolve according to its requirements. In dealing with these challenges, they have to handle an information overload that makes it difficult to select the right information and are forced to perform the cumbersome task of exploring excessively dense spaces, an impossible task without the assistance of intuitive and efficient searching tools.

In order to overcome this problem, *recommender systems* were developed in the 90s to facilitate the automatic and personalised selection of the products that best match individual user preferences. They have become an important research area [3] since they are problem-rich, have abundant practical applications in helping users to deal with the information overload and provide them with personalized recommendations, content, and services. A good example of these applications is Amazon.com [4], where customers receive recommendations on books, CDs and other products.

Recommender systems follow two main recommendation strategies: *content-based* and *collaborative filtering*. Content-based filtering relies on descriptions of the items that are being recommended [6] while collaborative filtering is based on similar past tastes and preferences [7]. There are also hybrid approaches that combine collaborative and content-based methods [3]. However, all these approaches have questionable issues or limitations. When it comes to evaluating the relevance of a set of available products, traditional recommender systems analyze user preferences, which must be properly modeled and stored in a user profile by means of more or less sophisticated mechanisms, none of them free from deficiencies. Generally, the most straightforward approaches simply perform syntactic comparisons against a pre-established set of keywords and hence suffer from the usual shortcomings of mechanisms that do not use semantics to consider the meaning of terms such as synonyms, multiple meanings, etc. Some proposals, based on automatic classifiers, evaluate the relevance of a certain product for a certain user through the use of occurrence patterns of the product attributes or characteristics on a pre-defined training set. Others disregard all product descriptions as a criterion and only consider the associated *rating* (level of interest) previously defined by the user or by other users of the system with similar preferences. For an extended review of common issues and limitations of recommender systems, see [3].

As new technological breakthroughs become mainstream, the amount of information available to the information society just keeps on growing. This has been estimated as 34 gigabytes for an average person on an average day [5] only in the USA. In other words, the information overload issue is not getting any better. For example, the amount of recorded music of all types available at any online music store (e.g. iTunes, Amazon.com, etc.) exceeds the average life expectancy. However, as all the music has been annotated with metadata (i.e., valuable knowledge) and the amount of music available will go on growing, we find two requirements that any recommender system should address in this domain: firstly, it has to make use of the available metadata that describe the items and, secondly, it has to be able to deal with large collections of items, as users tend to add more items to their collections as storage capacity increases. There is also a clear requirement for an effective recommender system: it must be capable of automatically creating a list of recommendations that match the criteria or preferences specified by the user.

In this paper we propose an *ant colony optimization* (ACO) algorithm for use as a recommender system for the automatic generation of music playlists that fulfils the user's predefined set of criteria. It has been designed to cover the two requirements specified above: i.e. it supports multi-criteria recommendation and guarantees good performance when dealing with large sets of items. The paper is organized as follows: in Section 2 we review similar studies. Section 3 gives an overview of ACO, provides a general definition of the problem and describes the functioning of the proposed algorithm. Section 4 presents the domain-specific problem used to test the algorithm, explains the terms of the ontology used and discusses the results obtained from the execution of the algorithm with different sets of criteria. In Section 5 we present our conclusions and describe future work.

2. Related works

In this section we describe approaches from the literature focused on generating music playlists, including both work based on traditional recommender system strategies (content-based and collaborative filtering) that do not allow users to specify arbitrary criteria or constraints and work that does allow this type of specification.

As stated above, content-based filtering techniques look at music attributes such as tempo, pitch and volume and then select songs that conform to these attributes. The idea is that if a user likes a particular song he will also like another similar to it in terms of these attributes. Examples of this approach can be found in [8] [9] [10]. Input from the user usually consists of one or more seed songs for which the system then outputs a set of songs. Generally, these systems disregard the multiple metadata associated with the song, so that the generated playlists only contain songs that are very similar in terms of their audio signal features, with the result that these approaches do not bother to sort the songs in the playlist. On the other hand, collaborative filtering is a multi-user approach: it uses other users' explicit preferences to match songs to a specific user. Each user expresses his preferences for a set of songs and the system then tries to expand this set of preferred songs by finding users who have a similar taste and recommending these users' music preferences. In [11], [12], and [13], three collaborative music playlist filtering systems are described that function only on the basis of user likes and dislikes. In a nutshell, both strategies share a common problem whenever the system has to deal with a new user: he has not rated enough songs to create a viable user model to make the recommender system fully effective.

With regard to approaches that allow the specification of constraints, we have to point out the work done in [14], a study in which the authors present a network flow approach to playlist generation where each node in the network represents a song, and costs and weights are associated to each arc in order to represent the constraints that have to be satisfied. The goal is to find a continuous path that links the first and last song in the playlist, that has a minimum cost and at the same time has constrained weights. Only two types of constraint are incorporated in the network model: absolute constraints and coherence constraints. The former are a maximum and a minimum percentage of each attribute (e.g. between 60 % and 75 % of the playlist items should have a certain attribute), and are represented by the weights associated with the arcs in the network model. Coherence constraints, on the other hand, enforce a certain correlation between successive songs in the sequence to ensure that they are similar and are represented by the costs associated with the arcs in the model. Because of the binary song representation, the constraints that can be specified by using these costs and weights of the network are quite simple. To solve the flow problem in playlist generation, it is transformed into an integer linear program which is solved by branch and bound. In the worst case, branch and bound is an exponential algorithm which is in conflict with our scalability requirement. This approach also requires the length of the playlist to be defined in advance by specifying the number of songs to be included in the list, which we believe is an undesirable limitation.

In [15] and [16] the authors use constraint satisfaction programming (CSP). The rationale behind this approach is to associate a constrained variable with each position in the playlist, where the domain of each of the variables is the music collection. The idea is to assign a value from the domain to each of the variables, which corresponds to a sequence of songs. In this approach the number of songs in the playlist is fixed by defining the variables in the CSP, which is something users may not want. The solution searching algorithm consists of reducing each domain in turn until it either leads to a solution or fails to find one. Every time a value is removed from a domain, the constraints acting on this domain are inspected to check if any constraint forbids another value belonging to another domain. If this occurs, it is also removed and the process is repeated recursively. This process is known as *constraint propagation*. Domain reduction then occurs either by deliberate choice (assigning a value to a variable), or as a consequence of the propagation required for the satisfaction of certain constraints. Upon failure (a domain becomes empty), the algorithm backtracks to the last decision taken, chooses a different one and tries again. If there is still no solution, the algorithm backtracks one step further and examines the stack of decisions bottom-up until it can either prove that there is no solution or finds one. The solution playlist is therefore constructed by repeatedly adding songs to the playlist and testing whether the playlist can still be completed. We thus only have a partial playlist at our disposal at each iteration, which may be a problem if the algorithm is stopped prematurely. This CSP algorithm has the same worst case complexity as the branch and bound algorithm described above, so again scalability is a concern.

In [2] the authors use the same model as in [15] and represent playlist generation as a CSP. However, they use a local search (LS) method, which starts with a random complete assignment of songs to the variables (playlist positions) and iteratively tries to improve upon this playlist by making small changes to it. In order to achieve this behaviour, they introduce a cost associated with the playlists: the more constraints that are violated by the playlist and the larger the violation is, the higher the cost. Consequently the best playlist generated is a complete assignment of the variables at each iteration. Since the LS algorithm does not investigate all possible playlists, but rather progressively refines solutions, it also scales up better than the approach defined in [15]. Another positive aspect is the fact that, instead of choosing a random initial playlist, we are able to use a playlist from a previous iteration of the system as a starting point to guide the algorithm to a final playlist. The downside of this approach is that it represents playlist generation as a constraint satisfaction problem, which means that the number of songs in the playlist is still fixed and also the modifications to the playlist are limited to changing the song that is at a single position in the playlist.

