Università degli studi di Padova
Dipartimento di Scienze Statistiche

Corso di Laurea Triennale in
Statistica e Tecnologie Informatiche

Relazione Finale

Analisi di Rete di Microblogging sul Machine Learning

Relatore: Livio Finos
Correlatore: Dario Solari
Dipartimento di Scienze Statistiche

Laureando: Andrea Salin
Matricola: 619466 - STI

Anno Accademico 2013/2014

Indice

Introduzione

i. Presentazione della Tesi

Questa relazione finale descrive la mia attività di tirocinio svolta con il Professore Livio Finos e con il supporto del Dottore Dario Solari: li ringrazio sentitamente per il grande aiuto datomi. Il lavoro svolto riguarda lo studio di un campione di tweets riguardanti a Machine Learning: tweet (in italiano cinguettio) è il nome affibbiato ai messaggi di un servizio di microblogging che, in questi anni, è diventato estremamente popolare ed è tutt'oggi il più diffuso, ovvero Twitter.

La relazione è così suddivisa:
1) Il primo capitolo presenta l'analisi del dataset: si inizierà con l'estrazione e preparazione dei dati passando successivamente alle prime semplici analisi descrittive che consentiranno di comprenderne la struttura e coglierne alcune informazioni importanti.
2) Il secondo capitolo introduce concetti base della SNA e della Teoria dei Grafi: si useranno strumenti di entrambe le metodologie per ricavare e analizzare diverse reti riguardanti la comunicazione tra gli utenti e le co-occorrenze di particolari termini.

Le analisi dell'intera tesi sono state fatte con il software statistico R (versione 3.1.1), mentre per la costruzione delle tabelle ci siamo serviti di Apache OpenOffice Calc.

Capitolo 1

Analisi descrittive

In questo capitolo vedremo come ottenere il dataset dei tweets e quindi come utilizzarlo per ricavare alcune informazioni iniziali attraverso semplici analisi. Partiremo dalla preparazione dei dati in nostro possesso, ovvero eseguiremo delle operazioni, come la pulitura del formato e la normalizzazione del testo dei tweets, che ci permetteranno di condurre le prime e anche successive analisi più facilmente. In seguito, ci concentreremo nello studio delle distribuzioni giornaliere dei tweets e dell'utente più attivo; affronteremo il problema dei retweets mostrando una possibile soluzione per, infine, ricavare le parole, gli n-grammi e gli hashtag più frequenti. Assegneremo inoltre delle etichette ad alcuni specifici termini che useremo nelle analisi della fase successiva.
Per effettuare le diverse operazioni, ci serviremo di alcune particolari funzioni presenti nelle librerie del software statistico R: nello specifico, l'uso della libreria "TextWiller" ci tornerà spesso utile.

1.1 Estrazione del corpus dei tweets

Il dataset utilizzato è composto da più di 85000 tweets di lingue diverse raccolti tra il 2013-12-17 e il 2014-02-24 (scopriremo in seguito che alcuni giorni sono assenti) riguardanti la "Machine Learning": la scelta è stata effettuata attraverso query ("machine learning" OR "data mining" OR "data science" OR "data scientist"). Il dataset si presenta suddiviso in diverse parti che andremo ad unire. Ottenuto l'intero corpus, ne ricaviamo le prime informazioni con il comando:

```
str(tw)

'data.frame':   86931 obs. of  24 variables:
 $ text         : chr  "RT @Android_Agent: Google hires
         engineer away from Microsoft, said to be  ...
 $ favorited    : int  0  0  0  0  0  0  0  0  0  0 ...
 $ favoriteCount: num  0  0  0  1  0  0  0  0  0  1 ...
 $ replyToSN    : chr  NA NA NA NA ...
 $ created      : POSIXct, format: "2013-12-16 18:58:28"
         "2013-12-16 18:58:32" ...
 $ truncated    : int  0  0  0  0  0  0  0  0  0  0 ...
 $ replyToSID   : chr  ""  ""  ""  "" ...
 $ id           : chr  "412657999199473664" ...
 $ replyToUID   : chr  ""  ""  ""  "" ...
 $ statusSource : chr  "<ahref=\"http://twitter.com/
         download/android\" ...
 $ screenName   : chr  "CynthiaJSimmons" "Jerebntz"
         "bryangoodrich" "dmaisaku_f132" ...
 $ retweetCount : num  1  13  0  0  0  0  0  0  0  0 ...
 $ isRetweet    : int  1   1  0  0  0  0  0  0  0  0 ...
 $ retweeted    : int  0   0  0  0  0  0  0  0  0  0 ...
 $ longitude    : num  NA NA NA NA NA NA NA NA NA NA ...
 $ latitude     : num  NA NA NA NA NA NA NA NA NA NA ...
 $ ts           : chr  "2013-12-16 19:39:43"
         "2013-12-16 19:39:43" "2013-12-16 19:39:43" ...
 $ lang         : chr  NA NA NA NA ...
 $ sentiment    : int  NA NA NA NA NA NA NA NA NA NA ...
 $ geocode      : chr  NA NA NA NA ...
 $ lang_twitter : chr  NA NA NA NA ...
```

Le variabili che useremo per i nostri studi sono le seguenti:

$ text: contiene il testo di ciascun tweet;

$ created: contiene le date e gli orari in cui i tweets sono stati inviati/creati;

$ screenName: contiene i nomi degli utenti/users.

1.2 Preparazione dei dati

Le operazioni che eseguiamo ora andranno a modificare i nostri dati, per renderli più agevoli alle fasi successive, e a rimuovere quelli meno significativi. Iniziamo con la pulitura del formato che ci permetterà di gestire meglio le date del dataset in possesso. Per farlo, utilizziamo la seguente funzione presente nella libreria "TextWiller" (Solari, Finos, Redaelli, con contributi di Marco Rinaldo, Branca, and Ferraccioli., 2013):

```
tw = fixTimeStamp(tw)
```

Ci serviamo della funzione "textcat", inclusa nella stessa libreria (Hornik, Mair, Rauch, Geiger, Buchta,and Feinerer, 2013), che ricaverà per ogni tweet il linguaggio. La tabella dei conteggi relativa è la 1.1.

```
lang=textcat(testo,ECIMCI_profiles)
sort(table(lang),decreasing=T)
```

Lingua	Cont.	Lingua	Cont.
Inglese	80324	Croato	110
Francese	1064	Rumeno	93
Svedese	1026	Polacco	91
Olandese	811	Ceco	82
Spagnolo	661	Portoghese	74
Latino	585	Albanese	67
Italiano	347	Finlandese	51
Tedesco	271	Bosniaco	49
Ungherese	174	Slovacco	45
Norvegese	124	Sloveno	45
Danese	123	Serbo	43

Tabella 1.1: Conteggi dei tweets secondo il linguaggio

Passiamo a verificare la presenza di tweets senza testo: il dataset contiene un totale di 599 tweets vuoti che decidiamo di rimuovere.

```
na = which(is.na(tw$text))
```

L'ultima di queste azioni per la preparazione dei nostri dati consiste nella normalizzazione dei tweets: grazie alla funzione "normalizzaTesti" della libreria "TextWiller" andiamo a pulirne il testo. Il risultato ottenuto lo aggiungeremo in una nuova variabile del nostro dataset:

```
tw$norm=normalizzaTesti(tw$text,normalizzacaratteri=TRUE,
     tolower=TRUE,perl=FALSE,fixed=FALSE)
```

La funzione, se richiesto, ricerca gli indirizzi web, i caratteri, la punteggiatura, gli slang e gli emoticon sostituendoli con particolari stringhe o eliminandoli. Vengono, inoltre, mostrati i conteggi dei caratteri speciali trovati nel testo originale. Vediamo, come esempio, l'effetto della normalizzazione fatta sul primo tweet del dataset:

```
tw$text[1]
[1] "RT @Android_Agent: Google hires engineer away from
Microsoft, said to be working on ‘machine learning’
http://t.co/g9oOongeIJ #android"

tw$norm[1]
[1] "rt @android agent google hires engineer away from
microsoft said to be working on ‘machine learning’
wwwurlwww #android"
```

1.3 Distribuzioni giornaliere

Iniziamo con qualche analisi descrittiva partendo dallo studio delle distribuzioni giornaliere dei tweets: al nostro dataset, aggiungiamo nuove variabili relative al giorno di creazione di ciascun messaggio. Il grafico relativo è rappresentato in figura 1.1 ottenuto grazie alla funzione “ggplot” del pacchetto “ggplot2”; mostriamo pure i conteggi giornalieri nella tabella 1.2.

```
tw$created<-as.POSIXct(tw$created)
tw$created.day <- as.POSIXct(round(tw$created,"day"))

ggplot(data.frame(tw$screenName,tw$created.day),
     aes(as.factor(tw.created.day)))+
     geom_bar(stat="bin",binwidth=1,drop=T)+
     theme(axis.text.x=element_text(angle=-90,size=10))

table(tw$created.day))
```

Sia dal grafico che dalla tabella, possiamo notare l'assenza di alcuni giorni (dal 2014-01-29 al 2014-02-07) e la presenza di un periodo di tempo dove sono stati inviati pochi tweets: questi, probabilmente, sono dovuti a un qualche errore o problema durante la raccolta dei dati.

Nelle fasi successive, ci concentreremo sulla parte del dataset considerata dalla funzione “textcat” di lingua inglese. Il numero totale di tweets è ora pari a 80324.

```
tw.eng=tw[which(lang=="en"),]
```

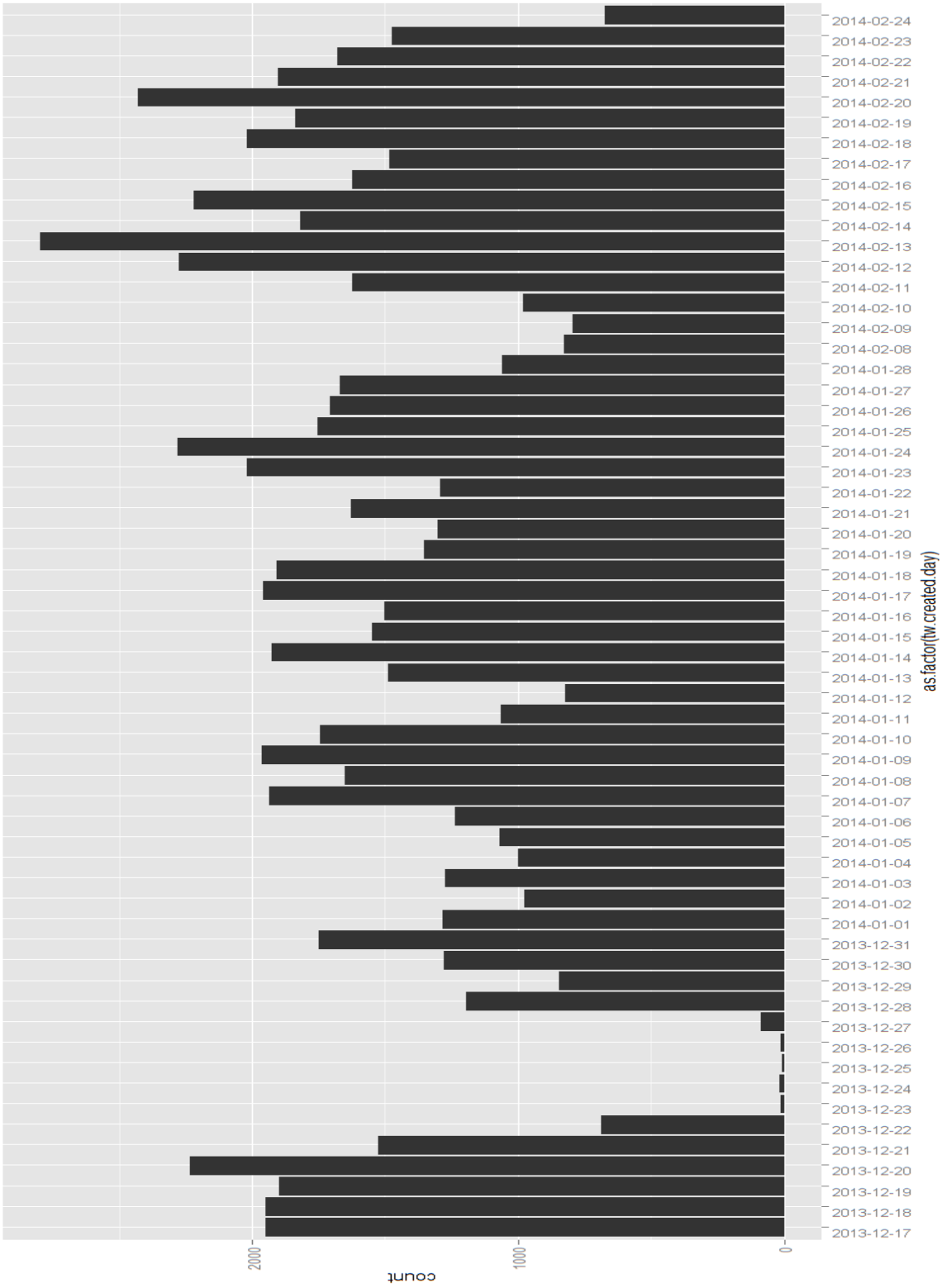


Figura 1.1: Distribuzione giornaliera dei tweets

Data	Cont.	Data	Cont.	Data	Cont.
2013-12-17	1947	2014-01-06	1236	2014-01-26	1709
2013-12-18	1947	2014-01-07	1936	2014-01-27	1668
2013-12-19	1900	2014-01-08	1649	2014-01-28	1062
2013-12-20	2234	2014-01-09	1961	2014-02-08	827
2013-12-21	1526	2014-01-10	1742	2014-02-09	797
2013-12-22	688	2014-01-11	1064	2014-02-10	980
2013-12-23	15	2014-01-12	824	2014-02-11	1621
2013-12-24	17	2014-01-13	1487	2014-02-12	2276
2013-12-25	8	2014-01-14	1926	2014-02-13	2797
2013-12-26	13	2014-01-15	1547	2014-02-14	1819
2013-12-27	89	2014-01-16	1504	2014-02-15	2218
2013-12-28	1197	2014-01-17	1959	2014-02-16	1623
2013-12-29	847	2014-01-18	1906	2014-02-17	1484
2013-12-30	1281	2014-01-19	1353	2014-02-18	2018
2013-12-31	1749	2014-01-20	1303	2014-02-19	1838
2014-01-01	1284	2014-01-21	1629	2014-02-20	2429
2014-01-02	976	2014-01-22	1292	2014-02-21	1902
2014-01-03	1272	2014-01-23	2019	2014-02-22	1680
2014-01-04	998	2014-01-24	2278	2014-02-23	1474
2014-01-05	1070	2014-01-25	1755	2014-02-24	672

Tabella 1.2: Conteggi giornalieri dei tweets

1.4 Analisi dell'utente più attivo

Di nostro interesse è anche ottenere l'utente più attivo e osservarne la relativa distribuzione giornaliera. Otteniamo i conteggi degli utenti più attivi con il comando (tabella 1.3):

```
sort(table(tw$screenName),decreasing=T)[1:20]
```

Utente	Cont.	Utente	Cont.
AnalyticsInnova	729	kdnuggets	154
jackwmson	615	imbigdata	125
anitayorker	404	dutchlight360	121
dataloco	357	InfoSysReddit	113
dmu8750	346	thedatacrunch	113
DataMiningAgent	303	DataMiningView	106
Datumbox	278	BigDataClub	98
data_nerd	264	Print2plan	98
BigPupazzoBlu	208	TJO_datasci	94
KirkDBorne	175	flavioclesio	92

Tabella 1.3: Conteggi dei tweets degli utenti più attivi

L'utente più attivo è “AnalyticsInnova” con un totale di 729 tweets; vediamo ora la sua distribuzione nel tempo con i seguenti comandi (figura 1.2):

```
AnalyticsInnova=subset(data.frame(tw$screenName,
     tw$created.day),tw$screenName=="AnalyticsInnova")
plot(table(AnalyticsInnova[,2]))
```

1.5 Il problema dei retweets

E' doveroso specificare che, senza ulteriori operazioni, i risultati di uno studio delle parole più frequenti, oppure una qualsiasi altra particolare ricerca interna al testo, verrebbero condizionati dalla presenza di quelli che sono definiti “retweet”: un retweet è la ripetizione di un tweet inviato da un utente ad opera di altri. Questi hanno allora fondamentale importanza poiché un individuo è tanto più “retweetato“ quanto maggiore è la sua popolarità e credibilità; ma possono anche rappresentare tweets interessanti, utili o di gran valenza nell'argomento di discussione e che quindi hanno maggior probabilità di essere riproposti dagli altri utenti. Da questo, possiamo comprendere come la presenza di retweet “predominanti” influenzino le frequenze di parole, o gruppi di esse, del nostro dataset. Di conseguenza, prima di procedere con lo studio dei termini di maggior ripetitività, gli scopi attuali sono quelli di individuare e, in seguito, di “rimuovere” (almeno) i retweets particolarmente riutilizzati e per farlo ci serviremo della funzione “RTHound” anch'essa contenuta nel pacchetto “TextWiller”.

1.5.1 RTHound

Individuare i retweets non è alquanto semplice e i motivi principali sono due: Twitter non è riuscito a codificare il messaggio come retweet oppure un utente ha citato solo parzialmente o ha apportato modifiche al testo. Per queste ragioni ci serviamo di una funzione per la ricerca dei retweet: la funzione RTHound. Questa si basa sulla distanza di Levenshtein o distanza di edit: la distanza di Levenshtein tra due stringhe A e B consiste nel numero minimo di modifiche elementari che consentono di trasformare A in B e con modifiche elementari si intendono la cancellazione, la sostituzione o l'inserimento di un carattere: ad esempio, la distanza tra la parola “cane” e la parola “cono” è di 2. La funzione costruisce una matrice quadrata contenente le distanze tra tutte le possibili combinazioni di coppie di tweets: essa servirà per clusterizzare i dati, accorpando i più simili tra loro e, una volta completata la procedura, cercherà il tweet più vecchio in ordine temporale in ogni cluster, che viene considerato il tweet originale e riassumerà quindi l'informazione contenuta in quel gruppo.

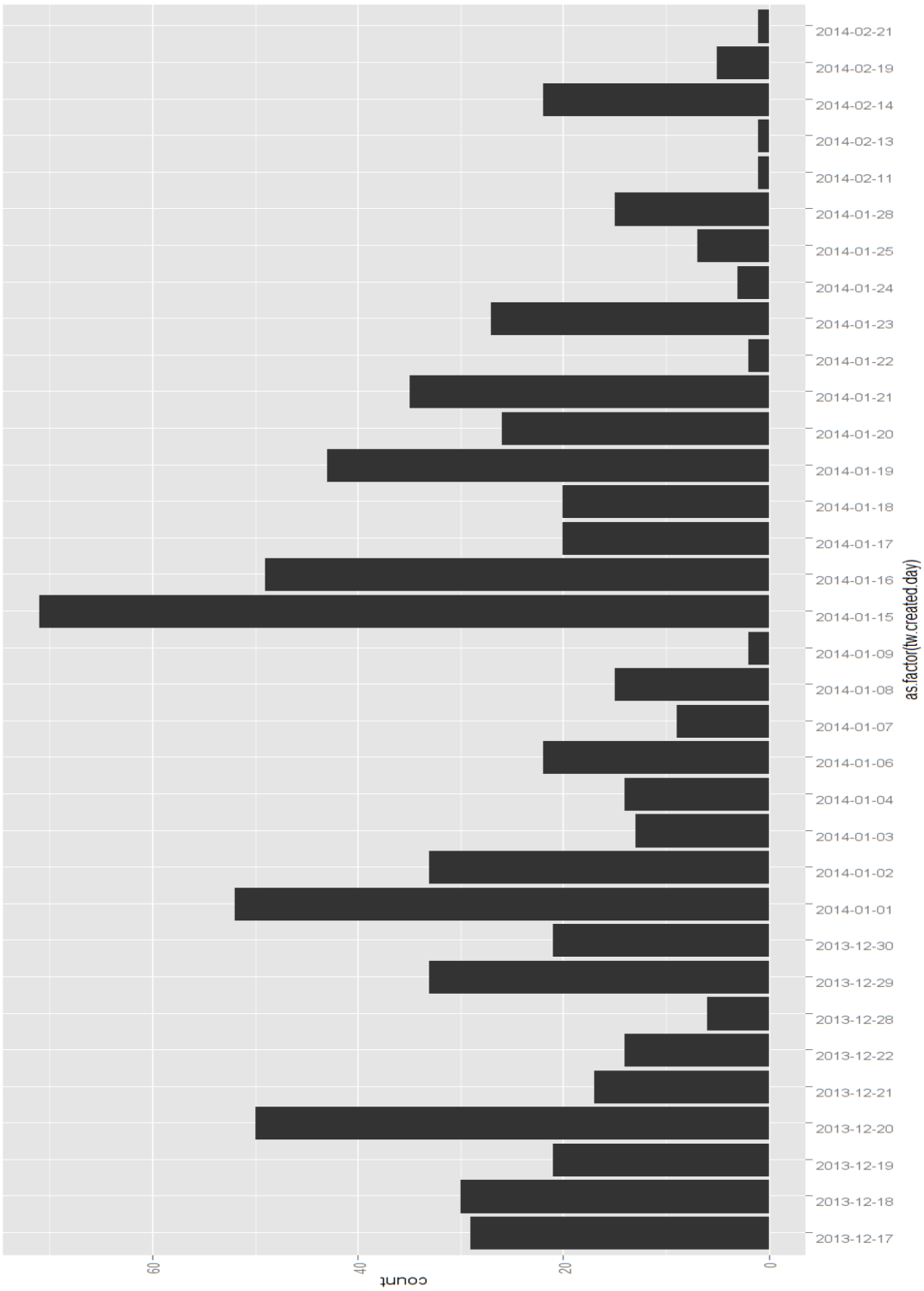


Figura 1.2: Distribuzione giornaliera dei tweets dell'utente AnalyticsInnova

L'operazione provoca, come è comprensibile, un sovraccarico computazionale, ma, per ovviare in parte a questo problema, la funzione divide il dataset di testi ordinati temporalmente in sottoinsiemi di cardinalità prefissata, aggiungendo dal secondo subset un numero dato di tweets del precedente: questo perché si suppone che i retweets abbiano un tempo di vita limitato. Calcola poi la matrice di dissimilarità basata sulla distanza di Levenshtein per ogni subset e raggruppa i tweets attraverso un algoritmo di clusterizzazione gerarchica con metodo "complete". In seguito, i tweets appartenenti allo stesso cluster vengono rimpiazzati dal tweet più vecchio e infine, attraverso un conteggio, vengono ordinati i retweet più frequenti. Vogliamo chiarire che la distanza utilizzata non è ovviamente l'unica, ma nel nostro caso essa funziona egregiamente. Comprendiamo, perciò, che una limitazione di queste operazioni consiste nel fatto che alcuni dei tweet principali, da cui sono partiti i retweet, non siano stati raccolti: per ovviare al problema, prendiamo i tweet più vecchi per ogni cluster. La funzione richiede i seguenti parametri:
- testo e' il vettore di tweets;
- S rappresenta la cardinalità di ogni sottoinsieme;
- L il numero di tweets da accorpare ad ogni sottoinsieme (escluso il primo);
- hclust.dist l'altezza corrispondete al taglio dell'albero;
- hclust.method il metodo di clustering;
- showTopN il numero di retweets da mostrare come output.

