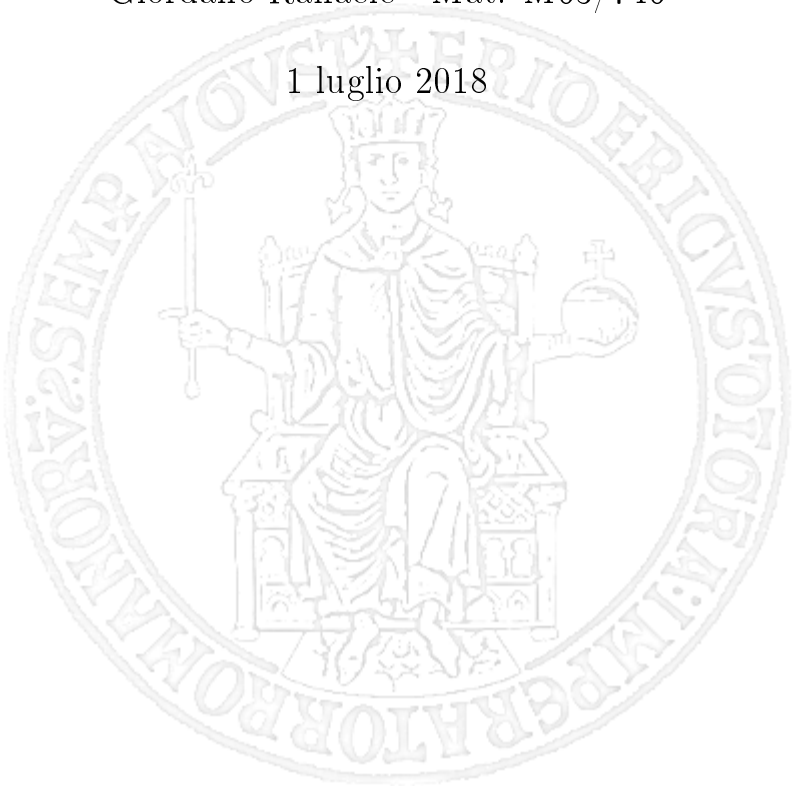
Big Data Analytics and Business Intelligence : Sentiment Analysis and Natural Language Processing

Galli Antonio - Mat. M63/0721
Gravina Michela - Mat. M63/708
Giordano Raffaele - Mat. M63/746

1 luglio 2018



Indice

Capitolo 1

Introduzione

1.1 Introduzione

Tale elaborato ha come obiettivo principale quello di effettuare la Sentiment Analysis e la Natural Language Processing delle review contenute nel database messo a disposizione da Yelp. In particolare, il progetto è stato realizzato secondo i seguenti principali:

- Definizione del problema: In tale capitolo sono spiegati, in grandi linee, lo scopo del progetto, le osservazioni principali che hanno caratterizzato le scelte di progetto e le analisi preliminari del contesto e dei dati a disposizione.
- Sentiment Analysis: In tale capitolo sono messi in evidenza gli strumenti che hanno permesso la realizzazione della sentiment Analysis.
- Architettura: In tale sezione è presentata l'architettura del sistema utilizzato.
- Implementazione: In tale capitolo viene riportata come l'analisi descritta nelle sezioni precedenti è stata implementata, riportando i relativi risultati.

1.2 Scopo del progetto

Lo scopo del progetto realizzato è quello di estrarre informazioni dai dati messi a disposizione al fine di dedurre il loro significato. I dati utilizzati sono stati messi a disposizione da YELP, il quale è un portale molto popolare che permette di pubblicare recesioni su varie attività di business. I dati sui quali abbiamo lavorato sono così composti:

- 5,2M di recensioni.
- 1,3M di utenti.
- 147K attività di business.

Nello specifico Yelp ha messo a disposizione 6 file in formato json:

1. user.json: che comprende la lista di utenti di yelp
2. tip.json: che comprende dei "suggerimenti" forniti dai vari utenti

3. review.json: che comprende le review espresse
4. photos.json: che comprende la lista delle foto messe a disposizione da Yelp
5. checkin.json: che tiene conto delle persone "servite" dai vari negozi nelle fasce orarie
6. business.json: che comprende la lista dei vari negozi

Yelp è una piattaforma user generated content in cui i contenuti sono prodotti dagli utenti che alimentano una comunità virtuale basata sul passaparola e sulla fiducia. Sfrutta quindi il marketing del passaparola (il cosidetto Wom) consentendo alle varie attività di business di espandersi e agli utenti di orientarsi più tranquillamente nella scelta dei negozi da "provare".

La nostra analisi, in particolare, si focalizza sulle review espresse dagli utenti con lo scopo di riucire ad estrapolare informazioni utili per la descrizione delle varie attività di business dal punto di vista della "reputazione e dell'apprezzamento popolare". Quindi la nostra analisi si concentra sull'opinione pubblica.

1.3 Analisi dei dati

In tale paragrafo viene riportata l'analisi dei dati messi a disposizione da Yelp. Non sono stati considerati tutti i file, ma solo quelli necessari ai fini del progetto. In particolare i file checkin.json e photos.json sono stati eliminati fin da subito in quanto non sono stati reputati utili. Il file tip.json, contenente i suggerimenti dell'utente, invece, è stato analizzato e si è visto che tali suggerimenti non sono stati considerati dagli altri user e quindi il suo valore è considerato trascurabile nell'analisi dell'opinione pubblica.