As we want users to be able to tailor the playlist to their likes/needs as far as possible, we propose here a recommender system algorithm that is able to deal with any arbitrary set of constraints specified by the user. However, to avoid the scalability problems discussed above with the existing approaches, we opted to look for a heuristic approach similar to that presented in [1], in which the authors evaluate the use of traveling salesman algorithms to organize a music collection into a large circular playlist that satisfies the constraint that, on average, consecutive tracks should be maximally similar. This means that the generation of the playlist is mapped to the *Travelling Salesman Problem* (TSP). The cities (i.e. nodes of the graph) therefore correspond to the tracks in the collection and the distances (edges) are the similarities between the tracks. Finding the optimal route means producing a circular playlist that contains all the tracks in the collection in which the sum of the similarities along the path is maximized. For this, a distance value representing the audio similarity of each pair of tracks is calculated. As in principle it only has to be executed when new music records are added to the collection, its performance can be considered as good. However, our aim is to solve the playlist generation problem for arbitrary sets of constraints given by the user whenever he requests a music playlist, while maintaining scalability and good performance. For this reason, and following on our previous studies, [17] [18], which incorporate good scalability features, we based our approach on the ant colony optimization metaheuristic. In addition, as it is not our goal to create playlists containing all the items in the collection, instead of mapping the playlist generation problem to the TSP we opted for the *Orienteering Problem* (OP), as explained below.

3. Semantic ACO algorithm for the Orienteering Problem

3.1 Ant Colony Optimization overview

Ant colonies are insect societies that accomplish complex tasks by presenting highly structured organizations and communication mechanisms. Ants' behaviour is based on the use of pheromones, including the trail pheromone, which is particularly interesting since it is used to mark paths on the ground from food sources to the nest. Several experiments [19] have shown that this communication mechanism is very effective in finding the shortest paths and has thus inspired several stochastic models [19] that imitate the dynamics of these colonies. Based on this natural behaviour and the proposed stochastic models, Dorigo and his colleagues [20] first introduced Ant Colony Optimization (ACO) as a metaheuristic that targets combinatorial optimization problems. ACO algorithms make use of artificial ants, which are stochastic constructive procedures that build step-by-step solutions following a decision process that is based on shared information on pheromone trails. ACO algorithms have been used to solve routing, assignment, scheduling, subset, machine learning and routing problems [21].

In ACO, *m* independent ants obtain *m* different solutions (tours) by iteratively choosing the next node to be appended to their respective tour. The selection of the next node to visit is carried out either by randomly exploring the search space or by applying a probabilistic state transition rule that takes into account the attractiveness of the nodes and the pheromone trails that have been deposited by other artificial ants (see [22] for a mathematical definition of this process). Once *m* solutions have been obtained, the best tour is selected, pheromone trails are intensified for the graph edges in the winning tour and evaporation is applied to the remaining trails in the graph. It has been demonstrated that, by iteratively proceeding in such a way, the algorithm converges to suboptimal solutions [21]. In terms of our specific problem domain, ants probabilistically select the next album to add to the playlist by using both pheromone information and the local-heuristic of nodes (a trade-off between the score assigned to the album, its length and its metadata). The ACO algorithm can be summarized in a pseudo-code schema as follows:

```
Set all parameters and initialize pheromone trails
Loop
    Sub-Loop
        Construct solutions based on the state transition rule
        Apply the online pheromone update rule (optional)
    Continue until all ants have been generated
    Apply local search (optional)
    Evaluate all solutions and record the best solution so far
    Apply the offline pheromone update rule
Continue until the stopping criterion is reached
```

In mathematical terms, the probabilistic selection rule applies either exploitation by selecting the best node to be added to the tour, the node that maximizes attractiveness and pheromone level as shown in formula (1), or exploration by selecting nodes randomly according to a probability distribution function shown in (2)

$$if \quad q \le q_0$$

$$p_{ij}^{k} = \begin{cases} 1, & if \ j = \arg\max\limits_{u \in \mathfrak{A}_i^k} \left\{ [\tau_{iu}]^{\alpha} \cdot [\eta_{iu}]^{\beta} \right\} \\ 0, & otherwise \end{cases} \qquad (1)$$

$$else$$

$$p_{ij}^{k} = \begin{cases} \dfrac{\left[\tau_{ij}\right]^{\alpha} \cdot \left[\eta_{ij}\right]^{\beta}}{\sum_{u \in \mathfrak{A}_i^k} \left[\tau_{iu}\right]^{\alpha} \cdot \left[\eta_{iu}\right]^{\beta}}, & if \ j \in \mathfrak{A}_i^k \\ 0, & otherwise \end{cases} \qquad (2)$$

where η_{ij} is a problem-specific local heuristic, τ_{ij} are the pheromone trails, $\mathfrak{A}_i^k$ the set of unvisited nodes and α, β are parameters that control the relative influence of the pheromone and local heuristic values respectively.

The pheromone update rule consists of two steps, an online updating to decrease the level of pheromone of the selected move and make it less attractive for other ants

$$\tau_{ij}^{\ new} \leftarrow (1-\varphi) \cdot \tau_{ij}^{\ old} + \varphi \cdot \tau_0 : \varphi \in (0,1] \qquad (3)$$

and an offline updating where an elitist approach is followed and only the arcs (*i*,*j*) of the best feasible solution found so far are updated as follows:

$$\tau_{ij}^{\ new} \leftarrow (1-\varphi) \cdot \tau_{ij}^{\ old} + \varphi \cdot S_{BEST} : \varphi \in (0,1] \qquad (4)$$

ACO also exhibits a design that can be easily parallelized, since colonies can be seen as individual computational nodes. This allows approaches based on this type of algorithms to solve complex problems (including NP-hard ones) while achieving a good trade-off between scalability and performance, and the optimality of the obtained solution, as shown in other studies [17] [27].

3.2 Orienteering problem overview

The OP was originally modelled for the sport of orienteering, in which the participants have to visit as many checkpoints as possible while navigating through unfamiliar country, aided only by a map and a compass. Since points are awarded not only on the basis of the time needed to reach the finish, but also on the number of checkpoints visited, competitors have to find some kind of trade off between the time used and the distance covered in order to maximize the checkpoints visited.

The OP can be modelled as follows: given a graph with n nodes in the Euclidean plane and assigning each node *i* a certain score $S_i \geq 0$ and the scores of the starting node denoted by *1* and the ending node denoted by *n*, set to 0; $S_1 = S_n = 0$. Each node can be visited at most once, while every path between two nodes is associated with a cost c_{ij}. The objective of the OP now is to optimize the score of a route without violating the cost (time/distance) constraint *Tmax*. The mathematical model of the OP is generally formulated as follows:

$$Max \sum_{i=1}^{n} \sum_{j=1}^{n} S_i x_{ij} \qquad (5)$$

Subject to:

$$\sum_{j=2}^{n} x_{1j} = \sum_{i=1}^{n-1} x_{in} = 1, \qquad (6)$$

$$\sum_{i=2}^{n-1} x_{ik} = \sum_{j=2}^{n-1} x_{kj} \leq 1, \quad k = 2, \ldots, n-1 \qquad (7)$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \leq T_{\max}, \qquad (8)$$

$$x_{ij} \in \{0,1\}, \quad i, j = 1, \ldots, n \qquad (9)$$

where x_{ij} are binary variables to indicate the selected arcs in the graph as part of a solution, and restrictions (6) and (7) ensure that no internal cycles will be present.

Despite having a simple definition, the OP belongs to the category of NP-Hard problems, i.e. there is no known algorithm that can obtain optimal solutions to this problem in polynomial time. Therefore, exact methods for the OP, such as those proposed by [23], [24], [25], [26], can only be applied to the simpler instances of the problem. However, methods that use ant colonies to solve the OP have been demonstrated [22] to obtain suboptimal solutions with better response times.

3.3 SACO-OP: An Ontology based ACO algorithm for the OP

Our approach, denoted here as *SACO-OP* (Semantic Ant Colony Optimization for the Orienteering Problem), is based on previous work [28], [29]. It carries out the comparison between queries and documents via conceptual distance measures [30] and searches multimedia audio databases of user requests by utilizing a *domain-specific ontology*. Ontologies are a specification of an abstract view of the world that is represented for a specific purpose [31]. Ontologies therefore define a set of representational terms called *concepts* which are interrelated to describe a target world.

SACO-OP is different from ACO-OP in that the former uses semantic descriptions of concepts to dynamically assign scores to nodes representing problem domain entities in a process called *semantic score assignment*. In addition, this process is not performed in a static way and only once for every user according to his profile, but instead is a dynamic process that considers the restrictions defined by the user.

