Popolarità in Twitter tramite HITS e PageRank

Introduzione

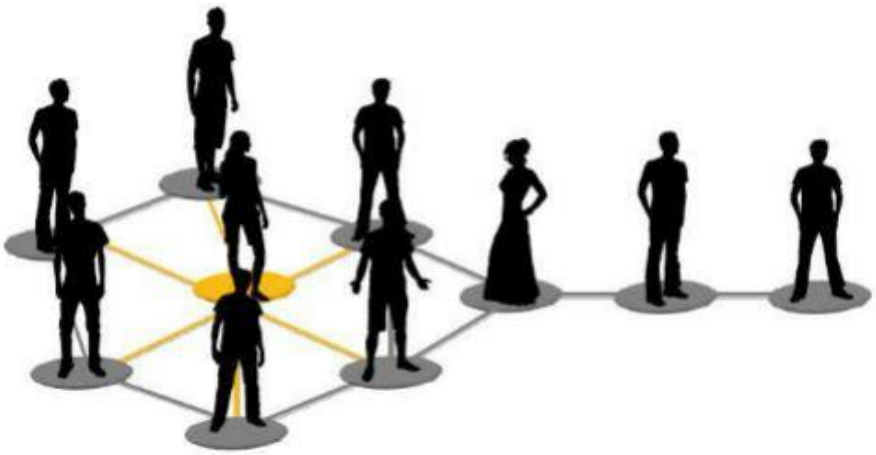
I Social Network come Facebook, Twitter, Google+ e LinkedIn hanno milioni di utenti e sono in continua evoluzione per questo sono un'ottima fonte di informazioni.
L'analisi dei Social Network si concentra principalmente sugli aspetti di Social Networking enfatizzando le relazioni tra utenti e i modelli di interazione tra utente e le informazioni sul contenuto. Uno dei temi di ricerca comune si concentra sulle misure di centralità in cui le informazioni utili delle persone connesse nella rete sociale sono rappresentate in un grafico.
In questo lavoro, abbiamo impiegato due algoritmi di ranking largamente utilizzati per la classificazione di pagine web in un contesto diverso: la classificazione degli utenti in un Social Network.

Gli algortmi in questione sono HITS (Hyperlink-Induced Topic Cerca) e PageRank.
Li abbiamo inoltre testati su input crescenti, ai fini di un analisi prestazionale e qualitativa.

Il Social Network su cui abbiamo fatto la nostra analisi è Twitter, una delle più grandi piattaforme di social networking, ampiamente utilizzato pe customer relationship management, marketing e branding.

Obiettivi

L'obiettivo del nostro lavoro è lo studio delle relazioni tra gli utenti di Twitter, analizzando la popolarità di ciascun utente tramite gli algoritmi HITS e PageRank, implementati in Java con l'utilizzo delle librerie del framework Hadoop MapReduce.



Dato in input una lista di coppie di utenti, nel formato Utente1 segue Utente2, il programma restituirà in output il ranking degli utenti ordinati per popolarità nei due diversi algoritmi, in cui non si terrà conto solo del numero dei seguaci ma anche l'importanza dei seguaci.

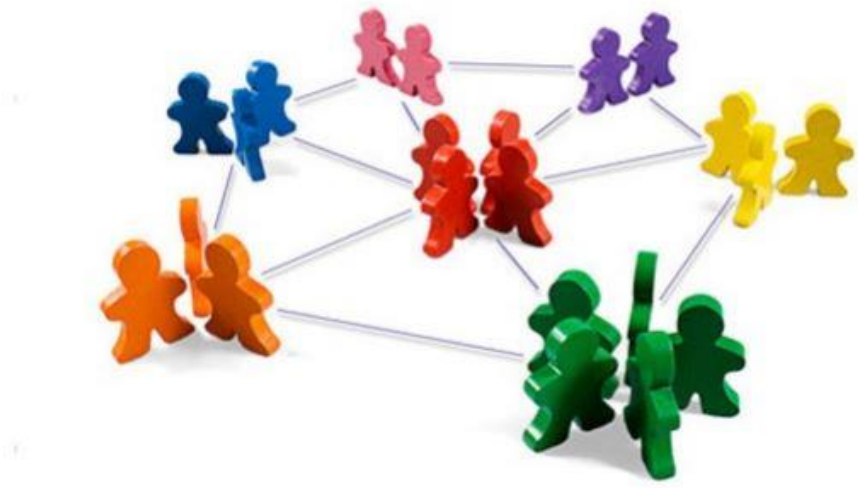
Inoltre di nostro interesse è stato anche analizzare in termini prestazionali la computazione dei due algoritmi al variare del dataset.

