Alma Mater Studiorum · Università di Bologna

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI

Corso di Laurea in Matematica

METODI MATEMATICI PER IL RECUPERO DEI NESSI CONCETTUALI IN UNO O PIÙ TESTI: ENTROPIA E ANALISI DELLA SEMANTICA LATENTE

Tesi di Laurea in Matematica

Relatore:
Chiar.mo Prof.
Mirko Degli Esposti

Presentata da:
FEDERICO
BARUFFALDI

Terza Sessione
Anno Accademico 20011-2012

Indice

Introduzione

Lo sviluppo delle tecnologie informatiche a cui si è assistito negli ultimi decenni ha portato ad una conseguente crescita dei dati immagazzinati; consequenzialmente è sopraggiunta la necessità di automatizzare il processo di analisi e recupero di documenti all'interno di grandi masse di dati. Tale automazione richiede naturalmente l'utilizzo di software capace di trattare diverse tipologie di documenti, valutarne il peso in base a ciò che essi contengono: imitare cioè i processi cognitivi di un essere umano nell'analisi dei documenti, con la potenzialità ulteriore di poterlo svolgere su quantità molto vaste di dati; in altre parole la soluzione del problema richiede la creazione di una forma di "intelligenza artificiale" in senso lato, che significa semplicemente la riproduzione di analisi tipicamente umane da parte di un calcolatore; di conseguenza diventa indispensabile costruire ed implementare metodi quantitativi per ricreare e produrre risultati strettamente qualitativi. In tal senso i documenti devono essere "destrutturati", ridotti ad una sequenza di simboli privi di significato dai quali dedurre proprietà successivamente interpretabili.

Gran parte delle tecniche di cui sopra poggiano su metodi statistici: in seguito ne verranno prese in considerazione due, entrambe fondate sulla statistica. La prima permette di mettere in atto quello che viene comunemente definito "word tagging", ovvero l'estrazione delle parole maggiormente sig-

nificative all'interno di un documento, analisi ad oggi sempre più frequente in molti blog, siti di quotidiani e svariate pagine web. Verranno presi in considerazione i metodi analitici e statistici con cui estrarre le parole di maggior interesse; verrà successivamente effettuata l'analisi di due testi tramite un software implementato in C che sfrutta tali metodi e verificato come l'eliminazione dai testi delle cosiddette "stopwords" (parole che considerate singolarmente non hanno rilevanza semantica, non sono caratterizzanti cioè del testo a cui appartengono come preposizioni, articoli, ecc...), non si discosta in maniera rilevante dai risultati ottenuti col testo completo.

La seconda tecnica, chiamata "Latent Semantic Analysis" è invece utile quando si ha a che fare con un corpus formato da numerosi testi e permette l'individuazione all'interno di essi della cosiddetta "semantica latente", ovvero in senso tecnico un insieme di variabili che possono essere interpretate come concetti astratti e permettono una maggior efficienza nell'analisi dei documenti in questione. Si analizzeranno innanzitutto i fondamenti statistici che conducono all'utilizzo della tecnica algebrica chiamata "single value decomposition" (SVD) come strumento chiave per l'individuazione della semantica latente; in seguito si cercherà di dare un'interpretazione ai risultati ottenuti tramite la SVD. Infine tramite il software Infomap NLP, implementato in C, si procederà ad un analisi del corpus della Divina Commedia per verificare l'efficacia del metodo.

Capitolo 1

Entropia delle parole in un testo

Lo scopo di questa analisi è quello di estrarre informazioni sul ruolo specifico delle parole attraverso un'analisi statistica del testo. Non è sufficiente prendere in considerazione quante volte una parola compare nel testo: è necessario piuttosto stabilire una graduatoria in base a come le parole vengono utilizzate. Per soddisfare questa esigenza viene utilizzata una forma adattata dell'entropia di Shannon. Consideriamo un corpus di testi dello stesso è autore come somma di P parti (a scala inferiore consideriamo un testo come somma di P capitoli, ogni capitolo come somma di sottocapitoli, e così via in base a ciò che stiamo analizzando). Chiamiamo N_i il numero di parole totali presenti nella parte i e n_i il numero di occorrenze di una determinata parola in quella parte; la frequenza di una parola nella parte i-esima è data dal rapporto $f_i = n_i/N_i$.
Per ogni parola è possibile definire una misura di probabilità p_i su una partizione data (ovvero con P fissato) come

$$p_i = \frac{f_i}{\sum_{j=1}^{P} f_j} \tag{1.1}$$

La quantità p_i indica la probabilità di trovare la parola nella parte i. L'entropia di Shannon associata alla distribuzione di probabilità discreta appena calcolata sarà:

$$S = -\frac{1}{\ln P} \sum_{i=1}^{P} p_i \ln p_i \tag{1.2}$$

L'entropia di una determinata parola (che in generale cambia in base alla partizione scelta) va considerata come una caratterizzazione della sua distribuzione. Ad esempio se una parola è uniformemente distribuita sulle P parti, cioè $p_i = 1/P$ per ogni i, dall'equazione (1.2) si ottiene $S = 1$. Al contrario se una parola compare solo in una parte j-esima della partizione si ottiene $p_j = 1$ e $p_i = 0$ per ogni $i \neq j$, di conseguenza $S = 0$, che significa il minimo grado di incertezza sulla posizione della parola nel testo.
Questi due casi possono essere rappresentati con buona approssimazione da due categorie di parole: la prima è quella delle parole funzionali quali possono essere gli articoli, le preposizioni, le congiunzioni e così via, che sono distribuite uniformemente in tutto il testo indipendentemente dai contenuti delle diverse parti; la distribuzione dei funzionali non è perciò, in linea teorica, inficiata dall'argomento trattato, cioè dal senso del testo. Passando all'altro estremo invece, quindi avvicinandosi a bassi valori di entropia, ci si aspetta di incontrare parole specifiche il cui senso è strettamente legato all'argomento trattato in una certa sezione del testo e la cui distribuzione ha perciò notevoli fluttuazioni.

Dal grafico emergono due differenti tendenze: la prima è quella dell'entropia di una parola ad aumentare di pari passo con la frequenza della parola

stessa, il che è interpretabile in chiave statistica ed implica il fatto che in media più una parola è frequente, più uniformemente è utilizzata. In secondo luogo è possibile esaminare in maniera più approfondita il ruolo di ogni parola e la sua rilevanza all'interno di un testo analizzando la deviazione della sua entropia dall'andamento medio dell'entropia. Per rendere ciò possibile è sufficiente mettere in pratica un esperimento numerico che consiste nel generare una versione casuale del testo scelto, cioè un nuovo testo formato dalle stesse parole prese con la stessa frequenza ma disposte in ordine del tutto aleatorio. In generale si osserva che la tendenza dell'entropia a crescere con la frequenza di una parola è conservata e che le grosse fluttuazioni di entropia, dovute alla presenza di parole poco frequenti con bassa entropia, sono cancellate nella versione casuale. In media si può affermare che le parole hanno entropie più alte nella realizzazione aleatoria che nel testo originario. Ciò è proprio quello che ci si aspetta da determinate classi di parole, come i nomi propri che fanno riferimento a determinate a situazioni e circostanze che caratterizzano solo determinate parti del testo e quindi distribuite in maniera non uniforme. Nella versione casuale invece tutta la disomogeneità risulta annullata causando un aumento dell'entropia di tali parole.

1.1 Studio analitico del testo casuale

La versione casuale del testo ha il vantaggio di essere trattabile analiticamente, almeno in forma leggermente modificata come segue.

Supponiamo di avere un corpus di N parole totali suddiviso in P parti con N_i parole nella parte i. Data una particolare parola w che compare n_j volte nella parte j, la probabilità condizionata di trovare tale parola nella

parte j viene definita come già detto in precedenza da

$$p(w/j) = \frac{n_j}{N_j}$$

Definiamo poi la probabilità a priori che la parola w compaia nella j-esima parte come

$$p(j) = \frac{N_j}{N}$$

Allora avremo

$$p(w) = \sum_{j=1}^{P} p(w/j)p(j) = \sum_{j=1}^{P} \frac{n_j}{N_j}\frac{N_j}{N} = \frac{n}{N}$$

dove $p(w)$ è la probabilità complessiva di trovare la parola w nel testo. La probabilità $p(w/j)$ ci dice quindi quanto facile sia trovare la parola w all'interno della parte j. Tuttavia il vero interesse risiede nell'analizzare quanto una parola identifichi una determinata sezione del testo, cioè data una determinata parola studiare la probabilità di trovarla nella parte j-esima. Tale quantità è fornita dalla probabilità $p(j/w)$ che è facilmente calcolabile alla luce delle precedenti definizioni grazie alle regola di Bayes sulla probabilità condizionata:

$$p(j/w) = \frac{p(w/j)p(j)}{p(w)} = \frac{\frac{n_j}{N_j}\frac{N_j}{N}}{\frac{n}{N}} = \frac{n_j}{n}$$

Quest'ultimo risultato ci mostra che data una parola che compare n volte in totale, la probabilità che compaia nella parte j-esima dipende esclusivamente dal numero delle sue occorrenze n_j nella data sezione.

Ricordiamo che lo scopo è definire una misura sull'informazione del testo che quantifichi la relazione tra l'eterogeneità con cui sono distribuite le parole (dovuto alla loro funzione linguistica) e la partizione con cui il testo è suddiviso. Per fare ciò si utilizza l'*Informazione reciproca di Shannon* (Shannon Mutual Information) tra le variabili aleatorie J (numero di partizioni del

testo) e W (parole):

$$M(J,W) = \sum_{w=1}^{K} p(w) \sum_{j=1}^{P} p(j/w) \log_2 \Big(\frac{p(j/w)}{p(j)} \Big) \tag{1.3}$$

Chiamiamo poi $\hat{M}(J,W)$ l'informazione reciproca calcolata su un particolare testo casuale ottenuto "rimescolando" tutte le parole. Per ottenere una quantità che non dipenda da una particolare realizzazione casuale, si utilizza una media di tutte le realizzazioni casuali rappresentata da $\big\langle \hat{M}(J,W) \big\rangle$. È possibile ora definire l'informazione sulla distribuzione delle parole come la differenza tra il valore dell'informazione reciproca calcolata sul vero testo e sulla media dei testi casuali:

$$\Delta I(s) = M(J,W) - \big\langle \hat{M}(J,W) \big\rangle$$

Espandendo i termini, tenendo presente che $p(w) = n_j/n = \big\langle \hat{p}(w) \big\rangle$, abbiamo:

$$\sum_{w=1}^{K} p(w) \sum_{j=1}^{P} p(j/w) \log_2 \Big(\frac{p(j/w)}{p(j)} \Big) - \sum_{w=1}^{K} p(w) \sum_{j=1}^{P} \big\langle \hat{p}(j/w) \big\rangle \log_2 \Big(\frac{\big\langle \hat{p}(j/w) \big\rangle}{\big\langle \hat{p}(j) \big\rangle} \Big) =$$

$$= \sum_{w=1}^{K} p(w) \sum_{j=1}^{P} \Big(p(j/w) \log_2 p(j/w) - \big\langle \hat{p}(j/w) \log_2 \hat{p}(j/w) \big\rangle \Big) -$$

$$- \sum_{w=1}^{K} \sum_{j=1}^{P} \Big(\big\langle p(w) \hat{p}(j/w) \log_2 \hat{p}(j) \big\rangle - p(w) p(j/w) \log_2 p(j) \Big) =$$

$$= \sum_{w=1}^{K} p(w) \big[\big\langle \hat{H}(J/w) \big\rangle - H(J/w) \big] - \sum_{j=1}^{P} \Big(\big\langle \hat{p}(j) \log_2 \hat{p}(j) \big\rangle - p(j) \log_2 p(j) \Big)$$

Poichè l'entropia di J non dipende dal "rimescolamento" del testo, la seconda sommatoria vale 0 per cui si ottiene:

$$\Delta I(s) = \sum_{w=1}^{K} p(w) \big[\big\langle \hat{H}(J/w) \big\rangle - H(J/w) \big] \tag{1.4}$$

dove

$$\big\langle \hat{H}(J/w) \big\rangle = - \sum_{j=1}^{P} \big\langle \hat{p}(j/w) \log_2 \hat{p}(j/w) \big\rangle \tag{1.5}$$

1.2 Calcolo analitico dell'entropia media

In questa sezione prenderemo in esame il problema del calcolo di $\langle \hat{H}(J,w) \rangle$ che finora è stato usato solo nominalmente ma di cui serve un'espressione analitica quindi computabile. Come già detto, l'entropia $\hat{H}$ è calcolata su una versione stocastica del testo ottenuta mescolando casualmente le parole presenti in esso. Le parentesi $\langle \dots \rangle$ indicano una media su tutti i possibili testi rimescolati.

Come prima, consideriamo un testo di N parole diviso in P parti della stessa lunghezza. Usando l'equazione (1.5), per una parola che appare n_j volte nella parte j con una frequenza n nel testo e ricordando che $p(j/w) = n_j/n$, l'entropia su una versione stocastica del testo prende la forma:

$$\hat{H}(J/w) = -\sum_{j=1}^{P} \frac{n_j}{n} \log_2 \frac{n_j}{n}$$

La media di tale entropia su tutte le possibili realizzazioni di testi stocastici viene calcolata come segue:

$$\langle \hat{H}(J,w) \rangle = -\sum_{n_1+\cdots+n_P=n} p(n_1,\dots,n_P) \sum_{j=1}^{P} \frac{n_j}{n} \log_2 \frac{n_j}{n} \tag{1.6}$$

con

$$n_j \leq N/P, \;\; j = 1,\dots,P$$

dove $p(n_1,\dots,n_P)$ è la probabilità di trovare una certa parola n_j volte nella parte j-esima:

$$p(n_1,\dots,n_P) = n! \prod_{j=1}^{P} \frac{1}{n_j} \Big(\frac{N_j}{N}\Big)^{n_j}$$

Si nota che nell'equazione (1.6), per ognuno dei P termini della seconda somma, la prima somma può essere presa su tutti gli indici tranne uno. Questo porta alla seguente forma dell'entropia media:

$$\left\langle \hat{H}(J/w) \right\rangle = -P \sum_{m=1}^{min(n,N/P)} p(m) \frac{m}{n} \log_2 \frac{m}{n} \tag{1.7}$$

Infine la probabilità marginale può essere facilmente calcolata ed è data dalla probabilità di trovare m volte la parola w assieme ad $N/P - m$ parole diverse da w:

$$p(m) = \frac{\binom{n}{m}\binom{N-n}{N/P-m}}{\binom{N}{N/P}}$$

1.2.1 Approssimazione dell'entropia media

L'espressione trovata nell'equazione (1.7) ha il difetto di essere scomoda nel caso si debba utilizzare all'interno di un codice per calcolatore, il che è necessario per il tipo di problema che stiamo analizzando poichè solamente un calcolatore può svolgere in tempi ragionevoli tutti i calcoli necessari per valutare in un testo le quantità finora analizzate. La scomodità risiede principalmente nel coefficiente $p(m)$, cioè la probabilità marginale, poichè contiene fattoriali (dati dai coefficienti binomiali) che la memoria di una macchina non riesce a contenere. Basti pensare che in tali coefficienti binomiali compare N cioè il numero totale delle parole di un testo, che può raggiungere le decine di migliaia o anche più. Per ovviare a questo problema, vediamo in seguito un'approssimazione che semplifica il tutto in maniera molto efficiente.