Data la mole di dati, la funzione richiede molto tempo per ottenere i risultati dal dataset. Facciamola partire con il comando:

```
RTHound(tw.eng$text, showTopN=50)
```

Abbiamo scelto un valore di showTopN pari a 50 e non superiore in quanto era sufficiente cercare i retweets “predominanti”, cioè quelli che si ripetevano per un numero molto elevato di volte. Individuati e sostituiti i retweets appartenenti allo stesso cluster con il più vecchio, li eliminiamo attraverso il comando “unique”: siamo ora in possesso di 71852 tweets.

1.6 Document Term Matrix

Possiamo ora procedere con lo studio delle parole più frequenti: introduciamo la Document Term Matrix. La DTM non è altro che una matrice le cui righe corrispondono ai tweets e le colonne a ciascun termine presente in tutti i testi. La matrice contiene, quindi, i conteggi di ogni termine per ogni tweet.
Mostriamo una sezione di esempio presa dalla nostra DTM (tabella 1.4):

Docs	human	ignor	illeg	imagin	imor	impact	imposs	impress
280	0	0	0	0	0	0	0	0
281	0	0	0	0	0	0	0	0
282	0	0	0	1	0	0	0	0
283	0	0	0	0	0	0	0	0
284	0	0	0	0	0	0	0	0
285	0	0	0	0	0	0	0	0
286	1	0	0	0	0	0	0	0
287	0	0	0	0	0	0	0	0
288	0	0	0	0	0	0	0	0

Tabella 1.4: Porzione di DTM

Per la costruzione della DTM ci dobbiamo appoggiare alla libreria "tm" (Feinerer, K. Hornik, and D. Meye, 2008). Il comando "DocumentTermMatrix" richiede un argomento di tipo vector, occorre quindi creare il vettore di testi:

```
corpus = Corpus(VectorSource(tw.eng$norm))

dtm = DocumentTermMatrix(corpus,
          control=list(stemming=TRUE,stopwords=eng,
          minWordLength=2,removeNumbers=FALSE,
          removePunctuation=FALSE,
          bounds=list(local = c(1,Inf))))
dtm
```

La funzione chiede, oltre al vettore:
- se eliminare lo stemming, cioè il processo di riduzione della forma flessa di una parola alla sua forma radice (detta tema);
- le stopwords, ovvero quelle parole che sono ritenute poco significative poiché usate troppo frequentemente come ad esempio articoli, preposizioni e congiunzioni. Siccome abbiamo scelto di voler tenere solo la parte di dataset in lingua inglese, le stopwords sono quelle inglesi presenti nella libreria "tm" (Lista in appendice);
- il numero di lettere minime per poter considerare una parola.
  Nel nostro caso il numero minimo è pari a due;
- se rimuovere numeri e punteggiatura
  (quest'ultima è già effettuata dalla normalizzazione).

La funzione ha riconosciuto un totale di 31840 termini che costituiscono le colonne della matrice, mentre le sue righe già le conosciamo: sono infatti i tweets, quindi il numero è pari a 71852. Mostriamo i termini che si ripetono più di 5000 volte:

```
findFreqTerms(dtm,5000)

"scienc"  "scientist"    "learn"   "machin"  "mine"
"new"     "#bigdata"     "data"    "via"     "wwwurlwww"
```

La DTM ora introdotta non rappresenta solo un semplice modo per ricavare le parole di maggior presenza nel testo, ma vedremo poi come sfruttarne il funzionamento per costruire la struttura di alcune reti della fase successiva.

1.7 Analisi degli N-Grammi e assegnazione delle Labels

Dalle frequenze delle singole parole, passiamo ora alla ricerca del gruppo di parole più frequente: studiamo, pertanto, gli n-grammi. Un n-gramma è una sottosequenza di n elementi di una data sequenza perciò, secondo l'applicazione, gli elementi in questione possono essere fonemi, sillabe, lettere, parole, ecc. Nell'analisi, ci limitiamo a considerare i bigrammi e i trigrammi (gruppi di due e tre parole) poiché, trattandosi di tweets e pertanto di frasi brevi, n-grammi di dimensione maggiore non ci porterebbero a risultati più significativi.

Vogliamo inoltre cercare tweets che contengono termini, o gruppi di essi, riguardanti la Machine Learning e poi assegnare a questi delle specifiche etichette che chiameremo labels o tags. Distinguiamo quindi sei categorie di labels:

1) MLSOFTWARE, cioè quei tweets che contengono parole che si riferiscono a diversi software come “python", "php", "java", "sas", "weka", "mahout";
2) MLAPPAREAS, quei tweet riferiti agli ambiti di applicazione della M.L. quindi “psychology”, “robotics”, “engineering", "statistics” ecc.;
3) MLTECHNIQUES, le tecniche, come ad esempio “textmining”, “cloudcomputing”;
4) MLALGORITHMS, alcune tipologie di algoritmi di M.L. quali “randomforest", "neuralnetwork”, “regression", "decisiontree”;
5) MLBRANDS, per appunto i brands come “facebook”, “twitter”, “microsoft”;
6) MLTWABOUT, i tweets riguardanti lavoro, news e conferenze, quindi “job”, “conferences” e “news”.

Con il calcolo dei bigrammi eseguito nella parte non “classificata” del dataset, verifichiamo la presenza di ulteriori parole da poter aggiungere alle nostre categorie (o crearne nuove): ad esempio, “Reddit” è stata aggiunta in seguito alla categoria “MLBrands”. Li possiamo ricavare utilizzando la funzione “textcnt” della libreria “tau” (Buchta, Hornik, Feinerer, and Meyer, 2012); mostriamo solamente i bigrammi ottenuti inizialmente nella tabella 1.5.

```
bigrams = textcnt(tw.eng$norm, n = 2, method = "string",
          split = "[[:space:]]+")
bigrams = bigrams[order(bigrams, decreasing = TRUE)]
bigrams[1:50]
```

Bigrams	Cont.	Bigrams	Cont.
data science	18367	retail marketing	3244
data mining	17178	wwwurlwww local	3244
machine learning	14950	science #entrepreneurs	3243
wwwurlwww rt	13310	triggers wwwurlwww	3242
data scientist	10942	#entrepreneurs emotional	2731
a data	3850	wwwurlwww wwwurlwww	2525
big data	3729	wwwurlwww data	2300
wwwurlwww via	3697	scientist wwwurlwww	2256
social data	3393	wwwurlwww #bigdata	2121
emotional triggers	3302	learning wwwurlwww	1929
and niche	3245	new york	1804
local and	3245	science wwwurlwww	1780
marketing social	3245	the data	1639
niche or	3245	how to	1584
or retail	3244	of the	1563

Tabella 1.5: Bigrammi individuati inizialmente

I risultati ottenuti mostrano coppie di termini come “data science”, “data mining”, ”machine learning”, “data scientist”, “big data”: proviamo ad unirli per osservare quali altre combinazioni si possono ricavare e decidiamo inoltre di rimuovere le stopwords inglesi e quelle parole poco significative che si ripetono con gran frequenza:

```
eng=stopwords(kind = "en")
tw.eng$norm <- tolower(tw.eng$norm)
tw.eng$norm <- paste0(" ",tw.eng$norm," ")
tw.eng$norm=removeStopwords(tw.eng$norm, stopwords = eng)

tw.eng$norm<-gsub('data science''datascience',tw.eng$norm)
tw.eng$norm<-gsub('data mining','datamining',tw.eng$norm)
tw.eng$norm <- gsub('machine learning',
                    'machinelearning', tw.eng$norm)
tw.eng$norm <- gsub('data scientist',
                    'datascientist', tw.eng$norm)
tw.eng$norm <- gsub('big data','bigdata', tw.eng$norm)
tw.eng$norm <- gsub('new york','newyork', tw.eng$norm)
tw.eng$norm <- gsub('social media',
                    'socialmedia', tw.eng$norm)
tw.eng$norm<-gsub('open source','opensource', tw.eng$norm)
tw.eng$norm <- gsub('business intelligence',
                    'businessintelligence', tw.eng$norm)
tw.eng$norm <- gsub('artificial intelligence',
                    'artificialintelligence', tw.eng$norm)
tw.eng$norm <- gsub('e commerce','ecommerce',tw.eng$norm)
tw.eng$norm <- gsub('newyork city',
                    'newyorkcity', tw.eng$norm)
```

```
tw.eng$norm <- gsub('newyork times',
                    'newyorktimes', tw.eng$norm)
tw.eng$norm <- gsub('fire department',
                    'firedepartment', tw.eng$norm)
tw.eng$norm <- gsub('new year','newyear',tw.eng$norm)

tw.eng$norm <- gsub('wwwurlwww','',tw.eng$norm)
tw.eng$norm <- gsub('([[:blank:]]rt)||
                    (rt[[:blank:]])','',tw.eng$norm)
tw.eng$norm <- gsub('([[:blank:]]via)||
                    (via[[:blank:]])','',tw.eng$norm)
```

L'idea è quindi quella di assegnare dei tags del tipo "###nome_label" in base ad alcune parole chiavi che ricerchiamo nel testo. Vediamo, come esempio, i comandi per gli algoritmi di Machine Learning (in appendice sono presenti gli altri):

```
ml.algorithms.topics <- c("[[:blank:]]#?log([^[:blank:]]+)
     ?([[:blank:]]+)?#?reg([^[:blank:]]+)?[[:blank:]]",
     "[[:blank:]]#?random([^[:blank:]]+)?([[:blank:]]+)?
     #?forest([^[:blank:]]+)?[[:blank:]]",
     "[[:blank:]]#?neural([^[:blank:]]+)?([[:blank:]]+)?
     #?net([^[:blank:]]+)?[[:blank:]]",
     "[[:blank:]]#?naive([^[:blank:]]+)?([[:blank:]]+)?
     #?bayes([^[:blank:]]+)?[[:blank:]]",
     "[[:blank:]]#?svms?[[:blank:]]|[[:blank:]]
     #?support([^[:blank:]]+)?([[:blank:]]+)?
     #?vector([^[:blank:]]+)?[[:blank:]]",
     "[[:blank:]]#?regression[[:blank:]]",
     "[[:blank:]]#?decision([^[:blank:]]+)?([[:blank:]]+)?
     #?tree([^[:blank:]]+?)[[:blank:]]|[[:blank:]]#?
     cart[[:blank:]]")
```

Salviamo le parole chiavi da ricercare (con le regular expression che ci aiuteranno nella ricerca);

```
ml.algorithms.labels <- c("###LOGISTICREG",
     "###RANDOMFOREST", "###NEURALNETWORK",
     "###NAIVEBAYES", "###SVM", "###REGRESSION",
     "###DECISIONTREE")
```

mentre i tags vengono memorizzati in un altra variabile.

```
ml.algorithms.labels <- paste0(ml.algorithms.labels,"
     ###MLALGORITHMS")
```

```
for(i in seq(ml.algorithms.topics))
{
     filtro <- grep(ml.algorithms.topics[i],tw.eng$norm)
     tw.eng$norm[filtro] <- paste0(tw.eng$norm[filtro],"
               ",ml.algorithms.labels[i])
}
rm(i)
```

Quest'ultima serie di comandi riguarda la ricerca e l'assegnazione delle labels e viene inoltre aggiunta la “super label” che ne rappresenta la categoria. Ripetute le operazioni per le sei categorie, vediamo quanti tweets abbiamo classificato con questo metodo:

```
length(grep("###",tw.eng$norm))
filtro.lab <- grep("###",tw.eng$norm)
round( length(filtro.lab)/nrow(tw.eng) ,3 )
```

Dei 71852 tweets in nostro possesso, ne abbiamo 29789 contenenti almeno una etichetta, cioè una percentuale di circa 41.5 sull'intero dataset. Mostriamo in tabella 1.6 quali sono le etichette più frequenti e quali gli n-grammi (bigrammi e trigrammi in tabella 1.7 e 1.8 rispettivamente) che si ottengono nella parte non classificata dei tweets.

```
testo<-tw.eng$norm
testo<-gsub(" ###MLALGORITHMS","",testo)
testo<-gsub(" ###MLAPPAREAS","",testo)
testo<-gsub(" ###MLTWEETABOUT","",testo)
testo<-gsub(" ###MLTWTECHNIQUES","",testo)
testo<-gsub(" ###MLSOFTWARE","",testo)
testo<-gsub(" ###MLBRANDS","",testo)
uni_label=textcnt(testo, n = 1, method = "string",
          split = "[[:space:]]+")
uni_label = uni_label[order(uni_label, decreasing = TRUE)]
uni_label[grep("^###",names(uni_label))][1:10]

big <- textcnt(tw.eng$norm[-filtro.lab], n = 2, method =
     "string", split = "[[:space:]]+")
big = big[order(big, decreasing = TRUE)]
big[1:20]

tri <- textcnt(tw.eng$norm[-filtro.lab], n = 3, method =
     "string", split = "[[:space:]]+")
tri = tri[order(tri, decreasing = TRUE)]
tri[1:20]
```

Non risultano altri termini da aggiungere.

Labels	Cont.	Labels	Cont.
###mljobs	5628	###google	1649
###marketing	3968	###mlnews	1411
###facebook	2917	###statistics	1295
###mlcourses	2676	###medicine	1270
###nsa	1998	###python	1251

Tabella 1.6: Labels più frequenti

Bigrams	Cont.	Bigrams	Cont.
datascience team	636	datamining talent	318
like datascientist	539	#bigdata #bigdata	315
cause effect	448	datamining #influence	315
explore cause	440	scarce datamining	313
effect like	434	datamining @klout	296
using datamining	412	soft skills	293
fight fires	354	datamining fight	289
newyorkcity using	349	lessons learned	284
giving datamining	324	build successful	281
make scarce	319	datascience central	279

Tabella 1.7: Bigrammi individuati nella parte non classificata

Trigrams	Cont.	Trigrams	Cont.
explore cause effect	437	soft skills matter	275
cause effect like	434	successful datascience team	267
effect like datascientist	413	build successful datascience	264
newyorkcity using datamining	343	machinelearning destruction mysterious	262
giving datamining #influence	315	data privacy machinelearning	261
make scarce datamining	313	datascience boss sixth	261
scarce datamining talent	313	trust datascience boss	259
datamining fight fires	285	destruction mysterious humanity	256
using datamining fight	284	privacy machinelearning destruction	256
boss sixth sense	275	skills matter datascience	240

Tabella 1.8: Trigrammi individuati nella parte non classificata

Pur sapendo che il nostro dataset ha ancora retweets che ne influenzano i risultati, ci riteniamo comunque soddisfatti: le labels precedentemente assegnate ci serviranno per creare interessanti reti nella prossima fase.

1.8 Analisi degli Hashtag

Terminiamo la nostra prima fase di analisi con lo studio degli hashtag: nello specifico, vogliamo mostrare un semplice modo per ricavarli dal testo e, analogamente a quanto fatto con i tweets e l'utente più attivo, ne osserviamo la distribuzione temporale. Gli hashtag sono un tipo di tag utilizzato in alcuni social network per creare delle etichette. Essi sono formati da parole (o combinazioni di parole concatenate) inserite nei commenti precedute dal simbolo "#" (cancelletto); vengono chiamati così poiché derivano dall'inglese hash e tag (etichetta). Gli hashtag meritano una particolare attenzione in quanto, specialmente su Twitter, vengono usati principalmente come strumenti per permettere agli utenti del web di trovare più facilmente un messaggio collegato ad un argomento e partecipare alla discussione, ma anche per incoraggiare a partecipare alla discussione su un argomento indicandolo come interessante. Sappiamo, quindi, che gli hashtag sono dei termini preceduti dal simbolo "#": in una nuova variabile salviamo il testo dei tweets e rimuoviamo le label; successivamente, utilizzando la funzione "grep", andiamo a cercare ogni tweet che contiene questa combinazione.

Salviamo in una nuova variabile il testo dei tweets, rimuoviamo le labels e cerchiamo gli hashtag:

```
algorithms<-gsub(" ###MLALGORITHMS",
     "",ml.algorithms.labels)
appareas=gsub(" ###MLAPPAREAS","",ml.appareas.labels)
brands=gsub(" ###MLBRANDS","",ml.brands.labels)
techniques=gsub(" ###MLTWTECHNIQUES",
     "",ml.techniques.labels)
about=gsub(" ###MLTWEETABOUT","",ml.twabout.labels)
software=gsub(" ###MLSOFTWARE","",ml.software.labels)
software=gsub(" #IBM","",software)
software=gsub(" #APACHE","",software)
testo<-tw.eng$norm[grep("###",tw.eng$norm)]
testo<-gsub(" ###MLALGORITHMS","",testo)
testo<-gsub(" ###MLAPPAREAS","",testo)
testo<-gsub(" ###MLTWEETABOUT","",testo)
testo<-gsub(" ###MLTWTECHNIQUES","",testo)
testo<-gsub(" ###MLSOFTWARE","",testo)
testo<-gsub(" ###MLBRANDS","",testo)
for(i in seq(software))
     { testo=gsub(software[i],"",testo) }
for(i in seq(algorithms))
     { testo=gsub(algorithms[i],"",testo) }
for(i in seq(brands))
     { testo=gsub(brands[i],"",testo) }
```

```
for(i in seq(appareas))
     { testo=gsub(appareas[i],"",testo) }
for(i in seq(techniques))
     { testo=gsub(techniques[i],"",testo) }
for(i in seq(about))
     { testo=gsub(about[i],"",testo) }
rm(i)

hashtag=grep("#\\w+",testo)
```

Possiamo ora ricavarne la distribuzione giornaliera (figura 1.3):

```
ggplot(data.frame(tw.eng$screenName[hashtag],
     tw.eng$created.day[hashtag]),aes(as.factor
     (tw.eng$created.day[hashtag])))+
     geom_bar(stat="bin",binwidth=1,drop=T)+
     theme(axis.text.x=element_text(angle=-90,size=10))
```

Abbiamo poi creato una funzione per ottenere tutti gli hashtag contenuti nei tweets. Questa richiederà solamente il testo in cui cercarli:

```
funz.hashtag<-function(txt)
{
     hashtag=grep("#\\w+",txt)
     hash=as.list(1:length(hashtag))
     for(i in 1:length(hashtag))
     {
          tweet=txt[hashtag[i]]
          hash[[i]]=str_extract_all(tweet,"#\\w+")
     }
     hash=unlist(hash)
     return(hash)
}

hash=funz.hashtag(testo)
length(unique(hash))
sort(table(hash),decreasing=T)[1:20]
```

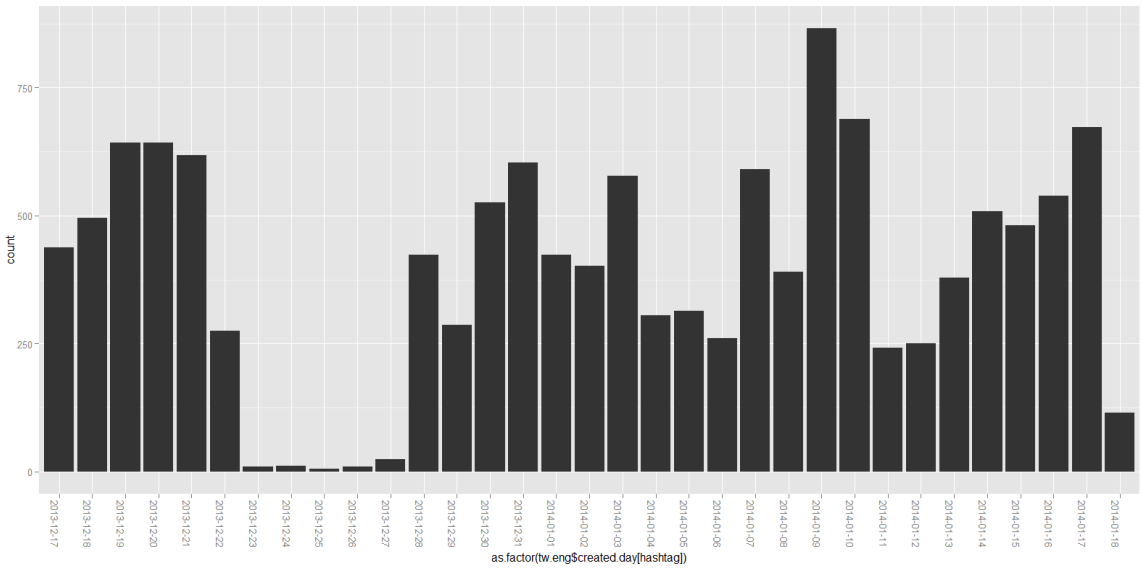


Figura 1.3: Distribuzione giornaliera dei tweets che contengono hashtag

Hashtag	Cont.	Hashtag	Cont.
#entrepreneurs	3244	#nsa	279
#bigdata	1356	#data	258
#jobs	935	#python	258
#job	833	#hadoop	255
#datascience	652	#hiring	254
#strataconf	576	#analytics	247
#machinelearning	391	#rstats	202
#datamining	328	#google	200
#facebook	320	#datascientist	192
#ml	295	#ibm5in5	177

Tabella 1.9: Conteggi dei 20 hashtag più usati

Si hanno in totale 2368 hashtag. Osservando, però, i primi 20 nella tabella 1.9, notiamo come alcuni che rappresenterebbero la stessa parola si differenziano per la flessione (esempio: #job e #jobs). Similmente a quanto avviene nella costruzione della DTM, uniamo gli hashtag con la stessa forma radice (tema) attraverso lo stemming: lo possiamo fare grazie alla funzione “wordStem” presente nella libreria “SnowballC”:

```
hash=wordStem(hash,language="english")
length(unique(hash))
```

Disponiamo ora di un totale di 2263 hashtag.