Motivazioni

Entrambi gli algoritmi sono largamente utilizzati nel mondo dei motori di ricerca per generare un ranking a partire da una query.
Analizzando il loro funzionamento, abbiamo visto che entrambi effettuano la loro computazione su un grafo: ogni nodo rappresenta una pagina ed ogni arco rappresenta il collegamento presente tra due pagine.
L'importanza di una pagina viene calcolata in base ai diversi tipi di collegamenti e all'importanza delle pagine a cui è collegata.

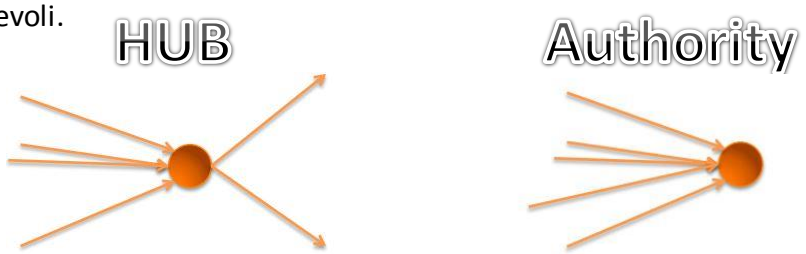
Abbiamo di conseguenza utilizzato lo stesso ragionamento su una lista di utenti collegati tra loro, scenario tipico di un Social Network come Twitter, per ottenere la lista degli top digital influencer di una comunità.

L'utilizzo di HITS e PageRank sul grafo di Twitter può essere molto utile per quello che è la nuova frontiera del business, il web advertising. Avere l'informazione su chi sono gli utenti più influenti di una comunità, che può essere segmentata per target audience, potrebbe aumentare il riscontro dei propri obiettivi di business in maniera immediata. Basti pensare a come potrebbe essere efficiente una campagna pubblicitaria se fatta dai top influencer di una comunità o come potrebbe essere trasmesso un messaggio di sensibilizzazione alla web community.



HITS

L'algoritmo Hyperlink-Induced Topic Search è un algoritmo di link analisys sviluppato da Jon Kleinberg. Si basa sul calcolo di 2 valori: **authority** score ed **hub** score che stimano rispettivamente la qualità del contenuto informativo sugli argomenti di ricerca e la quantità di link di una pagina verso pagine autorevoli.



Entrambi i valori sono calcolati tramite un algoritmo iterativo basato soltanto sulla struttura dei link.
L'idea alla base dell'algoritmo HITS è la Relazione di mutuo rinforzo tra hub e authority score:

Buoni hub puntano a buone authority e buone autority sono puntate da buoni hub.

Si evince quindi che i due valori sono strettamente correlati: uno è funzione dell'altro. Basato su iterazioni, dopo un numero ridotto l'algoritmo converge.
Vediamo in dettaglio il suo funzionamento:

Date due entità p e q ed un arco tra le due entità, l'aggiornamento di hub score e authority score avviene secondo le seguenti formule:

$$A(p) = \sum_{q \to p} H(q) \text{ (i.e., che puntano a } p\text{)}$$

$$H(p) = \sum_{p \to q} A(q) \text{ (i.e., che sono puntati da } p\text{)}$$

Le equazioni indicano che A(p) è la somma degli hub score delle entità che putano a p ed H(p) è la somma degli authority score delle entità puntate da p.

