UNIVERSITÀ DI PISA

Corso di Laurea in Informatica Umanistica

RELAZIONE

Visualizzatore di spazi semantici: un approccio sintagmatico.

Candidato: *Alessandro Croce*

Relatore: *Alessandro Lenci*

Correlatore: *Maria Simi*

Anno Accademico 2011-2012

INDICE

Introduzione

Lo scopo principale di questo lavoro è di costruire un modello di rappresentazione pratica della grande quantità di dati contenuta sui due database ricavati da *Distributional Semantic Memory* riguardanti, nello specifico, la vicinanza paradigmatica e la vicinanza sintagmatica dei vocaboli della lingua inglese.

Questo lavoro pertanto si basa sugli studi di Semantica Distribuzionale, un settore che, negli ultimi anni, sta attraversando un periodo di crescita esponenziale. Questa crescita è stata dovuta da un lato alle nuove ipotesi, nell'ambito delle scienze cognitive, riguardanti l'organizzazione del lessico mentale, dall'altro lato grazie alla sempre più crescente disponibilità di repertori testuali digitali di grandi dimensioni. La Semantica, in generale, è quella parte della linguistica che studia il significato delle parole, dei gruppi di parole, delle frasi e dei testi. In particolare la Semantica Lessicale, si occupa dello studio del significato delle singole parole, intese come unità lessicali, e dei lessemi. La base di partenza del mio studio risiede in un campo relativamente nuovo della Semantica, la Semantica Distribuzionale.

Con la Semantica Distribuzionale si ha una nuova serie di approcci all'analisi del significato. Questo nuovo ramo di indagine della Semantica, nato in Linguistica Computazionale e nelle scienze cognitive, si propone di cambiare totalmente il metodo di approccio allo studio del significato, basato su una prospettiva empiristica, dove ricopre un ruolo cruciale la distribuzione statistica delle parole nei contesti linguistici.

Nel particolare, la mia indagine verterà sugli studi di Alessandro Lenci e Marco Baroni riguardanti *Distributional Semantic Memory*. Si tratta di un modello costituito da un *grafo* con collegamenti pesati tra concetti estratto da un corpus. A partire dalle tuple, che possono essere estratte da questo *grafo*, vengono derivati più spazi semantici, ognuno dei quali può essere utilizzato per lo studio e la verifica di specifici *task* semantici, come misurare la similarità tra le parole, la categorizzazione dei vari termini e le analogie relazionali tra coppie di unità lessicali.

L'adattamento a tutti questi compiti (e molti altri) può essere ridotto a due operazioni basilari:

1) la costruzione di spazi semantici, come matrici di co-occorrenza, definite scegliendo le diverse unità del *grafo* come elementi di riga e colonna;
2) la misurazione della similarità nella matrice risultante sia tra righe specifiche che tra una riga e una media delle righe i cui elementi condividono una certa proprietà.

Delle molteplici chiavi di lettura dateci della *Distributional Semantic Memory* abbiamo approfondito lo studio riguardante i Vicini Paradigmatici e i Vicini Sintagmatici basandoci su due distinti database creati a partire dal modello *Type*DM.
Lo scopo finale di questo lavoro è stato di integrare tale quantità di dati con un sistema di visualizzazione pratica ed intuitiva. A tale proposito è stata scelta una rappresentazione mediante un *grafo radiale (R-Graph)* che è strutturato secondo l'architettura *Json*.
In questa breve relazione verranno perciò descritte le nozioni di base di Semantica Distribuzionale e in particolare gli studi di *Distributional Semantic Memory.* Dopo aver brevemente descritto l'uso e la struttura dei grafi, con in particolare la libreria *Javascript Infovis* (a cui si riferisce *R-Graph*), utilizzata per la rappresentazione grafica dei dati, saranno trattate nel dettaglio le operazioni preliminari e gli aspetti più tecnici dell'oggetto della tesi.

1 – Semantica Distribuzionale

1.1 Introduzione

I computer capiscono ben poco del significato del linguaggio umano. Ciò limita profondamente la nostra capacità di dare istruzioni al computer, la capacità dei computer stessi di spiegarci le loro azioni e la loro capacità di analizzare ed elaborare un testo. La Semantica Distribuzionale si muove in questa direzione tentando, appunto, di aggirare questi limiti proponendo uno studio del lessico da un punto di vista semantico, ponendo come suo concetto cardine la co-occorrenza nei testi dei termini lessicali.
Il principio epistemologico fondamentale che caratterizza la Semantica Distribuzionale è la cosiddetta *Ipotesi Distribuzionale* sviluppato da Zellig Sabbettai Harris (1954), secondo il quale, *due parole sono tanto più semanticamente simili quanto più tendono a ricorrere in contesti linguistici simili.*

"*Il grado di somiglianza semantica tra due espressioni linguistiche A e B è una funzione della somiglianza dei contesti linguistici in cui A e B possono co-occorrere*."[1]

Tale modello viene ben inquadrato dalle parole del linguista inglese John Ruipert Firth :

"You shall know a word by the company it keeps. An abstraction of information in the set of natural linguistic context in which a word occurs."[2]

Egli sosteneva che la lingua non deve essere studiata come un sistema mentale isolato, ma come una risposta a particolari situazioni contesto. Grazie a Firth, infatti,

1 Lenci, 2008.
2 Traduzione: *Si dovrebbe conoscere una parola basandosi sulle altre che la accompagnano. Un'informazione astratta nel contesto linguistico naturale in cui la parola occorre.*

vi è una valorizzazione di tutto ciò che fa da contorno ad una parola e quindi il significato di una parola deve essere dedotto dai legami che instaura con le altre unità linguistiche presenti nel medesimo contesto.

Anche se il metodo distribuzionale è in grado di rilevare la similarità semantica tra differenti espressioni linguistiche, non lo è parimenti nel riconoscere altri tipi di relazioni. Ciò a cui si limita l'*Ipotesi Distribuzionale* è misurare la distanza semantica che si ha tra due o più elementi all'interno di uno spazio semantico e ciò è sufficiente unicamente per definire i rapporti di sinonimia tra termini ma non per rilevare altri tipi di relazioni semantiche.

Considerazioni a tal proposito sono state fatte da Peter D. Turney (2006) che riteneva necessaria una distinzione tra similarità attributiva e similarità relazionale.

La similarità attributiva tra due termini è data dal fatto che i due concetti espressi presentano molti attributi in comune e perciò, di conseguenza, condividono molti contesti linguistici. La similarità attributiva tra due parole *a* e *b*, sim_a (*a*, i) $\in \Re$ dipende dal grado di corrispondenza tra le proprietà di *a* e *b*. Maggiore è la corrispondenza tra le loro proprietà, maggiore sarà la loro similarità attributiva.[3] Un tipico riscontro in questa direzione si può vedere nel caso dei sinonimi.

La similarità relazionale non riguarda gli attributi tra le singole parole bensì la corrispondenza tra la relazione che intercorre tra coppie di parole ed è tipica degli iperonimi[4] e degli iponimi[5]. La similarità relazionale tra le coppie di parole *a* : *b* e *c* : *d*, sim_r (*a* : *b*; *c* : *d*) $\in \Re$ dipende dal grado di corrispondenza tra le relazioni di *a* : *b* e *c* : *d*. Maggiore è la corrispondenza tra di loro, maggiore è sarà la loro similarità relazionale.[6] Per esempio i termini *giacca* e *cappotto* hanno un alto grado di similarità attributiva poiché i loro significati condividono molti attributi (entrambi sono degli indumenti e servono per coprirsi); invece *pesce* : *mare* e *uccello* : *cielo* hanno un alto grado di similarità relazionale (queste coppie di parole sono legate dalla comune relazione *animale* : *habitat*).

3 Turney (2006).

4 Parola che ha un significato generale rispetto ad altre, più specifiche (e.g. *animale* rispetto a *cane, gatto*).

5 Parola di significato più specifico rispetto ad un'altra, più generale (e.g. *tulipano* rispetto a *fiore*).

6 Turney (2006).

1.2 Spazi Semantici Vettoriali

Gli Spazi Semantici Vettoriali sono dei modelli algebrici per la rappresentazione delle parole sotto forma di vettori. Costituiscono una delle nuove tecnologie semantiche in grado di dare un'ulteriore spinta a quel processo che vede i computer avvicinarsi sempre più alla comprensione del linguaggio umano.

Il lessico viene concepito come uno spazio semantico di parole[7], i cui elementi sono separati da distanze che dipendono dal loro grado di similarità semantica. Viene proposto *un accostamento analogico tra le proprietà del significato e le proprietà dello spazio.*[8] La nozione di spazio semantico si basa su una semplice analogia con lo spazio geometrico. Come ciascun punto dello spazio è definito da un vettore di *n* numeri che rappresentano le sue coordinate rispetto a *n* assi cartesiani (le dimensioni dello spazio), così il contenuto semantico di una parola è rappresentato dalla sua posizione in uno spazio definito da un sistema di coordinate, determinato dai contesti linguistici in cui la parola può ricorrere.

Pertanto, secondo questo modello, le parole vengono proiettate in uno spazio metrico e ordinate secondo una certa distanza dipendente dal loro grado di similarità semantica.

Formalmente, uno spazio semantico di parole è definito dalla quadrupla *<T, B, M, S>*[9]. Dove *T* è l'insieme delle parole *target* che formano gli elementi che popolano lo spazio e di cui questo fornisce una rappresentazione semantica. *B* è la base che definisce le dimensioni dello spazio e contiene i contesti linguistici rispetto ai quali viene valutata la similarità distribuzionale delle parole *target*. *M* è una matrice di co-occorrenza che fornisce una rappresentazione vettoriale di ogni parola in *T*.

