허밍 질의 처리 시스템 상에서 빈번 멜로디를 사용한 효과적인 내용 기반 인덱싱 방법

An Efficient Content-Based Indexing Method Using Frequent Melody in Query-By-Humming System

유진희(Jin-Hee You)*, 박상현(Sang-Hyun Park)**, 하유민(You-Min Ha)**

요 약

내용 기반 인덱싱 방법 (Content Based Indexing Method)은 방대한 양의 음악 데이터를 효율적으로 저장하고 검색하는 방법으로써 널리 사용되고 있다. 본 논문에서는 효과적인 내용 기반 인덱싱 방법을 사용하여 사용자가 부정확한 멜로디를 질의하였을 경우 원하는 음악을찾아주는 허밍 질의 처리 시스템 (Query-By-Humming System)을 설계한다. 이를 위해 본 논문은 세 가지의 방법을 제안한다. 첫 번째는 검색 속도를 높이기 위해 인덱스에 저장할 멜로디를 문자 형태로 변환한다. 이때 사용되는 문자 변환 방법은 사용자의 허밍에 포함된 에러를 허용하는 방법으로써 검색 결과의 정확도를 높일 수 있다. 두 번째는 사용자가 자주 질의할 가능성이 높은 빈번 멜로디와 긴 쉼표 후에 시작되는 멜로디를 인덱싱 하여 검색 속도를 높이고 신뢰도가 높은 빈번 멜로디를 생성하는 빈번 멜로디 추출 알고리즘을 제안한다. 세 번째로는 정확도를 향상시키는 3단계 검색 방법을 제안한다. 이는 데이터베이스 접근을 최소화 하고 인덱스만을 통해 정확한 검색 결과를 얻기 위하여 제안되었다. 본 논문은 실제 음악데이터를 대상으로 다양한 실험을 수행함으로써 제안된 방법의 성능을 검증하였다.

주제어 : 멀티미디어 데이터베이스, 허밍 질의 처리 시스템, 음악 정보 검색, 내용 기반 검색, 인덱싱

ABSTRACT

Content based indexing method becomes known to broadly as the most efficient way of storing and retrieving vast music data. In this Paper propose Efficient Query-By-Humming System (EQBHS) that efficiently retrieve and store musics using content-based indexing when a user inquires with his incorrect humming. For this purpose, we propose three different methods. (1) To accelerate the query processing in the EQBHS, we convert the numeric-typed melody data stored in index to the character-typed one. For achieving more accurate result, we can use an error-allowed mapping method from a note to a character. (2) the frequent melody which occurs many times in the music and the melody partitioned by a rest on the assumption that users are to hum what they can easily remember in the music. For this purpose, we also propose the frequent melody extraction algorithm which shows a reliable performance. (3) Lastly, we propose the 3-stage searching method. The goal of this method is to minimize the database access and to search more accurately and faster by using indices. We verified the proposed methods by executing various experiments with a number of music data.

Key words : Multimedia Database, Query By Humming System, Music Information Retrieval, Content-Based Searching Method, Indexing

---

본 연구는 문화관광부 및 한국문화콘텐츠진흥원의 문화콘텐츠기술연구소(CT)육성사업의 연구결과로 수행되었음

* 연세대학교 컴퓨터과학과, 교신저자
** 연세대학교 컴퓨터과학과

+ 논문접수 : 2006년 4월 26일, 심사완료 : 2006년 8월 10일

1. 서 론

최근 멀티미디어 정보의 활용이 일반화되면서 방대한 양의 멀티미디어 정보를 효율적으로 저장하고 검색하는 방법의 필요성이 증대되고 있다. 그러나 현재 멀티미디어 저장과 검색 방법은 대용량의 데이터를 관리함으로써 저장 공간의 부족함이 발생하고 오랜 검색 시간이 필요하다는 문제점을 가지고 있다.

현재 일반적인 멀티미디어 검색 방법으로 텍스트 기반 검색 방법이 있다. 이 방법은 멀티미디어 데이터를 대표하는 키워드를 멀티미디어 자료에 미리 선정해 두고 사용자는 이러한 키워드를 사용하여 원하는 자료를 검색하는 방식이다. 그러나 사용자가 키워드를 기억하지 못할 경우 검색이 어렵다는 것과 키워드와 정확하게 일치하는 정보만 검색해 주기 때문에 유사한 내용을 가진 정보를 검색하기에는 부적절하다는 단점을 가지고 있다. 특히 음악 데이터의 검색 방법 중 가장 일반적으로 쓰이는 방법도 텍스트 기반의 검색 방법이다. 이러한 검색 방법에서는 음악을 대표하는 제목, 가수 명, 앨범 명 등의 메타정보를 사용하여 원하는 음악을 검색한다. 그러나 사용자가 위와 같은 정보를 기억하지 못할 경우 원하는 음악을 검색하기 어려울 뿐만 아니라 위와 같은 키워드를 알고 있더라도 유사한 멜로디를 가진 음악을 검색하기가 어렵다. 이러한 문제점을 해결하기 위해 본 논문은 음악의 멜로디를 사용하여 원하는 음악을 검색할 수 있는 내용 기반 검색 방법(Content Based Searching Method)을 사용한다[1].

내용 기반 검색이란 주어진 멀티미디어 데이터의 내용에 해당하는 속성 데이터를 추출한 후 이 속성 데이터를 기반으로 검색을 수행하는 방법이다. 이 방법은 유사성 검색을 토대로 하는 방법으로 유사한 내용을 가진 결과 값을 원하는 질의에 적합하다. 이를 기반으로 본 논문은 사용자가 부정확한 멜로디(Humming)로 질의하였을 경우 원하는 음악을 효율적으로 찾아주는 허밍 질의 처리 시스템(Query By Humming System)을 제안한다[2, 3, 4, 5, 6].

