UNIVERSITÀ DI PISA

Corso di Laurea in Informatica Umanistica

RELAZIONE

Web-application per la costruzione di spazi semantici distribuzionali

Candidato: *Francesco Asaro*

Relatore: *Alessandro Lenci*

Correlatore: *Maria Simi*

Anno Accademico 2012-2013

INDICE

INTRODUZIONE

Questa relazione si prefigge l'obiettivo di illustrare la realizzazione di un sito web che mette a disposizione degli utenti uno strumento con il quale è possibile creare lo spazio semantico distribuzionale generato da un file contenente un corpus di testo e un file contenente la lista delle parole target oggetto dell'indagine da svolgere. Tali file saranno caricati dall'utente attraverso gli appositi form di upload. Il risultato finale sarà costituito da un ulteriore file di testo al cui interno le co-occorrenze, selezionate in base alla dimensione della finestra di contesto che l'utente avrà immesso, saranno seguite da un valore che indica la corrispondente distanza semantica. Lo strumento fornisce inoltre la possibilità di ricercare una determinata parola all'interno dello spazio semantico generato, in modo da conoscere le cinque parole col più alto grado di similarità semantica rispetto alla parola cercata.

La relazione vuole inoltre fornire al lettore le conoscenze necessarie alla comprensione dello strumento. Pertanto i primi due capitoli sono dedicati, rispettivamente, ad un'introduzione al settore della semantica distribuzionale e ad un'introduzione agli spazi semantici vettoriali. Nel terzo capitolo saranno approfondite le risorse messe a disposizione dell'autore per la realizzazione dell'applicazione. La prima parte del quarto capitolo sarà dedicata all'analisi dei procedimenti che hanno consentito la preparazione delle risorse di base usate come modelli di riferimento ed infine la seconda parte sarà dedicata alla descrizione dello sviluppo dell'applicazione.

1. SEMANTICA DISTRIBUZIONALE

1.1 INTRODUZIONE

La semantica - intesa in questa sede nella sua accezione più generale di branca della linguistica che si occupa dello studio del significato delle parole – è stata fin dalle sue origini al centro di molteplici interessi per il suo intrinseco carattere interdisciplinare ed è stata quindi investigata da scienze quali la filosofia del linguaggio, la psicologia e la logica. Nei suoi più recenti sviluppi essa ha avuto modo di trarre vantaggio dalle più recenti tecnologie informatiche e dalla sempre maggiore disponibilità di grandi collezioni di testi selezionati. Se, da un lato, lo sfruttamento della potenza di calcolo dei computer ha consentito la rapida risoluzione di numerosi problemi che l'uomo e le sue sole forze non avrebbe potuto affrontare, dall'altro ha fatto emergere un'importante problema: i calcolatori, al giorni d'oggi, comprendono solamente una minima parte del linguaggio umano.
Tuttavia il contatto tra semantica e informatica ha spianato la strada a nuovi metodi di analisi del linguaggio naturale e alla nascita di nuovi approcci per portare avanti tali studi.

1.2 DEFINIZIONE

Alla base della semantica distribuzionale risiede ciò che è definita come *Ipotesi Distribuzionale* (Harris, 1954), la quale afferma che:

Il grado di similarità semantica tra due espressioni linguistiche A e B è una funzione della similarità dei contesti linguistici nei quali A e B possono comparire (Lenci, 2008).

In altri termini, parole che ricorrono in contesti simili tendono a possedere simili significati. Tutto ciò nell'assunzione generale che schemi statistici dell'utilizzo di una parola possono essere utilizzati per

comprendere il significato che un'espressione linguistica intende esprimere (Weaver, 1955; Furnas et al. 1983). A prescindere dalle numerose definizioni create dagli studiosi nel corso degli anni, l'idea generale è quindi quella che esiste una correlazione tra la similarità distribuzionale e la similarità semantica tra due o più espressioni linguistiche.

1.3 CENNI STORICI

Recentemente la semantica distribuzionale ha riacquistato una popolarità che negli anni ha spesso vacillato. In questa sezione del capitolo è mia intenzione ripercorrere i momenti salienti che ne hanno caratterizzato la storia.

In verità potrà sorprendere il lettore scoprire che l'origine di ciò che il linguista americano Zellig Harris chiamava Ipotesi Distribuzionale non è da ricercare all'interno del dominio della semantica ma piuttosto in quello della fonologia. Difatti l'approccio distribuzionale è stato inizialmente introdotto nell'analisi dei fonemi e generalizzato solo in un secondo momento al fine di poter essere destinato ad ogni analisi linguistica. L'ID assiste il linguista nella fase preliminare dell'analisi fonologica e morfologica contribuendo a risolvere la prima grande questione che egli si trova ad affrontare: estrarre gli elementi rilevanti ai fini della sua indagine. Per essere tali, gli elementi devono essere estratti su base distribuzionale:

x e y sono inclusi nello stesso elemento A se la distribuzione di x relativa ad altri elementi B, C, ecc. è in qualche modo uguale alla distribuzione di y (Harris, 1951).

Quest'operazione può essere considerata efficace solo se eseguita su tutti gli elementi contemporaneamente e pertanto si deduce che se due elementi A e B hanno significati diversi sarà possibile individuare contesti linguistici nei quali uno ricorre e l'altro no.

L’approccio statistico-distribuzionale non ha, però, sempre goduto di ottima reputazione subendo dapprima la crescita di interesse nei confronti della linguistica generativa di Noam Chomsky, il quale rovesciava le tesi della psicologia comportamentista sostenendo che il linguaggio è una capacità innata dell’essere umano e che il suo funzionamento è da spiegare nell’ambito dei principi cognitivi che risiedono alla base della mente umana. Inoltre le distribuzioni statistiche basate sui corpora sono state completamente dismesse come fonte affidabile di prova linguistica, insieme con il rifiuto della probabilità come modello formale per la descrizione della grammatica (Lenci, 2008). Il filosofo David Lewis non escludeva che l’analisi distribuzionale ci fornisse dati di reale interesse sul linguaggio ma sosteneva che lo stesso non può essere detto riguardo al significato, perché le grammatiche sono sistemi semantici astratti per mezzo dei quali i simboli sono associati ad aspetti del mondo (Lewis, 1972). Dal punto di vista della linguistica teorica il significato di un’espressione linguistica non può essere definito in base a come essa si distribuisce statisticamente in un testo ma deve necessariamente essere legato ad entità extra-linguistiche.

È nella linguistica dei corpora che l’ID ritorna a ricoprire un ruolo centrale ed è anzi considerato l’unico approccio in grado di fornire prove concrete per l’esplorazione del significato. Anche la lessicografia si dimostra favorevole all’utilizzo di evidenze statistiche estratte da grandi quantità di testo, ma probabilmente è nella psicologia che l’approccio distribuzionale trova maggiori consensi. D’altronde è di facile intuizione comprendere che ciò che la mente umana conosce di una parola non è la sua definizione enciclopedica ma piuttosto il modo in cui essa viene utilizzata all’interno di molteplici contesti. Gli esseri umani non apprendono l’utilizzo di una parola grazie alla lettura della rispettiva definizione in un vocabolario ma, più facilmente, mediante l’osservazione di come questa viene usata in situazioni contingenti (Miller & Charles, 1991).

1.4 SIMILARITÀ SEMANTICA

Sebbene il metodo distribuzionale sia in grado di rilevare la similarità semantica di differenti espressioni linguistiche, il suo limite consiste nell'impossibilità di distinguere le diverse tipologie di relazioni inferenziali. L'ID si basa, infatti, sul misurare la distanza semantica che intercorre tra due elementi all'interno di uno spazio distribuzionale e se ciò può ritenersi sufficiente per individuare casi di sinonimia (in quanto essa costituisce una relazione simmetrica tra due elementi), altrettanto non può essere detto per quanto concerne gli altri tipi di rapporti inferenziali.

Questi sviluppi sono stati registrati e analizzati da Peter D. Turney il quale ritiene, infatti, necessario operare una distinzione tra *similarità attribuzionale* e *similarità relazionale*.

L'applicazione del modello distribuzionale riesce nel compito di individuare i sinonimi poiché questi ultimi sono termini che denotano concetti che condividono simili attributi e, di conseguenza, parecchi contesti linguistici. Ad esempio, parole come "*giornale*" e "*rivista*", sono simili dal punto di vista attribuzionale poiché i loro significati condividono un'ampia classe di attributi: entrambi si presentano in forma cartacea, entrambi vengono letti, ecc.

La similarità attribuzionale tra due parole *a* e *b*, $\text{sim}_a(a, b) \in \Re$ dipende dal grado di corrispondenza tra le proprietà di *a* e *b*. Maggiore è la corrispondenza, maggiore sarà la loro similarità attribuzionale (Turney, 2006).

In certi casi la corrispondenza semantica che lega coppie di parole non è da riscontrare sul piano della similarità degli attributi come nel caso della sinonimia o della co-iponimia ma su quello della similarità relazionale come nel caso degli iperonimi e degli iponimi. Ad esempio, coppie di parole come "*frutta*" e "*uva*" sono collegate da un tipo di similarità che non riguarda propriamente gli attributi che queste due parole condividono ma piuttosto il tipo di relazione inferenziale che le unisce. La parola "*frutta*" costituisce, infatti, un iperonimo della parola "*uva*".

La connessione relazionale è utilizzata, in particolar modo, per rilevare la similarità semantica di coppie di parole. Si pensi a coppie del tipo "*contadino : terra*" e "*falegname : legno*", le quali sono legate dalla relazione comune "*lavoratore* : *materiale lavorato*".