Restrictions are defined in our approach by means of assigning relevance factors to elements in the ontological space. The goal of the relevance factor is to adjust the level of importance of certain ontology points over the rest within the ontological space. Based on this specification of relevance factors, the semantic score assignment process assigns scores to each node as follows:

- Any item in the collection is defined as a tuple (ID_i, O_i) where ID_i is a unique identifier, and O_i is a set of pairs $\{(term_k, value_k) \mid term_k \in T, value_k \in Domain(term_k)\}$ representing values associated to terms in a set T of a predefined ontology.
- A user restriction set R is defined as a set where $R=\{(term_j, value_j, relevance_j) \mid term_j \in T, value_j \in Domain(term_j), relevance_j \in [0,100]\}$.
- A distance metric over a term $term_k \in T$ is defined as a function d_{termk} *: Domain(term$_k$) x Domain(term$_k$)* $\rightarrow [1,100]$
- Given a user restriction set R, a collection of distance metrics over T, and given the set M of matching pairs $M=\{(term_j,value_j,) \in R \mid \exists\ (term_j,value_i,) \in O_i\}$, the score S_i associated to a node of the graph representing an item (ID_i, O_i) of the collection is calculated as follows:

$$S_i = \sum_{j=1}^{|M|} \frac{d_{term_j}(value_j, value_i) \cdot relevance_j}{|M|} \qquad (10)$$

In the above formalization, if M is the empty set, then S_i=1, so that the transition probability function defined in (7) can be correctly obtained. Note that the score is proportional to the conceptual distance between the terms expressed in the query and the metadata description in O_i. In the current implementation, the collection of distance functions is usually obtained by the manual intervention of an expert user, who is able either to establish a conceptual fixed distance between different elements in the ontology or to create an evaluation function to calculate the distance between two values of a term of the ontology. Given this definition for obtaining the score associated to each node in the problem space, the ACO local heuristic when applying exploitation is as follows: $\eta_{ij} = \frac{S_j}{c_{ij}}$. This means that nodes with a better ratio score/cost are more attractive to ants when exploiting pheromone information and will be selected during the solution building process.

As a result of applying a restriction on the items of a collection annotated with the terms of a certain ontology, we obtain a graph that represents the search space to be used by our SACO-OP algorithm. In this way, our approach can ensure both that the generated solutions will closely match the preferences specified by the user and good performance by reducing the problem search space, since restrictions help us to select the items which become relevant graph nodes in this first step. In the second step, restrictions are also used to assign a score to each node using formula (10). Edge costs are calculated from the metadata terms of the linked nodes.

All the ontology terms in the restriction are used to obtain the search space and create the graph and its nodes. Thus, given a restriction set *R* that may be represented as follows:

$$R = \{(\mathtt{term}_1,\mathtt{value}_1,\mathtt{r}_1),(\mathtt{term}_2,\mathtt{value}_2,\mathtt{r}_2),\ldots,(\mathtt{term}_n,\mathtt{value}_n,\mathtt{r}_n)\}$$

in order to create the nodes, we take the set of query terms $\{\mathtt{term}_1, \mathtt{term}_2,\ldots, \mathtt{term}_n\}$ and select from the user collection only those that have at least one of the restriction set terms among its metadata. For each one of these selected items our algorithm will create a graph node that will have, as associated metadata, all the pairs of *(term, value)* for every term appearing in the query. With this process all non relevant items are omitted and the search space is therefore reduced. After this process, the ACO algorithm can be executed as scores, and costs in the graph can be calculated from the nodes' associated metadata. The SACO-OP algorithm may be formulated as follows:

```
SACO_OP
Load ontology
Read restriction set
Select and create nodes for the search space according to restriction set terms
Perform nodes score assignment process
Apply ACO-OP algorithm
END SACO-OP
```

The conceptual model underlying our approach can be seen in Figure 1, which also includes the concepts defined above. The kind of available concepts defined by the ontology in use are represented by the Term class, and each one of its values is represented by objects of the TermValue class. As stated above, a set of terms is used not only to represent the metadata of each node, but also to build up the restriction set specified by the user in order to establish the characteristics of the solution they want to obtain. Each term has an associated Relevance factor. Additionally, as can be seen in Figure 1, each Term is related to some evaluation functions (EvaluationFunction class). Each of these will be used during the process of calculating each node score. There may also be other evaluation functions to compute distances between instances of TermValue. Regarding the evaluation functions, Figure 1 also shows that the system is extensible by using polymorphism, so that different evaluation functions may be defined. This is a key characteristic of the approach: since it will have to deal with domain-specific problems, it should allow domain experts to define suitable evaluation functions.

Figure 1. Conceptual model of SACO-OP.

4. Generating a music playlist with SACO-OP

In order to validate our proposal, we used an *RDF* (*Resource Description Framework*) knowledge base to represent the information of a music collection. RDF is based on the idea that anything can be identified by using web identifiers known as *Uniform Resource Identifiers* (URIs), and that these resources can be described with simple properties, with values assigned to the properties. The following is an example of the use of RDF to describe an item in a knowledge base built with XML syntax. It describes an album entitled *Ten* recorded by the music group *Pearl Jam*, published in *1991* and classified as *Grunge*.

```
<?xml version="1.0"?>

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:music="http://www.example.org/music#">

        <rdf:Description rdf:about="Ten">

                <music:type>Grunge</music:type>

                <music:artist>Pearl Jam</music:artist>

                <music:year>1991</music:year>

        </rdf:Description>

</rdf:RDF>
```

An example scenario of using SACO-OP with the semantic generation of playlists as the problem domain would be the creation of a playlist for a thematic party. Let us suppose the user wants to select music albums for a party with a 60's theme, for example, with a significant number of songs from *The Beatles, Twist* and *Rock*. In order to achieve his goal the user will simply define a restriction set with these metadata and the playlist will select music fitted to the specified criteria as far as possible. Another example of the usefulness of the automatic generation of playlists to match given criteria or preferences would be the scenario described in [1]: "Imagine the following situation: A keep-fit enthusiast leaves the house early in the morning for his daily jog while listening to music on his mp3. As he doesn't like to listen to the same music over and over again, he chooses the music that fits his mood without having to pause in his workout. While running, he browses through the collection, chooses a track and the player automatically keeps on playing similar tracks."

In the following subsection we will introduce the ontology terms, i.e. the metadata used in the music collection the tests on the SACO-OP algorithm.

4.1 Music records metadata

The music albums are the essential units in the knowledge base used in this study. Unlike other studies that used songs as the basic unit, we used albums since we focused on scenarios such as the one described in the previous section, where users prefer long playlists with specific characteristics. The music collection used in this case study contained 71 albums. It should be pointed out that this is the special approach followed in this case and that the system can use any type of data or granularity.

As terms for the ontology, i.e. metadata, we selected three different attributes to store related information on each album. These were *year* of publication, *music type* and *artist.* The first metadata is the year in which the album was published and ranges from 1966 to 2007 in this case study. The type is classified according to the characteristics, which can be some features such as rhythm or instruments, or can be based on criteria like region of origin, period, or cultural and social aspects. Music types can be divided into subtypes, but in this study the type also includes subtypes. An album may contain several types but, for the sake of simplicity, they are classified as a single type. However, the creation of the search space for a given query would allow an item to have more than one value for the same ontology term among its metadata. Finally, the artist metadata refers to the performer. The artist may also have composed the songs in the album, or may simply have done the mix session, as happens when the artist is a DJ.

As explained in section 3.3, our approach uses evaluation functions to calculate the distance between two values of a certain term of the ontology, in other words, our implemented evaluation functions return a number ranging from 0 to 100 that represents how different two values of the same term of the ontology are. These distance functions are used to compute both the score of a node of the search space and the cost of the edge that represents how similar two nodes are. Therefore, the more similar two values of a term are, the smaller the distance. Consequently, when computing the score of a node for a certain restriction set, the more metadata values associated to the node that are similar to the terms values of the restriction set, the higher the score of the node will be. Here we are again rewarding the conceptual/semantic closeness between the elements in the collection (music records) and the semantic restrictions specified by the user.

