Unione europea
Fondo sociale europeo

REGIONE DEL VENETO

LICEO GIORGIONE
CASTELFRANCO VENETO

PROGETTO TEKNE

INDICIZZAZIONE DEL LIVELLO SIMBOLICO DEL TESTO MUSICALE

MIMeSI

Codice progetto: 1027/1/4/2027/2014

Tipo: Percorsi didattici di studio, ricerca e sviluppo

Responsabile del progetto:
Prof. Michele Della Ventura

Tutor:
Prof. Michele Della Ventura
Prof.ssa Giuliana Lo Giudice

Studenti:
Costantini Andrea
Loss Tommaso
Moro Nicola
Stocco Francesco
Torresan Lorenzo

Approvato con DDR n. 366 del 16/12/2014 Anno formativo: 2014.

INDICE

Introduzione ……. 2

Lavori precedenti ……. 4

Rappresentazione della partitura …….. 7

Analisi del segmento musicale …… 14

Risultati ottenuti ……. 17

"Teknè, termine greco avente come equivalente successivo latino ars, è un'attività che ha come scopo quello di introdurre dei cambiamenti all'interno del mondo naturale, ovvero produrre oggetti che non si trovano in natura". (L'estetica di Platone)

Introduzione

Variamente definita, Internet è sostanzialmente la "*rete delle reti*", cioè un insieme di reti di computer sparse in tutto il mondo e collegate tra loro, a cui possono accedere migliaia di utenti per scambiare tra loro informazioni di vario tipo. Bill Gates l'ha definito come un "*Sistema nervoso digitale in grado di fornire un flusso ben integrato di informazioni nel luogo giusto e al momento oppotuno*" (dal libro "Business alla velocità del pensiero"). Internet presenta il problema tipico di qualsiasi sistema che contenga troppe informazioni, cioè quello di trovarle quando servono: non esiste un indice completo dei siti web e del loro contenuto.

Oggi la ricerca in Internet si fa principalmente attraverso i cosiddetti motori di ricerca (Google, Yahoo,…), grossi database che si muovono all'interno delle pagine del World Wide Web (www) per catalogare le informazioni secondo dei loro criteri.
Spesso la ricerca si fa per parole chiave e condizioni per trovare o eliminare corrispondenze: appositi algoritmi, consentono di ricercare, individuare e selezionare le informazioni più aderenti richieste.
Internet, ad esempio, permette:

- la compra-vendita di prodotti che, oltre a garantire generalmente una maggiore convenienza, offre la possibilità di recepire con rapidità dettagli riguardanti le caratteristiche del prodotto stesso, dell'acquirente e/o del distributore/venditore;
- individuare a partire dal nome dell'autore, la biografia,la bibliografia, il pensiero e tratti di opere (se rese disponibili);
- individuare patologie mediche partendo dalla corrispondenza con la descrizione dei sintomi;
- effettuare una traduzione istantanea delle pagine appartenenti al world wide web garantendone un'accessibilità e una divulgazione più ampia;
- effettuare traduzioni online di testi dell'utente;
- di accedere alle risorse che consentono uno studio e un approfondimento di una lingua straniera.
- …

Sostanzialmente tutto ciò che è testo, o riconducibile ad esso, può essere trovato e proposto all'utente.

Tutti i comuni motori di ricerca si basano sull’individuazione d’informazioni collegate ad una determinata sequenza di caratteri. I risultati possono essere determinati da diversi elementi:dalla stringa di testo inserita, dalla frequenza con cui le varie pagine vengono visualizzate oppure dall’azione di cookies che memorizzano le preferenze degli utenti.
In campo musicale, a parte informazioni testuali storiche/musicologiche, risulta invece mancante una vera e propria ricerca degli spartiti a livello simbolico (figura 1).

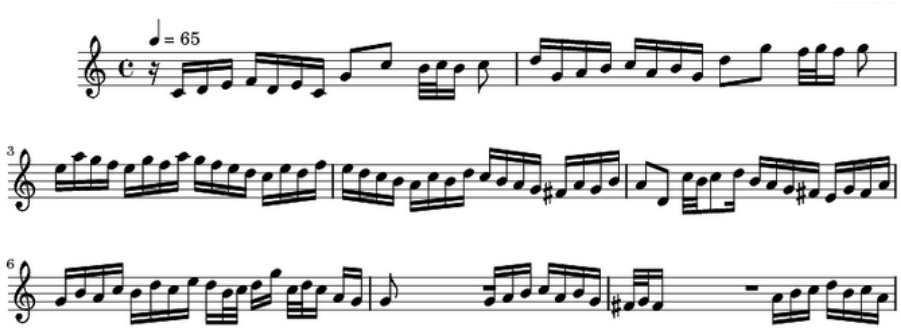


Figura 1 - rappresentazione a livello simbolico dell'invenzione a due voci di Bach BWV 772

Questo studio vuole presentare MIMeSI (Motore d’Indicizzazione Musicale e Similarità Informatica)un algoritmo in grado di ricercare, all’interno di un database condiviso, una partitura partendo dal suo livello simbolico.
L’obiettivo di MIMeSI è quello di offrire la possibilità a coloro che hanno ascoltato una melodia, di poterla ricercare in modo diretto tramite l’inserimento della sequenza stessa di note, senza la necessità di inserire dati testuali come il nome dell’autore o il titolo del brano.

Lavori Precedenti

In ambito musicale l'indicizzazione, ovvero l'organizzazione dei dati nel web, rischia molto spesso di diventare un'operazione difficile nel momento in cui vengono presi in esame gli elementi musicali.
I motori di ricerca classici (primo fra tutti Google) hanno un grosso limite: il testo. Incapaci dunque di concentrarsi sul suono, i motori di ricerca come Youtube utilizzano tag cioè etichette o parole chiave. La ricerca e l'indicizzazione in ambito musicale, concentrata sul segmento sonoro, viene a mancare anche se negli ultimi anni si sta cercando di colmare questa esigenza.

