Università degli studi di Padova
Dipartimento di Psicologia dello Sviluppo e della Socializzazione
Corso di Laurea Triennale in

Scienze Psicologiche dello sviluppo, della personalità e delle relazioni interpersonali

Relazione Finale

La tecnica della Latent Semantic Analysis e sue applicazioni

Relatore Prof. Antonio Calcagnì
Dipartimento di Scienze Psicologiche dello Sviluppo e della Socializzazione

Laureando Alberto Saccardin
Matricola N 1139301

Anno Accademico 2018/2019

Indice

Introduzione

Il metodo della Latent Semantic Analysis (LSA) è un modello teorico e uno strumento di elaborazione del linguaggio naturale: il processo per trattare in modo automatico, tramite un calcolatore elettronico, informazioni scritte o parlate in una lingua naturale, sviluppate cioè nelle culture umane. Viene usato per estrarre e rappresentare il significato delle parole.
È nato verso la fine degli anni '80 grazie a un team di ricercatori, capitanato da Tom Landauer (Landauer et al. 2013).
Nel seguente documento, diviso in tre parti, verrà data una definizione della tecnica, riassunta la componente matematica sottostante e fornito un esempio di applicazione.

Capitolo 1

Teoria e metodi

La tecnica LSA si basa sull'assunto che esistano centinaia di linguaggi umani differenti, ciascuno con decine di migliaia di parole. La maggior parte degli esseri umani apprende il linguaggio nel corso del proprio sviluppo; quindi ci deve essere un meccanismo condiviso da chiunque e in ogni cultura, che permetta di imparare il linguaggio tramite l'esperienza pratica, affrontando la quotidianità senza specificare esplicitamente definizioni o regole.
I bambini apprendono il linguaggio per esposizione con l'ambiente in cui crescono, iniziando da concetti presenti in ogni ambiente naturale (genitori, oggetti, animali) formando associazioni con le parole corrispondenti. Per un essere umano imparare il linguaggio è dunque relativamente semplice, rispetto alle distinzioni concettuali (oggetti e animali sono concetti diversi) e alle generalizzazioni che i linguaggi naturali richiedono (un cane rimane tale a prescindere dalla razza). Ciò fa presupporre che la mente umana sia predisposta all'acquisizione del linguaggio; per un computer, invece, acquisirlo semplicemente per esposizione con campioni di testo non è possibile, è necessario esplicitarne le regole.
La maggior parte dei dibattiti riguardo questa capacità si sono concentrati sulla dicotomia geni/ambiente. Il problema su cui si concentra la LSA è differente. Ciò su cui si focalizza è la costruzione del significato che avviene nel passaggio tra una parola e l'altra, costruito a partire dalle esperienze fatte precedentemente. Ma come viene stimato il significato?
Viene stimato per mezzo di computazioni statistiche, applicate ad ampi cam-

pioni di testo, che determinano la similarità tra parole e insiemi di parole. La similarità, invece, è indagata attraverso metodi di algebra lineare, in particolare, la singular value decomposition (SVD).
In più di un'occasione la LSA ha dimostrato di riuscire a riflettere la conoscenza umana: se utilizzato per valutare l'adeguatezza del contenuto di un saggio espositivo, condivide tra l'85 e il 90% delle informazioni con lettori umani esperti (Landauer 2002); riesce a simulare l'ordinamento delle parole e giudizi categoriali secondo criteri umani (Schreiner et al. 1997). Inoltre, la LSA ha trovato applicazione in un certo numero di aree. È in grado di selezionare materiale didattico per singoli studenti, fornire feedback ai piloti sulle tecniche di atterraggio, combinare posti di lavoro con candidati (Landauer e Dumais 1997).
Prima dell'avvento della LSA, estrarre il significato dal testo richiedeva un intervento umano notevole; schiere di programmatori impegnati per ore a riscrivere testi. Le analisi dimostrano che c'è un buon compromesso tra ciò che è trasmesso da un testo e ciò che viene estratto automaticamente con gli strumenti algebrici; questo perché la LSA fornisce il significato latente o sottostante. Per farlo postula congetture riguardanti la sua natura: secondo la teoria, il significato di un testo converge dalle parole di cui è composto. In alcuni casi specifici, l'ordine con cui appaiono le parole è importante, ma non quanto le parole stesse. Il significato è acquisito risolvendo un gran numero di equazioni simultaneamente, le quali catturano l'utilizzo contestuale delle parole.
Infine, la LSA esemplifica un nuovo approccio alla scienza cognitiva. Piuttosto che sezionare gli esperimenti di laboratorio, tenta di considerare i dati prodotti da partecipanti reali in veri compiti.

1.1 Cos'è il significato?

Filosofi, linguisti, umanisti, romanzieri e poeti hanno cercato di definire la parola "significato" in molti modi diversi, spaziando dalla verità delle cose a proprietà intrinseche di oggetti e avvenimenti nel mondo, a costrutti mentali riferiti al mondo esterno, fino a parlare di essenze mistiche irriducibili fisicamente. Alcuni affermano che siano concetti astratti o proprietà del

mondo che esistono a priori e indipendentemente da ogni rappresentazione che dipenda dal linguaggio. Ciò implica per definizione, che i computer non possano creare significato partendo dai dati: è qualcosa che esiste prima delle parole.
Il modo in cui impariamo il linguaggio potrebbe partire dalle nostre esperienze verbali, imparando le relazioni esistenti tra parole e locuzioni, e viceversa. Nell'ottica della teoria della LSA, l'informazione che arriva alla mente, in qualsiasi forma percettiva, non ha una priorità, sia che siano manipolazioni di parole o combinazioni astratte di idee (il cervello identifica un oggetto presentato per via visiva, così come il corrispondente in parola); l'esperienza percettiva deve essere mappata su espressioni linguistiche affinché ci sia corrispondenza.
Il significato non risiede solo nelle parole. Bambini e animali non parlano, eppure attribuiscono un senso sulla base di relazioni complesse tra percezioni e azioni.

1.2 L'equivalenza dei linguaggi

La LSA opera riducendo le dimensioni del testo originale a un numero limitato di fattori. Questo processo può essere facilmente applicato a qualsiasi linguaggio che nel complesso sia dotato di significato, siano parole francesi o ideogrammi cinesi. Ciò ha reso possibile costruire in modo automatico motori di ricerca basati su LSA con corrispettivi in Hindi, Arabico, Giapponese e Ebraico.
Tutti i linguaggi provati in questo modo dagli autori possono essere confrontati cosicchè i paragrafi in una lingua abbiano un alto grado di similarità con un'altra quasi fossero stati tradotti da esseri umani. Perché i linguaggi rappresentino sostanzialmente lo stesso insieme di significati devono anche rappresentare le stesse relazioni tra parole.

1.3 Struttura, metodi e limitazioni della LSA

Una mappa geografica è un'astrazione in cui vengono rappresentate alcune caratteristiche di un luogo. Ciò che viene raffigurato sono punti d'interesse e la distanza tra questi punti: è dal loro insieme che viene generata la mappa (considerando la città di New York due punti potrebbero essere Central Park e l'Empire State Building). Quel che più conta è la corretta disposizione di quei punti in una mappa, così che possa essere considerata affidabile. Similmente, in una mappa percettiva del linguaggio le parole sono in relazione tra loro, ed è dall'insieme di queste relazioni astratte che nasce il significato.
La LSA è una teoria del significato, perché offre una spiegazione di alcuni dei fenomeni che normalmente permettono di utilizzare il linguaggio, tramite l'espressione, la comprensione e la comunicazione di idee e conoscenze per mezzo di parole e connessioni tra parole.
Ciononostante, la LSA non può essere definita una teoria completa. Non considera l'ordine delle parole, che rende significative le frasi. Senza l'intervento umano non rappresenta adeguatamente la variabilità che può derivare da metafore, negazioni e proposizioni matematiche (espressioni linguistiche o simboliche di cui si può affermare con certezza la veridicità o meno, spesso definite come enunciati, ad esempio: "la Terra è un pianeta", "4 è un multiplo di 2"). Queste sono problematiche importanti nella comprensione, ma non implica che la teoria sia sbagliata. Tuttavia, stima l'ammontare relativo del contributo delle parole e del loro ordine in una frase o in un paragrafo nell'ordine dell'80%-90% (Landauer 2002).
L'idea degli sviluppatori di questo metodo è che un modello computazionale riuscito, anche se incompleto, fornisca un cardine per lo sviluppo, spiegando il fenomeno del linguaggio e del significato meglio di quanto non farebbero argomentazioni puramente filosofiche. L'interesse è orientato al modo in cui una macchina riesca a rendersi utile manipolando il linguaggio.
È una teoria che non riguarda ogni aspetto del linguaggio, ma solo un suo aspetto essenziale.

