Programmazione di

Applicazioni Data Intensive

Introduzione al Corso

Laurea in Ingegneria e Scienze Informatiche
DISI – Università di Bologna, Cesena

Proff. Gianluca Moro
Roberto Pasolini (laboratorio)
Dip. di Informatica - Università di Bologna, Cesena
nome.cognome@unibo.it

Crescente Quantità di Dati Disponibili

- I dati sono prodotti costantemente e in grandi quantità
  - La crescita del World Wide Web ha portato ad avere una enorme quantità di informazione disponibile pubblicamente
  - Analogamente sono aumentati i dati disponibili in ogni azienda
  - Superate da tempo le capacità umane per analizzarli
- Questi dati sono prodotti in varie forme, strutturate (dati numerici, serie temporali, ...) e non (testi, immagini, ...)
- *"In 2006, the amount of digital information created, captured, and replicated was 1,288 x 10^{18} bits. In computer parlance, that's 161 exabytes or 161 billion gigabytes. This is about 3 million times the information in all the books ever written."*

  D. Reinsel et al., "The Expanding Digital Universe", IDC white paper (2007)
- *"We must harness the Internet's energy before the information it has unleashed buries us."*

  V. G. Cerf, "An Information Avalanche" (2007)

Cosa si può fare con tanti dati ed opportune applicazioni ?

- DEEP LEARNING: algoritmi moderni di machine learning avanzati
- Reti Neurali capaci di identificare persone dalle immagini con accuratezza del 92% (e.g. Facebook)
  - capaci di descrivere cosa contiene un'immagine, diagnosticare melanomi con maggiore accuratezza degli specialistici medici ...
- Deepmind, startup acquisita da Google nel 2014 per 500 M$
  - ha sviluppato nuove reti neurali capaci di apprendere autonomamente senza addestramento, e.g. robot che imparano autonomamente senza essere programmati o addestrati per uno specifico scopo, e.g. stare in piedi, correre, evitare ostacoli
- Google, Tesla e numerosi automotive vendor hanno sviluppato auto a guida autonoma
- Riconoscimento vocale e comprensione del linguaggio naturale in assistenti virtuali, SIRI, Android, Cortana, Amazon etc.

Esempi di Applicazioni Aziendali Strategiche Fondate sull'utilizzo dei Dati

- Le moderne applicazioni strategiche sono quelle che riescono a trasformare i dati in patrimonio informativo, alcuni esempi:
  - scoprire le **propensioni di acquisto** di ogni cliente, i.e. prevedere quali prodotti/servizi sono di maggior interesse per un utente
  - determinare la **soddisfazione della clientela** dall'elaborazione testuale delle recensioni, e.g. attraverso social network (twitter, facebook etc.)
  - fornire **full text search intelligenti** sul proprio catalogo
  - **prevedere le vendite** di prodotti/servizi
  - **ottimizzare le scorte di magazzino** affinchè siano minimali, ma sufficienti ad evitare rotture di stock (i.e. mancanza di prodotti)
  - **ottimizzare la logistica del magazzino** ricollocando i prodotti al proprio interno affinchè il tempo di prelievo sia ridotto al minimo
  - prevedere i **clienti che l'azienda rischia di perdere**
  - sviluppo di **sistemi embedded & IoT intelligenti ...**

Applicazioni Data Intensive & Obiettivo del Corso

- I problemi citati richiedono applicazioni orientate alla gestione ed elaborazione dei dati, i.e. DATA INTENSIVE
- **Obiettivo del corso**:
  - fornire conoscenze teoriche e pratiche per lo sviluppo di **applicazioni data intensive intelligenti** fondate sulla raccolta, organizzazione e **trasformazione dei dati in patrimonio strategico informativo**
- Scoprirete che pochi rudimenti matematici che avete già studiato sono sufficienti per realizzare queste applicazioni:
  - *algebra lineare, statistica, analisi matematica, ricerca operativa*
  - l'informatica fondata sulla matematica vi rende competitivi in quell'area del mercato in grande crescita nota come **DATA SCIENCE**
    - e.g. il mercato del lavoro, dalle piccole alle grandi aziendi, offre posizioni di **data scientist** che richiedono queste competenze, già dalla selezione per fare uno stage in Google, Amazon, Microsoft, IBM, Apple etc.

