Metodologie di Analisi dei Dati Testuali

Software per l'Analisi dei dati Testuali

Metodi statistici per il Data Mining II
Prof.ssa Simona Balbi

Dott.ssa Maria Spano,
DISES, Università di Napoli Federico II
maria.spano@unina.it

Glossario:

- **forma grafica** (parola o type) è una sequenza di caratteri (bytes) di un alfabeto predefinito
- i caratteri non compresi nell'alfabeto sono detti separatori (deboli)
- ogni apparizione di una parola nel testo si definisce occorrenza

- la **frequenza** di una parola in un testo è data dal numero delle sue occorrenze

- un **segmento** è una sequenza di parole adiacenti; un **poliforme** è un segmento di senso compiuto; talvolta esso viene lessicalizzato ossia considerato come un'unica forma

- **frammento di testo** (una frase, un intero documento etc…) è un insieme di parole, eventualmente diviso in sezioni (ad es.: titolo, occhiello, testo di un articolo di giornale)

- il **corpus** è un insieme di frammenti, costituito da uno o più elementi

- **vocabolario del corpus:** lista delle parole diverse del corpus con le corrispondenti occorrenze

- **lessico di frequenza:** il vocabolario di un corpus di ampie dimensioni (milioni di occorrenze), derivante da una raccolta di testi rappresentativi di un dato dominio di linguaggio (parlato, scritto, di settore etc.…)

Per effettuare un'analisi automatica del testo è necessario codificare l'informazione testuale per trasformare il corpus oggetto di analisi in un formato direttamente trattabile con strumenti statistici.

L'insieme delle operazioni di pulizia del testo, note come fase di pretrattamento, consistono nello svolgimento di diversi passi integrati fra loro:

- Parsing
- Normalizzazione
- Costruzione del vocabolario di lavoro
- Estrazione dei segmenti
- Lessicalizzazione
- Tagging Grammaticale
- Tagging Semantico
- Lemmatizzazione

- **Parsing:** consiste in l'individuazione delle successioni di caratteri dell'alfabeto comprese tra i separatori. Questa procedura scansiona il testo, identificando le forme grafiche presenti in esso.

- **Normalizzazione**: agisce sull'insieme dei caratteri non separatori, eliminando le possibili repliche del dato (le forme grafiche con lettera iniziale maiuscola o minuscola). Inoltre, attraverso questa procedura è possibile uniformare le forme che presentano forte variabilità, come ad esempio date, sigle e nomi propri.

- **Costruzione del vocabolario di lavoro**: identificazione dell'unità minimale di senso attraverso la fase di lessicalizzazione. Per effettuare procedura di lessicalizzazione è, però, necessario identificare, fissando a priori una soglia di frequenza, le poliriematiche (es. carta di credito) e i poliformi presenti nel testo (segmenti ripetuti). Dalla loro identificazione è possibile marcare gli stessi all'interno del corpus, creando un nuovo vocabolario di forme testuali.

- **Tagging grammaticale:** una delle principali operazioni, che attraverso l'identificazione della categoria grammaticale delle parole, consente di suddividere le POS (Part Of Speech) funzionali (articoli, preposizioni, congiunzioni) dalle POS lessicali (nomi, aggettivi, verbi).

- **Tagging semantico:** attraverso l'uso di risorse statistico-linguistiche consente di annotare il vocabolario con meta-informazioni di tipo semantico.

- **Lemmatizzazione del testo:** Per lemma si intende la "forma canonica" con cui una data voce è presente nel dizionario: si considera quindi l'infinito per i verbi, il singolare per i sostantivi, il singolare maschile per gli aggettivi. La trasformazione delle forme grafiche in lemma, producendo una grande standardizzazione del testo e una grande riduzione della dimensione. (Questa è una scelta da ponderare, in quanto può cancellare gran parte dell'informazione.)

Come importare dati tramite file testuali

Il software R consente di leggere le informazioni redatte in formato testuale e mantenute nei corrispondenti file di testo. La provenienza di questi file può essere la più varia: possono essere creati da editor di testo (il blocco note di Windows), fogli elettronici (Excel) o da software di elaborazione statistica (SpSS, SAS, …) oppure da programmi creati ad hoc per l'utente utilizzando i normali liguaggi di programmazione quali Fortran, C, C++, JAVA.