Utilizziamo per far ciò una forma leggermente modificata della formula (1.7):

$$\left\langle S(n) \right\rangle = -\frac{P}{\ln P} \sum_{m=0}^{n} \frac{m}{n} \ln \left(\frac{m}{n}\right) \binom{n}{m} \frac{1}{P^m} \left(1 - \frac{1}{P}\right)^{n-m}$$

Per parole con alta frequenza, $n \gg 1$, possiamo approssimare la distribuzione binomiale con una Gaussiana $G(x;\mu,\sigma)$ con media $\mu = 1/P$ e varianza $\sigma^2 = (1/n)(P-1)/P^2$.
L'equazione può quindi essere riscritta come:

$$\langle S(n)\rangle \approx -\frac{P}{\ln P}\int_{-\infty}^{+\infty} x\ln x\; G(x;\mu,\sigma)dx$$

Al limite per $n \gg 1, \sigma \to 0$ e la distribuzione di probabilità gaussiana si concentra attorno al suo valore medio μ. Usando l'espansione della funzione $x\ln x$ in un intorno di μ si ottiene:

$$x\ln x \approx \mu\ln\mu + (1+\ln\mu)(x-\mu) + \frac{1}{2\mu}(x-\mu)^2$$

inserendo l'espressione trovata nell'integrale si ha:

$$\begin{aligned}\langle S(n)\rangle \approx &-\frac{P}{\ln P}\Big(\int_{-\infty}^{+\infty}\mu\ln\mu\; G(x;\mu,\sigma)dx \;+\; \int_{-\infty}^{+\infty}(1+\ln\mu)(x-\mu)G(x;\mu,\sigma)dx \;+\\ &+\int_{-\infty}^{+\infty}\frac{1}{2\mu}(x-\mu)^2G(x;\mu,\sigma)\Big) =\\ &=-\frac{P}{\ln P}\Big(\mu\ln\mu\int_{-\infty}^{+\infty}G(x;\mu,\sigma)dx \;+\; (1+\ln\mu)\int_{-\infty}^{+\infty}(x-\mu)G(x;\mu,\sigma)dx \;+\\ &+\frac{1}{2\mu}\int_{-\infty}^{+\infty}(x-\mu)^2G(x;\mu,\sigma)dx\Big)\end{aligned}$$

Poichè il primo integrale risulta 1, il secondo 0 e il terzo σ^2, abbiamo:

$$\langle S(n)\rangle \approx -\frac{P}{\ln P}\Big(\mu\ln\mu + \frac{1}{2\mu}\frac{1}{n}\frac{P-1}{P^2}\Big) = 1 - \frac{P-1}{2n\ln P}$$

Come già detto, tale approssimazione è valida solo per $n \gg 1$. Il seguente grafico evidenzia che un'approssimazione è accettabile già per $n > 10$ circa:

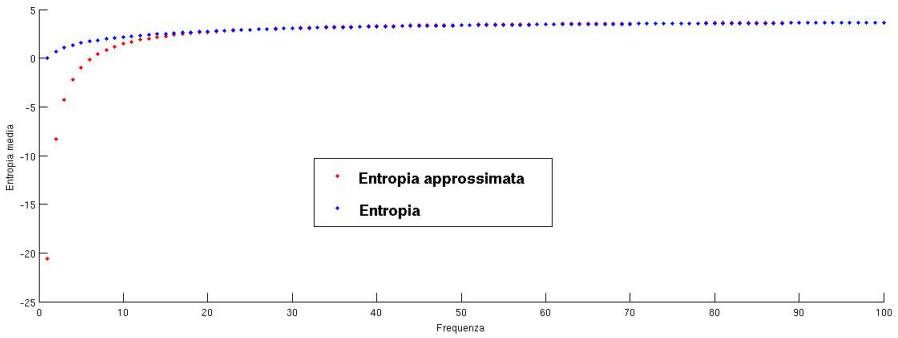

Figura 1.1: Entropia media esatta ed approssimata per un testo di 3000 parole suddiviso in P=50 parti.

1.3 Applicazione

Vediamo ora l'applicazione di quanto detto fin'ora a due testi reali, il "Manifesto del Partito Comunista" e "Alice Nel Paese delle Meraviglie". Nelle tabelle seguenti sono elencate le parole più rilevanti (cioè quelle la cui entropia si discosta maggiormente dall'entropia media calcolata su tutte le versioni casuali possibili del testo) insieme alla loro frequenza nel testo. Oltre all'analisi sull'entropia viene riportato successivamente la stessa analisi effettuata però sui testi ripuliti dalle cosiddette "stopwords", ossia tutte quelle parole (congiunzioni , avverbi, articoli, preposizioni, ecc...) che non hanno valore semantico e possono perciò essere considerate irrilevanti nella caratterizzazione di uno scritto. L'obiettivo è quello di confrontare il testo originale con quello privato delle stopwords per verificare se la modifica influisce sui risultati ottenuti.

1.3.1 Software utilizzato

Per l'analisi entropica ho utilizzato due programmi da me implementati in linguaggio C; il primo serve semplicemente ad eliminare le stopwords dal testo in considerazione, nonchè la punteggiatura, le maiuscole ed eventuali caratteri speciali, mentre il secondo effettua l'analisi dell'entropia delle parole nel testo. Il codice di quest'ultimo è riportato interamente nell'appendice A. Questi due programmi presentano un limite di tipo informatico, poichè non possono analizzare testi al di sopra di una certa dimensione (all'incirca 2 Megabyte); per tale motivo ho utilizzato per la sperimentazione testi relativamente limitati, il che non risulta tuttavia riduttivo in quanto le parole assumono diversa importanza in base alla distribuzione con cui occorrono nel testo, il che non dipende dalla lunghezza del testo stesso. Tali limiti possono essere realisticamente risolti attraverso una più efficiente allocazione della memoria utilizzata, requisito fondamentale data la grande mole di dati trattati.

1.3.2 Analisi quantitativa dell'impatto delle Stopwords

Vogliamo qui prendere in esame quanto eliminare artificialmente le stopwords dal testo originale influisca sulla graduatoria delle parole di maggior rilevanza estratte dal testo stesso, in particolare quindi cerchiamo un parametro che quantifichi la differenza della graduatoria ottenuta esaminando il testo originario e quella ottenuta dal testo ripulito dalle stopwords. Consideriamo rilevante anche il fatto che le parole in graduatoria cambino posizione relativa rispetto ad altre parole,oltre al fatto che siano ancora presenti o meno dopo l'avvenuta scrematura.

Consideriamo una graduatoria di n parole delle quali s siano stopwords. Affinchè il risultato sia il migliore possibile, ci si aspetta che eliminando le s stopwords, le prime $n-s$ parole della nuova graduatoria siano esattamente le stesse che comparivano nella graduatoria originaria. Se questo accadesse, sappiamo che per ogni stopwords eliminata le parole successive nella graduatoria "avanzerebbero" di una posizione nella classifica. Possiamo quindi quantificare gli avanzamenti totali a cui ogni parola (non stopword) è sottoposta dopo l'eliminazione delle stopwords:

$$E_{max} = \sum_{i=1}^{s} n - w_i - (s - i)$$

dove con w_i indichiamo la posizione in graduatoria della stopword i-esima, e consideriamo le stopword ordinate dalla più avanzata in graduatoria ($i = 1$) fino alla più arretrata ($i = s$).
La sommatoria indica che solo le parole successive alla stopword considerata, che sono $n-w_i$ subiscono un avanzamento. Togliamo inoltre $s-i$ cioè le stopwords rimanenti, poichè il loro avanzamento non ha rilevanza. Sviluppando la sommatoria otteniamo:

$$E_{max} = \sum_{i=1}^{s} n - \sum_{i=1}^{s} w_i - \sum_{i=1}^{s} s + \sum_{i=1}^{s} i = sn - \sum_{i=1}^{s} w_i - s^2 + \frac{s(s+1)}{2}$$

Infine:

$$E_{max} = sn - \sum_{i=1}^{s} w_i - \frac{s}{2}(s+1)$$

Non è possibile chiaramente ottenere un'espressione generale della sommatoria nell'espressione, poichè dipende dalla graduatoria delle stopwords.

Rimane ora soltanto da analizzare la variazione effettiva delle parole nella classifica prima e dopo l'abolizione delle stopwords. Per fare ciò, definite w_{j_i} e w_{j_f} rispettivamente la posizione della parola $j-esima$ nella classifica originaria e in quella modificata, possiamo esprimere la variazione totale effettiva di graduatoria come:

$$E_r = \sum_{j=1}^{m} w_{j_i} - w_{j_f}$$

Si osserva che ogni termine della graduatoria è positivo se la parola avanza di posizione, negativo se retrocede, nullo se resta nello stesso posto. Poniamo infine la variazione 0 sia nel caso che una parola scompaia dalla graduatoria, sia che compaia solo nella graduatoria finale, cioè nel caso che w_{j_i} e w_{j_f} non siano noti. Il calcolo dell'ultima sommatoria non è generalizzabile ma va svolto caso per caso empiricamente.

Per concludere, definiamo l'efficienza del metodo come il rapporto tra l'efficienza massima E_{max} e l'efficienza reale E_r:

$$E = \frac{E_r}{E_{max}}$$

1.3.3 Manifesto del Partito Comunista

Il "Manifesto del Partito Comunista" è un testo relativamente breve, composto da un totale di 10.902 parole, di cui 2.488 distinte. L'analisi è stata compiuta con una suddivisione in 20 parti del testo. Vediamo la classifica delle prime 49 parole:

Parola	Frequenza	Parola	Frequenza
proprietà	55	capitale	21
ha	61	dell	60
proletariato	65	industria	32
società	72	famiglia	8
lavoro	49	prodotti	15
comunisti	30	forze	11
socialismo	25	d	31
contro	46	partito	12
francese	16	movimento	24
operai	27	gli	47
i	151	classe	68
loro	66	lotta	30
delle	64	nel	41
il	254	operaio	11
sono	45	abolizione	15
borghese	58	sempre	27
essi	22	classi	32
idee	19	questa	17
comunanza	8	salariato	12
donne	9	era	17
quindi	17	produttive	9
più	88	così	27
produzione	53	comunismo	14
non	113	sociale	21
rapporti	37		

Vediamo adesso come varia la classifica esaminando il testo privato di tutte le stopwords:

Parola	Frequenza	Parola	Frequenza
proprietà	55	industria	32
comunisti	30	mezzi	17
proletariato	65	classe	68
lavoro	49	salariato	12
socialismo	25	famiglia	8
società	72	prodotti	15
produzione	53	partito	12
francese	16	proletari	14
operai	27	rapporti	37
borghese	58	forze	11
lotta	30	abolizione	15
comunanza	8	traffico	9
idee	19	operaio	11
donne	9	movimento	24
capitale	21	comunismo	14

In figura 1.2 è visualizzato il risultato tramite l'applicazione *Wordle* [1].

[1] http://www.wordle.net/

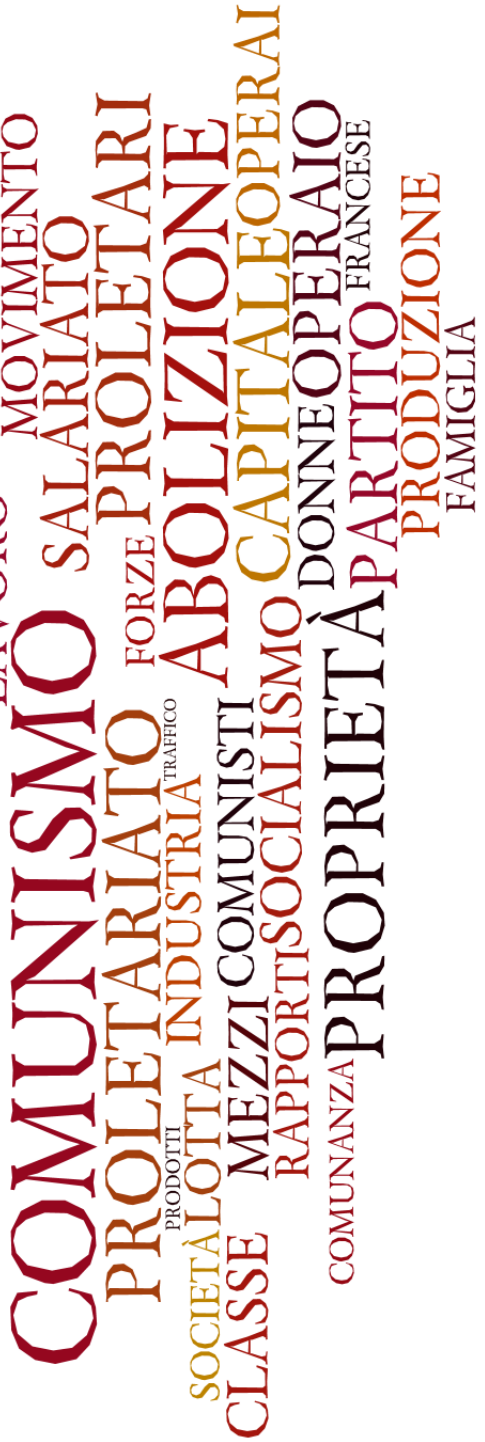

Figura 1.2: Visualizzazione delle parole rilevanti estratte dal Manifesto del Partito Comunista

La seconda tabella mostra quello che si ottiene togliendo le 19 stopwords dalla classifica generale delle prime 49. Analizziamo meglio cos'è avvenuto alle parole che erano nella classifica generale. Naturalmente siamo interessati a verificare che l'artificio di eliminare "manualmente" le stopwords non infici sulla classifica, o che perlomeno non ne modifichi le peculiarità e i tratti salienti.
Le stopwords eliminate, come già detto, sono 19:

`ha, contro, i, loro, delle, il, sono, essi, quindi, più, non, dell, d, gli, nel, questa, era, così, sempre.`

Delle 30 parole rimaste, 28 erano già presenti nella classifica originaria mentre 2 sono scomparse:

`classi, produttive`

e 2 sono le nuove entrate:

`proletari, mezzi`

Proviamo ora a calcolare l'efficienza. Le parole che non cambiano posizione sono 2, "proprietà" e "proletariato"; solo una retrocede, "società", due scompaiono dalla graduatoria e due subentrano, come già osservato. TUtte le altre avanzano di posizione, quindi ci si aspetta che l'"efficienza" abbia un valore abbastanza alto. L'efficienza reale, ottenuta sommando la differenza di posizione per ogni termine, vale:

$$E_r = \sum_{j=1}^{m} w_{j_i} - w_{j_f} = 236$$

mentre quella massima:

$$E_{max} = sn - \sum_{i=1}^{s} w_i - \frac{s}{2}(s-1) = 931 - 478 - 171 = 282$$

L'efficienza sarà quindi:

$$E = \frac{E_r}{E_{max}} = \frac{236}{282} = 0,83$$

cioè un'efficienza dell'83%, piuttosto alta come ci si aspettava.

1.3.4 Alice nel Paese delle Meraviglie

Ripetiamo l'analisi con un altro testo, "Alice nel Paese delle Meraviglie". Il testo è composto da 22.489 parole, di cui 4.208 distinte ed è stata utilizzata una suddivisione in $P = 30$ parti. Vediamo la classifica delle prime 50 parole:

Parola	Frequenza	Parola	Frequenza
testuggine	57	della	76
cappellaio	57	giù	22
falsa	54	io	62
grifone	51	te	102
re	63	rispose	100
topo	44	c	49
regina	79	cuoca	12
bruco	30	di	559
ghiro	38	più	115
il	651	un	399
duchessa	42	morale	9
zuppa	20	nuovo	30
gatto	32	testa	51
marzo	33	erano	33
la	585	loro	35
lepre	31	i	135
disse	285	cosa	63
bella	23	era	149
valletto	14	le	196
sono	61	gatti	13
guglielmo	17	statura	17
coniglio	49	vuoi	20
colombo	12	pozzo	10
dronte	13	giardinieri	9
giurati	22	senti	22

Il testo ripulito dalle stopwords (17 in tutto) è composto da 11.547 parole di cui 3.877 distinte. Vediamo la classifica delle prime 33 parole:

Parola	Frequenza	Parola	Frequenza
cappellaio	57	dronte	13
testuggine	57	giurati	22
falsa	54	giù	22
grifone	51	rispose	100
re	63	gatti	13
regina	79	dina	14
topo	44	cara	20
bruco	30	testa	51
ghiro	38	domandò	38
gatto	32	nuovo	30
zuppa	21	vuoi	20
duchessa	42	giardinieri	9
lepre	31	serpente	8
bella	23	carnefice	8
disse	286		
valletto	14		
coniglio	49		
guglielmo	17		
colombo	12		

In figura 1.3 è visualizzato con *Wordle*il risultato delle parole estratte.

