Università degli studi di Genova

Facoltà di Scienze Matematiche Fisiche e Naturali

Corso di Diploma in Informatica

Anno Accademico 2003/2004

Implementazione e Valutazione di Tecniche di Information Retrieval basate su Stem, Lemma e Synset

Candidato

Luisa Calcagno

Relatori

Prof. Francesco Masulli

Prof. Stefano Rovetta

Relatore esterno:

Prof. Paolo Rosso

(Universidad Politécnica de Valencia)

Non è forte colui che non cade mai,
ma colui che cadendo si rialza.
Johann Wolfgang von Goethe

Ringraziamenti

Innanzitutto desidero ringraziare i professori Francesco Masulli, Stefano Rovetta, ed in particolar modo Paolo Rosso per i suoi preziosi consigli.
Un grazie di cuore ai miei genitori, che mi hanno dato l'opportunità di concludere l'Università. Un grazie speciale, unico e soprattutto sentito a Davide, che mi permette tutti i giorni di essere me stessa e che mi incita a realizzare tutti i miei sogni. Grazie alla zia Anna, che ha sempre creduto in me fin da piccola e mi ha sempre dato una spinta. Grazie a mia sorella Francesca e suo marito Geny, che mi sono stati sempre vicini, sostenendomi ed appoggiandomi in tutti i modi possibili. Grazie a Michela, con cui ho condiviso tante esperienze durante questo percorso universitario. Grazie a Giacomo, Stefano, zio Angelo, zio Franco, zia Teresa, zio Ermano, zio Renzo. Grazie a tutti quelli con cui ho condiviso questi anni all'università: Barbara, Roberto, Stefania. Infine, grazie a tutti quelli che ho dimenticato. So che ci siete, però la mia memoria si ferma qui.

La seguente pubblicazione è stata realizzata nel corso del lavoro svolto per la tesi:

L.Calcagno, D.Buscaldi, P.Rosso, J.M. Soriano Gomez, F.Masulli, S.Rovetta, *Comparison of Indexing Techniques based on Stems, Synsets, Lemmas and Term Frequency Distribution.* In: III Jornadas en Tecnología del Habla, Valencia, 17-19 November 2004 (submitted)

Indice

Introduzione

Oggigiorno quasi tutta l'informazione è memorizzata in formato elettronico, per esempio sotto forma di biblioteche digitali, collezioni di giornali, etc. I meccanismi sviluppati fino ad ora per i sistemi di *Information Retrieval* sono in continua evoluzione grazie alla realizzazione di nuove interfacce ed al miglioramento delle tecnologie esistenti; tuttavia la maggior parte di queste tecniche sono ancora basate sul semplice confronto di parole chiave e sull'uso di informazioni statistiche estratte dai documenti. Un sistema che realizza l'accesso all'informazione sfruttando tali tecniche è detto *motore di ricerca*.

I motori di ricerca attualmente esistenti non considerano le relazioni semantiche. Al fine di permettere all'utente di accedere in maniera più efficiente all'informazione cui è interessato, è cruciale prendere in considerazione il significato delle parole all'interno dei documenti, introducendo in tal modo la possibilità di utilizzare informazioni semantiche e non solo sintattiche. Questo permetterebbe la realizzazione di motori di ricerca *concettuali*. Il progetto di diploma è organizzato nel modo seguente:

nel capitolo 1 vengono introdotti i concetti fondamentali dell'Information Retrieval;

nel capitolo 2 vengono descritti i compiti e la struttura di un sistema di Information Retrieval;

nel capitolo 3 vengono descritti i criteri di valutazione di un sistema di Information Retrieval e le collezioni di test su cui eseguire la valutazione;

nel capitolo 4 i tools e le risorse di elaborazione del linguaggio naturale utilizzate;

nel capitolo 5 vengono descritti gli esperimenti realizzati e i relativi dettagli;

nel capitolo 6, infine, vengono presentate le problematiche dell'Information Retrieval sul web, in confronto con l'Information Retrieval classico; viene inoltre accennata l'architettura di un motore di ricerca per il web.

Capitolo 1

Information Retrieval

1.1 Introduzione all'Information Retrieval

L'*Information Retrieval* (IR) riguarda la rappresentazione, la registrazione, l'organizzazione e l'accesso alle informazioni. Queste devono essere rappresentate ed organizzate in modo da fornire all'utente un facile accesso all'informazione cui è interessato. Sfortunatamente codificare il bisogno di informazione dell'utente non è un problema semplice. Le richieste di informazione dell'utente vengono tradotte in *queries* [1] che vengono elaborate da un motore di ricerca o sistema di IR. Nella forma più comune le queries sono espresse come insiemi di parole chiave (*keywords* o *termini indice*) che riassumono l'informazione desiderata.