1.3.1 File user.json

Tale file contiene la descrizione degli utenti. Un un utente ha la seguente struttura:

```
{
    "user_id":"encrypted user id",
    "name":"first name",
    "review_count":number of reviews,
    "yelping_since": date formatted like "2009-12-19",
    "friends":["an array of encrypted ids of friends"],
    "useful":"number of useful votes sent by the user",
    "funny":"number of funny votes sent by the user",
    "cool":"number of cool votes sent by the user",
    "fans":"number of fans the user has",
    "elite":["an array of years the user was elite"],
    "average_stars":floating point average like 4.31,
    "compliment_hot":number of hot compliments received by the user,
    "compliment_more":number of more compliments received by the user,
    "compliment_profile": number of profile compliments received by the user,
    "compliment_cute": number of cute compliments received by the user,
    "compliment_list": number of list compliments received by the user,
    "compliment_note": number of note compliments received by the user,
    "compliment_plain": number of plain compliments received by the user,
    "compliment_cool": number of cool compliments received by the user,
    "compliment_funny": number of funny compliments received by the user,
    "compliment_writer": number of writer compliments received by the user,
    "compliment_photos": number of photo compliments received by the user,
    "type":"user"
}
```

Un utente, quindi, oltre ai dati personali, presenta delle caratteristiche che esprimono la sua popolarità, come ad esempio i complimenti ricevuti, il numero di amici e fans. Una delle caratteristiche fondamentali di un "buon utente" è l'attributo elite. Lo status di elite si conquista attraverso recensioni ben scritte, foto di qualità, assiduità nell'inviare complimenti verso gli altri Yelper. Per far parte della squadra Elite si deve utilizzare il vero nome, completare il proprio profilo con informazioni dettagliate e postare una foto che identifichi chiaramente l'identità.

1.3.2 File review.json

Tale file contiene le varie recensioni. Una recensione ha la seguente struttura:

```
{
    "review_id":"encrypted review id",
    "user_id":"encrypted user id",
    "business_id":"encrypted business id",
    "stars":star rating, rounded to half-stars,
    "date":"date formatted like 2009-12-19",
    "text":"review text",
    "useful":number of useful votes received,
    "funny":number of funny votes received,
    "cool": number of cool review votes received,
    "type": "review"
}
```

1.3.3 File business.json

Tale file comprende le varie attività di business. Una attività di business ha la seguente stuttura:

```
{
    "business_id":"encrypted business id",
    "name":"business name",
    "neighborhood":"hood name",
    "address":"full address",
    "city":"city",
    "state":"state -- if applicable --",
    "postal code":"postal code",
    "latitude":latitude,
    "longitude":longitude,
    "stars":star rating, rounded to half-stars,
    "review_count":number of reviews,
    "is_open":0/1 (closed/open),
    "attributes":["an array of strings: each array element is an attribute"],
    "categories":["an array of strings of business categories"],
    "hours":["an array of strings of business hours"],
    "type": "business"
}
```

1.4 Analisi del contesto

Al fine di entrare in familiarità con l'analisi effettuata è necessario specificare alcuni concetti che sono indispensabili nella comprensione del progetto:

- Che cosa è una review? Cosa significa analizzarla?
- Quali sono le caratteristiche rilevanti di un utente?

- Cosa si intende per "opinione pubblica"?

1.4.1 Concetto di Review

Una review è un'opinione soggettiva espressa da un utente circa un'attività di business al fine di mettere in risalto le sue caratteristiche principali in termini sia positivi che negativi. In particolare una recensione è rivolta a potenziali clienti e quindi la sua funzione deve essere quella di aiutarli a decidere se scegliere o meno un determinato negozio, fornando indicazioni sulla sua qualità e sul suo contenuto.

Quindi una recensione influenza inevitabilmente il pensiero di un utente che la legge e quindi ha un impatto su quella che viene considerata "opinione pubblica".

1.4.2 Concetto di utente

In questo contesto, un utente è una entità che ha la facoltà di recensire una attività di business oppure esprimere il pensiero sulle recensioni espresse da altre utenti, confermandole o contraddicendole.

Un utente di Yelp ha una propria scheda dove sono riportati i propri dati personali e una serie di attributi che permettono la caratterizzazione di un "buon utente". In effetti consideriamo buono, un utente con le seguenti caratteristiche:

- è un utente elite: quindi la sua contribuzione al sito è stata riconosciuta dal portale
- ha un numero elevato di fans e quindi ciò implica che è una persona popolare
- ha scritto un numero elevato di recensioni e ciò implica che l'utente è molto attivo nella community di Yelp
- ha molto amici
- ha ricevuto molti riconoscimenti (useful, cool, funny) e complimenti da parte di altri user.

In definitiva consideriamo buono un utente la cui "parola" è capace di influenzare l'opinione pubblica, sia in positivo che in negativo. Ciò signifca che una recensione espressa da un utente "buono" merita un peso maggiore in quanto essa risulta essere in un certo senso "garantita" . In pratica un utente è buono se di lui ci si può fidare.

1.4.3 Concetto di opinione pubblica

Si definisce opinione pubblica il giudizio o modo di pensare collettivo della maggioranza dei clienti di una attività di business. Lo studio dell'opinione pubblica risulta molto importante ai fini di un business in quanto vale il seguente concetto:

il pensiero di un utente in relazione ad una attività di business, della quale non sa nulla (perchè non vi è mai stato), è fortemente influenzato dall'opimione pubblica. Praticamente l'opinione pubblica diventa il pensiero di base che l'utente ha senza la conoscenza diretta del nagozio.

Quindi negozi di cui si parla bene, sono facilitati nell'attirare i clienti, mentre, al contrario, è molto difficile che un cliente si rechi in un posto di cui si "parla male".

In effetti il nostro scopo è proprio quello di studiare l'opinione pubblica di una attività di business, cercando di quantificarla nel miglior modo possibile, prendendo in considerazione tutti i possibili fattori di influenza.

Alla base di un'opinione pubblica vi sono le recensioni scritte dai vari utenti; tuttavia l'opinione pubblica dipende non solo da ciò che è stato scritto, ma anche da "chi" lo ha scritto e da quanto successo ha riscosso. Pertanto una stima preliminare dell'opinione pubblica è data dalla seguente relazione:

$Vopinione = ContributoRecensione * ContributoUtente * SuccessoRecensione$

Analizziamo quindi i tre fattori:

- ContributoRecensione: è il contributo espresso dalla recensione costitutito da due elementi:
  - Stelline: è una metrica di valutazione (da 1 a 5) da parte di un utente
  - Testo: testo in linguaggio naturale espresso dall'utente. Tale elemento, processato opportunamente, quantifica la positività o negatività del pensiero dell'utente.
- ContributoUtente: è il peso associato a chi ha scritto la recensione. In effetti ogni pensiero viene sempre pesato in base al suo autore, per questo motivo una recensione scritta da un utente "buono", come precedentemente spiegato, ha un impatto maggiore sull'opinione pubblica.
- SuccessoRecensione: è il peso associato a quanto popolare è stata quella recensione. In effetti una recensione che ha avuto molto successo in quanto è stata letta e presa in considerazione da molti utenti, ha inevitabilmente un impatto maggiore sull'opinione pubblica.