Le operazioni di lettura

La funzione *scan*() rappresenta il modo più flessibile per importare un file testuale in R.

what="char" legge dati di tipo testuale
sep="\n" , sep="", sep="\t"

Scansione del testo e costruzione del vocabolario

```
Art1<- scan("Articolo1.txt", what="char", sep="\n")
#visualizzare l'elenco di types
Art1

Art1<- scan("Articolo1.txt", what="char", sep="")

# trasformare tutti in minuscolo
Art2<-tolower(Art1)

#utile per inserire una lettera al posto di un'altra
Art3<-chartr("a","o",Art2)

#introdurre il separatore per le parole
word.list<-strsplit(Art2, "\\W")

#creare una lista di parole
word.vector<-unlist(word.list)

#Ricerca di alcune parole: la funzione grep() può essere utilizzata per ricercare
espressioni. #Dove si trova la parola "referendum"?
grep("referendum", Art2)
```

```
#creare una lista delle parole da eliminare
stop.list<-c("di", "e", "del", "in", "la", "a", "il", "i", "che ", "")

#! La stop list può essere un file di testo esterno importato in R con la
#funzione read.table.

# eliminare dalla lista delle parole definite nel vettore "stop.list"
word.vector1<-word.vector[!(word.vector%in%stop.list)]
word.vector1

#costruire il vocabolario del corpus
vocab<-table(word.vector1)

#ordinamento lessicometrico del vocabolario
sort.vocab<-sort(vocab,decreasing=TRUE)

#visualizzare le forme in testa
head(sort.vocab)

#visualizzare le ultime forme del sort.vocab
tail(sort.vocab)

#selezionare le parole con frequenza >1
sort.vocab1<-sort.vocab[sort.vocab>1]
```

```
# esportare il vocabolario in un file .txt
write.table(sort.freq.tab2, "table1.txt", row.names = TRUE, col.names =
TRUE, sep=" ",quote=F)
```

```
#Costruire un dataframe dal vocabolario
w<- data.frame(sort.vocab1)
```

```
#Costruire la nuvola di parole ("wordcloud")
#E' necessario installare il pacchetto "wordcloud"
library(wordcloud)
wordcloud(w$word.vector1, w$Freq)
```

Word Clouds

http://www.wordle.net/create

http://www.tagxedo.com

Descrizione quantitativa del corpus

Per poter descrivere un *corpus* da un punto di vista quantitativo è necessario individuare alcune grandezze caratteristiche. Data una collezione di documenti, indichiamo con:

N -> la dimensione del *corpus*
(insieme di tutte le forme che compongono il *corpus*)

V -> l'ampiezza del vocabolario
(insieme di tutte le forme uniche presenti nel *corpus*)

Nella notazione tipica del Text Mining e del trattamento automatico dei testi le diverse forme del corpus sono solitamente indicate come **token**, mentre le forme appartenenti al vocabolario sono indicate come **type**

Dopo aver effettuato il parsing del corpus e aver completato la numerizzazione è possibile costruire il vocabolario

$$V = V_1 + V_2 + \ldots + V_i + \ldots V_{f_{max}} = \sum_{i=1}^{f_{max}} V_i$$

In generale V_i rappresenta il numero di forme che hanno nell'intero corpus un numero di occorrenze pari a i. Il primo elemento V_1 è il numero di hapax (hapax legomenon = detto una volta sola) del corpus, cioè tutte le forme presenti soltanto una volta

$$N = V_1 + 2V_2 + \ldots + iV_i + \ldots f_{max}V_{f_{max}} = \sum_{i=1}^{f_{max}} iV_i$$

Estensione lessicale $\frac{V}{N} x100$ *Numero di forme distinte rispetto alla estensione del corpus*

Il ***type/token ratio*** fornisce indicazioni riguardo alla possibilità di analisi statistica del *corpus*: per valori superiori al 20% il *corpus* non è sufficientemente esteso per poter cogliere la ricchezza del linguaggio da analizzare

Ricercatezza nel linguaggio $\frac{V_1}{V} x100$ *Numero di hapax rispetto al numero di forme distinte*

La percentuale di *hapax* nel vocabolario indica quanto il linguaggio utilizzato nel *corpus* è "ricercato": per valori superiori al 50% il *corpus* è costituito da troppe forme originali e quindi non è trattabile statisticamente