1.4 Ordine delle parole e significato

La LSA non funzionerebbe come tecnica se l'ordine delle parole giocasse un ruolo centrale così come nella Linguistica. Ciò è stato testato in più modi diversi: ignorando l'ordine delle parole, mettendo dei confini al contributo apportato dalle combinazioni e dall'ordine delle parole nel trasmettere significato, e considerando come avviene l'ordinamento delle parole in altri linguaggi.
La grammatica inglese, assieme a quella italiana, è poco flessibile. Si consideri il seguente esempio:
"order syntax? much. ignoring word Missed by is how"
(How much is missed by ignoring word order syntax?)
Frasi disordinate come quella in esempio sono relativamente facili da capire per ricostruzione; si può procedere estrapolandone l'essenza, ricomponendone il contenuto in forma grammaticalmente corretta sulla base della frase più probabile. Persino negazioni doppie, mancanti e non posizionate correttamente sono frequentemente ignorate dai lettori (Foltz, Kintsch e Landauer 1998), e la loro mancanza spesso non altera in modo considerevole il significato del testo. Gli autori hanno sperimentato queste argomentazioni nel seguente modo: è stato usato un algoritmo di ordinamento lineare applicato alla LSA per poter analizzare una serie di saggi, cosicché le 300-dimensionality similarities fossero distorte al minimo. Questo ordinamento è stato poi confrontato con un altro basato su punteggi forniti indipendentemente da due umani esperti riferiti agli stessi saggi. Infine, hanno misurato l'ammontare di informazione condivisa nel punteggio dei saggi (a) dai due esperti umani (basato presumibilmente su tutto ciò che hanno potuto estrarre da tutti gli aspetti della scrittura, inclusa la sintassi) e (b) tra il punteggio della LSA e degli umani. La misura usata è stata la cross entropy: questa rileva l'ammontare, in information-theoretic bits, di incertezza (entropia) relativa a una fonte che può essere ridotta, conoscendo le altre. È risultata una correlazione di .90 tra umani, e di .81 con la tecnica. La macchina ha condiviso il 90% delle informazioni con ciascun umano, tanto quanto essi hanno spartito tra loro. Sarebbe un azzardo trarre conclusioni sulla base di un unico esperimento; ciò che si può inferire in questo caso specifico è che il restante 10% è probabilmente

attribuibile all'ordine delle parole.

1.5 La computazione adottata dalla LSA

L'assunto più importante per la LSA è poter esplorare vincoli reciproci. Un vincolo viene rappresentato con una funzione (partendo da una frase), strutturato in modo tale da rappresentare il significato di ogni passaggio tra parole. Ciò prende il nome di "composizionalità", e stabilisce che la rappresentazione dei passaggi sia la somma delle rappresentazioni delle parole, in forma matematica:

$$\text{significato di passaggio} = \Sigma(m_{\text{parola1}}, m_{\text{parola2}}, m_{\text{parola3}}) \tag{1.1}$$

Nel loro esempio gli autori hanno usato un corpus di addestramento, che divide il testo in segmenti che trasmettano (idealmente) significato. Viene costruita una matrice; nelle righe vengono riportate le parole e nelle colonne i passaggi. Le singole celle contengono il numero di volte in cui ciascuna parola appare in un determinato passaggio.

1.6 Polisemia e significato delle parole

Una parola può avere più di un significato (si pensi all'esempio della parola "panda"). La LSA riesce a distinguere la stessa parola a seconda del ruolo che assume nel testo, questo perché il contesto in cui appare determina il suo contributo al significato globale. Di seguito un esempio con la parola "swallow", che può significare "ingoiare" o "ingerire" così come "rondine":
"swallow" – "The process of taking food into the body through the mouth by eating." Cosine = .57
"swallow" – "Small long winged songbird noted for swift graceful flight the regularity of its migrations." Cosine = .30
Il problema emerge quando due aspetti del significato di una parola sono ugualmente probabili.
Altre volte sono i significati delle parole ad influenzarsi reciprocamente: "my

surgeon is a butcher" (il mio chirurgo è un macellaio) e "my butcher is a surgeon" (il mio macellaio è un chirurgo). In quest'esempio l'ordine delle parole influenza il significato della frase.

1.7 L'importanza della riduzione

Si consideri la mappatura di punti geografici: nello specifico, la distanza tra Roma, Londra e New York, e si provi a tracciare una singola linea retta che le unisca. Non ci si riuscirebbe adottando un grafico a due dimensioni, poiché risulterebbero distorte. Usando tre dimensioni si otterrebbe una rappresentazione più fedele; la soluzione migliore, in questo caso, è trovare un numero di dimensioni tale per cui la risoluzione sia adattata allo scopo.
Elaborando un testo con la LSA si presenta un problema simile, poiché per parole come "macchina" e "automobile" la dimensione è la stessa, essendo i concetti correlati, ma la situazione cambia invece con coppie di parole quali "filosofia" e "automobile" o "basket" e "algebra".
Con dimensioni, in linguaggio matematico, si intendono il numero di vettori che compongono la base di uno spazio vettoriale. La riduzione della dimensionalità è una tecnica di mappatura, usata per rappresentare i dati in una dimensione inferiore e più interpretabile. Ad esempio, per visualizzare un diagramma 3D in 2D.
Per poter eseguire la riduzione si utilizza un algoritmo che ha sostanzialmente due obiettivi: eliminare il rumore dei dati (congiunzioni, avverbi, punteggiatura) e combinare i termini che correlano tra loro. Se la dimensione iniziale del dataset è di R^n verrà effettuata una riduzione ad una dimensione inferiore R^k, dove k<n. Ciò presenta vantaggi e svantaggi: se da un lato la riduzione comporta una compressione dei dati, riducendo la difficoltà per l'algoritmo che dovrà trattarli, dall'altro può comportare una degradazione delle informazioni; ciò si ripercuoterà sulle capacità predittive dell'algoritmo. La stessa parola in due testi diversi potrebbe non rimandare allo stesso significato, si considerino i seguenti esempi:

La Fiat Panda è un'auto superutilitaria prodotta dalla casa automobilistica italiana FIAT in tre serie: la prima, nata nel 1980 e disegnata da Giorgetto Giugiaro; la seconda nata nel 2003 e disegnata da Giuliano Biasio per Bertone; e la terza, nata nel 2012 e disegnata dal centro stile Fiat sotto la direzione di Roberto Giolito. Al termine della seconda generazione la vettura aveva venduto più di 6 milioni e mezzo di esemplari.	Il panda gigante o panda maggiore (Ailuropoda melanoleuca DAVID, 1869) è un mammifero appartenente alla famiglia degli orsi. Originario della Cina centrale, vive nelle regioni montuose del Sichuan. Verso la seconda metà del XX secolo, il panda è diventato un emblema nazionale in Cina, e dal 1982 è raffigurato sulle monete auree cinesi (serie Panda Dorato). Inoltre è diventato il simbolo del WWF.

Tabella 1.1: Comparazione della definizione di "panda"

In entrambi si trova la parola chiave "Panda". Mentre nel primo documento è utilizzato per indicare un'automobile, nel secondo documento, il termine è riferito a un animale. La LSA impara il significato di ogni parola attraverso: l'esperienza ripetuta di incontro con essa, la SVD, la riduzione della dimensionalità e la composizione di tutti i passaggi nella quale non si verifica.

1.8 Esempi di proprietà della LSA

II risultati delle computazioni sono valutati in coseni. Il valore può oscillare tra -1 e +1, ma nella pratica raramente si osserva un valore inferiore a 0 per coppie parola-parola, passaggio-passaggio o parola-passaggio simili. Ecco alcuni esempi tra frasi:

"*Several doctors operated on a patient*"
"*The surgery was done by many physicians*" (cosine = .66)

"*A circle's diameter*":
"*radius of sphere*" (cosine = .55)
"*music of the spheres*" (cosine = .03)

Questi invece sono alcuni esempi tipici parola-parola.

Coppia di parole	*Coseno*
thing-things	.61
man-woman	.37
husband-wife	.87
sugar-sweet	.42
salt-NaCl	.61
mouse-mice	.79
doctor-physician	.61

Quando la LSA computa un vettore avente significato della lunghezza di un'intera frase o paragrafo, l'identità delle parole letterali è persa e non reversibile. È una situazione analoga a quella di una persona che legge un testo; poco tempo dopo aver letto una frase o un paragrafo ciò che rimane in memoria (salvo alcune eccezioni) è un estratto contenente le informazioni del testo originario.

Capitolo 2

Basi matematiche dietro la LSA

La LSA è basata sul concetto di modello di spazio vettoriali (VSM), un approccio che usa l'algebra lineare per il recupero automatico dell'informazione.
Il VSM è stato adottato per gestire il recupero del testo da database contenenti un gran numero di informazioni eterogenee. Il modello matematico sottostante al VSM definisce vettori unici per ogni carattere e documento, formando delle queries: in informatica questo termine è usato per indicare l'interrogazione di un database fatta da un utente, per poter compiere delle operazioni sui dati.
La LSA è considerata un modello di spazio vettoriale troncato, $\mathrm{VSM_k}$, che rappresenta caratteri e documenti scoprendo la struttura semantica "latente". Per poterlo fare, sfrutta il significato dei caratteri rimuovendo il "rumore" presente a causa della variabilità nella scelta dei caratteri. Tale rumore è evidenziato dalla polisemia e dai sinonimi all'interno dei documenti (Deerwester et al. 1990). Da ciò risulta che la similarità tra documenti non dipenda dai caratteri presenti, ma dal contenuto semantico.
È usando la SVD che la LSA ottiene il significato di parole e documenti. La computazione richiede di convertire una matrice in input in vettori. Nel caso di matrici sparse (con molti elementi uguali a 0) i calcoli matematici vengono eseguiti per mezzo dell'"algoritmo di Lanczos con riortogonalizzazione selettiva".