Un'interessante applicazione per smartphone è Shazam (http://www.shazam.com/) (figura 2).
Shazam funziona analizzando il suono catturato e cercando il brano corrispondente sulla base di una "impronta acustica" in un database di oltre 11 milioni di canzoni.

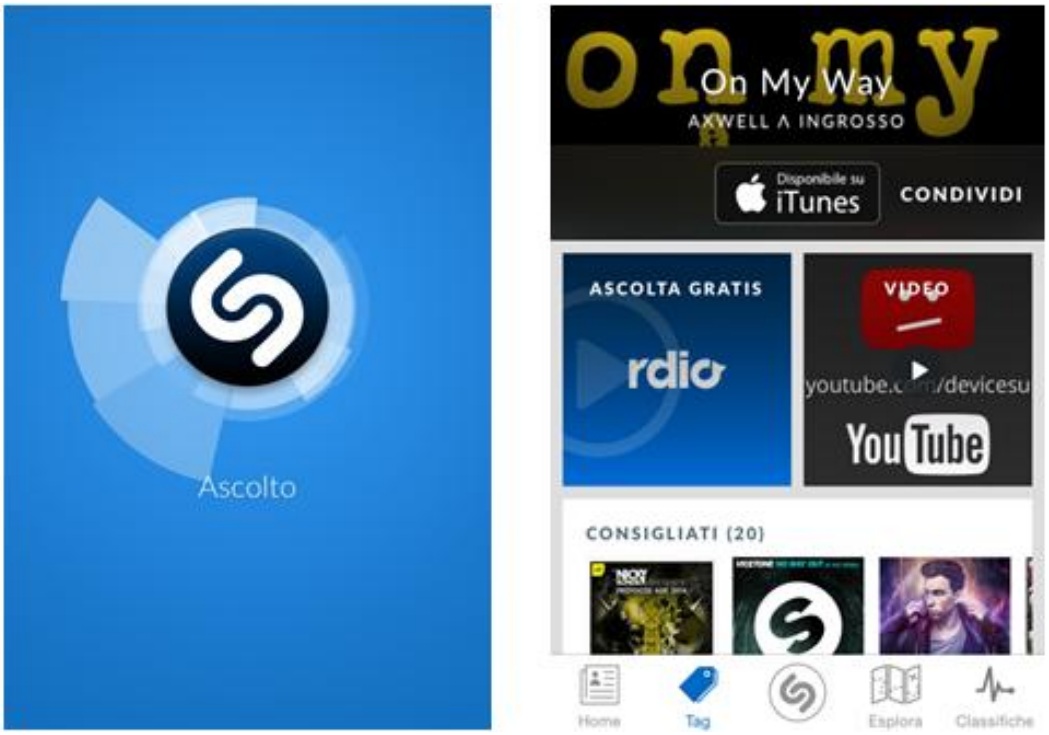
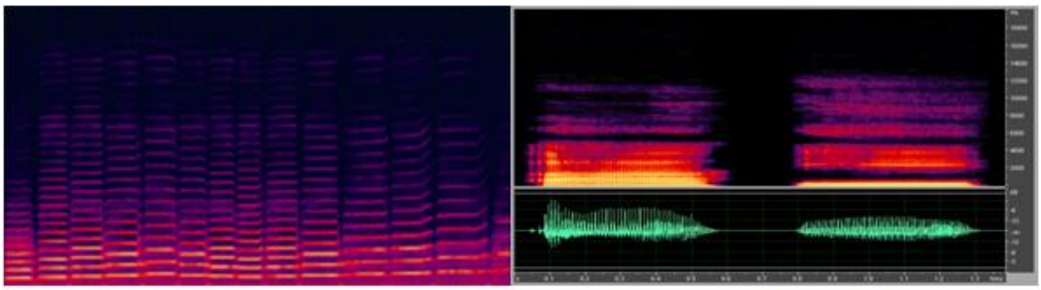


Figura 2 - Screenshot di Shazam

Shazam identifica le canzoni confrontando l'impronta digitale dell'audio campione, basato su un grafico tempo-frequenza chiamato spettrogramma.

Shazam memorizza un catalogo d'impronte audio digitali in un database. L'utente etichetta (tagga) la canzone per 10 secondi e l'applicazione crea un'impronta audio digitale basata su alcuni dei segmenti dello spettrogramma semplificato e l'area di destinazione (figura 3).



Spettrogramma del suono di un violino Spettrogramma di due vocali "a" e "i"

Figura 3 - Esempio di spettrogramma

Con l'impronta audio digitale, Shazam inizia la ricerca di somiglianze nel database. Se vi è una corrispondenza, l'informazione viene restituita all'utente, altrimenti viene restituito un errore.

Shazam può identificare musica preregistrata trasmessa da qualsiasi sorgente, come ad esempio la radio, la televisione, il cinema o i club, a condizione che il livello di rumore di fondo non sia molto alto, per evitare che un'impronta audio digitale venga catturata e che la canzone non venga riconosciuta nel database del software.

I limiti di Shazam sono invece rintracciabili nel momento in cui il soggetto da ricercare è un motivo attinente alla musica classica. Infatti, per quanto riguarda questa tipologia musicale, essa presenta caratteristiche particolari, individuabili soprattutto nelle varie interpretazioni che possono essere ricavate da un unico brano. Molto spesso si vengono a creare varie sfaccettature che rendono ogni rappresentazione diversa dalle altre e ciò crea evidenti difficoltà per un utilizzo di un'applicazione che confronta il proprio database di spettrogrammi con un brano che risulta diverso.

Esistono anche siti come Petrucci International Music Score Library Project (http://imslp.org/) e Progetto Mutopia (http://www.mutopiaproject.org/) utili soprattutto nel momento in cui l'oggetto della ricerca diventa la partitura in formato pdf.