Figura 1.3: Visualizzazione delle parole rilevanti estratte dal Alice nel Paese delle Meraviglie

Le stopwords eliminate, come già detto, sono 17:

`il, marzo, la, sono, della, io, te, c, di, più, un, erano, loro, i, cosa, era, le`

Delle 33 parole rimaste, 28 erano già presenti nella classifica originaria mentre 5 sono scomparse:

`cuoca, morale, statura, pozzo, senti`

e 5 sono le nuove entrate:

`dina, cara, domandò, serpente, carnefice`

Proviamo ora a calcolare l'efficienza. Le parole che non cambiano posizione sono 2, "proprietà" e "proletariato"; solo una retrocede, "società", due scompaiono dalla graduatoria e due subentrano, come già osservato. TUtte le altre avanzano di posizione, quindi ci si aspetta che l'"efficienza" abbia un valore abbastanza alto. L'efficienza reale, ottenuta sommando la differenza di posizione per ogni termine, vale:

$$E_r = \sum_{j=1}^{m} w_{j_i} - w_{j_f} = 118$$

mentre quella massima:

$$E_{max} = sn - \sum_{i=1}^{s} w_i - \frac{s}{2}(s-1) = 850 - 524 - 136 = 190$$

L'efficienza sarà quindi:

$$E = \frac{E_r}{E_{max}} = \frac{118}{190} = 0,62$$

cioè un'efficienza dell'62%. Osserviamo che il risultato, alla luce di quanto visto per l'altro testo, ha senso poichè eliminando le stopwords sono scomparse 5 parole rispetto alle 2 del testo precedente; è quindi normale aspettarsi un rendimento minore.

Capitolo 2

Analisi della Semantica Latente

L' Analisi della Semantica Latente (d'ora in poi la chiameremo LSA ovvero Latent Semantic Analysis) nasce dall'esigenza di migliorare il processo con cui vengono associate le parole di una ricerca con i documenti oggetto della ricerca in questione. Il problema consiste nel fatto che normalmente gli utenti ricercano i documenti sulla base di concetti, mentre le parole inserite per effettuare la ricerca non forniscono un evidenza dell'argomento del testo in cui sono contenute. Solitamente esistono più modi per esprimere un dato concetto, quindi i termini immessi per una ricerca possono non essere utilizzati del tutto nel dicumento che si sta cercando. Inoltre, molte parole hanno più significati quindi i termini di una ricerca possono comparire in un documento che non ha nulla a che vedere con il documento che stiamo cercando.

L'analisi dell'LSA opera sotto la premessa che **esiste una certa struttura semantica latente che è parzialmente nascosta dalla casualità delle scelta delle parole**. Per evidenziare questa struttura vengono perciò utilizzate tecniche algebriche e statistiche che hanno la capacità di eliminare il rumore. Il risultato è una descrizione parametrica continua di termini e

documenti basata sulla struttura soggiacente.
In pratica, la struttura latente è trasmessa attraverso schemi di correlazione, derivanti dal modo in cui le parole appaiono nei documenti; questo porta a un elementare modello del linguaggio utilizzato descritto da un ristretto insieme di parole. Inoltre, coerentemente con la natura dell'approccio di tipo quantitativo, semantica significa semplicemente che **le parole di un documento possano essere considerate come indicatrici di un documento o del suo argomento**. Nonostante queste semplificazioni, la descrizione che se ne ottiene è indubbiamente vantaggiosa. Sicuramente approfitta della struttura di ordine superiore nell'associazione delle parole con i documenti e di conseguenza migliora il riconoscimento dei documenti di maggior rilevanza sulla base dei termini che compaiono nella ricerca.

2.1 LSA

Supponiamo $\mathcal{M}$ sia un inventario di M unità (ad esempio le parole di un vocabolario) e $\mathcal{N}$ una collezione di N combinazioni di tali unità (un corpus di documenti o testi letterari). Lo scopo dell'LSA è di definire una mappatura tra gli insiemi discreti $\mathcal{M}$ e $\mathcal{N}$ e uno spazio vettoriale continuo $\mathcal{L}$, nel quale ogni parola r_i di $\mathcal{M}$ sia rappresentata da un vettore $\overline{u}_i$ in $\mathcal{L}$ e ogni testo c_j in $\mathcal{N}$ sia rappresentata da un vettore $\overline{v}_j$ in $\mathcal{L}$.
Come prima cosa si costruisce una matrice W con le co-occorrenze tra le parole e i testi del corpus. Il posto $w_{i,j}$ è ottenuto con la normalizzazione in base alla lunghezza dei testi e all'entropia delle parole:

$$w_{i,j} = (1 - \epsilon_i)\frac{k_{i,j}}{\lambda_j}$$

dove $k_{i,j}$ è il numero di volte che r_i compare in c_j, λ_j è il numero totale di parole che compaiono in c_j (normalizzazione della lunghezza) e ϵ_i è l'entropia normalizzata di r_i all'interno dell'intero corpus (normalizzazione dell'entropia). Per definizione di entropia, se $\tau_i = \sum_j k_{i,j}$ è il numero totale di occorrenze della i-esima parola nel corpus, risulta:

$$\epsilon_i = -\frac{1}{\log N} \sum_{j=1}^{N} \frac{k_{i,j}}{\tau_i} \log \frac{k_{i,j}}{\tau_i}$$

Il peso globale dato dalla quantità $1 - \epsilon_i$ indica il fatto che due unità che appaiono lo stesso numero di volte in c_j non apportano necessariamente la stessa quantità di informazione. Un valore di ϵ_i vicino ad 1 indica che la parola è distribuita uniformemente nel corpus, quindi porta con sè una minor informazione di un'altra con ϵ_i più vicino allo 0. Si può riassumere perciò dicendo che il peso globale $1 - \epsilon_i$ è una misura del "potere di indicizzazione" della parola r_i.

2.1.1 Analisi delle Componenti Principali (PCA)

L'analisi delle componenti principali deriva direttamente dalla teoria dell'analisi fattoriale, che consiste in un insieme di tecniche statistiche che consentono di ottenere una riduzione della complessità del numero di fattori che spiegano un fenomeno. Si propone quindi di determinare un certo numero di variabili "**latent**" (cioè fattori non direttamente misurabili nella realtà) più ristretto e riassuntivo rispetto al numero di variabili di partenza.
Si pensi, ad esempio, all'insieme dei voti di una classe di studenti in una scuola. I voti riguardano i rendimenti degli studenti in diverse materie (italiano, matematica, scienze, geografia, storia, ecc...). Gli studenti possono perciò essere considerati "variabili" che assumono valori diversi (i voti) sulle diverse

materie. Per semplificare il problema, è lecito supporre che le capacità di apprendimento degli studenti possano distinguersi in due fattori: materie umanistiche e materie scientifiche.
Grazie all'analisi fattoriale è possibile misurare queste due abilità attraverso la costruzione di due **variabili latenti di sintesi** (intesa come combinazione lineare) delle variabili originarie, ognuna pensata sulla base dell'importanza nel discriminare gli individui sulla base delle loro abilità scientifiche e umanistiche.
L'analisi delle componenti principali è perciò un metodo fattoriale per la sintesi di "p" variabili quantitative, tra loro correlate, attraverso l'identificazione di $h < p$ **variabili latenti** (non osservate) dette **componenti principali** che godono di due proprietà:

- sono tra loro *non correlate* (ortogonali) e legate linearmente alle variabili di partenza;
- sono determinate in ordine crescente rispetto alla percentuale di variabilità.

Per semplificare supponiamo di rappresentare sul piano cartesiano (figura 2.1) i punti-unità le cui coordinate sono i valori standardizzati delle due variabili. Con l'ACP si identifica **l'asse fattoriale nella direzione di massima variabilità** della nube dei punti-unità, in modo tale da deformare il meno possibile la distanza reciproca tra i punti una volta proiettati su tale asse: si minimizza cioè la somma delle distanze dei punti dall'asse (AB), che equivale per il teorema di pitagora a massimizzare la somma delle proiezioni dei punti sull'asse (OB) ovvero la distanza dei punti proiettati sull'asse dall'origine.

Derivazione analitica Supponiamo di avere una matrice di dati $X(n \times p)$:

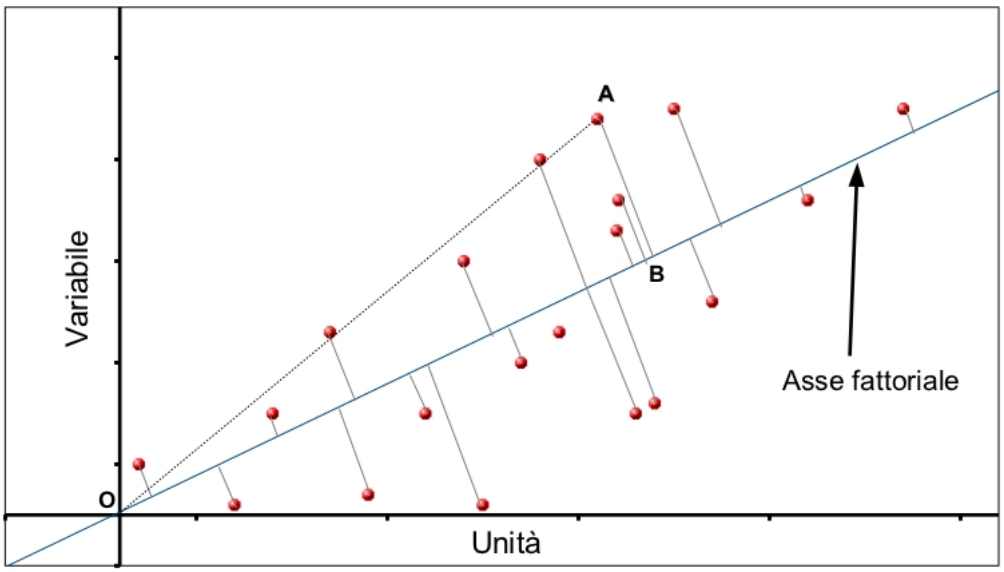

Figura 2.1: Sintesi di $p = 2$ variabili attraverso $h = 1$ asse fattoriale

$$X = \begin{pmatrix} x_{11} & \dots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{np} \end{pmatrix}$$

I vettori riga di X sono **punti-unità** nello spazio $\mathcal{R}^p$ generato dalle variabili, i vettori colonna invece sono **punti-variabile** nello spazio $\mathcal{R}^n$ generato dalle unità.

Voglio proiettare gli n vettori (d'ora in poi li chiameremo x_i, con $1 \leq i \leq n$) in $\mathcal{R}^p$ su una retta r_1, che sarà la prima componente principale, che viene identificata con il versore u_1: chiamiamo θ l'angolo formato da x_i e u_1, e OH_i la proiezione dell'i-esima unità x_i su r_1. Abbiamo che:

$$OH_i = x_i^{\mathrm{T}} \cdot u_1$$

Tale ragionamento può essere fatto per ognuno degli n punti, definendo così una matrice di proiezioni data da:

$$X \cdot u_1$$

Rimane da capire come scegliere la retta r_1. Per fare ciò si utilizza il metodo dei minimi quadrati, quindi si prende quella retta tale che minimizzi i quadrati delle distanze dei punti dalla retta; questo equivale, per il teorema di Pitagora, a massimizzare la quantità:

$$\sum_{i=1}^{n} OH_i^2 = (Xu_1)^{\mathrm{T}} \cdot Xu_1 = u_1^{\mathrm{T}} X^{\mathrm{T}} X u_1$$

Praticamente devo massimizzare $u_1^{\mathrm{T}} X^{\mathrm{T}} X u_1$ con il vincolo che u_1 abbia norma unitaria: risolvo il tutto con i moltiplicatori di Lagrange, definendo la funzione

$$F(u_1) = u_1^{\mathrm{T}} X^{\mathrm{T}} X u_1 - \lambda_1 (u_1^{\mathrm{T}} u_1 - 1)$$

Derivando rispetto a u_1 ottengo:

$$2X^{\mathrm{T}} X u_1 - 2\lambda_1 u_1 = 0$$

cioè

$$X^{\mathrm{T}} X u_1 = \lambda_1 u_1$$

Questo è un semplice problema agli autovalori, e visto che sto massimizzando avrò allo stesso tempo che u_1 è l'autovettore relativo al più grande autovalore della matrice $X^{\mathrm{T}} X$. Definito u_1 ho definito la prima componente principale:

$$r_1 = Xu_1 = v_1 u_{1_1} + \cdots + v_p u_{1_p}$$

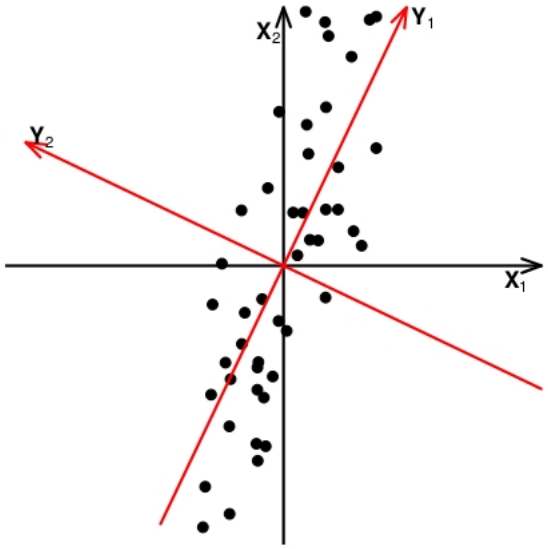

Figura 2.2: Le due componenti principali, in rosso: la prima è nella direzione di massima dispersione, la seconda è ottenuta univocamente ponendo il vincolo di essere ortogonale alla seconda.

Dove le v_i sono le variabili iniziali (da non confondere con le unità x_i) cioè i vettori colonna della matrice X, e gli u_{1_i} con $1 \leq i \leq n$ sono gli elementi i-esimi del versore u_1.

Inoltre la norma al quadrato di r_1 è:

$$\|r_1\| = u_1^{\mathrm{T}} X^{\mathrm{T}} X u_1 = \lambda_1$$

Cioè la norma della prima direzione principale coincide con l'autovalore corrispondente alla direzione del versore u_1 (che è l'autovettore corrispondente).

Abbiamo ottenuto il seguente risultato: *la prima componente principale r_1 è una combinazione lineare delle p colonne della matrice X con*

coefficienti uguali alle componenti dell'autovettore u_1, associato al massimo autovalore della matrice $X^{\mathrm{T}}X$.
Le successive componenti principali si ricavano nello stesso modo, col vincolo che siano ortogonali alle precedenti, ovvero che la variabile aleatoria corrispondente alla direzione principale ricavata sia *scorrelata* dalle precedenti.

Osservazioni La matrice $S = X^{\mathrm{T}}X$ è una matrice simmetrica. Cosa sappiamo sugli autovalori?

- $tr(S) = \sum \lambda_i$
- $det(S) = \prod \lambda_i$
- Gli autovalori di una matrice simmetrica sono reali;
- Gli autovettori di una matrice simmetrica sono a due a due ortogonali;
- Il rango di una matrice simmetrica è uguale al numero dei suoi autovalori non nulli.