Il principale scopo di un sistema di IR consiste nel recuperare documenti da una collezione in risposta alle queries dell'utente. E' importante sottolineare la differenza tra *Information Retrieval* e *Data Retrieval*: quest'ultimo, nel contesto di un sistema di IR, consiste principalmente nel determinare quali documenti di una collezione contengano le parole chiave della query,

ma ciò non è sufficiente a soddisfare il bisogno di informazione dell'utente. Infatti bisogna tener presente che all'utente di un sistema di IR interessa non tanto recuperare i *dati* che soddisfano una particolare query, bensì le *informazioni* riguardanti un particolare argomento.

Il processo di Data Retrieval consiste nel recuperare tutti gli oggetti che soddisfano delle condizioni definite con chiarezza, come quelle di una espressione regolare o di algebra relazionale. Mentre per un sistema di Data Retrieval il recupero di un oggetto errato tra migliaia rappresenta un completo fallimento, nel caso dell'IR gli oggetti recuperati erroneamente possono passare inosservati. Il principale motivo di questa differenza è che l'IR deve fare i conti con testo in linguaggio naturale che non è sempre ben strutturato e può essere semanticamente ambiguo.

Per essere efficace nel suo intento di soddisfare il bisogno di informazione dell'utente, un sistema di IR deve in qualche modo interpretare il contenuto dei documenti ed ordinarli a seconda del grado di rilevanza rispetto a ciascuna query. Per ottenere l'interpretazione del contenuto del documento è necessario estrarre l'informazione sintattica e semantica dal testo. La difficoltà non sta solo nella scelta del modo in cui estrarre tali informazioni, ma anche in come utilizzarle per stabilirne la rilevanza. Pertanto la nozione di *rilevanza* è fondamentale per l'IR. In effetti, lo scopo principale di un sistema di IR è di recuperare tutti i documenti rilevanti per la query dell'utente cercando di recuperare il minor numero possibile di documenti non rilevanti.

1.2 I concetti fondamentali dell'IR

L'efficacia del recupero delle informazioni è influenzata sia dal modo in cui l'utente specifica l'insieme di parole che esprimono la semantica della propria

richiesta di informazione, sia dal modo in cui un sistema di IR rappresenta i documenti in una collezione. Quest'ultimi vengono di solito rappresentati da un insieme di *index terms* o *keywords*, che vengono estratti direttamente dal testo dei documenti in maniera automatica o generate da un esperto. I computer moderni permettono di rappresentare un documento con l'intero insieme delle parole che lo costituiscono ed in tal caso si dice che la rappresentazione dei documenti è di tipo *full text*. Tuttavia, per collezioni molto voluminose potrebbe essere necessario dover ridurre l'insieme di keywords e questo può essere ottenuto attraverso: l'eliminazione di articoli e congiunzioni (*stop-words*), l'uso di *stemming* (che riduce parole a una radice grammaticale comune) e l'identificazione di gruppi di nomi (che elimina aggettivi, verbi e avverbi). Queste elaborazioni sul testo vengono dette *text operation*. La rappresentazione del documento attraverso l'insieme delle parole contenute è chiaramente la più precisa, ma anche quella che comporta i costi computazionali più alti. Un piccolo insieme di categorie generate da un esperto umano fornisce una rappresentazione più concisa, ma anche meno precisa. Un sistema di IR può adottare diverse rappresentazioni intermedie del testo (figura 1.1).

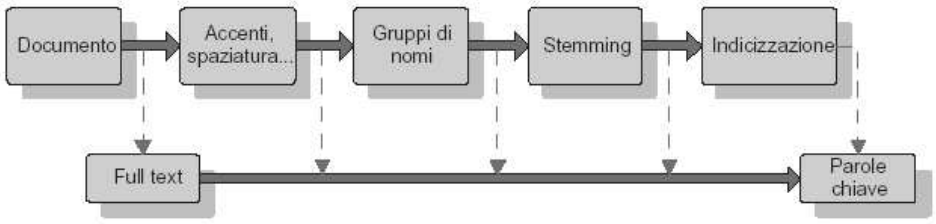


Figura 1.1: La rappresentazione logica di un documento da full text a keywords [1]

1.3 Impatto di Polisemia e Sinonimia sull'IR

Il fatto che l'informazione all'interno dei documenti e le queries siano rappresentate da espressioni del linguaggio umano costituisce un'ulteriore complicazione del task dell'Information Retrieval. Un primo problema è quello della *polisemia*: a differenza dei linguaggi formali, dove ai simboli terminali (le parole del linguaggio) corrisponde un unico significato, nel caso dei linguaggi naturali i simboli terminali possono avere più di un significato (in questo caso si dice che la parola è *polisemica*, mentre quando ad una parola corrisponde un unico significato, allora si dice che è *monosemica*), col risultato che l'ambiguità di una singola parola può venire propagata al resto della frase. La polisemia delle parole è più frequente di quanto si pensi. Per esempio l'aggettivo *vecchio* può avere sia il significato di "vecchio" utilizzato per descrivere qualcosa come appartenente ad un periodo od un'epoca precedente, sia quello di "vecchio" nel senso di "usato", "logoro"; mentre *pellicola* può essere sia un film che il supporto su cui vengono registrate le immagini in una macchina fotografica. Quindi dicendo *una vecchia pellicola* ci si può riferire sia ad un film d'annata, sia ad un rullino rovinato. L'effetto della polisemia nel processo di IR è che ogni volta che una query contiene parole polisemiche i documenti ritornati possono non contenere l'informazione ricercata; ad esempio, potrei voler cercare delle informazioni sull'isola di Java e vedermi ritornare dei documenti sul linguaggio Java.