Una possibile implementazione in pseudocodice dell'algoritmo è riportata di seguito:

Algorithm 1 HITS

```
1: procedure HITS(G = (V, E), K)
2:     A_0(p) ← 1 ∀p ∈ V
3:     H_0(p) ← 1 ∀p ∈ V
4:     for i = 1 to K do
5:         A_i(p) ← 0 ∀p ∈ V
6:         H_i(p) ← 0 ∀p ∈ V
7:         Z_A ← 0
8:         Z_H ← 0
9:         for p ∈ V do
10:            for q ∈ V do
11:                if (p, q) ∈ E then
12:                    H_i(p) ← H_i(p) + A_{i−1}(q)
13:                    Z_H ← Z_H + A_{i−1}(q)
14:                end if
15:                if (q, p) ∈ E then
16:                    A_i(p) ← A_i(p) + H_{i−1}(q)
17:                    Z_A ← Z_A + H_{i−1}(q)
18:                end if
19:            end for
20:        end for
21:        for p ∈ V do
22:            A_i(p) ← A_i(p) / Z_A
23:            H_i(p) ← H_i(p) / Z_H
24:        end for
25:    end for
26:    return A_K, H_K
27: end procedure
```

1. inizializzazione a 1 di tutti gli hub e authority score delle entità candidate;

2. aggiornamento di hub e authority score secondo le equazioni precedenti;

3. gli hub score sono normalizzati in modo che la loro somma sia pari a 1;

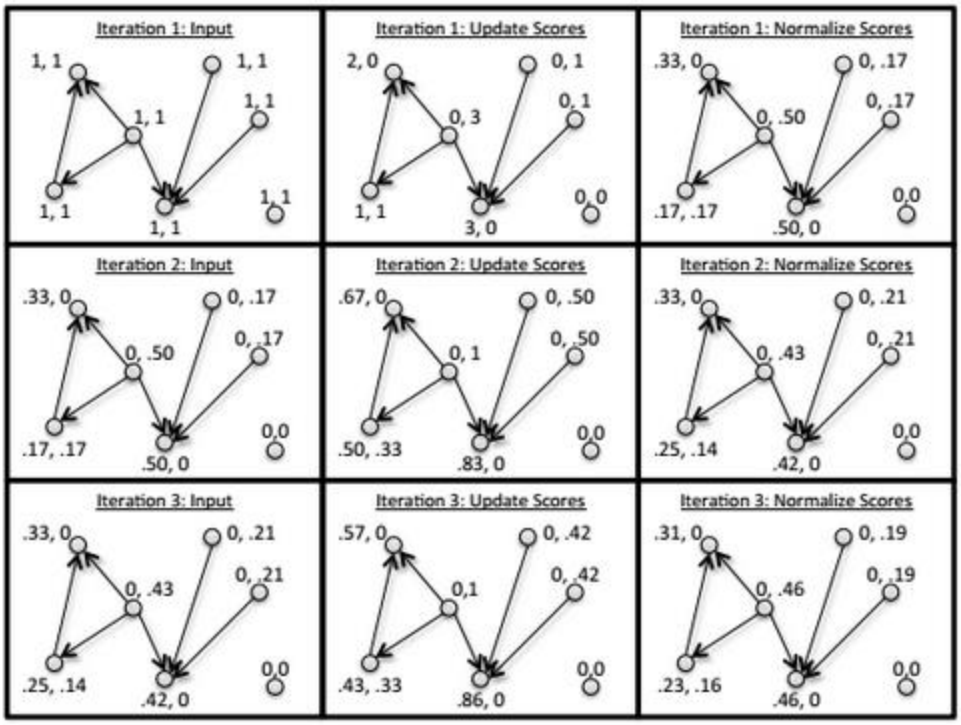
4. gli authority score sono normalizzati in modo che la loro somma sia pari a 1;

5. l'intero processo è poi ripetuto sugli score normalizzati per un numero fisso di iterazioni indicato da *K;*

6. l'algoritmo restituisce gli authority score e hub score di tutte le entità candidate.