The evaluation function for the year of publication metadata is quite simple. Considering the number of years covered by the music collection, it computes the distance by adjusting the difference between two years with the maximum difference of the collection. The definition of the distance functions for the music type and artist ontology terms was established by consulting music experts to simplify the process, although we could also have chosen statistical methods, such as analyzing user shopping behaviours.

5. Experimental Results

To evaluate our proposal we executed SACO-OP with several restriction sets and studied two variables: *frequency* and *saturation* for every solution. The frequency variable represents the percentage of nodes in the final solution whose semantic distance to the restriction is within a certain range. On the other hand, saturation measures the percentage of all the albums whose semantic distance to the restriction is within a certain range present in the final solution. For example, if there are 20 albums with a semantic distance between 50 and 59 for a specific restriction and 5 of these albums appear in the solution, the saturation will have a value of 25%.

All the evaluated restriction sets had the same Tmax value, 1200 minutes. This was to ensure that all the obtained solutions could not be built up only with items that closely adhered to the restrictions. In addition, all the restriction sets were evaluated at least 5 times to obtain a meaningful average value for the studied variables. Besides showing both the frequency and the saturation variables, the average duration of the selected albums will also appear in the graphs, since it is a relevant factor in studying the obtained solutions, as will be explained in the following sections.

5.1 Restriction set Folk = 100

The first restriction set used to evaluate the soundness of the SACO-OP algorithm had only one restriction, the music type should be folk with a relevance factor of 100. It should be noted that the relevance factor is significant only when there is more than one restriction in the set.

As can be observed in Figure 2, frequency is quite similar in both range 0 and range 50-59 (each range represents a cluster of items whose semantic distance to the user query is within certain bounds) and has a value slightly higher than 0.4 in both cases. To understand why the number of albums in both categories is similar in the final solution, even though those in the 50-59 range are semantically more distant, we have to observe the saturation value for range 0, which is close to 100% (see Figure 3). In other words, almost all the folk albums are included in the final solution, and the others are excluded because they violate the 1200 minute restriction. In fact, one of the folk albums has a longer duration than the others. The algorithm can therefore be said to effectively search the problem space adding to the solution albums in the semantic vicinity as long as the restriction is not violated. It can also be observed that as the semantic distance with respect to the restriction increases, the frequency of the appearance of albums in the corresponding range decreases, which is the expected behaviour.

Figure 2. Average frequency of solutions for restriction set (music type = folk, relevance = 100)

Several characteristics of the saturation variable can be seen in Figure 3. As can be observed, the saturation value for the 0-50 range exceeds 90% but does not reach 100%, although the number of albums in this range is around half the number of albums in the obtained solutions. To understand why not all folk albums are included in the final solution, it should be noted that one of the folk albums has a much greater length than the rest. The length of an album also has an impact on the obtained solutions, since the inclusion of shorter albums in the final solution may increase the overall score, i.e. the algorithm considers that it is better (higher benefit/score) to try to fill the available listening time with albums that are not so semantically close, rather than leave a gap in the listening time.

Figure 3. Average saturation of solutions for restriction set (music type = folk, relevance = 100)

5.2 Restriction set: Trance = 100

The second tested restriction set only differs from the previous one in the music type. This test is based on the results obtained in the previous one, to determine the significance of album length when the algorithm is calculating a playlist. As can be seen in Figure 4, albums in range 0-50 have an average length of about 120 minutes, which is significantly higher than the ones in the remaining ranges. This means that, on average, albums in this range have twice the length of other albums. Their scores should therefore be at least double the scores of other songs for the ant algorithm to find them attractive.

After running this experiment we observed that 0-50 range only has 72% saturation (see Figure 5), compared to more than 94% in the previous test. This confirms that our algorithm is not a straightforward greedy approach, but instead tries to find a balance between semantic closeness to the restriction and saturation of the available listening time.

Figure 4. Average frequency of solutions for restriction set (music type = trance, relevance = 100)

Figure 5. Average saturation of solutions for restriction set (music type = trance, relevance = 100)

5.3 Restriction set: Folk = 50; Trance = 50

Once simple restrictions consisting of a single term had been tested, we tested the soundness of our approach when several restrictions are specified for the same ontology term, both with the same relevance factor. The selection of these two specific music types is due to the large number of albums in the collection and also because the semantic distance between these categories is 95, i.e. Folk albums should appear in the 90-100 range when evaluated against the Trance restriction and vice-versa. Knowing in advance the semantic distance between these terms will help us when observing the behaviour of our algorithm and verifying its soundness when multiple categories of the same term are included in the restrictions set.

In order to study the results we will show frequency and saturation values from the point of view of the Folk type, so that distance ranges will be constructed with respect to this dimension. It may be observed in Figure 6 that, although similar frequency results are obtained as when folk music was the only restriction, there is a relevant difference. The frequency of all ranges close to the Folk restriction suffered a significant reduction, including the 0-50 range. The total reduction of 19.7% in frequency was used by the algorithm to increase the frequency of albums in the 90-100 range, in which Trance albums are selected observing the Folk restriction. The reason why this range constitutes only 20% of the final solution is because the duration of albums of this type is significantly higher than in the remaining ranges (see Figure 6). Therefore, as explained above, some other albums (in this case those in the 70-79 range) fill up the available listening time more effectively.

To sum up, the results confirm the behaviour designed for the algorithm: whenever two semantically separated restrictions are considered for the same ontology term and are given equal relevance, our algorithm tries to include elements satisfying both requirements in the final solution, but gives preference to those that have a better score/duration ratio.

Figure 6. Average frequency of solutions for restriction set ((music type = folk, relevance = 50), (music type = trance, relevance = 50)); showing results for Folk restriction

Significantly higher differences were observed in the saturation results than those obtained when studying the frequency factor. In Figure 7 it can be seen that the 0-50 range still has high saturation, only 5 points lower than the saturation obtained for the restriction set Folk = 100. However, nearby ranges suffered a huge reduction, mainly due to the selection of many albums in the 90-100 range. It can also be observed that albums in the 70-79 range have a higher level of saturation. This is explained by the fact that this range has a minimum value in the average duration that can be effectively used to try to fit the final solution to the 1200 minute time limit.

Figure 7. Average saturation of solutions for restriction set ((music type = folk, relevance = 50), (music type = trance, relevance = 50)); showing results for Folk restriction

5.4 Restriction set: Folk = 50; Javier Krahe = 50

The algorithm was subjected to a final test with two restrictions specified but this time for different ontology terms, both with the same relevance factor. In this case the restriction set was composed of a music-type term and an artist term (Folk and Javier Krahe, respectively).

As can be seen in Figure 8, the average frequency results obtained for the different semantic distance ranges are quite similar to those obtained for the restriction set (Folk = 100) (see Figure 2), although with some small differences. Albums with a semantic distance in the 50-59 and 60-69 ranges underwent a slight decrease in frequency, while albums in the 0-50 and 70-79 ranges increased their frequencies. The increase of the albums in the 0-50 range can be explained in the following way: as almost all the Javier Krahe albums are tagged as Folk albums, the score assignment process must assign a slightly higher score to them. The frequency increase of the albums in the 70-79 range can be explained by the low average album length in this range, which allows albums to be selected to fit the solutions to the time restriction.

Figure 8. Average frequency of solutions for restriction set ((music type = folk, relevance = 50), (artist = Javier Krahe, relevance = 50)); showing results for Folk restriction

A general reduction in all ranges was observed in the saturation results from the Folk albums, as can be seen in Figure 9. Although we did observe a frequency increase for albums in the 0-50 range, saturation in this range actually decreased, due to the fact that some of the Javier Krahe albums were longer than average.

Figure 9. Average saturation of solutions for restriction set ((music type = folk, relevance = 50), (artist = Javier Krahe, relevance = 50)); showing results for Folk restriction

Figure 10 gives the frequency results for the artist restriction. Since the collection included a low number of albums from this artist, the frequency of the 0-50 range is low, as expected, but this also means that saturation in this range will be very high. This is confirmed in Figure 11, where the average saturation reaches 95%. Albums from the 50-59 range have a high average frequency value, due to most of the Folk albums by other artists being in this range.