In questo caso la similarità tra due coppie di parole $a : b$ e $c : d$, $\text{sim}_r(a{:}b, c{:}d) \in \Re$, dipende dal grado di corrispondenza tra le relazioni di $a : b$ e $c : d$ (Turney, 2006). Maggiore è la corrispondenza, maggiore sarà la loro similarità relazionale.

Un'ulteriore distinzione riguardo i modi in cui le parole si possono distribuire in un testo, è quella operata da Shütze e Pedersen (1993): se due parole tendono a ricorrere l'una accanto all'altra, allora esse sono dette *associati sintagmatici* (es. "*cacciatore*" e "*preda*" o "*aereo*" e "*decollo*") mentre se due parole hanno simili vicini, vengono chiamate *paralleli paradigmatici* (es. "*architetto*" e "*ingegnere*", termini che ricorreranno spesso all'interno di contesti formati dalle medesime parole come ad esempio "*progettare*" o "*edificio*"). Da ciò si deduce che i primi sono spesso diverse parti del discorso mentre i secondi sono solitamente la stessa parte del discorso.

2. SPAZI SEMANTICI VETTORIALI

2.1 INTRODUZIONE

L'enorme mole di informazioni ricavabili da un corpus di testi selezionati necessita di metodi matematici in grado estrarre dati empirici rilevanti. Nel capitolo precedente è stato fatto riferimento all'analisi statistica perché questo sembra essere l'approccio più naturale per questo scopo, ma è necessario introdurre un altro elemento per formalizzare a tutti gli effetti le rappresentazioni contestuali ed evincerne dati tangibili. Dal momento che l'obiettivo è quello di registrare le occorrenze di ogni espressione linguistica nel suo contesto, i vettori si dimostrano essere i metodi matematici più inerenti allo scopo. Ogni vettore è, infatti,

costituito da un'ennupla di numeri ($x_1,\ldots,x_n$). Immaginiamo di poter associare ogni parola ad un vettore e saremo quindi in grado di registrare il punteggio di rilevanza statistica totalizzato da ogni parola in ciascun contesto in cui ricorre. In uno spazio semantico vettoriale ogni parola è rappresentata come un punto in uno spazio, cioè come un vettore in uno spazio vettoriale. In questo spazio, la vicinanza di due punti indica la similarità semantica di due espressioni linguistiche così come la lontananza ne indicherà la distanza semantica. Infine, una volta creato lo spazio vettoriale sarà opportuno organizzare i vettori in una matrice prima di procedere al loro confronto.[1]

2.2 RAPPRESENTAZIONE CONTESTUALE

Finora il concetto di contesto è stato inteso nell'accezione più generale del termine ma è necessario, a questo punto, fornirne una precisa definizione. Come visto in precedenza, l'ipotesi distribuzionale afferma che sia possibile conoscere aspetti del significato di un'espressione linguistica esaminando la rappresentazione distribuzionale derivata dai contesti in cui la suddetta espressione ricorre. Il contesto di una parola target è formato dalle parole che ricorrono insieme ad essa. Si distinguono due modalità con cui è possibile selezionare le co-occorrenze tra il target e le parole facenti parte del relativo contesto. Una delle due modalità prevede la selezione delle co-occorrenze lineari. In altri termini, vengono selezionate le parole che ricorrono con la parola target all'interno di una finestra di parole di dimensione *n*. La parola target può trovarsi al centro della finestra determinandone la simmetria oppure si può decidere di selezionare un numero di parole, precedenti al target, maggiore o inferiore rispetto a quelle successive, determinando quindi una finestra di contesto asimmetrica. L'altra modalità prevede la selezione delle co-occorrenze sintattiche. In questo caso, la parola target e le parole facenti parte del relativo contesto sono collegate da una

[1] In questa sede non è necessario approfondire i concetti dell'algebra lineare ma piuttosto conoscerne i caratteri fondamentali.

relazione sintattica (soggetto, complemento oggetto, verbo, ecc.). Questa seconda modalità di selezione può rilevarsi utile quando, ad esempio, all'interno dello stesso contesto compaiono due o più occorrenze della stessa parola ma con relazioni sintattiche diverse rispetto alla parola target. Fatta questa distinzione, è fondamentale aggiungere che ai fini del progetto saranno selezionate le sole co-occorrenze lineari.

2.3 MATRICI

Alla luce di ciò che è stato detto, dato un vettore i cui elementi siano stati derivati dalle occorrenze di una parola nelle finestre di contesto in cui compare; in una *matrice parola-contesto*, in cui le righe corrisponderanno alle parole e le colonne ai contesti nei quali queste parole ricorrono, simili vettori indicheranno simili significati delle parole alle quali sono associati[2] (Deerwester et al., 1990). Nella seguente tabella di esempio i vettori sono stati inseriti in una matrice i cui valori sono ricavati dal numero di occorrenze nei diversi contesti. Una matrice parola-contesto registra quindi le co-occorrenze tra una parola target (vettore riga) e una parola facente parte della sua finestra di contesto (vettore colonna). Quando le co-occorrenze di questo tipo presentano un elevato potere combinatorio - inteso come un'accentuata regolarità a mostrarsi vicini in un enunciato - vengono chiamate *collocazioni* e l'elemento che co-occorre nel formare una collocazione è definito *collocato*. L'ampiezza della finestra massima di parole che separano il target dal collocato è un parametro che viene fissato a priori.

	accendere	**correre**	**giallo**	**video**
computer	3	0	0	*4*
fiore	0	0	5	*0*
gatto	0	3	0	*2*
lampadina	*4*	*0*	*2*	*0*

[2] Le metodologie di comparazione dei vettori per individuarne la similarità saranno approfondite nel paragrafo successivo.

Finora è stato esaminato il caso in cui il fine della ricerca sia quello di identificare la similarità semantica tra due parole. Ma l'indagine linguistica non si esaurisce soltanto in questo tipo di analisi. In altre circostanze, come spesso accade nel settore dell'*Information Retrieval*, si vorrà ad esempio ricercare la similarità semantica tra interi documenti. In questo caso la *matrice termine-documento* sarà formata da tante righe quanti sono i termini e da tante colonne quanti sono i documenti. Le matrici di questo tipo si fondano su quella che è chiamata *ipotesi del contenitore di parole* (Salton et al., 1975). Un vettore documento conterrà, infatti, un elemento per ogni parola tipo che il documento contiene (non saranno pertanto presenti ripetizioni della medesima parola) e ogni criterio di ordinamento o legame grammaticale sarà trascurato. Il suddetto elemento rappresenterà il numero di occorrenze della parola tipo nel documento. Una matrice di questo tipo può essere rappresentata con la seguente tabella di esempio:

	Documento 1	**Documento 2**	**Documento 3**	**Documento *n***
Parola Tipo 1	0	0	2	*n*
Parola Tipo 2	4	0	0	*n*
Parola Tipo 3	0	3	0	*n*
Parola Tipo *n*	*n*	*n*	*n*	*n*

Si deduce che i vettori parola non sono altro che l'insieme dei vocaboli che formano il lessico dei documenti. Dato che ogni documento conterrà solo una piccola porzione dell'intero vocabolario, il valore della maggior parte degli elementi di una matrice termine-documento sarà 0. Questo tipo di approccio risulta efficace perché è comprensibile che l'argomento trattato in un documento influenzerà necessariamente la scelta dei vocaboli da parte dello scrittore. Di conseguenza i vettori colonna di

documenti riguardanti lo stesso argomento presenteranno simili sequenze numeriche.

Un'ulteriore tipologia di rappresentazione prevede la costruzione di una matrice i cui vettori riga corrispondono a coppie di parole (come *meccanico* : *auto* o *pilota* : *vettura*) mentre i vettori colonna corrispondono a configurazioni (come "X ripara Y" o "X guida Y"). La *matrice coppia-configurazione* è stata introdotta da Lin e Pantel (2001) affinché fosse possibile evidenziare la similarità semantica di configurazioni come "X mangia Y", "Y è mangiato da X" e "X ingerisce Y" e si basa su quella che è definita *ipotesi distribuzionale estesa,* la quale sostiene che configurazioni che ricorrono con simili coppie di parole tendono ad avere simili significati. Questo genere di approccio può essere utilizzato per individuare espressioni che sono parafrasi l'una dell'altra.

Mentre Lin e Pantel si focalizzarono sulla ricerca della similarità nelle relazioni in cui coppie di parole ricorrono, Turney et al. (2003) decisero di utilizzare la stessa classe di matrici per calcolare la similarità semantica delle coppie di parole ricorrenti in simili configurazioni, concentrandosi sulle le righe vettore della matrice coppia-configurazione piuttosto che sulle colonne. L'inverso dell'ipotesi distribuzionale estesa è chiamata *ipotesi di relazione latente*: coppie di parole che ricorrono in simili configurazioni tendono ad avere simili relazioni semantiche (Turney, 2008). In questo modo, binomi come *cestista* : *canestro*, *tennista* : *punto*, *golfista* : *buca*, *calciatore* : *rete*, che condividono la relazione semantica *sportivo* : *obiettivo*, tenderanno a ricorrere in simili configurazioni come "X totalizza Y" oppure "X segna Y". La matrice che riassume entrambe le ipotesi può essere rappresentata in questo modo:

	Configurazione 1	**Configurazione 2**	**Configurazione 3**	**Configurazione *n***
Coppia 1	1	0	0	*n*
Coppia 2	0	2	1	*n*
Coppia 3	2	1	0	*n*
Coppia *n*	*n*	*n*	*n*	*n*

2.4 PROCESSARE IL TESTO

Una volta selezionato il corpus di testo da esaminare, è necessario sottoporre quest'ultimo ad alcune operazioni di analisi linguistica.