2.1 Creare il Modello di spazio vettoriale

2.1.1 La matrice in input

Per poter creare un VSM deve essere prima creata una matrice caratteri-documenti. I caratteri sono contenuti nelle righe. Solitamente sono termini, ma a seconda dell'applicazione possono essere frasi o concetti. Nelle colonne invece, sono inseriti i documenti, che possono essere porzioni di testo come paragrafi singoli o molteplici, capitoli di libri, libri. La matrice **A** viene composta a partire dall'insieme di m vocaboli e n documenti. Il caso più frequente prevede che il numero di vocaboli sia maggiore rispetto al numero dei documenti ($m >> n$), sebbene ci siano situazioni in cui le proporzioni sono invertite ($n >> m$), come quando vengono selezionati documenti provenienti da Internet (Berry e Browne 2005; Berry, Drmac e Jessup 1999). Inizialmente, ciascuna colonna della matrice **A** contiene elementi con valore uguale o diverso da zero, a_{ij}. Ciascun elemento diverso da zero, a_{ij}, corrisponde alla frequenza $i$esima del vocabolo nel $j$esimo documento. Di seguito un breve esempio:

Label	*Titles*
M1	**Rock** and **Roll Music** in the 1960's
M2	Different **Drum Rolls**, a Demonstration of Techniques
M3	**Drum** and Bass **Composition**
M4	A Perspective of **Rock Music** in the 90's
M5	**Music** and **Composition** of Popular Bands
B1	How to Make **Bread** and **Rolls**, a **Demonstration**
B2	**Ingredients** for Crescent **Rolls**
B3	A **Recipe** for Sour**dough Bread**
B4	A Quick **Recipe** for Pizza **Dough** using Organic **Ingredients**

Tabella 2.1: Titoli per Argomenti di Musica e Cucina

Vocaboli	*Documenti*								
	M1	M2	M3	M4	M5	B1	B2	B3	B4
Bread	0	0	0	0	0	1	0	1	0
Composition	0	0	1	0	1	0	0	0	0
Demonstration	0	1	0	0	0	1	0	0	0
Dough	0	0	0	0	0	0	0	1	1
Drum	0	1	1	0	0	0	0	0	0
Ingredients	0	0	0	0	0	0	1	0	1
Music	1	0	0	1	1	0	0	0	0
Recipe	0	0	0	0	0	0	0	1	1
Rock	1	0	0	1	0	0	0	0	0
Roll	1	1	0	0	0	1	1	0	0

Tabella 2.2: Matrice con frequenza dei vocaboli corrispondenti ai titoli della tabella 2.1

I documenti etichettati da M1 a M5 sono titoli legati alla musica, i documenti da B1 a B4 hanno invece a che fare con la cucina; in nessun documento una parola chiave si ripete più di una volta.
Una funzione di pesatura viene applicata a ogni elemento diverso da zero. Nel recupero, i caratteri che maggiormente distinguono documenti particolari dai restanti sono considerati i più importanti; perciò, la funzione dovrebbe assegnare un basso peso a un termine usato frequentemente che si ripete in più documenti e un peso elevato ai caratteri che ricorrono in alcuni documenti ma non in tutti (Salton 1991). La LSA applica una funzione di pesatura globale e locale a ogni elemento diverso da zero, a_{ij}, per far sì che l'importanza del termine aumenti o diminuisca, nel documento (localmente) e attraverso l'intera collezione di documenti (globalmente). Il peso così assegnato è direttamente proporzionale a quanto spesso ci si imbatte nel termine all'interno del documento, e inversamente proporzionale a quante volte è ripetuto nell'insieme dei documenti.
Quindi, a_{ij} = local(i,j) * global(i), dove local(i,j) equivale alla pesatura locale del carattere i nel documento j, mentre global(i) è la pesatura globale (Dumais 1991; Letsche e Berry 1997). Le funzioni di pesatura locale utilizzano la frequenza dei caratteri, la frequenza binaria (0 se la parola chiave non

è presente, altrimenti 1), e il logaritmo della frequenza più 1. Le funzioni di pesatura globale invece includono normal, tf-idf (term frequency-inverse document frequency) e l'entropia; fondamentalmente tutte assegnano un basso peso ai termini che ricorrono spesso o in più documenti. Una delle funzioni di pesatura locale e globale più utilizzata è la log-entropy. Secondo Dumais questa funzione ha ottenuto i migliori risultati in recupero (Dumais 1991). La funzione di pesatura locale del logaritmo (frequenza del carattere +1) riduce l'effetto di grandi differenze tra frequenze. L'entropia, definita come:

$$1 + \sum_{j} \frac{p_{\mathrm{ij}} \log_2(p_{\mathrm{ij}})}{\log_2 n} \tag{2.1}$$

In cui $p_{\mathrm{ij}} = \frac{tf_{\mathrm{ij}}}{gf_{\mathrm{i}}}$: $\mathrm{tf_{ij}}$ equivale alla frequenza del carattere i nel documento j, mentre $\mathrm{gf_i}$ al totale del numero di volte in cui il carattere i appare nell'insieme di n documenti. Nella tabella che segue sono state applicate le funzioni di pesatura globale e locale log-entropy alla tabella precedente: Tipicamente la

	M1	M2	M3	M4	M5	B1	B2	B3	B4
bread	0	0	0	0	0	.653	0	.653	0
composition	0	0	.653	0	.653	0	0	0	0
demonstration	0	.653	0	0	0	.653	0	0	0
dough	0	0	0	0	0	0	0	.653	.653
drum	0	.653	.653	0	0	0	0	0	0
ingredients	0	0	0	0	0	0	.653	0	.653
music	.477	0	0	.477	.477	0	0	0	0
recipe	0	0	0	0	0	0	0	.653	.653
rock	.653	0	0	.653	0	0	0	0	0
roll	.352	.352	0	0	0	.352	.352	0	0

Tabella 2.3: Matrice dei titoli con applicata la pesatura, corrispondente ai titoli nella tabella 2.1

matrice in input **A** è considerata sparsa perché contiene molti più valori zero che non-zero. Di solito solo circa l'1% delle caselle della matrice contiene valori diversi da zero (Berry e Browne 2005). Negli esempi questo valore è di circa il 25%.

2.1.2 Decomposizione della matrice in input nei componenti ortogonali

Dopo la creazione della matrice **A**, occorre trasformarla in uno spazio ortogonale composto da caratteri e documenti mediante le decomposizioni ortogonali, per poter utilizzare i troncamenti dei vettori. La trasformazione permette di preservare alcune delle proprietà della matrice, tra cui la lunghezza e le distanze riferite ai vettori nelle righe e colonne che formano la matrice $m \times n$.
Cos'è una matrice ortogonale? È una matrice invertibile la cui trasposta coincide con la sua inversa; in forma matematica $\mathbf{Q}\mathbf{Q}^{\mathrm{T}}=\mathbf{I}$.
Una matrice quadrata (con lo stesso numero di righe e colonne) è detta invertibile se esiste un'altra matrice tale che il prodotto matriciale tra le due restituisca la matrice identità.
La matrice trasposta è la matrice ottenuta scambiandone le righe con le colonne.
La matrice inversa è quella matrice che, moltiplicata per la matrice di partenza, restituisce una matrice identità.
La matrice identità, anche detta matrice unità, è una matrice quadrata in cui tutti gli elementi della diagonale principale sono costituiti dal numero 1, mentre i restanti elementi sono 0.
Un esempio di matrice ortogonale:

$$\mathbf{Q} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \mathbf{Q}^{\mathrm{T}} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \mathbf{Q}\mathbf{Q}^{\mathrm{T}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Ci sono una varietà di metodi per decomporre la matrice nei suoi componenti ortogonali, quello usato più spesso dalla LSA è la SVD, questo per varie ragioni. Primo, decompone la matrice **A** in fattori ortogonali che rappresentano sia i caratteri che i documenti. Secondo, cattura sufficientemente la struttura semantica latente e permette di adeguare le rappresentazioni nello spazio vettoriale permettendo di scegliere il numero di dimensioni. Infine, permette di trattare dataset di dimensioni estese.
La SVD applicata a una matrice **A** con m righe $\times$ n colonne con rango (massimo numero di righe o colonne linearmente indipendenti in **A**) di $\mathbf{A}=r$, è

definita come:

$$A = U\Sigma V^{\mathrm{T}} \tag{2.2}$$

In cui **U** è una matrice ortogonale ($UU^{T}=I_{m}$), **V** è una matrice ortogonale ($VV^{T}=I_{n}$), e **Σ** è una matrice diagonale, le cui restanti celle diagonali equivalgono a zero (Bunch, Demmel e Van Loan 1989).
L'indipendenza lineare di un insieme di vettori appartenenti ad uno spazio vettoriale si verifica se nessuno di questi può essere espresso come una combinazione lineare degli altri.