Figure 10. Average frequency of solutions for restriction set ((music type = folk, relevance = 50), (artist = Javier Krahe, relevance = 50)); showing results for Javier Krahe restriction

Figure 11. Average saturation of solutions for restriction set ((music type = folk, relevance = 50), (artist = Javier Krahe, relevance = 50)); showing results for Javier Krahe restriction

6. CONCLUSIONS

In this paper we have dealt with one of the biggest problems that inhabitants of the so-called information society must address. Due to the vast amount of available information (information overload), we are forced to explore excessively dense spaces, turning the selection of interesting information into a cumbersome task, extremely hard to carry out without the assistance of intuitive and efficient searching tools. We believe recommender systems are one of the most significant advances in dealing with this problem. In this study we focused on the specific context of digital music, where users are forced to resort to electronic devices to access their ever-growing digital music collections. In this particular scenario there is a clear requirement for personalization of information and assistance in generating playlists to make the most of digital music collections. This requirement can be defined in terms of the general optimization problem known as the *Orienteering Problem,* which belongs to the category of NP-Hard problems. Obtaining optimal playlists automatically with a large number of albums cannot be done effectively with traditional computing strategies. To overcome this problem and achieve the required high level of personalization, we proposed the utilization of a modified version of an ACO algorithm that is able to tackle both the optimization problem and the personalization requirement. The algorithm presented here can be applied not only to generating music playlists but also, by the use of the appropriate ontology and the definition of evaluation functions, to many other domains where information tagged with descriptive metadata is retrieved while satisfying a specified number of preferences. Other possible uses of the algorithm include generating guided visits to tourist attractions, shopping centres, museums, art galleries, etc.

Our plans for future work include evaluating our algorithm on other domain-specific problems, with larger repositories of items, increasing expressivity of restriction sets by allowing the use of logic operators such as *or*, *and* or *not*, and a test with users to assess the usefulness and acceptance of the system proposed here.

ACKNOWLEDGMENTS

This work has been partially supported by the Spanish Ministry of Education and Science under the funding project CENIT-MIO! CENIT-2008 1019 and by the Microsoft Research Labs (Cambridge) under the "Create, Play and Learn" program.

REFERENCES

[1] Pohle, T., Pampalk, E., Widmer G. Generating Similarity-Based Playlists Using Traveling Salesman Algorithms. Proc. of the 8th Int. Conference on Digital Audio Effects (DAFx'05), Madrid, Spain, September 20-22, 2005.

[2] Aucouturier, J.-J., Pachet, F. Scaling up music playlist generation. In Proc IEEE Intl Conf on Multimedia Expo, 2002

[3] Adomavicius G. y Tuzhilin A. Towards the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6):739-749, 2005.

[4] Linden, G., Smith, B., York, J. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. IEEE Internet Computing, Jan-Feb 2003.

[5] Bohn, R.E., Short, J.E. How Much Information? 2009 Report on American Consumers. Global Information Industry Center, University of California, San Diego. Available at http://hmi.ucsd.edu/howmuchinfo_research_report_consum.php

[6] P. Melville, R. Mooney, and R. Nagarajan. Content-boosted collaborative filltering, In Proceedings of the 2001 SIGIR Workshop on Recommender Systems.

[7] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th International World Wide Web Conference 2001.

[8] S.C. Pauws and B. Eggen. Realization and user evaluation of an automatic playlist generator. Journal of New Music Research, 32(2):179–192, 2003.

[9] J.C. Platt, C.J.C. Burges, S. Swenson, C. Weare, and A. Zheng. Learning a Gaussian process prior for automatically generating music playlists. In Proc. of the 14th Conference on Advances in Neural Information Processing Systems, volume 14, 2001.

[10] A. Flexer, D. Schnitzer, M. Gasser, and G. Widmer. Playlist generation using start and end songs. In International Symposium on Music Information Retrieval (ISMIR'08), pages 173-178, 2008.

[11] N. Kravtsova, G. Hollemans, T.J.J. Denteneer, and J. Engel. Improvements in the collaborative filtering algorithms for a recommender system. Technical Note NL-TN-2001/542, Philips Research, Eindhoven, 2002.

[12] J.C. French and D.B. Hauver. Flycasting: On the fly broadcasting. In DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries, 2001.

[13] C. Hayes and P. Cunningham. Smart radio: Building music radio on the fly. In Proc. of Expert Systems 2000, 2000.

[14] M. Alghoniemy and A.H. Tewfik. A network flow model for playlist generation. In Proc. of the IEEE International Conference on Multimedia and Expo, 2001.

[15] F. Pachet, P. Roy, and D. C azaly. A combinatorial approach to content-based music selection. IEEE MultiMedia, 7(1):44–51, 2000.

[16] S.C. Pauws. Playlist generation by constraint satisfaction. Technical Note PR-TN-2002/283, Philips Research, Eindhoven, 2002.

[17] Catala, A., Jaén, J., and Mocholí, J. A. 2007. Strategies for Accelerating Ant Colony Optimization Algorithms on Graphical Processing Units. In Proc. of the 2007 IEEE Congress on Evolutionary Computation, 492-500.

[18] Javier Jaen, Jose Antonio Mocholi, Alejandro Catala, Elena Navarro. Digital ants as the best cicerones for museum visitors. Applied Soft Computing. ISSN 1568-4946 , DOI: 10.1016/j.asoc.2009.11.002.

[19] Goss, S., Aron, S., Deneubourg, J. L., and Pasteels, J.-M. 1989. Self-organized shortcuts in the Argentine ant. Naturwissenschaften, 76, 579-581.

[20] Dorigo, M., Maniezzo, V., and Colorni, A. 1996. The Ant System: Optimization by a Colony of Cooperating Agents. IEEE Trans. Systems, Man and Cybernetics, Part B, 26, 29-4.

[21] Dorigo, M., Stützle, T. The ant colony optimi-zation metaheuristic: Algorithms, applications and advances. In F. Glover and G. Kochen-berger (eds.): Handbook of Metaheuristics. Kluwer Academic Publishers, 251-285, 2003

[22] Liang, Y.C., Smith, A.E. An Ant Colony Approach to the Orienteering Problem. Journal of the Chinese Institute of Industrial Engineers, Vol. 23, no. 5, 403-414, 2003

[23] Fischetti, M., Gonzalez, J.J.S., Toth, P. Solving the Orienteering Problem through Branch-and-Cut. INFORMS Journal on Computing, Vol. 10, no. 2, 1998

[24] Hayes, M., Norman, J.M. Dynamic Pro-gramming in Orienteering: Route Choice and the Siting of Controls. Journal of the Operational Research Society, Vol. 35, no. 9, 1984

[25] Laporte, G., Martello, S. The Selective Trav-eling Salesman Problem. Discrete Applied Mathematics, Vol. 26, 1990

[26] Leifer, A.C., Rosenwein, M.B. Strong Linear Programming Relaxations for the Orienteering Problem. European Journal of Operational Re-search, Vol. 73, 1993

[27] Mocholí, J. A., Jaén, J., Canós, J. H. 2005. A Grid Ant Colony Algorithm for the Orienteering Problem. In Proc. of the 2005 IEEE Congress on Evolutionary Computation, Vol. 1, 942-949.

[28] A.F. Smeaton, A. Quigley, Experiments on using semantic distances between words in image caption retrieval, in: Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval, 1995.

[29] J. Gonzalo, F. Verdejo, I. Chugur, J. Cigarran, Indexing with wordnet synsets can improve text retrieval, Coling-ACL'98 Workshop: Usage of WordNet in Natural Language Processing Systems, 1998, pp. 38-44.

[30] L. Khan, D. McLeod, Audio structuring and personalized retrieval using ontologies, in: IEEE Advances in Digital Libraries, 2000.

[31] M.A. Bunge, Treatise on basic philosophy: ontology: the furniture of the world, Reidel, Boston, 1977.