Questi siti hanno un database di partiture di pubblico dominio e sono privi di motori di ricerca basati sull'audio: è possibile effettuare una ricerca attraverso parole chiave testuali come autore o titolo della composizione.

Ad oggi non è stato realizzato un algoritmo specifico che consenta di ricercare in un database un brano musicale a partire dal suo livello simbolico. Questo è dovuto essenzialmente all’aspetto non propriamente commerciale di tale necessità in quanto l’utilizzo dell’algoritmo risulta di interesse principalmente per musicisti (professionisti o dilettanti) e non per un pubblico generico.

Rappresentazione della partitura

L'analisi delle partiture per l'indicizzazione richiede la loro rappresentazione in un formato numerico, in modo da poter effettuare su di esse operazioni logico-matematiche.
Per questo MIMeSI legge le partiture scritte secondo il protocollo MIDI (Musical Instrument Digital Interface), ormai diventato una vera e propria lingua nelle applicazioni informatiche rivolte alla musica.
Un MIDI-file contiene un flusso di eventi MIDI, a ognuno dei quali associa un'informazione temporale, o *timestamp* (figura 4).
Si possono memorizzare le informazioni relative alla song e alle tracce che la compongono, compreso il tempo di metronomo e il metro (4/4, 3/4, 6/8,…).
Infine lo Standard Midi File permette anche di definire informazioni descrittive come i nomi delle tracce e degli strumenti o eventuali avvisi di copyright e testi delle canzoni.

Figura 4 - Timestap e dati MIDI

Ogni evento MIDI è preceduto da un numero (*timestamp*), che vi si riferisce, e che rappresental'intervallo di tempo che separa un evento dal precedente (figura 4). L'intervallo prende il nome di *delta-time* e può avere due significati, a seconda che il tempo sia calcolato in termini relativi e assoluti.

Nel primo caso, l'intervallo può essere calcolato in "tick": il timestamp indica allora quanti impulsi di clock il dispositivo deve aspettare per rendere attivo il successivo messaggio MIDI.
Nel secondo caso, il delta-time può indicare le suddivisioni di un secondo, rendendo così possibile il riferimento al tempo assoluto. In questo caso nel MIDI-file viene indicato anche il valore di suddivisioni del secondo in *frame* (24, 25, 29, 30).
Tra le informazioni veicolate dai messaggi, bisogna fare una distinzione tra eventi e meta-eventi (figura 5).
Sono eventi tutti i messaggi MIDI tipo Channel Message (messaggi relativi al canale), System Message (messaggi di sistema).

Tipo di meta-evento	Commento
01	Evento di testo per descrivere il nome di una traccia o il nome della voce usata; può essere usata anche per inserire il testo
02	Evento di copyright, il testo deve contenere la © seguita dall'anno di creazione e dal nome dell'autore
03	Evento di testo per indicare il nome della traccia
05	evento di testo per inserire le liriche del brano
2F	Evento che indica la fine di un blocco di traccia (Track.chunk)
51	Evento di velocità del brano: indica quanti microsecondi ci devono essere in un quarto di nota
58	Evento per descrivere la divisione del brano
59	Evento che indica la tonalità

Figura 5 - Descrizione dei meta-eventi

I MDI-file sono costituiti essenzialmente da *chunk* ovvero da blocchi d'informazione separata, ognuno dei quali ha una propria lunghezza espressa in byte.

Due sono i tipi di chunk definiti:

- *Headerchunk* (o chunk d'intestazione): ha una dimensione fissa di 14 byte ed è sempre all'inizio del file perché dichiara tre proprietà generali: il formato, il numero di tracce e la divisione in PPQ (*Parts per Quarter*, parti per quarto: l'unità di misura é relativa e non assoluta, in quanto dipende dal valore assunto dal quarto).

<Header Chunk> = <chunk type><length><format><ntrks><division>

- Il chunktype è costituito da 4 caratteri ASCII "MThd"
- Length è una rappresentazione in 32 bit del numero 6 (high byte first), cioé il numero di byte che, come vedremo, occupano i campi format, ntrks, division.
- Il terzo campo, format, è costituito da 2 byte (integer/short) che può assumere solo tre valori: 0, 1, 2.
  Nel formato 0 tutte le in formazioni sono conservate in una sola traccia.
  nel formato 1 è possibile definire più tracce autonome che devono essere eseguite simultaneamente.
  Nel formato 2 l'informazione viene memorizzata in più tracce separate ma sequenziali.
- Il quarto campo, ntrks, sempre costituito da 2 byte (integer/short), contiene il numero di tracce contenute nel MIDI file. Se il MIDI file è in formato 0, ntrks vale sempre 1.
- Il quinto ed ultimo campo, division, specifica il significato dei delta-time, cioè dei valori temporali che separano tra loro gli eventi MIDI (come vedremo meglio descrivendo i trackchunk). Questo campo è sempre di 16 bit (integer/short) e può essere espresso in due formati: metrical time e time-code-based time.

- *Trackchunk* la cui dimensione e il cui numero (salvo il caso del formato 0) è variabile (figura 6). I trackchunk hanno esattamente la stessa struttura in tutti e tre i formati. Ogni trackchunk è costituito dalla sequenza di eventi e meta-eventi MIDI relativi a una traccia, separati dall'indicazione dall'intervallo del delta-time.