현재 허밍 질의 처리 시스템의 관련된 연구 활동 중 대표적인 것으로 N-gram[7, 8, 9]을 사용하는 방법이 있다. 이는 방대한 음악 데이터의 처리량을 줄이기 위하여 제안되었다. 이 방법은 N개의 음표들을 하나의 단위로 취급하여 한 음표씩 오른쪽으로 이동시키며 인덱싱 하는 방법으로 이는 크기가 N인 슬라이딩 윈도우(Sliding Window)와 같은 개념이다. 이 방법은 인덱싱이 쉽다는 장점을 가지고 있지만 정확도 측면에서 좋은 검색 결과를 보장하지 못한다는 단점도 가지고 있다. 그 이유는 N-gram에서 사용한 특징 데이터를 문자 형태로 변환할 때 부정확한 멜로디라는 허밍의 특징을 고려하지 않고 에러 허용 없이 변환하므로 결국 사용자가 부른 허밍의 부정확한 정도가 클수록 검색 결과의 정확도가 급속도로 떨어지기 때문이다. 이와 같은 문제점을 보완하고 빠른 검색 속도를 보장하기 위해 본 논문은 3가지 방법을 제안한다. 우선 첫 번째 방법은 기존의 N-gram과는 달리 검색 결과의 정확도를 보장해주기 위하여 사용자의 허밍에 포함된 에러를 허용해 주는 효과적인 문자 변환 방법을 제안하였다. 두 번째 방법은

검색 시간을 줄이기 위해 의미 있는 멜로디를 추출하여 인덱싱 하는 효과적인 내용 기반 인덱싱 방법 (Efficient Content Based Indexing Method)을 제안한다. 이를 위해 사용자가 자주 질의할 가능성이 높은 멜로디를 의미 있는 멜로디로 선정하고 이를 효과적으로 추출하기 위한 빈번 멜로디 추출 알고리즘을 제안하였다. 세 번째로는 3단계 검색 방법을 통해 데이터베이스의 접근을 최소화 하여 검색 속도를 향상시킨다.

본 논문에서 제안하는 시스템은 크게 2가지의 단계로 나뉘어 진다. 첫 번째 단계는 데이터베이스를 구축하는 단계이고 두 번째 단계는 데이터베이스를 검색하는 단계이다. 첫 번째 단계인 데이터베이스 구축은 미디(MIDI)형식의 음악을 입력받아 특징 데이터를 추출한 후 1개의 데이터베이스와 3개의 인덱스를 생성한다. 다음으로 두 번째 단계인 검색 단계는 우선 사용자가 부른 허밍을 데이터베이스와 인덱스에 저장된 데이터의 형태와 같도록 변형하는 질의 전 처리 과정이 필요하다. 이 과정이 끝나면 검색결과를 얻을 때까지 인덱스 검색을 먼저 수행하고 검색 결과를 얻지 못할 시 데이터베이스를 검색하는 순서로 진행된다. 검색 결과는 질의 멜로디와 유사한 멜로디를 갖는 모든 음악을 사용자에게 전달해 준다.

본 논문이 제안하는 허밍 질의 처리 시스템은 노래실력이 없는 사람들이나 언어 장애를 가진 사람들도 쉽게 음악을 검색할 수 있을 뿐만 아니라 모바일 장비를 통해 질의를 받을 경우 시간과 장소에 크게 구애 받지 않는다는 장점이 있다.

본 논문은 다음과 같이 구성된다. 2장에서는 관련연구를 통하여 기존의 특징 데이터의 종류와 인덱스 기법에 대해 기술한다. 3장에서는 허밍 질의 처리 시스템의 구축을 단계별로 설명한다. 4장에서는 데이터베이스를 검색하는 과정을 보이고 5장에서는 성능 평과 결과를 보여준다. 마지막으로 6장에서는 본 논문을 요약하고 결론을 내린다.

2. 관련연구

현재 내용기반 음악 정보 검색 시스템을 위해 질의 특성에 따라 효과적인 특징 데이터를 찾는 연구가 활발히 진행되고 있다. 이는 정확한 음정과 박자를 가진 질의를 처리하기 위한 것과 허밍과 같은 부정확한 질의를 처리하기 위한 것으로 나뉘어 진다[10]. 부정확한 질의를 처리하기 위한 대표적인 시스템으로 본 논문과 같은 허밍 질의 시스템이 있다. 이 시스템을 위해 많이 사용되는 특징 데이터로는 UDR 스트링이[11] 있는데 이는 인접한 두 음표의 높이를 사용하여 뒤의 음표가 앞의 음표보다 음정이 높으면 ‘U(Up)’, 낮으면 ‘D(Down)’, 같으면 ‘R(Repeat)’로 표현한다. 그러나 이와 같은 데이터는 단지 음 높이의 곡선을 이용할 뿐 음의 길이는 무시한 정보이므로 정확한 음악 검색 방법에는 적합하지 않은 특징 데이터이다. 또 다른 방법으로는 음악의 템포(Tempo)를 사용하여 검색하는 시스템이 있다[12]. 그러나 템포 데이터 자체가 많은 특징 정보를 가지고 있지 않기 때문에 부정확한 허밍을 처리할 때 정확성이 낮

은 검색 결과를 가져온다. 이와 같은 단점을 보완하기 위해 허밍 질의에 적합하고 음악의 특징 정보를 많이 포함하는 효과적인 특징 데이터의 필요성이 대두된다. 이를 위해 기존의 여러 특징 데이터들을 비교해 본 결과, 인접한 두 음표의 높이 변화량과 길이 변화율을 사용한 경우 가장 정확한 결과를 얻을 수 있었다[10]. 본 논문에서는 허밍 질의 처리 시스템에 적합한 특징으로 두 음표의 음정 변화량(Pitch Interval)과 박자 변화율(Duration Ratio)을 사용한다.