La prima di queste operazioni è detta *tokenizzazione*. Il token può essere inteso come l'unità linguistica minima di un corpus. La selezione di ciò che costituisce un token dipende strettamente dal tipo d'indagine che si ha intenzione di svolgere. Nella maggior parte dei casi tokenizzare un testo corrisponde a selezionare le parole utilizzando gli spazi tra di esse come separatori. Nell'atto pratico l'operazione si rivela essere molto più complessa. Il tokenizzatore deve, infatti, essere in grado di gestire la punteggiatura, le parole collegate da trattini (es. *co-amministratore*), gli acronimi (es. *C.N.R.*) le espressioni multi-parola (es. *Stati Uniti d'America*), ecc.. Questa operazione deve essere inoltre differenziata e adattata in base alla lingua del corpus selezionato dato che ogni linguaggio naturale possiede caratteri peculiari.

Nell'ispezionare un testo, saranno molteplici i casi in cui parole costituite da sequenze di caratteri diversi veicolino in realtà lo stesso significato. Per questo motivo la seconda delle operazioni che è opportuno eseguire su di un testo tokenizzato è la *normalizzazione*. Questa può essere effettuata in diversi modi, uno dei più comuni è quello di ridurre tutti i caratteri maiuscoli in minuscoli in modo che token come "Inoltre" e "inoltre" siano fatti risalire alla medesima parola tipo. Ancora una volta l'operazione si rivela più complessa di ciò che potrebbe sembrare. Non sono rari i casi in cui l'uso di caratteri maiuscoli serva, in effetti, a distinguere il significato di due parole come ad esempio accade con i cognomi.

La *lemmatizzazione* è invece quel processo atto a ridurre la forma flessa di una parola alla sua radice o forma base detta per l'appunto *lemma*. Anche quest'ultimo processo è strettamente legato alla lingua del corpus in esame. Per quanto riguarda l'italiano, i verbi sono lemmatizzati, generalmente, al tempo infinito presente mentre i sostantivi al maschile o femminile singolare. Gli aggettivi sono lemmatizzati al maschile singolare nel caso siano a quattro uscite (es. bello, primo) e al maschile o femminile singolare per gli aggettivi a due uscite (es. utile, presente).[3] In linea di massima questo ragionamento può essere applicato a tutte le lingue flessive, come l'italiano, o isolanti, come l'inglese, ma risulta essere di difficile applicazione nelle lingue agglutinanti come il finlandese o il giapponese, nelle quali in una sola parola sono combinati molti significati mediante l'utilizzo di prefissi e suffissi.

Infine, tramite il processo di *annotazione*, i token vengono etichettati secondo la parte del discorso (soggetto, verbo, complemento oggetto, ecc.) che rivestono (annotazione sintattica) oppure in base al significato che esprimono in un determinato contesto (annotazione semantica) o ancora, a seconda del ruolo grammaticale svolto dal token nella struttura della frase (*parsing*).

2.5 ELABORARE LA MATRICE

Così come il testo richiede alcuni accorgimenti prima di essere collocato nella categoria di matrici adatta all'analisi da eseguire, sarà anche opportuno applicare alcune modifiche alla matrice prima di procedere al passaggio successivo. Prendendo in esame la classe di matrici parola-contesto trattata in precedenza, queste sono generate a partire dalla scansione sequenziale di un documento e, allo stesso tempo, dalla registrazione dei valori di co-occorrenza di una parola nelle finestre di contesto in cui ricorre. Malgrado questo primo passaggio possa sembrare privo di difficoltà, giacché trattasi di un mero conteggio, in realtà si complica sensibilmente con l'aumentare delle dimensioni del corpus.

[3] http://it.wikipedia.org/wiki/Lemma_(linguistica)

A causa della distribuzione zipfiana dei dati linguistici, la rappresentazione risultante è una matrice sparsa poiché la maggior parte dei suoi elementi avrà valore 0. Alcuni elementi invece conterranno valori molto alti e tenderanno a monopolizzare l'attenzione del calcolatore durante l'indagine. È lecito aspettarsi che parti del discorso come gli articoli, determinativi o indeterminativi, oppure verbi come "essere" o "avere" compaiano in numero parecchio elevato nella matrice di frequenza generata tramite il conteggio delle occorrenze e quindi tenderanno a primeggiare rispetto ad altri sebbene la loro similarità semantica sia probabilmente di scarso interesse ai fini dell'analisi. L'approccio più semplice sarebbe quello di limitare il numero di questo tipo di elementi procedendo con la rimozione di quelli meno interessanti e considerando solo i vettori che condividono coordinate il cui valore non sia zero. Questo sistema spesso non si dimostra essere sufficiente per distinguere tra gli eventi linguistici di maggiore interesse e quelli che invece possiedono poco potere semantico discriminatorio. Per questo motivo è conveniente attribuire un peso statistico appropriato a determinati elementi in modo da mettere maggiormente in risalto soltanto quelli strettamente rilevanti.

Nell'ambito dell'*information retrieval*, ad esempio, una parola acquisisce un peso statistico più elevato se ricorre con frequenza in un documento ma è molto rara negli altri. In altri termini, il peso di un elemento, che è frequente in un documento, cresce in maniera inversamente proporzionale alla sua frequenza complessiva nel corpus. Questa funzione è definita *tf-idf* (*term frequency–inverse document frequency*) e si esprime con la formula:

$$(tf - idf)_{ij} = tf_{ij} \times idf_i$$

Dove la frequenza di un termine in un documento è data dal rapporto tra il numero di occorrenze del termine nel documento e la dimensione del documento stesso:

$$tf_{ij} = \frac{n_{ij}}{|d_j|}$$

Mentre la rilevanza di un termine in un corpus è ottenuta tramite il logaritmo del rapporto tra il numero di documenti del corpus e il numero di documenti che contengono il termine:

$$idf_i = \text{LOG}\frac{|D|}{|\{d : t_i \in d\}|}$$

Nel caso di una matrice parola-contesto, uno dei metodi più diffusi ed indicati per misurare la rilevanza di associazioni tra due o più termini è la *mutua informazione*, la quale confronta la probabilità di riscontrare nel testo co-occorrenze di termini $<t_i, t_j>$ rispetto alla probabilità di riscontrare i termini $<t_i>$, $<t_j>$ individualmente. La mutua informazione è calcolata nel seguente modo:

$$MI = \text{LOG}_2\frac{p(t_i, t_j)}{p(t_i)p(t_j)} = \text{LOG}_2\frac{\frac{f(< t_i, t_j >)}{N}}{\frac{f(t_i)}{N}\cdot\frac{f(t_j)}{N}}$$

Dove al numeratore è collocata la frequenza relativa delle co-occorrenze dei due termini e al denominatore la frequenza relativa delle singole istanze. Da cui, semplificando, si ottiene:

$$\text{LOG}_2\frac{f(< t_i, t_j >)}{N}\cdot\frac{N^2}{f(t_i)f(t_j)} = \text{LOG}_2\frac{f(< t_i, t_j >)\cdot N}{f(t_i)f(t_j)}$$

In verità, una funzione di questo tipo favorirebbe eccessivamente le coppie rare. Basti immaginare il caso in cui coppie di parole siano costituite da termini registrati soltanto una volta all'interno dell'intero corpus. In questo caso la frequenza, sia quella della coppia sia quella dei singoli costituenti, sarà pari a uno e di conseguenza la mutua informazione sarà sicuramente tra le più alte tra quelle calcolate. In generale, un ragionamento equivalente si applica nel caso in cui coppie di parole ricorrono sempre insieme.

Per questo motivo è stata introdotta la *Local Mutual Information* (informazione mutua locale). Essa consiste nel moltiplicare la mutua informazione per il numero di occorrenze della coppia della quale è stata calcolata. In questo modo sarà attribuito un peso statistico maggiore alle coppie più frequenti. In termini algebrici:

$$LMI = f(< t_i, t_j >) \cdot \mathrm{LOG}_2 \frac{p(t_i,t_j)}{p(t_i)p(t_j)}$$

2.6 ANALISI VETTORIALE

A questo punto ogni valore della matrice conterrà il peso statistico delle co-occorrenze calcolato in modo tale che sia coerente col tipo di ricerca da condurre, pertanto i vettori riga della matrice sono pronti per il confronto. Le tecniche per misurare la similitudine tra due vettori sono molteplici ma quella che probabilmente è la più popolare prevede di misurare il coseno dell'angolo che essi formano. Tuttavia, prima di procedere al calcolo angolare, alcuni approcci considerano opportuno normalizzare la lunghezza dei vettori.