Programma del Corso: *Argomenti*

- introduzione al linguaggio di programmazione **Python** con cui svilupperemo applicazioni data intensive con interfaccia Web
  - il linguaggio più diffuso per applicazioni strategiche data intensive in ambito data science, intelligenza artificiale etc. (tutte le applicazioni citate in precedenza sono in Python), ma anche usato per Web App
- metodi per la raccolta e gestione dei dati
  - dati strutturati = dati descritti da un modello, e.g. modello relazionale
  - dati destrutturati = privi di modello, e.g. recensioni, immagine, video .. lavoreremo con dati testuali
- metodi per rappresentare e trasformare dati strutturati e destrutturati in patrimonio informativo strategico
  - vettori, matrici, aritmetica e scomposizione matriciale, regressione lineare e logistica, discesa sul gradiente, correlazioni statistiche, grafi
  - machine learning: un classificatore di base

Programma del Corso: *Applicazioni in Lab*

- Previsioni con regressione lineare
  - sviluppo di un'applicazione per predire il valore di immobili dalle vendite precedenti, sviluppo di un'applicazione in grado di stabilire dal testo di recensioni amazon l'orientamento positivo o negativo
- Recommendation di prodotti/servizi/news
  - sviluppo di un'applicazione che determini le recommendation di prodotti per ogni cliente con algebra lineare, con algoritmi di collaborative filtering e con SVD, recommendation impersonali e collocazione di prodotti in magazzino per minimizzare il pick-up time
- Previsioni con machine learning
  - sviluppo di un'applicazione che predica per ogni prodotto quali saranno le vendite, previsione degli approviggionamenti di magazzino
- Natural language processing
  - sviluppo di un'applicazione per determinare la customer satisfaction, con accesso a Twitter, in modo non supervisionato e supervisionato

Programma del Corso: *Tecnologie Python*

- Numpy
  - package for scientific computing, N-dimensional array objects, broadcasting functions, linear algebra, random number generators ...
- Scikit-learn
  - simple and efficient tools for machine learning and data science applications, built on NumPy, SciPy, and matplotlib
- NLTK & Whoosh
  - natural language toolkit for text processing and for full text search
- Surprise: recommendation toolkit
- Python DB API 2.0 & SQLAlchemy
  - DB API for the access to RDBMS; SQLAlchemy is the SQL toolkit and Object Relational Mapper and persistence in RDBMS
- Flask & Jinja (hints of django)
  - for the design & development of Web app in Python according the model view controller architectural pattern

Caso di studio

- Come caso di studio concreto per il corso è la realizzazione di un semplice **sito di e-commerce** e l'analisi dei dati raccolti dalle attività dei suoi utenti

Caso di studio: motivazioni

Un sito di e-commerce rappresenta un caso di studio esaustivo di applicazione data intensive

- I più grandi portali di e-commerce (es. Amazon) con cataloghi di milioni di prodotti necessitano di sistemi estremamente ottimizzati per la memorizzazione e la ricerca dei dati, in grado di gestire richieste di migliaia di utenti allo stesso tempo
- Gli ordini eseguiti dai clienti e i voti che danno ai prodotti costituiscono una grande mole di informazione strutturata, analizzabile in modo efficiente per fornire suggerimenti di acquisto, sia generali che personalizzati sui singoli clienti
- Dalle recensioni testuali dei clienti può essere stimato il loro grado di soddisfazione verso i singoli prodotti e si può valutare la reputazione dei rispettivi marchi