I modelli computazionali di Semantica Distribuzionale differiscono per la nozione di contesto che adottano, ovvero per il modo in cui viene definita la base *B*. Nella versione più comune, i vettori registrano co-occorrenze tra parole di un testo.

Rappresentando ogni parola come un vettore a *n* dimensioni, ciascuna delle quali registra il numero di volte in cui la parola compare in un certo contesto definito dalla

7 Lenci: *Spazi di Parole. Metafore e Rappresentazioni Semantiche.*
8 Lenci: *Spazi di Parole. Metafore e Rappresentazioni Semantiche.*
9 Lowe, 2001; Padó e Lapata, 2007.

base *B*. Ogni parola *target* corrisponde, dunque, a una riga della matrice *M*, le cui colonne corrispondono invece agli elementi in *B*.

Le dimensioni del vettore assegnato a una parola, non hanno alcun valore semantico intrinseco, ma servono solo a determinarne la sua posizione nello spazio e la distanza dalle altre parole. Il significato nasce solo dalle configurazioni di punti nello spazio.

L'ultimo elemento che definisce la struttura dello spazio semantico è la metrica *S* che misura la distanza tra i suoi punti nello spazio. Per determinare la posizione di due parole, è necessario comparare i loro vettori rispetto a tutte le dimensioni che li costituiscono. Maggiore è il numero di dimensioni nelle quali due vettori presentano valori simili, maggiore è la loro vicinanza nello spazio e la similarità semantica delle parole corrispondenti.

Questi modelli, conosciuti come spazi vettoriali, spazi semantici, spazi di parole, modelli semantici basati sui corpus o modelli semantici distribuzionali si rifanno all'*Ipotesi Distribuzionale* di Harris.

Gli Spazi Semantici Vettoriali hanno diverse proprietà interessanti. Una tra tutte è la capacità di estrarre automaticamente delle informazioni da un corpus dato, richiedendo una mole di lavoro molto meno elevata rispetto ad altri approcci. Si è notato inoltre che gli Spazi Semantici Vettoriali hanno buone prestazioni per quanto riguarda la misura della somiglianza del significato tra parole, frasi e documenti.

Tutti i modelli però presentano un tipico approccio locale, in cui ogni *task* semantico viene trattato come un singolo problema e richiede un proprio modello derivato da corpus e un proprio algoritmo, entrambi ottimizzati per ottenere le migliori prestazioni riguardanti quel determinato *task* semantico.

Questo tipo di approccio rappresenta un grande limite in quanto si discosta dalla reale attività umana di acquisizione e utilizzo di informazioni linguistiche. Gli uomini infatti si rifanno ad un'unica *memoria semantica*, una sorta di *database* generale della conoscenza a lungo termine, adattando le varie informazioni contenutevi alle varie *attività* che si presentano[10].

Questo approccio ha gettato delle ombre sulla reale efficienza degli Spazi Semantici Vettoriali.

Si possono distinguere due tipologie di Spazi Semantici Vettoriali: con struttura e senza struttura.

10 Murphy, 2002; Rodgers and McClellend, 2004.

Nei primi le co-occorrenze statistiche sono raccolte sotto forma di triple: coppie di parole e relazione sintattica o *pattern* lessico-sintattico che le collega. Per esempio, nella frase *Il pilota guida la macchina nera,* nei modelli con struttura, la parola *guida* non è considerata come un contesto legittimo per la parola *nera,* in quanto sono legate unicamente dal fatto che compaiono nella stessa finestra di contesto[11] ma non presentano relazioni linguistiche vere e proprie, inoltre sarebbe fatta una distinzione tra la relazione che connette *guida* e *macchina* e la relazione che collega *macchina* a *nera*.

La seconda tipologia invece si limita a rappresentare unicamente la coppia di parole che che co-occorrono in un contesto, senza considerare il tipo di relazione sintattica. Considerando la medesima frase usata in precedenza, in questo caso la parola *guida* risulta una caratteristica condivisa da *macchina* e da *nera* in quanto compare nella stessa finestra di contesto, non considerando il fatto che non si ha una vera relazione linguistica che collega le due parole *guida* e *nera* se non la loro prossimità lineare. Nei modelli con struttura ricoprono perciò un ruolo fondamentale le strutture sintattiche che definiscono le proprietà distribuzionali delle parole. Una parola per essere qualificata come contesto di una parola *target* deve essere collegata ad essa tramite relazioni lessico-sintattiche.

1.2.1 Matrici di co-occorrenza

Se siamo in possesso di una grande raccolta di documenti, e quindi di un grande numero di vettori di documenti, è conveniente organizzare i vettori in una matrice. Questo tipo di struttura si adatta perfettamente agli approcci inerenti la distribuzione dei dati. Le righe della matrice corrispondono alle parole e le colonne corrispondono ai documenti in cui ricorrono (ad esempio pagine web). Si tratta di una matrice *termine-documento*. Questo tipo di matrici sono fondate sull'*Ipotesi del Contenitore*

[11] La finestra di contesto è un valore che dipende dal numero di *token* precedenti e successivi alla parola *target* che si è scelto di selezionare per formare le co-occorrenze. La dimensione della finestra di contesto è stata motivo di dibattito tra gli studiosi di linguistica computazionale, anche se i risultati migliori sono stati raggiunti con finestre di contesto a dimensioni ridotte, in quanto la parole di contesto più vicine alla parola target hanno un peso maggiore nel determinarne il significato.

di Parole[12]. Le frequenze delle parole presenti in un documento tendono a indicare la pertinenza del documento ad una determinata tematica, individuando perciò, in base alle parole che vi sono contenute, di cosa tratta. Se due documenti trattano tematiche simili, allora i due vettori colonna corrispondenti tendono ad avere valori simili.
Un vettore documento conterrà tanti elementi quante sono le parole tipo contenute nel documento, evitando pertanto ripetizioni date dalla presenza di medesime parole.
Questa matrice può essere rappresentata con la seguente tabella di esempio dove delle parole (termini) vengono poste in relazione con alcuni documenti.

	Documento A	*Documento B*	*Documento C*	*Documento D*
Cantare	74	6	0	0
Lavorare	5	58	56	0
Sparare	3	89	7	1
Studiare	0	3	35	4

Tabella 1: Matrice *termine-documento*.

Lo Spazio Semantico Vettoriale di Gerard Salton (1975) è stata, probabilmente, la prima applicazione pratica di quanto è stato detto precedentemente: un algoritmo per l'estrazione di informazioni semantiche dall'uso delle parole.
Salton si è concentrato esclusivamente sulla misurazione della somiglianza tra i documenti. Da questa base di partenza Scott Deerwester (1990) ha osservato che con i mezzi a nostra disposizione siamo in grado di spostare la nostra attenzione alla misurazione della somiglianza tra le parole prendendo in considerazione unicamente i vettori riga della matrice *termine-documento*, tralasciando ciò che è contenuto nei vettori colonna. Deerwester è stato ispirato dagli studi di Salton, ma si accorse che il documento non rappresenta necessariamente una dimensione ottimale per misurare la somiglianza tra le parole. In questo caso si parla di una matrice *parola-contesto*.
Questo tipo di matrice registra quindi le co-occorrenze tra una parola *target* (riga) e una parola facente parte della sua finestra di contesto (colonna). Se le co-occorrenze si presentano in maniera frequente vicine tra di loro in un enunciato vengono

12 Salton et al., 1975: *Bag of words hypotesis*.

chiamate *collocazioni* e, nello specifico, l'elemento impegnato in tale collocazione è chiamato *collocato*. L'ampiezza della finestra massima di parole che separano il *target* dal collocato è un parametro che viene fissato a priori.
Nella tabella seguente viene rappresentata questo tipo di matrice, dove sia gli elementi delle righe che quelli delle colonne sono rappresentate da parole, le parole *target* nel primo caso e le parole di contesto nel secondo caso.

	correre	*aprire*	*leggere*	*felice*
frigorifero	0	15	0	0
cane	17	0	0	2
automobile	23	0	0	0
libro	0	12	35	0

Tabella 2: Matrice *parola-contesto*.

Un'ultima tipologia di matrice prevede che i vettori riga siano occupati da coppie di parole (come ad esempio *contadino : terra*) e i vettori colonna invece corrispondano a dei *pattern* in cui la coppia co-occorre (in questo esempio il *pattern* potrebbe essere riassunto nel modo **X** *semina* **Y**). La matrice *coppia-pattern* è stata introdotta da Lin e Pantel (2001) con lo scopo di misurare la similarità semantica di alcuni *pattern*, cioè dei vettori colonna. I due ricercatori proposero l'*Ipotesi Distribuzionale Estesa* che sostiene che *pattern* che co-occorrono con simili coppie di parole tendono ad avere simili significati.
Peter D. Turney et al. (2003) decisero di utilizzare questa tipologia di matrice per calcolare la similarità semantica delle coppie di parole che co-occorrono in *pattern* simili spostando la loro attenzione sui vettori riga piuttosto che sui vettori colonna. A partire da queste considerazioni formularono l'*Ipotesi di Relazione Latente* la quale sostiene che coppie di parole che ricorrono in simili *pattern* tendono ad avere simili relazioni semantiche. Nella tabella sottostante si ha una rappresentazione della matrice appena descritta.

	Pattern A	*Pattern B*	*Pattern C*	*Pattern D*
Coppia 1	5	1	0	0
Coppia 2	1	0	0	4
Coppia 3	0	1	3	0
Coppia 4	0	5	0	1

Tabella 3: matrice *coppia-pattern*.