Grazie alla relazione scritta, siamo in grado di quantificare l'opinione pubblica di una recenzione scritta da un cliente dell'attività di business, in quanto abbiamo proprio pesato la recensione in base a chi l'ha scritta e al suo successo.

Ma cosa accade se un utente che non è e non è mai stato cliente dell'attività recensita legge le varie review?

Ovviamente tale utente verrà influenzato da ciò che gli altri hanno detto. Quest'ultimo leggerà le varie review, ma come è possibile intuire, quelle che avranno una maggiore considerazione sono le recensioni più recenti.

Capitolo 2

Sentiment Analysis & Natural Language Processing

Lo scopo del progetto consiste nella quantificazione dell'opinione pubblica e quindi alla base vi è l'analisi delle review scritte dai vari utenti. Pertanto sono necessari degli strumenti che permettono di effettuare la processazione del linguaggio naturale e la sentiment analysis. Gli strumenti da noi utilizzati sono:

- Libreria Natural Language Toolkit (Vader) di Python: tale libreria è stata utilizzata per effettuare la sentiment analysis delle review
- Libreria TextBlob di Python: tale libreria è stata utilizzata per effettuare al processazione del linguaggio naturale.

2.1 Natural Language Toolkit (Vader)

La sentiment analysis è uno studio computazionale delle opinioni, dei pensieri delle persone che possono essere espressi in testi oppure attraverso le immagini. Consiste nell'elaborazione del linguaggio naturale, nell'analisi testuale e linguistica per identificare ed estrarre informazioni soggettive da diverse fonti.

Vader è un package di python che si installa con la libreria NLTK. Vader è un esempio di metodo lessicale per la sentiment analysis: gli approcci lessicali calcolano il sentiment score di una frase considerando lo score di ogni parola all'interno della frase stessa.

Vader si basa su un dizionario che effettua un mapping tra la parola e il suo significato emotivo: ad ogni parola viene associato un valore di intensità tra -4 e 4. Il sentiment score di un testo viene quindi effettuato con la somma delle intensità delle parole nel testo. Il risultato, tuttavia, è un valore compreso tra -1 e 1 in quanto viene effettuata una opportuna normalizzazione. In particolare, detta x la somma dei valori delle parole e α un valore di normalizzazione, la funziona utilizzata per effettuare la normalizzazione del risultato è la seguente:

$\frac{x}{\sqrt{x^2+\alpha}}$

Al crescere di x, il valore di normalizzazione di sposta verso 1 (o -1).

Una volta analizzato il testo, vader produce in uscita quuattro valori:

- positive: porzione del testo che cade nella categoria positiva

- negative:porzione del testo che cade nella categoria negativa
- neutral: porzione del testo che cade nella categoria neutra
- compound: somma dei valori delle varie parole (x), normalizzata tra -1 e 1

Per effettuare la sentiment Analysis vader si basa su alcune semplici euristiche:

- La prima euristica è la punteggiatura: "I like it" e "I like it!!!!" sono diversi. Quindi bisogna aumentare il valore delle frasi che hanno i punti esclamativi
- La seconda euristica riguarda la scrittura in maiuscolo: "AMAZING" e "amazing" sono diverse. Il valore (in positivo o negativo) delle parole in maiuscolo deve essere amplificato
- La terza euristica riguarda i modificatori di grado: ad esempio "sort of cute". Vader ha un dizionari proprio per gestire le parole che ammortizzano l'intensità.
- La quarta euristica riguarda l'uso della congiunzione "but", in quanto questa collega frasi con "sentimenti" contrastanti. Vader implementa proprio un "but checkers"

2.2 TextBlob

TextBlob è una libreria Python utlizzata per l'elaborazione di dati testuali. Essa fornisce una semplice API per la gestione di attività comuni di elaborazione del linguaggio naturale (NLP) come la generazione di tag di parte del discorso, estrazione di frasi e nomi, analisi del sentiment, classificazione, traduzione e altro. Di seguito è riportato un elenco delle features che caratterizzano Text Blob:

- Noun phrase extraction
- Part-of-speech tagging
- Sentiment analysis Classification (Naive Bayes, Decision Tree)
- Language translation and detection powered by Google Translate
- Tokenization (splitting text into words and sentences)
- Word and phrase frequencies
- Parsing
- n-grams
- Word inflection (pluralization and singularization) and lemmatization
- Spelling correction

In particolare nel nostro progetto è stata sfruttata la potenzialità di questa libreria per stimare la frenqueza delle parole all'interno delle review relative ad una determinata attività di business. La stima è stata effettuata sfruttando quello che è il parsing di stringhe in oggetti di tipo TextBlob. In effetti il parsing è servito per sfruttare due funzioni fondamentali della classe Blob ovvero:

- tags: è una funzione che prende in ingresso una stringa e ritorna una lista di tuple nella forma (word, POS tag). Grazie proprio all'associazione del tag alla parola è possibile filtrare le parole non utili ai fini dell'analisi.
- words: è una funzione che prende in ingresso una stringa e ritorna una lista di word tokens, escludendo i caratteri di punteggiatura. In questo modo si può sfruttare la funzione ".count(word)" per contare le occorrenze di una determinata parola in un dato testo.

Capitolo 3

Architettura

In tale capitolo è riportata l'architettura con il relativo stack tecnologico utilizzato ai fini della realizzazione del progetto. In paricolare sono stati utilizzati i seguenti componenti:

- Microsoft Azure: è una piattaforma cloud pubblica di Microsoft, che offre servizi di cloud computing.
- MongoDB: è un DBMS non relazionale (NoSQL), orientato ai documenti.
- Apache Spark: è un framework open source per il calcolo distribuito.