Capitolo 2

Analisi di Rete

In questo capitolo ci addentriamo negli studi della SNA: dopo una breve introduzione, parleremo di come una rete sociale può essere rappresentata attraverso matrici particolari e in forma di grafi, dove le matrici stesse ne costituiscono la struttura. Data l'importanza della teoria dei grafi sullo sviluppo della SNA, e quindi lo stretto legame che le unisce, ne esporremo i concetti, le teorie e le componenti.

A differenza di un qualsiasi servizio di posta elettronica, su Twitter non tutti i messaggi hanno un destinatario: un utente, infatti, per rispondere ad un altro deve necessariamente menzionarlo e il contenuto verrà inoltre letto anche dai loro rispettivi follower, ovvero le persone che per un qualche motivo e/o interesse vogliono seguirli. Vedremo, perciò, come ottenere gli utenti “citati”, quindi i destinatari, e i rispettivi “citanti” e ricaveremo la rete che ne rappresenterà la loro comunicazione. Effettueremo poi un'operazione analoga sfruttando però i risultati ottenuti dalla fase precedente: osserveremo quali utenti che, nel menzionare altri, hanno utilizzando parole a cui abbiamo assegnato le labels. Infine, vedremo come costruire le reti delle co-occorrenze tra le labels e, in seguito, tra gli hashtag più utilizzati attraverso il funzionamento della DTM.

Lo scopo quindi di questa fase finale sarà quello di mostrare come è possibile ottenere dal nostro dataset dei reticoli sia della tipologia One-Mode, ovvero quelle riferite ai “citanti-citati”, che alcuni casi di passaggio dalla Two-Mode alla One-Mode, le co-occorrenze, analizzandone i risultati.

Per effettuare le diverse operazioni ci serviremo prevalentemente di funzioni da noi create.

2.1 Analisi delle Reti Sociali

L'analisi delle reti sociali (Social Network Analysis - SNA) è una moderna metodologia di analisi delle relazioni sociali sviluppatasi a partire dai contributi di Jacob Levi Moreno, il fondatore della sociometria, scienza che analizza le relazioni interpersonali. Una rete sociale è costituita da un insieme (o da insiemi) di attori (actors) sociali e di relazioni (ties) definite tra tale insieme di attori: ad esempio, una rete sociale consiste di un qualsiasi gruppo di persone connesse tra loro da diversi legami sociali, che vanno dalla conoscenza casuale ai vincoli familiari. La SNA trova ora applicazione in diverse scienze sociali, come la sociologia, l'antropologia, la psicologia e l'economia, così come nel management, ed è stata utilmente impiegata nello studio di diversi fenomeni, come il commercio internazionale, la diffusione dell'informazione, lo studio delle istituzioni e il funzionamento delle organizzazioni. La SNA ha trovato di recente diverse applicazioni in campo fisico, biochimico, genetico e della computer science, pur mantenendo convenzionalmente l'appellativo "social", a ricordo della sua origine. Nella teoria delle reti sociali la società è vista e studiata come rete di relazioni, più o meno estese e strutturate. Il presupposto fondante è che ogni individuo (o attore) si relaziona con gli altri e questa sua interazione plasma e modifica il comportamento di entrambi. Lo scopo principale dell'analisi di network è quindi quello di individuare e analizzare tali legami tra gli individui (nodes) o gruppi di essi. Nel suo sviluppo la SNA ha fatto ampio uso di temi, concetti e strumenti di una branca della matematica nota come teoria dei grafi; andiamo ora a presentarne i concetti, i temi e le componenti principali che ci serviranno per le analisi delle nostre reti.

2.2 Teoria dei Grafi

Un grafo G è una struttura relazionale composta da un insieme finito di oggetti N, i cui elementi sono detti vertici (o nodi, punti), e da un insieme L di relazioni (geometricamente segmenti di retta o di curva) tra coppie di oggetti detti archi (o linee, spigoli). In L troviamo quindi le coppie ordinate e non di N che presentano un legame. La definizione di grafo ha, da un punto di vista strettamente matematico, delle notazioni e tra le notazioni possibili utilizzeremo la seguente: G(N,L). La figura 2.1 mostra un esempio con:

N={1, 2, ..., 6}

L={{1,2},{1,5},{2,3},
{2,5},{3,4},{4,5},{4,6}}

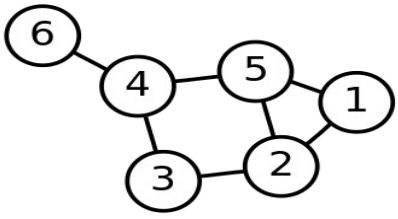


Figura 2.1: Grafo (non orientato) con 6 nodi e 7 linee

In un grafo, i nodi possono essere collegati direttamente ad un altro nodo oppure indirettamente attraverso una sequenza di linee: si definisce percorso W (walk) una sequenza di nodi e linee (non necessariamente tutti distinti) che descrive un tragitto all'interno del grafo. Un percorso con nodi e linee tutti distinti prende il nome di sentiero (path) P. Un percorso chiuso in cui ogni linea e ogni nodo sono inseriti in sequenza una ed una sola volta tranne il nodo di origine, si chiama ciclo (cycle) C. Un nodo è definito raggiungibile se esiste un percorso che lo colleghi agli altri nodi, indipendentemente dalla sua lunghezza (e quindi dagli intermediari che dovranno essere attraversati dal percorso). Un nodo isolato, al contrario, è definito come non raggiungibile e la sua distanza dagli altri è infinita. Un grafo si definisce connesso (figura 2.2.a) se esiste un percorso tra ogni coppia di nodi nel grafo (cioè non contiene nodi isolati). Se un grafo invece contiene anche un solo nodo isolato, si dice sconnesso (figura 2.2.b).

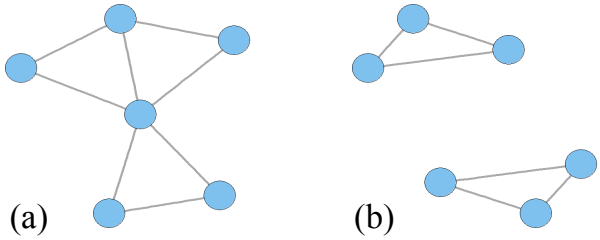


Figura 2.2: Grafo connesso (a) e grafo sconnesso (b)

Si distinguono due tipi di grafi: orientati o non orientati.

G(N,L) $N = \{n_1, n_2, ..., n_g\}$ $L = \{l_1, l_2, ...l_t\}$

In un grafo non orientato (figura 2.1) con g nodi e t linee che connettono coppie di nodi, ogni linea rappresenta una coppia non ordinata di nodi distinti: $l_k = (n_i,n_j)$. Essendo linee per coppie non ordinate di nodi, avremo che $l_k = (n_i,n_j) = (n_j,n_i)$ e pertanto tale linea verrà considerata una sola volta in L.

Un grafo orientato (figura 2.3.b) o digrafo Gd(N,L) è un insieme di nodi N e un insieme di archi orientati L. Ogni arco è una coppia ordinata di nodi distinti $l_k = <n_i,n_j>$. Ogni arco va da n_i (origine, mittente) a n_j (arrivo, destinatario). Un arco orientato è un arco caratterizzato da una direzione del legame, pertanto nel grafo, gli archi tra ogni coppia di nodi vengono contati separatamente, poiché possono avere valori differenti.

La linea che rappresenta il legame di un nodo con se stesso (n_i,n_i) viene detto loop o legame riflessivo. Un grafo che non ha loops e include una sola linea tra coppie di nodi è detto grafo semplice. In un grafo semplice non orientato, il numero massimo di possibili legami tra coppie di nodi sarà dato da g(g-1)/2; mentre per quello orientato, il numero massimo di archi possibili in L sarà pertanto dato da g(g-1).

Sostituendo i nodi con gli attori e le linee che li uniscono con relazioni, è possibile quindi rappresentare una rete sociale in forma di grafo. Distinguiamo pero' due principali tipi di rete: la One-Mode e la Two-Mode.

2.2.1 Reti One-Mode

Dato un set di attori N che contiene g attori, avremo: $N = \{n_1, n_2, ..., n_g\}$ dove n_i indica il generico attore i. Supponiamo che il set di attori N sia legato da una singola relazione di tipo dicotomico e direzionale. Data una generica coppia ordinata di nodi $<n_i, n_j>$, diremo che se esiste un legame tra i due nodi, n_i n_j, allora la coppia sarà un elemento di un insieme di coppie ordinate indicato con L. Gli elementi di L (che in tal caso rappresentano linee dirette o archi) sono indicati con l, pertanto avremo: $L = \{l_1, l_2,...l_k\}$. Se la relazione è dicotomica non direzionata, ciò significa che non si distingue tra i legami n_i n_j, n_j n_i, pertanto, una rete può essere descritta come un grafo G costituito da un insieme di nodi N e un insieme di linee L: G(N,L)

La struttura di un grafo, e quindi della rete che lo rappresenta, può essere riportata in una matrice che, nella tipologia One-Mode, viene chiamata matrice di adiacenza: è una matrice quadrata C di ordine g x g dove le unità sono gli attori. Questa matrice rappresenta le effettive relazioni o legami tra g attori e le singole celle mostrano, appunto, se coppie di individui sono legate da una comune relazione. L'assenza o la presenza di un legame tra coppie di nodi viene indicata con 0 o 1 rispettivamente. Nel caso di un grafo (o rete) non orientato (tabella 2.1.a) la matrice risulterà simmetrica poiché la coppia (n_i,n_j) è uguale a (n_j,n_i), mentre si tratterà di matrice non simmetrica (tabella 2.1.b) nel caso di un grafo orientato dove $(n_i,n_j) \neq (n_j,n_i)$. Sulla diagonale principale ci sono i Loops o legami riflessivi di un nodo n_i con se stesso.

	A	B	C	D
A	0	1	1	1
B	1	0	1	1
C	1	1	0	1
D	1	1	1	0

(a)

	A	B	C	D
A	0	1	1	0
B	1	0	1	0
C	0	0	0	1
D	0	1	1	0

(b)

Tabella 2.1: Matrici di adiacenza di un grafo non orientato (a) e di uno orientato (b)

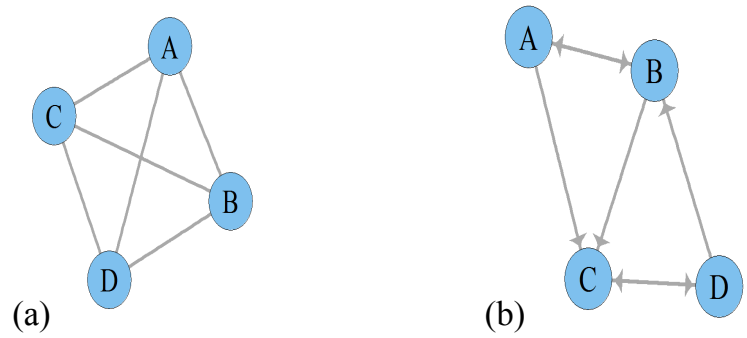


Figura 2.3: Rispettivi grafi delle matrici di adiacenza della tabella 2.1
(a) Non orientato (b) Orientato

2.2.2 Reti Two-Mode

Una rete può includere due set di attori: in tal caso si parla di rete Two-Mode. Indichiamo con N il primo set con g attori e con M il secondo set con h attori, ad esempio N = {n_1, n_2, ..., n_g} rappresenta individui e M = {m_1, m_2, ..., m_h} gli eventi. La matrice che si ottiene da questa tipologia di rete è spesso rettangolare e viene chiamata matrice di incidenza o di affiliazione: in riferimento all'esempio, nelle righe sono riportati gli individui (casi) e nelle colonne gli eventi (affiliazioni) cui i soggetti partecipano. Come in quella di adiacenza, nelle celle interne alla matrice possiamo trovare valori binari: 1 se l'individuo partecipa all'evento, 0 in caso contrario. Il calcolo dei marginali di riga e di colonna ci permette di sapere rispettivamente a quanti eventi ciascun individuo partecipa e quanti soggetti partecipano a ciascun evento.

E' possibile passare da una rete Two-Mode ad una One-Mode calcolando le due matrici di adiacenza che si ottengono da quella di incidenza. Sempre in riferimento all'esempio di prima abbiamo (tabella 2.2):

- A la matrice di affiliazione.
- $A*(A)^T$ sarà la matrice di adiacenza quadrata "caso per caso": questa matrice rappresenta le effettive relazioni o legami tra gli attori; filtrati dalle comuni relazioni o contatti. Le singole celle mostrano se coppie di individui sono legate da una comune affiliazione.
- $(A)^T*A$ sarà la matrice di adiacenza quadrata "affiliazione per affiliazione". Le celle di questa matrice indicano se le coppie di affiliazioni corrispondenti sono legate per mezzo di attori Comuni.

	A	B	C	D	E	
1	1	1	1	1	0	4
2	1	1	1	0	1	4
3	0	1	1	1	0	3
4	0	0	1	0	1	2
	2	3	4	2	2	

(a)

	1	2	3	4
1	-	3	3	1
2	3	-	2	2
3	3	2	-	1
4	1	2	1	-

(b)

	A	B	C	D	E
A	-	2	2	1	1
B	2	-	3	2	1
C	2	3	-	2	2
D	1	2	2	-	0
E	1	1	2	0	-

(c)

Tabella 2.2: Matrice di incidenza (a) e le due matrici di adiacenza da essa ottenute: $A * A^t$ (b) e $A^t * A$ (c)

Quando siamo interessati alla forza, alla frequenza o all'intensità dei legami tra coppie di attori, si parla di grafi segnati o pesati. Rappresentati in forma matriciale, a ciascuna cella viene assegnato il corrispondente valore: possono variare da 0, assenza di legame, ad n, essendo n il massimo valore assegnato al legame più forte; i marginali di riga o di colonna indicano la somma delle intensità dei legami.

2.3 Misure strutturali

Una rete (o reticolo) può essere analizzata secondo prospettive diverse; nel nostro lavoro studieremo la struttura complessiva del network e quindi osserveremo i legami nella loro conformazione strutturale: le misure strutturali offrono informazioni relative alle proprietà di un nodo all'interno del reticolo e alle proprietà della struttura del reticolo nel suo insieme. Ad esempio, una prima semplice misura riguarda la dimensione della rete, che è data dal numero dei nodi. La rete di dimensioni minori in assoluto è la diade, ossia quella costituita da due nodi. Presentiamo ora le metriche più importanti.

2.3.1 Densità e Inclusività

La densità in una rete descrive il livello generale dei legami fra i punti in un grafo: più sono numerosi i nodi direttamente collegati fra loro più un grafo è denso. La densità di un grafo si calcola come rapporto tra il numero delle linee di un grafo e il numero possibile di linee tra i nodi. Quindi:

- per grafi non orientati si ha $L \;/\; [\, N\,(N-1) / 2\,] \;=\; 2L / N\,(N-1)$
- per grafi orientati $L \;/\; N\,(N-1)$
- mentre per grafi pesati: $\Sigma w_k \;/\; N\,(N-1)$

dove L sono le linee, N è il numero di nodi in un grafo e w_k il valore delle k linee del grafo. I primi due indici variano da 0 a 1 (densità di un grafo completo quando tutti i nodi sono adiacenti uno all'altro).
L'inclusività è data dal rapporto tra il numero dei nodi raggiungibili, o connessi, (ottenuto anche dalla differenza tra i nodi totali con quelli isolati) con il numero totale dei nodi del grafo.

2.3.2 Centralità

Un concetto altrettanto importante, legato alla posizione strutturale ricoperta nel reticolo complessivo, è sicuramente quello di centralità del nodo. La centralità individua e definisce il posizionamento di un attore nel proprio ambiente rilevante in termini puramente relazionali, ovvero relativamente a tutti gli altri attori sociali con cui allaccia rapporti di scambio. L'idea di centralità si basa sul fatto che, a partire da un insieme di attori sociali e dato un insieme di relazioni fra di loro, è quasi sempre possibile dare agli attori un ordinamento gerarchico in base alla loro posizione nella struttura relazionale concreta che è possibile osservare. Si distinguono due tipi di centralità: quella locale, dove un punto è localmente centrale se ha un gran numero di connessioni con gli altri punti del suo ambiente circostante (basta quindi osservarne il grado), e quella globale, se un punto ha una posizione di importanza strategica nella struttura complessiva della rete.

La misura più semplice della centralità si ottiene dal calcolo dei gradi (degree), ovvero un nodo è centrale se ha un grado elevato, cioè se è adiacente a tutti gli altri nodi. Un soggetto con il grado più alto rappresenta metaforicamente il luogo nel gruppo dove "le cose accadono". In contrasto, i soggetti con un basso grado rappresentano le posizioni periferiche nella rete. Nel caso di un grafo non orientato il grado è dato dal numero di linee di connessione del nodo considerato, mentre in uno orientato il grado è dato dalla somma di una misura di entrata, indegree, che indica quanti nodi si connettono a quello osservato con una misura di uscita, outdegree, cioè a quanti nodi si connette il vertice considerato. E' possibile quindi (dove N è il numero totale dei nodi e d_i il degree dell'i-esimo nodo):

calcolare la somma totale dei gradi di tutti i nodi $S = \sum_i^N d_i$,

così come il grado medio dei nodi $\bar{d} \equiv \frac{S}{N}$

e la varianza del grado $S_d^2 \equiv \frac{\sum_{i=1}^{N} \left(d_i - \bar{d}\right)^2}{N}$ che sarà tanto più alta quanto

più varierà il grado di ciascun nodo. In altre parole indica se i nodi del grafo hanno un vicinato (nodi adiacenti) più o meno uguale. Come indice di centralità locale viene comunemente usato il valore standardizzato del grado:

$C_D(n_i) = d(n_i) / (N-1)$ ovvero si divide il degree per il numero di nodi totali.

I limiti di questa misura emergono in quei casi in cui non è tanto il numero di relazioni che si hanno, quanto la posizione di interscambio con gli altri nodi, quando cioè siamo interessati al potere e all'influenza di quel nodo, che può ricoprire un ruolo strategicamente più centrale. Una misura di centralità che cerca di superare questi limiti è la centralità basata sulle distanze con tutti gli altri nodi: cerca, quindi, di focalizzare la propria attenzione su quanto vicino un attore è agli altri. Un attore è perciò tanto più centrale quanto più è nella posizione di interagire velocemente con altri attori. Questo risulta allora molto efficace nel divulgare informazioni, proprio perché è quello che maggiormente nel gruppo ha contatti diretti, o indiretti ma brevi, con tutti gli altri. Questa misura, a differenza dei gradi, prende in considerazione anche i percorsi indiretti tra i nodi. Quanto minore è la somma delle distanze, tanto più centrale è l'attore, in quanto vicino a un gran numero di altri punti. Tale somma è data dal totale di riga (o di colonna) della matrice delle distanze. Per misurare questo indice di centralità basato sulla distanza (closeness) usiamo la funzione inversa delle distanze geodetiche (sentiero che collega il nodo i al nodo j):

con $C_C(n_i) = (N-1)\left[\sum_{j=1}^{N} d(n_i, n_j)\right]^{-1}$ si ha l'indice standardizzato.