E' dimostrato che l'algoritmo converge sempre, e solitamente questo avviene dopo un numero ridotto di iterazioni.

Per capire ancora meglio il funzionamento dell'algoritmo, ecco l'illustrazione del suo funzionamento su un grafo di esempio con 7 nodi e 6 archi, importando un numero di iterazioni pari a 3



Una volta che authority ed hub score sono stati calcolati, le entità possono essere classificate secondo i loro authority score. Tale lista conterrà le entità più authotitative all'interno del set.

Implementazione di HITS in Map Reduce

Per l'implementazione in Map Reduce abbiamo utilizzato 3 job. Prendendo in input un dataset di utenti twitter nel formato utente1 utente2, che implica che il primo utente segue il secondo utente, i job sono articolati in questo modo:

1. Il primo job si occupa di associare a ciascun utente una lista di utenti che segue (archi uscenti) e la lista di seguaci (archi entranti). Inoltre imposta authority score ed hub score a 1.0. Come output dalla prima iterazione dell'algoritmo si ha quindi un file nella forma:

Utente 1.0 1.0 listaUtentiSeguiti listaUtentiCheSegue

2. Il secondo job prende in input l'output del primo job ed effettua il calcolo di authority score ed hub score secondo le formule illustrate in precedenza. Questo job viene richiamato tante volte quante sono le iterazioni scelte per l'algoritmo. Se le iterazioni sono poche l'algoritmo non converge, se sono molte il tempo di esecuzione diventa eccessivo.

3. Infine l'output del secondo job viene dato in input al terzo ed ultimo job, che si occupa di creare 2 liste ordinate: una per authority score ed una per hub score.

PageRank

PagerRank è un algoritmo di classificazione sviluppato dai fondatori di google Sergey Brin e Lawrance Page. Questo algoritmo si basa sull'assegnazione di un peso numerico ad ogni entità (il pageRank appunto) che tiene conto del valore delle entità che puntano ad essa.

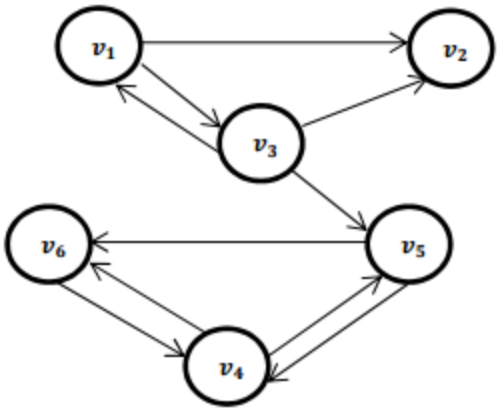
Il PageRank di una pagina P_I denominato con r(P_I) è la somma dei pageRanks di tutte le pagine che puntano a P_I.

$$r(P_i) = \sum_{P_j \in B_{P_i}} \frac{r(P_j)}{|P_j|}$$

Dove b_{p_i} è la somma di tutte le pagine che puntano a P e $|P_J|$ è il numero dei link uscenti della pagina P_J. Come per HITS, anche qui è necessaria una prima iterazione in cui vengono dati i valori iniziali. Nel caso di PageRank, si avrà:

$$r_0(P_i) = \frac{1}{n}$$

Questo valore sarà utilizzato come iniziale per tutte le pagine N. Vediamo un esempio di applicazione:



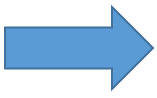
Iterazione 0	Iterazione 1	Iterazione 2	Rank all'interazione 2^a
$r_0(P_1) = 1/6$	$r_1(P_1) = 1/18$	$r_2(P_1) = 1/36$	5
$r_0(P_2) = 1/6$	$r_1(P_2) = 5/36$	$r_2(P_2) = 1/18$	4
$r_0(P_3) = 1/6$	$r_1(P_3) = 1/12$	$r_2(P_3) = 1/36$	5
$r_0(P_4) = 1/6$	$r_1(P_4) = 1/4$	$r_2(P_4) = 17/72$	1
$r_0(P_5) = 1/6$	$r_1(P_5) = 5/36$	$r_2(P_5) = 11/72$	3
$r_0(P_6) = 1/6$	$r_1(P_6) = 1/6$	$r_2(P_6) = 14/72$	2

Implementazione di PageRank in Mapereduce

L'implementazione di PageRank in MapReduce segue gli stessi step di HITS:

1. Il primo job si occupa di associare per ciascun utente una lista di utenti che segue (archi uscenti) e di inizializzare il valore di rank a 1.0

INPUT	
User 1	User 2
User 2	User 1
……	……



OUTPUT	
Key	**Value**
User1 1.0	User 2 , User .., ….
……. 1.0	……., ……., …….

2. Il secondo Job prende come input l'output della prima iterazione ed inizia il calcolo del PageRank in base alla formula precedente. Viene lanciato iterativamente (Noi abbiamo scelto 5 iterazioni).

INPUT	
Key	**Value**
User1 1.0	User 2 , User .., ….
……. 1.0	……., ……., ……..

OUTPUT	
Key	**Value**
User1 1.345	User 2 , User ..,
……. 1.83	……., ……., …..

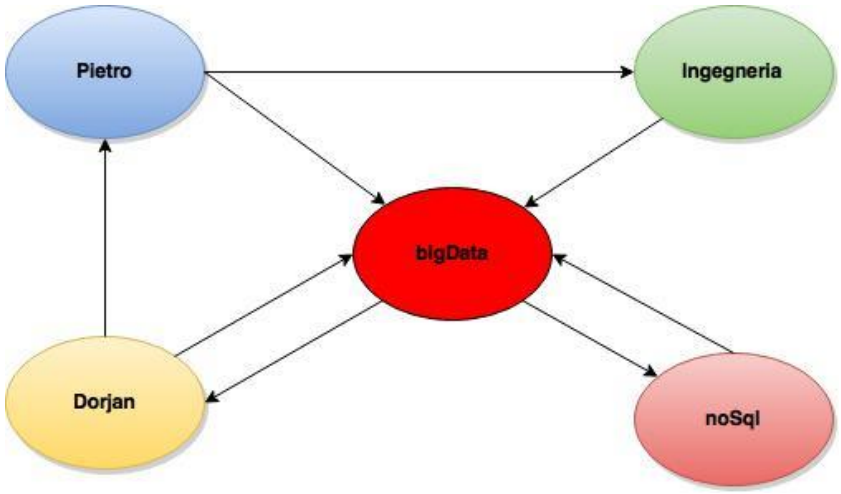
3. Il terzo job si occupa dell'ordinamento della lista in base al PageRank.

OUTPUT	
Key	**Value**
User1 1.345	User 2 , User ..,
……. 1.83	……., ……., …..

OUTPUT	
Key	**Value**
2.84	User 34
2.304	User …
….	……

Sperimentazione

Per comprendere meglio il funzionamento in MapReduce dei due algoritmi, vediamo un esempio su un piccolo dataset. Supponiamo di avere il seguente grafo di relazioni:



Abbiamo quindi 5 entità che rappresentano gli utenti con le relative relazioni, in cui gli archi rappresentano la relazione di follow/follower tra la coppia di utenti.