Come si può notare in tutte e tre le matrici si ha una rappresentazione sparsa dei dati in quanto la maggior parte dei suoi elementi presenterà un valore pari a zero. Altri elementi invece presentano valori molto alti. Questa disuguaglianza è anche dovuta alla presenza nei testi di quei termini come ad esempio gli articoli, gli avverbi o verbi, come gli ausiliari *essere* ed *avere,* che ricorrono in maniera molto più frequente rispetto agli altri termini ma che, a differenza di questi, spesso apportano un contributo inutile all'indagine che si vuole fare. Una soluzione a questo problema potrebbe essere l'esclusione a priori di questo tipo di elementi e considerare unicamente i vettori che condividono coordinate il cui valore sia diverso da zero.
Questo sistema però non si è rivelato totalmente efficace per isolare gli eventi linguistici di maggiore interesse da quelli che hanno scarsa importanza per l'indagine linguistica. Per questo motivo è stato attribuito un peso statistico appropriato a determinati elementi in modo tale da mettere maggiormente in risalto unicamente quelli rilevanti.
E' stato attribuito un maggiore peso statistico a quei termini che ricorrono con elevata frequenza in un documento ma sono rari in tutti gli altri documenti del corpus. L'ipotesi è che questi elementi, se condivisi da due vettori sono molto più discriminanti della similarità tra i vettori. Ad esempio, per misurare la similarità semantica tra i termini *topo* e *ratto* hanno molto più valore i contesti *sezionare* e *sterminare* piuttosto che i contesti *essere* ed *avere*. Questo concetto, per le matrici *termine-documento,* viene definito *tf-idf* (*term frequency–inverse document frequency)*[13] dove appunto il peso di un elemento che è molto frequente in un documento cresce in modo inversamente proporzionale alla sua frequenza complessiva in tutto il corpus.

13 Spark Jones, 1962.

Per ottenere la frequenza di un termine in un documento perciò sarà necessario calcolare il rapporto tra il numero di occorrenze dello stesso e la dimensione totale del documento.
Se ci troviamo di fronte un matrice del tipo *parola-contesto* il metodo più diffuso per pesare i contesti è la *mutual information* (MI), la quale attua un confronto tra la probabilità di riscontrare in un documento co-occorrenze di coppie di termini rispetto alla probabilità di riscontrare gli stessi termini individualmente.

$$MI = \log_2 \frac{p(t_i, t_j)}{p(t_i)p(t_j)}$$

Laddove p indica la probabilità e t_i e t_j, prima presi in coppia e poi singolarmente, indicano la coppia di parole in esame.
Un approccio di questo tipo però favorisce in maniera eccessiva le coppie che nei documenti si presentano raramente, come ad esempio i termini che all'interno dell'intero corpus si presentano una sola volta. Si tratta di un caso limite e, in questo caso, la frequenza, sia per quanto riguarda la coppia che, di riflesso, anche i singoli costituenti, sarà pari a 1 e perciò presenterà una *mutual information* tra le più alte di tutte quelle calcolate nell'intero corpus.
Una possibile alternativa alla MI è la *local mutual information* (LMI) che moltiplica il valore ottenuto per la *mutual information* per il numero di occorrenze della coppia per la quale è stata calcolata. Con questo espediente sarà possibile attribuire un peso statistico maggiore alle coppie che si presentano con maggior frequenza all'interno del corpus.

$$LMI = f(< t_i, t_j >) \cdot \log_2 \frac{p(t_i, t_j)}{p(t_i)p(t_j)}$$

Dopo aver opportunamente trattato i valori risultanti della matrice con la *local mutual information* si può procedere al confronto diretto tra i vettori riga. La misura

più comune usata per misurare la vicinanza spaziale tra due vettori è il *coseno* dell'angolo che essi formano. Se due vettori sono geometricamente allineati sulla stessa linea nella stessa direzione, l'angolo che si forma tra di loro è 0°, e il coseno è 1 (massima similarità); viceversa se i due vettori sono indipendenti e perciò ortogonali, il loro angolo è vicino a 90° e il coseno è uguale a 0 (assenza di similarità)[14].

1.3 Distributional Memory

La *Distributional Memory*, nata dagli gli studi effettuati da Alessandro Lenci e Marco Baroni, propone un diverso approccio alla semantica basata sui corpus. Tale modello è costituito da diversi spazi semantici derivati dalle informazioni estratte a partire da tuple, estratte a loro volta da un corpus di grandi dimensioni, che contengono informazioni riguardanti unità lessicali collegate tra di loro. La forza di tali collegamenti è espressa tramite un peso calcolato tramite la LMI. Tutte queste informazioni costituiscono la base per ulteriori ricerche semantiche quali la misurazione della similarità semantica, la categorizzazione di tali unità concettuali e le analogie relazionali tra coppie di unità lessicali.

Distributional Memory appartiene alla famiglia dei modelli di Semantica Distribuzionale *con struttura,* che tengono conto del ruolo cruciale svolto dalle strutture sintattiche nella definizione delle proprietà distribuzionali delle parole.

Per qualificarsi come contesto di una parola *target*, una parola deve essere collegata ad essa mediante relazioni lessico-sintattiche. A differenza di altri modelli con struttura, la struttura a tuple è rappresentata come un oggetto geometrico a 3 elementi, vale a dire un tensore di terzo ordine.

I tensori sono oggetti geometrici che descrivono relazioni lineari tra vettori (tensori di I grado) e altri tensori[15]. Un tensore può essere rappresentato come un *array* multi-dimensionale di valori numerici.

14 Lenci: *Spazi di Parole. Metafore e Rappresentazioni Semantiche.*
15 http://it.wikipedia.org/wiki/Tensore

Le funzionalità della *Distributional Memory* possono essere riassunte in due grandi categorie: la costruzione di spazi semantici, a partire da matrici di co-occorrenza definite selezionando diverse unità lessicali, e la misurazione della similarità nella matrice risultante tra specifiche righe in cui gli elementi che le costituiscono condividono particolari proprietà.

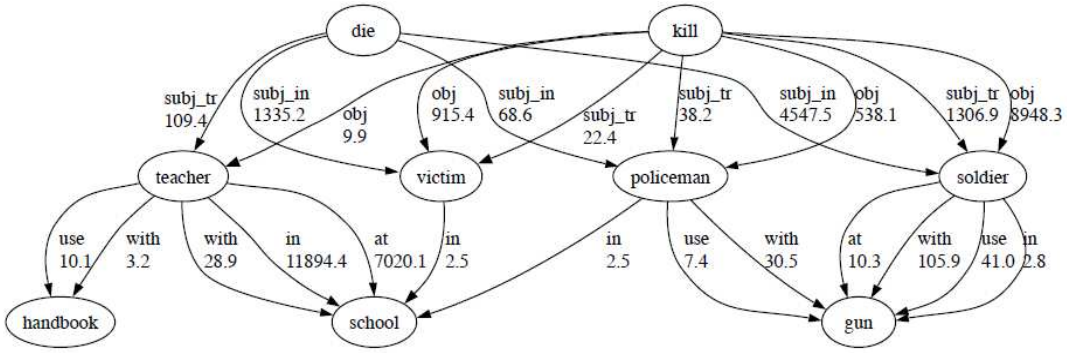

Figura 1:Grafo che rappresenta un piccolo frammento di *Distributional Memory*.

Distributional Memory, come anticipato, è costituita da tuple pesate del tipo *unità lessicale + link + unità lessicale* estratte da un corpus di grandi dimensioni. Le *unità lessicali* nelle tuple sono generalmente parole. Il *link* contiene informazioni su come le due *unità lessicali* siano collegate tra di loro in un contesto. Infine è presente il *peso* che in genere deriva dal conto delle co-occorrenze degli elementi della tupla, calcolato via *local mutual information*. Il modo in cui le tuple sono identificate e pesate quando popolano la memoria è di fondamentale importanza per la qualità del modello risultante.

Da *Distributional Memory* sono stati ricavati 3 diversi modelli che corrispondono ai diversi modi di costruire la struttura delle tuple. Tutti e 3 si basano sull'idea generale di estrarre tuple del tipo *unità lessicale + link + unità lessicale.*

Tali modelli sono stati estratti da 3 corpora: ukWaC[16], di circa 1.915 miliardi di *token;* Wikipedia Inglese[17], scaricato a metà 2009, di circa 820 milioni di *token*; e il

16 http://wacky.sslmit.unibo.it

British National Corpus[18] di circa 95 milioni di *token*. Il risultante corpus nato dall'unione di questi tre è stato tokenizzato[19], annotato[20] e lemmatizzato[21] con TreeTagger[22] e analizzato con il *parser*[23] MaltParser[24]. Il corpus risultante contiene circa 2,83 miliardi di *token*, da cui sono stati selezionati i 20.000 nomi e i 5.000 verbi e aggettivi più frequenti come parole *target*, da questo computo naturalmente vengono esclusi i termini di *stop*[25].

***Dep*DM**. Questo modello si basa sull'intuizione classica secondo cui le dipendenze sintattiche sono una buona approssimazione delle relazioni semantiche tra parole. Questo modello, tra quelli derivati da DM, presenta il più basso grado di lessicalizzazione dei *link*.

Le coppie di parole delle tuple sono del tipo nome-verbo, nome-nome e aggettivo-nome e presentano varie tipologie di *link*:

sbj_intr: soggetto di un verbo che non presenta un oggetto diretto (*The president talks → president*, sbj_intr, *talk*);

sbj_tr: soggetto di un verbo che presenta un oggetto diretto (*The carpenter cuts wood → carpenter*, sbj_tr, *cut*);

obj: oggetto diretto (*The student eats an aple → apple*, obj, *eat*);

iobj: oggetto indiretto in costruzioni con doppio oggetto (*The teacher gave the book to the student → student*, iobj, *give*);

nmod: aggettivo qualificativo di un nome (*A clever cheater → clever*, nmodm, *cheater*);

coord: nomi coordinati (*Ladies and Gentlemen → ladies*, coord, *gentlemen*);

prd: nome predicativo (*The caterpillar becomes a butterfly → caterpillar*, prd, *become*);

---

17 http://en.wikipedia.org/wiki/Wikipedia:Database_download

18 http://www.natcorp.ox.ac.uk

19 Operazione mediante la quale si suddivide il testo in *token*, è relativamente semplice per lingue che adoperano gli spazi per delimitare le parole; molto complessa per lingue a sistema ortografico continuo.