2.1.3 Troncamento dei componenti ortogonali

In figura si può vedere una rappresentazione della SVD applicata alla matrice **A** (Berry, Dumais e O'Brien 1995; Witter e Berry 1998).

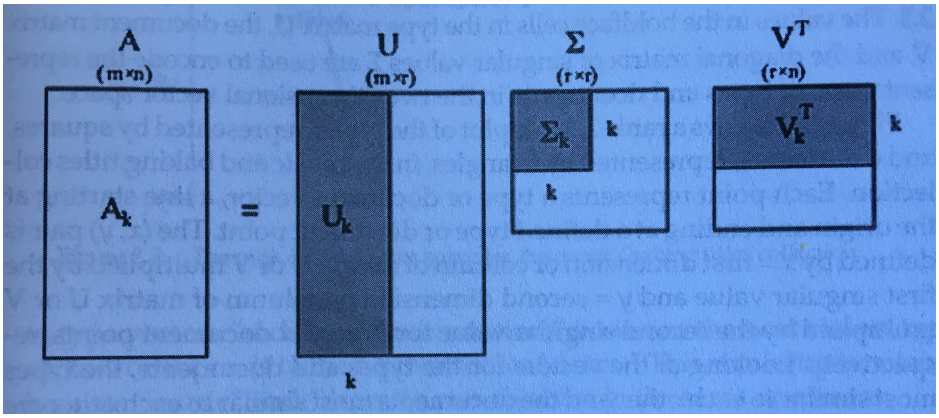

Figura 2.1: Troncamento della SVD, da *The Handbook of the LSA*

Visto che **A** può essere scritta come la somma delle matrici di rango 1: $A = \sum_{i=1}^{r} u_{\mathrm{i}}\sigma_{\mathrm{i}}v_{\mathrm{i}}^{\mathrm{T}}$, r può essere ridotta a k per creare $A_{\mathrm{k}} = \sum_{i=1}^{k} u_{\mathrm{i}}\sigma_{\mathrm{i}}v_{\mathrm{i}}^{\mathrm{T}}$. La matrice $\mathbf{A}_{\mathrm{k}}$ è l'approssimazione più vicina (poiché la distanza è minimizzata) alla matrice ortogonale **A** (Berry e Browne 2005; Berry, Dumais e O'Brien 1995; Jiang et al. 1999). La matrice $\mathbf{A}_{\mathrm{k}}$ è creata ignorando o settando a zero tutti i k elementi, escluso il primo, dei vettori contenenti i caratteri in **U**, dei vettori contenenti i documenti in **V**, e i primi valori singolari di k in **Σ**.

Riducendo la dimensione da r a k, vengono rimosse le informazioni superflue che potrebbero generare variabilità nel trattamento dei vocaboli, considerate "rumore". È tramite il processo di troncamento della SVD e creando $\mathbf{A}_k$ che viene catturata la struttura semantica sottostante. Nello spazio vettoriale della dimensione k le parole con un significato simile sono "vicine" le une alle altre, anche se non presenti nello stesso documento, così come lo sono documenti concettualmente simili, ma senza vocaboli in comune (Berry, Dumais e O'Brien 1995). Questo spazio vettoriale è il fondamento che permette alla LSA di valorizzare la struttura semantica.
Usando la precedente collezione di documenti (tabella 2.1), rielaborata attraverso le successive matrici, la SVD può essere computata e troncata in uno spazio vettoriale a 2 dimensioni riducendo il rango a k=2. Nella figura 2.2, i

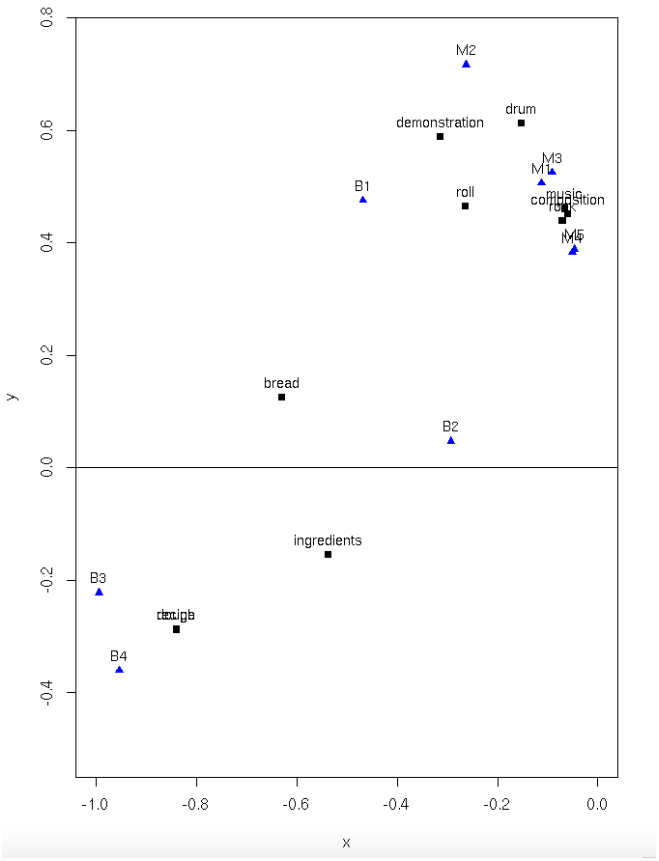

Figura 2.2: Vicinanza di termini e documenti in uno spazio vettoriale a 2 dimensioni

caratteri sono rappresentati dai quadrati, mentre i documenti dai triangoli.
La similarità tra vettori di parole e documenti è determinata attraverso gli angoli tra vettori. Se due vettori sono simili, l'angolo che li separerà sarà piccolo. Nell'esempio, i documenti M4, "A Perspective of Rock Music in the 90's", e M1 "Rock and Roll Music in the 1960's" sono i documenti più vicini a M3, "Drum and Bass Composition", sebbene non condividano termini con quest'ultimo.
Come scegliere il rango o il numero di dimensioni da utilizzare? La scelta si fonda su prove empiriche: per dataset estesi le dimensioni possono variare da 100 a 300 (Jessup e Martin 2001; Jiang et al. 1999; Lizza e Sartoretto 2001).

Matrice U con Vettori Vocaboli									
Bread	-.42	.09	.20	.33	.48	-.33	-.46	-.21	-.28
Composition	-.04	.34	-.09	-.67	.28	-.43	-.02	-.06	.40
Demonstration	-.21	.44	.42	.29	-.09	-.02	.60	-.29	.21
Dough	-.55	-.22	-.10	-.11	.12	.23	.15	.15	.11
Drum	-.10	.46	.29	-.41	-.11	.55	-.26	-.02	-.37
Ingredients	-.35	-.12	-.13	-.17	-.72	-.35	-.10	-.37	-.17
Music	-.04	.35	-.54	.03	.12	-.16	.41	.18	-.58
Recipe	-.55	-.22	-.10	-.11	.12	.23	.15	.15	.11
Rock	-.05	.33	-.60	.29	-.02	.33	-.28	-.35	.37
Roll	-.17	.35	.05	.24	-.33	-.19	-.25	.73	.22
Matrice Σ con Valori Singolari									
	1.52	0	0	0	0	0	0	0	0
	0	1.32	0	0	0	0	0	0	0
	0	0	1.18	0	0	0	0	0	0
	0	0	0	1.05	0	0	0	0	0
	0	0	0	0	.91	0	0	0	0
	0	0	0	0	0	.65	0	0	0
	0	0	0	0	0	0	.38	0	0
	0	0	0	0	0	0	0	.23	0
	0	0	0	0	0	0	0	0	.10
	0	0	0	0	0	0	0	0	0
Matrice V con Vettori Documenti									
M1	-.07	.38	-.53	.27	-.08	.12	-.20	.50	.42
M2	-.17	.54	.41	.00	-.28	.43	.34	.22	-.28
M3	-.06	.40	.11	-.67	.12	.12	-.49	-.23	.23
M4	-.03	.29	-.55	.19	.05	.22	.04	-.62	-.37
M5	-.03	.29	-.27	-.40	.27	-.55	.48	.21	-.17
B1	-.31	.36	.36	.46	.15	-.45	.00	-.32	.31
B2	-.19	.04	-.06	-.02	-.65	-.45	-.41	.07	-.40
B3	-.66	-.17	.00	.06	.51	.12	-.27	.25	-.35
B4	-.63	-.27	-.18	-.24	-.35	.10	.35	-.20	.37