Questo varia da 0 a 1: mentre il valore 1 indica che il nodo è centrale e si trova ad una distanza minima da tutti gli altri nodi, il valore 0 indica i nodi periferici (per lo 0 assoluto i nodi isolati).

Un altro concetto di centralità (betweenness) è quello che esprime la posizione intermedia di un individuo che può giocare la parte del mediatore o del guardiano. I nodi che si collocano in una posizione di intermediari (cioè localizzati sui percorsi che collegano coppie di nodi non adiacenti) possono esercitare un potere di controllo sul flusso delle informazioni. Il concetto di betweenness prende quindi in considerazione le geodetiche presenti nel grafo, contando quante volte ogni nodo si trova coinvolto in quelle tra altri attori.

2.4 Rete One-Mode: comunicazione tra gli utenti

Il nostro primo obiettivo è quello di mostrare una rete che rappresenti gli utenti più citati e i rispettivi citanti. Come spiegato nei paragrafi precedenti, il passo iniziale consiste nell'ottenere la matrice di adiacenza: è una matrice quadrata, con ordine pari al numero degli utenti, le quali celle conterranno il numero di volte in cui l'utente i ha citato l'utente j. Si tratta quindi di una rete di

tipo One-Mode, direzionale e con le frequenze come peso. In un tweet, un utente che vuole, per un certo motivo, citarne un altro riporta il suo nome preceduto dal carattere "@". Come fatto con gli hashtag, la funzione da noi costruita ricercherà nel testo tutte le parole successive alla "chiocciola" e inserirà i nomi in una particolare matrice detta "edgelist", cioè una matrice composta da due colonne che conterranno rispettivamente i citanti (gli "screenName" del dataset) e i citati (ricercati nel testo). Vedremo in seguito come sarà la libreria "igraph" a trasformarla in matrice di adiacenza.

```
funz.chi_cita_chi<-function(data)
{
     retweet=grep(pattern="@( |)(  |)\\w+",data$text)

     who_ret=as.list(1:length(retweet))
     who_post=as.list(1:length(retweet))

     for(i in 1:length(retweet))
     {
          tweet=data$text[retweet[i]]
          estrai=str_extract_all(tweet,"@( |)(  |)\\w+")

          who_post[[i]]=gsub("@","",unlist(estrai))
          who_ret[[i]]=rep(data$screenName[retweet[i]],
               length(who_post[[i]]))
     }

     who_ret=unlist(who_ret)
     who_post=unlist(who_post)
     Who=cbind(who_ret,who_post)
     return(Who)
}

Ccc=funz.chi_cita_chi(tw.eng)
```

Purtroppo, la mole di dati ci porta a dover considerare solo una parte della matrice ottenuta in quanto, con i mezzi che abbiamo, si hanno problemi a creare un grafo che la raffiguri. Con una seconda funzione da noi ideata, prendiamo solo alcuni degli utenti più citati; nello specifico questa richiede:
- num: numero di citati da considerare;
- taglio: vengono selezionati solo i citanti, dei rispettivi citati considerati, che hanno lasciato almeno una quantità di messaggi pari a questo numero (se taglio = zero, allora sono tutti presenti)
- matrice: la matrice "edgelist" che contiene tutti i citanti e i loro relativi citati.

```
funz.citato<-function(num,taglio,matrice)
{
     ordine=names(sort(table(matrice[,2]),decreasing=T))

     post=as.list(1:num)
     ret=as.list(1:num)
     for(i in 1:num)
     {
          posizione=which(matrice[,2]==ordine[i])
          post[[i]]=rep(ordine[i],length(posizione))
          ret[[i]]=matrice[posizione,1]
     }
     ret=unlist(ret)
     post=unlist(post)
     Cit=cbind(ret,post)

     who_ret=names(which(sort(table(Cit[,1]),
          decreasing=T)>taglio))
     who_post=names(sort(table(Cit[,2]),decreasing=T))
     for(i in 1:length(who_post))
     {
          list.post=as.list(1:length(who_ret))
          list.ret=as.list(1:length(who_ret))
          for(j in 1:length(who_ret))
          {
               dim=which((Cit[,1]==who_ret[j])&
                         (Cit[,2]==who_post[i]))
               list.ret[[j]]=rep(who_ret[j],length(dim))
               list.post[[j]]=rep(who_post[i],length(dim))
          }
          list.ret=unlist(list.ret)
          list.post=unlist(list.post)
          if(i==1)
          { G=cbind(list.ret,list.post) }
          else
          { G=rbind(G,cbind(list.ret,list.post)) }
     }
     return(G)
}

Cit<-funz.citato(10,0,Ccc)
```

Ottenuta la matrice dei citanti-citati (in questo caso i 10 più citati), passiamo alla sua rappresentazione attraverso il grafo: ci serviamo di un ulteriore nostra funzione (presente nella sua interezza nell'appendice), mostrandone il funzionamento e il suo output grafico.

```
funz.grafo<-function(matrice,citati,citatori)
```

La funzione necessita di:
- una matrice "edgelist", quindi di una matrice di due colonne;
- un valore intero positivo che indica il numero di utenti citati (ordinati dal più citato al meno) da evidenziare nel grafo;
- un valore come "citati", ma riferito ai citanti.

```
edge = graph.edgelist(matrice)
adj = get.adjacency(edge,sparse=T)
grafo = graph.adjacency(adj,mode="directed",weighted=T)
```

La libreria "igraph" ci permette, con dei semplici comandi, di ricavare il grafo dalla matrice "edgelist" passata alla funzione, poi, da questa, ne ricava la matrice di adiacenza e infine ne ricostruisce il grafo orientato e pesato. Sebbene questo possa sembrare un passaggio futile, la manipolazione e gestione di un grafo di tipo "graph.adjacency" ci è risultata molto più comoda.

I comandi restanti rendono il grafo di più facile comprensione: i legami con grandezza basata dal loro peso; dimensione dei nodi proporzionale al loro degree, mentre quella dei nomi degli utenti citati è proporzionale al numero di volte che sono stati citati; assegnazione dei colori (rosso citati, verde citanti) ecc. Nella figura 2.4 mostriamo quindi la rete "chi cita chi" evidenziando i 10 più citati e i 30 citanti più attivi in generale.

```
g.citati <- funz.grafo(Cit,10,30)

plot(g.citati, layout = layout.fruchterman.reingold)
```

Riportiamo alcune informazioni generali nella tabella 2.3:

Citanti più attivi	Cont.	Utenti più Citati	Cont.	Degree "in"	Valore	Degree "out"	Valore
AnalyticsInnova	674	scoopit	1093	HarvardBiz	702	KirkDBorne	5
dmu8750	212	HarvardBiz	773	kdnuggets	540	BigPupazzoBlu	4
jackwmson	84	kdnuggets	771	BigDataBorat	477	bootsielon	4
RedPressures	48	BigDataBorat	505	Coursera	321	sagebiel	4
dutchlight360	35	KirkDBorne	396	FastCompany	307	jackwmson	4
CarmenMcKell	30	Coursera	328	KirkDBorne	242	AnalyticsInnova	3
GarfieldFisher	29	FastCompany	316	John4man	221	tatyanas40	3
erfolgsinfo	21	klout	310	scoopit	215	data_nerd	3

Tabella 2.3: Alcune informazioni della rete "Chi Cita Chi"

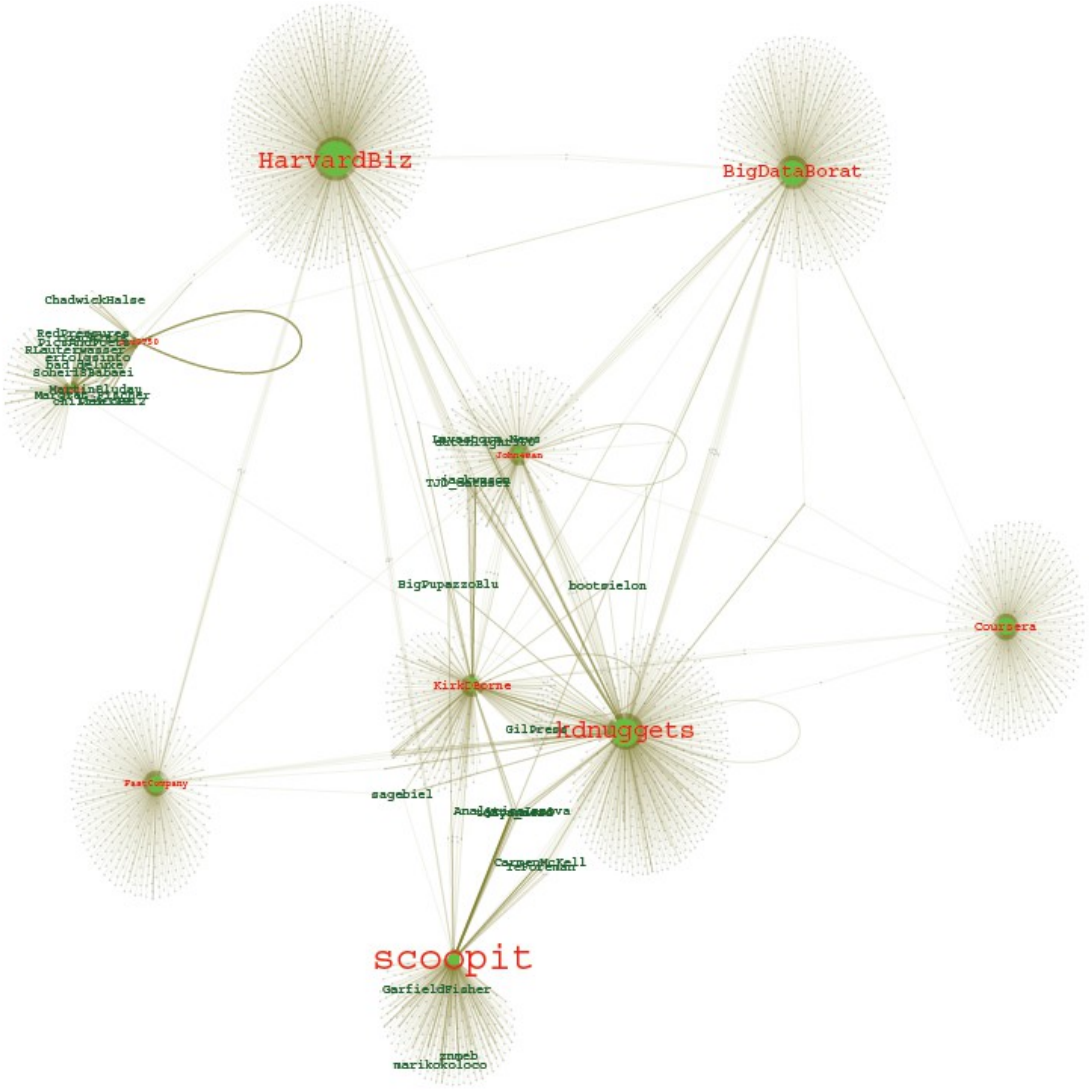


Figura 2.4: Grafo della rete "Chi Cita Chi"

La tabella ci aiuta a comprendere meglio il grafo ottenuto: “AnalyticsInnova” non solo è l'utente più attivo del dataset, ma è anche l'utente che più cita altri; è però da sottolineare che la sua attenzione è rivolta maggiormente verso l'utente “scoopit”. “scoopit” a sua volta è il più citato, ma gli utenti diversi a citarlo non sono molti e abbiamo infatti “HarvardBiz” come utente con maggiore indegree (citato da ben 702 utenti differenti). Avendo costruito la rete partendo dagli utenti più citati è normale che gli outdegree non abbiano un valore alto: in questo caso, è “KirkDBorne” ad avere outdegree maggiore con 5 diversi individui menzionati.

2.5 Rete One-Mode: chi cita chi secondo la label

La rete precedente ci dà tuttavia solo una visione generale riguardo la comunicazione tra gli utenti, cerchiamo pertanto di ottenere maggiori informazioni ricercando citante e citato secondo un termine specifico; in particolare vediamo quali reti otteniamo secondo le labels assegnate in precedenza. Capiamo, quindi, che anche questa serie di reticoli sono di tipo One-Mode (abbiamo solo un set di attori, cioè gli utenti), orientati e pesati.

Salviamo in delle variabili le labels senza la loro rispettiva super label:

```
about=gsub(" ###MLTWEETABOUT","",ml.twabout.labels)
appareas=gsub(" ###MLAPPAREAS","",ml.appareas.labels)
brands=gsub(" ###MLBRANDS","",ml.brands.labels)
techniques=gsub(" ###MLTWTECHNIQUES","",
          ml.techniques.labels)
algorithms<-gsub(" ###MLALGORITHMS","",
          ml.algorithms.labels)
software=gsub(" ###MLSOFTWARE","",ml.software.labels)
software=gsub(" #IBM","",software)
software=gsub(" #APACHE","",software)
```

Il procedimento che andiamo ora mostrare è simile per tutte le super label, quindi lo vediamo solo per la categoria “Algorithms” (in appendice le altre):

```
m.algor=as.list(1:length(algorithms))
g.algor=as.list(1:length(algorithms))
d.algor=as.list(1:length(algorithms))

for(i in seq(algorithms))
{
     m.algor[[i]]=funz.term(algorithms[i],tw.eng)
     g.algor[[i]]=funz.grafolabel(m.algor[[i]])
     d.algor[[i]]=degree(g.algor[[i]])
}
rm(i)
```

Attraverso un ciclo for, si effettuano delle operazioni concatenate: viene innanzitutto calcolata la matrice edgelist "chi cita chi", costruita grazie alla funzione "funz.term", dell'i-esima label; si ricava il grafo (la funzione è diversa dalla precedente, la riportiamo in appendice) dalla matrice e infine da questo si ricavano i degree. Il tutto viene riportato in tre diverse liste. La funzione "funz.term" è la seguente:

```
funz.term<-function(term,data)
{
      quali=grep(term,data$norm)
      Chi=funz.chi_cita_chi(data[quali,])
      return(Chi)
}
```

Vediamo cosa abbiamo ottenuto mostrando solamente "Algorithms" e i "Software" (tabelle 2.4 e 2.5 , alcuni grafi in figura 2.5, 2.6, 2.7, 2.8).

###LOGISTICREG	Citati	Datumbox 8	neuroconscience 3	es_arman 2	Kirubakumaresh 2
	Citatori	neuroconscience 3	earino 2	es_arman 2	kiki_tsaki 2
	degree	Datumbox 6	neuroconscience 6	es_arman 4	RidgwayGR 4
###SVM	Citati	brianabelson 6	cowlet 3	Kirubakumaresh 2	AndrewYNg 1
	Citatori	udmrzn 6	t1c1 3	TJO_datasci 2	alisdair 1
	degree	brianabelson 6	udmrzn 6	t1c1 3	cowlet 3
###NEURALNETWORK	Citati	Coursera 9	uoftnews 7	hackerdojo 3	HPC_Guru 3
	Citatori	m4x1milian 3	MoritaYasuaki 3	AsalBarak 2	BarcampsUS 2
	degree	Coursera 9	uoftnews 7	HPC_Guru 4	m4x1milian 3
###REGRESSION	Citati	Datumbox 8	neuroconscience 3	es_arman 2	Kirubakumaresh 2
	Citatori	neuroconscience 3	AnalyticsInnova 2	earino 2	es_arman 2
	degree	Datumbox 6	neuroconscience 6	es_arman 4	RidgwayGR 4
###NAIVEBAYES	Citati	Datumbox 5	dineshasanka 5	sqlsvruniverse 4	datumbox 2
	Citatori	dannyeuu 2	digitlk 2	gogula 2	lushanthan 2
	degree	Datumbox 5	sqlsvruniverse 5	dineshasanka 5	datumbox 2
###DECISIONTREE	Citati	IndeedEng 19	hyams 6	cowlet 3	BigMLcom 2
	Citatori	IndeedEng 4	JoshuaBaer 4	aficionado 3	5280BigData 2
	degree	IndeedEng 16	hyams 7	cowlet 3	JoshuaBaer 3

Tabella 2.4: Alcune informazioni della rete "Chi Cita Chi" secondo "Algorithms"

Dalla tabella, possiamo notare che i tweets riguardanti gli "Algorithms" non sono poi molti: dobbiamo comunque considerare che vengono solo selezionati i tweets che contengono almeno una citazione di un utente. Lo user "Datumbox" risulta il più menzionato soprattutto per le labels "Logisticreg",

“NaiveBayes” e “Regression”. In “Regression” e in “Logisticreg” abbiamo come citante più attivo “neuroconscience”.

###PYTHON	Citati	kdnuggets 67	peteskomoroch 60	YhatHQ 57	adam_rab 45
	Citatori	BigdataITJobs 45	rabihayaacoub 19	drago_carlo 8	jackw mson 8
	degree	kdnuggets 62	peteskomoroch 61	YhatHQ 55	GaelVaroquaux 45
###MAHOUT	Citati	ted_dunning 11	grapealope 9	mapr 9	wesmc 9
	Citatori	Ellen_Friedman 5	mapr 4	boorad 3	jakrigg 3
	degree	mapr 12	Ellen_Friedman 11	grapealope 10	ted_dunning 9
###SPSS	Citati	IronsideGroup 8	SeelBMAnalytics 4	ironsidegroup 2	p2people 2
	Citatori	IronsideGroup 6	NYDatagirl 2	p2pWebMobileIt 2	SeelBMAnalytics 2
	degree	IronsideGroup 8	SeelBMAnalytics 4	joe_kasz 2	NYDatagirl 2
###SQL	Citati	kdnuggets 55	01 18	findmjob 15	BigDataNetwork 3
	Citatori	rabihayaacoub 13	jackw mson 3	NorCalDotNet 3	anasrini 2
	degree	kdnuggets 54	01 16	gopivotal 4	BigDataBlogs 3
###SCALA	Citati	Scalding 22	TwitterOSS 20	BigDataSc 6	darrenjw 6
	Citatori	rabihayaacoub 6	BigDataSc 3	ktosopl 3	adeiso 2
	degree	TwitterOSS 21	Scalding 21	BigDataSc 9	darrenjw 6
###RUBY	Citati	honeybadgerapp 3	ifesdjeen 3	mxlearn 3	Althea415 2
	Citatori	joelgrus 4	edemkumodzi 2	kwasireports 2	phloxblog 2
	degree	joelgrus 4	mxlearn 3	honeybadgerapp 3	ifesdjeen 3
###SAS	Citati	adam_rab 71	IAPA_org_au 2	mounttalent 2	oreillymedia 2
	Citatori	BigdataITJobs 70	Davis_Bill 2	FayeMerrideth 2	GajendraTalent 2
	degree	sas_anz 2	Which50 2	IAPA_org_au 2	James_Enoch 2
###HADOOP	Citati	adam_rab 45	kdnuggets 33	BigDataBody 29	strataconf 24
	Citatori	BigdataITJobs 45	BigDataSpeaker 23	rabihayaacoub 22	jackw mson 12
	degree	kdnuggets 33	strataconf 24	cray_inc 19	insideBigData 18
###WEKA	Citati	kdnuggets 8	WekaMOOC 6	DataMiningAgent 4	scoopit 3
	Citatori	AnalyticsInnova 3	akila87 2	cosmico_gato 2	davidzehcnas 2
	degree	kdnuggets 8	WekaMOOC 6	DataMiningAgent 3	abifet 2
###RPROJECT	Citati	kdnuggets 55	KirkDBorne 32	hadleywickham 12	bcaffo 11
	Citatori	sagebiel 6	idigdata 5	mjcavaretta 5	BigDataHoops 4
	degree	kdnuggets 55	KirkDBorne 33	hadleywickham 15	jtleek 15

Tabella 2.5: Alcune informazioni della rete “Chi Cita Chi” secondo “Software”

Con i “Software” abbiamo una situazione diversa, infatti otteniamo sia reticoli di grande che di piccola dimensione. Fra gli utenti più menzionati, “kduggets” è quello che appare maggiormente, mentre per quanto riguarda i citanti, abbiamo “BigdataITJobs”. E' inoltre curioso il risultato ottenuto per “SAS”: l'utente “adam_rab” viene molto citato da “BigdataITJobs”, ma nella rete non ci sono nodi collegati a più di due.