3.1 Microsoft Azure

Come detto precedentemente Azure è una piattaforma cloud pubblica di Microsoft, che offre servizi di cloud computing. Tramite essa vengono erogati servizi appartenenti a diverse categorie quali: risorse di elaborazione, archiviazione e memorizzazione dati, trasmissione dati e interconnessione di reti, analisi, intelligence, monitoraggio e gestione, nonchè servizi per lo sviluppo di applicazioni. In particolare i servizi messi a disposizione possono essere classificati in tre specifiche aree, a seconda della modalità di erogazione adottata:

- Infrastructure-as-a-Service (IaaS)
- Platform-as-a-Service (PaaS)
- Software-as-a-Service (SaaS)

Ciascun servizio erogato prevede un pagamento in base al consumo e la metodologia con cui ne viene determinato il costo è specifica per il servizio stesso. Ai fini del nostro progetto si è sfruttata tale piattaforma per usufruire di una macchina virtuale con le seguenti caratteristiche:

- Sistema operativo: Linux Ubuntu 17.04
- 8 core , 16GB di memoria
- Processore Intel Xeon E5

3.2 MongoDB

MongoDB è un database NoSQL orientato ai documenti, che nasce nel 2007 in California come servizio da utilizzare nell'ambito di un progetto più ampio, ma che presto è diventato un prodotto indipendente ed open-source. Esso memorizza i documenti in JSON, formato basato su JavaScript e più semplice di XML, ma comunque dotato di una buona espressività. Esso quindi si allontana dalla struttura tradizionale basata su tabelle dei database relazionali in favore di documenti in stile JSON con schema dinamico, rendendo l'integrazione di dati di alcuni tipi di applicazioni più facile e veloce. Come detto precedentemente, mongoDB è un database orientato ai documenti, ma cosa è un documento? Un docuemnto è fondamentalmente un albero che può contenere molti dati, anche annidati. I documenti sono raggruppati in collezioni che possono essere anche eterogenee. Ciò significa che non c'è uno schema fisso per i documenti. Tra le collezioni non ci sono relazioni o legami garantiti da MongoDB: in altre parole, non esiste un concetto analogo al vincolo di integrità referenziale.

Modello logico a parte, le caratteristiche chiave di MongoDB sono:

- Query ad hoc:
  - MongoDB supporta ricerche per campi, intervalli e regular expression. Le query possono restituire campi specifici del documento e anche includere funzioni definite dall'utente in JavaScript.
- Indicizzazione:
  - qualunque campo in MongoDB può essere indicizzato (gli indici in MongoDB sono concettualmente similari a quelli dei tradizionali RDBMS). Sono disponibili anche indici secondari, indici unici, indici sparsi, indici geospaziali e indici full text.
- Alta Affidabilità:
  - MongoDB fornisce alta disponibilità e aumento del carico gestito attraverso i replica set. Un replica set consiste in due o più copie dei dati. Ogni replica può avere il ruolo di copia primaria o secondaria in qualunque momento. La replica primaria effettua tutte le scritture e le letture. Le repliche secondarie mantengono una copia dei dati della replica primaria attraverso un meccanismo di replicazione incluso nel prodotto. Quando una replica primaria fallisce, il replica set inizia automaticamente un processo di elezione per determinare quale replica secondaria deve diventare primaria. Le copie secondarie possono anche effettuare letture, con dati eventualmente consistenti di default.
- Sharding e bilanciamento dei dati:
  - MongoDB scala orizzontalmente usando lo sharding. L'utente deve scegliere una chiave di sharding, che determina come i dati di una collection saranno distribuiti tra i vari nodi. I dati sono divisi in intervalli (basati sulla chiave di shard) e distribuiti su molteplici shard (uno shard è un replica set, quindi con una replica primaria e due o più repliche secondarie).
  - MongoDB include un meccanismo di bilanciamento dei dati, spostando gli intervalli di dati da uno shard troppo carico ad uno shard meno carico, in modo da bilanciare la distribuzione dei dati all'interno del cluster.

- File Storage:
  - MongoDB può essere usato anche come un file system, traendo vantaggio dalla caratteristiche di replicazione e di bilanciamento su più server per memorizzare file, anche di grandi dimensioni.
  - Questa funzione, chiamata GridFS, è inclusa nei drivers di MongoDB e disponibile facilmente per tantissimi linguaggi di sviluppo. MongoDB espone delle funzioni per la manipolazione dei file. GridFS è usato, ad esempio, nei plugin di NGINX e lighttpd. Invece di memorizzare il file in un singolo documento, GridFS divide il file in tante parti più piccole, chiamate chunks, e memorizza ognuno di questi chunk in un documento separato.
  - Ai file possono essere associati dei metadati, su cui è possibile anche creare degli indici full-text.
- Aggregazione:
  - MongoDB supporta due modalità di aggregazione dei dati: il MapReduce e l'Aggregration Framework. Quest'ultimo lavora come una pipeline e permette di ottenere risultati molto più rapidamente del MapReduce grazie all'implementazione in C++.
- Capped Collection:
  - MongoDB supporta collection a dimensioni fisse chiamate capped collection. Questo tipo di collection mantengono l'ordine di inserimento e una volta raggiunta la dimensione definita, si comportano come code circolari.