Figure 1

Click here to download high resolution image

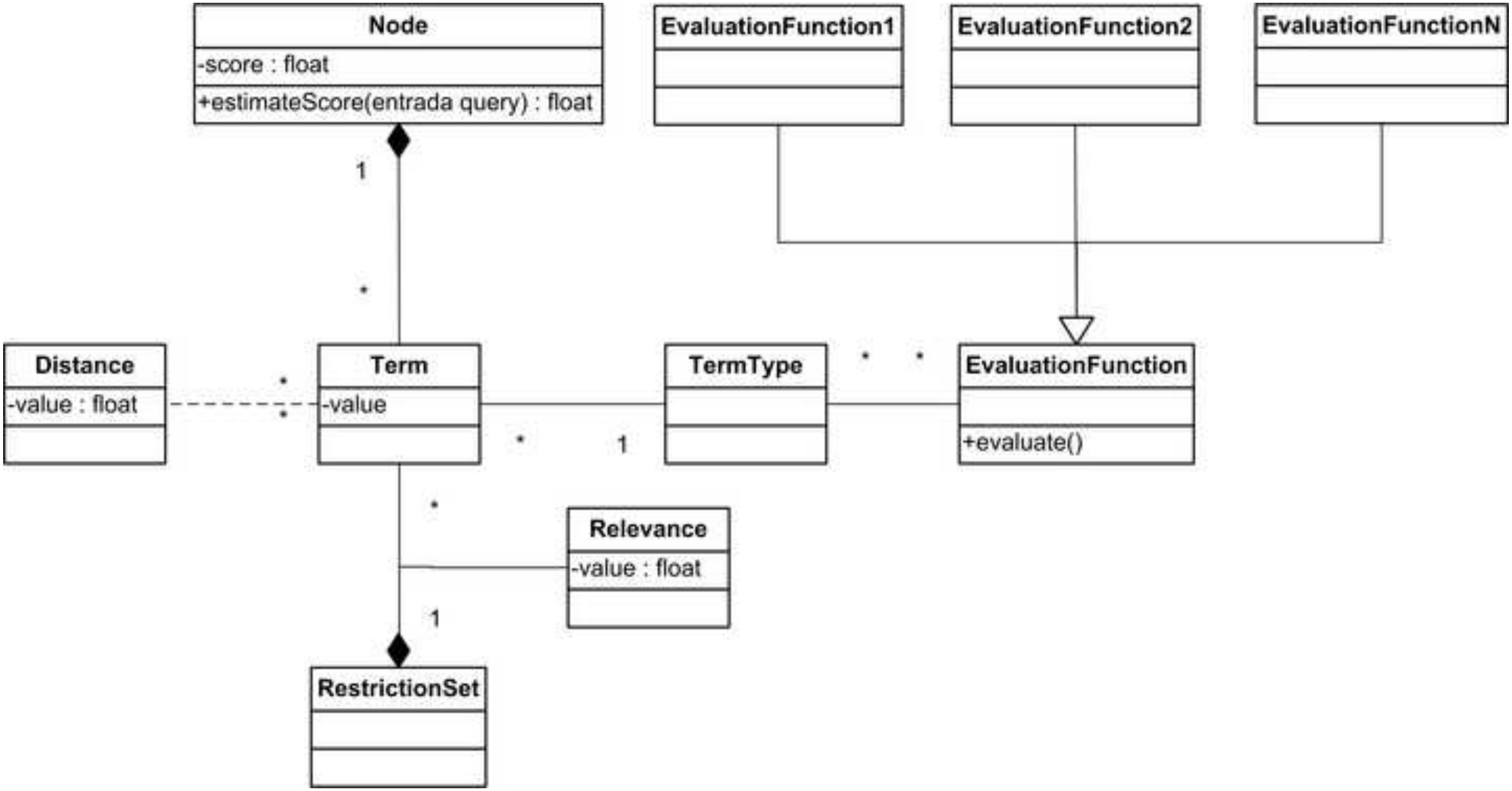
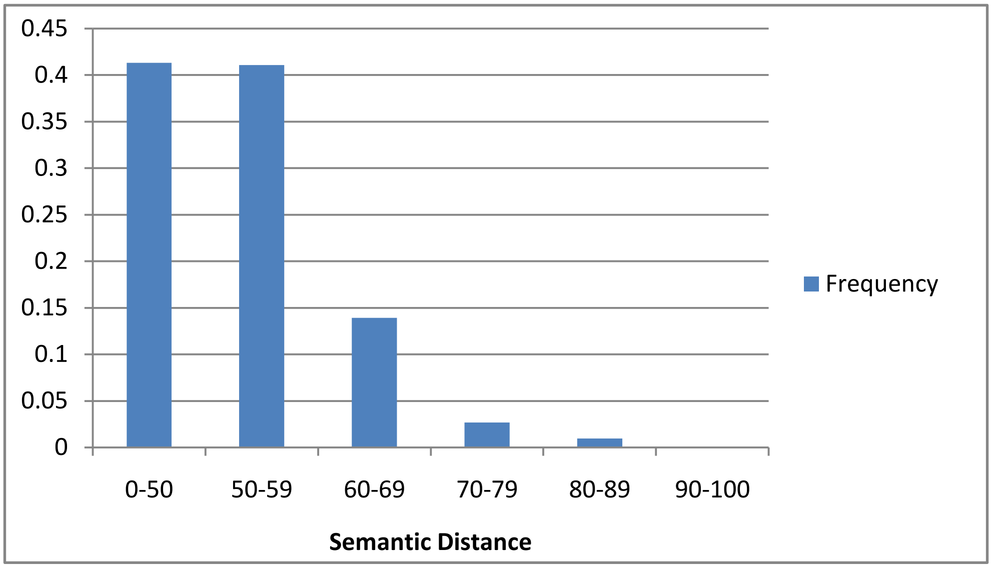


Figure 2



0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0-50
50-59
60-69
70-79
80-89
90-100
Semantic Distance
Frequency

Figure 3

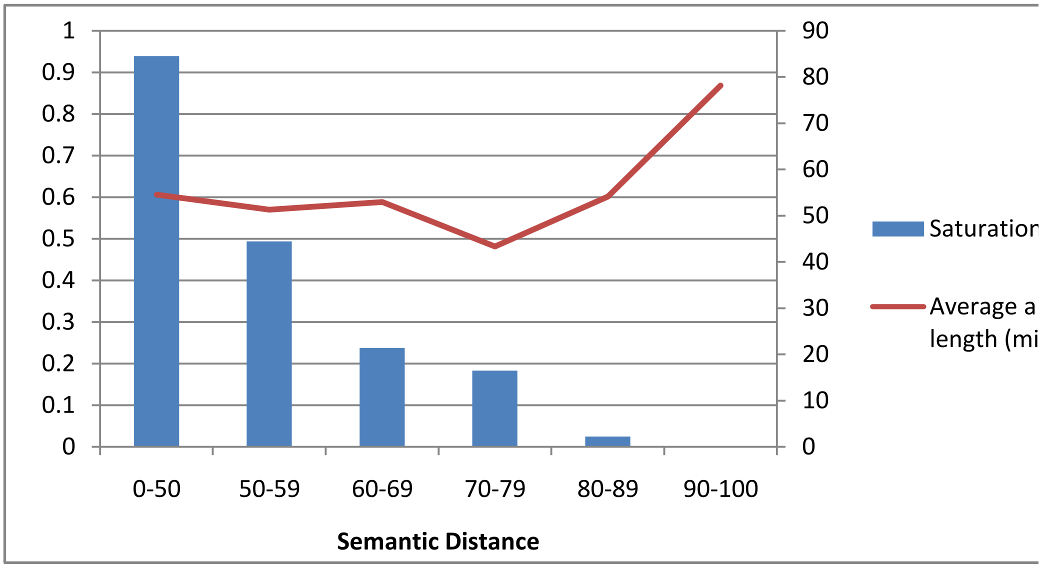

1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
90
80
70
60
50
40
30
20
10
0
0-50
50-59
60-69
70-79
80-89
90-100
Semantic Distance
Saturatior
Average a
length (mi