20 I corpus annotati sono corpus in cui viene codificata dell'informazione linguistica in associazione al testo.

21 Ricondurre le parole al proprio lemma.

22 http://www.ims.uni-stuttgard.de/projekte/corplex/TreeTagger/

23 Un parser è un programma che analizza uno *stream* continuo di input in modo da determinare la sua struttura grammaticale grazie ad una data grammatica formale.

24 http://www.msi.vxu.se/~nivre/research/MaltParser.html

25 Unità lessicali che hanno scarso valore concettuale che si presentano con ampia frequenza nel testo come articoli, avverbi o i comuni verbi ausiliari essere e avere.

verb: collegamento tra soggetto e complemento (*The cat eats the meat → cat*, verb, *meat*);
preposition: preposizione che collega un nome con un altro o con un verbo (*The priest talks with people → priest*, with, *talk*).

Di queste tuple ne è stato calcolato il peso usando la *local mutual information* (LMI). Le tuple con LMI minore o uguale a zero sono state scartate. Per ogni tupla *unità lessicale + link + unità lessicale* è stato aggiunto, in tutti i modelli, anche la stessa corrispondenza al contrario col medesimo peso *unità lessicale + link-1 + unità lessicale.*
Questo modello presenta 796 *link* differenti. Il numero di tuple presenti in questo tensore è di circa 110 milioni, includendo sia quelle dirette che le inverse.

***Lex*DM**. Questo modello si basa sull'idea che il materiale lessicale che collega due parole risulta molto importante nell'individuare la loro relazione sintagmatica. Qui, tra i modelli di DM, si ha il più alto livello di lessicalizzazione dei *link*. La struttura delle tuple sarà composta, oltre che dalle due parole e dal *link*, anche da un suffisso, formato da due sotto-stringhe separate dal simbolo +, le quali possono presentare caratteristiche morfologiche (viene esplicitato infatti se si tratta di nomi *n*, aggettivi *j* o verbi *v*, questi ultimi con tempo verbale ed eventuale ausiliare *aux*), articoli, avverbi o aggettivi connessi alle parole *target*.

Le tipologie di *link* presenti in questo modello sono:
verb: verbo che collega il soggetto con un complemento, se il verbo è presente nella lista composta dai 52 verbi con maggiore frequenza allora il *link* è lo stesso verbo scritto nella sua forma originale (*The teacher drinks mineral water → teacher*, verb+*n-the+n-j*, *water*);
is: predicato nominale composto da copula e aggettivo (*The shoe is dirty → dirty*, is+*j+n-the*, *shoe*);
preposition-link_noun-preposition: questo tipo di *link* include le espressioni del tipo *of a number of, in a kind of*; per un totale di 48 diverse selzionate semi-automaticamente. (*the sale of a number of cars → car*, of-number-of+*ns+n-the*, *sale*);

attribute_noun: uno dei 127 nomi estratti da Wordnet che esprimono le caratteristiche dei concetti come *size*, *color* oppure *height*. In questo caso ci riferiamo a quei *pattern* del tipo *attribute_noun of a NOUN is ADJ* o anche *ADJ attribute_noun of NOUN* (*the color of sea is blue → blue*, color+*j*+*n*, *sea*);
as_adj_as: in questo *pattern* sono collegati aggettivi e nomi della forma *as ADJ as NOUN* (*as bright as the sun → bright*, as_adj_as+*j*+*n-the*, *sun*);
such_as: collega due nomi presenti nella forma *NOUN such as NOUN* e s*uch NOUN as NOUN* (*plants such as oaks → plant*, such_as+*ns*+*ns*, *oak*).

Anche in questo caso il peso delle tuple è calcolato tramite la LMI.
Rispetto a *Dep*DM il grado di lessicalizzazione è il doppio. Questo modello presenta più di 3,3 milioni di *link*, compresi gli inversi, che generano 335 milioni di tuple.

***Type*DM**. Questo modello è basato sull'idea che ciò che conta non è tanto la frequenza di un *link*, ma la varietà delle forme che lo esprimono. E' chiamato *Type*DM perché appunto conta i tipi di forme del *link*, non i singoli *token*. La LMI non è più calcolata sull'occorrenza della tripla *unità lessicale* + *link* + *unità lessicale,* ma sul numero di differenti tipi di suffissi presenti su un medesimo *link* quando occorre su due parole.
TypeDM è il più efficiente dei tre modelli. In totale contiene 30.693 lemmi (20.410 nomi, 5.026 verbi e 5.257 aggettivi). Si tratta dei 20.000 nomi e dei 5.000 verbi e aggettivi più frequenti nel corpus. Questo modello presenta inoltre 25.336 *link* differenti, con un totale di 130 milioni di tuple, includendo anche gli inversi, collocandosi in una posizione intermedia tra i modelli proposti.

1.3.1 Spazi Semantici e Matrici di DM

Diversi spazi semantici vengono generati *on demand* attraverso la *matricizzazione* del tensore. La matricizzazione consiste nel trasformare un tensore (array

multidimensionale) in una matrice (tensore di II ordine). A seconda dell'ordine in cui sono state selezionate le fibre[26] del tensore si otterranno diverse matrici.

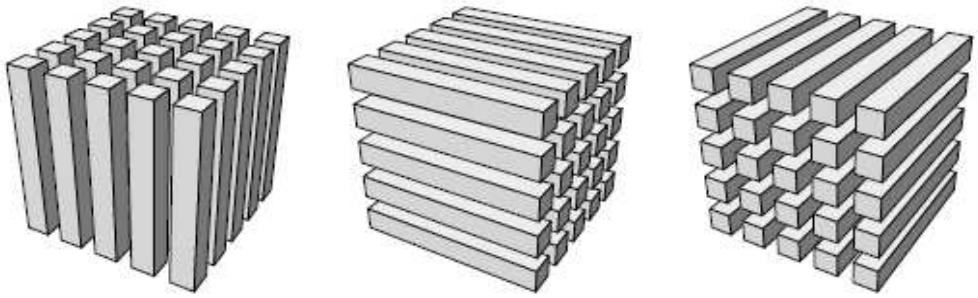

Figura 2: Illustrazione grafica di tensori matricizzati.

Da *Distributional Memory* si possono ricavare tre differenti tipi di matrici, ognuna delle quali è stata testata su uno o più *task* semantici e confrontata con gli altri principali modelli esistenti. Le matrici sono del tipo: *unità lessicale-link + unità lessicale (CxLC, concept-by-link + concept), unità lessicale + unità lessicale-link (CCxL, concept + concept-by-link), unità lessicale + link-unità lessicale (CLxC, concept + link-by-concept).*
DM a differenza degli altri modelli consente un approccio globale a qualsiasi *task* semantico, basta unicamente organizzare le matrici in maniera differente ma comunque utilizzando le medesime informazioni.

CxLC: Uno dei compiti principali della linguistica computazionale risiede nella misurazione della similarità tra parole in base al loro contesto di co-occorrenza con altre parole[27]. A questo fine le informazioni estrapolate dal grafo sono organizzate in una matrice in cui le *unità lessicali* (i nodi) sono righe, e i nodi connessi agli archi che ne fuoriescono, insieme all'informazione stessa che dà l'arco (il *link*), sono colonne. Con questa matrice possiamo comparare i vettori riga usando tipiche

26 Una fibra di un tensore è un frammento unidimensionale ottenuto fissando tutti gli indici eccetto uno.
27 Sahlgren, 2006.

tecniche geometriche in modo tale da focalizzare le proprietà condivise dalle parole come ad esempio la similarità tassonomica (sinonimi e contrari).

	$subj\_in^{-1}$ die	$subj\_tr^{-1}$ kill	obj^{-1} kill	with gun	use gun
teacher	109.4	0.0	9.9	0.0	0.0
victim	1335.2	22.4	915.4	0.0	0.0
soldier	4547.5	1306.9	8948.3	105.9	41.0
policeman	68.6	38.2	538.1	30.5	7.4

Tabella 4: Un frammento dello spazio semantico di tipo CxLC.

CCxL: Un'altra matrice che si può ricavare da *Distributional Memory* ha come vettori riga le due *unità lessicali* in coppia (i nodi del grafo) e come vettori colonna le etichette degli archi che le congiungono (i *link*). Questa matrice contiene le medesime informazioni della precedente ma presenta una diversa disposizione delle stesse, questo si può costatare dal *peso* che compare nell'incrocio tra le rispettive righe e colonne per entrambe le matrici. Questa matrice è usata per misurare la similarità relazionale tra le coppie di *unità lessicali*[28].

		in	at	with	use
teacher	school	11894.4	7020.1	28.9	0.0
teacher	handbook	2.5	0.0	3.2	10.1
soldier	gun	2.8	10.3	105.9	41.0

Tabella 5: Un frammento dello spazio semantico di tipo CCxL.

CLxC: Un'ultima matrice che si può ricavare dal grafo di *Distrutional Memory* mostra *pattern* di similarità tra i vari soggetti o oggetti dei verbi predicativi. Si possono perciò estrarre delle generalizzazioni trovando delle analogie di significato su diverse combinazioni lessico-semantiche. Nella tabella possiamo notare come gli oggetti di *uccidere* (*kill*) siano piuttosto simili ai soggetti di *morire* (*die*).

28 Turney, 2006.

		teacher	victim	soldier	policeman
kill	subj_tr	0.0	22.4	1306.9	38.2
kill	obj	9.9	915.4	8948.3	538.1
die	subj_in	109.4	1335.2	4547.5	68.6

Tabella 6: Un frammento dello spazio semantico di tipo ClxC.