Interpretazione delle componenti principali Come già accennato, l'ACP viene spesso utilizzata per cercare di studiare variabili "latenti". In tal senso le "variabili artificiali" a cui dà luogo rappresenterebbero misurazioni di variabili "nascoste", non osservabili direttamente. Altre volte l'ACP viene utilizzata come metodo per "riassumere" i dati a disposizione. Tuttavia il significato da attribuire alle componenti principali non è univoco am si basa su un interpretazione che può essere di volta in volta differente sulla base di elementi ambientali esterni; entrano quindi in gioco l'esperienza e la sensibilità di chi ha il compito di attribuire loro un significato.
Detto questo, alcune osservazioni possono essere fatte. Per prima cosa la

j-esima componente principale r_j è definita come combinazione lineare delle variabili $v_1, v_2, \ldots v_p$ con coefficienti $u_{j_1}, u_{j_2}, \ldots u_{j_p}$. Ovvero:

$$r_{ij} = u_{j_1} v_{i_1}, u_{j_2} v_{i_2}, \ldots u_{j_p} v_{i_p}$$

quindi:

$$y_j = X u_j$$

Pertanto il generico coefficiente u_{j_h} rappresenta il peso che la variabile v_h ha nella determinazione della componente principale r_j $(h = 1, \ldots, p)$; quanto più grande è u_{j_h} in valore assoluto, tanto maggiore sarà il peso che i valori v_{i_h} $(i = 1, \ldots, n)$ hanno nel determinare r_{ij}.
Ciò significa che la componente principale r_j sarà maggiormente caratterizzata dalle variabili v_h a cui corrispondono i coefficienti u_{j_h} più grandi in valore assoluto. In tal modo sono proprio i coefficienti u_{j_h} a conferire un significato alla componente principale r_j.

2.1.2 Singular Value Decomposition (SVD)

Il passo successivo alla costruzione della matrice $(M \times N)$ W è la sua scomposizione con il metodo della "Singular Value Decomposition" (SVD), una tecnica molto simile alla ricerca degli autovettori e autovalori per le matrici quadrate, e all'analisi dei risultanti fattori.
La scomposizione (di ordine R) è data da:

$$W \approx \hat{W} = U S V^{\mathrm{T}} \tag{2.1}$$

dove U è la matrice singolare sinistra $(M \times R)$ con u_i come vettori colonna $(1 \leqslant i \leqslant M)$, S è la matrice diagonale $(R \times R)$ dei valori singolari (l'analogo

degli autovalori delle matrici quadrate) $s_1 \geqslant s_2 \geqslant \cdots \geqslant s_R > 0$, V è la matrice singolare destra $(N \times R)$ con v_j come vettori riga, $(1 \leqslant j \leqslant N)$. Infine $R < min(M, N)$ è l'ordine della scomposizione.
Entrambe le matrici destra e sinistra U e V sono ortonormali sulle colonne, cioè $U^{\mathrm{T}}U = V^{\mathrm{T}}V = I_R$. Per questo i vettori colonna di U e V definisco una base ortonormale per lo spazio vettoriale di dimensione R (quello che abbiamo definito come $\mathcal{L}$ ovvero lo spazio LSA) generato dai vettori u_i e v_j. La matrice $\hat{W}$ viene definita come la matrice di rango R che meglio approssima in norma W, per ogni norma invariante per trasformazioni unitarie (cioè tale che $\|A\| = \|UAV\|$ per ogni matrice A con U e V unitarie). Ciò significa, per ogni matrice A di rango R:

$$\min_{\{A:rank(A)=R\}} \|W - A\| = \|W - \hat{W}\| = s_{R+1}$$

dove $\| \bullet \|$ è la norma in L_2, e s_{R+1} è il più piccolo valore singolare rimasto nella scomposizione di ordine $(R + 1)$ di W.

2.1.3 Relazione tra SVD e PCA

Intuitivamente, l'Analisi delle Componenti Principali richiede che vengano calcolati autovettori e autovalori della matrice di covarianza XX^{T}, dove con X indichiamo la matrice iniziale dei dati. Poichè la matrice di covarianza è simmetrica per costruzione, è diagonalizzabile, e gli autovettori possono essere normalizzati così da risultare ortonormali:

$$XX^{\mathrm{T}} = WDW^{\mathrm{T}}$$

D'altra parte, applicando la Singular Value Decomposition alla matrice dei dati X possiamo scrivere:

$$X = USV^{\mathrm{T}}$$

e costruendo la matrice di covarianza utilizzando la precedente scomposizione abbiamo:

$$XX^{\mathrm{T}} = (USV^{\mathrm{T}})(USV^{\mathrm{T}})^{\mathrm{T}}$$

che, utilizzando il fatto che V sia ortonormale, dà:

$$XX^{\mathrm{T}} = US^2U^{\mathrm{T}}$$

La corrispondenza tra le due metodologie è evidente: le radici quadrate degli autovalori di XX^{T} sono i valori singolari di X, con tutto ciò che ne consegue.
Di fatto, usare la SVD per effettuare la PCA è numericamente molto più conveniente, poichè non necessita di calcolare la matrice di covarianza XX^{T} all'inizio della procedura, problema che può risultare non ben posto in alcuni casi.

2.1.4 Spazio LSA

La scomposizione (2.1) può essere interpretata come una rappresentazione di ogni parola e di ogni concetto come combinazione lineare di concetti astratti (nascosti), i quali generano lo spazio lineare di W. Quindi, la mappatura risultante corrisponde ad un'efficiente rappresentazione dei dati empirici. L'idea che sta alla base del procedimento è che $\hat{W}$ contenga in sè le associazioni strutturali più rilevanti di W e ignori gli effetti di ordine superiore (il rumore). La vicinanza tra i vettori in $\mathcal{L}$ è quindi determinata dallo schema generale del linguaggio delle composizioni in $\mathcal{N}$. La mappatura $(\mathcal{M}, \mathcal{N}) \longmapsto \mathcal{L}$

permette infine di applicare tecniche algoritmiche note, nello spazio vettoriale continuo $\mathcal{L}$. Per fare ciò è tuttavia necessario definire prima su $\mathcal{L}$ un'opportuna metrica che sia anche adeguata al formalismo della SVD.
Osserviamo che la misura con cui le unità r_i e r_j hanno uno schema simile all'interno dela totalità del corpus può essere dedotta dalla cella (i, j) della matrice WW^{T}; allo stesso modo la misura in cui due testi c_i e c_j contengono schemi di parole simili al loro interno può essere osservato nella posizione (i, j) della matrice $W^{\mathrm{T}}W$; infine la misura in cui la parola r_i è globalmente legata al testo c_j rispetto all'intero corpus può essere osservato dal posto (i, j) della matrice W stessa. Tutto questo porta alla seguente analisi riguardante parole e testi.

Confronto di due parole: Poichè $WW^{\mathrm{T}} = US^2U^{\mathrm{T}}$ e S è diagonale, la cella (i, j) di WW^{T} può essere ottenuta consideranto il prodotto interno tra la i-esima e j-esima riga della matrice US; definiamo $\overline{u}_i = u_iS$ e $\overline{u}_j = u_jS$. Per misurare la vicinanza tra 2 parole viene spontaneo considerare la metrica definita dal coseno dell'angolo compreso tra i due vettori:

$$K(r_i, r_j) = \cos(u_iS, u_jS) = \frac{u_iSu_j^{\mathrm{T}}}{\|u_iS\|\|u_jS\|} \tag{2.2}$$

per $1 \leq i, j \leq M$. $K(r_i, r_j) = 1$ se due parole compaiono sempre nello stesso tipo di testi, mentre $K(r_i, r_j) \leq 1$ se sono utilizzate in contesti diversi. Nonostante (2.2) non costituisca una distanza ben definita, a partire da essa è facile definirne una. Per esempio, nell'intervallo $[\pi, 2\pi]$, $\mathcal{D}(r_i, r_j) = \cos^{-1} K(r_i, r_j)$ soddisfa le proprietà di una distanza sullo spazio $\mathcal{L}$.

Confronto di due testi: Per le stesse ragioni appena esposte consideriamo la funzione

$$K(c_i, c_j) = \cos(v_i S, v_j S) = \frac{v_i S v_j^{\mathrm{T}}}{\|v_i S\| \|v_j S\|} \tag{2.3}$$

per $1 \leq i, j \leq N$; allo stesso modo, $K(i, j) = 1$ se due testi contengono le stesse parole mentre $K(i, j) \leq 1$ al diminuire delle parole in comune.

Confronto di parole e testi: Infine, essendo $W = USV^{\mathrm{T}}$

$$K(r_i, c_j) = \cos(u_i S^{1/2}, v_j S^{1/2}) = \frac{u_i S v_j^{\mathrm{T}}}{\|u_i S^{1/2}\| \|v_j S^{1/2}\|} \tag{2.4}$$

per $1 \leq i \leq M$ e $1 \leq j \leq N$. In questo caso, $K(r_i, c_j) < 1$ man mano che la correlazione tra la parola r_i e il testo c_j all'interno del corpus cala.

2.1.5 Un esempio

Vediamo ora un esempio di applicazione, tratto da [5].
Consideriamo nove titoli, i primi cinque dei quali riguardanti l'interazione uomo-computer e gli altri quattro riguardanti la teoria dei grafici. Nella tabella seguente sono riportati i titoli; le parole sottolineate sono quelle che verranno prese in considerazione per l'analisi.

c1	Human machine interface for ABC computer applications
c2	A survey of user opinion of computer system response time
c3	The EPS user interface managment system
c4	System and human system engineering testing of EPS
c5	Relation of user perceived response time to error measurment
m1	The generation of random, binary, ordered trees
m2	The intersection graph of paths in trees
m3	Graph minors IV: Widths of trees and well-quasi-ordering
m4	Graph minors: A survey

Di seguito è riportata la matrice parole-titoli X, formata dai 5 articoli riguardanti l'interazione uomo-computer e quattro sulla teoria dei grafici. Ogni posizione è occupata dal numero di volte che una parola (riga) compare in un titolo (colonna) per ogni parola che compare in almeno due titoli:

$$\{X\} =$$

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

$$\{X\} = \{W\}\{S\}\{P\}'$$

$\{W\} =$

0.22	-0.11	0.29	-0.41	-0.11	-0.34	0.52	-0.06	-0.41
0.20	-0.07	0.14	-0.55	0.28	0.50	-0.07	-0.01	-0.11
0.24	0.04	-0.16	-0.59	-0.11	-0.25	-0.30	0.06	0.49
0.40	0.06	-0.34	0.10	0.33	0.38	0.00	0.00	0.01
0.64	-0.17	0.36	0.33	-0.16	-0.21	-0.17	0.03	0.27
0.27	0.11	-0.43	0.07	0.08	-0.17	0.28	-0.02	-0.05
0.27	0.11	-0.43	0.07	0.08	-0.17	0.28	-0.02	-0.05
0.30	-0.14	0.33	0.19	0.11	0.27	0.03	-0.02	-0.17
0.21	0.27	-0.18	-0.03	-0.54	0.08	-0.47	-0.04	-0.58
0.01	0.49	0.23	0.03	0.59	-0.39	-0.29	0.25	-0.23
0.04	0.62	0.22	0.00	-0.07	0.11	0.16	-0.68	0.23
0.03	0.45	0.14	-0.01	-0.30	0.28	0.34	0.68	0.18

$\{S\} =$

3.34								
	2.54							
		2.35						
			1.64					
				1.50				
					1.31			
						0.85		
							0.56	
								0.36

$\{P\} =$

0.20	0.61	0.46	0.54	0.28	0.00	0.01	0.02	0.08
-0.06	0.17	-0.13	-0.23	0.11	0.19	0.44	0.62	0.53
0.11	-0.50	0.21	0.57	-0.51	0.10	0.19	0.25	0.08
-0.95	-0.03	0.04	0.27	0.15	0.02	0.02	0.01	-0.03
0.05	-0.21	0.38	-0.21	0.33	0.39	0.35	0.15	-0.60
-0.08	-0.26	0.72	-0.37	0.03	-0.30	-0.21	0.00	0.36
0.18	-0.43	-0.24	0.26	0.67	-0.34	-0.15	0.25	0.04
-0.01	0.05	0.01	-0.02	-0.06	0.45	-0.76	0.45	-0.07
-0.06	0.24	0.02	-0.08	-0.26	-0.62	0.02	0.52	-0.45

Figura 2.3: La SVD completa della matrice X

Di seguito è riportata la matrice $\hat{X}$, che è una ricostruzione bidimensionale della matrice X basata sulle righe e colonne più scure della SVD (figura 2.3). Confrontando le righe oscurate e cerchiate delle matrici X e $\hat{X}$ si nota come l'LSA induca delle relazioni di similarità aumentando o diminuendo i valori per riordinare i legami reciproci tra i dati.

$\{\hat{X}\} =$

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	0.16	0.40	0.38	0.47	0.18	-0.05	-0.12	-0.16	-0.09
interface	0.14	0.37	0.33	0.40	0.16	-0.03	-0.07	-0.10	-0.04
computer	0.15	0.51	0.36	0.41	0.24	0.02	0.06	0.09	0.12
user	0.26	0.84	0.61	0.70	0.39	0.03	0.08	0.12	0.19
system	0.45	1.23	1.05	1.27	0.56	-0.07	-0.15	-0.21	-0.05
response	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
time	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
EPS	0.22	0.55	0.51	0.63	0.24	-0.07	-0.14	-0.20	-0.11
survey	0.10	0.53	0.23	0.21	0.27	0.14	0.31	0.44	0.42
trees	-0.06	0.23	-0.14	-0.27	0.14	0.24	0.55	0.77	0.66
graph	-0.06	0.34	-0.15	-0.30	0.20	0.31	0.69	0.98	0.85
minors	-0.04	0.25	-0.10	-0.21	0.15	0.22	0.50	0.71	0.62

Osserviamo le due caselle in grigio nella colonna $m4$, corrispondenti alle righe di *survey* e *trees*. La parola *trees* non compariva nel titolo $m4$ della teoria dei grafici, ma poichè contenteva *graph* e *minors*, il valore 0 di *trees* è stato sostituito con 0.66, che può essere interpretato come **quante volte comparirebbe in un insieme infinito di titoli contenenti *graph* e *minors***. All'opposto, il valore 1 di *survey*, che appariva una volta in $m4$, è stato rimpiazzato da 0.42, ad indicare che tale parola è inaspettata in quel contesto e va considerata come poco rilevante al fine di caratterizzare il passaggio. In pratica, costruendo la rappresentazione in dimensione ridotta, cioè utilizzando solo i valori lungo due dimensioni ortogonali, si dà una stima di quali parole compaiono in ogni contesto usando solo le informazioni che sono state estratte.

Il che significa dire: "*questo segmento di testo è descritto nel modo migliore utilizzando un tot del concetto astratto 1 e un tot del concetto astratto 2, e questa parola contiene tot del concetto 1 e tot del concetto 2; combinando*

queste due informazioni (con l'aritmetica dei vettori), la mia ipotesi migliore è che la parola X compare 0.6 *volte nel contesto Y*".

Vediamo di analizzare due esempi di relazioni parola-parola: utilizziamo per far ciò le righe oscurate/cerchiate corrispondenti alle parole human, user e minors (in questo contesto *minor* è un termine tecnico della teoria dei grafici) nella matrice originale e in quella ricostruita bidimensionalmente. In quella originale, human non appare mai nello stesso passaggio, sia di user che di minors, non hanno cioè co-occorrenze, contiguità o associazione. La correlazione (utilizzando l'indice di correlazione di Spearman per facilitare l'interpretazione) è di -0.38 tra human e user e 0.29 tra human e minors. Tuttavia, nella matrice ricostruita, come conseguenza delle loro relazioni indirette, entrambi gli indici sono molto alterati: la correlazione human-user sale a 0.94 e quella human-minors scende a $-o.83$. Quindi, poichè i termini *human* e *user* compaiono in contesti di significato simile, anche se mai nello stesso passaggio, la soluzione a dimensione ridotta li rappresenta come più simili; al contrario succede tra *human* e *minors*.