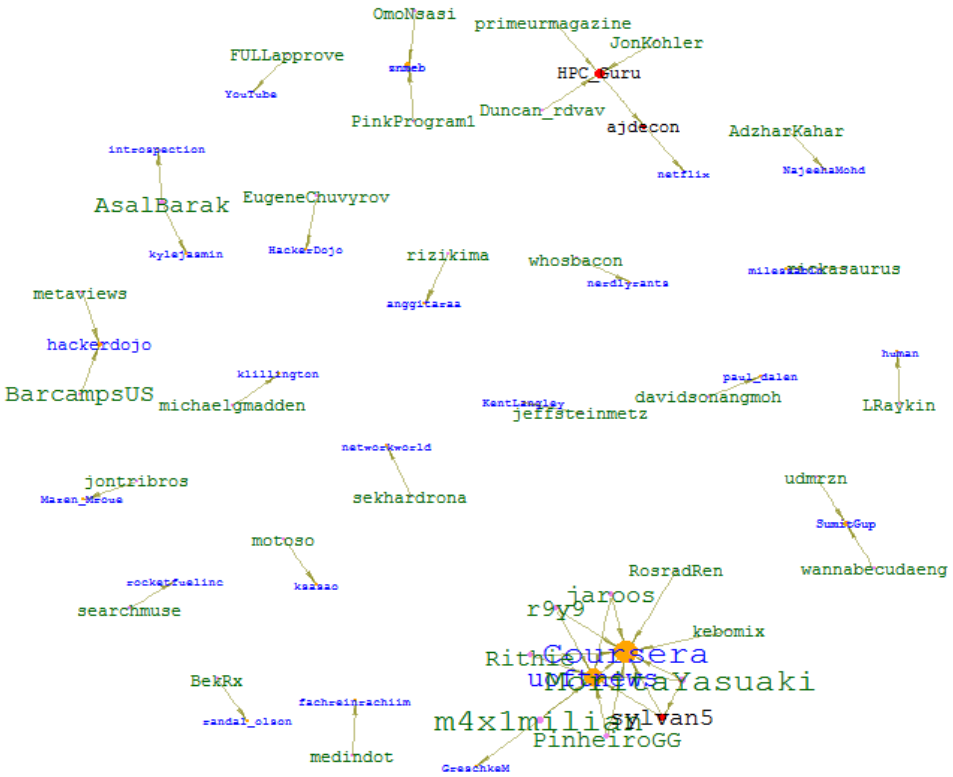


Figura 2.5: Grafo della rete "Chi Cita Chi" secondo "###neuralnetwork"

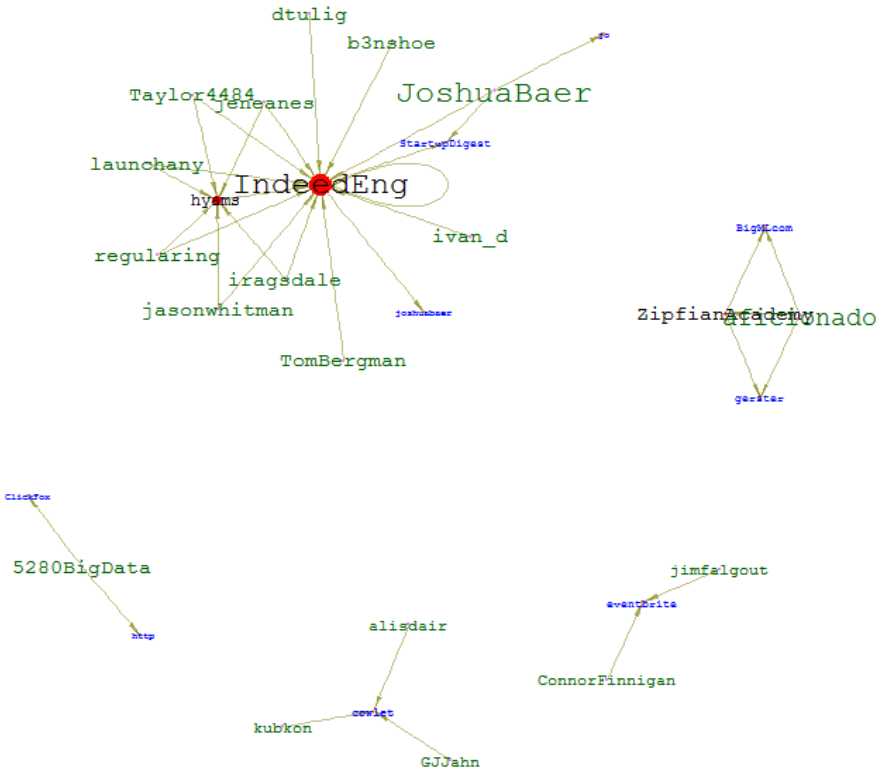


Figura 2.6: Grafo della rete "Chi Cita Chi" secondo "###decisiontree"

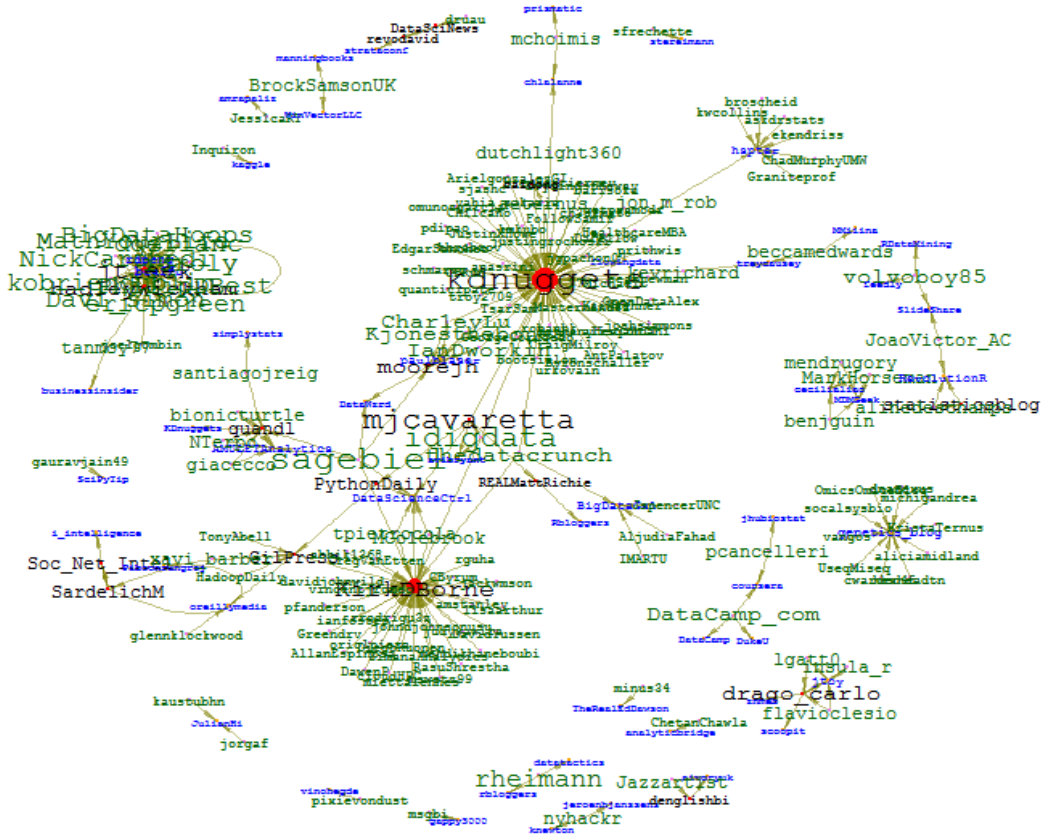


Figura 2.7: Grafo della rete “Chi Cita Chi” secondo “###rproject”

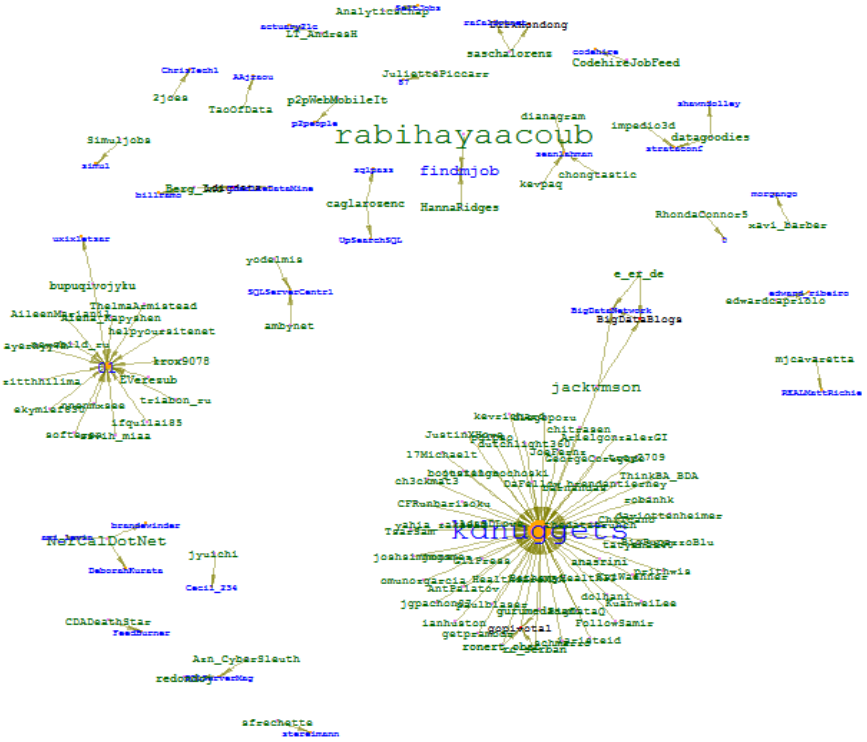


Figura 2.8: Grafo della rete “Chi Cita Chi” secondo “###sql”

2.6 Passaggio da Two-Mode a One-Mode: co-occorrenze labels

Col fine anche di presentare degli interessanti passaggi da reti di tipo Two-Mode a One-Mode, in questa fase, ne costruiremo diverse che rappresenteranno le co-occorenze delle labels contenute nei testi: per farlo, abbiamo pensato alla DTM come a una matrice di incidenza dove le righe sono i tweets e le colonne le etichette e, da questa, possiamo poi calcolare la matrice di adiacenza "label * label" che mostrerà, appunto, se coppie di labels si presentano nello stesso testo. Vediamo i comandi:

```
testo<-tw.eng$norm[grep("###",tw.eng$norm)]
testo<-gsub(" ###MLALGORITHMS","",testo)
testo<-gsub(" ###MLAPPAREAS","",testo)
testo<-gsub(" ###MLTWEETABOUT","",testo)
testo<-gsub(" ###MLTWTECHNIQUES","",testo)
testo<-gsub(" ###MLSOFTWARE","",testo)
testo<-gsub(" ###MLBRANDS","",testo)
#for(i in seq(software))
     { testo=gsub(software[i],"",testo) }
for(i in seq(algorithms))
     { testo=gsub(algorithms[i],"",testo) }
for(i in seq(brands))
     { testo=gsub(brands[i],"",testo) }
for(i in seq(appareas))
     { testo=gsub(appareas[i],"",testo) }
for(i in seq(techniques))
     { testo=gsub(techniques[i],"",testo) }
for(i in seq(about))
     { testo=gsub(about[i],"",testo) }
rm(i)
```

Vengono cercati i tweets di nostro interesse, cioè quelli che contengono tre "cancelletti", ed eliminate le super labels. I for invece servono a rimuovere i tags appartenenti a ogni categoria: in questo modo possiamo scegliere quali co-occorrenze si vogliono ricavare semplicemente "censurando" i for, ovvero rendendoli dei commenti. Tenendo "disattivato" un solo ciclo si hanno le co-occorrenze "interne" alla categoria scelta: ad esempio, nel caso mostrato avremo quelle interne ai software. Disattivando, invece, più cicli si avranno le co-occorrenze "esterne", cioè tra diverse categorie.

```
l.all<-as.list(seq(testo))
for(i in seq(testo))
{
     l.all[i]<-str_extract_all(testo[i],"###\\w+")
}
c=Corpus(VectorSource(l.all))
```

```
dtm.all=DocumentTermMatrix(c, control = list(
     stemming=F, minWordLength=2, removeNumbers=F,
     removePunctuation=F, bounds=list(local=c(1,Inf))))
rm(i,c)
```

Viene quindi costruita la DTM con solo i tags.

```
inc.all = t(as.matrix(dtm.all))
inc.all = inc.all[,colSums(inc.all)!=0]
adia.all = inc.all%*%t(inc.all)
diag(adia.all) = 0
g.adj.all <- graph.adjacency(adia.all,
          mode="undirected",weighted=T)
```

Si passa al calcolo della matrice di adiacenza “label*label”; si rimuovono i legami riflessivi e viene creato infine il grafo (non orientato e pesato).

```
egam = log(E(g.adj.all)$weight+2)/
          max(log(E(g.adj.all)$weight+2))
V(g.adj.all)$size = 10*degree(g.adj.all)/
          max(degree(g.adj.all))
V(g.adj.all)$label = V(g.adj.all)$name
V(g.adj.all)$label.family="mono"
V(g.adj.all)$label.cex = 0.8
V(g.adj.all)$label.color = hsv(0, 0.1, 0.5, 0.8)
for(i in seq(algorithms))
     { V(g.adj.all)$label.color[V(g.adj.all
          $name==tolower(algorithms[i])] = "blue" }
for(i in seq(appareas))
     { V(g.adj.all)$label.color[V(g.adj.all
          $name==tolower(appareas[i])] = "darkorange" }
for(i in seq(software))
     { V(g.adj.all)$label.color[V(g.adj.all
          $name==tolower(software[i])] = "darkgreen" }
for(i in seq(brands))
     { V(g.adj.all)$label.color[V(g.adj.all
          $name==tolower(brands[i])] = "red" }
for(i in seq(about))
     { V(g.adj.all)$label.color[V(g.adj.all
          $name==tolower(about[i])] = "brown" }
for(i in seq(techniques))
     { V(g.adj.all)$label.color[V(g.adj.all
          $name==tolower(techniques[i])] = "purple" }
V(g.adj.all)$frame.color = "darkgray"
V(g.adj.all)$color = "gainsboro"
E(g.adj.all)$width = egam
E(g.adj.all)$color = rgb(.5,.5,0,egam)
rm(i,egam)
```

Quest'ultima serie di comandi riguarda la parte estetica del grafo: si assegna un colore alle labels in base alla categoria a cui appartengono; la grandezza dei nodi è proporzionale al loro degree, mentre le linee hanno dimensione e opacità in base al peso ovvero secondo il numero di volte che co-occorrono le due coppie di nodi legati.

Dalle reti così ottenute, di cui mostreremo le più interessanti in seguito, possiamo trarre diverse informazioni riguardanti le discussioni, aventi come tema generale quello della Machine Learning, che avvengono tra gli utenti senza conoscerne l'intero testo dei tweets: possiamo per l'appunto speculare o fare considerazioni osservando solamente i grafi costruiti in quanto rappresentano una piccola sintesi del testo stesso, a maggior ragione considerando inoltre che i tweets sono messaggi il cui limite massimo è di 140 caratteri, quindi molto brevi. Ad esempio, considerando le co-occorrenze tra le labels "Stanford" e "Courses", si può ipotizzare che nel testo si stia parlando di corsi offerti dall'Università di Stanford. Vediamo allora quello che i grafi possono dirci iniziando dalle co-occorrenze interne.

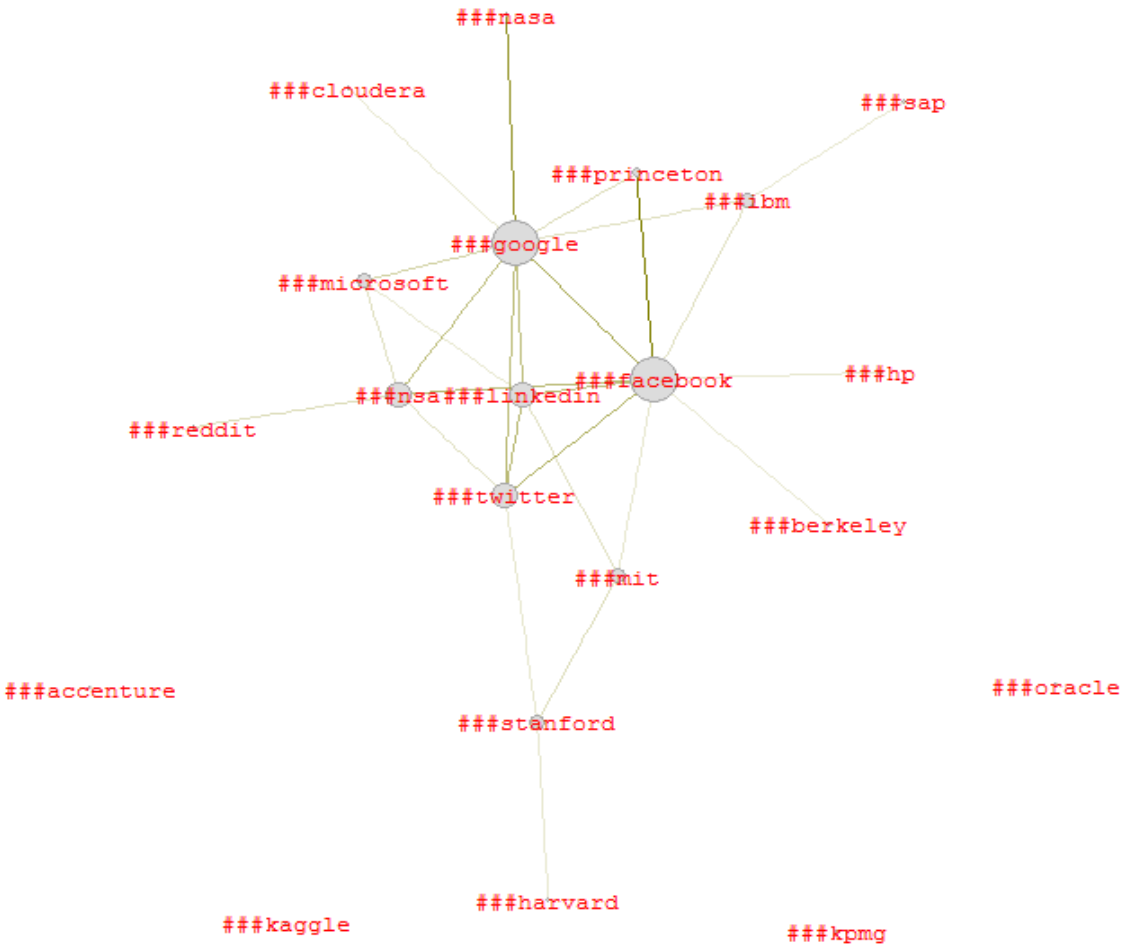


Figura 2.9: Grafo delle co-occorrenze interne a "Brands"

Il grafo in figura 2.9 mostra quali termini riferiti a “Brands” si presentano a coppie negli stessi tweets. Come nodi centrali (degree) vediamo “Google” e “Facebook”. “Google” e “NASA” sono unite da un forte legame, ad indicare probabilmente la collaborazione della “NASA” a offrire immagini satellitari per l'applicazione “Maps” di “Google” (e altre). Notiamo inoltre linee piuttosto marcate che uniscono i “Brands” che si riferiscono ai Social Media, quindi “Google”, “Facebook”, “LinkedIn” e “Twitter”. “Harvard” e “Stanford” fanno pensare soprattutto alle loro rinomate Università: possiamo già ipotizzare che queste avranno diverse co-occorenze con quelle labels assegnate a termini come “Jobs”, “Courses” e le varie “Appareas”, ovvero gli ambiti di applicazione. La stessa cosa la diciamo per “Oracle”: pur non avendo alcun legame con gli altri “Brands”, ce ne aspettiamo molti tra questa e i termini che riguardano i “Software”.