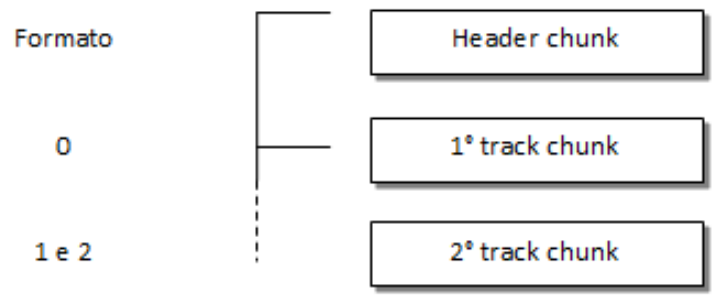
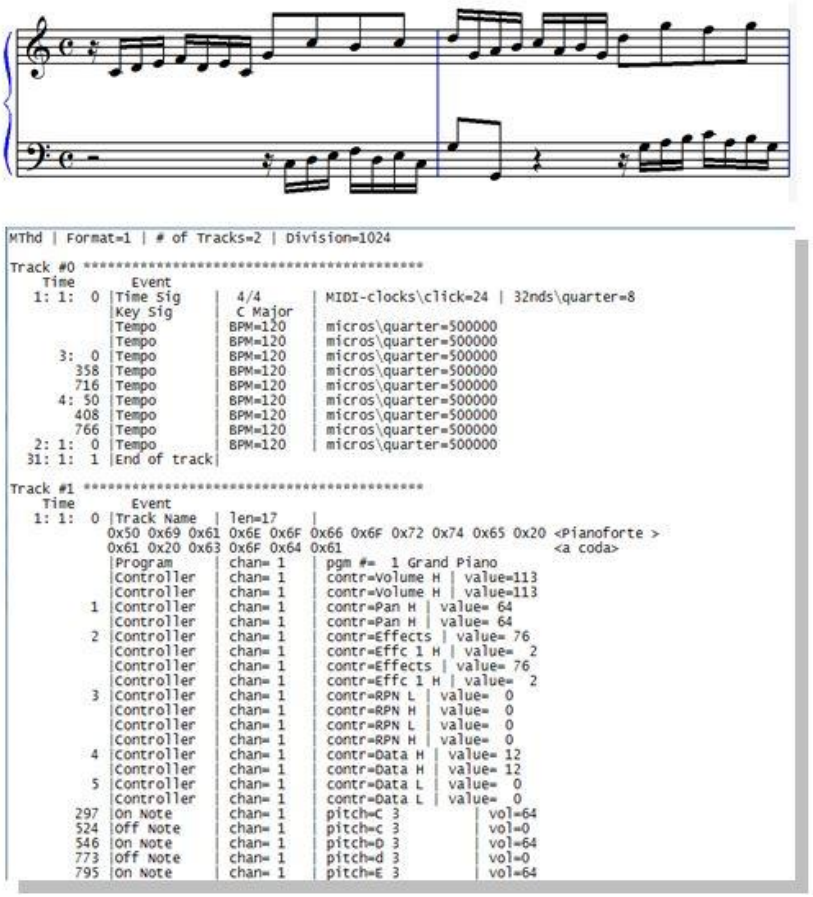


Figura6 - Header e track chunck

Ciascuna partitura musicale in formato MIDI può essere rappresentata attraverso una tabella contenente tutte le informazioni in formato testuale. In figura 7 sono rappresentate le prime due battute dell'invenzione a due voci di Bach BWV 772 e la corrispondente rappresentazione testuale.



```
MThd | Format=1 | # of Tracks=2 | Division=1024

Track #0 ****************************************
    Time       Event
   1: 1:  0 |Time Sig     |  4/4       | MIDI-clocks\click=24 | 32nds\quarter=8
            |Key Sig      |  C Major   |
            |Tempo        | BPM=120    | micros\quarter=500000
            |Tempo        | BPM=120    | micros\quarter=500000
      3:  0 |Tempo        | BPM=120    | micros\quarter=500000
        358 |Tempo        | BPM=120    | micros\quarter=500000
        716 |Tempo        | BPM=120    | micros\quarter=500000
      4: 50 |Tempo        | BPM=120    | micros\quarter=500000
        408 |Tempo        | BPM=120    | micros\quarter=500000
        766 |Tempo        | BPM=120    | micros\quarter=500000
   2: 1:  0 |Tempo        | BPM=120    | micros\quarter=500000
  31: 1:  1 |End of track |

Track #1 ****************************************
    Time       Event
   1: 1:  0 |Track Name   | len=17     |
            0x50 0x69 0x61 0x6E 0x6F 0x66 0x6F 0x72 0x74 0x65 0x20 <Pianoforte >
            0x61 0x20 0x63 0x6F 0x64 0x61                          <a coda>
            |Program      | chan= 1    | pgm #=  1 Grand Piano
            |Controller   | chan= 1    | contr=Volume H | value=113
            |Controller   | chan= 1    | contr=Volume H | value=113
          1 |Controller   | chan= 1    | contr=Pan H | value= 64
            |Controller   | chan= 1    | contr=Pan H | value= 64
          2 |Controller   | chan= 1    | contr=Effects | value= 76
            |Controller   | chan= 1    | contr=Effc 1 H | value=  2
            |Controller   | chan= 1    | contr=Effects | value= 76
            |Controller   | chan= 1    | contr=Effc 1 H | value=  2
          3 |Controller   | chan= 1    | contr=RPN L | value=  0
            |Controller   | chan= 1    | contr=RPN H | value=  0
            |Controller   | chan= 1    | contr=RPN L | value=  0
            |Controller   | chan= 1    | contr=RPN H | value=  0
          4 |Controller   | chan= 1    | contr=Data H | value= 12
            |Controller   | chan= 1    | contr=Data H | value= 12
          5 |Controller   | chan= 1    | contr=Data L | value=  0
            |Controller   | chan= 1    | contr=Data L | value=  0
        297 |On Note      | chan= 1    | pitch=C 3          | vol=64
        524 |Off Note     | chan= 1    | pitch=C 3          | vol=0
        546 |On Note      | chan= 1    | pitch=D 3          | vol=64
        773 |Off Note     | chan= 1    | pitch=d 3          | vol=0
        795 |On Note      | chan= 1    | pitch=E 3          | vol=64
```