La *sinonimia*, ovvero l'esistenza di parole con significato equivalente od identico (ad esempio *convegno* e *riunione*), ha per certi versi un effetto contrario: infatti in questo caso, in risposta ad una query che contenga una parola con sinonimi, la probabilità che l'insieme dei documenti ritornati sia incompleto rispetto all'insieme dei documenti rilevanti per la query è sicuramente

superiore al caso in cui la query non contenga parole con sinonimi.

Il problema della sinonimia può essere risolto facendo ricorso a risorse lessicali come i *thesauri*, i quali, data una certa parola, permettono di trovarne i sinonimi. Invece la risoluzione della polisemia avviene attraverso il processo di *disambiguazione semantica* (in inglese *Word Sense Disambiguation* o WSD). Purtroppo la realizzazione di un algoritmo efficiente per la disambiguazione semantica è tuttora un problema aperto nel campo dell'elaborazione del linguaggio naturale. E' stato dimostrato [7] che l'indicizzazione semantica può migliorare l'Information Retrieval, ma solo se la disambiguazione è eseguita con meno del 30% di errore. Purtroppo lo stato dell'arte attuale nel campo della WSD non è in grado di fornire disambiguatori con una precisione superiore al 70% per tutte le categorie lessicali.

Capitolo 2

I sistemi di Information Retrieval

La vista a scatola chiusa di un sistema di Information Retrieval (in inglese *Information Retrieval System*, o IRS) è costituita da tre parti principali: input, output ed il processore [20]. L'input del sistema è caratterizzato dalla richiesta dell'utente, espressa in uno specifico linguaggio, e dai documenti su cui fare la ricerca di informazione. I documenti e le queries, come si è già visto nel capitolo precedente, per poter essere utilizzati dal sistema devono avere una medesima rappresentazione che può essere *diretta*, cioè coincidente con i documenti stessi o *indiretta*, ovvero data da surrogati. La seconda parte del sistema è costituita dal processore, che realizza la fase vera e propria di ricerca e di recupero sulle rappresentazioni dei documenti in risposta alle richieste dell'utente, quindi è il blocco principale in cui vengono implementate le tecniche e le strategie di recupero delle informazioni, che verranno spiegate più avanti in dettaglio, ovvero quei meccanismi che permettono di confrontare la richiesta con le rappresentazioni di documenti. Infine l'output consiste in

una serie documenti che dovrebbero soddisfare la richiesta dell'utente. L'utente potrebbe soddisfare il suo bisogno di informazione in diverse passate traendo spunto dall'output per nuove direzioni di ricerca (ramo "feedback-" della figura 2.1). Un sistema di Information Retrieval prima di iniziare

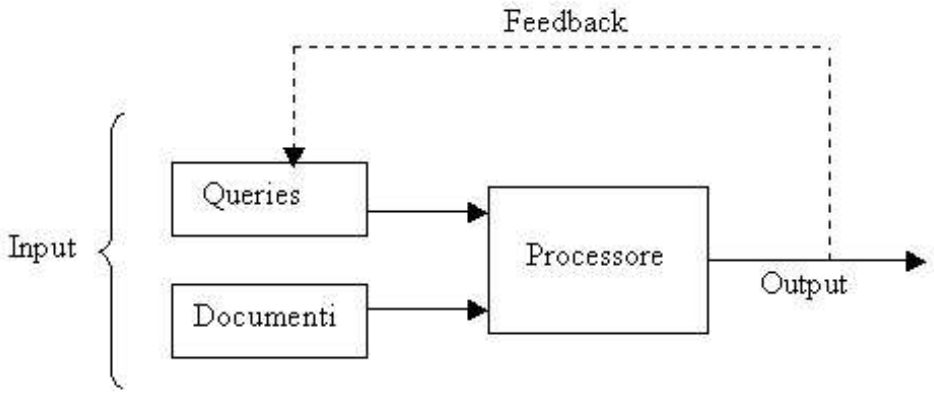


Figura 2.1: Schema di un IR system [20].