Figure 4

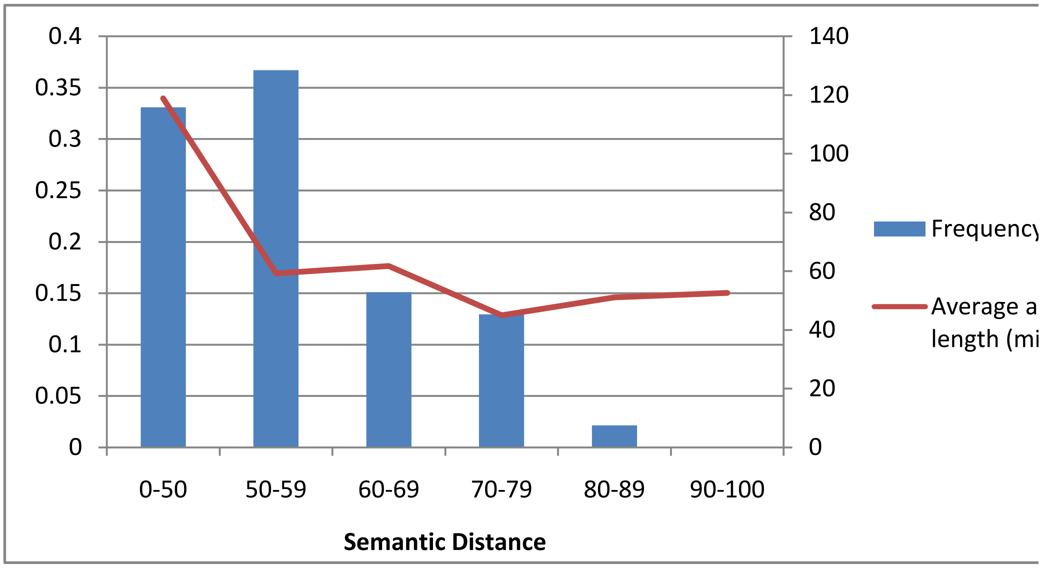

0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
140
120
100
80
60
40
20
0
0-50
50-59
60-69
70-79
80-89
90-100
Semantic Distance
Frequency
Average a
length (mi

Figure 5

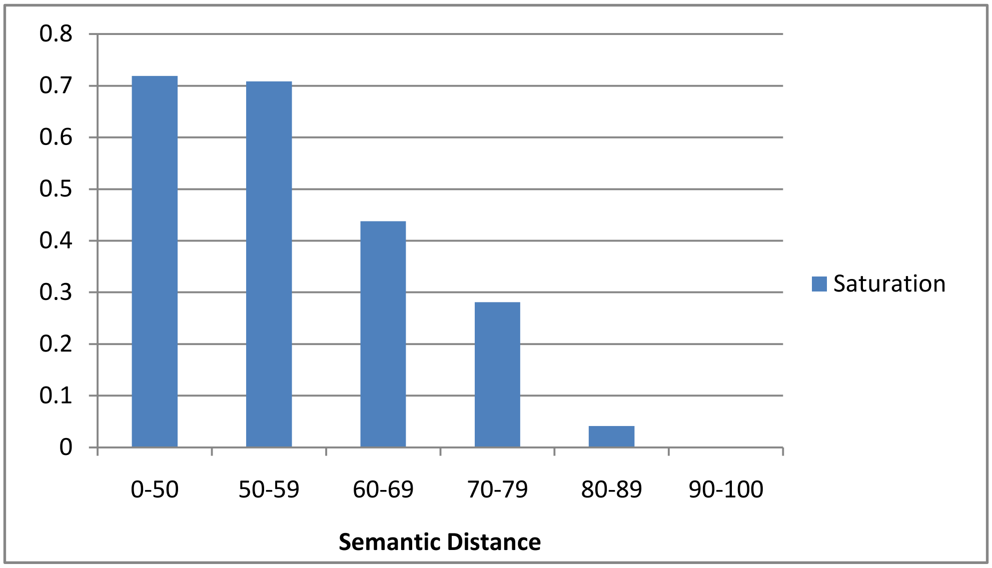

0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0-50
50-59
60-69
70-79
80-89
90-100
Semantic Distance
Saturation

Figure 6

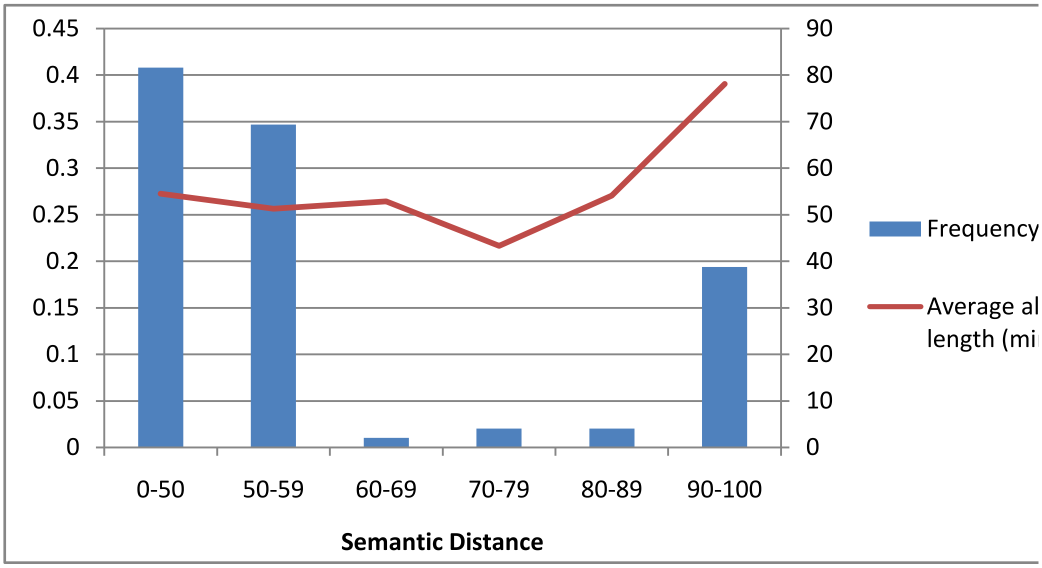

0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
90
80
70
60
50
40
30
20
10
0
0-50
50-59
60-69
70-79
80-89
90-100
Semantic Distance
Frequency
Average al
length (mi

Figure 7

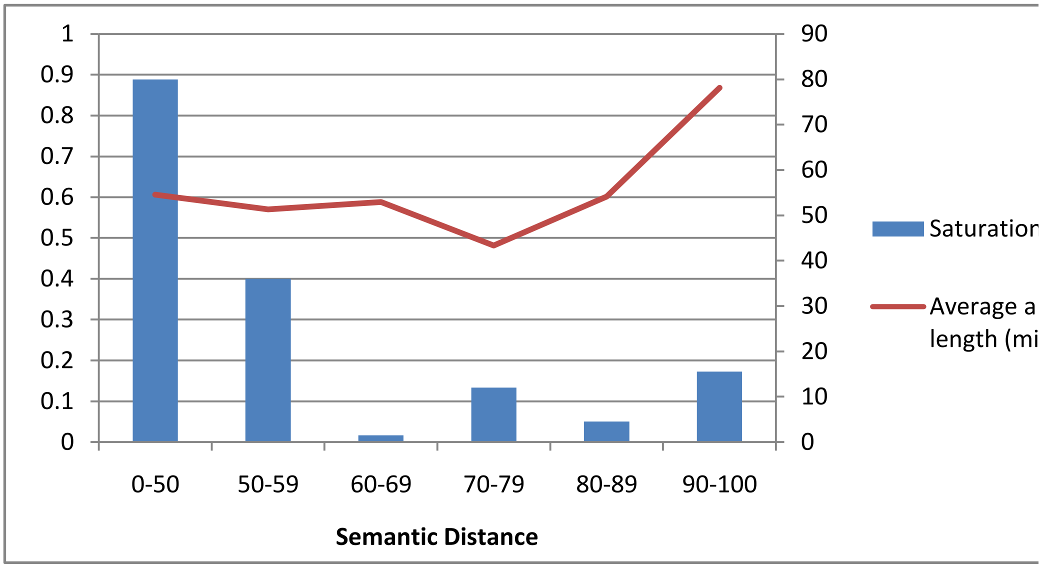
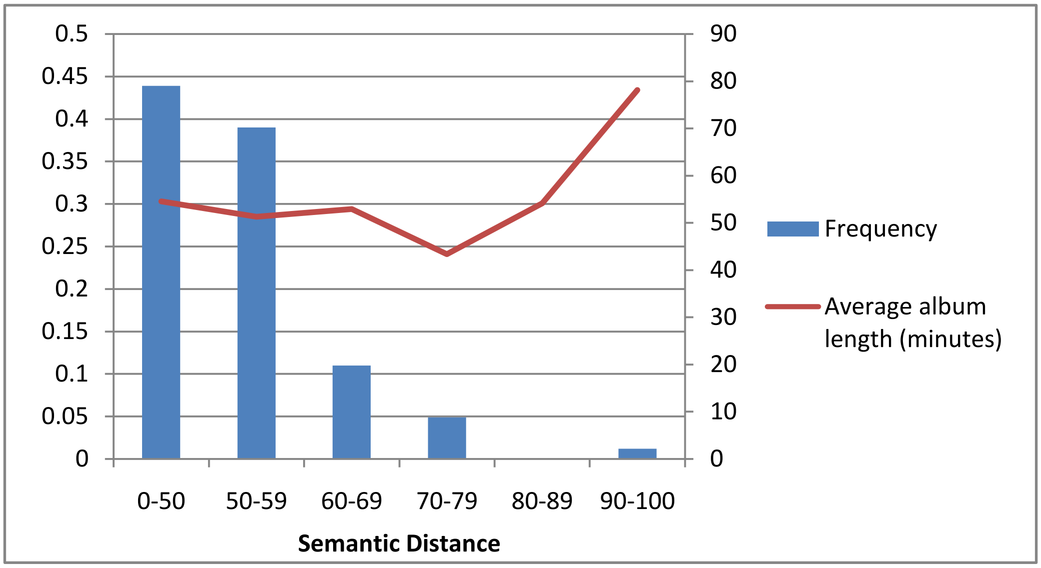