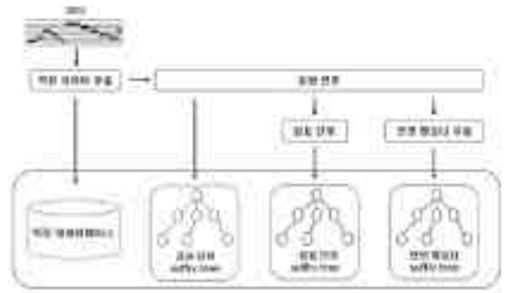
내용기반 음악 정보 검색에 관련된 또 다른 연구 활동으로는 검색 시간을 줄이기 위하여 데이터베이스를 검색하는 것보다 중요도가 높은 멜로디를 인덱싱 하여 검색하는 방법이 있다[13]. 대표적으로 검색 속도를 증가시키기 위해 한 음악의 시작 멜로디를 사용자가 자주 질의할 것이라는 점을 고려하여 이 멜로디를 인덱싱 하는 방법이 있다[14]. 이 방법은 사용자가 시작 멜로디를 질의하지 않는다면 인덱스에서 검색이 이루어지지 못하고 데이터베이스를 검색해야 하는 문제점이 있다. 또 다른 방법으로 N-gram 방법을 사용하여 반복된 멜로디의 패턴을 추출하는 연구가 있다[15]. 그러나 이 방법은 마디와 같은 객관적인 단위를 무시한 채 반복된 멜로디를 추출하였기 때문에 생성된 빈번 멜로디의 경우의 수가 많아져 인덱스 크기가 커진다는 단점을 가지고 있다[16]. 이러한 점을 보완하기 위해서 본 논문은 일정한 박자를 가지고 있는 마디를 사용하여 마디 단위로 패턴을 분석한 후 한 음악을 대표하는 적은 수의 신뢰도 높은 빈번 멜로디를 추출하는 방법을 제안한다. 이는 기존의 [16]방법과 비교하여 인덱스의 크기가 줄어드는 장점이 있다.

3. 허밍 질의 처리 시스템 구축

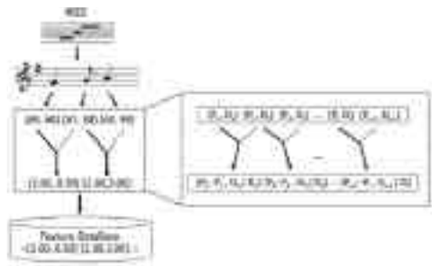
이 장에서는 허밍 질의 처리 시스템을 위해 그림 1과 같이 데이터베이스를 구축하는 과정을 설명한다. 3.1절에서는 특징 데이터 추출 과정을 설명하고 3.2절에는 특징 데이터를 문자 값으로 변환하는 방법을 설명한다. 3.3절에서는 빈번 멜로디를 추출하는 과정을 설명하고 3.4절에서는 인덱스를 생성하는 과정을 설명한다.

3.1 특징 데이터 추출과 저장

현재 미디 형태의 음악 데이터를 분석하여 음표의 음정과 박자를 추출해 주는 시스템이 많이 개발되어 있다. 본 논문은 기존 시스템을 이용하여 미디 형태의 음악 데이터로부터 음정과 박자를 추출한다. 음정은 미디 음악에서 옥타브(Octave)라는 형태로 그 높이 값이 이미 정의되어 있다. 옥타브의 범위는 0부터 127까지이고 표기는 음계와 숫자로 표현한다. 예를 들어 옥타브가‘ F5'인 음표의 높이 값은 ‘ 65 ’이고 박자는 미디 음악에서 4분 음표, 즉, 한 박자의 길이(Time base)를 기준으로 계산된다. 그러나 음악마다 한 박자의 길이가 다를 수 있기 때문에 모든 음악의 박자가 동일한 기준을 갖도록 변환해 주기 위해서 기준이 될 임의의 값을 정하여 모든 음표의 박자를 변경해 준다. 모든 음악의 음정과 박자를 추



<그림 1> 데이타베이스 설계 구성도



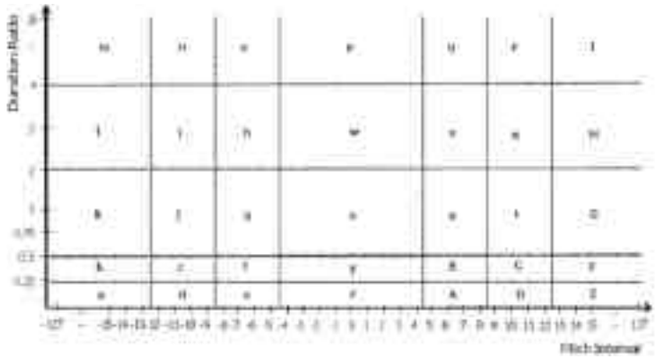
<그림 2> 데이타베이스 설계 구성도

출한 후 <그림 2>와 같이 2차원 벡터 형태로 특징 데이터를 계산한다.

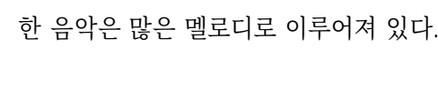
사용자가 입력한 허밍의 특징은 정확한 음정과 박자가 아닌 부정확한 멜로디이다. 이는 원 음악과 다르게 전체 음정을 높게 부르거나 낮게 부를 수 있다. 예를 들어 음악을 여자가 부른 경우와 남자가 부른 경우, 두 음악의 전체 높이는 다를 수 있다. 또는 원 음악보다 빠르게 부르거나 느리게 부를 수도 있다. 그러므로 위의 경우들을 고려하여 사용자가 부른 허밍의 음정 변화량과 박자 변화율을 계산한다. 우선 음정의 변화량이란 인접한 두 음표 사이에서 뒤의 음표의 음정과 앞의 음표의 음정의 차를 의미한다. 즉, 음정이 증가했는지 감소했는지 또는 동일한지를 계산하는 것이다. 박자의 변화율은 인접한 두 음표 사이에서 뒤 음표의 길이를 앞 음표의 길이로 나누어 증가율을 계산된다. 여기서 증가율을 사용하는 이유는 전체적인 빠르기의 비율을 비교하기 위함이다. 특징 데이터를 모두 추출한 후 <그림 2>와 같이 2차원 벡터 형태의 특징 데이터를 데이터베이스에 저장한다.