Tabella 2.4: La SVD della matrice con pesatura applicata in tabella 2.3

2.2 Computare il modello di spazio vettoriale

2.2.1 Autovettore e Autovalore

Il termine autovettore è stato tradotto dalla parola tedesca Eigenvektor, coniata da David Hilbert nel 1904. Eigen significa "proprio", "caratteristico". Il piano cartesiano e lo spazio euclideo sono esempi particolari di spazi vettoriali: ogni punto dello spazio può essere descritto tramite un vettore, rappresentato graficamente da un segmento che collega l'origine al punto. In uno spazio vettoriale è possibile effettuare trasformazioni lineari sui vettori che lo costituiscono: esempi di trasformazioni lineari sono le rotazioni, le omotetie (un vettore viene amplificato o contratto) e le riflessioni (un vettore viene trasformato nel suo speculare rispetto a un punto, retta o piano assegnati). Un autovettore per la trasformazione lineare L è un vettore diverso da 0 che a seguito dell'applicazione di L non cambia la sua direzione, limitandosi ad essere moltiplicato per uno scalare λ, il rispettivo autovalore (riferibile solo alle matrici quadrate). Il vettore può quindi soltanto cambiare modulo (venendo amplificato o contratto) e verso (venendo ribaltato).
Nella figura 2.3 l'immagine a sinistra viene modificata mentre l'asse verticale rimane fisso. Il vettore blu ha cambiato lievemente direzione diversamente da quello rosso, quindi il vettore rosso è un autovettore della trasformazione. Inoltre, poiché il vettore rosso non è stato né allungato, né compresso, né ribaltato, il suo autovalore è 1. Tutti i vettori sull'asse verticale sono multipli scalari del vettore rosso, e sono tutti autovettori.

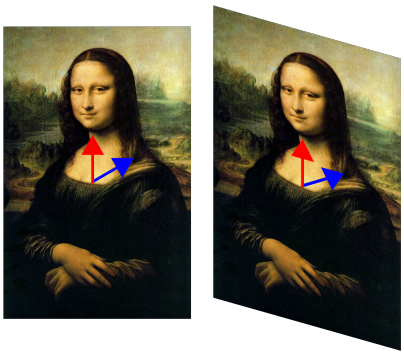

Figura 2.3: Esempio applicazione di un autovettore

2.2.2 Risolvere un Eigenproblem

Calcolare il numero di dimensioni per poter ridurre lo spazio vettoriale della matrice in input non è facile. Data una matrice $\mathbf{A}$ dove $m \geq n$ il problema principale sarà trovare i più grandi autovalori e autovettori k possibili della matrice $\mathbf{B}=\mathbf{A}^{\mathrm{T}}\mathbf{A}$. Trovare gli autovettori di $\mathbf{B}$ produrrà i vettori dei documenti (le colonne della matrice ortogonale $\mathbf{V}$ nella SVD equivalgono agli autovettori di $\mathbf{B}$), mentre trovare gli autovalori di $\mathbf{B}$ produrrà i valori singolari (le radici quadrate non negative di $\mathbf{B}$). I vettori associati ai vocaboli vengono invece calcolati come $U_{\mathrm{k}} = AV_{\mathrm{k}}\Sigma_{\mathrm{k}}^{-1}$. Nel caso in cui ci siano più documenti che parole, $n>m$, la computazione della SVD dovrà indagare il maggior numero di autovalori e autovettori di $\mathbf{B}=\mathbf{A}\mathbf{A}^{\mathrm{T}}$. Inversamente alla situazione precedente, la ricerca degli autovettori di $\mathbf{B}$ produrrà i vettori dei caratteri (le colonne della matrice ortogonale $\mathbf{U}$ nella SVD), trovare gli autovettori di $\mathbf{B}$ produrrà di nuovo i valori singolari, mentre per trovare i vettori dei documenti si applicherà la formula $V_{\mathrm{k}} = A^{\mathrm{T}}U_{\mathrm{k}}\Sigma_{\mathrm{k}}^{-1}$. Per ricapitolare, data la matrice simmetrica $\mathbf{B}$, creata a partire dalla matrice sparsa in input $\mathbf{A}$, l'obiettivo è trovare il valore più ampio assegnabile agli autovalori e autovettori di $\mathbf{B}$. Quindi, la computazione eseguita dalla SVD si basa sul risolvere un eigenproblem sparso e simmetrico (Berry 1992; Bunch, Demmel e Van Loan 1989).

2.3 Modificare il modello di spazio vettoriale

2.3.1 Querying

To query, in inglese, significa domandare, interrogare. Una volta prodotto lo spazio vettoriale con rango ridotto, trovare parole e documenti vicini a una data query diventa semplice. All'interno dello spazio vettoriale k ci si riferisce a una query come a uno *pseudo-documento* (Deerwester et al. 1990). Una query equivale alla somma pesata dei vettori dei caratteri ridimensionata dall'inverso dei valori singolari, in questo modo ogni dimensione dello spazio vettoriale k viene pesata individualmente:

$$query = q^{\mathrm{T}}U_{\mathrm{k}}\Sigma_{\mathrm{k}}^{-1} \tag{2.3}$$

In cui $\mathbf{q}^T$ è il vettore contenente celle vuote e frequenze dei caratteri con applicata pesatura corrispondenti ai caratteri specificati nella query.
Una volta che lo pseudo-documento è formato, viene applicata una misura di similarità per determinare quali documenti e termini sono più vicini all'interno della query. La misura usata più di frequente è la similarità del coseno, che restituisce un indice che permette di classificare documenti e vocaboli associati in ordine decrescente, così da poter mostrare per primi i valori più elevati. Formata questa lista, solo i documenti sopra una certa soglia sono considerati rilevanti (Letsche e Berry 1997).
Riferendosi al precedente esempio, la piccola collezione di titoli inerenti musica/cucina, si può computare una query "Recipe for White Bread" come pseudo-documento e proiettarlo nello spazio con rango 2 della Figura 2.2. Il calcolo del coseno per il vettore della query è mostrato nella tabella sottostante, in cui sono inseriti solo i documenti più rilevanti, con valore superiore a .80.

Document	*Cosine*
B3: A Recipe for Sourdough Bread	.99047
B4: A Quick Recipe for Pizza Dough using Organic Ingredients	.92550
B2: Ingredients for Crescent Rolls	.90822

2.3.2 Aggiungere e rimuovere dati

La capacità di aggiungere dati a uno spazio vettoriale o di rimuoverli è importante perché le informazioni presenti nella collezione potrebbero aver bisogno di essere modificate. Esistono tre metodi a proposito: "folding-in" o "folding-out" (per l'aggiunta di dati o per la rimozione), la ricomputazione della SVD, l'aggiornamento della SVD o il metodo "downdating del modello ridotto", descritti da (Berry, Dumais e O'Brien 1995). Comunque, ad oggi, non esiste un metodo ottimale per la modifica dei dati che sia più accurato della ricomputazione dello spazio vettoriale già esistente.

Capitolo 3

Un esempio di applicazione

Per poter analizzare il testo con la Latent Semantic Analysis è necessario processare i documenti, poi creare lo spazio semantico.
Nel seguente capitolo verrà utilizzato R per poter eseguire i calcoli e creare le rappresentazioni grafiche. La LSA è stata applicata a un campione di tre documenti contenenti frasi estratte dalle pubblicità italiane relative a:

- Lottomatica, lotterie (*lotto.txt*);
- Scommesse sportive (*sport.txt*);
- Scommesse inerenti il poker (*poker.txt*).

I pacchetti utilizzati sono stati:

```
library(quanteda);library(readtext);library(dplyr);
library(wordcloud); library(tm); library(ggplot2)
```

3.1 Preparare i documenti

L'insieme dei documenti collezionati formerà ciò che viene chiamato corpus testuale; quanto più sarà ampio il corpus, tanto più saranno accurate le analisi. In questo contesto, un documento è una porzione di testo riferita a un singolo argomento.
Il testo viene prima importato in formato txt:

```
poker <- readtext("/Directory/poker.txt")
sport <- readtext("/Directory/sport.txt")
lotto <- readtext("/Directory/lotto.txt")
```

In seguito viene importata la lista delle stopwords (parole che saranno poi eliminate dal corpus):

```
stopwords <- readLines("/Directory/stopwords.txt", encoding = "UTF-8")
```

Successivamente si procede con la creazione del corpus.
Un oggetto con classe "corpus" contiene il testo originale, variabili e metadati (informazioni riguardo alla data di creazione e alla posizione) riferiti al documento, e informazioni per una successiva elaborazione del corpus; il corpus viene creato mediante:

```
poker_c <- corpus(poker)
lotto_c <- corpus(lotto)
sport_c <- corpus(sport)
azzardo <- poker_c + lotto_c + sport_c
```

3.2 Creare lo spazio semantico

Creato il corpus, i documenti hanno bisogno di essere pre-processati per essere funzionali; i termini vengono normalizzati. La normalizzazione è un processo che permette di trasformare le parole riducendo il rischio che parole simili e equivalenti per significato (come “è” e “essere”) vengano riconosciute come termini differenti.
Le trasformazioni più comuni sono:

- Stemming: Un’operazione dipendente dal linguaggio, consiste nel creare un’unica rappresentazione di una parola indipendentemente dalla forma che può assumere (passato, presente, futuro, singolare, plurale, etc.), troncando la parola;

- Rimuovere le stop words: Le parole che non trasmettono significato vengono rimosse, come congiunzioni e avverbi;

- Trasformazioni ortografiche: Necessarie per uguagliare le parole, vengono rimossi accenti, le forme abbreviate e ridotte le lettere maiuscole;
- Rimuovere la punteggiatura: La punteggiatura viene rimossa e sostituita da spazi bianchi;
- Identificare termini non comuni: Alcuni termini, composti da più parole ma che costituiscono un unico concetto, vengono combinati, solitamente per mezzo di "underscore", ad esempio: "New York" con "New_York";
- Lemmatizzazione: Similmente allo stemming crea un'unica rappresentazione di più parole, ma lo fa ricorrendo a un dizionario che recupera le informazioni morfologiche.