Di seguito viene mostrata l'architettura relativa a mongoDB:

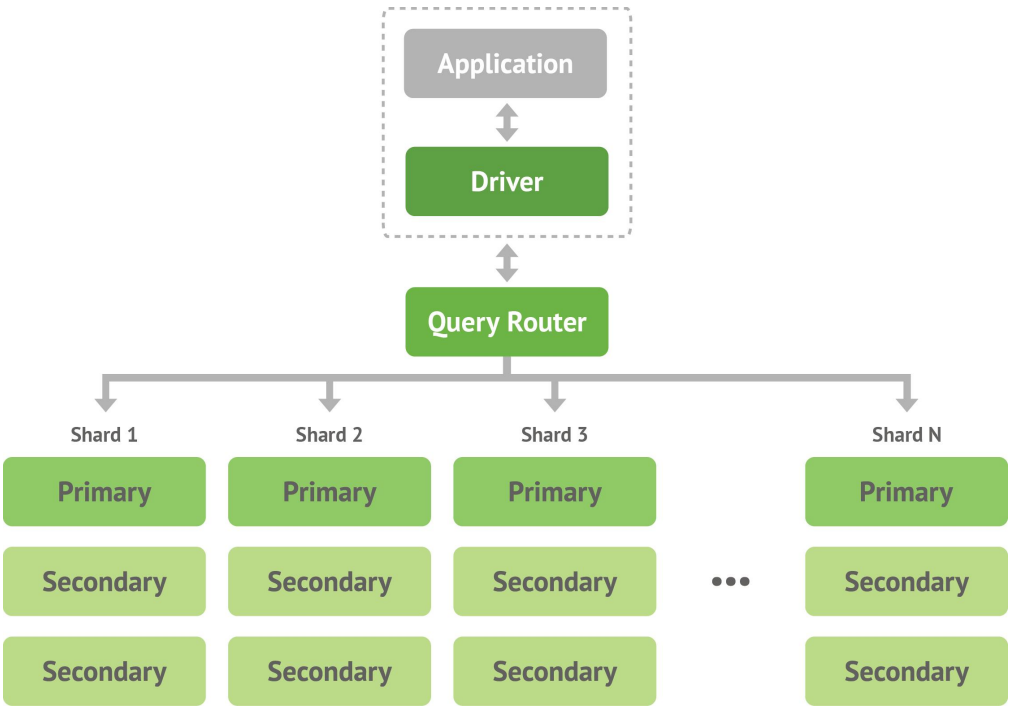


Figura 3.1: MongoDB Architecture

3.3 Spark

Apache Spark è un framework open source per il calcolo distribuito sviluppato dall'AMPlab della Università della California e successivamente donato alla Apache Software Foundation. A differenza del paradigma MapReduce, basato sul disco a due livelli di Hadoop, le primitive "in-memory" multilivello di Spark forniscono prestazioni fino a 100 volte migliori per talune applicazioni. Ciò permette ai programmi utente di caricare dati in un gruppo di memorie e interrogarlo ripetutamente. Spark richiede un gestore di cluster e un sistema di archiviazione distribuita. Per il primo supporta nativamente un cluster Spark ma anche Hadoop YARN, per il secondo Spark può interfacciarsi con Hadoop Distributed File System (HDFS), Apache Cassandra, Amazon S3, ma anche soluzioni personalizzabili. Spark supporta anche soluzioni pseudo-distribuite in modalità locale, usate di solito per lo sviluppo o scopo di test, dove l'archiviazione distribuita non è richiesta e si usa il file system locale; in tale scenario, Spark è eseguito su una macchina singola. L'architettura di Spark si compone di un master scheduler e di tanti worker che eseguono il calcolo distribuito, quindi secondo quello che è lo standard architetturare dei sistemi distribuiti. Ovviamente gli sviluppatori di Spark fanno notare nel loro paper che Spark è stato pensato per dare il meglio nelle elaborazioni batch, mentre non funziona in maniera performante in sistemi distribuiti che richiedono frequenti update granulari sui set di dati archiviati. In quel caso, meglio evitare di usare Spark o Hadoop e invece usare un normale sistema di database o un sistema di distributed shared memory, che sono invece pensati per questo. Di seguito è riportata l'architettura relativa al framework Apache Spark:

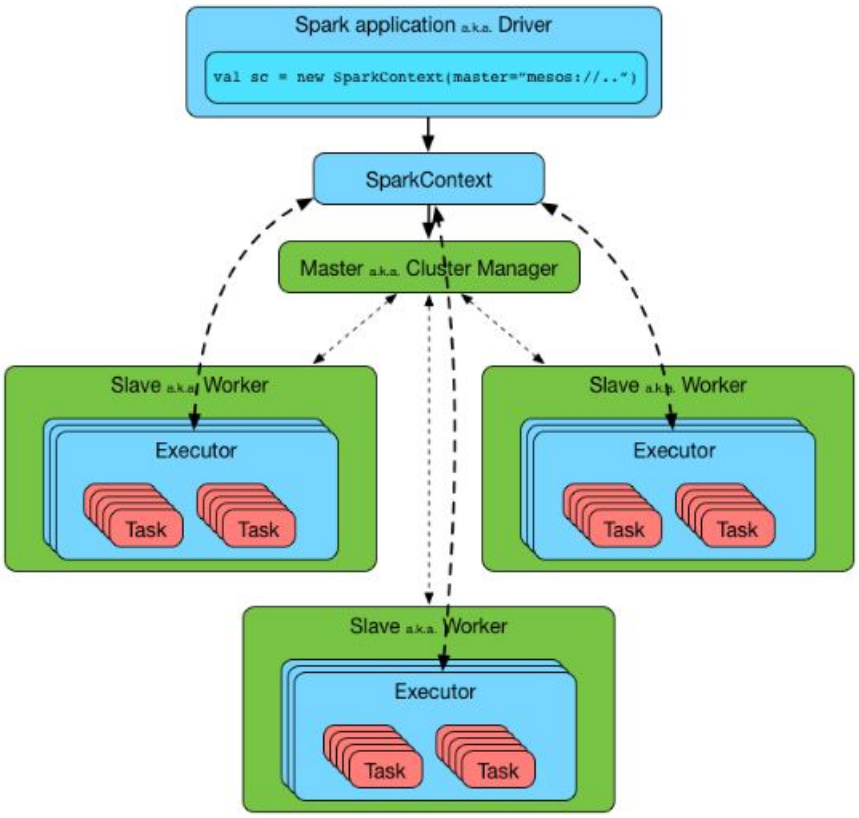


Figura 3.2: Spark Architecture

Capitolo 4

Realizzazione

In tale capitolo viene riportata la realizzazione del progetto, in particolare si sono implementati i seguenti script:

- sparkWF.py: tale script conta la frequenza delle parole contenute nelle review relative ad un dato business, mettendo in evidenza le parole più frequenti con il proprio grado di "positività".
- valoreUtente.py: tale script ha il compito di calcolare il valore di importanza attribuito ad un utente.
- valoreReview.py: tale script ha il compito di calcolare il grado di "positività" del testo di una review.
- tuning.py: tale script ha il compito di effettuare il tuning dei parametri della funzione che stima il valore di opinione pubblica.
- valoreOpinione.py: tale script ha il compito di stimare il valore di opinione pubblica.
- generaCsv.py: tale script è stato realizzato per generare i csv utili ai fini dell'analisi.
- valoreCumulativo.py: tale script ha il compito di stimare il valore cumulativo di opinione pubblica.