Caso di studio: requisiti

- Il sito presenta un catalogo di prodotti, organizzati in una gerarchia di categorie
- Gli utenti del sito devono essere in grado di autenticarsi con i loro rispettivi account
- Ogni utente può raccogliere prodotti in un carrello per poi compiere l'ordine di questi prodotti
- L'utente può consultare gli ordini fatti in passato
- Ogni prodotto è corredato da recensioni, compilate dagli utenti che lo hanno acquistato in passato
  - Ogni recensione corrisponde ad uno specifico acquisto del prodotto
  - Ogni recensione è costituita da un punteggio (1-5 stelle) e del testo

Caso di studio: dati di esempio

- Per ottenere un esempio realistico di applicazione, forniamo un database prepopolato con un set di dati di esempio estratti da Amazon.com (versione USA di Amazon)
- La quantità di dati è relativamente contenuta rispetto ai casi reali più *data intensive*, in modo da rendere più rapide le attività di laboratorio che prevedono l'analisi dell'intero set
  - 11.000 prodotti organizzati in decine di categorie
  - 11.000 utenti, esecutori di 100.000 ordini e autori di 500.000 recensioni
- Le tecnologie che presentiamo sono ad ogni modo scalabili, usate nella realtà anche per moli di dati molto più grandi

Caso di studio: dati analizzabili

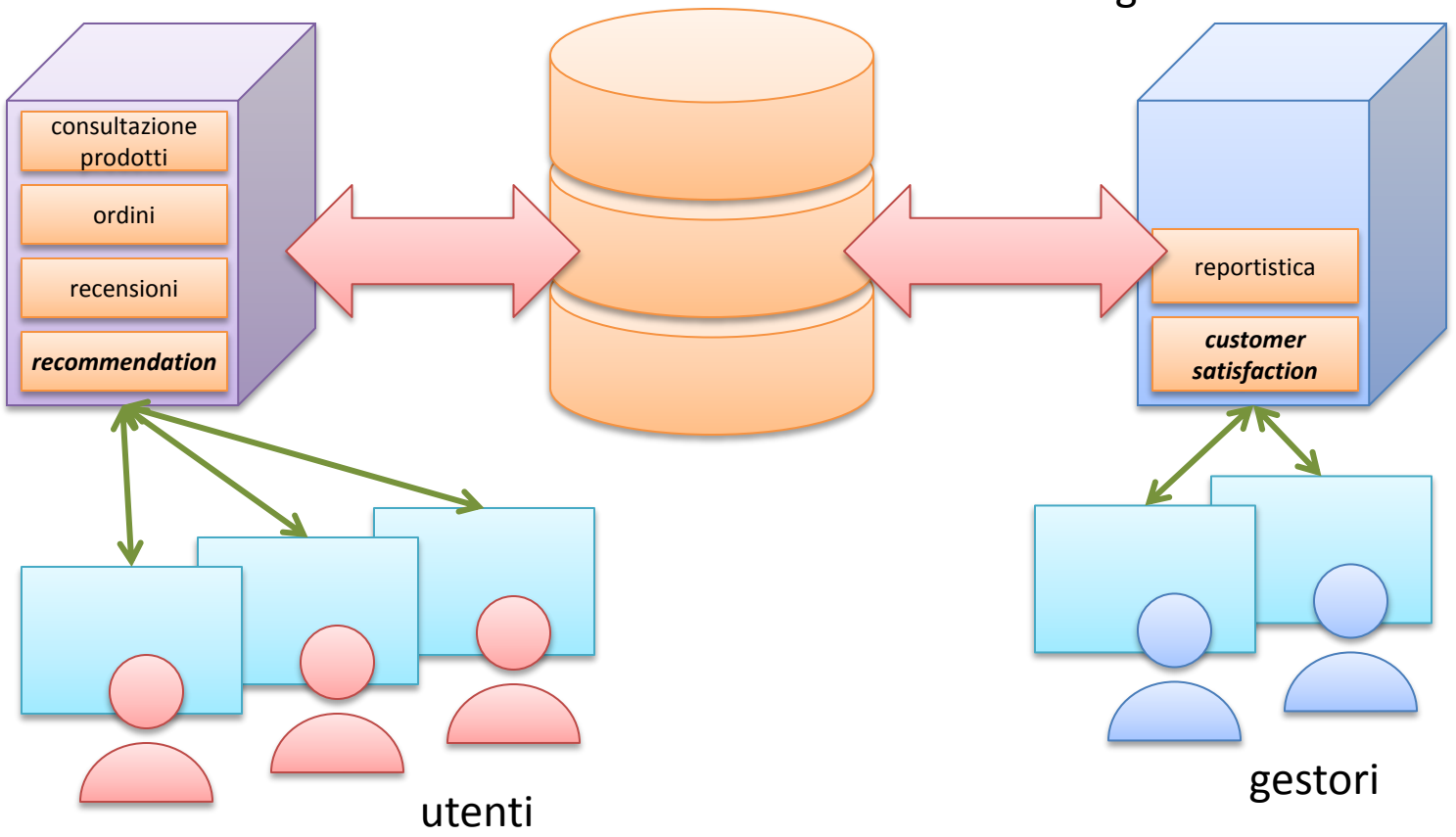
- Lo storico degli acquisti degli utenti e le loro recensioni costituiscono una mole potenzialmente enorme di dati
- Questi dati possono essere analizzati per estrarre informazioni utili e fornire un migliore servizio ai clienti e ai venditori
- Agli utenti, possono essere messi in evidenza i prodotti a cui potrebbero essere maggiormente interessati (***recommendation***)
- Dalle recensioni di ciascun prodotto (potenzialmente migliaia), è possibile estrarre informazioni sommarie sull'opinione generale degli acquirenti nei suoi confronti (***customer satisfaction***)