Anche in questa matrice non abbiamo nessuna nuova informazione ma si tratta semplicemente di una diversa riorganizzazione dei dati a disposizione di *Distrutional Memory*.

2 – Visualizzazione dell'Informazione

2.1 Introduzione

La statistica è la scienza che studia i dati che si riferiscono a fenomeni di cui si desidera avere una comprensione e che serve sintetizzare ricorrendo a metodi matematici.
La raccolta di informazioni e in particolare di dati numerici è lavoro quotidiano di chi si occupa di scienza. Una volta raccolti i dati necessari per le proprie analisi, si deve scegliere una o più modalità efficaci per evidenziare il valore informativo contenuto nei dati stessi. Una scelta potrebbe essere quella di una rappresentazione tabellare dei dati oppure, in maniera più intuitiva, si preferisce rappresentare i dati raccolti sotto forma di grafico per rendere più facile la lettura delle informazioni a disposizione. Questo espediente infatti ci permette di cogliere velocemente le caratteristiche essenziali dei dati e confrontarli tra di loro. Tra i tipi di grafici maggiormente usati per questo tipo di rappresentazioni abbiamo i grafici a barre, i grafici a settori circolari (grafici a torta), gli istogrammi e i grafici a punti.

2.2 Javascript Infovis Toolkit

La *Javascript Infovis Toolkit* (JIT) è una libreria *JavaScript,* ideata e realizzata da Nicolas Garcia Belmonte, utilizzata per la rappresentazione di connessioni semantiche tra due o più entità. *Javascript Infovis Toolkit* fornisce specifiche strutture di dati al fine di ottenere una veloce combinazione azione-risultato in seguito ad una richiesta da parte di una *query* dinamica. E' corredato da un'ampia gamma di componenti integrati che semplificano la gestione di ricche strutture di dati, il design e l'estensione delle visualizzazioni.

Le caratteristiche principali di questo *toolkit* sono:

- **Struttura dei dati unificata**: la struttura base dei dati è una tabella di colonne contenenti oggetti di tipo omogeneo che possono essere, ad esempio, numeri o stringhe.
- **Minimo utilizzo della memoria:** utilizzando colonne omogenee piuttosto che di tipo composto, la memoria richiesta per memorizzare le tabelle, gli alberi o i grafi, migliora notevolmente e in generale anche il tempo impiegato per gestirli è ridotto.
- **Insieme unificato dei componenti interattivi:** usando questo *framework* unitario, un gran numero di componenti interattivi necessari per la visualizzazione delle informazioni sono generici e riutilizzabili per tutti i tipi di dati concreti e visualizzazioni dei dati stessi.
- **Veloce:** *Javascript Infovis Toolkit* può utilizzare la grafica accelerata fornita da *Agile2D*, un'implementazione di *Java2D* basata sulle *OpenGL API* per la grafica hardware accelerata. Su macchine con l'accelerazione hardware, alcuni effetti grafi vengono visualizzati 200 volte più velocemente rispetto a quelle standard con un'implementazione *Java2D*.
- **Estensibile:** *Javascript Infovis Toolkit* ha lo scopo di incorporare nuove tecniche di visualizzazione delle informazioni, inoltre viene distribuito con i codici sorgente e con licenza libera. Al momento del download oltre ad un file *jit.js*, di circa 17.000 righe di codice, che contiene tutte le funzioni generali della libreria, sono forniti anche i singoli file che si riferiscono ai grafi specifici e ci permettono di modificarne la struttura vera e propria.

Questo *toolkit* implementa funzioni avanzate di visualizzazione delle informazioni, nove tipologie in totale, come *TreeMaps*, una visualizzazione ad alberi della tipologia *SpaceTree*, una tecnica di tipo focus + contesto per tracciare *HyperTree* (alberi iperbolici), una disposizione radiale di alberi con animazioni chiamata *R-Graph*, matrici di adiacenza e diagrammi *Node-Link* (che forniscono visualizzazioni in diverse varianti) e altre tipologie di visualizzazioni più comuni come grafici a barre, grafici a torta e grafici ad aerea.

InfoVis Toolkit è organizzato in tre parti principali: tabelle e colonne, visualizzatori dei dati (grafi) e componenti.

Tabelle e Colonne: Un insieme di dati viene memorizzato sotto forma di tabella, in cui ogni riga rappresenta un record ogni colonna rappresenta un attributo. Questo schema è naturale per insiemi di dati tabulari ma ciò che sorprende è che ciò si ripropone anche per alberi e grafi. Alberi e grafi infatti sono rappresentati come una sorta di involucro sopra le tabelle.

Grafi: Un grafo è costituito da un insieme di elementi detti nodi o vertici collegati fra di loro, da archi o lati. Più formalmente si definisce grafo una coppia ordinata G = (V, E) di insiemi, in cui V è l'insieme dei nodi ed E l'insieme degli archi tale che gli elementi di E siano coppie di elementi di V. Due vertici del grafo connessi da un arco prendono nome di estremi dell'arco, che a sua volta viene identificato con la coppia formata dai suoi estremi.
In questo *toolkit* i grafi trasformano un insieme di attributi semantici, memorizzati nelle colonne della tabella, in una rappresentazione visuale. Inoltre eseguono operazioni quali il filtraggio, lo zoom, la navigazione e il raggruppamento. Ogni visualizzatore mostra una lista di attributi visuali che possono essere associati con le colonne. Gli attributi visuali standard includono il colore, la dimensione, l'etichetta, la trasparenza e l'ordinamento.

Componenti: i grafi possono essere personalizzati grazie a dei componenti destinati a specifici ruoli di interazione, confondendo quasi il limite tra componenti di visualizzazione delle informazioni e i tradizionali componenti interattivi o *widgets*. In aggiunta a tali componenti, *InfoVis toolkit* fornisce molti altri componenti che consentono di manipolare interattivamente i grafi. Da *default*, ogni grafo è affiancato da un pannello di controllo che ne consente la manipolazione o la configurazione.

2.2.1 R-Graph

Questa tipologia di grafi si basa sul *radial-tree layout* in cui la vista è determinata dalla specifica selezione di un nodo. Lo spostamento da un nodo ad un altro comporta un'animazione che sottolinea e accompagna la transizione. In questo tipo di grafo infatti è consentita la navigazione da parte dell'utente mediante la selezione di nodi che, una volta selezionati, diventeranno il nodo centrale, *focus* dell'intero grafo.
Si tratta di un tipo di grafo orientato. Possiamo distinguere due tipologie di grafi: i grafi orientati e i grafi non orientati. Un grafo è detto orientato se è un insieme di vertici e archi orientati, cioè caratterizzarti da una direzione particolare. Tale direzione solitamente è caratterizzata da un verso indicato con una freccia che rappresenta la “testa”, e una “coda” che rappresenta la parte opposta. Un grafo si dice non orientato se i suoi archi non hanno nessun orientamento e le relazioni tra i nodi sono biunivoche.
Il grafo radiale si basa su rapporti di *parentela* tra gli elementi che si trovano sui propri nodi, stabilendo così delle gerarchie che comprendono dei rapporti di dipendenza del tipo *genitore-figlio,* in cui un elemento è dipendente da un altro.
Tutti i nodi sono distribuiti nel grafo e disposti secondo cerchi concentrici. Al centro si situa il nodo *focus,* cioè quello selezionato, sul primo anello (quello più vicino al centro) si dispongono i *figli* del nodo centrale e negli altri anelli, man mano che ci si allontana dal centro, si dispongono le successive *generazioni* che comprendono i nodi che a loro volta saranno dipendenti dai nodi *figli* del nodo centrale.
La posizione angolare di un nodo su un anello corrispondente è determinata sia dalla vicinanza del suddetto con il nodo centrale, sia dal numero di nodi presenti sull'anello. In linea generale lo spazio presente su un anello viene suddiviso in base al numero dei nodi che vi risiedono creando un angolo identico tra loro e il nodo centrale. I nodi che sono posizionati sugli anelli più esterni, rispetto a quello centrale, invece si predispongono su una precisa porzione dell'anello che corrisponde alla proiezione immaginaria dell'angolo del nodo *genitore* rispetto al nodo centrale.

La posizione angolare di ogni nodo viene calcolata come appena detto, qui viene riportata la funzione che adempie a questo compito:

```
getNodeAndParentAngle: function(id){
    var theta = false;
    var n = this.graph.getNode(id);
    var ps = n.getParents();
    var p = (ps.length > 0)? ps[0] : false;
    if (p) {
      var posParent = p.pos.getc(), posChild = n.pos.getc();
      var newPos = posParent.add(posChild.scale(-1));
      theta = Math.atan2(newPos.y, newPos.x);
      if (theta < 0)
        theta += 2 * Math.PI;
    }
    return {
      parent: p,
      theta: theta
    };
  }
```

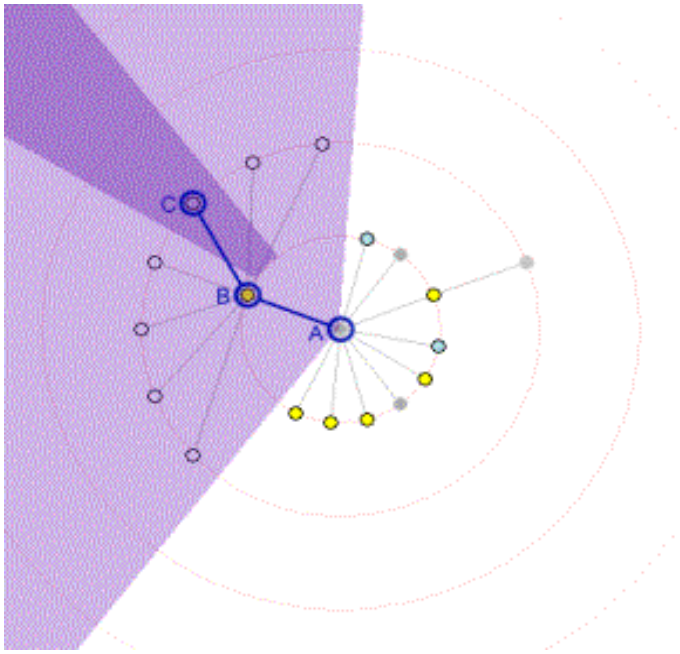

Figura 3: Illustrazione della tecnica di layout radiale.