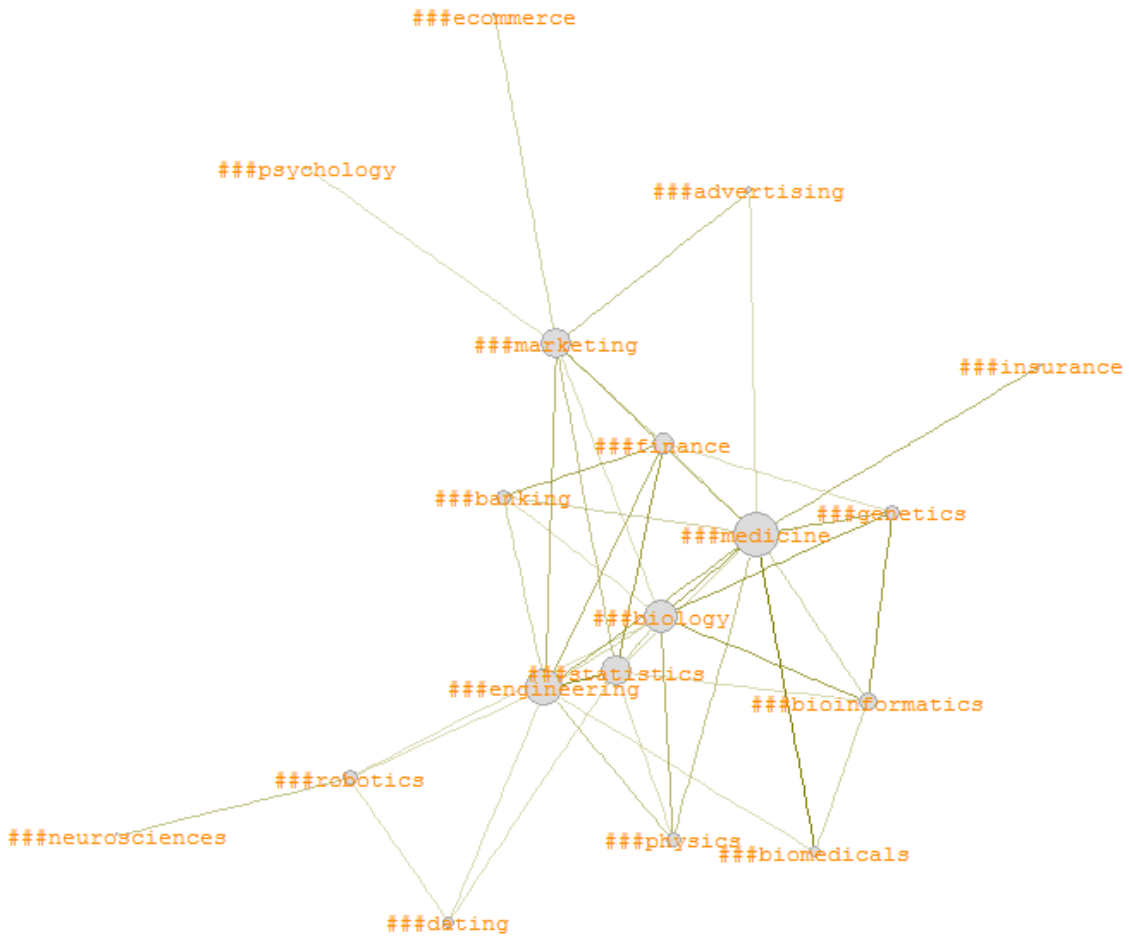


Figura 2.10: Grafo delle co-occorrenze interne a “Appareas”

Abbiamo ora il grafo (figura 2.10) delle co-occorenze interne alle "Application Areas" ("Appareas"). Come ci si poteva aspettare, vediamo unite quelle discipline accomunate dalle diverse metodologie e/o strumenti di cui fanno uso o facenti parte dei rami di una stessa scienza. Abbiamo "Statistics" particolarmente legata sia a "Engineering" e "Finance"; "Medicine" e "Biology" sono altrettanto unite da forti linee con "Genetics", "Bioinformatics" e "Biomedicals"; "Marketing" legata a "Advertising", "Psychology" e "E-Commerce"; infine "Robotics" con "Neurosciences".

Passiamo ora alle co-occorrenze "esterne": abbiamo deciso di mostrarle a coppie di categorie, perché altrimenti i grafi risultavano poco leggibili.

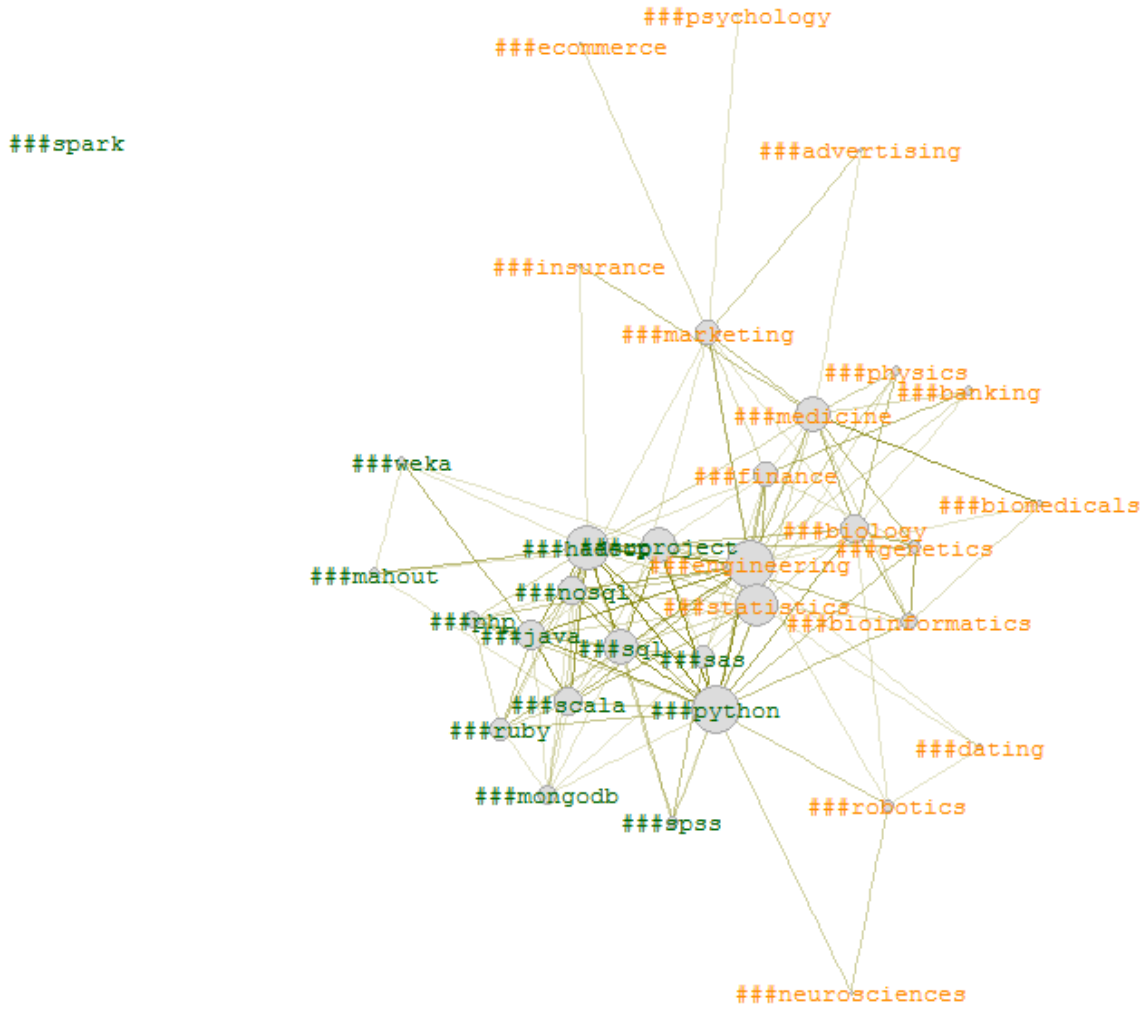


Figura 2.11: Grafo delle co-occorrenze tra "Appareas" e "Software"

Come prevedibile, il grafo tra le “Appareas” e i “Software” (figura 2.11) mostra chiaramente sia forti che numerose co-occorrenze tra i software e quegli ambiti che ne fanno maggior uso, quindi principalmente “Engineering” e “Statistics”; seguiti da “Robotics”, “Bioinformatics” e “Neurosciences” con ”Python”.

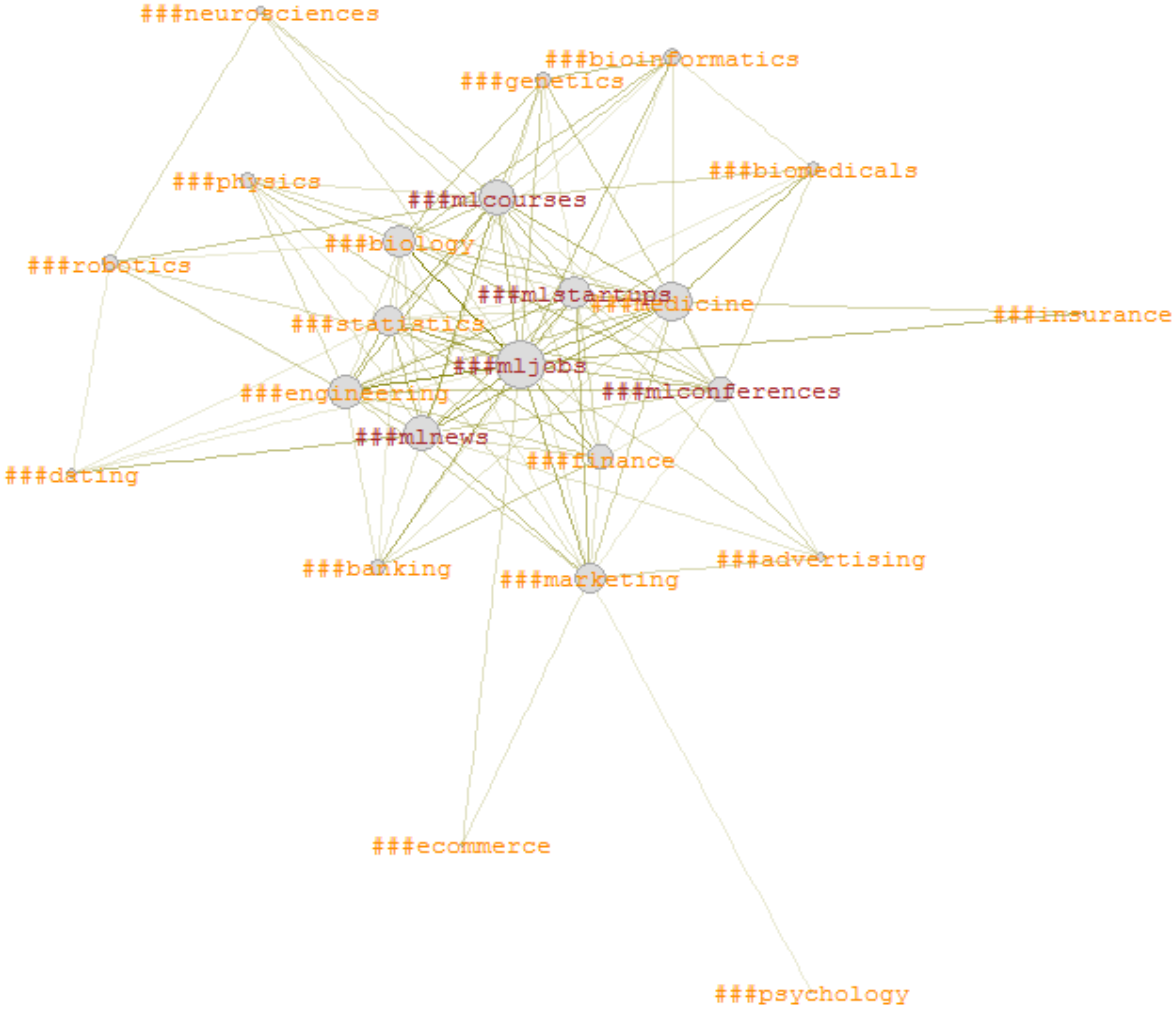


Figura 2.12: Grafo delle co-occorrenze tra “Appareas” e “TwAbout”

Il grafo in figura 2.12 fa riferimento a quei tweets il cui testo parla probabilmente degli sbocchi lavorativi, dei corsi, delle novità e delle conferenze secondo le diverse “Application Areas”: possiamo notare come le labels della categoria “TwAbout” diventino nodi centrali della rete ad indicare appunto quanto questi co-occorrino con i nodi dell'altra categoria.

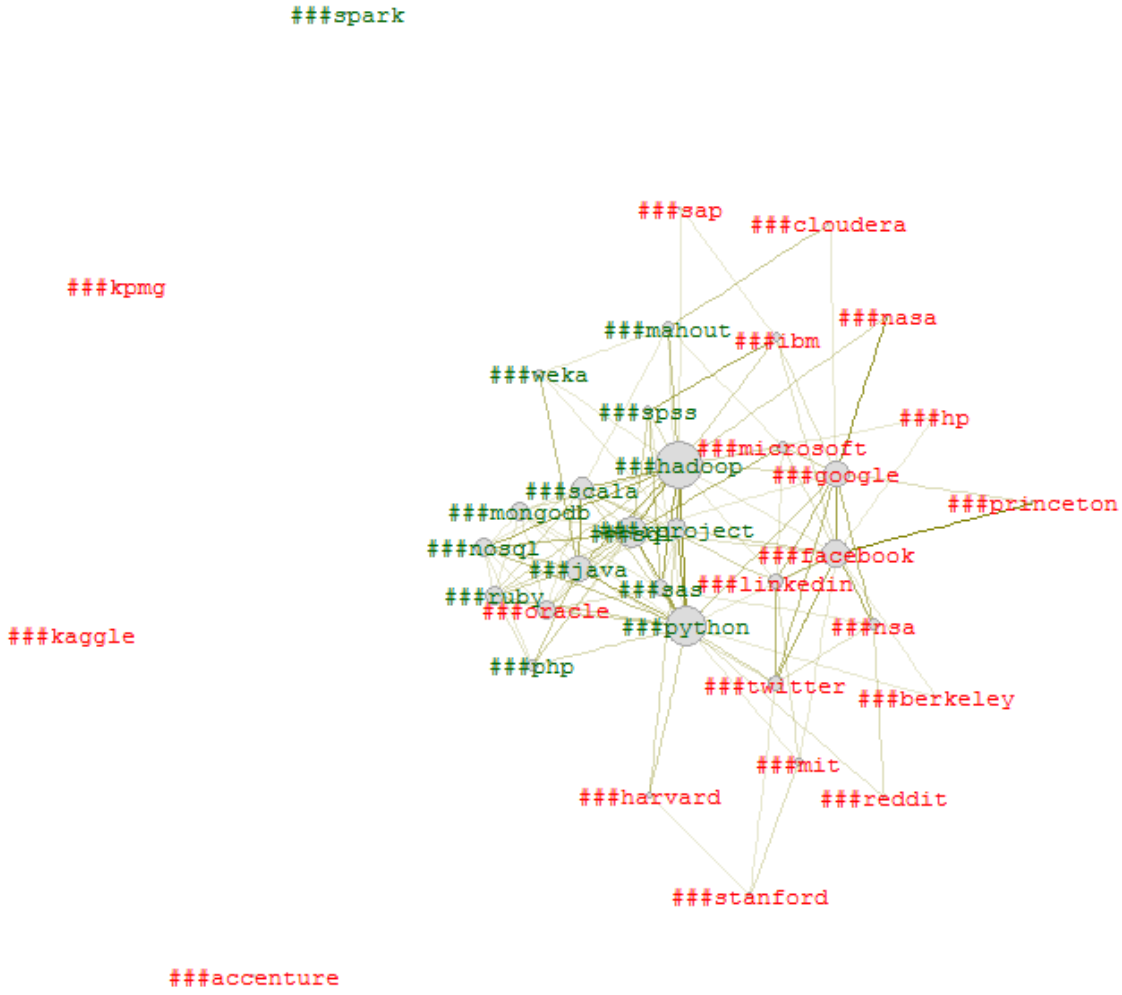


Figura 2.13: Grafo delle co-occorrenze tra “Brands” e “Software”

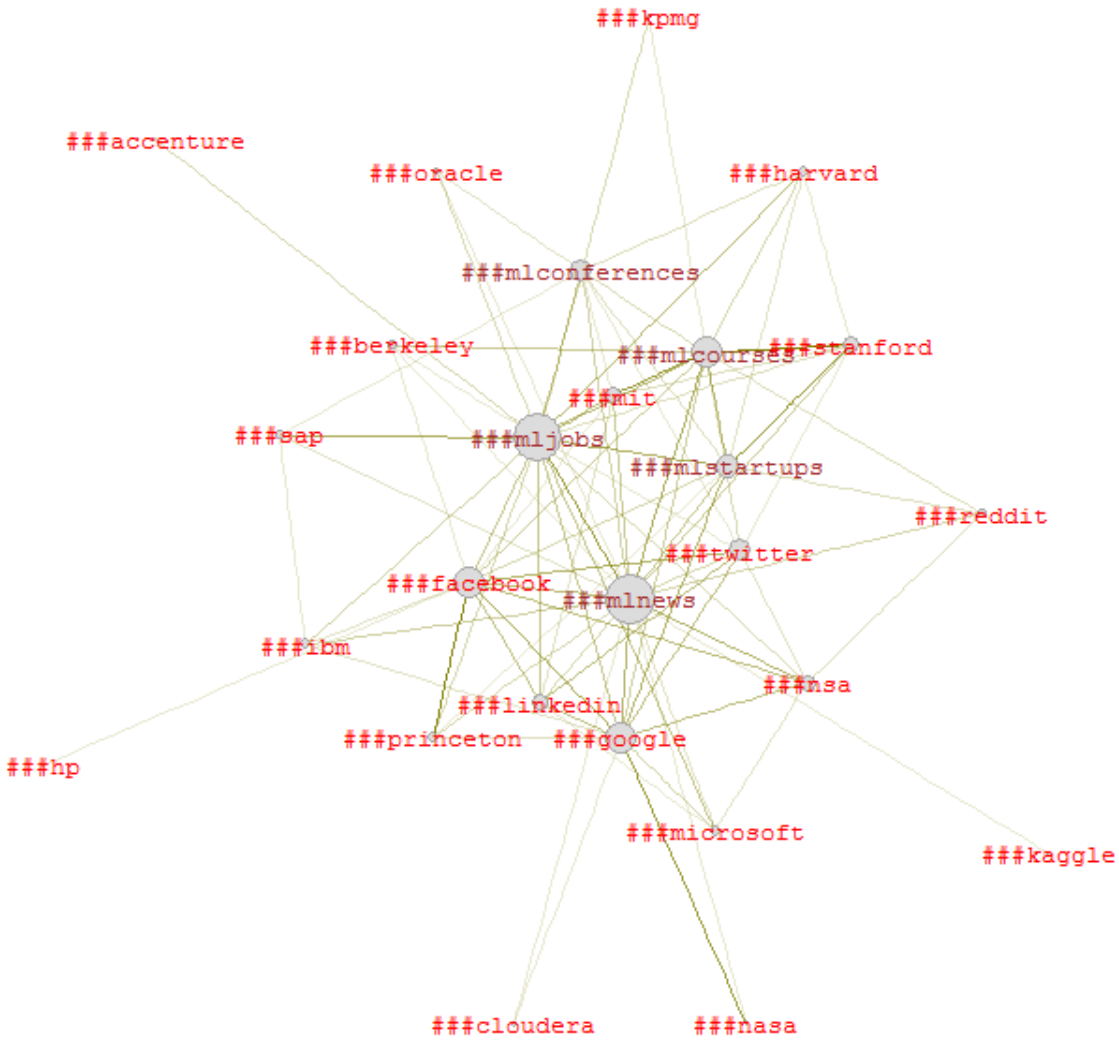


Figura 2.14: Grafo delle co-occorrenze tra "Brands" e "TwAbout"

I grafi confermano quanto detto precedentemente: nel primo (figura 2.13), "Oracle" è la labels che maggiormente co-occorre con quelle della categoria "Software"; nel secondo (figura 2.14), "Stanford" è fortemente legata a "Courses" e "Startups", mentre "Harvard" co-occorre particolarmente con "jobs".