Caso di studio: componenti

Sito web: fornisce il servizio agli utenti

Base di dati

Strumenti di analisi: usati dai gestori del sito



Database Relazionali

- I database basati sul modello relazionale costituiscono l'approccio più comune alla memorizzazione di grandi masse di dati con frequenti operazioni di lettura e scrittura
- Uno dei RDBMS più usati è PostgreSQL: software libero aderente allo standard SQL e con funzionalità avanzate
- L'uso di un database relazionale richiede la definizione di uno schema dei dati
  - Dei vincoli devono essere definiti per garantire la correttezza dei dati
  - L'uso di opportuni indici è fondamentale per l'efficienza
- Python DB API 2.0 è l'interfaccia standard per l'accesso ai RDBMS
- Costruiremo da zero lo schema del database usato nell'applicazione, completo di vincoli e indici

Applicazioni Web

- I servizi fruibili via Web (social network, e-commerce ecc.) sono tra gli esempi più comuni di applicazioni data intensive
  - Usate da più utenti contemporaneamente, che consumano e producono dati in continuazione
  - dati molto preziosi per inferire profili utenti, pubblicità mirate, propensioni di acquisto, interessi, opinioni etc.
- Sempre più applicazioni Web sono basate su Python
  - in particolare applicazioni con forti requisiti di affidabilità ed efficienza
- Vedremo le tecnologie più diffuse per applicazioni Web in Python
  - Flask per la parte controller nella creazione dinamica di pagine Web
  - Jinja per la parte view per generare output e interfacce verso l'utente
- Queste saranno utilizzate per la costruzione dell'applicazione di e-commerce presentata come caso di studio

Object-Relational Impedance Mismatch

- L'uso di un database relazionale per la persistenza di dati di un'applicazione object-oriented comporta problemi dati dalla discrepanza tra i due paradigmi, detta *impedance mismatch*
- Nel paradigma object-oriented abbiamo oggetti identificati dalla loro posizione in memoria, spesso composti da altri oggetti (collegati tramite puntatori) accessibili tramite metodi
- Nel database relazionale, i dati corrispondenti ad un oggetto composto sono distribuiti su diverse tabelle, identificati e riferiti tramite chiavi: sono necessari join tra tabelle
- La conversione dei dati tra i due modelli richiede soluzioni non banali da progettare