HITS:

Alla prima iterazione il processo setta ad 1.0 sia authority score che hub score ed associa a ciascun utente una lista di utenti che segue (archi uscenti) e la lista di seguaci (archi entranti), di seguito l'output di Hadoop:

```
Dorjan 1.0 1.0  Pietro,bigData          bigData
Ingegneria 1.0 1.0      bigData         Pietro
Pietro 1.0 1.0  Ingegneria,bigData              Dorjan
bigData 1.0 1.0         Dorjan,noSql            Pietro,Dorjan,Ingegneria,noSql
noSql 1.0 1.0   bigData         bigData
```

Nelle iterazioni successive verranno aggiornati hub score ed authority score fino a convergere in uno stato stabile. Nel processo è stato impostato sempre a 5 il numero iterazioni

```
Dorjan  1.0     1.3416407864998738      Pietro,bigData  bigData
Ingegneria      1.0     1.0     bigData Pietro
Pietro  1.0     1.3416407864998738      Ingegneria,bigData      Dorjan
bigData 1.9711971193069777      1.414213562373095       Dorjan,noSql    Pietro,Dorjan,Ingegneria,noSql
noSql   1.0     1.0     bigData bigData
```

Oss: Si nota una differenza minima tra le due iterazioni, dovuto al fatto che il dataset è molto piccolo. Nei test successivi fatti su input di dimensione dell'ordine di Giga, la differenza tra le iterazioni è molto più marcata.

Infine si hanno due liste in output, di seguito viene mostrato il ranking degli utenti ordinati in base ad authority score:

```
1.0     noSql
1.0     Pietro
1.0     Ingegneria
1.0     Dorjan
1.978853754943805       bigData
```

Questo output rispecchia esattamente il grafo che abbiamo utilizzato come input. Solo un'entità sovrasta le altre per autorità, l'entità bigData. Le altre entità hanno peso uguale in quanto si influenzano a vicenda e nessuna prevale sulle altre.

PageRank:

Nella prima iterazione viene settato a 1.0 il pagerank di tutti gli utenti, ed associato a ciascuno una lista che rappresenta gli utenti che segue.Vediamo l'output della prima iterazione:

```
Dorjan  1.0     Pietro,bigData
Ingegneria      1.0     bigData
Pietro  1.0     Ingegneria,bigData
bigData 1.0     Dorjan,noSql
noSql   1.0     bigData
```

Nelle successive iterazioni verrà effettuato il calcolo vero e proprio secondo la formula illustrata in precedenza. Le varie iterazioni hanno lo scopo di far convergere i valori ad un risultato stabile. Nel

nostro caso abbiamo preferito un numero di iterazioni pari a 5 che ci sembra un giusto compromesso tra accuratezza dell'input e tempi di esecuzione.

```
Dorjan   0.9086821      Pietro,bigData
Ingegneria      0.36491615      bigData
Pietro   0.6304773      Ingegneria,bigData
bigData  2.1872425      Dorjan,noSql
noSql    0.9086821      bigData
```

Figura 1. Dopo la 5 iterazione

Il risultato finale sarà una lista ordinata in base al pageRank, che tiene conto sia del numero di utenti che seguono l'entità sia del relativo PageRank dei seguaci. Detto in termini più semplici, un'entità avrà alto pageRank se è seguito da molte entità o se è seguito da poche entità con alto pageRank

```
0.36491615      Ingegneria
0.6304773       Pietro
0.9086821       Dorjan
0.9086821       noSql
2.1872425       bigData
```

Figura 2. Ranking utenti

Confronto degli output:

```
1.0     noSql                   |   0.36491615      Ingegneria
1.0     Pietro                  |   0.6304773       Pietro
1.0     Ingegneria              |   0.9086821       Dorjan
1.0     Dorjan                  |   0.9086821       noSql
1.978853754943805       bigData |   2.1872425       bigData
```

HITS **PageRank**

Figura 3. Algoritmo HITS(Authority) e PageRank a confronto

Come si può notare, entrambi gli algoritmi convergono verso lo stesso risultato. La differenza principale sta nelle entità che circondano il core. Pagerank in questo caso offre una valutazione dell'insieme di tutte le entità basate principalmente sugli archi entranti. HITS invece calcola il Core della comunità basandosi sia su archi entranti che su archi uscenti, che influenzano sia hub score che authority score. Per capire ancora meglio il concetto, aggiungiamo altre 2 relazioni: Dorjan->Ingegneria e BigData->Ingegneria. Gli output sono i seguenti:

1.0	noSql		0.37408626	Pietro
1.0	Pietro		0.698226	Dorjan
1.0	Dorjan		0.698226	noSql
1.7268450704412373	Ingegneria		1.0653903	Ingegneria
1.9507601533293157	bigData		2.1640716	bigData
HITS			PageRank	

Figura 4. Ranking dopo l'aggiunta di due relazioni

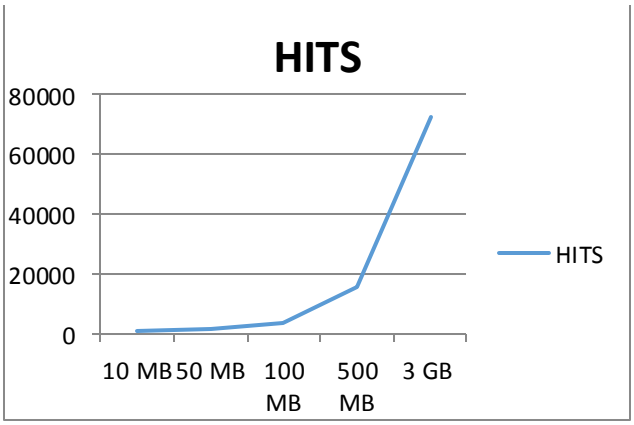
Come si può notare, l'aggiunta di archi entranti all'entità Ingegneria ha comportato un cambiamento sostanziale in HITS nel suo authority score, mentre non ha minimante intaccato le entità non coinvolte. In questo caso secondo HITS, al core della comunità si è aggiunta anche l'entità Ingegneria mentre il contorno rimane invariato. Per quanto riguarda pageRank, l'inserimento dei due archi entranti ha modificato la valutazione dei Rank di tutte le entità del grafo e non solo quella di Ingegneria. Questo dimostra e come PageRank dia una visione d'insieme dei nodi coinvolti nell'analisi.