il processo di recupero dell'informazione deve eseguire su una collezione di documenti un pre-processo in cui vengono applicate al testo le operazioni che trasformano i documenti originali in una rappresentazione degli stessi. In questa fase ogni documento viene descritto tramite termini indice e ad ogni termine indice viene assegnato un peso che quantifica l'importanza di un termine per descrivere il contesto semantico del documento, come verrà descritto nella sezione 1.5 di questo stesso capitolo. Una volta definita una vista logica dei documenti si passa alla fase di indicizzazione in cui il sistema di Information Retrieval costruisce un indice del testo. Definito un indice questo viene collegato ad uno o più documenti. Un indice è una struttura dati critica poichè permette in seguito una ricerca veloce su grandi volumi di

dati. Dopo aver indicizzato il database dei documenti, il sistema è pronto per effettuare una ricerca, quindi l'utente può specificare la propria richiesta, la quale viene analizzata e trasformata dal sistema tramite le stesse operazione applicate al testo. Il sistema quindi processa la query e cerca i documenti che rispondono meglio alla richiesta. L'indice scelto precedentemente costruito influenza la velocità di processamento delle query. Il sistema prima di fornire all'utente i documenti recuperati li riordina in base a un grado di rilevanza determinato da una misura di similarià. Nella figura 2.2 viene data una possibile architettura di un sistema di Information Retrieval e vengono rappresentati i principali compiti di sistema di Information Retrieval

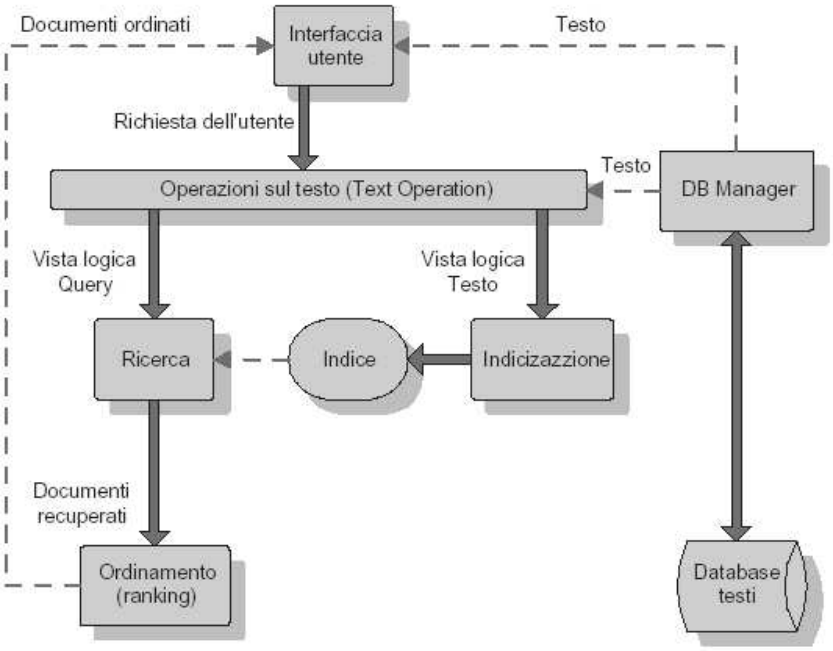


Figura 2.2: Il processo di recupero dell'informazione [1]

2.1 I Modelli di IR

Uno dei problemi principali dei sistemi di IR è quello di predire quali documenti siano rilevanti e quali no; la rilevanza si ottiene a partire da un algoritmo di ranking, il quale tenta di stabilire, sulla base di una misura di similarità, un ordinamento dei documenti recuperati. I documenti in cima alla lista hanno una probabilità maggiore di essere rilevanti. Un tale algoritmo opera secondo dei criteri di rilevanza dei documenti, ossia insiemi di regole che permettono di stabilire quali documenti siano rilevanti e quali no; criteri diversi producono differenti modelli di IR. Il modello definisce la filosofia di fondo di un IRS, ovvero attorno a quali principi generali si è sviluppato il sistema. L'uso di un modello concettuale influenza o determina il linguaggio di interrogazione, la rappresentazione dei documenti, la struttura dei file ed i criteri di recupero dei documenti.

2.1.1 Caratterizzazione formale di un Modello di IR

Una definizione formale di un modello di IR è la seguente:

Definizione 2.1.1 *Un Modello di IR è una quadrupla* $[D, Q, F, R(q_i, d_j)]$ *dove*

1. *D è un insieme di viste logiche (o rappresentazioni) dei documenti nella collezione*
2. *Q è un insieme di viste logiche (o rappresentazioni) relative al bisogno di informazione dell'utente*
3. *F è un framework per modellare le rappresentazioni dei documenti, delle queries e delle loro relazioni*

4. *$R(q_i, d_j)$ è una funzione di ordinamento (ranking) che associa un numero reale ad ogni coppia costituita da una query $q_i \in Q$ e da una rappresentazione di documento $d_j \in D$. Tale ranking definisce un ordine tra i documenti rispetto alla query q_i.*

2.1.2 Tassonomia dei Modelli di IR

I tre modelli classici in IR sono:

- *Booleano*
- *Vettoriale*
- *Probabilistico*

Negli anni sono state proposte delle varianti ai modelli classici: le varianti relative al modello Booleano sono: *Fuzzy* e *Booleano esteso*, quelle relative al modello Vettoriale sono: *Vettore generalizzato*, *Latent Semantic Indexing* e *Neural Network*. Infine le varianti al modello probabilistico sono *Inference Network* e *Belief Network*. In seguito verrà spiegato in dettaglio il modello

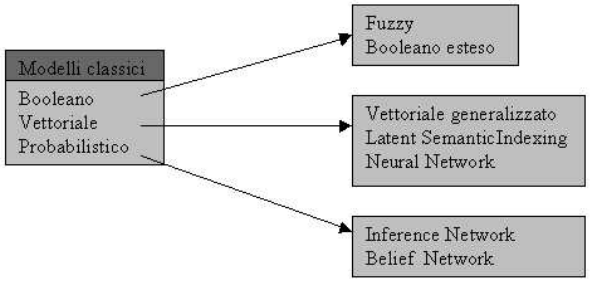


Figura 2.3: Tassonomia dei modelli di IR [1].

Vettoriale, utilizzato in questo lavoro, il modello Booleano, antenato di quello Vettoriale e il modello Latent Semantic Indexing che costituisce al momento attuale l'unico modello basato sulla semantica.

2.1.3 Modelli classici

I modelli classici, come si è visto, considerano ogni documento come descritto da un insieme di parole chiave rappresentative dette anche termini indice. Un termine è essenzialmente una parola la cui semantica aiuta a ricordare gli argomenti principali di un documento. Quindi i termini sono utilizzati per indicizzare e riassumere il contenuto di un documento. In generale i termini sono essenzialmente nomi, poichè in quanto tali hanno un significato preciso ed è quindi più facile capire la loro semantica. Gli aggettivi, i verbi, gli avverbi e le congiunzioni sono meno utili, in quanto hanno una funzione complementare. Dato un documento, non tutti i termini sono ugualmente utili per descriverne il contenuto, infatti ve ne possono essere alcuni che possono essere più vaghi di altri. Decidere sull'importanza di un termine non è un problema banale. Tuttavia vi sono proprietà di un termine che possono essere misurate facilmente, ad esempio data una collezione di centinaia di migliaia di documenti una parola che appare in ognuno di essi è completamente inutile come termine, perchè non dice nulla su quale documento l'utente può essere interessato. D'altro canto, una parola che appare in pochi documenti risulta più utile, in quanto restringe considerevolmente il numero dei documenti a cui l'utente può essere interessato. Dunque termini distinti hanno una varia rilevanza per descrivere la semantica del documento. Questo effetto viene catturato attraverso l'assegnazione di pesi numerici ad ogni termine del documento, che quantificano l'importanza del termine nel descrivere il

contenuto del documento.

Definizione 2.1.1 *Sia t il numero dei termini indice in una collezione e k_i un generico termine indice, $K = k_1, ...k_t$ è l'insieme di tutti i termini indice. Un peso $w_{i,j} > 0$ è associato ad ogni termine k_i di un documento d_j. Per ogni termine k_i che non compare nel testo del documento d_j, $w_{i,j} = 0$. Al documento d_j è associato un vettore di termini $\vec{d_j} = (w_{1,j}, w_{2,j}, ..., w_{m,j})$. Inoltre sia g_i la funzione che restituisce il peso associato al termine k_i in ogni vettore m-dimensionale allora $g_i(\vec{d_j}) = w_{i,j}$.*

In generale si assume (per semplicità) che i pesi dei termini siano mutuamente indipendenti, ciò significa che il peso $w_{i,j}$ associato alla coppia (K_i, d_j) non dà alcuna informazione riguardo al peso $w_{i+1,j}$ associato alla coppia (K_{i+1}, d_j).

2.1.4 Modello Booleano

Il modello Booleano è basato sulla teoria degli insiemi e sull'algebra booleana. Dal momento che il concetto di insieme è intuitivo, il modello Booleano fornisce un framework, costituito dagli insiemi di documenti e dalle operazioni standard su essi, facilmente comprensibile da un utente medio. È proprio per questa sua semplicità che ebbe molto successo in passato.

Nel modello Booleano i documenti e le queries sono rappresentati attraverso vettori di booleani, dove la presenza di un termine indice è segnalata da un 1, l'assenza da uno 0. Più precisamente, una query q è composta dai termini indice collegati dai tre operatori *not*, *and* ed *or* e viene espressa come disgiunzione di vettori congiuntivi (*Disjuntive Normal Form-DNF*). Per esempio la query $q = k_a \wedge (k_b \vee \neg k_c)$ può essere scritta in DNF come $[\vec{q}_{dnf} = (1,1,1) \vee (1,1,0) \vee (1,0,0))]$ dove ogni componente è un vettore booleano di pesi binari associato alla tupla (k_a, k_b, k_c).