Il versore $\hat{v}$, cioè il vettore di modulo unitario, è ottenuto dividendo ogni dimensione v per la lunghezza originale del vettore $\boldsymbol{v}$. Considerato il vettore $\boldsymbol{v}$ di dimensioni $v_1, v_2, \ldots, v_n$ tale che la sua lunghezza sia espressa come:

$$\|v\| = \sqrt{\sum_{i=1}^{i=n} v_i^2}$$

Il versore $\hat{v}$ si ottiene con la seguente formula:

$$\hat{v} = \frac{v}{\|v\|}$$

Pertanto dati i vettori ***v*** e ***k*** contenenti *n* elementi ciascuno, tali che:

$$v = \langle v_1, v_2, v_3, \dots, v_n \rangle$$

$$k = \langle k_1, k_2, k_3, \dots, k_n \rangle$$

Il calcolo del coseno dell'angolo α formato da ***v*** e ***k*** equivale al prodotto scalare tra i due vettori dopo che questi sono stati normalizzati.

$$\cos \alpha = \frac{\sum_{i=1}^{n} v_i \cdot k_i}{\sqrt{\sum_{i=1}^{n} v_i^2 \cdot \sum_{i=1}^{n} k_i^2}} =$$

$$= \frac{v \cdot k}{\sqrt{v \cdot v} \cdot \sqrt{k \cdot k}} = \frac{v \cdot k}{\sqrt{v^2} \cdot \sqrt{k^2}} = \frac{v \cdot k}{v \cdot k} =$$

$$= \frac{v}{\|v\|} \cdot \frac{k}{\|k\|}$$

Il coseno è una quantità che in trigonometria oscilla tra valori compresi tra -1 e 1 ma nell'analisi linguistica le occorrenze di un termine non

possono presentare valori negativi a meno che questi non vengano introdotti mediante particolari tecniche per la stima del peso statistico. Di conseguenza, nella maggior parte dei casi, il valore risultante dalla suddetta formula sarà compreso tra 0 (quando i vettori sono ortogonali e l'angolo formato da essi è di 90°) e 1 (quando i vettori puntano nella stessa direzione e l'angolo è di 0°). Pertanto valori di coseno prossimi ad 1 indicheranno un elevata similarità semantica.

3. RISORSE

3.1 INTRODUZIONE

Col capitolo precedente si conclude la sezione di questa relazione dedicata a fornire i fondamenti teorici necessari alla comprensione dello strumento realizzato. In questo capitolo sono illustrate le risorse sulle quali l'applicativo è stato sviluppato e valutato e i programmi grazie ai quali la sorgente testuale è stata elaborata affinché l'esecuzione del codice permettesse, in un primo momento, l'estrazione delle co-occorrenze da esaminare, successivamente, la generazione di una matrice parola-contesto i cui valori contenuti all'interno sono stati calcolati in base alla stima del peso statistico di ogni bigramma, e infine la misura della distanza semantica tramite calcolo del coseno.

3.2 CORPUS DI RIFERIMENTO

Tenendo ben a mente che lo strumento è stato realizzato con lo scopo di essere compatibile con qualsiasi tipo di collezione di testi[4], nella fase di implementazione è stato preso a modello il corpus TEMIS (Syntactically

[4] Purché questi presentino un preciso formato di disposizione dei dati che sarà illustrato nel paragrafo successivo.

and Semantically Annotated Italian Legislative Corpus). Il TEMIS è una collezione di testi legislativi emessi da tre diverse istituzioni ed è composto come segue:

- 465.201 tokens (20.222 frasi) di testi legislativi emessi dalla Comunità Europea;
- 114.944 tokens (5.797 frasi) di testi legislativi emessi dalla Regione Piemonte;
- 15.804 tokens (504 frasi) inclusi nel TEMIS corpus.

Per un totale di 595.949 tokens (26.523 frasi).

Il corpus è stato realizzato da Venturi (2011) e in seguito ampliato[5] grazie anche al contributo del sottoscritto nel contesto del tirocinio formativo universitario svolto presso l'Istituto di Linguistica Computazionale del C.N.R. di Pisa, in cui mi sono occupato in particolare dell'annotazione sintattica e semantica di una parte del testo. Il TEMIS corpus presenta al suo interno una composizione molto variegata di documenti legislativi di ogni sorta, quali decreti, leggi regionali e nazionali, direttive europee, circolari ministeriali, ecc. e ricopre un insieme eterogeneo di domini che spaziano dai diritti umani, all'ambiente, alla libertà di espressione. L'annotazione sintattica è stata eseguita in via semi-automatica, ovvero, ad una fase di annotazione automatica del testo attraverso il parser di dipendenze *DeSR*[6], è seguita una fase di revisione manuale effettuata con l'utilizzo dello strumento *DgAnnotator* (Dependency Grammar Annotator)[7] al fine di ridurre al minimo gli errori del parser automatico.

Il formato di codifica morfo-sintattica del TEMIS corpus è stato sviluppato in prima istanza per l'annotazione dell'ISST-TANL, un corpus composto da articoli giornalistici, selezionati con l'obiettivo si comprendere una vasta quantità di tematiche e realizzato grazie al lavoro

[5] Design and Development of TEMIS: a Syntactically and Semantically Annotated Corpus of Italian Legislative Texts. In Proceedings of the 4th Workshop on "Semantic Processing of Legal Texts", held in conjunction with LREC 2012, Istanbul, Turkey, 27th May.

[6] G. Attardi and F. Dell'Orletta. 2009. Reverse revision and linear tree combination for dependency parsing. In Proceedings of NAACL-HLT.

[7] http://medialab.di.unipi.it/Project/QA/Parser/DgAnnotator/

sinergico dell'Istituto di Linguistica Computazionale "Antonio Zampolli" (ILC–CNR) e del Dipartimento di Informatica dell'Università di Pisa (Montemagni e Simi, 2007) nel quadro del progetto TANL (*Text Analytics and Natural Language processing*). Tale formato di codifica è basato sul tagset ILC/PAROLE, il quale aderisce allo standard internazionale EAGLES. Esso consta di 14 tag per l'etichettatura delle parti del discorso generiche (*coarse-grained*) e 37 tag per l'etichettatura di quelle specifiche (*fine-grained*) e di 6 tratti morfologici che variano in base al token a cui fanno riferimento (gen, num, mod, per, ten, sup) [8].

Il formato di annotazione delle dipendenze segue invece lo standard tabulare CoNLL-2007 usato nel contesto dello "*Shared Task on Dependency Parsing*" (Nivre et al., 2007) e prevede che i token siano disposti uno per riga e ognuno di essi arricchito con informazioni concernenti il lemma corrispondente, la parte del discorso generica e specifica, le caratteristiche morfologiche, la testa della relazione di dipendenza e il tipo di relazione. Segue un esempio di annotazione nel formato tabulare CoNLL-2007:

1	Il	il	R	RD	num=s|gen=m	2	det	_	_
2	periodo	periodo	S	S	num=s|gen=m	15	subj_pass	_	_
3	di	di	E	E	_	5	comp	_	_
4	cui	cui	P	PR	num=n|gen=n	3	prep	_	_
5	all'	al	E	EA	num=s|gen=n	2	mod_rel	_	_
6	articolo	articolo	S	S	num=s|gen=m	5	prep	_	_
7	5	5	N	N	_	6	mod	_	_
8	,	,	F	FF	_	9	punc	_	_
9	paragrafo	paragrafo	S	S	num=s|gen=m	6	mod	_	_
10	6	6	N	N	_	9	mod	_	_
11	della	di	E	EA	num=s|gen=f	9	comp	_	_
12	decisione	decisione	S	S	num=s|gen=f	11	prep	_	_
13	1999/468/CE	1999/468/CE	A	A	num=s|gen=n	12	mod	_	_
14	è	essere	V	VA	num=s|per=3|mod=i|ten=p	15	aux	_	_
15	fissato	fissare	V	V	num=s|mod=p|gen=m	0	ROOT	_	_
16	a	a	E	E	_	15	comp	_	_
17	tre	tre	N	N	_	18	mod	_	_
18	mesi	mese	S	S	num=p|gen=m	16	prep	_	_
19	.	.	F	FS	_	15	punc	_	_

[8] http://poesix1.ilc.cnr.it/ISST-TANL-MStagset-web.pdf

3.3 SCRIPT

Affinché fosse possibile conseguire il risultato desiderato, sono stati utilizzati cinque script contenenti codice scritto nel linguaggio di programmazione PERL[9]. Tali script consistono in una versione rivista di script per la costruzione di spazi semantici distribuzionali realizzati dal lavoro congiunto di Marco Baroni e Alessandro Lenci. L'esecuzione avviene in maniera consequenziale. Ognuno di essi riceve in input un file di testo in un determinato formato[10] e restituisce in output uno o più file di testo che saranno utilizzati come input dagli script successivi. È opportuno sottolineare che il progetto parte dall'ipotesi che i testi caricati dagli utenti siano già stati annotati sintatticamente e quindi ogni token sia corredato dalla corrispondente POS. Gli script di cui lo strumento fa uso sono, in ordine di esecuzione, i seguenti:

1. *print_directional_bigrams.pl*

   Questo script riceve in input:

   a) un file di testo contenente un corpus linguistico il cui contenuto deve essere costituito da token disposti secondo il formato di un token per ogni riga e ogni frase del corpus dovrà iniziare con un tag <*s*> che fungerà da delimitatore.