3.2 문자 변환

2차원 벡터 형태의 특징 데이터가 저장된 데이터베이스를 검색할 경우 모든 음악을 차례로 비교 하여 검색 결과를 얻어낸다. 이때 데이터베이스에 저장된 2차원 벡터 형태는 숫자 값으로 되어있고 검색을 위한 질의와의 비교는 유클리디안 거리(Euclidean Distance) 계산 방법을 사용한다. 그러나 이 방법은 검색 속도가 느리기 때문에 이를 보완하기 위해 인덱스를 구성한다. 인덱스에 저장할 데이터 형태는 검색 속도를 증가시키기 위해 숫자 값이 아닌 문자 형태로 지정하고 문자열 매칭(String Matching) 알고리즘을 사용하여 검색



<그림 3> 문자 변환 기준의 예



<그림 4> 문자 변환 결과

한다.

2차원 벡터 형태의 숫자 값을 문자 값으로 변환하기 위해 <그림 3>과 같은 기준을 정한다. 그림과 같이 35개의 구간으로 나눈 후, 각 구간에 하나의 문자를 할당한다. 따라서 비슷한 값을 가진 2차원 벡터 값은 같은 문자로 치환되게 된다. 이는 허밍이 부정확하다는 특징을 반영하기 위해 각 2차원 형태의 데이터는 오차를 허용하는 문자로 변환되는 것이다. 예를 들어 <그림 4>의 데이터를 문자 변환 기준에 따라 특징 데이터를 문자로 변환한다면 <x, w, x, w, y, u>의 결과 값을 얻을 수 있다.

3.3 빈번 멜로디 추출

한 음악은 많은 멜로디로 이루어져 있다. 멜로디란 음악적인 표현과 인간의 감정을 가장 잘 나타내는 요소로서 갖가지 음높이와 길이를 가진 음을 결합한 것을 말한다. 이를 바탕으로 본 논문에서는 멜로디를 의미가 있는 임의의 길이의 음표와 쉼표의 집단으로 정의하였고 이를 2가지 형태로 나누었다.

첫 번째 형태의 의미를 갖는 멜로디는 한 음악에서 여러 번 반복되어 나타나는 임의의 길이를 가진 멜로디이다. 사용자가 음악을 들었을 때 쉽게 기억할 수 있는 멜로디는 한 음악에서 여러 번 반복되어 나타나는 멜로디일 가능성이 크다. 즉, 사용자가 음악 검색을 위해 쉽게 기억할 수 있는 멜로디를 흥얼거릴 가능성이 높다는 점을 고려하여 본 논문에서는 빈번 발생 멜로디를 사용한 인덱스를 생성한다.

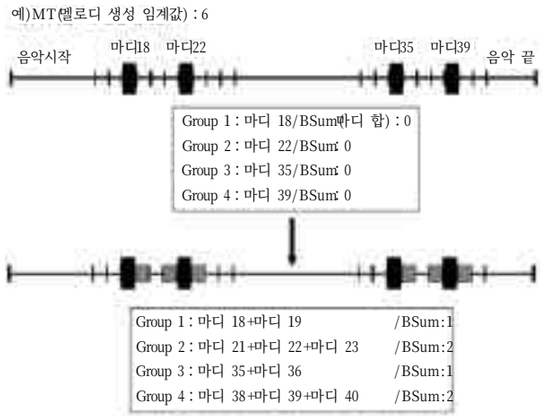
두 번째 형태의 의미를 갖는 멜로디는 일정 길이 이상의 쉼표 사이에 있는 멜로디이다. 일정 길이 이상의 긴 쉼표는 멜로디들을 구분짓는 큰 의미를 가진 경계로 볼 수 있다. 이는 음악 데이터들을 분석해 본 결과 멜로디의 흐름이 긴 쉼표를 통해 나뉘어 진다는 것을 알 수 있었다. 대부분의 음악에서 쉼표는 후렴 부분 전, 2절이 시작되기 전 등에 많이 나타난다. 즉, 음악의 후렴 부분과 처음 부분을 사용자가 기억하기 쉬울 것이라는 가정 하에 쉼표의 의미를 도입하였다. 이절에서는 첫 번째 의미를 갖는 빈번 멜로디를 추출하는 과정을 설명한다.

<표 1> 한 음악에서 빈번멜로디 추출 알고리즘

Input:	MID (문자열 형태의 모든 음악), BarID (마디 위치 정보), N (빈번 개수 임계값), MT (멜로디 생성 임계값)
Output:	문자열 형태의 빈번 멜로디 그룹
1;	BarID를 사용하여 마디 단위로 모든 MID의 문자열을 나눈다.
2:	각 마디 단위로 문자열을 비교하여 마디가 나타난 빈번 개수 K를 계산한다.
3:	K로 정렬하여 정렬 순서 S를 저장한다. (같은 빈번 횟수를 가진 마디는 S와 같다)
4:	S가 1인 마디의 발생 횟수 K만큼 그룹을 생성한다.
5:	S가 1인 마디들의 BarID가 i라면 이를 B_i로 선정하고 각 그룹의 대표 마디라 부른다.
6:	각 그룹의 B_i에서부터 빈번 멜로디 생성이 시작한다.
7-1:	각 그룹마다 B_i를 기준으로 i-1마디의 K_{i-1}값이 N이상일 때 그 그룹에 포함시킨다.
7-2:	각 그룹마다 B_i를 기준으로 i+1마디의 K_{i+1}값이 N이상일 때 그 그룹에 포함시킨다.
7-3	마디 합(BSum)을 계산하기 위해 B_i의 S와 그룹에 포함시킨 마디들의 S와의 차를 각각 구하여 기존의 BSum과 합친다.
8:	마디 합이 MT이하일 때까지 i를 증가시켜가며 7번 과정을 반복한다.
9:	모든 그룹이 생성된 후 그룹안의 마디가 중복되거나 이웃하면 그룹을 합친다.
10:	이웃하지 않은 그룹이 모두 생성되면 각 그룹을 하나의 빈번 멜로디로 정하고 문자열 형태로 저장한다.