Per esaminare meglio l'effetto prodotto dalla riduzione dimensionale sulle relazioni tra i titoli, calcoliamo le intercorrelazioni tra ogni titolo e gli altri, prima basate sui dati originali (figura 2.4) e poi su quelli ottenuti dalla ricostruzione (figura 2.5). Nei dati originali, le correlazioni tra i 5 titoli HCI è generalmente bassa, anche se i titoli riguardano argomenti simili; metà degli indici sono zero, tre negativi, due positivi, con una media di 0.02. Le correlazioni i quattro titoli sulla teoria dei grafi è mista, con un valore medio dell'indice di correlazione di 0.44. Infine la correlazione tra i titoli sull'HCI e quelli sulla teoria dei grafi (la parte evidenziata in grigio) ha un valore medio

	c1	c2	c3	c4	c5	m1	m2	m3
c2	-0.19							
c3	0.00	0.00						
c4	0.00	0.00	0.47					
c5	-0.33	0.58	0.00	-0.31				
m1	-0.17	-0.30	-0.21	-0.16	-0.17			
m2	-0.26	-0.45	-0.32	-0.24	-0.26	0.67		
m3	-0.33	-0.58	-0.41	-0.31	-0.33	0.52	0.77	
m4	-0.33	-0.19	-0.41	-0.31	-0.33	-0.17	0.26	0.56

Figura 2.4: Correlazione tra i dati nella matrice originaria

c2	0.91							
c3	1.00	0.91						
c4	1.00	0.88	1.00					
c5	0.85	0.99	0.85	0.81				
m1	-0.85	-0.56	-0.85	-0.88	-0.45			
m2	-0.85	-0.56	-0.85	-0.88	-0.44	1.00		
m3	-0.85	-0.56	-0.85	-0.88	-0.44	1.00	1.00	
m4	-0.81	-0.50	-0.81	-0.84	-0.37	1.00	1.00	1.00

Figura 2.5: La correlazione nello spazio bidimensionale

di -0.3, nonostante la poca distanza concettuale tra i due argomenti.

Nella ricostruzione bidimensionale i raggruppamenti per argomento sono molto più evidenti: il coefficiente di correlazione cresce enormemente da 0.02 a 0.92. **Ciò avviene non tanto perchè i titoli HCI fossero particolarmente simili tra loro, la qual cosa è falsa, ma piuttosto perchè erano "in contrasto" con i titoli non-HCI nello stesso modo.** Allo stesso modo, le correlazioni tra i titoli sulla teoria dei grafi prendono nella ricostruzione il valore 1, e quelli tra le 2 classi di titoli diventano molto negativi, con media -0.72.

2.2 Applicazione pratica del metodo

Vediamo ora un'applicazione concreta della Latent Semantic Analysis, che è anche probabilmente quella più comune e che tutti probabilmente conoscono: la ricerca di pagine rilevanti nel web, che senza ottimizzazioni come quella che stiamo trattando risulterebbe poco efficiente e soggetta a tempistiche che la renderebbero inattuabile, dato il numero sconfinato di documenti presenti al giorno d'oggi nel web.
L'algoritmo matlab utilizzato per l'analisi è contenuto nell'appendice B. Per fare ciò ho considerato un numero limitato di pagine internet, in particolare 5 pagine di wikipedia:

- pagina wikipedia sul **rock**
- pagina wikipedia sul cantante **Tom Waits**
- pagina wikipedia del film **Coffee and Cigarettes**
- pagina wikipedia del regista **Jim Jarmusch**
- pagina wikipedia sullo scrittore **Italo Calvino**

Naturalmente l'esperimento può essere effettuato con pagine internet qualsiasi e in numero arbitrario.
In seguito ho scelto, casualmente, alcune parole che ritenevo rilevanti per queste 5 pagine:

1. musica
2. cinema
3. letteratura
4. arte

5. visconte

6. scrivere

7. suonare

8. autore

9. psichedelico

10. band

11. caffè

Naturalmente non esiste alcun vincolo su quali parole scegliere, generalmente un processo automatizzato di Latent Semantic Analysis utilizza tutte le parole presenti in tutti i testi, e può essere ottimizzato estrendo prima quelle maggiormente rilevanti, ma per rendere l'esperimento maggiormente comprensibile consideriamo solo le parole elencate; vedremo in seguito che questo non è necessariamente restrittivo.
Il secondo passo del processo è costruire la matrice delle occorrenze *termini-documenti* come fatto nell'esempio precedente: ogni vettore-riga rappresenta una parola con il numero di occorrenze nei diversi documenti:

$$\mathbf{X} = \begin{pmatrix} 160 & 6 & 2 & 0 & 5 \\ 0 & 0 & 2 & 4 & 2 \\ 1 & 3 & 0 & 2 & 18 \\ 0 & 0 & 0 & 0 & 10 \\ 0 & 0 & 0 & 0 & 8 \\ 2 & 0 & 0 & 0 & 4 \\ 1 & 1 & 0 & 0 & 0 \\ 4 & 0 & 0 & 4 & 4 \\ 5 & 0 & 0 & 0 & 0 \\ 86 & 0 & 0 & 0 & 0 \\ 0 & 5 & 24 & 12 & 0 \end{pmatrix}$$

Il passo successivo è quello di normalizzare la matrice; i metodi possibili sono molti, in questo caso utilizzerò il metodo Tf-idf (term frequency-inverse document frequency), che produce l'effetto di dare importanza ai termini che compaiono in un documento ma raramente compaiono nell'insieme dei

documenti. La matrice che ne risulta è la seguente:

$$\mathbf{tfidf} = \begin{pmatrix} 2.5455 & 1.1541 & 0.6439 & 0 & 1.0694 \\ 0 & 0 & 1.4739 & 2.2109 & 1.4739 \\ 0.3219 & 0.8322 & 0 & 0.6439 & 1.6643 \\ 0 & 0 & 0 & 0 & 10.0352 \\ 0 & 0 & 0 & 0 & 9.2877 \\ 2.6439 & 0 & 0 & 0 & 3.9658 \\ 1.3219 & 1.3219 & 0 & 0 & 0 \\ 2.2109 & 0 & 0 & 2.2109 & 2.2109 \\ 7.7133 & 0 & 0 & 0 & 0 \\ 17.2433 & 0 & 0 & 0 & 0 \\ 0 & 2.4481 & 4.1159 & 3.3790 & 0 \end{pmatrix}$$

A questo punto entra in gioco la *singular value decomposition* della matrice, che mi permette di individuare le variabili latenti, cioè i *concetti* (matematicamente parlando, i concetti sono direzioni di massima varianza) predominanti della collezione di documenti. Ecco come appare la matrice dei valori singolari, prima di applicare la riduzione dimensionale:

$$\mathbf{S} = \begin{pmatrix} 19.4764 & 0 & 0 & 0 & 0 \\ 0 & 14.5621 & 0 & 0 & 0 \\ 0 & 0 & 6.5501 & 0 & 0 \\ 0 & 0 & 0 & 2.3248 & 0 \\ 0 & 0 & 0 & 0 & 1.3823 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Evidentemente la scelta ottimale da compiere sarebbe quella della riduzione a 2 dimensioni (corrispondenti ai due valori singolari più grandi) o massimo 3. Proviamo comunque a tenere anche la quarta dimensione; vedremo poi l'effetto prodotto.

Calcoliamo ora la matrice M, cioè la matrice di proiezione dei documenti sullo spazio con base i vettori singolari; essa è ottenuta moltiplicando l'inversa di S per la trasposta di U; quest'ultima è la matrice dei vettori singolari delle parole, cioè è una matrice di pseudo-parole, contiene perciò sulle colonne le coordinate delle parole sulle 4 dimensioni considerate

$$\mathbf{M} = \begin{matrix} -0.0070 & -0.0006 & -0.0014 & -0.0030 & -0.0028 & -0.0081 \\ -0.0038 & -0.0075 & -0.0078 & -0.0469 & -0.0435 & -0.0171 \\ 0.0178 & 0.0544 & 0.0151 & -0.0118 & -0.0109 & -0.0056 \\ 0.1842 & -0.1973 & 0.0382 & 0.0311 & 0.0288 & 0.0111 \end{matrix}$$

$$\begin{matrix} -0.0035 & -0.0066 & -0.0202 & -0.0452 & -0.0003 \\ 0.0006 & -0.0096 & 0.0041 & 0.0093 & -0.0013 \\ 0.0112 & 0.0304 & -0.0029 & -0.0064 & 0.1357 \\ 0.1755 & -0.2608 & -0.0036 & -0.0081 & 0.0995 \end{matrix}$$

Ogni riga può essere pensata come un concetto, al quale ognuna delle 11 parole contribuisce più o meno: ciò significa che abbiamo 4 direzioni principali, che sono le nuove variabili dello spazio dimensionalmente ridotto, ognuna delle quali è combinazione lineare delle variabili di partenza, cioè le parole. Esse contribuiscono perciò in misura variabile alla "costruzione" dei concetti principali. Nell'immagine (2.6), per esempio, le variabili di maggior peso sono quelle corrispondenti alle parole *psichedelico*, *band* e in misura minore *musica*. A livello interpretativo, questa può essere pensata come la direzione relativa al concetto *musica rock*. Naturalmente questa è solo un'etichetta che non ha nessuna valenza quantitativa, ma dà un'idea di quello che viene visualizzato. Stesso discorso vale per le immagini (2.7) e (2.8): la prima evidenzia le parole *arte*, *visconte*, *scrivere* e in musura minore anche *autore* e *letteratura*, possiamo perciò interpretarlo come la direzione relativa a Italo Calvino; la seconda evidenzia invece le parole *cinema*, *caffè* e *autore*, possiamo dire che è perciò il concetto relativo a Jim Jarmusch. Infine la figura 2.9 è di difficile interpretazione.

Figura 2.6: Primo concetto: le parole in evidenza sono psichedelico, band e, in misura minore, anche musica, scrivere e autore. Possiamo definire il concetto come musica.

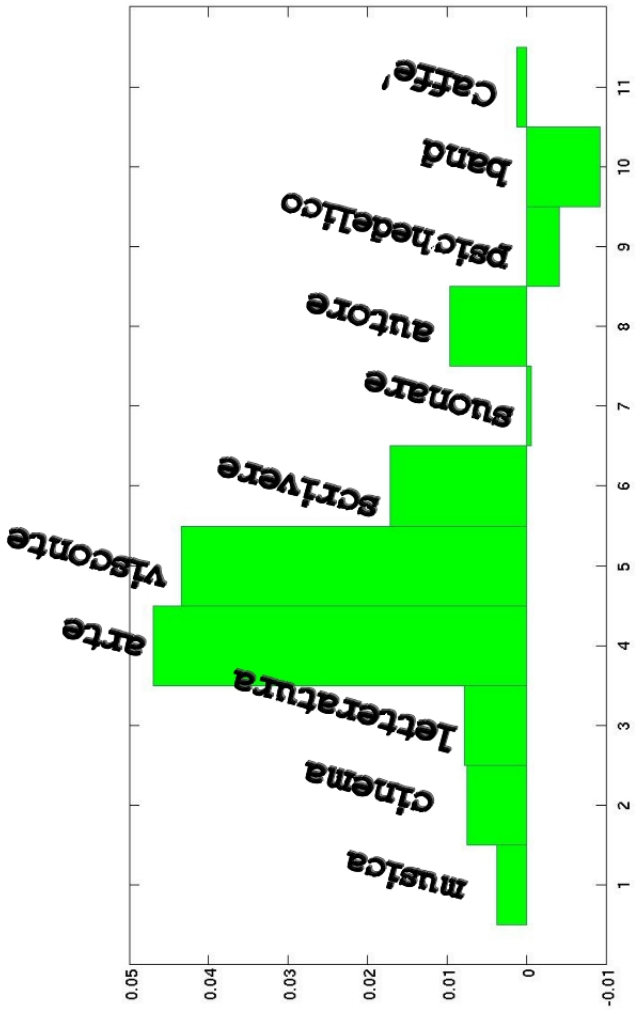

Figura 2.7: Secondo concetto: le parole in evidenza sono arte, visconte e scrivere. Possiamo definire il concetto come Calvino.

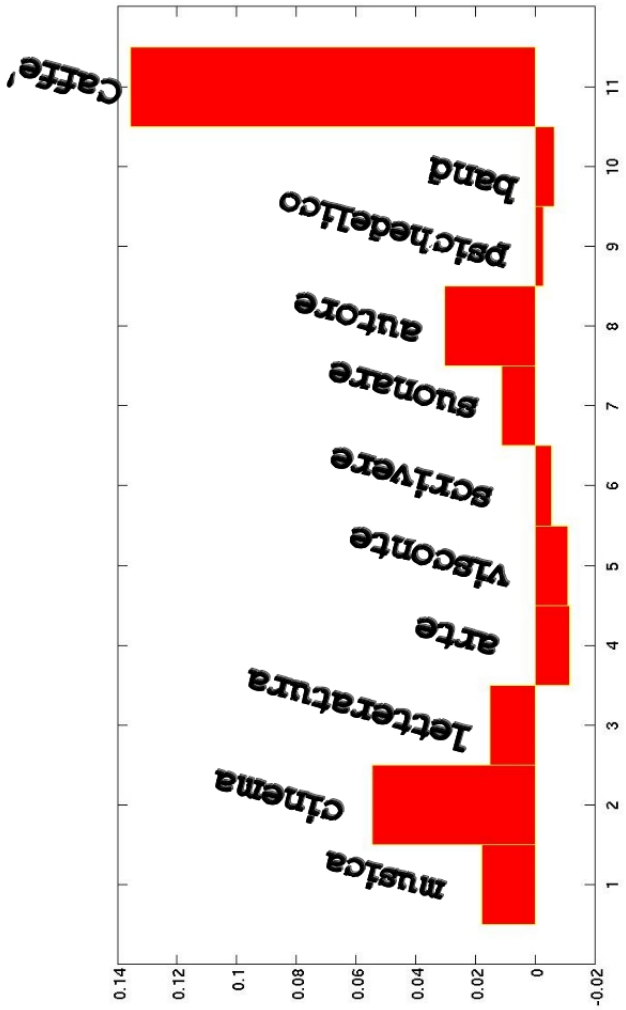

Figura 2.8: Terzo concetto: le parole in evidenza sono cinema, autore e caffè. Possiamo definire il concetto come Jim Jarmusch.

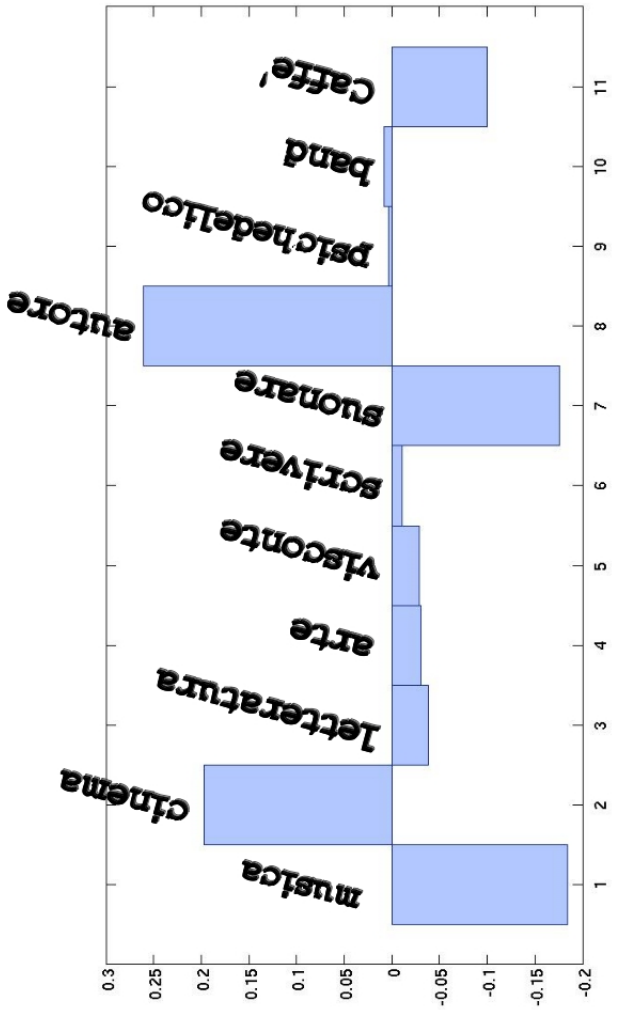

Figura 2.9: Il concetto numero 4.