Esempio: Ordine ad Oggetti e su Database Relazionale

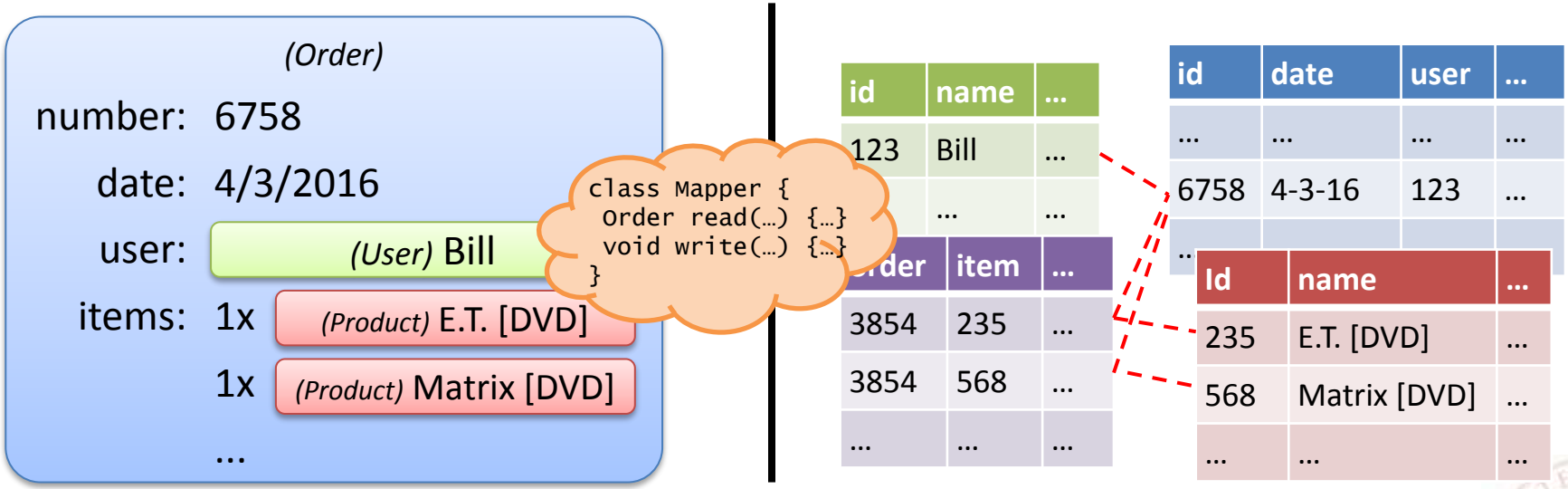
ad oggetti

Ordine #6758

Utente: Bill Data: 4/3/2016

prodotto	quantità	prezzo
E.T. [DVD]	1	€ 2,99
Matrix [DVD]	1	€ 4,99
…	…	…

relazionale



Object-Relational Mapping

- Con *object-relational mapping* ci si riferisce alle soluzioni a livello di progetto e implementazione adottate per risolvere l'impedance mismatch, per cui esistono diversi approcci
  - Devono permettere di leggere e scrivere oggetti persistenti sul DB, garantendo consistenza, accessi concorrenti tramite transazioni ecc.
- Le soluzioni basate su *Data Access Object* prevedono la creazione di uno strato dell'applicazione che incapsuli la logica di accesso al database e la separi dal resto dell'applicazione
- Vedremo in primo luogo come realizzare uno strato DAO per la nostra applicazione basato su psycopg (API pyhon postgres)
- Si vedrà che l'implementazione dei DAO richiede di scrivere grandi quantità di codice: non è l'approccio più conveniente

Framework per la Persistenza

- I framework di persistenza forniscono un'implementazione dei meccanismi tipici di object-relational mapping, riducendo notevolmente il carico di lavoro degli sviluppatori
- Una volta configurato correttamente in base alle esigenze e al modello dei dati della propria applicazione, un framework offre un API di alto livello per gestire oggetti persistenti su DB
- Come software di riferimento vedremo django e sqlalchemy, due framework open source ampiamente diffusi
  - Vedremo come rendere la parte di interazione col db ortogonale all'applicazione stessa, più rapido da implementare
  - Vedremo nel dettaglio il mapping tra classi e tabelle del DB e l'uso delle API per la gestione dei dati persistenti