<표 1>은 빈번 멜로디 추출 알고리즘을 설명한 것이다.이 알고리즘은 마디 단위로 문자열을 비교해 나가면서 마디의 패턴을 분석하여 빈번 마디를 추출한 후 가까이 위치해 있는 빈번 마디를 그룹핑(Grouping)함으로써 빈번 멜로디를 생성해 나가는 방식이다. 여기서 마디를 사용하는 이유는 마디란 동일한 박자를 가지고 있는 객관적인 단위이기 때문이다.

예를 들어 ‘ 그 후로 오랫동안 ’이란 제목을 가진 음악을 빈번 멜로디 추출 알고리즘에 적용할 경우, 알고리즘의 초기단계인 단계 1과 2 그리고 3을 수행한 결과는 <표 2>와 같다. 문자열 <j, x, k, j, i, x>는 18번째, 22번째, 35번째, 39번째 마디에서 나타나 빈번 개수인 K값은 ‘ 4 ’가 된다. 이들 마디는 위의 음악에서 가장 빈번히 발생한 것으로 빈번 마디라 부르고 이 마디들이 모여 빈번 멜로디를 구성한다. 여기서 정렬 순서를 사용하여 빈번 멜로디를 생성하는 이유는 각 음악마다 빈번 개수가 다르기 때문에 이러한 빈번 개수를 사용하여 빈



<그림 5> 음악에서 빈번 멜로디가 생성 된다.

<표 2> 음악 ‘ 그 후로 오랫동안 의 마디 단위 패턴 분석의 결과

마디 안의 문자열	마디 ID (BarID)	빈번 개수(K)	정렬 순서(S)
<j,x,k,j,i,x>	18, 22, 35, 39	4	1
<k,k,j,x,g>	19, 36	2	2
<j,x,j,k,k,k>	20, 37	2	2
<k,j,j,w,g>	21, 38	2	2
…	…	…	…
<j,x,g,k,E,x,i,x>	49	1	3

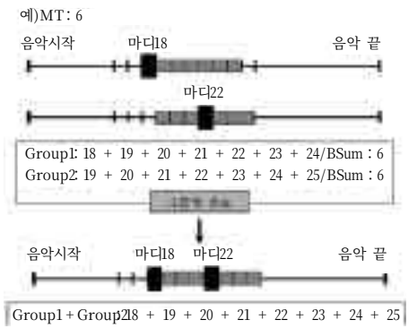
번 멜로디를 생성하는 것이 정렬순서를 기준으로 빈번 멜로디를 생성해 가는 것보다 복잡하기 때문이다.

<그림 5>는 단계 4와 5 그리고 6을 마치고 단계 7을 수행한 것이다. 그림 5의 수직선은 하나의 음악을 나타내고 눈금은 마디의 경계를 의미한다. 이 단계에서 마디 합인 BSum을 계산하기위해 정렬 순서 S를 사용한다. 그룹의 대표 마디의 위치 즉, BarID가 i라면 이 마디의 정렬 순서는 Si가 되고 그 값은‘ 1 이다. 이 대표마디를 기준으로 좌우의 마디의 K가 각각 N이상이면 좌우 마디를 합친다. 다음으로 합친 마디의 정렬 순서를 S_{i+1} 과 S_{i-1}이라 하고 마디 합은 <그림 6>과 같이 계산된다.

빈번 멜로디를 생성할 때 두 가지의 매개변수가 필요하다. 첫 번째 매개변수는 빈번 개수 임계값(N)으로서 이를 사용하는 이유는 빈번 개수가 N미만인 마디는 빈번 멜로디에 포함되지 않도록 하기 위해서이다. 두 번째 매개변수는 멜로디 생성 임계값(MT)으로서 빈번 멜로디 생성을 중단하기 위해 필요한 것으로써 BSum 값이 MT를 초과하면 빈번 멜로디 생성을 중단한다. <그림 7>과 같이 마디 합이 MT를 초과하여 단계 7과 8을 마치고 단계 9를 수행한다. 이때 이웃하는 그룹에 속하는 마디가 중복되거나 이웃한다면 하나의 그


```
int i=0;
while(BSum ≤ MT){
    if(K_{i+1} ≥ N)
        BSum+=| S_i - S_{i+1} | ;
    if(K_{i-1} ≥ N)
        BSum+=| S_i - S_{i-1} | ;
    i++;
}
```

<그림 6> BSum(마디 합)계산 알고리즘

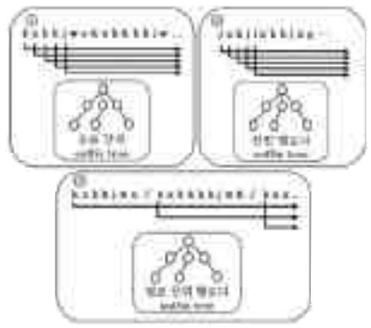


<그림 7> 중복된 그룹 처리

룹으로 합친다. 이는 두 개의 그룹에 속하는 빈번 멜로디가 하나의 빈번 멜로디로 바뀐다는 의미이다. 이 단계를 수행하는 이유는 중복된 빈번 멜로디를 따로 취급하여 저장하는 것보다 하나의 빈번 멜로디로 합쳐 저장하는 것이 저장 공간을 효율적으로 사용하는 것이기 때문이다. 빈번 멜로디 알고리즘이 모두 수행되면 각 음악마다 문자열 형태의 빈번 멜로디가 생성 된다.