Figura 7 - Invenzione a due voci di Bach BWV 772

Ogni chunk (sia esso header o track) inizia con l'indicazione del type (tipo: appunto, header o track), e con l'indicazione della lunghezza (length) della sezione dati successiva: type e length occupano ciascuno 4 byte. La prima parte di ogni chunk è perciò costituita da 8 byte, cui seguono altri byte, secondo il numero dichiarato nella length. La prima riga è la visualizzazione dell'headerchunk. *MThd* sta per MIDI Trackheader, e i 4 caratteri ASCII occupano un byte ciascuno. I quattro byte successivi, che specificano la length, non vengono visualizzati per comodità: in ogni caso nell'header la length è sempre 6, poiché le restanti informazioni sono costituite da 3 coppie di byte. La prima coppia specifica il formato (*Format*= 1), la seconda il numero delle tracce (*# of Tracks* = 2), la terza la divisione (*Division* = 1024). Se la Division fosse negativa, allora i delta-time codificherebbero un intervallo di tempo secondo l'SMPTE (*Society of Motion Picture and Televi-*

sionEngineers: metodo standard di sincronizzazione audio e video. Il codice SMPTE è assoluto, cioè esprime completamente l'intervallo di tempo tra due eventi che hanno un indirizzo espresso in SMPTE, non come il MIDI clock che esprime gli intervalli come tick per nota da un quarto, non esprimendo la velocità metronometrica). Dopo l'header segue il primo trackchunk.

Essendo il MIDI-file considerato, in formato 1, la prima traccia (0) contiene le informazioni temporali (tempo map), e altre informazioni generali, poste all'inizio del file per essere reperite facilmente. La colonna *Time* indica il tempo in misure, quarti e delta-time del quarto precedente, mentre quella *Event* contiene una descrizione degli eventi. Nel caso vi sia del testo, i caratteri sono riportati in esadecimale, con la visualizzazione alfanumerica tra parentesi uncinate. In apertura (*Time 1:1:0*), alcuni meta-eventi segnalano informazioni relative al nome della traccia corrente o al copyright. Altri meta-eventi descrivono la tempo map: l'indicazione di time signature (*Time Sig 4/4*) specifica anche il numero di trentaduesimi per quarto (*32nds/quarter = 8, cioè 8 trentaduesimi per quarto*).

Il tempo è espresso in BPM per comodità di lettura, ma segue l'indicazione relativa alla durata in microsecondi del quarto (*micros/quarter = 500000*). Chiude il trackchunk un meta-evento specifico e obbligatorio, *End of Track*. La traccia successiva contiene le informazioni relative alla prima riga della notazione. In figura 6 abbiamo indicato alcune relazioni tra eventi della traccia 1 e la rispettiva notazione musicale.

Nel MIDI-file, al messaggio *End of Track*fa seguito un altro trackchunk.

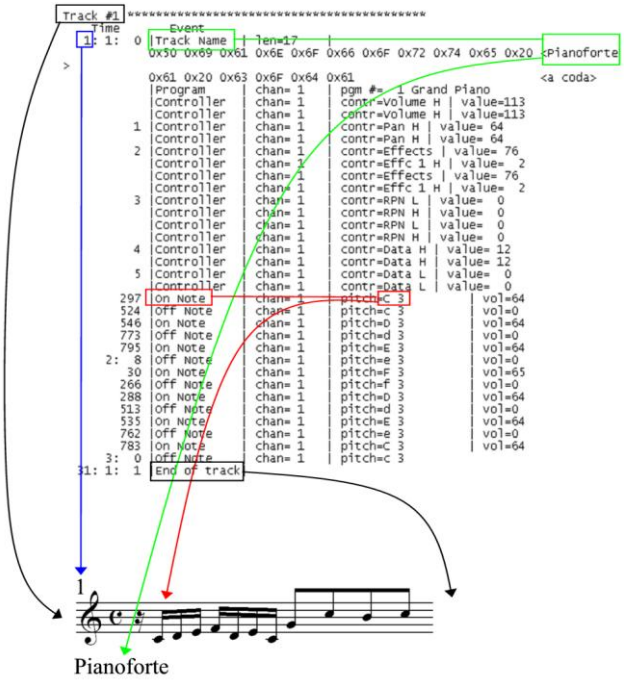


Figura 8 - Notazione e eventi MIDI

Ad ogni nota corrisponde un numero univoco identificativo. Al tasto più grave del pianoforte è attribuito il valore 1, al successivo il valore 2 e così via (figura 9).

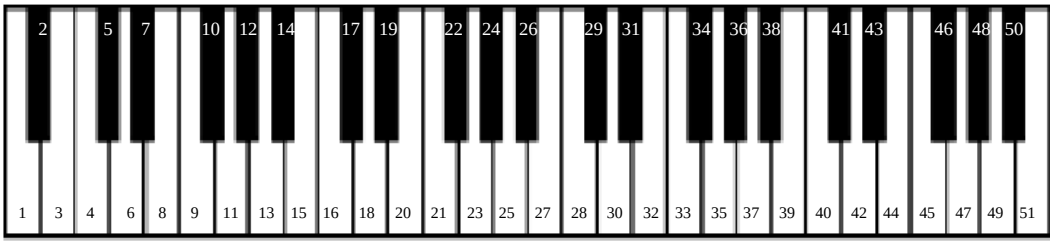


Figura 9 - Rappresentazione numerica della tastiera

Nel caso specifico dell’invenzione a due voci n.1di Bach, si ha la seguente rappresentazione numerica (figura 10):

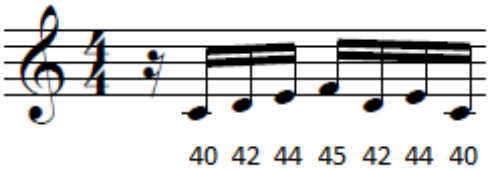


Figura 10–Rappresentazione numerica della partitura

Con questa rappresentazioneè possibile ricavare l’andamento intervallare della partitura, calcolandone i delta tra due note consecutive cioè il numero di semitoni che separano le due note stesse (tabella 1).