Suggerimenti di Acquisto **Impersonali**: Regole Associative

- I dati raccolti da un'applicazione data intensive possono essere analizzati per estrarre informazioni di alto livello
- Nel caso del sito di e-commerce, possiamo analizzare i dati generati dall'attività degli utenti per suggerire prodotti che possono potenzialmente essere interessati ad acquistare
- Dall'analisi dei prodotti venduti nei singoli ordini, è possibile estrarre regole associative che indichino quali prodotti siano frequentemente venduti insieme ad altri
- Vedremo come realizzare un semplice algoritmo per individuare, per ciascun prodotto, quelli maggiormente correlati ad esso

Suggerimenti di Acquisto **Personali**: Recommendation di Prodotti

- Analizzando le recensioni date dagli utenti, i sistemi di recommendation possono fornire suggerimenti personalizzati per ciascun utente in base ai propri gusti
  - Diversi delle regole associative, che non considerano i singoli utenti
- RecDB è un'estensione integrata in PostgreSQL che consente di ottenere recommendation direttamente dal DB, tramite semplici query SQL con una clausola aggiuntiva
- Surprise è una libreria pyhton open source che fornisce diversi algoritmi di recommendation configurabili, con varie possibili fonti di dati e funzionalità per valutarne l'accuratezza
- Vedremo come integrare le recommendation nel nostro sito, mostrando suggerimenti precalcolati periodicamente

Full Text Search

- I dati testuali di un'applicazione (descrizioni, recensioni, …) sono *destrutturati*, non sono analizzabili direttamente come i dati strutturati (numeri, date, …) e devono essere invece convertiti in rappresentazioni appropriate
- I sistemi di *Full Text Search* consentono di indicizzare grandi moli di dati testuali ed effettuare in essi ricerche di singole parole, frasi, parole simili ecc.
- Whoosh è una libreria pyhton open source per costruire indici di documenti testuali ed effettuare ricerche su essi
  - Full Text Search & retrieval sui dati persistenti
  - ricerche per similarità semantica
  - Powerful query language, scoring algorithm, text analysis
- Integreremo Whoosh nella nostra applicazione per consentire la ricerca full text avanzata di prodotti e recensioni

Analisi della Customer Satisfaction

- Una volta strutturati, i dati testuali possono essere analizzati per estrarre conoscenza potenzialmente utile
- Dall'analisi delle recensioni scritte dai clienti, è possibile dedurre il loro grado di soddisfazione nei confronti dei prodotti acquistati, etichettandolo come positivo o negativo
- Un possibile approccio consiste nell'individuare parole note a priori che esprimono sentimenti positivi e negativi e valutare ogni recensione in base alla frequenza con cui appaiono
- Un approccio più avanzato è basato sul *machine learning*: un algoritmo di apprendimento analizza recensioni pre-etichettate ed estrae in automatico un modello di conoscenza utilizzabile per dedurre la polarità di altre recensioni

Organizzazione del Corso

Lezioni

- circa 50 ore tra lezioni in aula ed esercitazioni in laboratorio

Modalità di esame (6 CFU)

- Prova orale con:
  - Discussione di progetto di laboratorio di gruppo (1-3 studenti) concordato con il docente, su uno o più argomenti del corso, con un argomento per ogni partecipante (gruppo con 3 membri -> 3 argomenti).
  - Domande sul programma del corso

Materiale didattico

- testi suggeriti: Data Science con Python, Dmitry Zinoviev, Apogeo
  Data Science from Scratch, Joel Grus, O'Reilly Media
- Slide fornite dai docenti in aula e laboratorio
- Riferimenti bibliografici per approfondimenti opzionali (sito web del corso)
- Software e set di dati disponibili in laboratorio e scaricabili gratuitamente