3.4 인덱스

본 논문은 <그림 8>과 같이 3개의 인덱스를



<그림 8> 3개의 인덱스 설계

생성 한다.첫 번째 인덱스에는 모든 음악을 문자 형태로 변환한 후 음표단위로 서픽스 트리(Suffix Tree)에 저장하고 두 번째 인덱스인 빈번 멜로디 인덱스는 3.3절에서 추출한 빈번 멜로디를 음표단위 즉, 문자단위로 서픽스 트리에 저장한다. 세 번째 인덱스인 쉼표단위 인덱스는 일정한 길이 이상의 쉼표를 경계로 멜로디를 나눈 후 멜로디 단위로 서픽스 트리에 저장한다. <그림 8>의 세 번째 인덱스를 보면‘ / ’를 경계로 문자열, 즉, 멜로디들이 나뉘어져 있다.‘ / ’는 일정한 길이 이상의 쉼표를 의미하며 쉼표와 쉼표사이를 하나의 의미 있는 멜로디로 정하여 서픽스 트리에 삽입한다.

여기서 3개의 인덱스를 사용하는 이유는 우선 사용자가 쉽게 기억할 수 있는 빈번 멜로디를 질의할 가능성이 높다는 점을 고려하여 빈번 멜로디 인덱스를 생성하였고 다음으로 사용자가 쉼표 다음에 나타나는 멜로디, 즉, 음악의 시작부분 또는 후렴 부분, 간주가 끝난 후 시작되는 부분에서 질의할 가능성이 높다는 점을 고려하여 쉼표단위 인덱스를 생성하였다. 그러나 사용자가 빈번 멜로디와 쉼

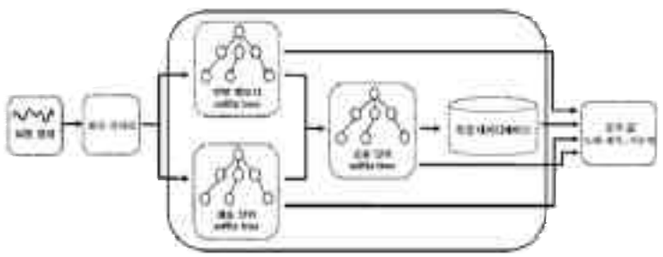
표단위 멜로디가 아닌 다른 부분의 멜로디를 질의하였을 경우 위와 같은 2개의 인덱스로는 검색이 수행되지 못하고 데이터베이스를 검색해야 하는 문제점이 있으므로 이를 보완하기 위해 모든 음악을 음표단위로 인덱스 하는 음표단위 인덱스를 생성한 것이다.

4. 데이터베이스 검색

이 절에서는 사용자가 허밍 질의를 하였을 때 데이터베이스를 검색하는 단계를 설명한다. <그림 9>는 데이터베이스를 검색하는 전체 흐름을 나타낸 것이다.

데이터베이스 검색은 두 단계로 나누어진다. 첫 번째 단계는 질의 전처리 단계로 입력된 질의를 특징 데이터베이스와 인덱스에 저장된 데이터 형태와 같도록 변경해준다. 입력된 질의는 사용자가 마이크를 통해 흥얼거린 허밍이다. 이는 데이터베이스를 구축할 때 음악데이터의 입력 형태인 미디와 다르게 웨이브 (Wave)형태를 지니고 있다. 그러므로 미디 음악에서 추출한 데이터와 같이 웨이브 형태의 음악에서도 음표의 높이와 박자를 추출하고 이를 사용하여 특징 데이터를 계산한다. 또한 인덱스를 검색하기 위해 특징 데이터를 <그림 3>과 같은 문자 변환 기준을 사용하여 문자 형태로 변환한다. 이와 같은 질의 전처리 과정이 끝나면 <그림 9>와 같이 빈번 멜로디 인덱스와 쉼표 인덱스를 검색하는 두 번째 단계로 넘어간다. 이때 문자열을 비교하기 위해 정확 매칭(Exact Matching) 알고리즘을 수행한다. 이 단계에서는 빈번 멜로디 인덱스와 쉼표 단위 인덱스를 병렬로 검색하여 실시간으로 결과 값을 가져온다. 모든 서픽스 트리의 단말 노드에 저장된 음악 아이디를 통해 일치하는 음악의 제목과 가수 명을 사용자에서 전달한다. 이때 사용자가 원하는 음악을 찾게 되면 사용자 입장에서 강제로 검색을 중단할 수 있다.

사용자가 빈번 멜로디 또는 쉼표 다음에 시작하는 멜로디를 질의하지 않았다면 이 단계에서 결과 값을 얻지 못할 것이다. 이때 다음 단계로 모든 음악을 음표단위로 저장한 인덱스를 검색하기 시작한다. 이때도 마찬가지로 정확 매칭 알고리즘을 수행하여 음악의 제목과 가수 명을 얻어온다. 만약 이 단계에서도 검색이 완료되지 않았다면 특징 데이터베이



<그림 9> 데이터베이스 검색 흐름도

스를 검색해야 한다. 이때 특징 데이터베이스에는 2차원 벡터 형태는 숫자 값이 저장되어 있으므로 질의의 형태도 2차원 벡터 형태의 특징 데이터를 사용해야하고 비교는 유클리디안 거리(Euclidean Distance) 계산 방법을 사용한다. 그러나 실험을 통해 인덱스 검색 단계에서 사용자가 입력한 질의와 유사한 음악이 모두 찾아짐을 알 수 있다. 즉, 특징 데이터베이스를 접근하지 않고 인덱스만을 통해 검색 결과를 얻어오므로 검색 속도의 향상을 가져온다.