1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
90
80
70
60
50
40
30
20
10
0
0-50
50-59
60-69
70-79
80-89
90-100
Semantic Distance
Saturatior
Average a
length (mi

Figure 8



0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
90
80
70
60
50
40
30
20
10
0
0-50
50-59
60-69
70-79
80-89
90-100
Semantic Distance
Frequency
Average album length (minutes)

Figure 9

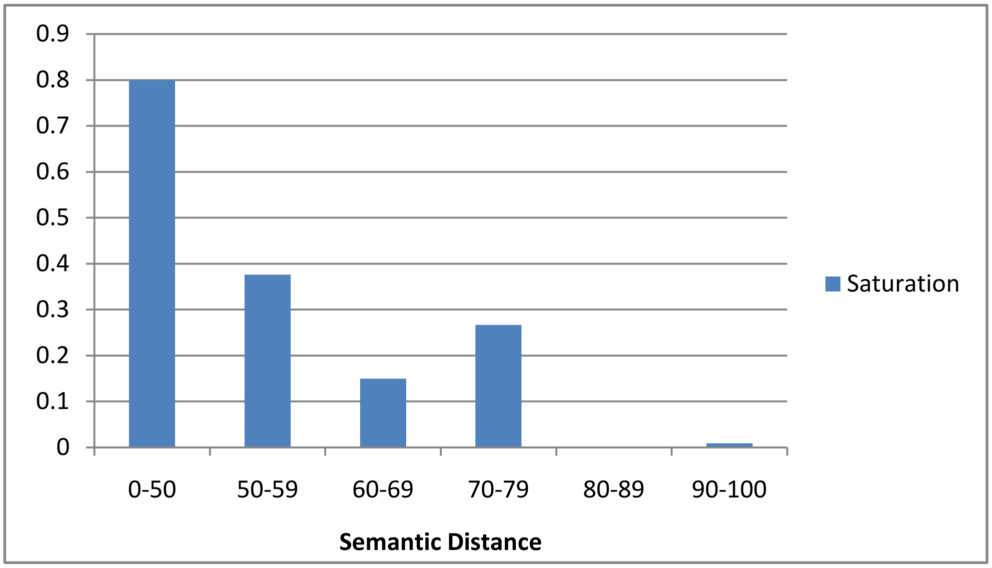

0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0-50
50-59
60-69
70-79
80-89
90-100
Semantic Distance
Saturation

Figure 10

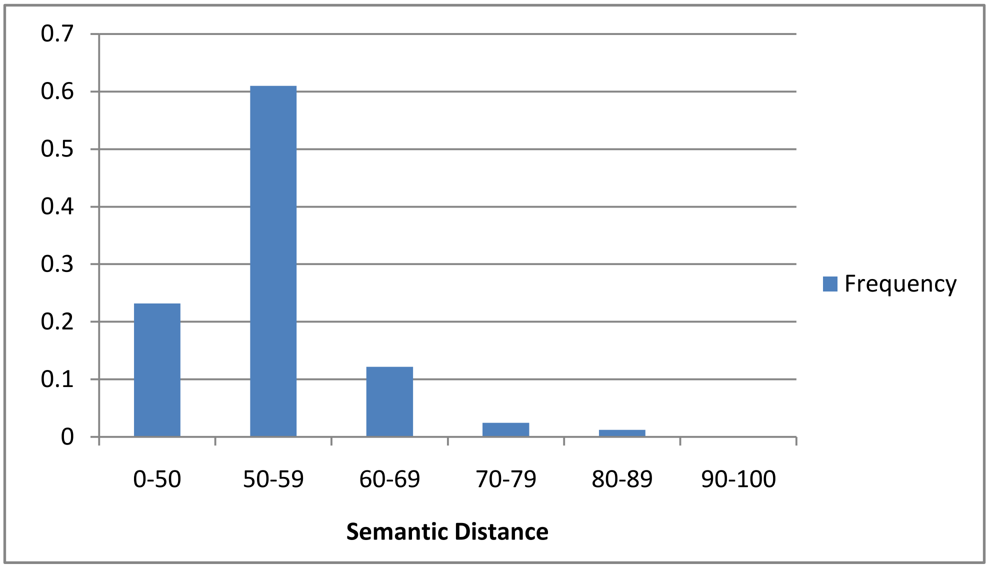
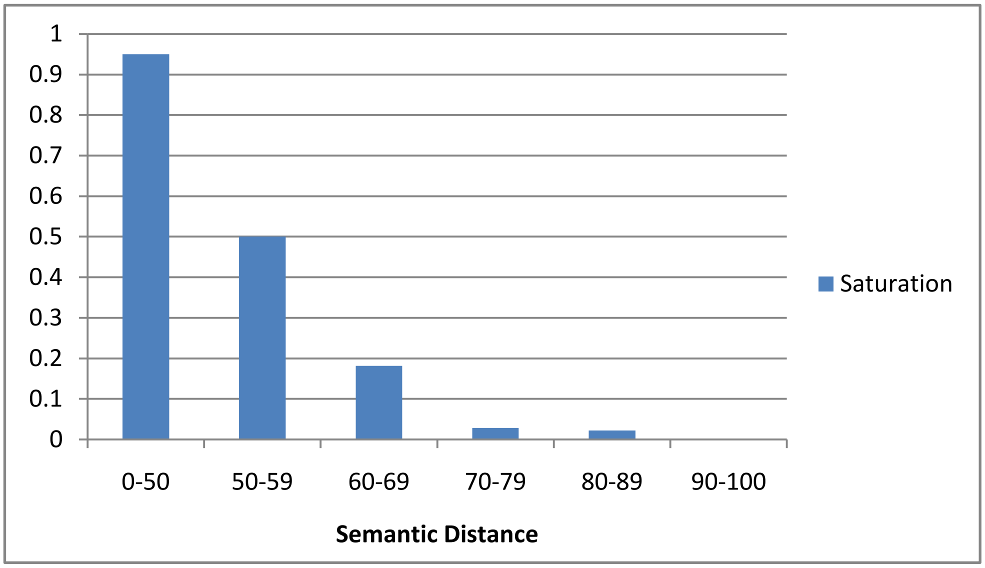

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0-50
50-59
60-69
70-79
80-89
90-100
Semantic Distance
Frequency

Figure 11


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0-50
50-59
60-69
70-79
80-89
90-100
Semantic Distance
Saturation