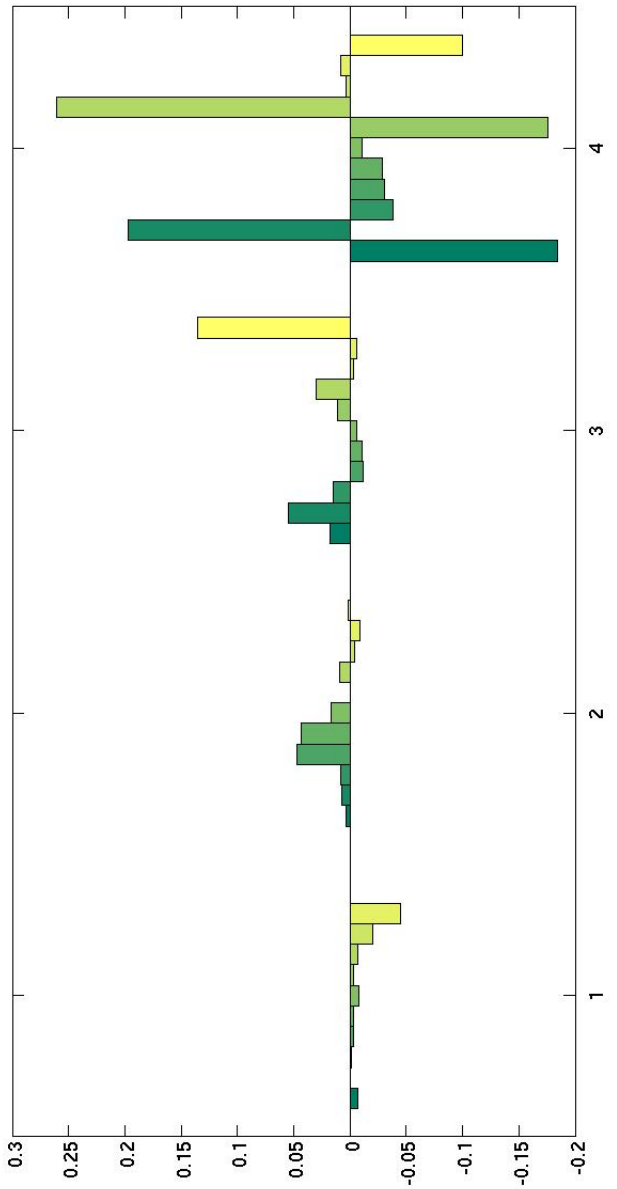

Figura 2.10: Una panoramica sui 4 concetti estratti.

2.2.1 Ricerca dei documenti rilevanti

Infine proviamo ad effettuare una ricerca all'interno di questo piccolo corpus di 5 testi, e verifichiamo se effettivamente i testi risultano riordinati dal più al meno rilevante rispetto alla parola (o anche più) inserita come termine di ricerca.
Inserisco, come prima ricerca, le parole **letteratura** e **visconte**, corrispondenti alle posizioni 3 e 5. Al fine del processo di ricerca, consideriamo un "finto documento" che contiene un'occorrenza di ogni termine inserito; una volta proiettato sul nuovo spazio semantico creato in precedenza (a dimensione ridotta) esso può quindi essere confrontato attraverso la somiglianza cosinusoidale con gli altri testi e vedere a quale "assomiglia" maggiormente. Il vettore relativo alla ricerca **letteratura** e **visconte** è

$$q = (0, 0, (1+\log_2(1))*\log_2(c/df(3)), 0, 0, (1+\log_2(1))*\log_2(c/df(5)), 0, 0, 0, 0, 0)$$

Dove $df(3)$ e $df(5)$ sono il numero di documenti in cui le parole compaiono. Si osservi che solamente il terzo e quinto posto del vettore sono diversi da zero, in quanto sono le entrate relative alle due parole della ricerca.
Una volta "tradotto" tale vettore nel nuovo spazio vettoriale, non resta che confrontarlo con i 5 documenti del corpus e verificare il suo grado di somiglianza attraverso il coseno:

$$sim(q, d_i) = \cos(q, d_i) = \frac{\overrightarrow{q}\,\overrightarrow{d_i}}{|q||d_i|}$$

Di seguito vediamo i documenti riordinati per somiglianza, con a fianco il coefficiente di somiglianza relativo:

Doc. Num.	Documento	Coefficiente
5	Italo Calvino	0.0452
2	Tom Waits	0.0308
1	Rock	0.0144
3	Coffee and Cigarettes	0.0066
4	Jim Jarmusch	-0.0300

2.3 Analisi della Divina Commedia con Infomap NLP

Infomap NLP è un software libero realizzato dallo Stanford Natural Language Processing Group [1] e dal gruppo di ricerca sulla Latent Semantic Analysis dell'Università del Colorado [2]. Infomap, implementato interamente nel linguaggio C (scaricabile liberamente da [3]), è un software che opera la Latent Semantic Analysis su un testo o un corpus di testi. Esso produce un modello di "spazio semantico" attraverso la riduzione dimensionale presentata nei capitoli precedenti, una volta ottenuto il quale è possibile effettuare alcuni tipi di ricerche: è possibile ad esempio ottenere, inserendo una o più parole, le parole o i documenti più pertinenti alla richiesta all'interno del corpus, sia inserendo il tag di uno o più documenti (naturalmente tra quelli presenti nel corpus) ottenere i documenti e le parole più pertinenti alla richiesta. Per la documentazione relativa all'installazione e all'utilizzo, nonchè alla descrizione dell'algoritmo di Infomap, la pagina del software è facilmente consultabile [4].

[1] http://nlp.stanford.edu/index.shtml
[2] http://lsa.colorado.edu
[3] http://sourceforge.net/projects/infomap-nlp/files/
[4] http://infomap-nlp.sourceforge.net/doc/

Per testare il software, si è considerato come corpus quello costituito dai100 canti della Divina Commedia, suddivisi come noto in 34 canti dell'inferno, 33 del purgatorio e 33 del paradiso.
Per ognuno di essi è stata poi effettuata un'associazione, tramite il software, dei 19 canti "semanticamente" più inerenti (20 se si considera il canto stesso, che viene automaticamente indicato come il più somigliante a sè stesso). Ad esempio inserendo come richiesta il tag del canto sesto dell'inferno, si ottengono come primi 20 risultati i canti considerati più "vicini semanticamente", cioè i canti dell'inferno 13, 3, 23, 16, 32, 29, 8, 24, 14, 9, 1, 33, 19, 7 (quindi 14 dei 19 canti associati sono relativi all'inferno) e i canti del purgatorio 14, 22, 20, 13, 21. Analizzando ciò per ognuno dei 100 canti, si ottiene una tabella a 2000 entrate. Essa è naturalmente di difficile interpretazione, o meglio, analizzare l'effettiva efficacia del software richiede un'analisi piuttosto attenta. Per efficacia del software si intende la capacità di collegare effettivamente i canti più semanticamente simili tra loro. Nel canto sesto dell'inferno ad esempio, possiamo affermare che il software ha agito discretamente bene associandogli nei primi 19 risultati 15 canti dell'inferno. Naturalmente il ragionamento si basa sull'ipotesi a priori che i canti di inferno, paradiso e purgatorio si distinguano semanticamente tra loro, se non altro per vocabolario utilizzato, personaggi ricorrenti (Virgilio all'inferno e nel purgatorio, Beatrice in paradiso) e così via. Si riporta di seguito la tabella ottenuta: con **pa**, **pu**, **i**, si indicano rispettivamente i canti di paradiso, purgatorio e inferno seguiti dal numero del canto.

Come già sottolineato, l'analisi di una simile quantità di dati non è semplice. Per rendere più semplice l'analisi dell'efficacia di Infomap, ci serviamo

di un altro software, Gephi [5], concepito per la visualizzazione dei *network*, o *reti di correlazioni*. Può non risultare chiaro il significato di *reti di correlazioni*, in realtà è più semplice e vicino all'esperienza comune di quanto non si possa pensare. Si prenda ad esempio un social network qualsiasi, quale può essere facebook. Esso prevede che ogni persona abbia una cerchia di amici, ognuno dei quali è a sua volta nelle conoscenze di una o più persone, e così via. Facciamo un esempio semplice, e circoscritto, che includa 4 persone A, B, C, D. Visualizziamo le cerchie di amici attraverso una semplice tabella, in cui ogni riga corrisponde ad una persona e le sue amicizie:

A	B		
B	A	C	
C	B	D	A
D	C		

Quando una persona è nella cerchia di amicizie di un'altra, è direttamente collegata ad essa; ad esempio A è direttamente collegato a B (e viceversa). Un collegamento più debole tra due persone può tuttavia sussistere per "*interposta conoscenza*" (un amico in comune) come avviene ad esempio per A e C: A non ha tra gli amici D, ma entrambi conoscono B, cioè compaiono nella stessa riga almeno una volta. Un legame ancora più debole esiste tra A e D: A conosce solo B, il quale conosce C che a sua volta conosce D. Come risulta chiaro, si può estrapolare da un tabulato simile un network di intercorrelazioni tra gli elementi.

Gephi è un software nato proprio per l'analisi di network che attraverso l'utilizzo di algoritmi di *clustering* (che avvicinano gli elementi più correlati e allontanano quelli che non lo sono), visualizza e rende più facilmente interpretabili i gruppi che si formano sulla base di questi dati.

[5] http://gephi.org/

L'idea di base è utilizzare l'analisi fatta sulla Divina Commedia e visualizzarla, rendendo immediata l'analisi dell'efficienza di Infomap ad una prima occhiata.

Il risultato è visualizzato in figura 2.11.

Come si può notare, il risultato ottenuto è piuttosto soddisfacente: l'algoritmo di clustering utilizzato da Gephi evidenzia infatti l'interdipendenza tra 4 nuclei principali: in azzurro un nucleo composto in maniera preponderante dai canti del paradiso, in rosso uno composto per la maggior parte da canti dell'inferno, infine due cluster in verde e viola composti in maggioranza da canti del purgatorio.
Tale visualizzazione permette inoltre di osservare, tramite i "fili" che collegano ogni canto agli altri, come non necessariamente i canti del paradiso siano correlati ad altri canti del paradiso, o quelli dell'inferno ad altri dell'inferno, ma smplicemente che questo avviene statisticamente più spesso.
Come conclusione finale si può osservare che i canti più evidenti, con un font maggiore, sono i più "inerconnessi" agli altri tra tutti quelli del cluster. I motivi per cui certi canti sono maggiormente connessi agli altri non è dato sapere, anche perchè la "semantica latente" utilizzata per la comparazione dei documenti è, come dice il nome, una proprietà non facilmente individuabile semplicemente confrontando gli argomenti di questo o quell'altro canto, o le parole più ricorrenti. Ci si accontenta perciò di aver verificato che l'analisi evidenzia delle effettive differenze tra i canti delle tre cantiche.
E' fondamentale tenere ben presente che la Latent Semantic Analysis è un metodo basato cioè su metodi statistici, come visto in precedenza; ciò significa che quando richiediamo i testi più rilevanti associati nel nostro caso ad un canto della Divina Commedia non è detto che tali risultati siano necessariamente attinenti; quello che maggiormente influisce sulla bontà del risultato

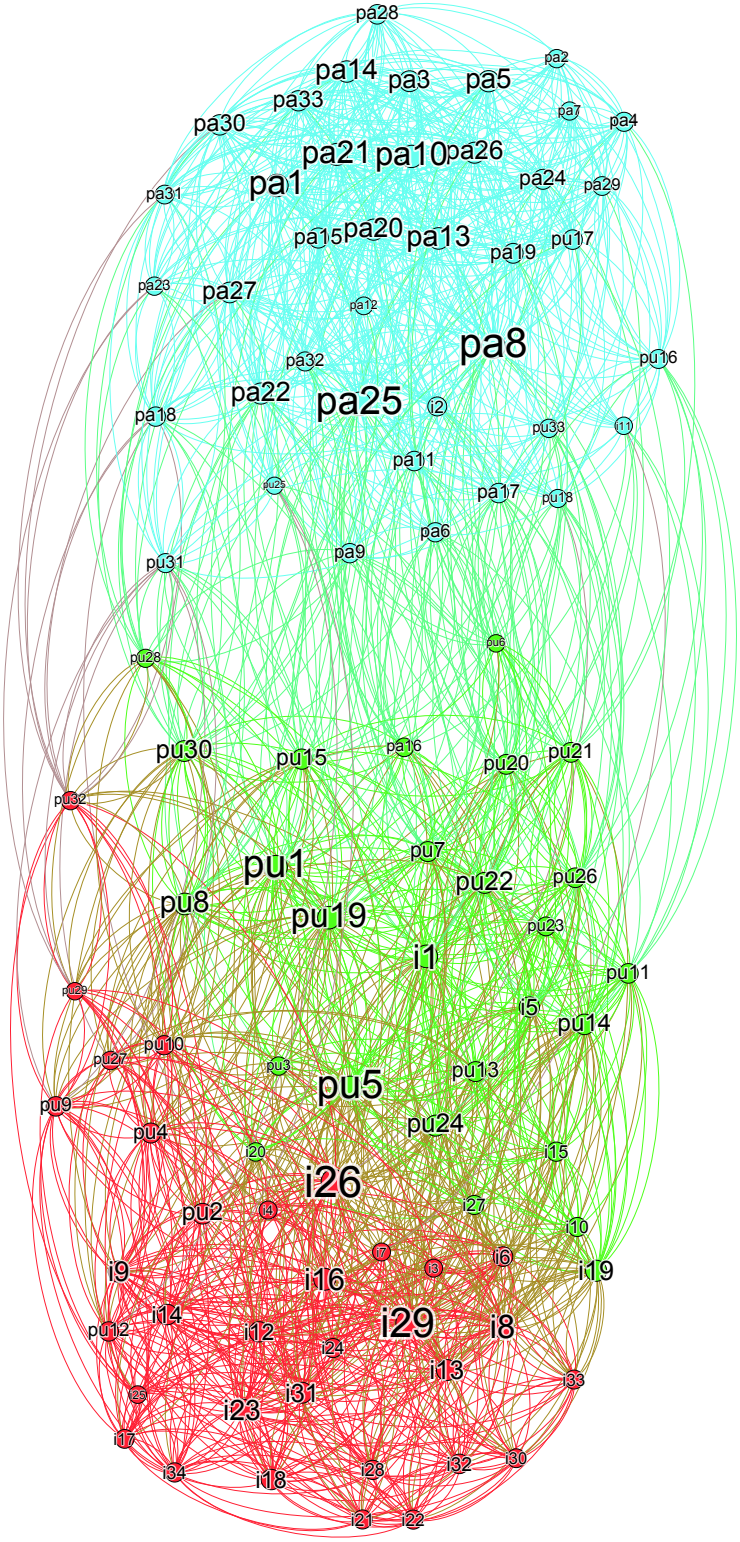

Figura 2.11: Visualizzazione della Divina Commedia: in rosso il cluster formato in maggioranza da canti dell'inferno; in azzurro il cluster del paradiso; in verde e viola due cluster in maggioranza composti da canti del purgatorio.

ottenuto è che *la maggior parte di essi* sia coerente con quanto ci si aspetta. Nelle figure 2.12 e 2.13 sono visualizzati infatti i risultati ottenuti richiedendo i canti associati ai canti 3 e 26 dell'inferno: nel primo caso i canti associati sono in maggioranza dell'inferno, e solo 6 su 19 sono del purgatorio e del paradiso. Nel secondo caso invece il risultato è meno soddisfacente poichè ben 12 risultati su 19 sono canti del purgatorio o del paradiso. Ciononostante, come dimostrato dal grafico totale delle correlazioni di tutti e cento i canti, statisticamente si riscontrano risultati analoghi al primo; in caso contrario non sarebbe così evidente la separazione nei tre grandi "cluster" visualizzati in figura 2.11.