5. 성능 평가

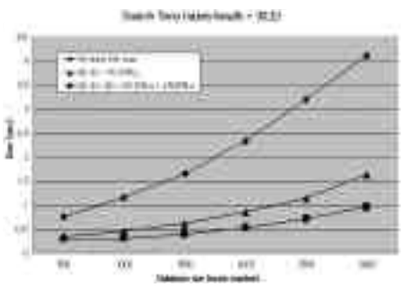
본 장에서는 제안하는 허밍 질의 처리 시스템의 성능 평가를 위한 실험 결과와 데이터 및 질의 생성 방법, 실험 결과의 비교 척도 등을 설명한다.

5.1. 음악 데이터와 질의 생성

실험에 사용할 데이터로는 데이터베이스의 설계를 위한 미디 형식의 음악 파일과 검색을 위한 허밍 형태의 질의가 필요하다. 우선 데이터베이스와 인덱스에 저장할 음악 데이터를 위해 3000곡을 수집하여 특징 데이터를 추출한 후 저장하였다. 그리고 질의는 허밍의 특징을 가진 부정확한 멜로디로써 네 가지 형태로 생성 하였다. 첫 번째 질의 형태는 실제의 정확한 멜로디 보다 음정이 전체적으로 높고 낮은 2개의 멜로디를 생성하였고 두 번째 질의 형태로 실제 정확한 멜로디 보다 전체적으로 음표의 박자가 빠르고, 느린 2개의 멜로디를 생성하였다. 또한 세 번째의 질의 형태로는 전체적으로 음정과 박자가 변형된 4개 멜로디를 생성하였다. 마지막으로 네 번째의 질의 형태는 멜로디 중간에 나타나는 음표들의 음정과 박자가 변형된 멜로디로 그 변형의 정도에 따라 12개의 멜로디를 생성하였다. 총 사용된 질의의 개수는 20개로 성능 평가를 위해 사용된다.

5.2. 데이터베이스의 크기에 따른 검색시간 비교

데이터베이스의 크기에 따라 인덱스를 사용했을 때와 사용하지 않았을 때 그리고 빈번 멜로디와 쉼표 단위 인덱스를 사용하지 않았을 때와 사용하였을 때의 검색 시간을 비교한다. <그림 10>은 데이터베이스에 저장된 음악의 곡수에 따라 각 시스템의 검색 시간을 실험한 결과이다. 곡수는 500곡부터 시작하고 500곡 단위로 3000곡 까지 증가시켜 가며 검색 시간을 측정하였다. 그리고 질의는 30초를 흥얼거렸을 때의 문자열 길이를 기준으로 하



<그림 10> 데이터베이스의 크기에 따른 검색시간 비교

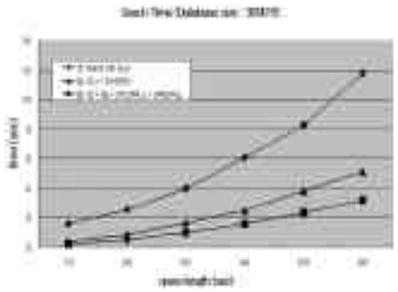
였다.

본 실험은 3가지 형태의 시스템을 비교한다. 첫 번째 형태는 인덱스를 사용하지 않고 오직 데이터베이스만 순차적으로 검색하는 Naive DB Scan 형태이고 두 번째 형태는 모든 음악의 멜로디를 인덱싱한 1차 인덱스를 사용하여 1차 인덱스를 먼저 검색하고 검색결과가 나오지 않았을 경우 특징 데이터베이스를 검색하는 형태이다. 마지막으로 세 번째 형태는 본 논문이 제안하는 방법으로 빈번 멜로디 인덱스인 2차 인덱스와 쉼표단위 인덱스인 3차 인덱스를 사용하여 이들을 먼저 검색한 후 검색결과를 얻지 못하면 1차 인덱스를 검색하는 순서의 형태이다.

<그림 10>과 같이 성능 결과는 본 논문이 제안하는 빈번 멜로디 인덱스와 쉼표단위 인덱스를 사용하여 검색을 수행하였을 때가 데이터베이스만을 순차적으로 검색 하였을 때와 1차 인덱스를 사용한 후 검색을 수행하였을 때 보다 검색 속도가 향상되었음을 보여준다. 이는 빈번 멜로디 인덱스와 쉼표단위 인덱스의 크기가 작기 때문에 검색 시간이 짧은 것과 이 검색 단계에서 검색이 수행되었다는 의미이다.

5.3. 질의 길이에 따른 검색시간 비교

<그림 11>은 질의의 길이에 변화에 따른 검색 시간을 비교한 결과이다. 사용자는 질의를 10초 동안 흥얼거렸을 때를 시작으로 10초 단위로 60초, 즉, 1분 동안 흥얼거렸을 때가지 증가시키며 검색 시간을 검색 시스템에 따라 비교한다.



<그림 11> 질의 길이에 다른 검색시간 비교

실험 결과, <그림 11>과 같이 데이터베이스 검색 시간의 증가량보다 인덱스를 사용하였을 때의 검색 시간 증가량이 적음을 알 수 있다. 그리고 본 논문이 제안하는 시스템이 질의 길이가 길어도 검색 시간이 짧다는 것을 실험을 통해 확인 하였다.

