# Sound of Data

**Riccardo Cervero** 000000
**Marco Ferrario** 000000
**Pranav Kasela** 000000
**Federico Moiraghi** 799735

*Universit degli Studi di Milano Bicocca*

Anno Accademico 2018/19

## Indice

**Obiettivo del progetto è analizzare la discussione mediatica riguardante i soggetti del mondo musicale contemporaneo e passato.**

# Parte I
# Introduzione

Per raggiungere i traguardi posti dal progetto *Sound of Data*[1], si son dovuti raccogliere dati sufficienti per costruire un *knowledge graph* adeguato alla materia: scaricato un *dump* di `musicbrainz.org` come database relazionale (già esportato in formato *Tabular Separated Values* dai manutentori), si è dapprima importato in Apache Hadoop[2] per effettuare una rapida pulizia preliminare e infine esportato in modo tale da costruire un grafo Neo4J[3]. Costruita quindi la base di conoscenza su cui operare, si sono raccolti *tweet* in tempo reale grazie ad Apache Kafka[4], per poi analizzarli in automatico con un rudimentale strumento di *instance matching*.

# Parte II
# Costruzione dello *knowledge graph*

## 1 Data Cleaning con python e Pig

I dati vengono scaricati dal sito di musicbrainz attraverso il file get_data.py, che oltre a scaricare i dati effettua un'analisi preliminare di sostituire tutti i caratteri "
N" con campi vuoti in tutti i file attraverso il comando sed e salva solo i file necessari per il database a grafo che si vuole creare. La lista dei generi è presente sempre sul sito di musicbrainz, con uno scraper viene salvato nel codice e confrontato con il file tag e vengono eliminati tutti i tag che non sono dei generi.

I file vengono trasferiti su HDFS per essere processati, in particolare usiamo Apache Pig, che è una piattaforma per processare i dati, la decisione di utilizzare Pig piuttosto che SQL è dovuta al fatto che il caricamento dei dati in SQL è molto lenta e inoltre Pig riesce a sfruttare la potenza di HDFS. L'engine usato con Pig è Tez, che rende molto più veloce l'esecuzione del processamento dei dati saltando diverse fasi di scrittura dei risultati parziali su HDFS.

Una volta finita la pulizia i dati vengono trasferiti fuori da HDFS sul filesystem, e vengono unite i risultati di pig con il comando cat. A questo punto i dati sono pronti per essere caricati su Neo4j.

## 2 Neo4J

I dati puliti da Pig vengono caricati su neo4j usando neo4j-import.

[1] `https://github.com/pkasela/Sound-of-Data`
[2] `https://hadoop.apache.org`
[3] `https://neo4j.com`
[4] `https://kafka.apache.org`

# Parte III
# Analisi dei tweet

## 3 Analisi con Apache Kafka

## 4 Riconoscimento delle istanze nel testo

Data la mole di tweet scaricabili, si è deciso di costruire un strumento di *instance matching* creato *ad hoc* per i tweet. Vista la scarsa abilità di algoritmi basati su reti neurali e *deep learning* ad analizzare il breve (e quindi decontestualizzato) testo di un tweet, si è costruito un modello per l'identificazione di ipotetiche entità su cui basarsi per confronti col database (grazie alla API[5] offerta da `musicbrainz.org` stesso).

### 4.1 Identificazione delle entit

Le entità sono riconosciute non mediante *machine learning* ma grazie a semplici stratagemmi linguistici. Prima di tutto il testo del tweet è ripulito da eventuali abbreviazioni gergali; subito dopo sono ricercate le parole nel testo che non risultano essere italiane: analizzando tweet in lingua italiana, si presume che una stringa in lingua diversa abbia una certa importanza. Per fare questo è usato un mero correttore ortografico che evidenzia quali parole non sono riconosciute; di aiuto nel compito è anche una semplice espressione regolare che tenta di stabilire quali parole non seguono la costruzione sillabica italiana (rientrando quindi o nella categoria dei sostantivi della quinta classe o, nei casi fortunati, nelle entità cercate): secondo le regole linguistiche, una sillaba correttamente formata è composta da un numero massimo di tre consonanti seguita da una vocale e da al più una sola consonante (o vocale con suono consonantico, formando quindi un dittongo). Alle entità così individuate si aggiungono tutte le parole scritte in maiuscolo (che in un testo di così bassa formalità non sempre coincidono coi nomi propri), anche se a inizio frase. Grazie all'uso della punteggiatura (le virgolette) e delle preposizioni, si tenta inoltre di stabilire se l'entità rilevata è un presunto autore o una presunta opera e quindi cercata all'interno del database.

#### 4.1.1 Prestazioni del modello

#### 4.1.2 Margini di miglioramento

### 4.2 Riconoscimento delle entit

[5] `https://python-musicbrainzngs.readthedocs.io/en/v0.6/`

## Parte IV
# Visualizzazioni dei dati

## Parte V
# Risultati e conclusioni