2.7 Passaggio da Two-Mode a One-Mode: hashtag

Nello stesso modo in cui abbiamo ottenuto la rete delle co-occorrenze delle labels, ricaviamo ora quella degli hashtag più frequenti. Iniziamo quindi con ottenere la nostra nuova DTM utilizzando la seguente funzione:

```
funz.dtmhash<-function(data)
{
     hashtag=grep("#\\w+",data)

     tweets=data[hashtag]

     lista.hash=as.list(1:length(hashtag))

     for(i in 1:length(hashtag))
     {
          lista.hash[i]=str_extract_all(tweets[i],"#\\w+")
     }

     c=Corpus(VectorSource(lista.hash))

     dtm=DocumentTermMatrix(c, control=list(stemming=TRUE,
          minWordLength=2, removeNumbers=FALSE,
          removePunctuation=FALSE,
          bounds=list(local=c(1,Inf))))

     return(dtm)
}
```

Passiamo alla funzione il testo senza le labels:

```
dtmhash=funz.dtmhash(testo)
```

Possiamo ora sfruttare la DTM calcolata per prendere gli hashtag aventi una frequenza relativa maggiore dello 0.992 (vogliamo ottenere solo un numero esiguo, altrimenti si avrebbe un grafo poco leggibile e confusionario):

```
rows = rowSums(t(as.matrix(dtmhash)))

m=t(as.matrix(dtmhash))

lim = quantile(rows, probs=0.992)
good = m[rows > lim,]
```

Controlliamo ed eliminiamo la presenza di colonne nulle, cioè i tweets che non contengono gli hashtag selezionati, e calcoliamo la matrice di adiacenza (anche in questo caso non ci interessano i legami riflessivi, loops, quindi assegniamo alla diagonale principale degli zeri):

```
good = good[,colSums(good)!=0]

M = good %*% t(good)
diag(M) = 0
```

I comandi che seguono riguardano la parte della creazione del grafo e della sua presentabilità: assegniamo i colori dei nodi secondo una clusterizzazione con il metodo delle k-medie degli hashtag.

```
g.hash = graph.adjacency(M, weighted=TRUE,
          mode="undirected",add.rownames=TRUE)
egam <- (0.9)*(log(E(g.hash)$weight)+.4)/max(log(E(g.hash)
$weight)+.4)

sup <- quantile(rowSums(M),probs=0.9)

kmg = kmeans(M, centers=8)
gk = kmg$cluster
gbrew = c("red", brewer.pal(8, "Dark2"))
gpal = rgb2hsv(col2rgb(gbrew))
gcols = rep("", length(gk))
for (k in 1:8)
{
     gcols[gk == k] = hsv(gpal[1,k],
          gpal[2,k], gpal[3,k], alpha=0.5)
}

V(g.hash)$size = 6*degree(g.hash)/max(degree(g.hash))
V(g.hash)$label = V(g.hash)$name
V(g.hash)$label.family="mono"
V(g.hash)$degree = degree(g.hash)
V(g.hash)$label.cex = 0.7
V(g.hash)$label.color = hsv(0, 0.1, 0.5, 0.8)
V(g.hash)$label.color[rowSums(M)>sup] = "darkgreen"
V(g.hash)$frame.color = NA
V(g.hash)$color = gcols

E(g.hash)$width = egam
E(g.hash)$color = rgb(.5,.5,0,egam)
```

Mostriamo infine il grafo in figura 2.15.

```
plot(g.hash, layout=layout.fruchterman.reingold)
```

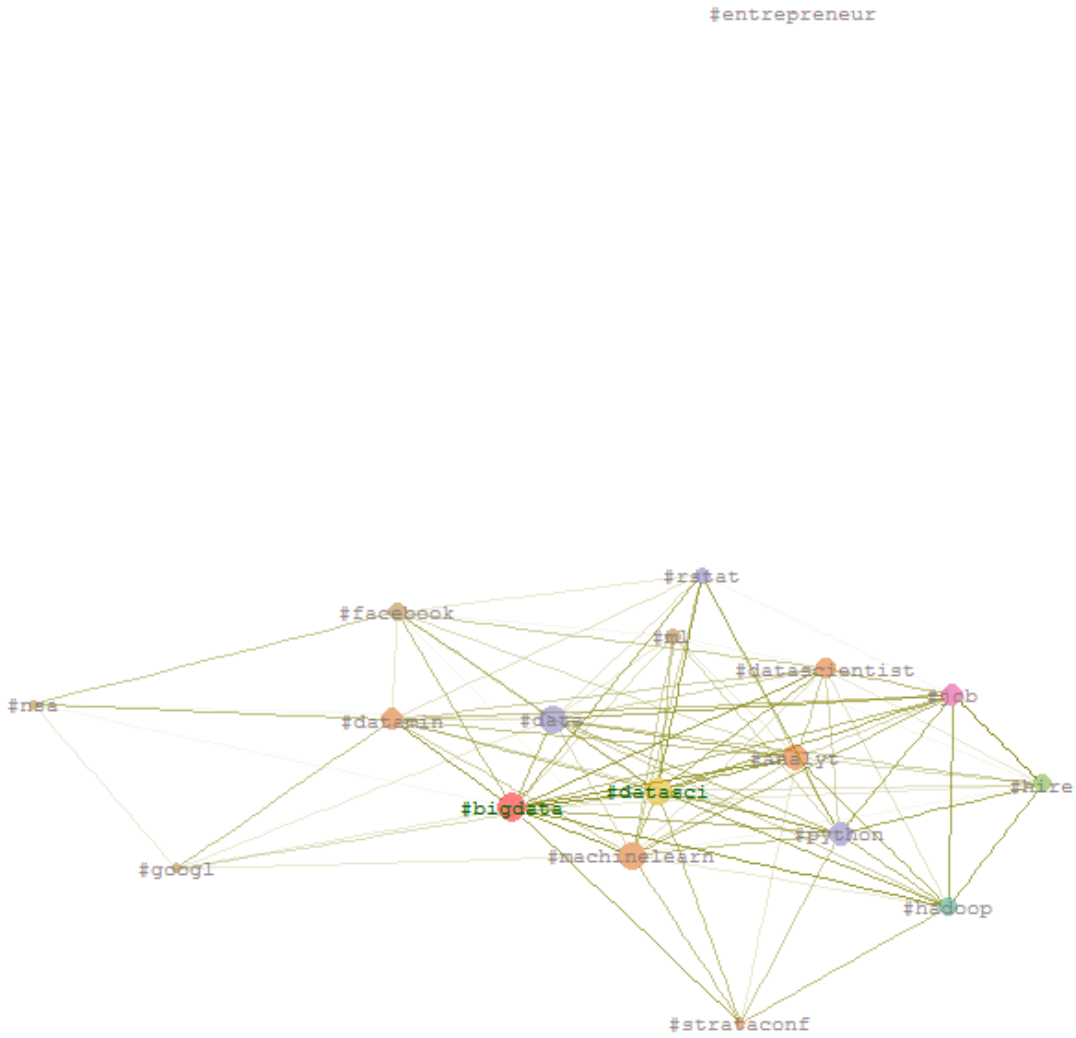


Figura 2.15: Grafo delle co-occorrenze tra gli hashtag più frequenti

Conclusioni

Siamo partiti dall'analisi di un dataset di tweets riguardanti la Machine Learning: dopo aver effettuato un veloce preprocessing dei dati in nostro possesso, abbiamo studiato la loro distribuzione giornaliera e, in seguito, pure quella dell'utente più attivo col fine di comprenderne la struttura e ricavarne le prime informazioni generali. Abbiamo, successivamente, affrontato il problema dei retweets in parte risolto servendoci della funzione “RTHound” per cercare almeno i più frequenti; li abbiamo poi rimossi con il comando “unique”. La prima fase termina con la ricerca di specifici termini riguardanti la Machine Learning a cui sono state assegnate delle etichette di sei diverse categorie.

Nella seconda fase del lavoro, siamo passati alla costruzione di reti per ricavare ulteriori informazioni dal nostro dataset. Ne abbiamo creata innanzitutto una che rappresentasse la comunicazione tra gli utenti: nello specifico volevamo osservare quali sono gli utenti che vengono menzionati e i loro rispettivi cintati. Da questa prima rete abbiamo ottenuto pure una curiosità riguardante l'utente "AnalyticsInnova": non solo è risultato essere lo user più attivo dalle analisi precedenti, ma la rete ci ha mostrato che egli è pure il citante più attivo e la sua attenzione è particolarmente rivolta verso l'utente "scoopit", al punto da renderlo il più citato.

Le labels precedentemente assegnate hanno poi rappresentato il fulcro su cui costruire le reti successive: i primi reticoli vogliono mostrare come gli utenti sono legati tra loro secondo una labels specifica; i secondi invece ne studiano le co-occorrenze dove abbiamo ottenuto risultati interessanti: abbiamo potuto fare speculazioni e considerazioni sull'argomento di discussione dei tweets osservando unicamente le co-occorrenze tra le labels delle diverse categorie. I tags della categoria "TwAbout" sono inoltre risultati essere quelli che più si presentano assieme ai tags delle altre "super labels".

Abbiamo infine ottenuto un altro risultato curioso dalla rete delle co-occorrenze degli hashtag più utilizzati dagli utenti: #entrepreneurs, pur essendo l'hashtag di maggior presenza nel dataset, non co-occorre con gli altri.

Appendice A Codici

Assegnazione Labels

```
### MACHINE LEARNING APPLICATION AREAS
ml.appareas.topics <- c(
      "[[:blank:]]([^[:blank:]]+)?psycho([^[:blank:]]+)?[[:blank:]]",
      "[[:blank:]]([^[:blank:]]+)?biolog([^[:blank:]]+)?[[:blank:]]",
      "[[:blank:]]([^[:blank:]]+)?biomed([^[:blank:]]+)?[[:blank:]]",
      "[[:blank:]]([^[:blank:]]+)?(medic|health)([^[:blank:]]+)?
            [[:blank:]]|[[:blank:]]#?(disease|vaccine)[[:blank:]]",
      "[[:blank:]]([^[:blank:]]+)?bioinf([^[:blank:]]+)?[[:blank:]]",
      "[[:blank:]]([^[:blank:]]+)?financ([^[:blank:]]+)?
            [[:blank:]]|[[:blank:]]#?trading[[:blank:]]",
      "[[:blank:]]([^[:blank:]]+)?bank([^[:blank:]]+)?[[:blank:]]",
      "[[:blank:]]([^[:blank:]]+)?(physics|physicians?)[[:blank:]]",
      "[[:blank:]]([^[:blank:]]+)?
            (engineering|engineers?)[[:blank:]]",
      "[[:blank:]]([^[:blank:]]+)?
            statistic([^[:blank:]]+)?[[:blank:]]",
      "[[:blank:]]([^[:blank:]]+)?(robot(ic)?s?)[[:blank:]]",
      "[[:blank:]]#?advertising[[:blank:]]",
      "[[:blank:]]#?neuro([[:blank:]]+)?
            #?scien([^[:blank:]]+)?[[:blank:]]|
                  [[:blank:]]#neuro[[:blank:]]",
      "[[:blank:]]([^[:blank:]]+)?
            (genet|genom)([^[:blank:]]+)?[[:blank:]]",
      "[[:blank:]]#?marketing[[:blank:]]",
      "[[:blank:]]#?dating[[:blank:]]",
      "[[:blank:]]#?ecommerce[[:blank:]]",
      "[[:blank:]]([^[:blank:]]+)?
            insurance([^[:blank:]]+)?[[:blank:]]")
ml.appareas.labels <- c(      "###PSYCHOLOGY",
                              "###BIOLOGY",
                              "###BIOMEDICALS",
                              "###MEDICINE",
                              "###BIOINFORMATICS",
                              "###FINANCE",
                              "###BANKING",
                              "###PHYSICS",
                              "###ENGINEERING",
                              "###STATISTICS",
                              "###ROBOTICS",
                              "###ADVERTISING",
                              "###NEUROSCIENCES",
                              "###GENETICS",
                              "###MARKETING",
                              "###DATING",
                              "###ECOMMERCE",
                              "###INSURANCE")
# add hierarchy labels
ml.appareas.labels <- paste0(ml.appareas.labels," ###MLAPPAREAS")
for(i in seq(ml.appareas.topics))
{
      filtro <- grep(ml.appareas.topics[i],tw.eng$norm)
      tw.eng$norm[filtro] <- paste0(tw.eng$norm[filtro],
            " ",ml.appareas.labels[i])
}
rm(i)
```

```
###### TWEET ABOUT
ml.twabout.topics <- c( "[[:blank:]]([^[:blank:]]+)?
      (#?job|career|hire|hiring|recruit)([^[:blank:]]+)?[[:blank:]]",
      "[[:blank:]]#?curriculum[[:blank:]]",
      "[[:blank:]]#?(conf|conferences?|meetings?)[[:blank:]]",
      "[[:blank:]]#?strataconf[[:blank:]]",
      "[[:blank:]]#?(news|interviews?)[[:blank:]]",
      "[[:blank:]]#?startups?[[:blank:]]",
      "[[:blank:]]#?moocs?[[:blank:]]",
      "[[:blank:]]([^[:blank:]]+)?cours([^[:blank:]]+)?[[:blank:]]")
ml.twabout.labels <- c(       "###MLJOBS",
                              "###MLJOBS",
                              "###MLCONFERENCES",
                              "###MLCONFERENCES",
                              "###MLNEWS",
                              "###MLSTARTUPS",
                              "###MLCOURSES",
                              "###MLCOURSES")
# add hierarchy labels
ml.twabout.labels <- paste0(ml.twabout.labels," ###MLTWEETABOUT")
for(i in seq(ml.twabout.topics))
{
      filtro <- grep(ml.twabout.topics[i],tw.eng$norm)
      tw.eng$norm[filtro] <- paste0(tw.eng$norm[filtro],"
",ml.twabout.labels[i])
}
rm(i)

###### MACHINE LEARNING TECNIQUES
ml.techniques.topics <- c(
      "[[:blank:]]#?sentiment([^[:blank:]]+)?([[:blank:]]+)?#?
            analysis([^[:blank:]]+)?[[:blank:]]",
      "[[:blank:]]#?topic([^[:blank:]]+)?([[:blank:]]+)?#?
            model([^[:blank:]]+)?[[:blank:]]",
      "[[:blank:]]#?text([^[:blank:]]+)?([[:blank:]]+)?#?
            (mining|analysis)([^[:blank:]]+)?[[:blank:]]",
      "[[:blank:]]#?nl(p|proc)[[:blank:]]",
      "[[:blank:]]#?info([^[:blank:]]+)?([[:blank:]]+)?#?
            graphic([^[:blank:]]+)?[[:blank:]]",
      "[[:blank:]]#?time([^[:blank:]]+)?([[:blank:]]+)?#?
            serie([^[:blank:]]+)?[[:blank:]]",
      "[[:blank:]]#?cloud([^[:blank:]]+)?([[:blank:]]+)?#?
            computing([^[:blank:]]+)?[[:blank:]]",
      "[[:blank:]]#?computer([^[:blank:]]+)?([[:blank:]]+)?#?
            vision([^[:blank:]]+)?[[:blank:]]",
      "[[:blank:]]#?recommend([^[:blank:]]+)?([[:blank:]]+)?#?
            (system|engine)([^[:blank:]]+)?[[:blank:]]",
      "[[:blank:]]#?seo[[:blank:]]")
ml.techniques.labels <- c(    "###TM_SENTIMENTANALYSIS",
                              "###TM_TOPICMODELS",
                              "###TEXTMINING",
                              "###TM_NLP",
                              "###INFOGRAPHICS",
                              "###TIMESERIES",
                              "###CLOUDCOMPUTING",
                              "###COMPUTERVISION",
                              "###RECOMMENDATIONSYSTEMS",
                              "###SEO")
# add hierarchy labels
ml.techniques.labels ← paste0(ml.techniques.labels,
            " ###MLTWTECHNIQUES")
```

```
for(i in seq(ml.techniques.topics))
{
      filtro <- grep(ml.techniques.topics[i],tw.eng$norm)
      tw.eng$norm[filtro] <- paste0(tw.eng$norm[filtro],
            " ",ml.techniques.labels[i])
}
rm(i)

###### MACHINE LEARNING SOFTWARE
ml.software.topics <- c(
      "[[:blank:]]#?(i?python|scypy|numpy)[[:blank:]]",
      "[[:blank:]]#?spss[[:blank:]]",
      "[[:blank:]]#?scala[[:blank:]]",
      "[[:blank:]]#?sas[[:blank:]]",
      "[[:blank:]]#?weka[[:blank:]]",
      "[[:blank:]]#?(apache)?mahout[[:blank:]]",
      "[[:blank:]]#?(apache.?#?spark)[[:blank:]]",
      "[[:blank:]]#?java[[:blank:]]",
      "[[:blank:]]#?sql[[:blank:]]",
      "[[:blank:]]#?nosql[[:blank:]]",
      "[[:blank:]]#?mongodb[[:blank:]]",
      "[[:blank:]]#?php[[:blank:]]",
      "[[:blank:]]#?ruby[[:blank:]]",
      "[[:blank:]]#?hadoop[[:blank:]]",
      "[[:blank:]](#r|#?rstats|#?rbloggers|#?
            rprogramming)[[:blank:]]")
ml.software.labels <- c(      "###PYTHON",
                              "###SPSS",
                              "###SCALA",
                              "###SAS",
                              "###WEKA",
                              "###MAHOUT",
                              "###SPARK",
                              "###JAVA",
                              "###SQL",
                              "###NOSQL",
                              "###MONGODB",
                              "###PHP",
                              "###RUBY",
                              "###HADOOP",
                              "###RPROJECT")
# add hierarchy labels
ml.software.labels <- paste0(ml.software.labels," ###MLSOFTWARE")
for(i in seq(ml.software.topics))
{
      filtro <- grep(ml.software.topics[i],tw.eng$norm)
      tw.eng$norm[filtro] <- paste0(tw.eng$norm[filtro],
            " ",ml.software.labels[i])
}
rm(i)

###### MACHINE LEARNING BRANDS
ml.brands.topics <- c( "[[:blank:]]#?ibm(5in5)?[[:blank:]]",
                       "[[:blank:]]#?oracle[[:blank:]]",
                       "[[:blank:]]#?nasa[[:blank:]]",
                       "[[:blank:]]#?google[[:blank:]]",
                       "[[:blank:]]#?(facebook|fb)[[:blank:]]",
                       "[[:blank:]]#?twitter[[:blank:]]",
                       "[[:blank:]]#?stanford[[:blank:]]",
                       "[[:blank:]]#?princeton[[:blank:]]",
                       "[[:blank:]]#?harvard[[:blank:]]",
```

```
                        "[[:blank:]]#?mit[[:blank:]]",
                        "[[:blank:]]#?cloudera[[:blank:]]",
                        "[[:blank:]]#?kaggle[[:blank:]]",
                        "[[:blank:]]#?accenture[[:blank:]]",
                        "[[:blank:]]#?kpmg[[:blank:]]",
                        "[[:blank:]]#?sap[[:blank:]]",
                        "[[:blank:]]#?microsoft[[:blank:]]",
                        "[[:blank:]]#?hp[[:blank:]]",
                        "[[:blank:]]#?linkedin[[:blank:]]",
                        "[[:blank:]]#?berkeley[[:blank:]]",
                        "[[:blank:]]#?nsa[[:blank:]]",
                        "[[:blank:]]#?reddit[[:blank:]]")
ml.brands.labels  <- c( "###IBM",
                        "###ORACLE",
                        "###NASA",
                        "###GOOGLE",
                        "###FACEBOOK",
                        "###TWITTER",
                        "###STANFORD",
                        "###PRINCETON",
                        "###HARVARD",
                        "###MIT",
                        "###CLOUDERA",
                        "###KAGGLE",
                        "###ACCENTURE",
                        "###KPMG",
                        "###SAP",
                        "###MICROSOFT",
                        "###HP",
                        "###LINKEDIN",
                        "###BERKELEY",
                        "###NSA",
                        "###REDDIT")
# add hierarchy labels
ml.brands.labels <- paste0(ml.brands.labels," ###MLBRANDS")
for(i in seq(ml.brands.topics))
{
      filtro <- grep(ml.brands.topics[i],tw.eng$norm)
      tw.eng$norm[filtro] <- paste0(tw.eng$norm[filtro],
            " ",ml.brands.labels[i])
}
rm(i)
```

Funzione “funz.grafo”

```
funz.grafo<-function(matrice,citati,citatori)
{
      # Creazione grafo da una matrice di "edgelist"
      edge = graph.edgelist(matrice)
      # Conversione a matrice di adiacenza
      adj = get.adjacency(edge,sparse=T)
      # Creazione grafo dalla matrice di adiacenza
      grafo = graph.adjacency(adj,mode="directed",weighted=T)

      # Nomi citatori in ordine
      primi=names(sort(table(matrice[,1]),decreasing=T))
      # Nomi citati
      nomi=names(sort(table(matrice[,2]),decreasing=T))
      # Salvo le Label
      labs=V(grafo)$name
```

```
# Dimensioni
deg <- 6*(degree(grafo)/max(degree(grafo)))
# Dei nodi
egam <- (log(E(grafo)$weight)+.4)/max(log(E(grafo)$weight)+.4)
# Dei collegamenti
conteggi <- as.numeric(sort(table(matrice[,2]),decreasing=T))/
            max(table(matrice[,2])) # Delle Label

V(grafo)$name=NA        # Annullo tutti i nomi
V(grafo)$color<-NA      # Annullo i colori dei nodi
V(grafo)$frame.color=NA
V(grafo)$label.cex=0.3
V(grafo)$size=deg       # Assegno dimensione nodi pari a deg
# Assegno a tutti i nodi colore quasi assente
V(grafo)$frame.color=rgb(0,0,0,.1)

# Citatori
for(i in 1:citatori)
{
      # Mostro i nomi piu' "importanti"
      V(grafo)$name[which(labs==primi[i])]=
            labs[which(labs==primi[i])]

      # Assegno colore dei nomi
      V(grafo)[which(V(grafo)$name==
            primi[i])]$label.color="darkgreen"

      # Assegno colore dei nodi
      V(grafo)[which(V(grafo)$name==primi[i])]$color="blue"
      V(grafo)[which(V(grafo)$name==
            primi[i])]$frame.color="blue"
}

# Citati
if(length(nomi)<citati) { citati=length(nomi) }
for(i in 1:citati)
{
      # Mostro i nomi dei citati
      V(grafo)$name[which(labs==nomi[i])]=
            labs[which(labs==nomi[i])]

      # Assegno colore dei nomi
      V(grafo)[which(V(grafo)$name==nomi[i])]$label.color="red"

      # Assegno dimensione dei nomi
      V(grafo)[which(V(grafo)$name==
            nomi[i])]$label.cex=conteggi[i]

      # Assegno colore dei nodi
      V(grafo)[which(V(grafo)$name==nomi[i])]$color="green"
      V(grafo)[which(V(grafo)$name==
            nomi[i])]$frame.color="green"
}

V(grafo)$label.family="mono"        # Assegno font delle Label
# Assegno colore dei collegamenti
E(grafo)$color=rgb(.5,.5,0,egam)
E(grafo)$width=egam     # Assegno dimensione dei collegamenti
E(grafo)$arrow.size=0.1       # Assegno dimensione1 delle frecce
E(grafo)$arrow.width=0.1      # Assegno dimensione2 delle frecce
```

```
	return(grafo)
}
```