Definizione 2.1.2 *Per il modello Booleano, le variabili dei pesi dei termini indice sono tutti binari, cioè* $w_{i,j} \in \{0, 1\}$. *Una query* q *è una convezionale espressione booleana. Sia* $\vec{q}_{dnf}$ *la DNF di una query* q*, e* $\vec{q}_{cc}$ *ognuna delle sue componenti. La similarità di un documento* d_j *alla query* q *è definita come:*

$$sim(d_j, q) = \begin{cases} 1 & se \quad \exists \vec{q}_{cc} \quad | \quad (\vec{q}_{cc} \in \vec{q}_{dnf}) \wedge (\forall k_i, \quad g_i(\vec{d}_j) = g_i(\vec{q}_{cc})) \\ 0 & altrimenti \end{cases}$$

Se $sim(d_j, q) = 1$ *allora il modello Booleano predice che il documento* d_j *è rilevante per la query* q*, altrimenti, la predizione è che il documento non è rilevante.*

Il modello Booleano, pur essendo semplice e chiaro, presenta lo svantaggio di non avere la nozione di corrispondenza parziale con la query. Per esempio, per la query $q = k_a \wedge k_b \vee \neg k_c$ il documento $\vec{d}_j = (0, 1, 0)$ non è considerato rilevante, pur contenendo il termine indice k_b. La corrispondenza esatta alla query può implicare a prendere in considerazione pochi o troppi documenti rispetto al numero totale di documenti rilevanti. Un altro svantaggio è dato dal fatto che con le espressioni booleane non è semplice esprimere le richieste di informazione.

2.1.5 Modello Vettoriale

Il modello Vettoriale sopperisce alle carenze del modello Booleano, fornendo un framework che permette la corrispondenza parziale con la query. Questo è reso possibile assegnando ai termini indice dei pesi non binari, nei documenti e nelle query, per poter esprimere le occorrenze di un dato termine in un documento. Questi pesi vengono poi utilizzati per calcolare il grado di somiglianza tra la query dell'utente ed ogni documento memorizzato nel sistema.

Definizione 2.1.3 *Nel modello Vettoriale, il peso $w_{i,j}$ associato alla coppia (k_i, d_j) è positivo e non binario. Inoltre, anche ai termini indice della query viene assegnato un peso. Sia $w_{i,q}$ il peso associato alla coppia $[k_i, q]$, dove $w_{i,q} \geq 0$. Il vettore della query $\vec{q}$ è quindi definito come $\vec{q} = (w_{1,q}, w_{2,q}, ..., w_{t,q})$, dove t è il numero totale dei termini indice nel sistema. Il vettore di un documento d_j è rappresentato da $\vec{q} = (w_{1,q}, w_{2,q}, ..., w_{t,q})$.*

Quindi, un documento d_j e una query q sono rappresentati come un vettore di dimensioni t. Per esempio si consideri come q la query costituita dalle parole *speech*, *language* e *processing* ($t = 3$), e tre documenti (ad esempio capitoli di un libro) d_0, d_1 e d_2 che contengono, rispettivamente:

- d_0: 1 volta le parole *speech* e *processing*, e 2 volte la parola *language*;
- d_1: 6 volte la parola *speech* e 1 volta la parola *processing*;
- d_2: 5 volte la parola *language* e 1 volta la parola *processing*.

L'obiettivo è di sapere in che capitolo si parla dei tre argomenti: *speech*, *language* e *processing*. Allora una rappresentazione dei vettori risultanti per ciascun documento è mostrata in figura 2.4 . Nel modello Vettoriale ci si propone di valutare il grado di somiglianza tra il documento d_j e la query q come la correlazione tra i vettori $\vec{d_j}$ e $\vec{q}$. Questa correlazione può essere quantificata, ad esempio, come il *coseno dell'angolo* tra i due vettori (figura 2.5), cioè [1]

$$sim(d_j, q) = \frac{\vec{d_j} \cdot \vec{q}}{|\vec{d_j}||\vec{q}|} = \frac{\sum_{i=1}^{t} w_{i,j} w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \sqrt{\sum_{i=1}^{t} w_{i,q}^2}}$$

dove $|\vec{d_j}|$ e $|\vec{q}|$ sono il modulo del vettore del j-simo documento e del vettore relativo alla query. Poichè $w_{i,j} \geq 0$ e $w_{i,q} \geq 0$ il risultato di $sim(q, d_j)$