Per poter normalizzare e trasformare le parole bisogna dapprima "tokenizzare" il testo, ossia dividerlo nelle singole parole di cui è composto; ecco un esempio di tokenizzazione:

```
poker_t <- poker_c %>%
tokens(remove_numbers=TRUE, remove_punct=TRUE, remove_symbols=TRUE) %>%
tokens_tolower() %>%
tokens_select(stopwords, selection = "remove", padding = FALSE, verbose =
    TRUE)
azzardo_t <- azzardo %>%
tokens(remove_numbers=TRUE, remove_punct=TRUE, remove_symbols=TRUE) %>%
tokens_tolower() %>%
tokens_select(stopwords, selection = "remove", padding = FALSE, verbose =
    TRUE)
```

Nell'esempio sono stati rimossi i numeri, la punteggiatura, i simboli, riportate tutte le lettere in minuscolo e rimosse le stopwords; la funzione padding serve per lasciare uno spazio bianco dove è stato rimosso il token, verbose invece per notificare il numero di caratteri eliminati.
Per quanto le trasformazioni fatte dagli algoritmi siano utili ci sono dei casi in cui però possono fallire: lo stemming, ad esempio, potrebbe considerare i termini "universo", "universale" e "università" come lo stesso termine. A tutti i documenti devono essere applicate le stesse trasformazioni, per evitare risultati privi di significato.

Una volta applicate le trasformazioni si può passare alla creazione della TDM, una matrice termini × documenti (*term-document matrix*). Per farlo bisognerà applicare le funzioni di pesatura locale e globale.
La TDM nell'esempio è stata creata utilizzando una matrice DFM, partendo dai dati tokenizzati:

```
poker_d <- dfm(poker_t, stem = FALSE)
sport_d <- dfm(sport_t, stem = FALSE)
lotto_d <- dfm(lotto_t, stem = FALSE)
azzardo_d <- dfm(azzardo_t, stem = FALSE)
```

In seguito la matrice è stata trasposta per facilitarne la lettura:

```
azzardo_m <- t(azzardo_d)
```

Termini	**poker**	**lotto**	**sport**
mancano	0	0	1
mandi	0	1	0
mangiato	1	0	0
mano	0	2	4
mantieni_la_calma	0	0	1
mappe	0	0	1
mar	0	1	0
mare	1	0	0
maria	0	1	0
martedì	1	1	0

Tabella 3.1: Matrice estratta da azzardo_m

Creata anche la TDM si può passare infine alla creazione dello spazio semantico cui applicare la SVD.
Dapprima si costruisce una variabile cui vengono applicate le funzioni di pesatura locale e globale:

```
tfidf_azzardo <- dfm_tfidf(azzardo_d,scheme_tf = "prop", scheme_df = "
    inverse")
```

```
tfidf_azzardo <- t(tfidf_azzardo)
tfidf_azzardo <- tfidf_azzardo[order(rownames(tfidf_azzardo)),]
```

Lw_tf (Local Weighting Term Frequency) indica che non vengono applicati calcoli alle variabili. Gw_idf (Global Weighting Inverse Document Frequency) calcola la frequenza inversa del documento in una matrice $n \times m$. Ogni cella corrisponde a 1 + il logaritmo del numero di documenti diviso per il numero di documenti in cui il termine appare.
Si crea poi lo spazio vettoriale per la LSA partendo dalla matrice con applicata pesatura:

```
myLSAspace <- lsa(tfidf_azzardo, dims=dimcalc_share())
```

La variabile ora costruita contiene tre liste, una con i valori dei termini (matrice $\mathbf{U}$), una i valori dei documenti (matrice $\mathbf{V}$) e una i valori singolari ($\mathbf{\Sigma}$).

Matrice U con Vettori Vocaboli			
accendilo	-.01520	.00026	-.00036
acceso	-.00018	-.02712	-.00234
accetta	-.00018	-.02712	-.00234
accettazione	-.00033	-.00262	.03789
accontenta	-.01520	.00026	-.00036
accontenti	-.00018	-.02712	-.00234
accorgi	-.00016	-.00131	.01895
adesso	0	0	0
adoro	-.00016	-.00131	.018949
adrenalina	-.00018	-.02712	-.00234
Matrice Σ con Valori Singolari			
	.03061	.02015	.01805
Matrice V con Vettori Documenti			
Poker	-.99984	.01143	-.01389
Lotto	-.01472	-.07697	.99692
Sport	-.01032	-.99697	-.07713

Tabella 3.2: La SVD a 3 dimensioni

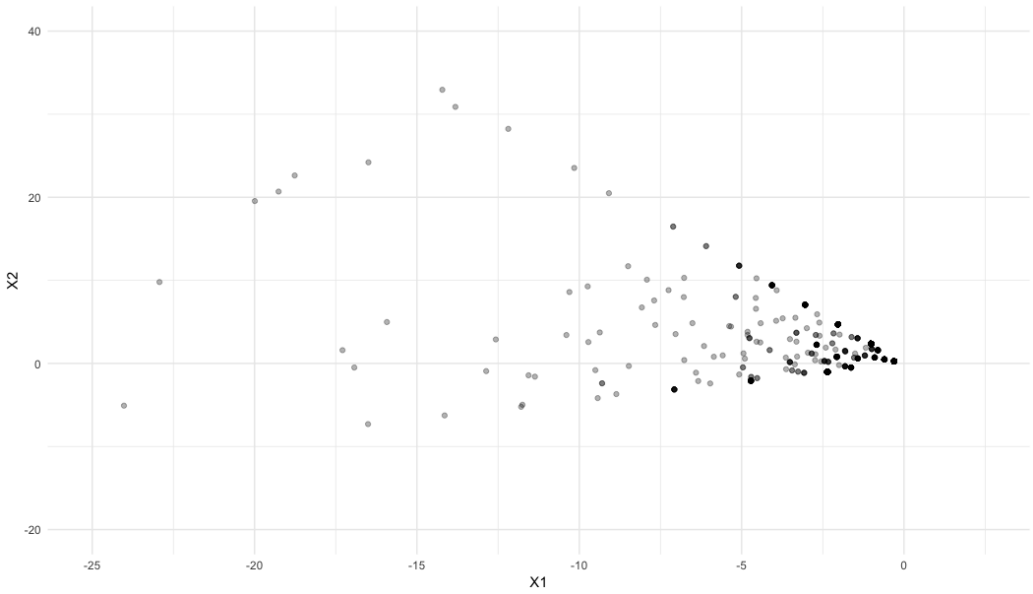

Figura 3.1: Grafico della dispersione dei termini all'interno dei tre documenti

Costruito lo spazio latente si possono applicare le query, in questo esempio verranno utilizzate due frasi:

- *Giocare a poker ti renderà ricco*;
- *Se vuoi vivere appieno gioca.*

```
q <- fold_in(query("Giocare a poker ti rendera ricco", rownames(
    myNewMasuccex)),myLSAspace)
qd <- 0
for (i in 1:ncol(myNewMatrix)) {
  qd[i] <- cosine(as.vector(q),as.vector(myNewMatrix[,i]))
}
```

Come ci si potrebbe aspettare la correlazione è alta con il documento *poker* (.999), ma scarsa con i due restanti (.280 con *lotto* e .017 con *sport*). La seconda query invece

```
q <- fold_in(query("Se vuoi vivere appieno gioca", rownames(myNewMatrix)),
    myLSAspace)
qd <- 0
```

```
for (i in 1:ncol(myNewMatrix)) {
  qd[i] <- cosine(as.vector(q),as.vector(myNewMatrix[,i]))
}
```

mostra un andamento opposto: correlazione molto alta con *lotto* (.987) e *sport* (.993) ma bassa con *poker* (.117).

3.3 Indici e breve rappresentazione dei dati

L'analisi non permette di trarre conclusioni di nessun tipo. Viene comunque applicato l'indice di diversità lessicale e rappresentati i documenti tramite WordCloud dopo aver modificato gli oggetti sulla base del vocabolario in comune (i termini sono ora 75 in totale).
Preso ciascun documento singolarmente le cinque parole più utilizzate sono state:

- Lotto: "Euro" (8,7%), "Vincere" (6,8%), "Puoi" (6,8%), "Giocare" (6%), "Gioco" (5,6%);
- Poker: "Gioco" (18,9%), "Euro" (8%), "Bonus" (5,9%), "Online" (4,4%), "Giochi" (3,9%);
- Sport: "Giocatori" (12,4%), "Bonus" (7,7%), "Euro" (6,2%), "Gioca" (5,2%), "Giocare" (4,3%).