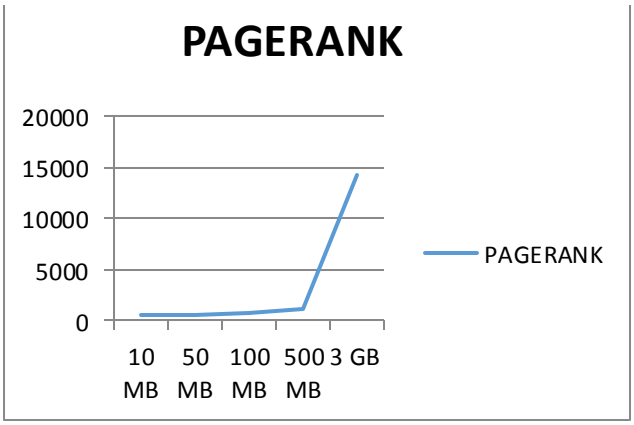
Risultati ottenuti

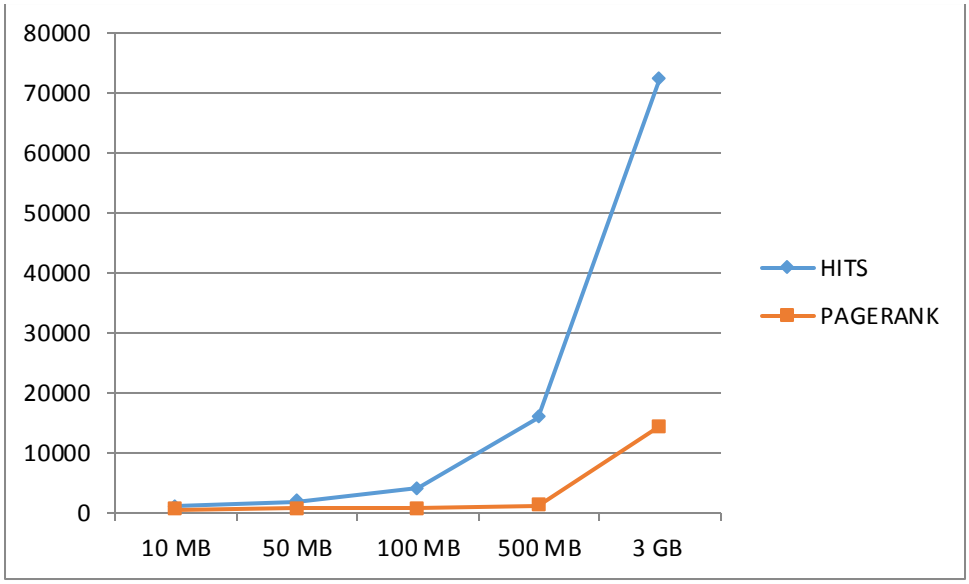
Siamo riusciti a implementare in tutte le specifiche entrambi gli algoritmi, il progetto è stato testato in **Amazon Elastic MapReduce** (**EMR**) di **Amazon** Web Service (AWS) dandogli in input diversi dimensioni di dataset.
Di seguito alcune statistiche dell'elaborazione:

Dimensione Dataset	HITS (hh:mm:ss)	PAGERANK (hh:mm:ss)
10 MB	00:15:52	00:09:24
50 MB	00:29:20	00:10:18
100 MB	01:06:02	00:11:02
500 MB	04:23:04	00:19:50
3 GB	20:06:03	03:58:49

Il trend del tempo di esecuzione al crescere dell'input:







Come si può notare il tempo di esecuzione di HITS è molto superiore a PageRank, questo è dovuto al fatto che si tiene in considerazione sia la relazione degli utenti seguiti sia dei seguaci.

CONCLUSIONI

Dopo lo sviluppo del progetto, possiamo concludere che entrambi offrono un risultato concorde con le aspettative: una classificazione di utenti in base alle relazioni tra di essi.

Entrambi gli algoritmi implementati presentano sia pregi che difetti: Page rank offre una buona qualità del risultato, mostrando sia il core sia l'intorno della comunità. Va però detto che non considera collegamenti entranti ed è molto sensibile ai collegamenti uscenti. Il tempo di esecuzione, come si evince anche dai grafici, è molto contenuto. HITS al contrario considera sia archi entranti che uscenti ed offre un risultato meno preciso di PageRank , considerando solo cambiamenti rilevanti all'interno del core della comunità. Al contrario di pageRank,Il tempo di esecuzione è molto elevato. Entrambi gli algoritmi sono notevolmente influenzati dal numero di iterazioni utilizzate e trovare il giusto compromesso tempo/accuratezza non è semplice, sopratutto quando l'input è di notevoli dimensioni.

Possiamo quindi concludere che sia HITS che PageRank sono valide soluzione al problema delle comunità in ambito social, con alcuni accorgimenti:

- PageRank è ottimo per tempi di risposta ma considerando solo archi entranti, ovvero la lista di "follower" di un determinato utente, a nostro parere non è particolarmente indicato. Può offrire una buona panoramica iniziale del dataset su cui basare un lavoro successivo

- HITS risulta molto pesante per i tempi di esecuzione, ma l'output che produce è più in linea con le specifiche del problema rispetto a pageRank. La comunità nel dataset viene trovata in tutti gli input testati ed è facilmente individuabile dai valori di authority score.

SVILUPPI FUTURI

Il prossimo step potrebbe essere la rappresentazione visuale del risultato tramite un grafo, utilizzando la dimensione dei nodi come indice per authority Score/PageRank. Questo tipo di visualizzazione permette una rapida identificazione degli utenti principali all'interno della comunità.

Sarà inoltre interessante l'utilizzo degli stessi algoritmi su parametri differenti: citazioni, retweet, tipo di condivisioni, etc. identificando quindi comunità diverse.