   Es.:

   <s>
   il-R
   informazione-S
   contenere-V
   in-E
   opuscolo-S
   impegnare-V
   il-R
   organizzatore-S
   o-C
   il-R

[9] http://www.perl.org/
[10] Il formato dei file di testo è stato modificato attraverso l'utilizzo di espressioni regolari che saranno analizzate nel capitolo successivo.

venditore-S
,-F
a-E
meno-S
che-C
:-F

Si noti che in questo e negli esempi a seguire, sono mostrati frammenti del TEMIS corpus usato in fase di implementazione, nel quale ogni token è costituito dal lemma della parola corrispondente, seguito da un trattino, seguito a sua volta dalla parte del discorso generica del lemma. Il formato è quindi "*lemma-POS*";

b) un file di testo contenente una lista di parole pivot (o target). Queste parole costituiranno gli elementi dei quali si vuole conoscere il comportamento distribuzionale rispetto alle parole del corpus, in altri termini rappresenteranno i vettori riga e colonna della matrice parola-contesto. Pertanto la natura di questo file è strettamente legata al tipo di indagine da svolgere. In fase di implementazione si è optato per selezionare unicamente i sostantivi, gli aggettivi e i verbi che ricorrevano nel TEMIS corpus con una frequenza superiore a cinquanta. Questa scelta è stata compiuta poiché è stato deciso di escludere dall'analisi i token con scarso potere semantico discriminatorio, quali gli articoli o le preposizioni, e in generale tutte quelle parole che ricorrevano in numero eccessivamente basso in relazione ai circa 600.000 token di cui il TEMIS corpus è composto. Anche in questo caso i token devono essere disposti secondo il formato di uno per riga ma il tag delimitatore di frase <*s*> non deve essere presente.

Es.:

contenere-V
opuscolo-S
organizzatore-S

c) l'ultimo elemento che lo script riceve come input è la dimensione della finestra di contesto. Questo valore indica la massima distanza, misurata in token, che intercorre tra la posizione del pivot e la posizione del contesto. In altri termini il valore indica quanti token, precedenti e successivi alla parola target, dovranno essere selezionati per formare ciascuna co-occorrenza. La finestra di contesto di dimensione *n* selezionerà *n*-1 co-occorrenze. Pertanto, considerati gli esempi precedenti, un finestra di contesto con *n* pari a 3 selezionerà le seguenti co-occorrenze:

contenere-V il-R
contenere-V informazione-S
contenere-V in-E
contenere-V opuscolo-S
opuscolo-S contenere-V
opuscolo-S in-E
opuscolo-S impegnare-V
opuscolo-S il-R
organizzatore-S impegnare-V
organizzatore-S il-R
organizzatore-S o-C
organizzatore-S il-R

Negli ultimi quindici anni circa, la dimensione di quella che si definisce finestra di contesto è stata oggetto di discussione da parte di molti ricercatori di linguistica computazionale. Lund e Burgess (1996) usarono una finestra di contesto di dieci parole. Schütze (1998) usò una finestra di cinquanta parole centrata sulla parola target. Rapp (2003) ha invece dimostrato che risultati ottimali si raggiungono al restringimento della finestra, conseguendo un risultato del 92.5% di risposte corrette su 80 domande a risposta multipla riguardo l'individuazione di sinonimi nel Test of English as a Foreign Language (TOEFL), usando una finestra di contesto di dimensioni pari a quattro (±2 parole, centrata sulla parola target). Ne consegue che le parole

del contesto più vicine alla parola target sono molto più importanti di quelle lontane nel compito di determinare il significato.

Questo script restituisce in output un file di testo contenente le co-occorrenze selezionate in base alla finestra di contesto e separate da una lettera, l o r (left, right), ad indicare se la parola del contesto si trovi a sinistra o a destra rispetto al target.

Es.:

maniera-S l in-E
maniera-S l indicare-V
maniera-S r leggibile-A
maniera-S r ,-F

2. *asso.measures.pl*

Questo script riceve in input un file di testo contenente coppie di parole seguite ciascuna dalla relativa frequenza di ricorrenza all'interno del corpus. Lo script prevede che le triple così formate debbano essere disposte in modo che ogni riga contenga una ed una sola tripla.

Es.:

accesso-S indicazione-S 2
accesso-S informazione-S 33
accesso-S interessato-A 1
accesso-S lettera-S 6

Lo script misura la salienza statistica di ogni co-occorrenza applicando il calcolo della Local Mutual Information come visto nel paragrafo 2.5 e restituisce in output un file di testo che presenta su ogni riga il formato:

parola1 parola2 freqcooc freqpar1 freqpar2 lmi

Es.:

accertare-V	*atto-A*	*2*	*269*	*83*	*12.3896*
accertare-V	*automatico-A*	*1*	*269*	*8*	*8.5698*
accertare-V	*autorità-S*	*1*	*269*	*1111*	*1.4522*
accertare-V	*avere-V*	*17*	*269*	*1990*	*79.8784*
accertare-V	*biocidi-S*	*1*	*269*	*126*	*4.5925*

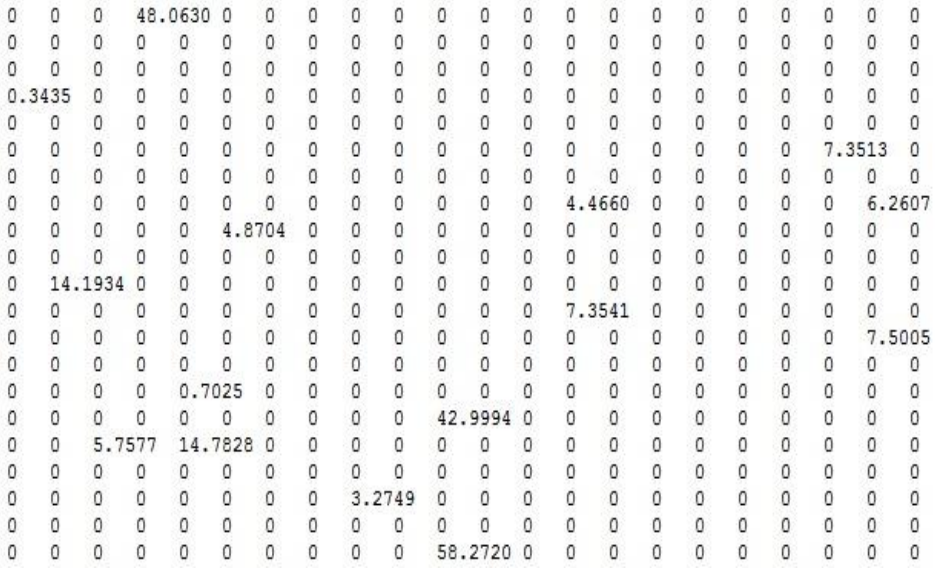
3. *build_matrix_from_tuples.pl*

   Questo script genera una matrice di co-occorrenza di tipo parola-contesto, come visto nel paragrafo 2.2, a partire da coppie di parole seguite dal rispettivo punteggio di salienza statistica. Pertanto il formato del file in input presenta triple nel formato:

   parola1 parola2 punteggio

   Il primo elemento di ogni tripla costituirà uno dei vettori riga della matrice mentre il secondo elemento costituirà uno dei vettori colonna, che in questo caso rappresentano le parole del contesto vicino alle quali il primo elemento ricorre.
   Lo script genera un file con estensione *.mat*, contenente la matrice parola-contesto nella quale il primo campo sarà riempito con i vettori riga di cui sopra, mentre i restanti campi saranno popolati dal punteggio che ogni vettore riga ha totalizzato in ciascun vettore colonna in cui ricorre. Se l'elemento del vettore riga non ricorre nel contesto dell'elemento del vettore colonna, oppure il punteggio di Local Mutual Information è negativo, allora il campo corrispondente sarà riempito con il valore 0.

   Es.: (frammento di file matrice)



4. *build_pairs.pl*

   Lo script riceve in input un file di testo popolato da una lista di termini disposti secondo il formato di uno per riga. L'esecuzione genera un file contenente tutte le possibili combinazioni di parole riscontrate nel file in input. In questo caso, quest'operazione viene effettuata sul file che contiene le parole target oggetto dell'indagine, selezionate precedentemente. Pertanto, l'esecuzione di *build_pairs.pl* su un file in input elencante le seguenti parole:

   articolo
   regolamento
   disposizione

   genererebbe un file contenente le seguenti combinazioni:

   articolo regolamento
   articolo disposizione
   regolamento disposizione

5. *compute_cosines_of_pairs.pl*

   Per ultimo, il suddetto script riceve in input il file appena creato dallo script precedente, contenente combinazioni di parole e il file con estensione *.mat* creato tramite l'esecuzione dello script

build_matrix_from_tuples.pl contenente la matrice parola-contesto il cui primo campo di ogni vettore riga è occupato dal termine che il vettore rappresenta mentre i restanti campi sono riempiti col punteggio di salienza statistica che ogni elemento del vettore riga ha totalizzato nel relativo contesto.

L'esecuzione dello script calcola il coseno dell'angolo formato dai vettori riga della matrice nello spazio semantico vettoriale al fine di rilevarne la similarità semantica.

Il file in output presenterà quindi un elenco di co-occorrenze seguite dal corrispettivo coseno, nel seguente formato:

Es.:

approvazione-S	*consiglio-S*	*0.08071*
approvazione-S	*chimico-A*	*0.01244*
approvazione-S	*strategia-S*	*0.03459*
approvazione-S	*raggiungere-V*	*0.00960*
approvazione-S	*controllare-V*	*0.00956*

Se uno o entrambi gli elementi della coppia non sono presenti nella matrice, lo script restituirà un coseno pari a zero.

4. IMPLEMENTAZIONE

4.1 INTRODUZIONE

Come è stato possibile osservare nel capitolo precedente, ogni script incluso nello strumento, per funzionare correttamente, richiede che il file di testo sul quale venga eseguito presenti un determinato formato di distribuzione e presentazione del contenuto. I file di testo generati da alcuni degli script non possiedono un formato compatibile con l'esecuzione dello script successivo. Pertanto è necessario modificare il contenuto del file prima che questo sia utilizzato come input dallo script che lo necessita. Per assolvere tale compito l'esecuzione degli script è intervallata da comandi interpretati da *bash*[11], la shell testuale dei sistemi operativi *unix* e *unix-like*.