Per navigare nel grafo l'utente seleziona uno dei nodi visibili che, a quel punto, diventerà il nodo centrale. La nuova vista del grafo che si formerà eseguirà una ricerca sul nuovo *nodo-focus* in modo da poter disporre sia i *figli* che i nodi delle *generazioni* sottostanti alla distanza adeguata alla nuova vista. Il passaggio a questa nuova vista prevede un riordinamento della struttura dell'intero grafo causando perciò confusione e disorientamento nell'utente. Per ridurre questo effetto viene usata un'animazione che prevede una transizione graduale che permette all'utente di individuare facilmente il nuovo nodo centrale e al contempo vedere dove si sposta quello precedente da come si può dedurre dalla Figura 4.

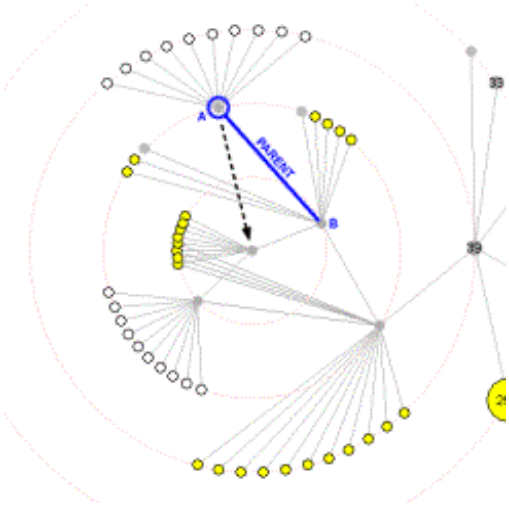

Figura 4: Spostamento dei nodi dopo che varia il nodo centrale.

Dato che i nodi sono disposti radialmente ha più senso interpolare linearmente le coordinate polari dei nodi piuttosto che le coordinate rettangolari. Così che un nodo che rimane su un determinato anello scivola sulla sua circonferenza, mentre un nodo che cambia anello si sposta in maniera uniforme con un movimento a spirale da un anello all'altro. Un'animazione del genere, in cui i nodi ruotano attorno al centro, risulta essere molto fluida. Il raggruppamento dei nodi in questo modo riduce drasticamente lo sforzo cognitivo per la comprensione dell'animazione.

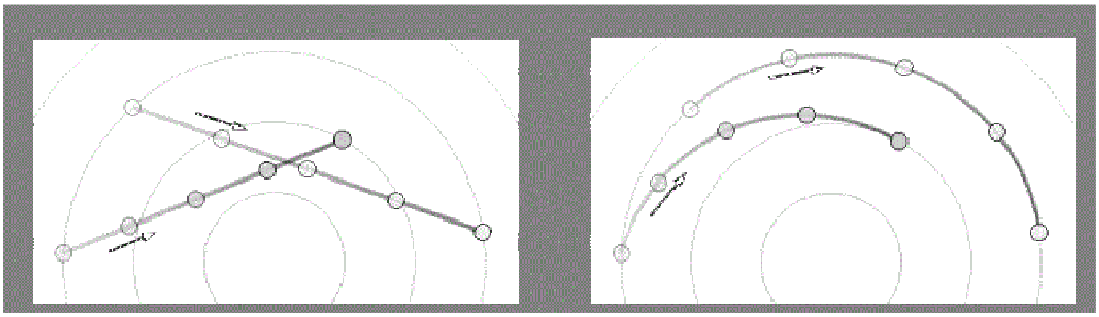

Figura 5: Interpolazione in coordinate rettangolari (sinistra) e interpolazione in coordinate polari (destra).

Per assicurare una costanza visiva, in termini di velocità di esecuzione, sia all'inizio che alla fine dell'animazione, viene preferita un'esecuzione che inizialmente ha una velocità bassa per poi delicatamente accelerare e infine decelerare prima della sua conclusione (*slow-in, slow-out*). Questo metodo aiuta l'utente a prevedere i movimenti del grafo e a proiettarsi in anticipo alla nuova visione che dopo pochi istanti gli si presenterà. Questa animazione è implementata utilizzando un segmento della funzione arcotangente come possiamo vedere nelle Figura 6.

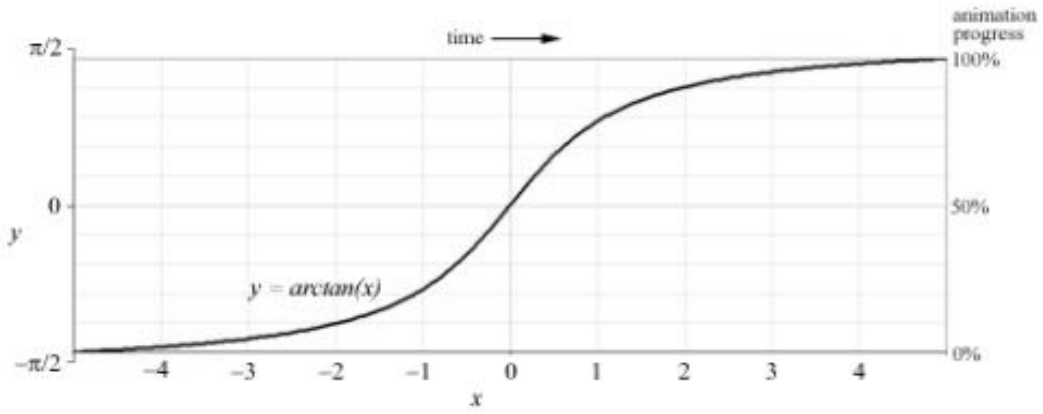

Figura 6: Animazione temporale *slow-in, slow-out*.

La struttura propria del grafo è contenuta in un file chiamato *rgraph.js* ed è fornito dal creatore della libreria.

R-Graph si basa su una struttura *JSON (JavaScript Object Notation)* che si presenta come un *array* di nodi, ognuno dei quali ha delle proprietà:

- **id**, un identificatore univoco del nodo.
- **name**, il nome del nodo.
- **data**, contiene una serie di informazioni accessorie del nodo o dell'arco.
- **adjacencies**, attributo utilizzato per espandere il grafo a più generazioni, consente anche di aggiungere ulteriori proprietà quali ad esempio *nodeFrom* e *nodeTo* che permettono di definire in maniera gerarchica le dipendenze tra nodi.

Qui in basso viene riportato sinteticamente un esempio della struttura di base contenuta nel file:

```
var json = [{
      “id”:  “0001”,
      “name”:  “node1”,
      “data”: {....},
      "adjacencies":  [{
            "nodeFrom": "node1",
            "nodeTo": "node2",
            "data": {
                "labelid":  "....",
                "labeltext":  "...."}
      },{
           "nodeFrom": "node1",
           "nodeTo": "node3",
           "data": {
                "labelid":  "....",
                "labeltext":  "...."}
      }];
```

R-Graph è comprensivo anche di un apposito file in CSS da cui sarà possibile personalizzare tutti gli aspetti grafici.

3 – Visualizzatore di Spazi Semantici

3.1 Introduzione

In questo capitolo saranno descritte nel dettaglio le fasi salienti che hanno portato alla realizzazione del Visualizzatore di Spazi Semantici. A partire dalla creazione del database e delle operazioni preliminari che ha comportato per la sua ottimizzazione, fino alla creazione delle *query* e all'integrazione del database con *R-Graph* che ha reso possibile una lettura intuitiva e dinamica dei dati.
Per il raggiungimento dell'obbiettivo è stato deciso di ricorrere ad uno dei più comuni linguaggi di programmazione attualmente in uso, il PHP (**P**HP: ***H****ypertext* ***P****reprocessor,* originariamente acronimo di ***P****ersonal* ***H****ome* ***P****age*). Il PHP viene originariamente concepito per la programmazione Web, ovvero per la realizzazione di pagine dinamiche. Attualmente è usato principalmente per sviluppare applicazioni web lato *server*.

3.2 Il Database

La base di partenza per la realizzazione di questo lavoro consiste nel tensore di *Distributional Memory* nel modello *Type*DM[29], creato da Alessandro Lenci e Marco Baroni che rappresenta i vicini sintagmatici. Si tratta di un tensore di dimensioni significative (circa 4 Gb) contenente, in totale, più di 130 milioni di tuple.
Per raggiungere gli scopi prefissati e per una facile gestione, il tensore è stato trasformato in un database chiamato *tensor*.

29 http://clic.cimec.unitn.it/dm/

Ogni record del database corrisponde ad una tupla costituita a sua volta da due *unità lessicali* (nomi, verbi o aggettivi; in ogni tupla una delle due unità lessicali costituisce la parola *target* e l'altra la parola di contesto ad essa collegata) legate tra di loro tramite un elemento grammaticale o logico chiamato in questo caso *link (*preposizioni, verbi transitivi o ruoli semantici che dichiarano il ruolo delle due unità lessicali nel contesto in esame) e infine uno *score* che quantifica il peso statistico della tupla, calcolato tramite la LMI. Nel database sono presenti altre informazioni, come la categoria grammaticale delle unità lessicali, che però in questa sede risultano marginali.