Note	**Tipo intervallo**	**Delta**
Do-Re	Ascendente	2
Re-Mi	Ascendente	2
Mi-Fa	Ascendente	1
Fa-Re	Discendente	-3
Re-Mi	Ascendente	2
Mi-Do	Discendente	-4

Tabella 1 – Struttura intervallare della partitura

La melodia può quindi essere rappresentata come il seguente vettore intervallare: <2 2 1 -3 2 -4>.

Per quanto riguarda la componente ritmica della partitura essa può conferire un criterio di precisione aggiuntiva ma allo stesso tempo limitare l’utilizzo del software ad una ristretta cerchia di utenti. Per questo si è deciso di limitare MIMeSI ad un approccio esclusivamente melodico.

Analisi del segmento musicale

Il principio di funzionamento di MIMeSI si basa sul confronto intervallare del segmento musicale indicato dall'utente con segmenti estrapolati dalle partiture MIDI. Questo confronto può essere fatto con segmenti identici, simili o affini per caratteristiche strutturali.

Identicità

Due segmenti musicali si definiscono identici qualora i vettori intervallari che li rappresentano risultano uguali (figura 11).

Ad esempio l'intervallo Fa-Sol-La (< 2 2 >) risulta identico all'intervallo Do-Re-Mi (< 2 2 >).

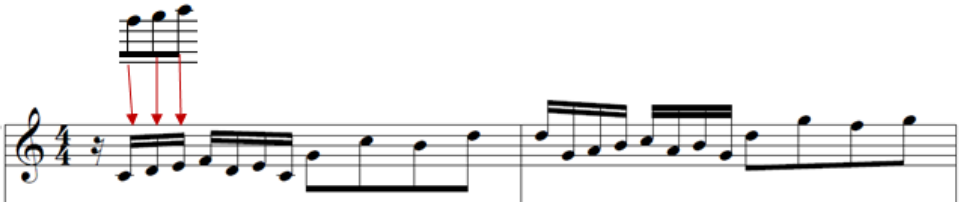


Figura 11 - Segmento dall'invenzione n°1 di Bach

Retrogradazione

Un segmento musicale è retrogradato qualora venga invertito l'ordine delle note, quindi degli intervalli che lo compongono (figura 12).

Ad esempio l'intervallo Fa-Sol-La (<2 2 >) retrogradato diventa La-Sol-Fa (< -2 -2 >).

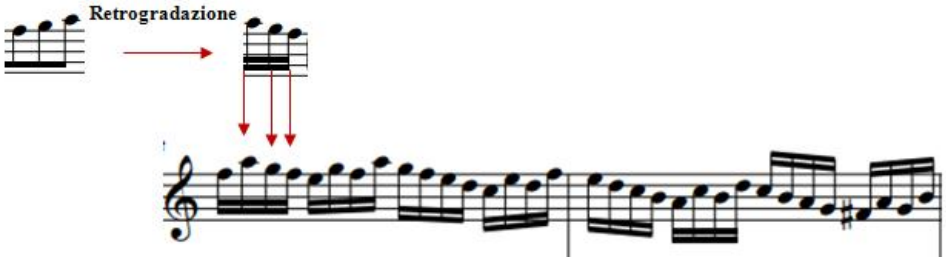


Figura 12 - Segmento dall'invenzione n°1 di Bach

Inversione

L'inversione di un segmento musicale presenta un andamento opposto rispetto a quello del segmento originale (figura 13).

Ad esempio l'intervallo Do-Sol-Fa-Re (< 7 -2 -3 >) invertito diventa Do-Fa-Sol-Si*b* (< -7 2 3 >).

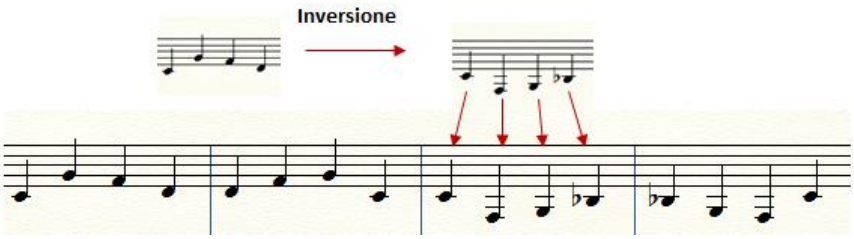


Figura 13 - Segmento con esempio di inversione

Retrogradazione dell’inversione

L’inversione del retrogrado di un segmento musicale presenta un andamento risultato della combinazione della retrogradazione e dell’inversione (figura 14).

Ad esempio l’intervallo Do-Sol-Fa-Re (< 7 -2 -3 >) diventa Si*b*-Sol-Fa-Do (< -3 -2 7 >).

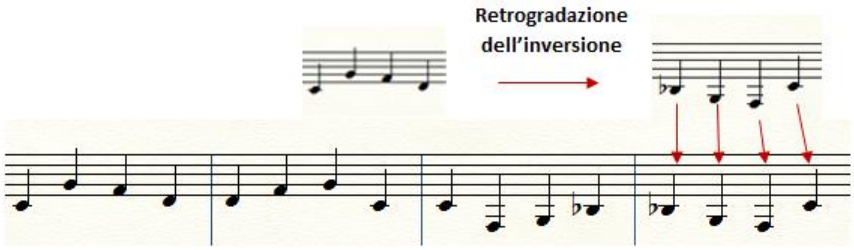


Figura 14 - segmento con esempio di retrogradazione dell'inversione

Affinità

Due segmenti musicali sono affini quando gli intervalli del primo sono trovati nel secondo anche se tra i suoni della melodia ne sono presenti altri con lo stesso andamento (figura 15).

Ad esempio l’intervallo Do-Fa-Re-Mi-Do (< 5 -3 2 -4 >) è affine all’intervallo Do-Re-Mi-Fa-Re-Mi-Do (< 2 2 1 -3 2 -4 >).