La maggior parte di questi comandi fa uso delle espressioni regolari o *regex* (*regular expression*). Queste sono sequenze di simboli che identificano un insieme di stringhe. Si considerano come funzioni che fissano determinati parametri di ricerca che la stringa in input deve soddisfare. Tuttavia, il loro scopo non si esaurisce nella mera ricerca. Le espressioni regolari sono adoperate in particolar modo nella sostituzione di porzioni di testo e sono, per questo motivo, largamente utilizzate in tutti quei contesti in cui è necessario modificare file di testo che presentano caratteristiche ricorrenti, come ad esempio gli elenchi.

Per loro natura le espressioni regolari possono essere formulate in infiniti modi ed è comune il caso in cui diverse espressioni permettano di raggiungere il medesimo scopo. Quelle discusse in questo capitolo sono le espressioni regolari sviluppate dall'autore di questa relazione, ma non devono essere considerate come le uniche in grado di raggiungere gli obiettivi prefissati.

In questo capitolo sono illustrati i comandi *bash* utilizzati per preparare i file corpus e pivot usati come modelli di riferimento per lo sviluppo dello strumento, e i comandi *bash* interni allo strumento, utilizzati per

[11] http://it.wikipedia.org/wiki/Bash

modificare il formato dei file prodotti in output dagli script, in modo da risultare compatibili come file in input per l'esecuzione degli script successivi. Infine sarà illustrato lo sviluppo del sito che costituirà l'interfaccia vera e propria dello strumento.

4.2 PREPARAZIONE DELLE RISORSE

Nel paragrafo 3.2 è stato descritto il formato del corpus preso come modello durante lo sviluppo del progetto. È facile comprendere come il formato con cui il TEMIS corpus si presenti non sia compatibile col tipo di formato richiesto dal primo dei cinque script esaminati nel capitolo precedente. Infatti, lo script *print_directional_bigrams.pl* deve ricevere in input un file corpus che elenchi un insieme di frasi separate da tag <s>. Per questa ragione la prima delle operazioni da portare a termine è selezionare soltanto due delle colonne di cui è composta ogni riga del TEMIS corpus, in particolare, la colonna contenente il lemma e la colonna contenente la parte del discorso generica. Per adempiere questo compito, è stata utilizzata la seguente espressione regolare:

```
s/^[^\s]+\s[^\s]+\s([^\s]+)\s([^\s]+).*/$1-$2/g
```

Questo tipo di sintassi seleziona per intero ogni riga del TEMIS corpus e la sostituisce con il contenuto del terzo e del quarto elemento che in essa figura, separando i due elementi con un simbolo di trattino "-".

Es.: (riga TEMIS corpus prima dell'applicazione della regex)

22 condizioni condizione S S num=p|gen=f 2 conj _ _

Es.: (riga TEMIS corpus dopo l'applicazione della regex)

condizione-S

La prossima operazione da compiere è quella di inserire il tag delimitatore <s> all'inizio di ogni frase. Per questo scopo è stato sfruttato il fatto che le frasi racchiuse nel TEMIS corpus siano separate da righe vuote ed è stata, pertanto, utilizzata un'espressione regolare che sostituisce a due o più ritorni a capo consecutivi, un ritorno a capo seguito dal tag <s>, seguito a sua volta da un ulteriore ritorno a capo.

```
s/([\n]){2,}/$1<s>$1/g
```

A questo punto il file corpus avrà assunto il formato richiesto.

Es.:

<s>
2-N
.-F
<s>
qualora-C
venire-V
mettere-V
a-E

4.3 PROCESSI INTERNI ALL'APPLICAZIONE

Le espressioni regolari e i comandi approfonditi nel paragrafo precedente sono stati progettati per adattare, all'esecuzione dell'applicazione, le risorse usate come modelli di riferimento in fase di implementazione. In questo paragrafo saranno invece analizzati i comandi e le espressioni regolari che intervallano l'esecuzione di ciascuno script e, pertanto, sono parte integrante dello strumento sviluppato.

A questo punto è necessario capire come ogni script modificherà il contenuto e il formato del file che riceverà in input affinché sia possibile modificare il file generato in output per renderlo compatibile con l'esecuzione di un altro script.

Nel paragrafo 3.3 si è visto come il primo degli script che l'applicazione esegue, cioè *print_directional_bigrams.pl*, generi un file il cui formato presenta coppie di parole separate da una lettera "l" o "r" ad indicare se la parola del contesto si trovi a sinistra (*left*) o a destra (*right*) della parola

target. Quest'informazione è superflua ai fini della nostra analisi dal momento che vogliamo prendere in considerazione le co-occorrenze indipendentemente dalla loro posizione rispetto alla parola target, inoltre il secondo script in ordine di esecuzione, *asso.measures.pl*, deve ricevere in input un file contenente le coppie di parole (in questo caso quelle generate dal primo script) seguite dalla relativa frequenza di co-occorrenza.
Pertanto la prima operazione da effettuare è rimuovere la lettera, qualunque essa sia, che divide la coppia. L'espressione regolare che si occupa di questo compito è la seguente:

```
s/\t[lr]{1}\t/\t/g
```

Lo script *asso.measures.pl* dovrà calcolare la Local Mutual Information di ogni coppia e, dato che un'informazione fondamentale per eseguire tale calcolo è rappresentata dalla frequenza delle coppie, il prossimo passo è contare il numero di occorrenze di ogni coppia e a tale scopo si è fatto ricorso ai comandi *sort* e *uniq* eseguibili da shell. Il primo è un comando di ordinamento mentre il secondo confronta ogni riga con quelle adiacenti eliminando i duplicati. Entrambi, eseguiti con gli opportuni parametri, hanno permesso la creazione di un file in cui ogni coppia di termini è preceduta dal valore di frequenza nel corpus. Il comando utilizzato è composto da due sottocomandi combinati con l'uso della sintassi *pipe* espressa con la barra verticale "|", il primo dei due sottocomandi ordina il file in input alfabeticamente mentre il secondo elimina i duplicati contando le occorrenze.

```
sort FILE | uniq -c
```

Questi due comandi, combinati, generano un file di questo tipo:

Es.:

65 eliminazione-S di-E
1 eliminazione-S direttiva-S
1 eliminazione-S dovere-V

19 eliminazione-S e-C

L'esempio mostra come la frequenza di ogni coppia preceda la coppia stessa, ma il formato del file che lo script *asso.measures.pl* riceve in input deve essere il seguente:

parola1 parola2 frequenza

Perciò la seguente espressione regolare inverte l'ordine degli elementi spostando la frequenza di ogni co-occorrenza al termine della riga.

```
s/\s([0-9]+)\s([^\s]+)\s([^\s]+)/$2\t$3\t$1/g
```

Invece la seguente espressione regolare rimuove eventuali spazi che si possono essere venuti a creare all'inizio di ogni riga in seguito al riordinamento degli elementi.

```
s/^\s+//g
```

Come visto nel paragrafo 3.3, lo script *asso.measures.pl* genera in output un file che mostra su ogni riga una coppia di parole, seguita dalla relativa frequenza di co-occorrenza, seguita a sua volta dalle frequenze di occorrenza delle singole parole conteggiate separatamente e per ultimo il punteggio di Local Mutual Information.
Per prima cosa si procederà alla rimozione di tutte le co-occorrenze che possiedono un valore negativo di Local Mutual Information poiché il loro grado di associazione reciproca sarebbe troppo basso perché sia di qualsivoglia interesse ai fini dell'indagine.
Anche in questo caso l'operazione è portata a compimento tramite l'utilizzo di un'espressione regolare. Quest'ultima seleziona ogni riga in cui compare un segno "-" prima del punteggio di associazione statistica e la elimina.

```
s/[^\s]+[\s][^\s]+[\s][^\s]+[\s][^\s]+[\s][^\s]+[\s]-
[^\s]+//g
```

La sopraccitata espressione può generare la comparsa di righe prive di testo all'interno del file sul quale viene applicata. Quindi, come già visto in precedenza, si procede alla rimozione di tali righe applicando l'espressione regolare adatta allo scopo.

L'ultima modifica da applicare al file prima di affidarlo all'esecuzione dello script che si occuperà di creare la matrice di co-occorrenza consiste nella rimozione delle colonne concernenti le frequenze delle co-occorrenze e dei singoli termini poiché non sono compatibili col formato richiesto dallo script *build_matrix_from_tuples.pl*. A tal proposito è stata usata la seguente espressione regolare:

```
s/([^\s]+[\s][^\s]+)[\s][^\s]+[\s][^\s]+[\s][^\s]+[\s
][^\s]+)/$1\t$2/g
```

In seguito alle ultime modifiche il file si presenta in questo modo:

Es.:

categoria-S nitrato-S 5.7374
categoria-S notificare-V 2.3624
categoria-S N-S 17.4748
categoria-S o-C 32.5733
categoria-S occorrere-V 2.0439
categoria-S ogni-D 5.2291
categoria-S ogniqualvolta-B 5.4895
categoria-S olio-S 11.3860

A questo punto l'applicazione avrà tutte le risorse necessarie per:

1. Creare la matrice parola-contesto tramite esecuzione di *build_matrix_from_tuples.pl* sul file appena esaminato.
2. Generare un file di testo contenente tutte le possibili combinazioni delle parole target tramite l'esecuzione di *build_pairs.pl*.

3. Calcolare il coseno dell'angolo formato dai vettori riga e colonna della matrice eseguendo lo script *compute_cosines_of_pairs.pl* sui file generati dagli script *build_matrix_from_tuples.pl* e *build_pairs.pl*.