In particolare gli script sono presentati di seguito nell'ordine in cui sono stato realizzati ed eseguiti, in quanto tale ordine rappresenta il processo di analisi condotto per la realizzazione del progetto.

4.1 valoreUtente.py

Tale script è stato realizzato per poter calcolare il valore di ogni utente. Nel contesto di tale progetto, un utente viene considerato "buono", e quindi con un valore alto, se è "attivo" nella community e pertanto presenta le seguenti caratteristiche:

1. E' un utente Elite: ciò significa che la community ha premiato il suo modo di operare.
2. E' un utente con un numero elevato di fans.
3. E' un utente con un numero elevato di amici.

4. E' un utente con un numero elevato di review scritte.

5. E' un utente che è stato per diversi anni eletto Elite.

6. E' un utente che ha ricevuto molti compliementi da altri.

Prendendo in considerazione tali fattori, il valore dell'utente è stato calcolato in forma chiusa con una relazione che stima il valore di un utente in base alle caratteristiche sopra elencate. In particolare ad un utente che non possiede tali caratteristiche viene associato un valore molto basso. Tale ragionamento è stato effettuato al fine di poter riconoscere degli utenti fake, i quali effettuano delle recensioni non veritiere. In effetti a nostro avviso un utente ha una elevata probabilità di essere fake se ha le seguenti caratteristiche:

1. Ha poche recensioni, con stelle elevate e magari tutte relative ad un'unica attività di business.

2. Non ha amici , ne fans.

3. Non è mai stato eletto Elite.

Di seguito è riportato il listato relativo a tale script:

```
#!/usr/bin/python

import pymongo
from pymongo import MongoClient
import pprint
import json

#Connessione a MongoDB
client = MongoClient('localhost',27017)
db = client.YelpDB

#Caricamento utenti
users = db.user.find()

#Calcolo valore utente
jsonlist = []
for element in users:
  subel = {}
  subel['user_id']= element['user_id']    #id utente
  numAnniElite = len(element['elite'])
  numAmici = len(element['friends'])
  sCompl = element['compliment_hot'] + element['compliment_more'] +
      element['compliment_profile'] + element['compliment_cute'] +
      element['compliment_list'] + element['compliment_note'] +
      element['compliment_plain'] + element['compliment_cool'] +
      element['compliment_funny'] + element['compliment_writer'] +
      element['compliment_photos']
  if '2017' in element['elite']:
```

```
    Elite17 = 1
  else:
    Elite17 = 0
  subel['valore'] = 0.27*Elite17+ 0.05*numAnniElite+0.13*element['
     fans']+0.06*element['review_count']+0.1*numAmici+0.1*sCompl
     +0.19*element['useful']+0.03*element['funny']+0.05*element['cool
     ']
  jsonlist.append(subel)
with open('sintesiuser.json','w') as outfile:
    json.dump(jsonlist,outfile)
collection = db.sintesiuser
clear = collection.remove()
result = collection.insert_many(jsonlist)
client.close()
```

Inoltre di seguito è riportato il listato di tale programma implementato in pyspark:

```
#!/usr/bin/python

import pymongo
from pymongo import MongoClient
import pprint
import json
from pyspark import SparkConf, SparkContext
import string
import csv
import re

conf = SparkConf()
conf.setMaster("local[*]")
conf.setAppName("val utente")
conf.set("spark.executor.memory", "16g")
sc = SparkContext(conf = conf)

#Connessione a MongoDB
client = MongoClient('localhost',27017)
db = client.YelpDB

#Caricamento utenti
users= db.user.find().limit(50000)

#Calcolo valore utente
jsonlist = []
def worker(element):
  numAnniElite = len(element['elite'])
  numAmici = len(element['friends'])
```

```
30    sCompl = element['compliment_hot'] + element['compliment_more'] +
          element['compliment_profile'] + element['compliment_cute'] +
          element['compliment_list'] + element['compliment_note'] +
          element['compliment_plain'] + element['compliment_cool'] +
          element['compliment_funny'] + element['compliment_writer'] +
          element['compliment_photos']
31    if '2017' in element['elite']:
32      Elite17 = 1
33    else:
34      Elite17 = 0
35    valore = 0.27*Elite17+ 0.05*numAnniElite+0.13*element['fans']+0.06*
          element['review_count']+0.1*numAmici+0.1*sCompl+0.19*element['
          useful']+0.03*element['funny']+0.05*element['cool']
36          subel=[element['user_id'],numAnniElite, numAmici, sCompl,
                element['fans'],element['review_count'],element['useful'],
                 element['funny'], element['cool'],valore]
37          return subel
38  # jsonlist.append(subel)
39  user = sc.parallelize(users)
40  results = user.map(worker)
41  print ' ****************************** RESULT
        ************************'
42  csvf= [["user_id","numAnniElite","numAmici","sCompl","fans","
        review_count","useful","funny","cool","valore"]]
43  csvf= csvf+ results.take(results.count())
44
45  fileu=open("AnalisiUt.csv",'w') with fileu:
46      writer = csv.writer(fileu, delimiter=",", quotechar="|", quoting=
            csv.QUOTE_MINIMAL)
47      writer.writerows(csvf)
48
49  client.close()
```

4.2 valoreReview.py

Tale script è stato realizzato per poter calcolare il grado di positività del testo contenuto in una determinata review. In particolare si è fatto uso della libreria NLTK di Python, al fine di analizzare il testo. Il punteggio ritornato viene calcolato effettuando la media dei punteggi attribuiti alle varie frasi. Successivamente al punteggio attribuito dalla libreria viene sommato il contributo relativo alle stelle associate alla review da parte dell'utente. In effetti le stelle sono state considerate parte integrante dell'opinione dell'autore e queste sono state normalizzate al fine di ricadere nel range [-1,1]. In particolare è stata fatta la seguente normalizzazione:

Stelle	Valore normalizzato
5	1
4	0,5
3	0
2	-0,5
1	-1

L'output di tale script viene memorizzato in una collection del database, la quale è costituita da documenti ognuno dei quali rappresenta la forma sintetica di una review, in quanto non compare più il testo, ma il relativo punteggio e all'id dell'utente viene associato anche il punteggio relativo a quest'ultimo. Tali informazioni sono necessarie in quanto verranno utilizzate solo dopo aver trovato i parametri migliori della funzione di opnione pubblica. Di seguito è riportato il listato relativo a tale script:

```
#!/usr/bin/python

#Tale script serve per determinare il valore di opinione per le varie review.
#In particolare, sono prese in considerazione solo le review relative ad un business che ha un numero di review maggiore di 500.

from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk import tokenize
import nltk
nltk.download('vader_lexicon')
nltk.download('punkt')
import pymongo
from pymongo import MongoClient
import pprint
import json
################################

#Connessione a MongoDB
client=MongoClient('localhost',27017)
db=client.YelpDB

#Seleziona le review relative ai business con review_count maggiore di 500
print "--- Lettura negozi"
curnegozi = db.business.find({"review_count": {"$gte": 500 }})
#tutti i negozi con review_cout magg o uguale a 500
negozi=[]
for element in curnegozi:
    negozi.append(element['business_id'])
#usa la lista per ricavare le review relative ai negozi
print "--- Lettura review"
reviews= db.review.find({"business_id":{"$in":negozi}})
```

```
31 #lista degli utenti che hanno scritto le review relative ai business
32 print "--- Lista utenti"
33 listaid=db.review.distinct("user_id", {"business_id":{"$in":negozi}})
34 #prendo gli utenti con l'id ritornato sopra.
35 utenti= db.sintesiuser.find({"user_id": {"$in": listaid}})
36 #dobbiamo scorrere la lista delle review per effettuare il calcolo
      dei valori
37 #massimo degli utenti in sintesiuser
38 utm=db.sintesiuser.find().sort("valore",-1).limit(1)
39 massimo=utm.next()['valore']
40 #minimo
41 utm=db.sintesiuser.find().sort("valore",1).limit(1)
42 minimo=utm.next()['valore']
43 utentidict=[]
44 print "--- Creazione dizionario degli utenti"
45 for element in utenti:
46     utentidict.append([element['user_id'],element['valore']])
47 utentidict=dict(utentidict)
48 print "--- Analisi review e calcolo valore"
49 #Quindi abbiamo selezionato le review relative a tutti i negozi con
50  #review count > 500 e inoltre abbiamo preso tutti gli utenti
51  #che hanno scritto quelle review
52 jsonlist= []
53 sid = SentimentIntensityAnalyzer()
54 indice=0
55 for element in reviews:
56     print indice
57      #per tener conto delle elaborazioni
58     indice=indice+1
59     subel = {}
60     subel['review_id']= element['review_id']          #reviewid
61     subel['business_id']= element['business_id']      #id negozio
62     subel['user_id']= element['user_id']              #id utente
63     subel['date'] = element['date']                   #data
64     subel['stars']= element['stars']                  #stelle per
          elaborazione
65     #adesso dobbiamo calcolare il valore, dato dalla seguente
          espressione
66       #(lib+sterlinenormalizzato)*(1 + useful0.7+funny0.1+cool0.2 )
            *(1+ valoreUtente)
67     lines_list = tokenize.sent_tokenize(element['text'])
68     #per ogni frase viene calcolata lo score di polarita
69     #I vari score si sommano tra loro
70     #Infine viene effettuata una media
71         app=0
72     for sentence in lines_list:
```

```
        ss = sid.polarity_scores(sentence)
        app=ss['compound']+app
    lib = app/len(lines_list)  #media
    #normalizzazion stelle:
    # 1 stella --> -1
    # 2 stelle --> -0.5
    # 3 stelle --> 0        ==> punteggio= 0.5*Stella-1.5
    # 4 stelle --> 0.5
    # 5 stelle --> 1
           #normalizzazione tra 0 e 10
    valoreUtente = utentidict[subel['user_id']]
    valoreUtentenorm=10/(massimo-minimo)*(valoreUtente-massimo)+10
    subel['funny']=element['funny']
    subel['cool']=element['cool']
    subel['useful']=element['useful']
    subel['vlutente']=valoreUtentenorm
    subel['valorelib']= lib                        #valorelib
    subel['valoretext']=(lib+0.5*element['stars']-1.5)/2 #valore solo
        testo
    #valore opinione
    #subel['valoreOp']= (lib+0.5*element['stars']-1.5)*(1 + 0.7*
       element['useful']+0.1*element['funny']+0.2*element['cool'])*(1
        + valoreUtente)
    jsonlist.append(subel)
#mettere l'output su un file json che si chiama sintesireview.json
#inserire il file nel database.
#jsonobj=json.dumps(jsonlist)
with open('sintesireviewidea.json','w') as outfile:
    json.dump(jsonlist,outfile)
#vedere come salvare sul db!
collection = db.sintesireviewidea
clear = collection.remove()
           #pulisco la collection da eventuali scritture precedenti
result = collection.insert_many(jsonlist)
   #inserisco gli elementi calcolati '
client.close()
```

4.3 tuning.py

Tale script è stato realizzato per poter effettuare il tuning dei parametri relativi alla formula che permette di stimare il valore dell'opinione pubblica. Come già anticipato nel capitolo 1, l'opinione pubblica è funzione di 3 elementi fondamentali:

1. ContributoRecensione: è il contributo espresso dalla recensione costitutito da due elementi:
    - Stelline: è una metrica di valutazione (da 1 a 5) da parte di un utente

- Testo: testo in linguaggio naturale espresso dall'utente. Tale elemento, processato opportunamente, quantifica la positività o negatività del pensiero dell'utente.