La grande quantità di dati ha reso necessaria un'ottimizzazione del database in quanto rallentava parecchio l'accessibilità dei dati da parte dell'utente. Per questo motivo si è optato per l'eliminazione dei dati che, pur essendo importanti negli studi di *Distributional Memory*, erano superflui per questo lavoro. L'ottimizzazione del database non ha previsto l'eliminazione di tutti i record identificati come inversi, cioè tutti quelli della forma *unità lessicale – link-1 - unità lessicale*, in quanto, anche se si tratta di dati ripetuti nel database, sono importanti in quanto ci danno l'informazione statistica per quelle parole che sono sia parole *target* che parole che si trovano nella finestra di contesto di altre parole *target*, inoltre, tra i record restanti sono stati mantenuti solamente quelli che presentavano uno *score* più elevato, più precisamente i primi 15 risultati.

Tramite un software di gestione dei database, *DbVisualizer*[30], è stata applicata al database una funzione *MySql* che ha permesso la sua ottimizzazione e che ha portato come risultato finale un database di gran lunga inferiore a quello di partenza.

30 http://www.dbvis.com/

Questo risultato è stato ottenuto grazie alla seguente funzione *MySql*:

```
SELECT *
FROM(SELECT @row_num :=
      IF (@prev_value = word1, @row_num +1, 1) AS
      RowNumber,word1,link,word2,score,@prev_value := word1
      FROM tensor,
      (SELECT @row_num := 1) x,
      (SELECT @prev_value := '') y
      ORDER BY word1, score DESC)
subquery WHERE RowNumber < 16
```

Come vediamo dalla funzione è stato creato un nuovo campo nel database (*RowNumber*) che effettua il conteggio delle righe per ogni parola contenuta nel campo *word1*, una *subquery* infine ci restituisce solo i primi 15 risultati per ogni parola ordinati in maniera decrescente secondo lo *score*, ovvero il peso.

Questa operazione ha reso il database molto leggero, di facile gestione e manipolazione.

Il database è stato gestito, in locale, tramite il *tool open source* PhpMyAdmin[31].

1	law	obj	enforce	2837.0618
2	law	obj	violate	2627.0473
3	law	obj	break	2342.4746
4	law	with	comply	2095.7753
5	law	sbj_intr	apply	2088.5515
6	law	obj	pass	2068.6401
7	law	prd	become	2037.6318
8	law	obj	obey	1972.0006
9	law	obj	enact	1861.5150
10	law	sbj_intr	require	1561.5097
11	law	obj	apply	1426.0308
12	law	sbj_tr	prohibit	1373.7998
13	law	obj	repeal	1265.7289
14	law	obj	change	1202.4214
15	law	sbj_intr	allow	1113.9490

Figura 7: Veduta di una porzione del database con le tuple ordinate.

31 http://www.phpmyadmin.net/home_page/

3.2.1 Query al database

Ciò che permette all'utente di interagire con il database e le informazioni in esso contenute sono le *query* che vengono richiamate ogni volta che viene ricercata una parola. Se infatti viene inserita una parola nell'apposita *form*, e la parola è presente nel database, la seguente *query* in *MySql*, dove inseriamo anche una variabile in PHP (che contiene la parola cercata), ci restituirà i vicini sintagmatici di quella parola.

```
$query ='SELECT * FROM tensor WHERE word1 LIKE "'.$_GET['parola'].'"
         ORDER BY score DESC LIMIT 15';
```

Nella *query* possiamo notare come vengono selezionate tutte le entrate del database *tensor* in cui la parola cercata corrisponde a una parola presente nel campo *word1*, ordinate in maniera decrescente secondo il loro coefficiente di similarità (*score*), viene impostato inoltre un limite di 15 risultati.
Si tratta di una *query* iterativa che viene richiamata più volte in quanto viene estesa anche ai *figli* della parola cercata.

```
$query_children ='SELECT * FROM tensor WHERE word1 LIKE "'
                 .$record['word2'].'"ORDER BY score DESC LIMIT 15';
```

Questa *query*, come detto, opera come la precedente ma la parola di cui dovrà restituire le dipendenze riguarda i risultati della precedente ricerca.
Grazie alla funzione di *OnClick* sui nodi, una volta cliccato uno dei nodi del grafo, dopo che brevemente sarà eseguita l'animazione di centramento del nodo selezionato, si verrà reindirizzati sulla pagina stessa e verrà avviata automaticamente la *query* che effettuerà la ricerca a partire dalla parola espressa dal nodo. Ciò che dovrà essere ricercato, questa volta, è ciò che è stato precedentemente trovato nel campo *word2*.

3.3 Struttura del Visualizzatore

La struttura del Visualizzatore che è stato creato ha come base di partenza *R-Graph* della libreria *JavaScript Infovis Toolkit*. L'impostazione base del grafo ha comportato una serie di modifiche ed implementazioni che hanno reso possibile una maggiore resa intuitiva per la tipologia di dati che sono stati esaminati.
Oltre ai file *jit.js* e *rgraph.js*, propri della libreria, è stato creato anche un file in PHP in cui è stata implementata la struttura del grafo. E' su questo file che risiede la *query* precedentemente descritta che consente di popolare gli *array* che sono alla base della struttura del grafo.
Grazie all'uso di variabili, i record estratti dal database sono stati riorganizzati in *array* tramite delle assegnazioni.

```
$data[] = array('name' => $_GET['parola'], 'data' => $data_relation,
          'adjacencies' => $data_label);
```

L'*array data* è un *array* di *array* che contiene informazioni, derivate dai record precedentemente estratti, che sono i dati che verranno inseriti nella *var json*, presente su *rgraph.js*. *Name*, *data* e *adjacencies* infatti non sono altro che gli elementi strutturali del grafo. A *name* viene assegnato semplicemente il valore della stringa che abbiamo inserito per fare la ricerca; in *data* ci saranno le informazioni riguardanti le occorrenze della parola cercata, che verranno visualizzate fuori dal grafo; in *adjacencies* infine si trovano le relazioni che si hanno tra i nodi e le etichette degli archi che li collegano, le quali sono contenute nel campo *link* del database.
Queste informazioni sono codificate dal file PHP al file *.js* tramite la funzione **json_encode** che restituisce le stringhe, a partire dall'*array*, sotto forma di notazione JSON.

```
var json = <?php echo json_encode($data); ?>;
```

Il file *rgraph.js* oltre a contenere la struttura dei dati permette anche di effettuare delle modifiche riguardanti la navigazione e l'aspetto di *R-Graph*. Sono stati

modificati infatti, tra le altre cose, il valore iniziale e la velocità dello zoom e le proprietà grafiche degli archi.

Uno degli aspetti più interessanti che sono stati affrontati è stata la creazione delle etichette sui rami del grafo. Si tratta di un'implementazione inizialmente non presente su *R-Graph*, e perciò è stata creata a partire da zero.

Per adempiere a questo compito è stato necessario modificare il file *jit.js.*

```
$jit.RGraph.Plot.EdgeTypes.implement({
        'label_arrow_line': {
            'render': function(adj, canvas) {
              this.edgeTypes.line.render.call(this, adj, canvas);
              var pos = adj.nodeFrom.pos.getc(true);
              var posChild = adj.nodeTo.pos.getc(true);
              var data = adj.data;
                  if(data.labelid && data.labeltext) {
              var domlabel = document.getElementById(data.labelid);
              var radius = this.viz.canvas.getSize();
              var x = parseInt((pos.x + posChild.x -
                     (data.labeltext.length * 5)) / 2);
               var y = parseInt((pos.y + posChild.y ) /2);
               var ctx = this.viz.canvas.getCtx();
               ctx.font="14px verdana";
               ctx.fillStyle ="#FFFFFF";
               ctx.fillText(data.labeltext, x, y);
                }
             }
             }
            }
```

Usando uno dei metodi previsti dalla libreria (Plot.EdgeTypes.implement) è stata creata una nuova tipologia di archi (*edge*) chiamata *label_arrow_line.* Così facendo vengono disegnati gli archi con questo nuovo metodo e gli vengono assegnate delle coordinate cartesiane in base alla posizione dei nodi che collegano.

Costruiti gli archi, se presenti le etichette in *data (*gli elementi *labelid* e *labeltext*), queste verranno ridimensionate e posizionate secondo coordinate spaziali a partire dall'arco corrispondente. Il colore e lo stile di queste etichette è indipendente da quello degli archi.

Molti dei record del database presentavano per i campi *word1* e *word2* i medesimi valori e si distinguevano solo per il valore dell’etichetta (*link*). Questo comportava medesime entrate nella struttura del grafo in nodeFrom e nodeTo così che veniva riconosciuto un solo valore tra queste occorrenze e non venivano visualizzati i restanti. Per ovviare a questo problema è stato associato ad ogni corrispondenza

trovata, un valore random, non visibile nel grafo, che ci permette di disambiguare questi record così da poterli visualizzare contemporaneamente. Si tratta di un metodo *javascript* in cui viene stabilito un intervallo numerico da cui verranno casualmente estratti i numeri da assegnare. In questo caso l'intervallo scelto è stato 0-9.

```
chr(rand(0,9));
```

3.4 Visualizzazione dei Dati

Accedendo alla pagina del Visualizzatore di Spazi Semantici, nella sezione riguardante i vicini sintagmatici (*Syntagmatic Neighbours*), l'utente visualizza in un *box* laterale le istruzioni preliminari per utilizzo e una chiave interpretativa dei dati che verranno visualizzati in seguito alle ricerche eseguite.
Utilizzando un'apposita *form*, posta in alto a destra, è possibile, a partire da una parola di riferimento, ricavarne i 15 vicini sintagmatici.

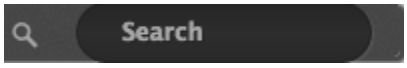

Figura 8: *form* da cui si avvia la ricerca.