Al termine di questi tre passaggi, il file risultante presenterà coppie parole seguite dal relativo coseno. Il file così composto sarà ordinato in ordine decrescente di coseno con l'esecuzione di un ultimo comando.

```
sort -r -k 3 FILE
```

Quest'operazione è necessaria ai fini di una corretta implementazione della funzionalità di ricerca che lo strumento mette a disposizione dell'utente e che sarà illustrata nel paragrafo successivo.

4.4 PROGETTAZIONE

4.4.1 INTRODUZIONE

Per la realizzazione dello strumento si è deciso di ricorrere ad uno dei linguaggi di programmazione più popolari nel mondo del *web development*. Nonostante siano trascorsi circa diciassette anni dalla sua nascita, il PHP (*PHP: Hypertext Preprocessor*) continua ad essere un saldo punto di riferimento per lo sviluppo di numerosissimi siti web. Basti pensare a piattaforme del calibro di facebook e wikipedia. Il linguaggio PHP è interpretato da un web server che invia al client un documento, solitamente html, che il browser è in grado di mostrare all'utente.

In questa sede, l'obiettivo è la creazione di un *wrapper*, cioè un contenitore in grado di raggruppare gli script e i comandi analizzati nei paragrafi precedenti e garantirne la corretta esecuzione sequenziale. Tale wrapper è stato realizzato in PHP e pertanto l'esecuzione degli script è interamente affidata alla macchina server, la quale deve disporre

necessariamente di un interprete PERL e di una sua estensione, chiamata *Perl Data Language*[12], che consente a PERL di manipolare velocemente grandi quantità di dati.
Gli elementi dinamici del sito, come l'animazione che segnala all'utente che l'esecuzione è in corso, sono stati realizzati tramite l'uso di JavaScript[13], altro diffusissimo linguaggio di programmazione utilizzato in ambito web.
In questo paragrafo saranno esaminate le funzioni più peculiari dello strumento.

4.4.2 CREAZIONE SESSIONE UTENTE

Ogni qualvolta un utente accede al sito, il frammento di codice seguente provvederà alla creazione di un ID univoco, generato casualmente a partire da una sequenza di numeri e lettere, che identificherà ogni singolo utente per tutta la durata della sua sessione di lavoro. Questo ID sarà inoltre utilizzato per la creazione di una cartella, risiedente sul server, che ogni utente avrà a disposizione. All'interno di questa cartella saranno inseriti sia i file caricati tramite gli opportuni form di upload, sia quelli prodotti dai diversi script.

```
session_start();

 if (!isset($_SESSION['user'])){
       $usercodelength = 20;
       $base = "abcdefghijklmnopqrstuvwxyz0123456789";
       $usercode="";
       for($i=0;$i<=$usercodelength;$i++) {

 $usercode.=substr($base,rand(0,strlen($base)),1);
       }
       $_SESSION['user'] = $usercode;
       mkdir('contents/'.$_SESSION['user'] , 0777);
 }
```

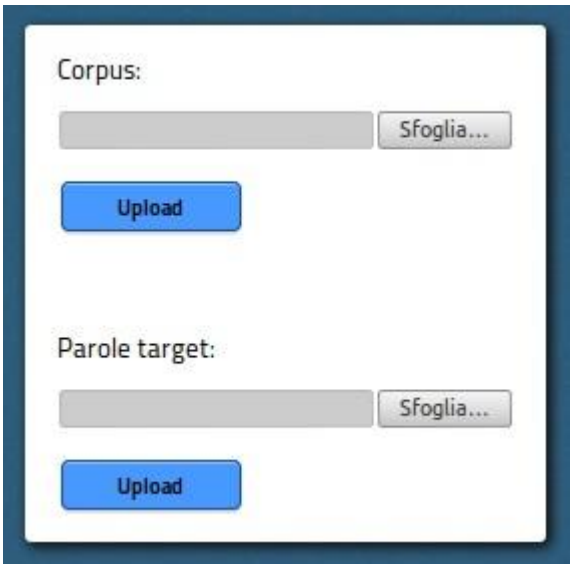
La funzione PHP *substr($string,$start,$lenght)* riceve come parametri una stringa e due numeri interi e restituisce una sottostringa della stringa in input ricavata utilizzando i due numeri interi, rispettivamente, come

[12] http://pdl.perl.org/
[13] http://en.wikipedia.org/wiki/JavaScript

indice del primo carattere della sottostringa e come lunghezza di quest'ultima. Il primo numero intero ricevuto dalla funzione *substr* è un valore numerico casuale compreso tra 0 e 20 e selezionato tramite la funzione PHP *rand($min,$max)*, invece il secondo parametro è sempre il valore intero 1. In questo modo, ad ogni esecuzione del ciclo *for* sarà selezionato un carattere casuale della stringa alfanumerica contenuta nella variabile *$base* che costituisce la base della selezione. Dopo un totale di 20 esecuzioni del ciclo *for* sarà stato finalmente generato il codice univoco legato all'utente corrente e sarà creata la corrispettiva cartella di lavoro.

4.4.3 CARICAMENTO DELLE RISORSE

La prima delle operazioni che l'utente si troverà ad effettuare è il caricamento dei file che costituiranno le risorse alla base della sua analisi. Tramite un apposito riquadro, creato per lo scopo, l'utente potrà selezionare i due file e caricarli, uno per volta, nello strumento.



I form di upload accettano esclusivamente file di tipo txt. Il tentativo di caricamento di qualsiasi altro formato di file restituirà un errore. Questo è stato realizzato utilizzando la seguente funzione.

```
function upload($file, $folder){
	if($file == "corpus"){
		$nomefile = "corpus.parsed";
	if ($_FILES[$file]['type'] == 'text/plain'){
		$_SESSION['msgcorpus'] = 'File Caricato.';
		move_uploaded_file($_FILES[$file]['tmp_name'],
		$_SESSION['folder']."/" . $nomefile);
	}
	else {
		$_SESSION['msgcorpus'] = 'Formato non
		valido.';
		}
	} elseif($file == "target"){
		$nomefile = "target.txt";
	if ($_FILES[$file]['type'] == 'text/plain'){
		$_SESSION['msgtarget'] = 'File Caricato.';
		move_uploaded_file($_FILES[$file]['tmp_name'],
		$_SESSION['folder']."/" . $nomefile);
	}
	else {
		$_SESSION['msgtarget'] = 'Formato non
		valido.';
	}
	}
}
```

Per prima cosa il primo costrutto *if/else* verifica se la chiamata alla funzione si è verificata in seguito al tentativo di upload del file corpus o del file contenente le parole target. Questa distinzione è necessaria perché entrambi i file, qualunque sia il loro nome originale, saranno rinominati rispettivamente in *corpus.parsed* e *target.txt*. Così facendo non si incorrerà in nessun tipo di problema nell'esecuzione degli script.

Con la successiva istruzione condizionale *if/else* sarà effettuato il controllo sul tipo di file caricato. Questo controllo sarà superato, come si è detto, soltanto se il file è di tipo testuale, onde evitare che l'utente malintenzionato o semplicemente distratto possa caricare sul server file di altro tipo, quali script che potrebbero compromettere la sicurezza del sito. In caso di esito positivo, il file sarà spostato, con la funzione PHP *move_uploaded_file($filename,destination)*, nella cartella, creata in precedenza durante l'inizializzazione della sessione, e avente come nome il codice alfanumerico casuale attribuito all'utente corrente.

4.4.4 ESECUZIONE DELL'APPLICAZIONE

Ora che entrambi i file si trovano al loro posto. L'utente può immettere, attraverso un'apposita sezione, la dimensione della finestra di contesto che, come detto in precedenza, determinerà il numero di co-occorrenze selezionate dall'applicazione.

La pressione del pulsante start innescherà l'esecuzione del file *core.php*. In esso risiede la totalità degli script, delle espressioni regolari e dei comandi analizzati nei paragrafi 3.3 e 4.3. Ognuno di essi è eseguito utilizzando la funziona PHP *shell_exec($command).*

Es.:

```
shell_exec('sort tuple.txt | uniq -c >
target_contesto_frequenza.txt');
```

Per tutta la durata dell'esecuzione dei diversi comandi, è stato deciso di mostrare una barra animata con l'obiettivo di segnalare all'utente che l'applicazione sta processando i file e quindi di attenderne la completa esecuzione.

4.4.5 DOWNLOAD

Alla scomparsa della barra animata, farà la sua comparsa, in cima alla pagina, un pulsante tramite il quale l'utente potrà scaricare il file di testo contenente il risultato della sua indagine, cioè la distanza semantica, espressa in coseno, tra le parole target e le parole che fanno parte del loro contesto.