Funzione “funz.grafolabel”

```
funz.grafolabel<-function(matrice)
{
	# Creazione grafo da una matrice di "edgelist"
	edge = graph.edgelist(matrice)
	# Conversione a matrice di adiacenza
	adj = get.adjacency(edge,sparse=T)
	# Creazione grafo dalla matrice di adiacenza
	grafo = graph.adjacency(adj,mode="directed",weighted=T)

	V(grafo)$label.color=NA

	# Dimensioni
	deg <- 6*(degree(grafo)/max(degree(grafo)))
	egam <- (log(E(grafo)$weight)+.5)/max(log(E(grafo)$weight)+.5)

	for(i in seq(V(grafo)$name))
	{
		freqB=length(which(matrice[,2]==V(grafo)$name[i]))/
			max(table(matrice[,2]))
		freqA=length(which(matrice[,1]==V(grafo)$name[i]))/
			max(table(matrice[,1]))

		if(freqA==0)
		{
			V(grafo)[which(V(grafo)$name==
				V(grafo)$name[i])]$label.color="blue"
			V(grafo)[which(V(grafo)$name==
				V(grafo)$name[i])]$label.cex=0.4+freqB
			V(grafo)[which(V(grafo)$name==
				V(grafo)$name[i])]$color="orange"
			V(grafo)[which(V(grafo)$name==
				V(grafo)$name[i])]$frame.color="orange"
		}
		if(freqB==0)
		{
			V(grafo)[which(V(grafo)$name==
				V(grafo)$name[i])]$label.color="darkgreen"
			V(grafo)[which(V(grafo)$name==
				V(grafo)$name[i])]$label.cex=0.4+freqA
			V(grafo)[which(V(grafo)$name==
				V(grafo)$name[i])]$color="violet"
			V(grafo)[which(V(grafo)$name==
				V(grafo)$name[i])]$frame.color="violet"
		}
		if((freqB!=0)&&(freqA!=0))
		{
			V(grafo)[which(V(grafo)$name==
				V(grafo)$name[i])]$label.color="black"
			if(freqB>freqA)
			{
				V(grafo)[which(V(grafo)$name==
					V(grafo)$name[i])]$label.cex=0.4+freqB
			}
			else
```

```
                  {
                        V(grafo)[which(V(grafo)$name==
                              V(grafo)$name[i])]$label.cex=0.4+freqA
                  }
                  V(grafo)[which(V(grafo)$name==
                        V(grafo)$name[i])]$color="red"
                  V(grafo)[which(V(grafo)$name==
                        V(grafo)$name[i])]$frame.color="red"
            }
      }

      V(grafo)$label.family="mono"        # Assegno font delle Label
      V(grafo)$size=deg

      E(grafo)$color=rgb(.5,.5,0,0.5)     # Colore dei collegamenti
      E(grafo)$width=0.1+egam             # Dimensione dei collegamenti
      E(grafo)$arrow.size=0.4             # Dimensione1 delle frecce
      E(grafo)$arrow.width=0.4            # Dimensione2 delle frecce

      return(grafo)
}
```

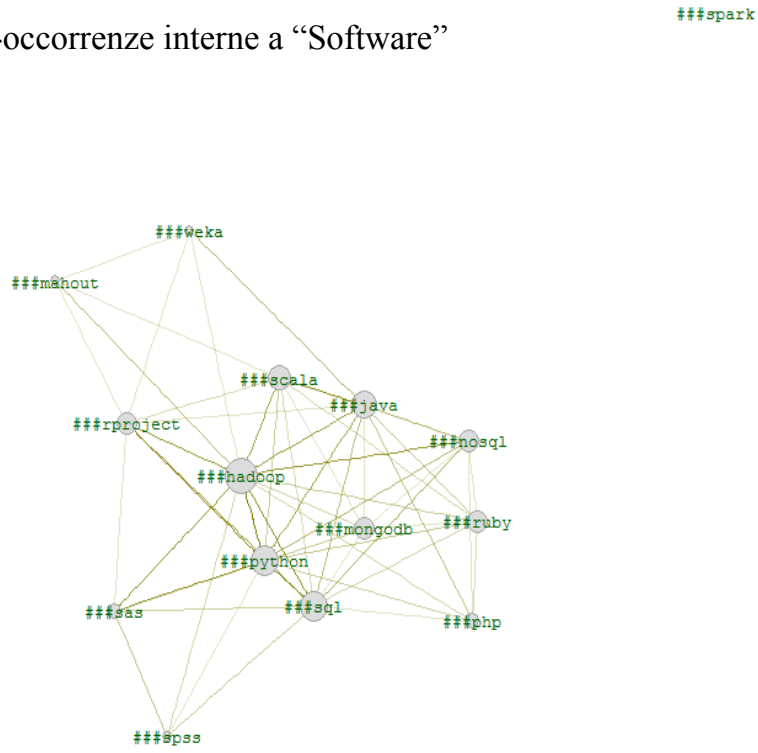
Costruzione delle reti “Chi Cita Chi secondo la label”

```
### SOFTWARE
software=gsub(" ###MLSOFTWARE","",ml.software.labels)
software=gsub(" #IBM","",software)
software=gsub(" #APACHE","",software)
m.soft=as.list(1:length(software))
g.soft=as.list(1:length(software))
d.soft=as.list(1:length(software))

for(i in seq(software))
{
      m.soft[[i]]=funz.term(software[i],tw.eng)
      g.soft[[i]]=funz.grafolabel(m.soft[[i]])
      d.soft[[i]]=degree(g.soft[[i]])
}
rm(i)

### APPAREAS
appareas=gsub(" ###MLAPPAREAS","",ml.appareas.labels)
m.app=as.list(1:length(appareas))
g.app=as.list(1:length(appareas))
d.app=as.list(1:length(appareas))

for(i in seq(appareas))
{
      m.app[[i]]=funz.term(appareas[i],tw.eng)
      g.app[[i]]=funz.grafolabel(m.app[[i]])
      d.app[[i]]=degree(g.app[[i]])
}
rm(i)

### BRANDS
brands=gsub(" ###MLBRANDS","",ml.brands.labels)
m.brands=as.list(1:length(brands))
g.brands=as.list(1:length(brands))
d.brands=as.list(1:length(brands))
```

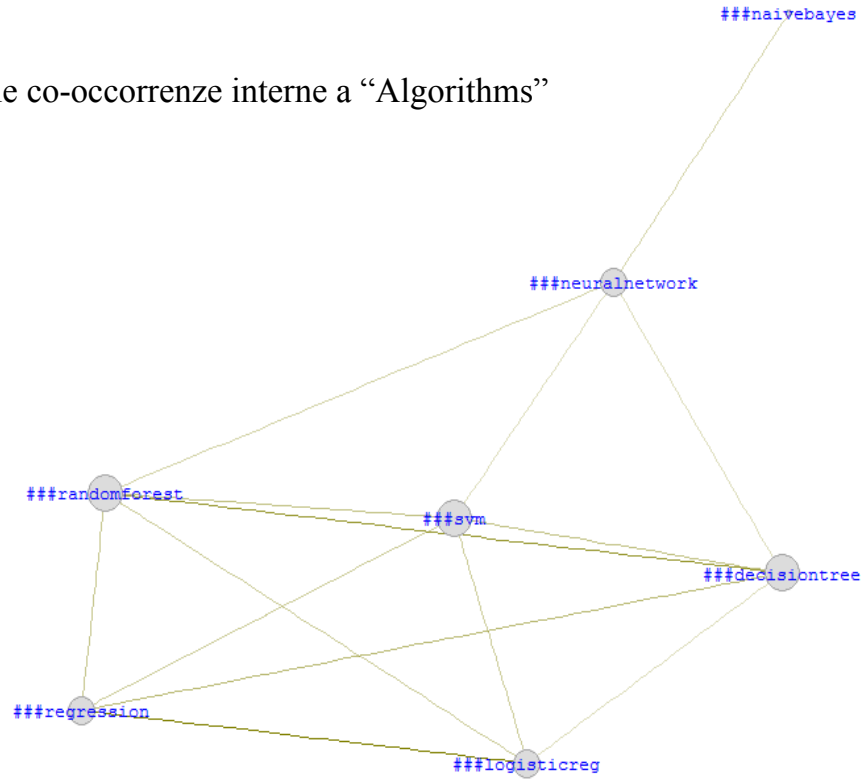
```
for(i in seq(brands))
{
      m.brands[[i]]=funz.term(brands[i],tw.eng)
      g.brands[[i]]=funz.grafolabel(m.brands[[i]])
      d.brands[[i]]=degree(g.brands[[i]])
}
rm(i)

### TECHNIQUES
techniques=gsub(" ###MLTWTECHNIQUES","",ml.techniques.labels)
m.tech=as.list(1:length(techniques))
g.tech=as.list(1:length(techniques))
d.tech=as.list(1:length(techniques))

for(i in seq(techniques))
{
      m.tech[[i]]=funz.term(techniques[i],tw.eng)
      g.tech[[i]]=funz.grafolabel(m.tech[[i]])
      d.tech[[i]]=degree(g.tech[[i]])
}
rm(i)

### TWABOUT
about=gsub(" ###MLTWEETABOUT","",ml.twabout.labels)
m.about=as.list(1:length(about))
g.about=as.list(1:length(about))
d.about=as.list(1:length(about))

for(i in seq(about))
{
      m.about[[i]]=funz.term(about[i],tw.eng)
      g.about[[i]]=funz.grafolabel(m.about[[i]])
      d.about[[i]]=degree(g.about[[i]])
}
rm(i)
```

Appendice B Grafi

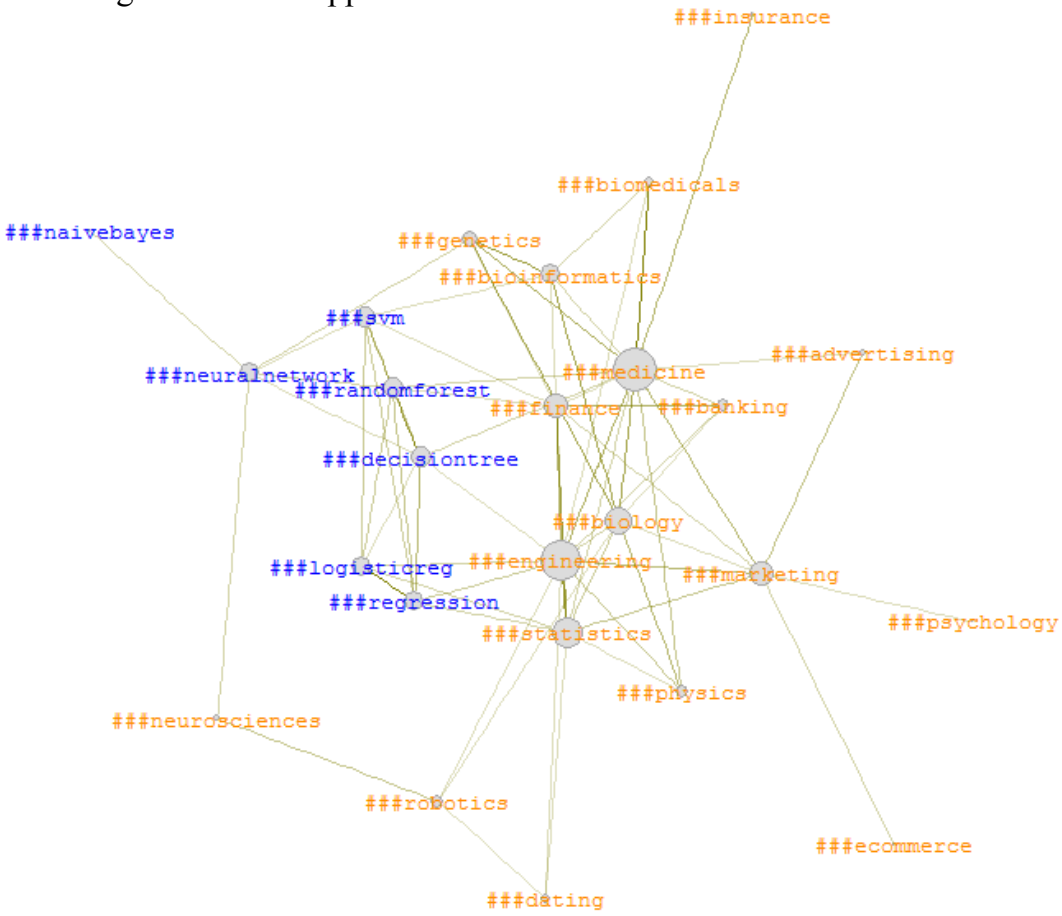
Grafo delle co-occorrenze interne a “Software”



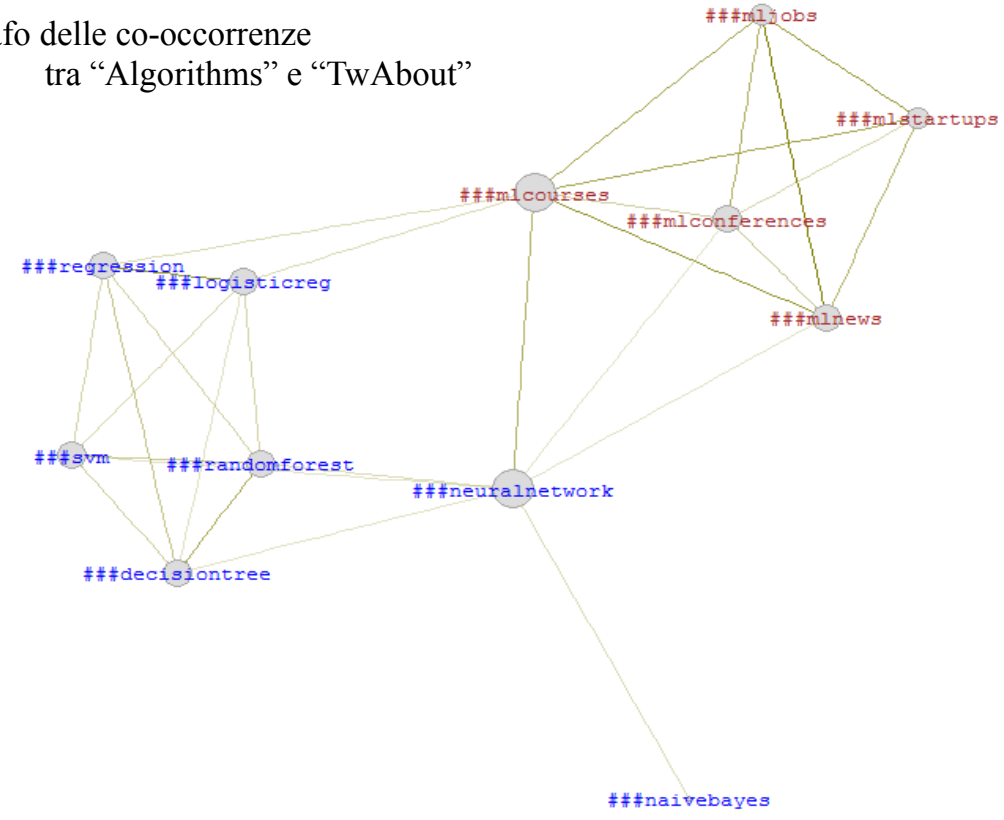
Grafo delle co-occorrenze interne a “Algorithms”



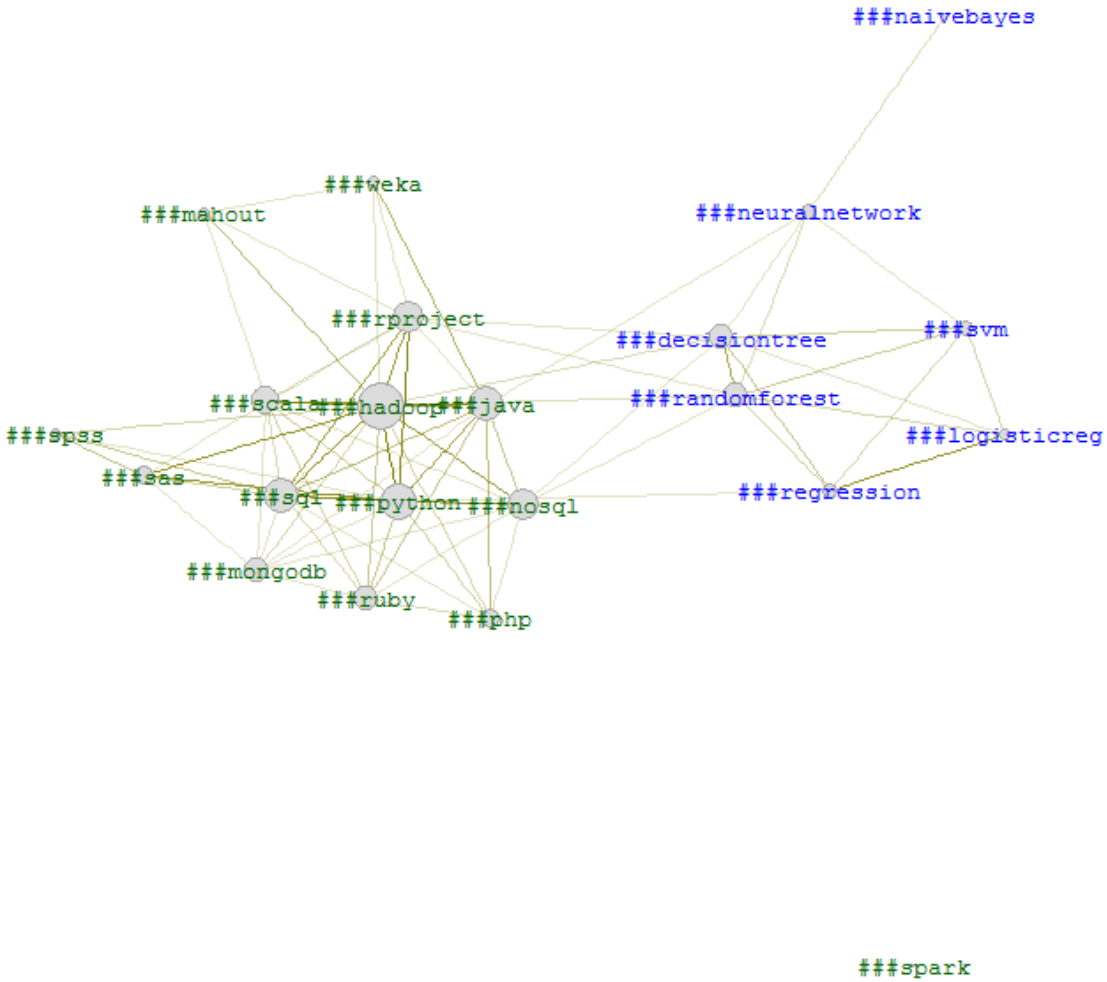
Grafo delle co-occorrenze tra
"Algorithms" e "Appareas"



Grafo delle co-occorrenze
tra "Algorithms" e "TwAbout"



Grafo delle co-occorrenze tra “Algorithms” e “Software”



Appendice C StopWords Inglesi

	1	2	3	4	5	6
1	i	what	you're	doesn't	as	further
2	me	which	he's	don't	until	then
3	my	who	she's	didn't	while	once
4	myself	whom	it's	won't	of	here
5	we	this	we're	wouldn't	at	there
6	our	that	they're	shan't	by	when
7	ours	these	i've	shouldn't	for	where
8	ourselves	those	you've	can't	with	why
9	you	am	we've	cannot	about	how
10	your	is	they've	couldn't	against	all
11	yours	are	i'd	mustn't	between	any
12	yourself	was	you'd	let's	into	both
13	yourselves	were	he'd	that's	through	each
14	he	be	she'd	who's	during	few
15	him	been	we'd	what's	before	more
16	his	being	they'd	here's	after	most
17	himself	have	i'll	there's	above	other
18	she	has	you'll	when's	below	some
19	her	had	he'll	where's	to	such
20	hers	having	she'll	why's	from	no
21	herself	do	we'll	how's	up	nor
22	it	does	they'll	a	down	not
23	its	did	isn't	an	in	only
24	itself	doing	aren't	the	out	own
25	they	would	wasn't	and	on	same
26	them	should	weren't	but	off	so
27	their	could	hasn't	if	over	than
28	theirs	ought	haven't	or	under	too
29	themselves	i'm	hadn't	because	again	very

Bibliografia

Christian Buchta, Kurt Hornik, Ingo Feinerer, and David Meyer. tau: Text Analysis Utilities, 2012. URL http://CRAN.R-project.org/package=tau. R package version 0.0-18.

Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. InterJournal, Complex Systems:1695, 2006. URL http://igraph.org.

Dario Solari, Livio Finos, Matteo Redaelli, con contributi di Marco Rinaldo, Maddalena Branca, and Federico Ferraccioli. TextWiller: Collection of functions for text mining, specially devoted to the italian language, 2013. R package version 1.0.

Angela Bohn, Ingo Feinerer, Kurt Hornik and Patrick Mair: Content-Based Social Network Analysis of Mailing Lists.
http://journal.r-project.org/archive/2011-1/RJournal_2011-1_Bohn~et~al.pdf
http://www.r-project.org/conferences/useR-2009/slides/Bohn+Feinerer+Hornik+Theussl.pdf

http://en.wikipedia.org/wiki/Social_network_analysis

http://www.infotoday.com/online/may12/Belter-Visualizing-Networks-of-Scientific-Research.shtml

http://solomonmessing.wordpress.com/2012/09/30/working-with-bipartiteaffiliation-network-data-in-r/

https://sites.google.com/site/miningtwitter/questions/talking-about

http://cs.brynmawr.edu/Courses/cs380/spring2013/section02/slides/05_Centrality.pdf

http://didattica.uniroma2.it/assets/uploads/corsi/39157/reti.pdf