Una volta avviata la ricerca i risultati sono proiettati sul grafo, posti in maniera radiale attorno alla parola cercata. Ognuno di essi è rappresentato da un nodo collegato tramite un arco etichettato alla parola scelta per la ricerca. A partire dalla visualizzazione ottenuta, l'utente ha la facoltà di *navigare* il grafo con le operazioni di *panning* e *zooming*, inoltre cliccando su uno dei nodi, dopo l'animazione di centramento del nodo selezionato, viene avviata una ricerca automatica della parola presente sul nodo in maniera analoga a quanto descritto precedentemente. Sugli archi sono presenti le *etichette* che chiariranno all'utente il tipo di relazione che si ha tra le due parole poste sui nodi collegati.

In un *box* posto sopra il grafo si ha una stringa di testo che descrive le operazioni svolte dal grafo mentre processa i dati e presenta le scritte *Ready* (se è possibile procedere con la ricerca), *Centering* seguito dal nome del nodo che sta per essere centrato (compare durante l'animazione di centramento) e *Done* ad operazione finita.
In un *box* laterale, nella posizione in cui prima venivano date le istruzioni preliminari per l'utilizzo del Visualizzatore, in forma tabulare sono disposte tutte le corrispondenze trovate nel database per la parola cercata, analogamente a quello che è visualizzato nel grafo. Queste informazioni ci danno una più completa, anche se meno intuitiva, visione della tupla di rifermento; sono visualizzati infatti, nell'ordine, il vicino sintagmatico della parola cercata, il *link* e lo *score.* Tramite questa doppia vista grafo-box l'utente può interpretare in maniera molto intuitiva le informazioni appena cercate.

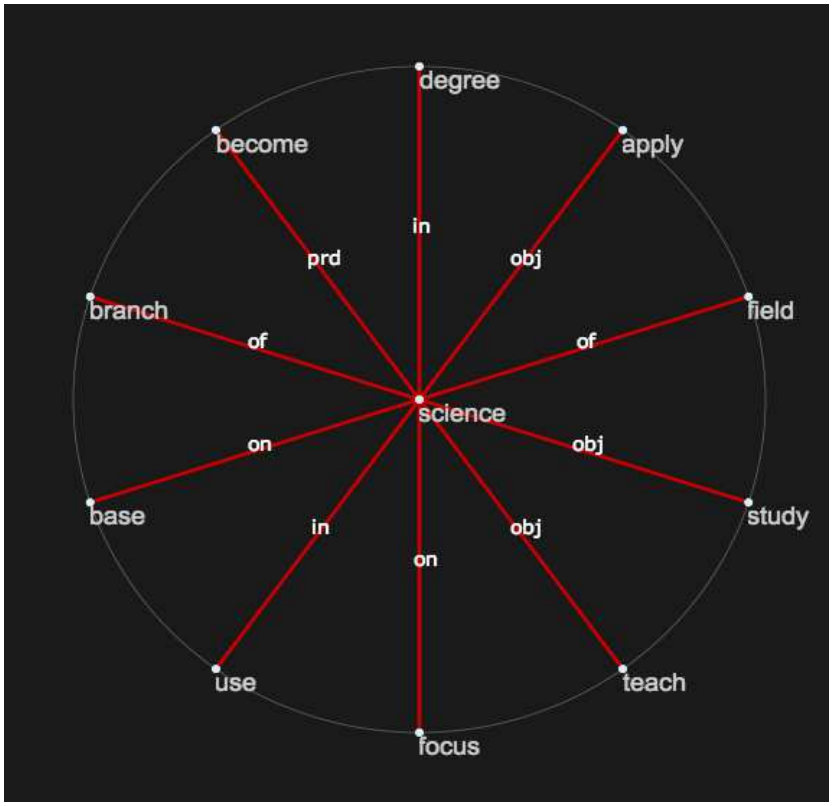

Figura 9: Tipica vista del Visualizzatore dopo una ricerca.

Conclusioni

Con questo lavoro si è tentato di dare una facile lettura e interpretazione alla grande quantità di dati presenti nei database creati a partire dal modello *Type*DM di *Distributional Memory*. Gli obbiettivi del progetto sono stati raggiunti grazie all'utilizzo del grafo radiale *R-Graph* della libreria *Javascript Infovis Toolkit* che ha permesso un tipo di visualizzazione pratica ed intuitiva dei dati a disposizione. Il Visualizzatore infatti soddisfa a tutti i criteri di usabilità.

Le difficoltà maggiori per la realizzazione di questo progetto sono state riscontrate nella modellazione della libreria Javascript per raggiungere lo scopo prefissato, e nella gestione della grande quantità di dati in possesso.

Sicuramente il grafo che ne è risultato potrebbe essere ulteriormente migliorato con l'implementazione di funzioni che consentano di caricare contemporaneamente una maggiore quantità di dati, e perciò un'espansione a più *generazioni*, che permetterebbe una navigazione più omogenea da parte dell'utente, potendo ad esempio *conservare* un percorso di ricerca evidenziando gli archi che sono stati selezionati a partire dal nodo centrale.

L'ottimizzazione del database effettuata ha comportato l'eliminazione di molti dati, in quanto elaborare 4 GB di dati avrebbe rallentato le operazioni di ricerca e la visualizzazione dei dati sul Visualizzatore avrebbe comportato tempi di attesa superiori al minuto. La totale disponibilità dei dati uniti ad un differente metodo di visualizzazione avrebbe potuto dare una visione più completa ed esaustiva con la possibilità, ad esempio, di richiamare specifiche corrispondenze parola-parola.

Bibliografia

Baroni, M. & Lenci, A. (2010), "*Distributional Memory: A general framework for corpus-based semantics*". Journal of Computational Linguistics, Volume 36 Issue 4, Pages 673-721.

Baroni, M., & Lenci, A. (2009), "*One semantic memory, many semantic tasks"*. Proceedings of the EACL *Workshop on Geometrical Models of Natural Language Semantics*, Athens, 31st March.

Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., & Harshman, R.A. (1990), "*Indexing by latent semantic analysis*". Journal of the American Society for Information Science (JASIS), 41 (6), 391-407.

Feteke, J.D. (2004), "*The InfoVis Toolkit*". Université Paris-Sud.

Firth, J. R. (1957), "*A synopsis of linguistic theory 1930-1955*". In *Studies in Linguistic Analysis*, pp. 1-32. Blackwell, Oxford.

Harris Zellig S. (1954), "*Distributional structure*". 146-62 [reprinted in Harris Zellig S. (1970), "*Papers in Structural and Transformational Linguistics"*. Dordrecht: Reidel. 775-794].

Lenci, A. (2008), "*Distributional semantics in linguistic and cognitive research*", in A. Lenci (a cura di), *From context to meaning: distributional models of the lexicon in linguistics and cognitive science*, numero speciale dell'*Italian Journal of Linguistics*, XX/1:1-31.

Lenci, A. (2009), "*Spazi di Parole. Metafore e Rappresentazioni Semantiche*". *Paradigmi*, 27:83-100.

Levin, B. (1993), "*English Verb Classes and Alternations. A Preliminary Investigation"*. Chicago: University of Chicago Press.

Lin, D., & Pantel, P. (2001), "*DIRT – discover of inference rules from text*". In Proceedings of ACM SIGKDD Conference on *Knowledge Discovery and Data Mining* 2001, pp. 323-328.

Lowe, W. (2001), "*Towards a theory of semantic space*". In *Proceedings of the Twenty-rst Annual Conference of the Cognitive Science Society*, pp. 576-581.

Padó, S., & Lapata , M. (2007), "*Dependency-based construction of semantic space models*". *Computational Linguistics* XXXIII/2. 161-199.

Ruiz-Casado, M., Alfonseca, E., & Castells, P. (2005), "*Using context-window overlapping in synonym discovery and ontology extension"*. Department of Computer Science Universidad Autonoma de Madrid.

Sahlgren, M. (2006), "*The Word-space model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in highdimensional vector spaces*". Ph.D. Dissertation, Department of Linguistics, Stockholm University.

Salton, G., Wong, A., & Yang, C. (1975), "*A vector space model for automatic indexing*". *Communications of the ACM*, 18 (11), 613-620.

Sparck Jones, K. (1972), "*A statistical interpretation of term specicity and its application in retrieval*". *Journal of Documentation*, 28 (1), 11-21.

Spearman, C. (1904), "*General intelligence, objectively determined and measured*". *American Journal of Psychology*, 15, 201-293.

Turney, P. D., Littman, M. L., Bigham, J., & Shnayder, V. (2003), "C*ombining independent modules to solve multiple-choice synonym and analogy problems*". In *Proceedings of the International Conference on Recent Advances in Natural Language Processing* (RANLP-03), pp. 482-489, Borovets, Bulgaria.

Turney, P. D. (2006), "*Similarity of semantic relations*". *Computational Linguistics*, 32(3): 379–416.

Turney, P.D., & Pantel, P. (2010), "*From Frequency to Meaning: Vector Space Models of Semantics*". *Journal of Artificial Intelligence Research* 37, 141-188.

Yee, K., Fisher, D., Dahmija, R., Hearst, M., (2001), "*Animated Exploration of Dynamic Graphs with Radial Layout*". University of California, Berkley.

Ringraziamenti

A conclusione di questo lavoro di tesi non posso non ringraziare chi mi ha sostenuto durante tutto questo percorso. Un ringraziamento particolare va alla mia famiglia che sempre mi ha sostenuto sia da un punto di vista economico che morale. Ringrazio anche i colleghi di corso che in questi anni hanno contribuito alla mia crescita professionale e che mi hanno sempre sostenuto, in particolare Fausto, Francesco, Giancarlo, Tommaso e Christian.