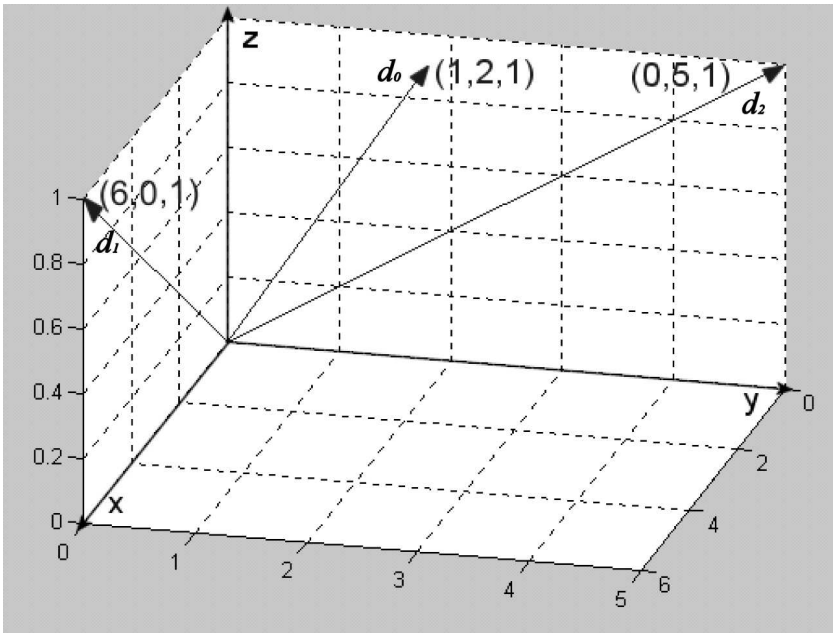


Figura 2.4: Un esempio di rappresentazione vettoriale dei documenti rispetto ad una query (x:speech, y:language e z:processing).

assume un valore compreso tra 0 e 1. Così, invece di tentare di predire se un documento è rilevante o no, il modello Vettoriale ordina i documenti secondo il loro *grado di somiglianza* alla query.

I pesi dei termini indice possono essere calcolati in molti modi. L'idea alla base delle tecniche più efficienti è legata al concetto di *cluster*, proposto da Salton. Si pensi ai documenti come ad una collezione C di oggetti e alla query come una vaga specificazione di un insieme A di oggetti. In questo scenario, il problema dell'IR può essere allora ridotto a quello di determinare quali documenti sono nell'insieme A e quali no; in particolare in un problema di clustering occorre risolvere due sottoproblemi. Occorre determinare quali sono le caratteristiche che descrivono al meglio gli oggetti dell'insieme A (si parla quindi di similarità *intra-cluster*) e quali sono le caratteristiche che meglio distinguono gli oggetti dell'insieme A dagli oggetti della collezione C

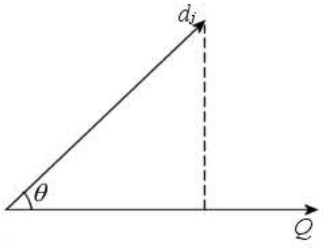


Figura 2.5: Il coseno di θ adottato come $sim(d_j, q)$.

(dissimilarità *inter-cluster*).

Nel modello Vettoriale, la similarità intra-cluster è quantificata misurando la frequenza di un termine k_i in un documento d_i; questa viene detta *Fattore TF (Term Frequency)* e misura quanto un termine descrive i contenuti del documento. La dissimilarità inter-cluster è invece quantificata misurando l'inverso della frequenza di un termine k_i tra i documenti della collezione (*Fattore IDF* o *Inverse Document Frequency*). La motivazione dell'uso del Fattore IDF è che i termini che appaiono in molti documenti non sono molto utili a distinguere un documento rilevante da uno che non lo è.

Definizione 2.1.4 *Sia N il numero totale di documenti del sistema e n_i il numero dei documenti in cui compare il termine indice k_i. Sia $freq_{i,j}$ la frequenza del termine k_i nel documento d_i (cioè il numero di volte in cui il termine k_i è usato nel testo del documento d_i). Allora la frequenza normalizzata $f_{i,j}$ del termine k_i nel documento d_j è data da [1]*

$$f_{i,j} = \frac{freq_{i,j}}{\max_i freq_{i,j}}$$

dove il massimo è calcolato tra tutti i termini menzionati nel testo del documento d_j. Se il termine k_i non compare nel documento d_j allora $f_{i,j} = 0$.

Inoltre, l'Inverse Document Frequency per il termine k_i è dato da

$$idf_i = \log \frac{N}{n_i}$$

Il più noto schema per determinare i pesi dei termini usano i pesi dati da

$$w_{i,j} = f_{i,j} \times idf_i$$

o da una variante di questa formula. Questi sono detti schemi $TF - IDF$.

Sono state proposte alcune varianti alla espressione per i pesi $w_{i,j}$, tuttavia quella descritta prima rimane un buon schema per la determinazione dei pesi per molte collezioni. Per il peso dei termini della query Salton propone la formula [1]

$$w_{i,q} = (0.5 + 0.5 \times {}_{i,q}) \times idf_i = (0.5 + \frac{0.5 \times freq_{i,q}}{\max_l freq_{l,q}}) \times \log \frac{N}{n_i}$$

dove $freq_{i,q}$ indica la frequenza del termine k_i nel testo della richiesta di informazione q. I vantaggi del modello Vettoriale sono diversi, in particolare l'uso dei pesi migliora la performance del recupero dei documenti e consente il recupero di documenti che approssimano le condizioni della query. Inoltre la formula di ranking con il coseno ordina i documenti secondo il grado di somiglianza con la query. Uno svantaggio invece del modello Vettoriale è dato dal fatto che i termini indice sono sempre assunti mutualmente indipendenti.