Analisi quantitativa del vocabolario

```
#N<- numero di forme nel corpus (numero di token)
#V<-numero di forme distinte nel vocabolario (numero di type)
#V1<-numero di hapax

word.vector0<-word.vector[!(word.vector%in%"")]
N<-length(word.vector0)
vocab0<-table(word.vector0)
V<-length(vocab0)
V1<-length(vocab0[vocab0==1])

#Type-Token Ratio
#Più TTR risulta basso meno ricco è il vocabolario del corpus (soglia 20%).
TTR<-V/N

#Frequenza media delle forme
Inv_TTR<-1/TTR
Inv_TTR

#La percentuale di hapax nel vocabolario
#Più è alto questo valore maggiore è la ricercatezza del linguaggio
Per_hap<-V1/V
```

Il pacchetto tm per la gestione, la trasformazione e la creazione della tabella lessicale

Caricare il pacchetto
library(tm)

Leggere un file di testo
La funzione per creare una struttura è: *Corpus*(x, readerControl)
x rappresenta la fonte dati:
- DirSource - utilizzabile soltanto per directory e file di sistema
- VectorSource - vettore di caratteri o valori stringa
- DataframeSource - manipola una directory, un vettore interpretando ciascuna componente come un documento

Esplorare il *corpus*
Per conoscere il contenuto del corpus possiamo usare diverse funzioni:
print(nomecorpus) <-per sapere da quanti documenti è formato il corpus
summary(nomecorpus) <- per conoscere quali sono i metadati del corpus
inspect(nomecorpus) <- per conoscere il contenuto del corpus

Esportare un *corpus*
Per salvare i documenti in un file txt
writeCorpus(nomecorpus)

Pre-processing (pretrattamento)
Dopo aver caricato il corpus, possiamo procedere con le operazioni preliminari, ad esempio, rimozione delle parole vuote ecc. Nel pacchetto tm tutte queste operazioni sono riassunte dal concetto della trasformazione e sono possibili attraverso la funzione *tm_map*().

Eliminare spazi
tm_map(nomecorpus,**stripWhitespace**)

Trasformare in minuscolo/maiuscolo
tm_map(nomecorpus, **tolower/toupper**)

Rimuovere la punteggiatura
tm_map(nomecorpus, **removePunctuation**)

Stemming
Lo **stemming** è il processo di riduzione della forma flessa di una parola alla sua forma radice, detta tema. Il tema non corrisponde necessariamente alla radice morfologica (lemma) della parola. (ad esempio, che andare, andai, andò mappino al tema *and*, anche se quest'ultimo non è una valida radice per la parola)
tm_map(nomecorpus, **stemDocument**)

Rimuovere le stop words
tm_map(nomecorpus, **removeWords, stopwords("lingua_del_corpus")**)

Importazione e pretrattamento del corpus

```
#Caricare i pacchetti tm e koRpus
library(tm)
library(koRpus)

#Creare il corpus dai file che si trovano nella cartella
text.corp<-Corpus(DirSource("Alice1",encoding="UTF-8"))

#Esplorare il contenuto del corpus
inspect(text.corp)
text.corp[[1]]$content oppure
text.corp[[1]][1]

# Procedere con le operazioni di pretrattamento
text.corp1<-tm_map(text.corp,stripWhitespace)
text.corp2<- tm_map(text.corp1, removeNumbers)
text.corp3<-tm_map(text.corp2, tolower)
text.corp4<-tm_map(text.corp3, removeWords, stopwords("en"))
text.corp5<-tm_map(text.corp4, removePunctuation)
```

```
#creare e esplorare una tabella lessicale
text.corp5<-tm_map(text.corp5,  PlainTextDocument)
tdm<-TermDocumentMatrix(text.corp5)
inspect(tdm)
dim(tdm)

## operazioni sulla tabella lessicale

#selezionare le forme con occorenza maggiore o uguale a 5
findFreqTerms(tdm, 5)

# selezionare le forme con occorenza compresa tra 2 e 5
findFreqTerms(tdm, lowfreq=2, highfreq=5)

#creare una nuova tabella lessicale senza hapax
tdm1<-tdm[findFreqTerms(tdm, 2),]
marg_riga<-apply(tdm1, 1, sum)

#Trovare i termini che presentano un’associazione pari almeno di 0.7 con
#la parola “alice"
findAssocs(tdm, “alice", 0.7)
```

```
##schema di ponderazione di frequenza
bin <- TermDocumentMatrix(text.corp5, control = list(weighting = weightBin,
stopwords = TRUE))
bin
inspect(bin)


## schema di ponderazione di frequenza
bow <- TermDocumentMatrix(text.corp5, control = list(weighting = weightTf,
stopwords = TRUE))
bow
inspect(bow)


## schema di ponderazione tfidf
tfidf <- TermDocumentMatrix(text.corp5, control = list(weighting =
weightTfIdf, stopwords = TRUE))
tfidf
inspect(tfidf)
```