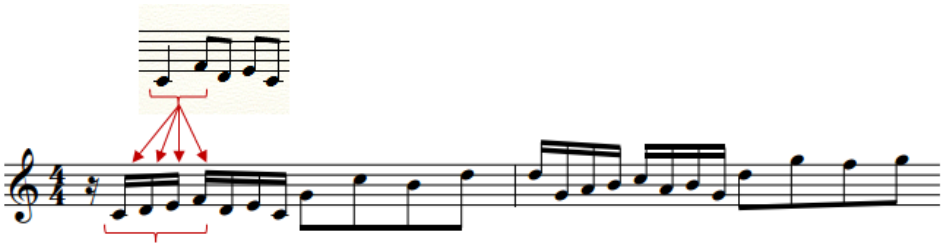


Figura 15 - segmento dall'invenzione n°1 di Bach

Similarità

Due segmenti musicali risultano similari qualora il loro andamento (ascendente o discendente) coincida indipendentemente dai valori dei loro intervalli (figura 16).

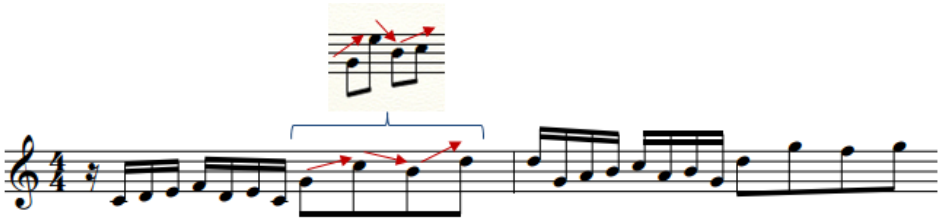


Figura 16 - segmento dall'invenzione n°1 di Bach

Risultati ottenuti

Il programma MIMeSI è un motore di ricerca, ovvero un algoritmo in grado di analizzare un insieme di dati e restituire unindicedei contenuti disponibili classificandoli in base al grado di rilevanza. Differisce però dai comuni motori testuali (Google, Yahoo, ecc...) nell'oggetto della ricerca e negli algoritmi utilizzati per l'indicizzazione: l'input dell'utente consiste in una serie di note.
Questo "motivo" viene ricercato in un Database di spartiti dall'algoritmo, che stila la lista dei risultati secondo regole derivate dall'arte della composizione, quindi propriamente musicali.

MIMeSI consiste in un'interfaccia grafica realizzata in html e php (figura 17), tramite la quale l'utente inserisce la propria chiave di ricerca e, a processo completato, legge i risultati.

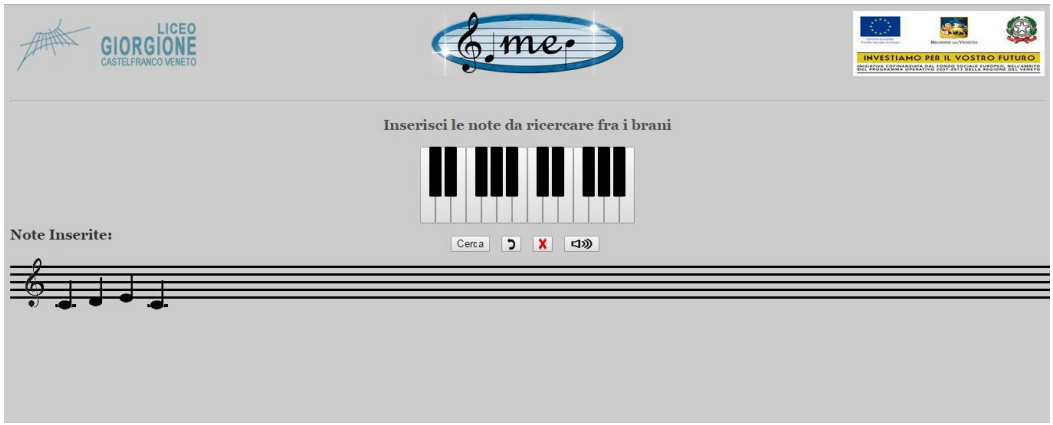


Figura 17 - Interfaccia di MIMeSI

L'algoritmo traduce le note inserite in notazione numerica, scrivendole in un file in formato CSV (Comma SeparatedValues). Questo file viene letto da un programma in C++, che effettua la ricerca vera e propria nel Database (figura 18).

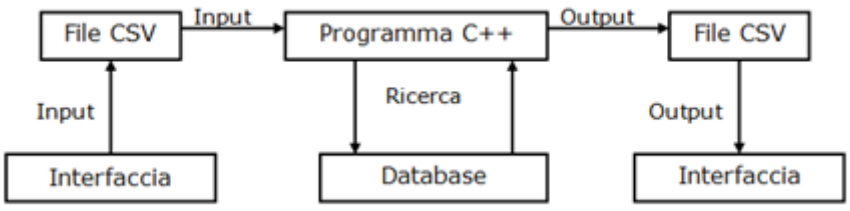


Figura 18 – Diagramma di flusso diMIMeSI

Il "motivo" viene inserito dall'utente attraverso una tastiera virtuale realizzata mediante l'interfaccia html (figura 17) restituendo come feedback il suono e il livello simbolico corrispondente. L'interfaccia presenta quattro funzioni attivabili mediante pulsanti dedicati: cerca, back, reset e riproduci (figura 19).

Figura 19 - Pulsanti dedicati

Premendo il pulsante "cerca" viene lanciato il programma in C++ che svolge il lavoro di analisi comparando il "motivo" inserito con le partiture di ogni brano presente nel database attraverso i criteri di analisi musicale precedentemente descritti.
MIMeSI restituisce, al termine dell'elaborazione, i risultati ordinati in una tabella la quale permette la riproduzione del brano e la visualizzazione della partitura dello stesso (figura 20).