Nel caso precedente si è verificata l'efficacia di Infomap nell'associazione tra documenti. Cercheremo ora di sperimentare l'associazione di documenti ad una o più parole presenti nel corpus, con l'obiettivo di riscontrare lo stesso comportamento di "aggregazione" tra documenti e parole appartenenti allo stesso ambito. Per fare ciò si sono scelte casualmente singole parole, o terzetti di parole, presenti nella Divina Commedia, e si è tenuto poi conto dei primi 30 canti associati ad ognuna di esse. Il risultato è visualizzato nella figura 2.14.

Anche in questo caso il procedimento è statistico, non è perciò detto che tutti i canti associati alla parola "inferno" siano effettivamente canti dell'inferno, come si può notare in figura 2.15, dove 9 risultati su 30 riguardano canti del purgatorio o del paradiso; come prima ciò che conta è che la maggior parte di essi lo sia.

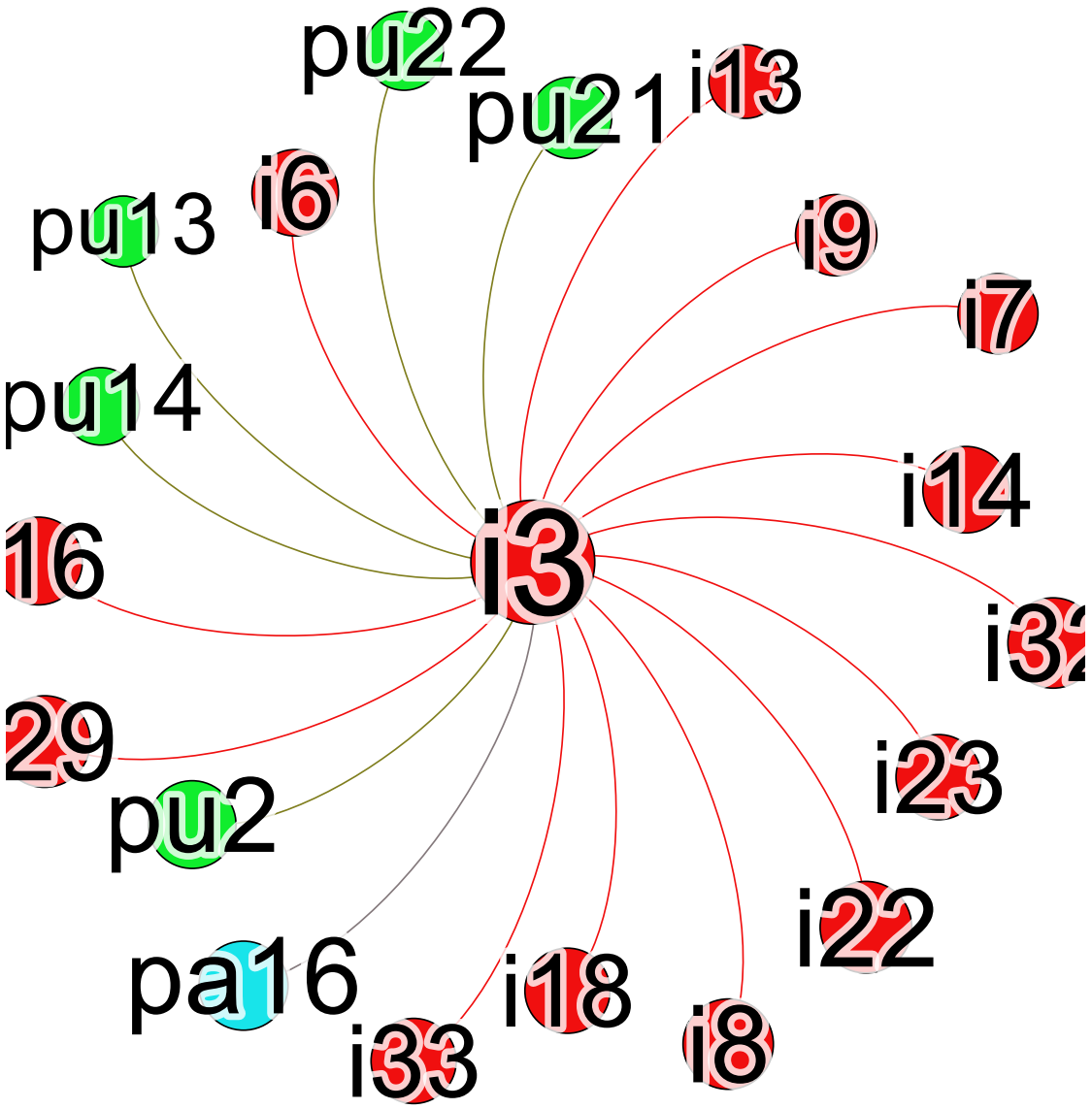

Figura 2.12: Visualizzazione dei canti correlati al canto terzo dell'inferno.

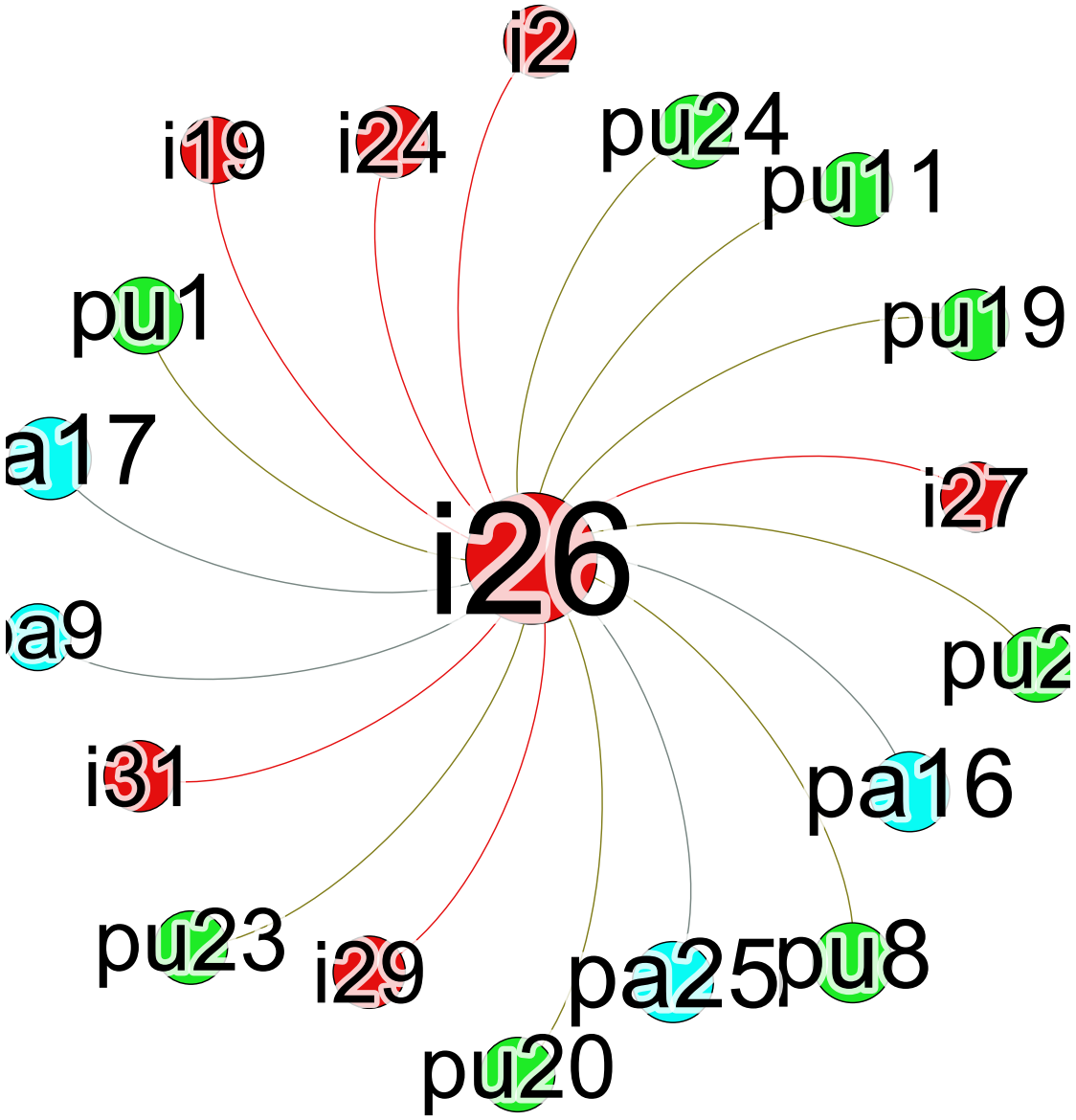

Figura 2.13: Visualizzazione dei canti correlati al canto ventiseiesimo dell'inferno.

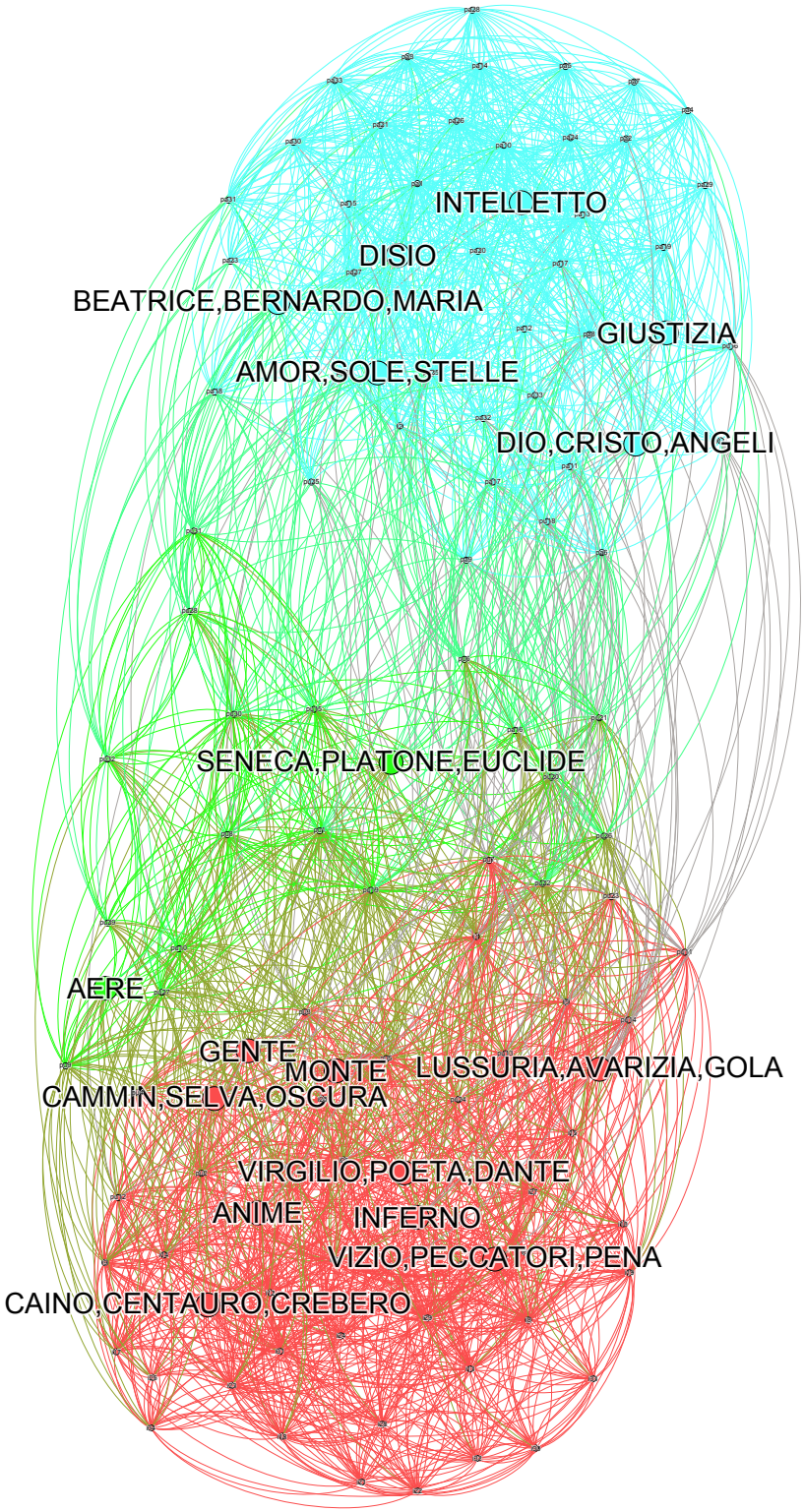

Figura 2.14: Visualizzazione dei canti correlati a singole parole o a terzine di parole.

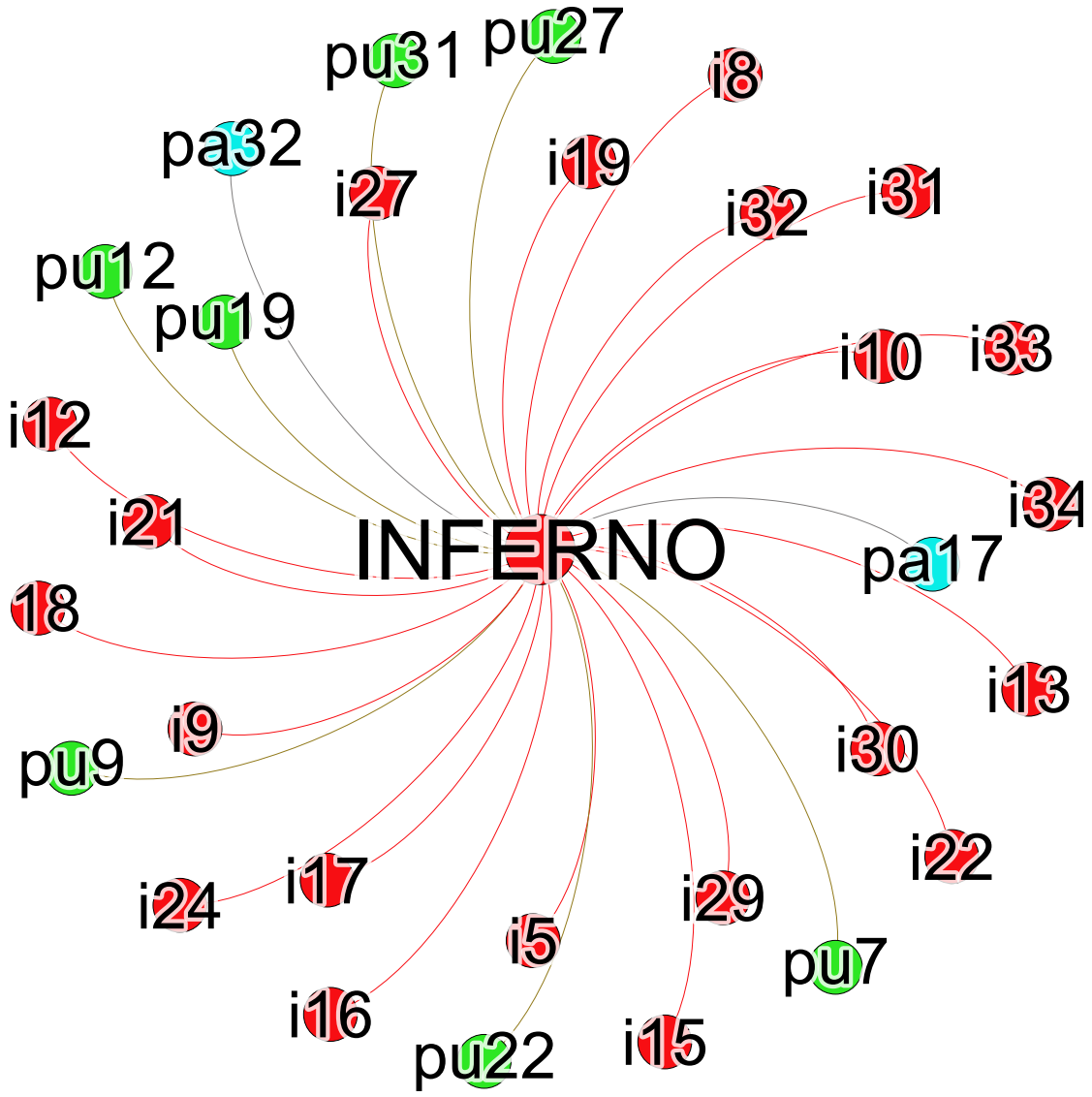

Figura 2.15: Visualizzazione dei canti correlati alla parola ”inferno”.