La matrice TermDocument Matrix di solito risulta sparsa, quindi molti termini appaiono in pochi documenti. Una riduzione drastica delle dimensioni di questa matrice di solito non fa perdere di significato alle relazioni inerenti la matrice.

```
# rimuovere termini sparsi
#Più basso il valore del parametro nella funzione removeSparseTerms più parole sparse eliminiamo.
rms<-removeSparseTerms(tdm1, 0.9)

## analisi delle corrispondenze lessicali, carico la libreria ca
library(ca)
## ca è la funzione che consente di implementare l'analisi delle corrispondenze
ca<-ca(rms)
summary(ca)
plot(ca)
#plot(ca(rms), what = c("all", "none"), mass = FALSE, )
#plot(ca(author), what = c("none", "all"), mass = TRUE, contrib = "relative")
#plot(ca(author), what = c("none", "all"), mass = TRUE, contrib = "absolute")

## coordinate parole
dimr<-ca$rowcoord
dimr
## coordinate ristoranti
dimc<-ca$colcoord
dimc
```

Il pacchetto TwitteR

Per il download di tweets è necessario creare un account personale su Twitter.
Dopo aver effettuato il login, collegarsi alla pagina apps.twitter.com
Creare un'applicazione scegliendo un nome a piacere, che sia collegata al proprio account.
Sarà possibile ottenere in tal modo le credenziali (consumer_key, consumer_token……..) per far si che R si interfacci con Twitter

In R:

```
library(twitteR)
library(RCurl)

###Processo di autorizzazione API
consumer_key <- "……."
consumer_secret <- "……."
access_token <- "……."
access_secret <- "……."

setup_twitter_oauth(consumer_key,
consumer_secret, access_token, access_secret)
```

Una volta effettuata la procedura di autenticazione, tramite la funzione searchTwitter, sarà possibile scaricare i tweets in R

```
searchTwitter(searchString, n=25, lang=NULL, since=NULL, until=NULL,
              locale=NULL, geocode=NULL, sinceID=NULL, maxID=NULL,
                   resultType=NULL, retryOnRateLimit=120, …)
Rtweets(n=25, lang=NULL, since=NULL, …)
```

```
###Esempio1 #cerco gli ultimi 10 tweets che contengono l'hashtag vaccino
vaccino<-searchTwitter("#vaccino", n=10)
###Esempio2 #cerco gli ultimi 100 tweets che contengono gli hashtag vaccino e vaccini
vaccino1<-searchTwitter("#vaccino+#vaccini", n=100)
###Esempio3 #cerco i tweets in italiano che contengono il termine vaccino nelle date specificate
vaccino<-searchTwitter("vaccino",lang="it",since= "2016-11-30",until = "2016-12-01"))

#per estrarre unicamente il testo del tweet
vaccino_testo<-sapply(vaccino,function(x)
x$getText())
```

In alternativa per scaricare i tweets è possibile utilizzare il componente aggiuntivo di google Twitter Archiver installabile al seguente link:

https://chrome.google.com/webstore/detail/twitter-archiver/pkanpfekacaojdncfgbjadedbggbbphi

Software per l'analisi dei dati testuali

- ✓ **IRaMuTeQ (www.iramuteq.org)**
- ✓ **Taltac(www.taltac.it)**
- ✓ **Lexico (www.tal.univ-paris3.fr/lexico/lexico3.htm)**
- ✓ **SATO (sato.ato.uqam.cam)**
- ✓ **DtmVic (www.dtmvic.com/05_SoftwareI.html)**
- ✓ **TLAB (tlab.it/default.php)**