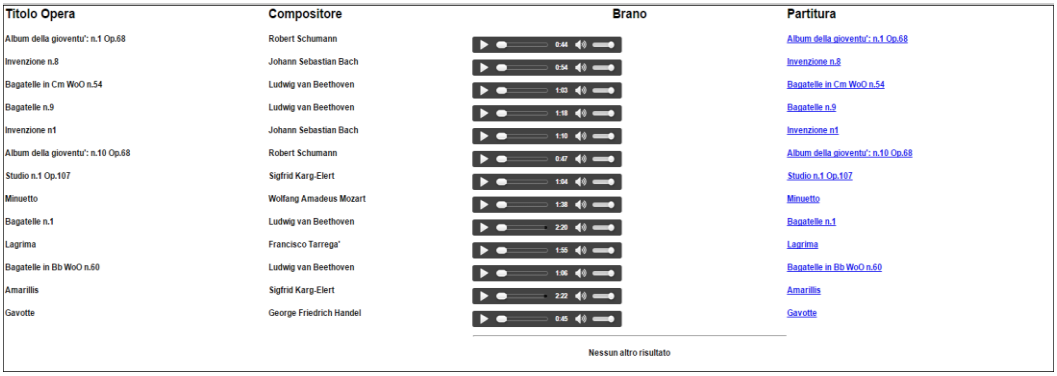


Figura 20 – Riepilogo risultati

Funzioni di ricerca

Le funzioni utilizzate da MIMeSI sono tre, ognuna dedicata ad un criterio specifico. Ciascuna funzione utilizza un ciclo d'iterazione per effettuare la comparazione del vettore "motivo" con il vettore spartito mediante traslazione. Il programma effettua, per ciascun brano, una comparazione degli intervalli, partendo dalla prima nota, per ciascun gruppo di N note. Se il campione da ricercare è do-re-mi (< 2 2 >) e lo spartito è composto da 1-2-3… intervalli, il programma farà il confronto con il vettore< 2 2 >,< 1 2 >,< 2 3 >, e così via…

- **Identicità**: questo nucleo dell'algoritmo (figura 21) svolge il semplice confronto, intervallo per intervallo, individuando il numero di uguaglianze secondo il princi-

pio d'identicità. Questa funzione viene utilizzata per la ricerca di retrogradi, inversi e retrogradi inversi successivamente alla modifica del "motivo" inserito attraverso le appositi funzioni.

```
for (int i=1;i<=(DIMN-l_delta);i++)
{
    for (int j=1;j<=l_delta;j++)
    {
        if ((S[i+j-1]>88)||(S[i+j-1]<-88)) break;
        if (S[i+j-1]==C[j]) c++;
        else break;
        if (c==l_delta) punteggio++;
    }
    c=0;
}
```

Figura 21 - Funzione di ricerca identicità

- **Affinità**: questo nucleo dell'algoritmo (figura 22) svolge il confronto, intervallo per intervallo, secondo il principio di affinità.

```
for (int j=1;j<=(DIMN-i);j++)
{
    if ((S[i+j-1]>88)||(S[i+j-1]<-88)) break;
    if ((S[i+j-1]*C[n])>0)
    {
       if (((delta+S[i+j-1])>C[n])&&((delta+S[i+j-1])>0)) break;
       if (((delta+S[i+j-1])<C[n])&&((delta+S[i+j-1])<0)) break;
       if ((delta+S[i+j-1]) == C[n])
       {
          n++;
          if ((n-1)==l_delta)
          {
              Intruso[intruso]++;
              break;
          }
          delta = 0;
       }
       else
       {
           delta+=S[i+j-1];
           intruso++;
       }
    }
    else break;
}
```

Figura 22 - Funzione di ricerca dell'affinità

- **Similarità**: questo nucleo dell'algoritmo (figura 23) svolge il confronto, intervallo per intervallo, secondo il principio di similarità.

```
for (int i=1;i<=DIMN;i++)
{
    for (int j=1;j<=l_delta;j++)
    {
        if ((S[i+j-1]>88)||(S[i+j-1]<-88)) break;
        if ((S[i+j-1])*C[j]>0)
        {
            c++;
            sommap+=int(sqrt((C[j]-S[i+j-1])*(C[j]-S[i+j-1])));
            if (c==l_delta)
            {
                    Somma[sommap]++;
                    break;
            }
        }
        else break;
    }
    c=0;
    sommap=0;
}
```

Figura 23 - Funzione di ricerca della similarità

Assegnazione del punteggio

Per ciascun segmento musicale identico, retrogrado, inverso e retrogrado dell’inverso, MIMeSI assegna un punto allo spartito. In questo modo non si esclude all’utente la possibilità di aver ricordato una delle versioni “alternative” del tema ricercato, invece di quella canonica. Questa possibilità diventa importante per le musiche scritte dal periodo barocco in poi, dove questi artifici musicali sono strumenti della composizione.

Per ciascun intervallo affine viene assegnato un punteggio calcolato dividendo 1 (cioè il punto assegnato ad un intervallo identico) per il numero di “intrusi” (ignorati dal programma per trovare l’intervallo stesso). Perciò, esclusi i casi con un singolo intruso, più il numero di intrusi aumenta, meno rilevanza avrà l’intervallo nel determinare il punteggio associato al brano rispetto ad un intervallo identico.

Per ciascun intervallo similare viene assegnato un punteggio diviso per la somma di delta in eccesso, calcolati dal programma, maggiorata di uno. Perciò più aumenta il numero meno rilevanza avrà l’intervallo nel determinare il punteggio associato al brano rispetto ad un intervallo identico.

La ragione della diversità nel calcolo del punteggio fra affini e similari risiede nel fatto che gli affine hanno statisticamente una maggiore incidenza, perciò assegnare ad un intervallo affine con un semitono in eccesso lo stesso punteggio di uno identico porta a dare un peso eccessivo a questa categoria.
Completata la ricerca MIMeSI, tramite un algoritmo di ordinamento di tipo Heapsort, ordina i risultati ottenuti in base al punteggio, escludendo quelli con punteggio nullo (figura 20).