L'indice di diversità lessicale (permette di indagare la complessità dei documenti e quanto si differenziano lessicalmente) viene applicato alle matrici DFM, ed è pressapoco simile per tutti e tre i documenti, leggermente più alto per "*sport*"

```
sapply(c(poker_dF,sport_dF,lotto_dF), textstat_lexdiv)
```

- Lotto: .181
- Poker: .194
- Sport: .357

Questo perchè l'indice si basa sul rapporto tra i caratteri nel vocabolario e i tokens; *sport* conta 210 frequenze a differenza di *lotto* (414) e *poker* (387).

Lotto è il documento maggiormente omogeneo tra i tre. Il decremento è costante sin dal primo valore in termini di proporzione. In figura si può vedere come i primi quattro termini abbiano dimensioni simili, seguiti poi dal quinto e dal sesto elemento. Al venticinquesimo termine si contano 5 frequenze.

Figura 3.2: WordCloud di *lotto*

Poker invece è il documento più sproporzionato: "gioco" il primo termine, ricorre ben 73 volte, in ordine decrescente si situano poi "euro" (31), "bonus" (23) e "online" (17). Già dal quinto termine si assiste a una frequenza di cinque volte inferiore rispetto al primo. Il grafico mostra come "gioco" sia il termine preponderante.

Figura 3.3: WordCloud di *poker*

Sport invece si situa a metà tra i documenti precedenti: rispetto a *lotto* presenta una maggiore disomogeneità (facendo riferimento al grafico si può notare come la parola "giocatori" spicchi), in confronto a *poker* però c'è un minore dislivello tra le proporzioni (la figura 3.4 conta un numero più elevato di parole in azzurro rispetto alla figura 3.3).

Figura 3.4: WordCloud di *sport*

Conclusioni

Lo scopo di questo elaborato è fornire una descrizione della Latent Semantic Analysis ai non esperti. Questa breve sezione riassume, con poche parole, i principali vantaggi e svantaggi nell'utilizzare questa tecnica.
La facilità di utilizzo e di comprensione lo rendono uno strumento accessibile per molte persone, anche perchè permette implementazioni con alcuni linguaggi di programmazione (ad esempio JAVA e Python). Utilizzando R (per cui sono disponibili diversi pacchetti) i calcoli sono semplificati, e la costruzione dello spazio semantico è poco dispendiosa poichè la tecnica si basa sulla decomposizione delle matrici. Il modo in cui i termini vengono trattati permette di risparmiare tempo, inoltre è progettato affinchè errori rappresentazionali dovuti alla polisemia e ai sinonimi siano ridotti al minimo (anche se ciò dipende dal database di partenza). Lo spazio vettoriale viene ridotto, così da permettere il contenimento del massimo delle informazioni con il minimo rumore; in questo modo si possono operare computazioni tramite algoritmi che non potrebbero essere eseguite altrimenti.
Per quanto sia una tecnica utile e flessibile presenta però anche alcuni aspetti negativi. Il fatto che si basi sull'algebra lineare non permette di trarre inferenze, e viene generato un modello lineare che non permette di rappresentare adeguatamente dipendenze non lineari. Quando poi si sceglie di operare su dataset particolarmente estesi la rappresentazione grafica non sempre è efficiente, questo è dovuto alla densità che si viene a creare. Sebbene indaghi la dimensione latente, per farlo opera su un certo numero di dimensioni, che non possono eccedere il rango della matrice; è un limite osservabile soprattutto con matrici a basso rango. Pur permettendo di distinguere parole che presentano polisemia, lo fa tramite algoritmi che funzionano spesso ma non

sempre. Il vocabolario prevalentemente usato è quello inglese: lo strumento offre la possibilità di lavorare su una moltidudine di idiomi ma con una lingua come quella italiana bisogna prestare particolare attenzione a come venga rielaborata dalla macchina e necessita di un maggiore intervento umano. Infine il modello prodotto non è leggibile da un umano e non facilmente interpretabile se non per mezzo di computazioni.

Bibliografia

Berry, Michael W (1992). «Large-scale sparse singular value computations». In: *The International Journal of Supercomputing Applications* 6.1, pp. 13–49.

Berry, Michael W e Murray Browne (2005). *Understanding search engines: mathematical modeling and text retrieval*. Vol. 17. Siam.

Berry, Michael W, Zlatko Drmac e Elizabeth R Jessup (1999). «Matrices, vector spaces, and information retrieval». In: *SIAM review* 41.2, pp. 335–362.

Berry, Michael W, Susan T Dumais e Gavin W O'Brien (1995). «Using linear algebra for intelligent information retrieval». In: *SIAM review* 37.4, pp. 573–595.

Bunch, James R, W James Demmel e Charles F Van Loan (1989). «The strong stability of algorithms for solving symmetric linear systems». In: *SIAM Journal on Matrix Analysis and Applications* 10.4, pp. 494–499.

Deerwester, Scott et al. (1990). «Indexing by latent semantic analysis». In: *Journal of the American society for information science* 41.6, pp. 391–407.

Dumais, Susan T (1991). «Improving the retrieval of information from external sources». In: *Behavior Research Methods, Instruments, & Computers* 23.2, pp. 229–236.

Foltz, Peter W, Walter Kintsch e Thomas K Landauer (1998). «The measurement of textual coherence with latent semantic analysis». In: *Discourse processes* 25.2-3, pp. 285–307.

Jessup, ER e JH Martin (2001). «Taking a new look at the latent semantic analysis approach to information retrieval». In: *Computational information retrieval* 2001, pp. 121–144.

Jiang, Jingqian et al. (1999). «Mining consumer product data via latent semantic indexing». In: *Intelligent Data Analysis* 3.5, pp. 377–398.

Landauer, Thomas K (2002). «On the computational basis of learning and cognition: Arguments from LSA». In: *Psychology of learning and motivation*. Vol. 41. Elsevier, pp. 43–84.

Landauer, Thomas K e Susan T Dumais (1997). «A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge.» In: *Psychological review* 104.2, p. 211.

Landauer, Thomas K et al. (2013). *Handbook of latent semantic analysis*. Psychology Press.

Letsche, Todd A e Michael W Berry (1997). «Large-scale information retrieval with latent semantic indexing». In: *Information sciences* 100.1-4, pp. 105–137.

Lizza, Marco e Flavio Sartoretto (2001). «A comparative analysis of LSI strategies». In: *Computational information retrieval*, pp. 171–181.

Salton, Gerard (1991). «Developments in automatic text retrieval». In: *science* 253.5023, pp. 974–980.

Schreiner, ME et al. (1997). «How latent semantic analysis (LSA) represents essay semantic content: Technical issues and analysis». In: *Proceedings 19th Annual Meeting of the Cognitive Science Society*, p. 1041.

Witter, Dian I e Michael W Berry (1998). «Downdating the latent semantic indexing model for conceptual information retrieval». In: *The Computer Journal* 41.8, pp. 589–601.

Codice R utilizzato nel Capitolo 2

Codice 1: Pacchetti utilizzati

```
library(quanteda); library(tm); library(lsa);
library(LSAfun); library(xtable)
```

Codice 2: Preprocessamento esempio

```
M1 <- c("Rock Roll Music")
M2 <- c("Drum Roll Demonstration")
M3 <- c("Drum Composition")
M4 <- c("Rock Music")
M5 <- c("Music Composition")
B1 <- c("Bread Roll Demonstration")
B2 <- c("Ingredients Roll")
B3 <- c("Recipe dough Bread")
B4 <- c("Recipe Dough Ingredients")
```

Codice 3: Preprocessamento creazione corpus

```
M1_c <- corpus(M1)
M2_c <- corpus(M2)
M3_c <- corpus(M3)
M4_c <- corpus(M4)
M5_c <- corpus(M5)
B1_c <- corpus(B1)
B2_c <- corpus(B2)
B3_c <- corpus(B3)
B4_c <- corpus(B4)
col_c <- M1_c + M2_c + M3_c + M4_c + M5_c + B1_c + B2_c + B3_c + B4_c
```

Codice 4: Spazio semantico tokenizzazione

```
M1_t <- M1_c %>%
  tokens() %>%
  tokens_tolower()
B1_t <- B1_c %>%
  tokens() %>%
  tokens_tolower()
col_t <- col_c %>%
  tokens() %>%
  tokens_tolower()
```

Codice 5: Creazione matrici DFM

```
M1_d <- dfm(M1_t, stem = FALSE)
B1_d <- dfm(B1_t, stem = FALSE)
col_d <- dfm(col_t, stem = FALSE)
```