6. 결론 및 향후 연구

본 논문에서는 방대한 양의 음악 데이터베이스에서 사용자가 부정확한 음악을 입력하였을 때 데이터베이스에 저장된 모든 음악을 차례대로 비교하여 검색하는 비효율적인 방법을 보완하기 위해 효과적인 인덱스를 설계하였다. 이 인덱스에는 사용자가 한 음악에서 빈번 하게 나타난 멜로디를 쉽게 기억할 수 있고 자주 질의할 수 있다는 가정 하에 마디 단위 패턴분석을 통한 빈번 멜로디와 쉼표 단위 멜로디를 추출하여 저장하였다 이러한 인덱스를 사용하여 검색하였을 경우 기존의 데이터베이스에 저장된 방대한 음악데이터를 순차적으로 비교할 필요 없이 인덱스만 검색하여 결과 값을 빠르게 얻을 수 있다. 또한 인덱스의 검색 속도를 향상시키기 위해 숫자 형태의 멜로디를 문자열로 변환하여 저장하였다. 또한 검색 결과의 정확도를 보장하기위해 사용자의 허밍에 포함된 에러를 허용하는 문자 변환 방법을 제안하였다. 마지막으로 3단계 검색 결과를 통해 데이터베이스의 접근을 최소화하였다. 본 논문이 제안한 방법들의 성능을 검증하기 위해 성능 분석을 수행하였고 그 결과 데이터베이스에서 결과 값을 얻은 경우보다 질의 처리 시간이 3 ~ 4배 향상되었음을 알 수 있었다. 향후 연구로는 허밍 질의 처리 시스템에서 사용되는 가장 일반적인 인덱싱 방법인 N-gram 방법과 성능을 비교함으로써 본 논문에서 제안하는 시스템의 성능이 우수함을 검증하려한다.

참 고 문 헌

[1] P. Rolland, G. Raskinis, and J. Ganascia, “ Musical Content-Based Retrieva : An Overview of the Melodiscov Approach and System ,” In Proc. ACM Multimedia, pp. 81-84, 1999.

[2] A. Ghias, J. Logan, and D. Chamberlin, “ Query By Humming", In Proc. ACM Multimedia, pp. 231-236, 1995.

[3] A. Uitdenbogard, J. Zobel, “ Melodic Matching techniques for Large Music Databases", In Proc. ACM Multimedia, pp. 57-66, 1999.

[4] N. Kosugi, Y. Nishihara, T. Sakata, M. Yamamuro, and K. Kushima,“ A Practical Query-By-Humming System for a Large Music Database", In Proc. ACM Multimedia, pp. 333-342, 2000.

[5] J. Reiss, J. Aucouturier and M. Sandler, “ Efficient multidimensional searching routines for music information retrieval", In Proc. ISMIR, pp. 163-171, 2001.

[6] S. Pauws, “ CubyHum : A Fully Operational Query by Humming System", In Proc. ISMIR, pp. 187-196, 2002.

[7] J. Downie.“ Evaluating a Simple Approach to Music Information Retrieval : Conceiving Melodic N-Grams as Text", Thesis, University of Western Ontario, 1999.

[8] S. Doraisamy, S. Ruger, “ Robust polyphonic music retrieval with n-grams", Journal of Intelligent Information Systems, 21 : 1, pp. 53-70, 2002.

[9] R. B. Dannenberg, Ning Hu, “ Understanding Search Performance in Query-by-humming Systems", In ISMIR, 205. 2004,

[10] R. L. Kline, E. P. Glinert,“ Approximate matching algorithms for music information retrieval using vocal input", In Proc. ACM Multimedia, pp. 130-139, 2003.

[11] L. Prechelt, R. Typke,“ An interface for melody input", ACM Transactions on Computer-Human Interaction, pp. 133-149, 2001.

[12] E. Scheirer,“ Tempo and Beat Analysis

of Acoustic Musical Signals", Acoustic Society of America, 103 : 1 January, pp. 588-601, 1998.

[13] K. Ioannis, N. Alexandros, Apostolos N. Papadopoulos, M. Yannis, " Audio Indexing for Efficient Music Information Retrieva", International Multimedia Modelling Conference, pp. 22-29, 2005.

[14] K. Andreas, " Themefinder : A Web-based Melodic Search Tool, Melodic Similarity : Concepts, Procedures and Applicatons", Computing in Musicology, pp.231-236, 1998.

[15] A. Pienimaki," Indexing Music Databases Using Automatic Extraction of Frequent Phrases", In Proc. ISMIR, pp. 25-30, 2002.

[16] M. Melucci, N. Orio," A Comparison of Manual and Automatic Melody Segmentation", In Proc. ISMIR, pp. 7-14, 2002.

유 진 희

2004. 8. : 한성대학교 정보시스템공학과 (학사)
2005. 2 ~ 현재 : 연세대학교 컴퓨터과학과 석사과정 재학 중
관심분야 : 데이터마이닝. 멀티미디어 데이터베이스
e-mail : jhyou@cs.yonsei.ac.kr

박 상 현

1989. 2. : 서울대학교 컴퓨터공학과 (학사)
1991. 2. : 서울대학교 컴퓨터공학과 (석사)
2001. 2. : UCLA대학교 전산학과 (박사)
1991. 3 ~ 1996. 8. : 대우통신연구원
2001. 2 ~ 2002. 6. : IBM T. J Watson Research Center Post-Doctoral Fellow.
2002. 8 ~ 2003. 8. : 포항공과대학교 컴퓨터공학과 조교수
2003. 9 ~ 현재 : 연세대학교 컴퓨터과학과 조교수
관심분야 : 데이터마이닝, 바이오 인포매틱스, 스트림 데이터 마이닝, 임베디드 데이터베이스, XML
e-mail : sanghyun@cs.yonsei.ac.kr

하 유 민

2004. 8. : 연세대학교 컴퓨터과학과 (학사)
2005. 3 ~ 현재 : 연세대학교 컴퓨터과학과 석사과정 재학 중
관심분야 : 데이터마이닝, 스트림 데이터베이스, 멀티미디어 데이터베이스
e-mail : ymha@cs.yonsei.ac.kr