L’apparizione del pulsante di download è gestita con PHP. In dettaglio, il pulsante rimarrà nascosto fino a quando nella cartella di lavoro dell’utente non sarà stato creato, dallo script opportuno, il file risultante dall’esecuzione di tutti i comandi. Lo stesso ragionamento è applicato al riquadro che ospita la funzione di ricerca, giacché anche quest’ultima è legata al file generato in output dall’ultimo comando. La porzione di codice che si occupa di adempiere questo compito è la seguente:

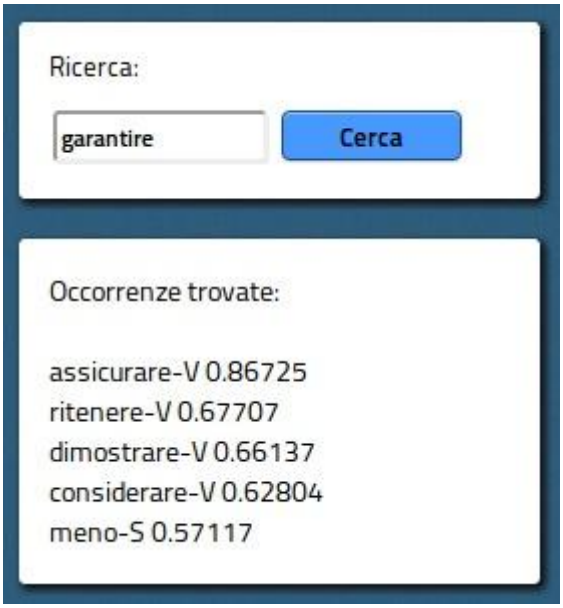
```
$check = "contents/".$_SESSION['user']."/cos-list.txt";
if (file_exists($check)){

    $downloadbutton = '<div id="download">
                       <a  id="downloadlink"
                       href="download.php?filename=cos-
                       list.txt"
                       TARGET="_blank">Download</a>
                       </div>';

    $searchdiv = '<div id="search">
                  <p>Ricerca:</p><br/><input type="text"
                  id="parola" placeholder="Parola da
                  ricercare" size="15"/><input
                  type="button" class="submit" id="ricerca"
                  value="Cerca"/>
                  </div>
                  <div id="result">

                  </div>';
}
```

In questo caso è la funzione *file_exists($file)* ad essere il fulcro dell’operazione da effettuare. Questa riceve come parametro una stringa corrispondente al percorso di un file e restituisce il valore booleano *TRUE* se il file esiste o *FALSE* nel caso contrario. Nel primo tra i due casi, le variabili $downloadbutton e $searchdiv saranno inizializzate al codice HTML relativo, rispettivamente, al pulsante di download al riquadro di ricerca.

4.4.6 RICERCA

L’applicazione consente di effettuare la ricerca di una parola all’interno dell’insieme delle parole target che l’utente ha caricato per scoprire le

cinque parole con la più alta similarità semantica rispetto a quella ricercata.



Questo compito è affidato al codice contenuto all'interno del file *ricerca.php*. La funzione PHP scelta per svolgere il compito di ricercare una parola all'interno di un file di testo è chiamata *preg_match_all($regex,$text,$matches,$flag)*. Tale funzione ricerca tutte le stringhe conformi all'espressione regolare passata nel parametro *$regex* all'interno della stringa di testo passata nel parametro *$text* e inserisce tutti i testi riconosciuti all'interno dell'array *$matches* secondo l'ordine specificato dal *$flag*.

In questo caso, l'espressione regolare corrisponde alla parola cercata dall'utente, invece la stringa di testo, passata come secondo parametro alla funzione *preg_match_all*, comprende il contenuto dell'intero file contenente le coppie di parole seguite dal relativo valore di similarità semantica, ordinate secondo ordine decrescente di coseno.

Per correggere l'output della funzione e fare in modo che lo strumento mostrasse soltanto la parola semanticamente affine a quella cercata e non l'intera riga coppia-valore, sono state utilizzate altre due funzioni in modo combinato. La prima, *preg_match($regex,$text)*, ricerca una stringa conforme all'espressione regolare passata come primo parametro

all'interno della stringa passata come secondo parametro (in questo caso la riga del file di testo nel formato coppia-valore), mentre la seconda, *preg_replace($regex,$replaced,$text)*, ricerca una stringa conforme all'espressione regolare passata come primo parametro all'interno della stringa di testo passata come terzo parametro, cioè la coppia di parole seguite dal coseno, e la sostituisce utilizzando l'espressione regolare passata come secondo parametro.

Queste operazioni sono state inserite in un ciclo iterativo *for* che si interrompe dopo cinque esecuzioni in modo da limitare il numero di risultati mostrati.

```
if((preg_match_all($pattern, $contents, $matches,
PREG_SET_ORDER)) AND ($_POST['parola'] != '')){
    $s .= "<p>Occorrenze trovate:</p><br/>";
    for($i = 0; $i < 5; $i++){
        if(preg_match($replaced1, $matches[$i][0])){
            $s .= preg_replace($replaced1,    $replacement1,
            $matches[$i][0])."<br/>";
        }
        else{
            $s .= preg_replace($replaced2,   $replacement2,
            $matches[$i][0])."<br/>";
        }
     }
}
```

4.4.7 CHIUSURA SESSIONE

Il sito è progettato in modo tale che nel momento in cui l'utente decide di terminare il lavoro e di chiudere il browser o soltanto la scheda contenente lo strumento, una funzione si occuperà di rimuovere la cartella che gli era stata precedentemente assegnata, compreso il relativo contenuto. Questa soluzione è stata pensata poiché ogni cartella di lavoro raggiunge dimensioni considerevoli e quindi si rivela necessario liberare il server dal contenuto ormai inutile, in modo da non saturarlo dopo numerose esecuzioni.

La funzione utilizzata in questo caso, è stata realizzata ad-hoc e prevede l'utilizzo della funzione PHP *unlink($filepath)* la quale riceve come

parametro il percorso ad un file o ad una cartella e provvede alla sua istantanea rimozione.

Quest'azione è innescata mediante l'utilizzo dell'evento javascript *window.onbeforeunload* che intercetta la chiusura della pagina o del browser.

5. CONCLUSIONI

Oggigiorno, i metodi di analisi semantica applicabili ad un corpus linguistico sono molto numerosi ed in costante crescita. Calcolare il coseno dell'angolo formato dai vettori riga di una matrice parola-contesto all'interno di uno spazio semantico vettoriale è soltanto uno dei diversi modi per evincere dati empirici rilevanti in termini di similarità semantica tra espressioni linguistiche. Questa relazione ha voluto fornire una breve introduzione alle numerose tematiche che un argomento come la semantica distribuzionale può portare in superficie ed ha voluto approfondire uno degli approcci più popolari a questa tipologia di analisi.

Il sottoscritto ritiene che lo strumento realizzato possa fornire un valido contributo a chiunque abbia come obiettivo l'individuazione di coppie di parole aventi un forte legame semantico combinatorio. Lo scopo prefissato era appunto quello di progettare un'applicazione in grado di velocizzare ed automatizzare il più possibile una serie di passaggi ed elaborazioni che, se compiute manualmente, comporterebbero un notevole dispendio di tempo.

Lo strumento è sicuramente suscettibile di miglioramenti, ad esempio, uno dei limiti che l'utente potrebbe riscontrare è dato dal formato richiesto per i file in upload. Attualmente sono disponibili online svariati strumenti orientati al Natural Language Processing[14] ed è auspicabile che in un immediato futuro il tool oggetto di questa relazione possa essere interfacciato con uno di essi in modo da consentire agli utenti l'upload di un file di testo generico non ancora tokenizzato o annotato.

[14] http://medialab.di.unipi.it/wiki/SemaWiki

6. 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.

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.

Furnas, G.W., Landauer, T.K., Gomez, L.M., & Dumais, S.T. (1983), "*Statistical semantics: Analysis of the potential performance of keyword information system*". Bell System Technical Journal, 62 (6), 1735-1806.

Harris, Z.S. (1951), "*Methods in Structural Linguistics*". Chicago: University of Chicago Press.

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.

Lewis, D. (1972), "*General semantics*". In Harman Gilbert & Donald Davidson (eds.). "*Semantics of Natural Language*". Dordrecht: Reidel. 162-218.

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.

Lund, K., & Burgess, C. (1996), "*Producing high-dimensional semantic spaces from lexical co-occurrence*". Behavior Research Methods, Instruments, and Computers, 28 (2), 203-208.

Miller, G.A. & Charles, W.G. (1991), "*Contextual correlates of semantic similarity*". Language and Cognitive Process VI. 1-28.

Rapp, R. (2003), "*Word sense discovery based on sense descriptor similarity*". In proceedings of the Ninth Machine Translation Summit, pp. 315-322.

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

Shütze, H. (1998), "*Automatic word sense discrimination*". Computational Linguistics, 24 (1), 97-124.

Shütze, H. & Pedersen, J. (1993), "*A vector model for syntagmatic and paradigmatic relatedness*". In Making Sense of Words: Proceedings of the Conference, pp. 104-113, Oxford, England.

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

Turney, P.D. (2008a), "*The latent relation mapping engine: Algorithm and experiment*". Journal of Artificial Intelligence Research, 33, 615-655.

Turney, P.D. (2008b), "*A uniform approach to analogies, synonyms, antonyms, and association*". In Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008), pp. 905-912, Manchester, UK.

Turney, P.D., & Littman, M.L. (2003), "*Measuring praise and criticism: Inference of semantic orientation from association*". ACM Transactions on Information Systems, 21 (4), 315-346.

Turney, P.D., Littman, M.L., Bigham, J., & Shnayder, V. (2003), "*Combining independent modules to solve multiple-choice synonim 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., & Pantel, P. (2010), "*From frequency to meaning: Vector space models of semantics*", Journal of Artificial Intelligence Research (JAIR), XXXVII, 141-188.

Venturi G. (2012), "*Design and Development of TEMIS: a Syntactically and Semantically Annotated Corpus of Italian Legislative Texts*", In Proceedings of the 4th Workshop on "Semantic Processing of Legal Texts", held in conjunction with LREC 2012, Istanbul, Turkey, 27th May.

Weaver, W. (1955), Translation. In Locke, W., & Booth, D. (Eds.), "*Machine Translation of Languages: Fourteen Essays"*. MIT Press, Cambridge, MA.