2. ContributoUtente: è il peso associato a chi ha scritto la recensione. In effetti ogni pensiero viene sempre pesato in base al suo autore, per questo motivo una recensione scritta da un utente "buono", come precedentemente spiegato, ha un impatto maggiore sull'opinione pubblica.

3. SuccessoRecensione: è il peso associato a quanto popolare è stata quella recensione. In effetti una recensione che ha avuto molto successo in quanto è stata letta e presa in considerazione da molti utenti, ha inevitabilmente un impatto maggiore sull'opinione pubblica.

In particolare il successo della recensione è costituito dai tre fattori "cool, funny, useful", che rappresentano degli apprezzamenti effettuati da altri utenti. Ad ognuno dei tre fattori è stato attribuito un peso, rispettivamente "alpha, beta, gamma", e pertanto la formula che stima il valore di opinione pubblica è:

$Vopinione = ContributoRecensione * (1 + ContributoUtente) * (1 + \alpha * useful + \beta * cool + \gamma * funny)$

Il fattore 1 è stato aggiunto al fine di evitare i casi in cui il contributo utente, oppure il contributo di successo, sono nulli. Nello specifico:

- Contributo recensione: è un valore che è compreso nel range [-1,1]
- Contributo utente: è un valore che è stato normalizzato nel range [0,10]
- Successo recensione: è un valore che è stato normalizzato nel range [0,10]

Da quanto detto si evince che il contributo della recensione può essere solo amplificato in modulo dai restanti fattori moltiplicativi, e quindi una recensione positiva scritta da un utente "buono" e apprezzata, avrà un valore di opinione più alto rispetto al suo valore di base. Ciò significa che il valore di opinione conserva il segno del contributo di recensione di partenza.

Il tuning dei parametri è stato effettuato considerando che valori sempre crescenti di useful, funny e cool, portano ad un aumento del valore di opinione. Quindi sono stati determinati i valori di aplha beta e gamma che massimizzano il valore di opinione pubblica di una review e ciò è stato realizzato andando a determinare le triple di valori che massimizzano ogni review, ed è stata scelta la tripla con frequenza maggiore. Di seguito è riportato il listato relativo allo script:

```
1  #!/usr/bin/python
2  import pymongo
3  from pymongo import MongoClient
4  import numpy as np
5  from sklearn.metrics
6  import mean_squared_error from math
7  import sqrt
8  import itertools
9  import operator
10 from collections import Counter
11 #Connessione a MongoDB
12 client = MongoClient('localhost',27017)
```

```
db = client.YelpDB
cursor=db.sintesireviewidea.find()
print cursor.count()
val=[(0,0.1,0.9),(0,0.2,0.8),(0,0.3,0.7),(0,0.4,0.6),(0,0.5,0.5)
    ,(0,0.6,0.4),(0,0.7,0.3),(0,0.8,0.2),(0,0.9,0.1),(0.1,0,0.9)
    ,(0.1,0.1,0.8),(0.1,0.2,0.7),(0.1,0.3,0.6),(0.1,0.4,0.5)
    ,(0.1,0.5,0.4),(0.1,0.6,0.3),(0.1,0.7,0.2),(0.1,0.8,0.1)
    ,(0.1,0.9,0),(0.2,0,0.8),(0.2,0.1,0.7),(0.2,0.2,0.6),(0.2,0.3,0.5)
    ,(0.2,0.4,0.4),(0.2,0.5,0.3),(0.2,0.6,0.2),(0.2,0.7,0.1)
    ,(0.2,0.8,0),(0.3,0,0.7),(0.3,0.1,0.6),(0.3,0.2,0.5),(0.3,0.3,0.4)
    ,(0.3,0.4,0.3),(0.3,0.5,0.2),(0.3,0.6,0.1),(0.3,0.7,0),(0.4,0,0.6)
    ,(0.4,0.1,0.5),(0.4,0.2,0.4),(0.4,0.3,0.3),(0.4,0.4,0.2)
    ,(0.4,0.5,0.1),(0.4,0.6,0),(0.5,0,0.5),(0.5,0.1,0.4),(0.5,0.2,0.3)
    ,(0.5,0.3,0.2),(0.5,0.4,0.1),(0.5,0.5,0),(0.6,0,0.4),(0.6,0.1,0.3)
    ,(0.6,0.2,0.2),(0.6,0.3,0.1),(0.6,0.4,0),(0.7,0,0.3),(0.7,0.1,0.2)
    ,(0.7,0.3,0),(0.8,0,0.2),(0.8,0.1,0.1),(0.8,0.2,0),(0.9,0,0.1)
    ,(0.9,0.1,0)]

valori=[]
 tuplemax=[]
calcolomax=[]
i=1
for e in cursor:
    for t in val:
      valori.append(abs(e['valoretext']*(1+t[0]*e['useful']+t[1]*e['
          cool']+t[2]*e['funny'])*(1+e['vlutente'])))
    print i
    i=i+1
    tupla=val[valori.index(max(valori))]
    tuplemax.append(tupla)
    f=open("tuning.txt", "a+")
    f.write("Tupla: %5.1f %5.1f %5.1f \n" %(tupla[0],tupla[1],tupla
        [2]))
    f.close()
    del valori[:]
cnt = Counter(tuplemax)
      print cnt.most_common(3)
#f=open("tuning.txt", "a+")
#f.write("Tupla: %5.1f %5.1f %5.1f  \n" %(tup[0],tup[1],tup[2]))
#f.close()
client.close()
```

4.4 valoreOpinione.py

Tale script è stato realizzato al fine di determinare il valore di opinione di una determinata review.

4.5 valoreCumulativo.py

Tale script è stato realizzato al fine di determinare il valore cumulativo dell'opinione pubblica. Ciò è stato fatto per cercare di modellare quello che è il pensiero di un utente che si trova a leggere delle review di un relativo business. In particolare il comportamento modellato è il seguente: si associa un'importanza maggiore alle review più attuali, e alle review precedenti si associa un'importanza man mano sempre più piccola. Quindi il pensiero dell'utente sarà influenzato dalle review appena lette a cui viene aggiunto un contributo dovuto alle review lette precedentemente.