Codice 6: Spazio semantico aggiustamento matrice

```
collection_m <- as.matrix(col_d)
collection_m <- t(collection_m)
collection_m <- collection_m[order(rownames(collection_m)),]
colnames(collection_m) <- c("M1", "M2", "M3", "M4", "M5", "B1", "B2",
        "B3", "B4")
```

Codice 7: Spazio semantico pesatura locale e globale

```
tfidf_lista <- dfm_tfidf(col_d,scheme_tf = "count", scheme_df = "inverse")
tfidf_lista <- t(tfidf_lista)
colnames(tfidf_lista) <- c("M1", "M2", "M3", "M4", "M5", "B1", "B2", "B3",
     "B4")
tfidf_lista <- tfidf_lista[order(rownames(tfidf_lista)),]
```

Codice 8: SVD spazio semantico

```
svd(tfidf_lista)
```

Codice 9: Creazione spazio semantico latente

```
myLSAspace <- lsa(tfidf_lista, dims=dimcalc_share())
lambda <- myLSAspace$sk/sum(myLSAspace$sk) #approssimazione dei valori
    singolari
TCS <- myLSAspace$tk[,1:2]%*%diag(myLSAspace$sk[1:2]) #termini nello "
    spazio semantico"
DCS <- diag(myLSAspace$sk[1:2])%*%t(myLSAspace$dk[,1:2]) #documenti nello
    "spazio semantico"
```

Codice 10: Creazione grafico

```
plot(x = TCS[,1], y = TCS[,2], xlab="x", ylab="y", xlim = c(-1,0), ylim =
    c(-0.5,.75), pch=15, main = "Vicinanza termini–documenti all'interno
    dello spazio semantico")
text(TCS[,1], TCS[,2], labels=rownames(TCS), cex= 0.8, pos=3)
points(x = DCS[1,], y = DCS[2,], pch=17, col="blue")
text(DCS[1,], DCS[2,], labels=colnames(DCS), cex= 0.8, pos=3)
abline(h=0)
```

Codice 11: Conversione spazio semantico latente in matrice

```
myNewMatrix <- as.textmatrix(myLSAspace)
```

Codice 12: Ricerca query

```
q <- fold_in(query("Recipe for White Bread", rownames(myNewMatrix)),
    myLSAspace)
qd <- 0
for (i in 1:ncol(myNewMatrix)) {
  qd[i] <- cosine(as.vector(q),as.vector(myNewMatrix[,i]))
}
qd #visualizzare correlazione con i documenti
```

Codice R utilizzato nel Capitolo 3

Codice 13: Pacchetti utilizzati

```
library(quanteda);library(readtext);library(dplyr); library(lsa);
library(wordcloud); library(tm); library(ggplot2)
```

Codice 14: Preprocessamento

```
poker <- readtext("/Directory/poker.txt")
sport <- readtext("/Directory/sport.txt")
lotto <- readtext("/Directory/lotto.txt")
```

Codice 15: Caricamento stopwords

```
stopwords <- readLines("/Directory/stopwords.txt", encoding = "UTF-8")
```

Codice 16: Creazione corpus

```
poker_c <- corpus(poker)
lotto_c <- corpus(lotto)
sport_c <- corpus(sport)
azzardo <- poker_c + lotto_c + sport_c
```

Codice 17: Tokenizzazione

```
poker_t <- poker_c %>%
tokens(remove_numbers=TRUE, remove_punct=TRUE, remove_symbols=TRUE) %>%
tokens_tolower() %>%
tokens_select(stopwords, selection = "remove", padding = FALSE, verbose =
    TRUE)
azzardo_t <- azzardo %>%
```

```
tokens(remove_numbers=TRUE, remove_punct=TRUE, remove_symbols=TRUE) %>%
tokens_tolower() %>%
tokens_select(stopwords, selection = "remove", padding = FALSE, verbose =
    TRUE)
```

Codice 18: Creazione matrici DFM

```
poker_d <- dfm(poker_t, stem = FALSE)
sport_d <- dfm(sport_t, stem = FALSE)
lotto_d <- dfm(lotto_t, stem = FALSE)
azzardo_d <- dfm(azzardo_t, stem = FALSE)
```

Codice 19: Aggiustamento matrice

```
azzardo_m <- t(as.matrix(azzardo_d))
azzardo_m <- azzardo_m[order(rownames(azzardo_m)),]
```

Codice 20: Pesatura

```
tfidf_azzardo <- dfm_tfidf(azzardo_d,scheme_tf = "prop", scheme_df = "
    inverse")
tfidf_azzardo <- t(tfidf_azzardo)
tfidf_azzardo <- tfidf_azzardo[order(rownames(tfidf_azzardo)),]
```

Codice 21: Creazione spazio LSA

```
myLSAspace <- lsa(tfidf_azzardo, dims=dimcalc_share())
```

Codice 22: Grafico LSA

```
TDM_w <- lw_tf(azzardo_m) * gw_idf(azzardo_m) #usata un'altra funzione di
    pesatura perche permette di visualizzare i dati con maggiore
    dispersione e minore densita
myLSAspace <- lsa(TDM_w, dims=dimcalc_share())
lambda <- myLSAspace$sk/sum(myLSAspace$sk) #approssimazione dei valori
    singolari
TCS <- myLSAspace$tk[,1:2]%*%diag(myLSAspace$sk[1:2]) #termini nello "
    spazio semantico" (CS)
DCS <- diag(myLSAspace$sk[1:2])%*%t(myLSAspace$dk[,1:2])
```

```
dfT <- data.frame(TCS)
p <- ggplot(dfT,aes(x=X1,y=X2)) +
        geom_point(alpha = 0.3) +
        xlim(-25,2.5) +
        ylim(-20,40)+
        theme_minimal()
```

Codice 23: Ricerca con query

```
q <- fold_in(query("Giocare a poker ti rendera ricco", rownames(
    myNewMatrix)),myLSAspace)
qd <- 0
for (i in 1:ncol(myNewMatrix)) {
  qd[i] <- cosine(as.vector(q),as.vector(myNewMatrix[,i]))
}
q <- fold_in(query("Se vuoi vivere appieno gioca", rownames(myNewMatrix)),
    myLSAspace)
qd <- 0
for (i in 1:ncol(myNewMatrix)) {
  qd[i] <- cosine(as.vector(q),as.vector(myNewMatrix[,i]))
}
```

Codice 24: Ricerca vocabolario in comune tra i tre documenti

```
IA1 <- intersect(poker_d@Dimnames$features, lotto_d@Dimnames$features)
IAF <- intersect(IA1, sport_d@Dimnames$features)
```

Codice 25: Caricamento stopwords con vocabolario in comune

```
stopwordsL <- readLines("/Directory/stopwordsL.txt",
                        encoding = "UTF-8")
stopwordsS <- readLines("/Directory/stopwordsS.txt",
                         encoding = "UTF-8")
stopwordsP <- readLines("/Directory/stopwordsP.txt",
                         encoding = "UTF-8")
```

Codice 26: Tokenizzazione sui documenti con vocabolario in comune

```
poker_tF <- poker_c %>% tokens(remove_numbers=TRUE,
```

```
                                remove_punct=TRUE,
                                remove_symbols=TRUE) %>%
  tokens_tolower() %>%
  tokens_select(stopwordsP,
                selection = "remove",
                padding = FALSE,
                verbose = TRUE)
lotto_tF <- lotto_c %>% tokens(remove_numbers=TRUE,
                                remove_punct=TRUE,
                                remove_symbols=TRUE) %>%
  tokens_tolower() %>%
  tokens_select(stopwordsL,
                selection = "remove",
                padding = FALSE,
                verbose = TRUE)
sport_tF <- sport_c %>% tokens(remove_numbers=TRUE,
                                remove_punct=TRUE,
                                remove_symbols=TRUE) %>%
  tokens_tolower() %>%
  tokens_select(stopwordsS,
                selection = "remove",
                padding = FALSE,
                verbose = TRUE)
```

Codice 27: Creazione matrici DFM con vocabolario in comune

```
poker_dF <- dfm(poker_tF, stem = FALSE)
sport_dF <- dfm(sport_tF, stem = FALSE)
lotto_dF <- dfm(lotto_tF, stem = FALSE)
```

Codice 28: Ricerca parole più frequenti

```
topfeatures(poker_d, n)
topfeatures(sport_d, n)
topfeatures(lotto_d, n)
```

Codice 29: Calcolo diversità lessicale

```
sapply(c(poker_dF,sport_dF,lotto_dF), textstat_lexdiv)
```

Codice 30: Creazione WordCloud

```
set.seed(100)
textplot_wordcloud(poker_d,
                   min_count = 1,
                   random_order = FALSE,
                   rotation = .25,
                   color = c("deepskyblue","dodgerblue3","darkblue"))
set.seed(100)
textplot_wordcloud(sport_d,
                   min_count = 2,
                   random_order = FALSE,
                   rotation = .25,
                   color = c("deepskyblue","dodgerblue3","darkblue"))
set.seed(100)
textplot_wordcloud(lotto_d,
                   min_count = 2,
                   random_order = FALSE,
                   rotation = .25,
                   color = c("deepskyblue","dodgerblue3","darkblue"))
```