Appendice A

Codice C per l'analisi dell'entropia

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <time.h>
#include "algoritmi.h"


/**PULISCE TESTO DA SIMBOLI GRAFICI,NUMERI, MAIUSCOLE**/

//input indirizzo del vettore testo, e numero dei caratteri del testo;
//restituisce il vettore testo pulito
int clean(char *vett)
    {
      int i,j,x,flag,tot=0;
```

```
      FILE *doc;
      char nome[15];

      printf("\nNome del file da analizzare (.txt): ");
      scanf("%s",&nome);
      printf("\n");

      doc=fopen(nome,"r");

      while (feof(doc)==0)     { vett[tot]=getc(doc);  tot++; }

      fclose(doc);

// conversione: simboli, accentate, numeri --> spazi,
// maiuscole --> minuscole
      for(i=0; i<tot; i++)
      {
//simboli
if ( vett[i]==';'
  || vett[i]==','
  || vett[i]==':'
  || vett[i]=='.'
  || vett[i]==')'
  || vett[i]=='('
  || vett[i]=='?'
  || vett[i]=='!'
  || vett[i]=='"'
```

```
  || vett[i]==' '
  || vett[i]=='\n'
  //apostrofo
  || vett[i]==39
  //numeri
  || vett[i]=='0'
  || vett[i]=='1'
  || vett[i]=='2'
  || vett[i]=='3'
  || vett[i]=='4'
  || vett[i]=='5'
  || vett[i]=='6'
  || vett[i]=='7'
  || vett[i]=='8'
  || vett[i]=='9' )

vett[i]='-';



//maiuscole
for(j=0;j<26;j++)
  if(vett[i]==65+j)
    vett[i]=97+j;
      }

/**RIDUCO GLI SPAZI CHE SOSTITUISCONO
```

```
CARATTERI E NUMERI A UNO SOLO**/
    for (i=0;i<tot;i++)
    {
      flag=0;
      for(x=0;x<10;x++)
if (vett[i+x]=='-')
  flag++;
else
  break;
      if (flag>0)
for (x=i;x<tot;x++)
  vett[x+1]=vett[x+flag];
    }
    return(tot);
   }



/**CONTATORE DI FREQUENZA**/
//RICEVE L'INDIRIZZO DI UNA STRUCT, UN INIZIO E UNA FINE
//RIEMPIE LA STRUCT DELLE PAROLE DI UN TESTO DALLA LETTERA init
//ALLA LETTERA fin CON LE FREQUENZE DELLE PAROLE
//RESTITUISCE IL NUMERO DI PAROLE LETTE TOTALI
//E ALP NUMERO DI PAROLE DISTINTE

int wcounter(freq *counter, char *vett, int start, int end, int *alp)
{
```

```
     int i,j=0,x,z,m,w=0,k=0;

    // CICLO DI INIZIALIZZAZIONE STRUCT
    for (x=0; x<MAXWORD; x++)
    {
      counter[x].num=0;
      counter[x].delta_entr=0;
      for (z=0; z<MAXLENGHT; z++)
counter[x].w[z]='%';
    }

/**COPIO IL VETTORE DI CARATTERI NELLA STRUCT DELLE FREQUENZE**/

    for (i=start;i<end;i++)
    // se la parola non è finita copio i caratteri nel contatore
      if (vett[i]!='-')
{
  counter[j].w[k]=vett[i];
  k++;
}

      else
{
/**RICERCA DI PAROLE PRECEDENTI UGUALI --> CALCOLO FREQUENZA TOTALE **/
  if( j>0 )  //non lo faccio per la prima parola
    for (z=0;z<j;z++)  // per tutte le parole prima della j-esima
    // se la parola j-esimma c'è già
```

```
      if ( strcmp(counter[z].w,counter[j].w)==0 )
{
  for (m=0;m<MAXLENGHT;m++)
  // cancello la parola j-esima perchè c'è già
    counter[j].w[m]='%';
  // aumento la frequenza della z-esima
  counter[z].num ++;
  // torno indietro di una posizione
  j--;
}
/**FINE CONTROLLO**/
  w++; //aumenta sempre perchè conta le parole totali, anche ripetute
  j++;  //aumenta ma può anche diminuire perchè conta le parole distinte
  k=0;
}

//cancello l'ultima parola (viene contato anche lo spazio)
    for (z=0; z<MAXLENGHT; z++)
       counter[j].w[z] = '%';

//AUMENTO LA FREQUENZA DI TUTTE LE PAROLE TROVATE DI UNO
//PERCHÈ LA PRIMA VOLTA CHE VENGONO TROVATE LA LORO FREQUENZA
//NON AUMENTA
    for (z=0;z<j;z++)
      counter[z].num ++;
//NUMERO DI PAROLE DISTINTE
    *alp=j;
```

```
//RESTITUISCO IL NUMERO DI PAROLE DELLA SEZIONE
    return(w-1);  // tolgo uno perchè viene contato anche lo spazio

}

/**COEFFICIENTE BINOMIALE**/
double binomiale(int n,int k)
{
  float bin[300][300];
  int i,j;

  if (k>n)
    printf("Err: valori non corretti in binomiale");
  else
    {
      bin[0][0]=1;
      for (j=1;j<=n;j++)
bin[0][j]=0;

      for (i=1;i<=n;i++)
{
  bin[i][0]=1;
  for (j=1;j<=n;j++)
     bin[i][j]=bin[i-1][j]+bin[i-1][j-1];
}
    }
```

```
    return(bin[n][k]);
}

/**CONTA NEL VETTORE vett IL NUMERO DI CARATTERI DI n PAROLE
A PARTIRE DA start**/
int car(char *vett, int start, int n)
{
  int t=0,flag=0;
  flag=0;
  while (flag<n)
    {
      if ( vett[t+start] == '-' )  flag++;
      t++;
    }
  return(t);
}
/**CALCOLA LA VARIAZIONE DELL'ENTROPIA DI OGNI PAROLA RISPETTO A QUELLA MEDIA
E STILA LA CLASSIFICA**/
void compute_p_given(void)
{
freq counter[MAXWORD],sez_counter[MAXWORD],tmp;
time_t ti,tf,temptot;
char vett[SIZE];
int  words, n, p, tot;
int i,j,x,k,sez,start=0,end,sup,alp,sez_alp;
float entr_w[MAXWORD],prob_w[MAXWORD],tmp_entr,n1,n2;
float prova;
```

```
srand(time(NULL));

(void)time(&ti);

// Caricamento testo e pulizia
tot=clean(vett);

printf("In quante parti vuoi suddividere il testo? ");
scanf(" %d",&p);
printf("\n");

// Catalogazione parole e frequenze dell'intero testo
words=wcounter(counter,vett,0,tot,&alp);  // words-->parole totali
    // alp-->parole distinte
    // tot-->lettere totali
    // vett-->vettore testo

/**VETTORE FREQUENZE PAROLE NELL'INTERO TESTO**/
for (i=0;i<alp;i++)
  prob_w[i]= (float)counter[i].num/words;
/*****************************************************/

/**CALCOLO LA VARIAZIONE ENTROPIA DELLE SINGOLE PAROLE PER IL P DATO**/
  start=0;
  for (sez=0;sez<p;sez++)
    {
```

```
      end=start+car(vett,start,words/p);
      //frequenze delle parole nella sez
      wcounter(sez_counter,vett,start,end,&sez_alp);
      start=end;
      printf("Sto controllando la sezione: %d su %d \n",sez,p);

      for (j=0;j<alp;j++)
for (i=0;i<sez_alp;i++)  //per ogni parola DISTINTA della sezione
  if ( strcmp( counter[j].w , sez_counter[i].w ) == 0 )
    {
      n2 = counter[j].num;   //occorrenza della parola nel testo
      n1 = sez_counter[i].num; //occorrenza della parola nella sezione
      entr_w[j] = entr_w[j] - (n1/n2)*log(n1/n2) ;
    }
    }

/**VETTORE VARIAZIONE ENTROPIA**/
for (j=0;j<alp;j++)
 counter[j].delta_entr= prob_w[j] *
  ( log(p) - (float)(p-1)/( 2*counter[j].num ) - entr_w[j] );
       //USO APPROSSIMAZIONE DELL'ENTROPIA MEDIA
/*****************************************************************/

/********************** FINE CALCOLO ******************************/

    (void)time(&tf);
    temptot=tf-ti;
```

```
    printf("\nTEMPO IMPIEGATO: %d secondi\n",temptot);


/**ordino le parole in base alla variazione di entropia e scrivo le prime
tot a richiesta**/
    sup=alp;
    while (sup > 1) /* in questo modo si evita 1 passaggio*/
     {
         for (i=0; i<sup-1; i++)
         {
           if (counter[i].delta_entr < counter[i+1].delta_entr)
           {
     tmp_entr = entr_w[i];
             tmp = counter[i];

     counter[i] = counter[i+1];
     entr_w[i] = entr_w[i+1];

             counter[i+1] = tmp;
     entr_w[i+1] = tmp_entr;
           }
         }
     sup--;
     }
/************************************************/
printf("\nParole tot: %d, Parole distinte: %d
lettere: %d\n",words,alp-1,tot);
```

```
printf("\nDigita il numero di parole da inserire nella classifica: ");
scanf("%d",&j);
printf("\n");
  for (i=0;i<j;i++)
    printf("%d)   Parola: %s Freq: %d  Entr: %f Entr. Media: %f
    Variazione: %f \n",i+1,counter[i].w,counter[i].num,entr_w[i],log(p) -
     (float(p-1)/(2*counter[i].num),counter[i].delta_entr);
}
```

Appendice B

Codice Matlab per la ricerca dei documenti rilevanti

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% c = numero dei documenti;
% tf = matrice delle occorenze di dimensione w x c dove w = numero
%      dei termini presi in considerazione;
% df = vettore riga w x 1 che contiene il totale delle occorrenze nei c
%      documenti;
% tfidf = matrice delle occorenze normalizzata con il metodo della
%         term-frequency-inverse document frequency; ha le stesse
%         dimensioni di tf;
% NOTA: le seguenti U S V sono considerate a riduzione dimensionale già
%       avvenuta:
% U matrice dei vettori singolari di sinistra di dimensione wx4;
% S matrice dei valori singolari 4x4;
% V matrice dei valori singolari di destra di dimensione 4xc;
```

```
%
% M matrice di proiezione dei documenti
% q vettore di ricerca
% Cc matrice dei documenti proiettati
% qc documento richiesta proiettato
% sim matrice di somiglianza cosinusoidale
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc;

c=5;
tf=[120 6 2 0 5; 0 0 2 4 2; 1 3 0 2 18; 0 0 0 0 10; 0 0 0 0 8; 2 0 0 0 4;
    1 1 0 0 0; 4 0 0 4 4; 5 0 0 0 0; 86 0 0 0 0; 0 5 24 12 0];
df=[4 3 4 1 1 2 2 3 1 1 3];
tfidf=zeros(size(tf));
for i=1:size(tf,1)
   for j=1:size(tf,2)
      if tf(i,j)>0
          tfidf(i,j)=(1+log2(tf(i,j)))*log2(c/df(i));
      end
   end
end

tfidf
[X Y Z]= svd(tfidf);

Y
```

```
[U S V] = svds(tfidf,4);

S

% matrice di proiezione
M=inv(S)*U'

figure
BAR(M(1,1:11),'hist')
colormap summer
figure
BAR(M(2,1:11),'hist')
colormap summer
figure
BAR(M(3,1:11),'hist')
colormap summer
figure
BAR(M(4,1:11),'hist')
colormap summer
figure
BAR(M)
colormap summer


% inserimento della ricerca: 3: letteratura + 5: visconte: questo può
% essere considerato come un "documento" da confrontare con gli altri
```

```
q=[ 0 0 (1+log2(1))*log2(c/df(3)) 0 0 (1+log2(1))*log2(c/df(5)) 0 0 0 0 0 ];

% proiezione del documento sul nuovo spazio: ottengo il vettore
% richiesta pronto per essere confrontato con gli altri documenti
qc=M*q';

% proietto tutti i documenti sul nuovo spazio a dimensione ridotta: Cc
% contiene sulle colonne i vettori dei documenti proiettati
Cc = inv(S)*U'*tfidf;

% costruisco il vettore di somiglianza cosinusoidale
sim = qc'*Cc;
sim=sim ./ [norm(Cc(:,1)) norm(Cc(:,2))
            norm(Cc(:,3)) norm(Cc(:,4)) norm(Cc(:,5))]
```

Bibliografia

[1] M.A. Montemurro, D. Zanette (2001): *"Entropic Analysis of the Role of Words in Literary Texts"*

[2] M.A. Montemurro, D. Zanette (1999): *"Towards the Quantification of the Semantic Information Encoded in Written Language"*

[3] J.P. Herrera, P.A. Pury (2008): *"Statistical Keyword Detection in Literary Corpora"* European Physic Journal, B 63, 135-146.

[4] E.Alvarez-Lacalle, B.Dorow, J.P. Eckmann, E. Moses (1999): *"Hierarchical Structures Induce Long-Range Dynamical Correlations in Written Texts"* PNAS, May 23, 2006, vol.103 no.21.

[5] T.K. Landauer, P.W. Foltz, D. Laham (1998): *"Introduction to Latent Semantic Analysis"*, Discourse Processes, 25, 259-284.

[6] Jerome L. Bellagarda (2005): *"Latent Semantic Mapping: Dimensionality Reduction Via Globally Optimal Continuous Parameter Modeling"*.

[7] S Deerwester: *"Indexing by Latent Semantic Analysis"*, J. Am. Soc. Inform. Sci., vol 41, 1990.

[8] M.W. Berry: *"Using Linear Algebra for Intelligent Information Retrieval"*, SIAM Review, vol.37, no.4, 1995.

[9] Thomas K. Landauer, Susan T. Dumais: *"A Solution ti Plato's Problem: the Latent Semantic Analysis Theory of Acquisition, Induction and Representation of Knowledge"*, Psychological Review 1997, vol.104, no.2,211-240.

[10] Thomas K. Landauer, Darrell Laham, Marcia Derr: *"From Paragraph to Graph: Latent Semantic Analysis for Information Visualization"*, PNAS, april 6, 2004, vol.101, suppl.1.

[11] T.K. Landauer, *"How Well Can Passage Meaning Be Derived Without Using Word Order: A Comparison of La- tent Semantic Analysis and Humans"*, Proc. Cognit. Science Soc., pp. 412–417, 1997.

[12] J.R. Bellegarda, *"A Novel Word Clustering Algorithm Based on Latent Semantic Analysis"*, ICASSP, At- lanta, GA, pp. I172–I175, May 1996.