2.1.6 Latent Semantic Indexing

Il *Latent Semantic Indexing* (LSI) è stato introdotto nel 1988 [1] con l'idea di risolvere i problemi delle tecniche di recupero basate sulla corrispondenza tra i termini delle queries ed i termini dei documenti. Questi problemi sono legati a due caratteristiche del linguaggio parlato, ossia :

- *Sinonimia*: più nomi si riferiscono ad un medesimo oggetto od idea;
- *Polisemia*: molte parole possiedono più di un significato.

La polisemia ha l'effetto di ridurre la precisione delle ricerche, in quanto la ricerca di termini polisemici può indurre a recuperare documenti irrilevanti, mentre la sinonimia può causare la perdita di documenti che contengono termini sinonimi di quelli presenti nella query. L'idea principale alla base del modello Latent Semantic Indexing é quella di rimpiazzare indici che utilizzano insiemi di keywords con indici che utilizzano concetti (gruppi di termini semanticamente correlati). Questo viene effettuato mappando lo spazio vettoriale dei termini indice in uno spazio di dimensioni minori, a cui vengono associati i concetti.

Definizione 2.1.5 *Sia t il numero dei termini indice in una collezione e N il numero totale dei documenti allora si definisce $M = (M_{i,j})$ la matrice termini-documenti dove t è il numero di righe e N il numero delle colonne. Ad ogni elemento $a_{i,j}$ della matrice viene assegnato un peso $w_{i,j}$ associato alla coppia termine-documento $[k_i, d_j]$.*

Il proposito del LSI è di decomporre la matrice di associazione M in tre diverse componenti utilizzando la *Single Value Decomposition* (SVD).

$$M = KSD^t$$

La matrice K è la matrice degli autovettori derivati dalla matrice di correlazione termine-termine data da MM^t. La matrice D^t è la matrice degli autovettori derivati dalla trasposta della matrice documento-documento ottenuta mediante M^tM. La matrice S, infine, è una matrice diagonale $r \prod r$ di valori singolari dove $r = \min(t, N)$ è il rango di M. Solo gli s massimi valori singolari di S sono mantenuti nelle rispettive colonne in K e D^t (ovvero

i rimanenti valori singolari di S vengono eliminati). La matrice risultante M_s è la matrice di rango s più vicina alla matrice originale M nel senso dei minimi quadrati. Questa matrice è data da

$$M_s = K_s S_s D_s^t$$

dove s, $s < r$ è la dimensionalità dello spazio concettuale ridotto. La selezione del valore per s deve essere effettuata cercando di bilanciare due effetti opposti. Innanzitutto, s dovrebbe essere grande abbastanza per permettere di riempire tutta la struttura con i dati reali. In secondo luogo, s dovrebbe essere sufficientemente piccolo per escludere tutti i dettagli non rilevanti delle rappresentazioni (che sono presenti nella rappresentazione convenzionale basata sui termini indice). La relazione tra qualsiasi due documenti nello spazio a dimensionalità ridotta s può pertanto essere ottenuto dalla matrice $M_s M_s^t$ nel modo seguente:

$$M_s^t M_s = (K_s S_s D_s^t)^t K_s S_s D_s^t = D_s S_s K_s^t K_s S_s D_s^t = D_s S_s S_s D_s^t = (D_s S_s)(D_s S_s)^t$$

Nella matrice sopra descritta, l'elemento (i, j) quantifica la relazione tra il documento d_i ed il documento d_j. Per classificare i documenti in base ad una certa query dell'utente, è sufficiente modellare la query come un *pseudo-documento* nella matrice originale M (termini-documenti). Si supponga che la query è modellata come documento numero 0. Allora, la prima riga della matrice $M_s^t M_s$ fornisce la classificazione di tutti i documenti rispetto a questa query. Siccome le matrici utilizzate nel modello LSI sono di rango s, con $s << t$, e $s << N$, allora esse costituiscono un efficiente schema di indicizzazione per i documenti nella collezione. Inoltre, consentono l'eliminazione del rumore (presente nelle rappresentazioni basate su termini indice) e la rimozione della ridondanza. Il modello LSI introduce una interessante concettualizzazione del problema dell'Information Retrieval basato sulla teoria

della decomposizione a valore singolo. Tuttavia il fatto che possa dimostrarsi più efficace di altri modelli nelle applicazioni pratiche rimane da dimostrare.

2.1.7 K-Means Clustering

Nell'ambito dell'Information Retrieval si definisce cluster un gruppo omogeneo di documenti, che sono tra loro più fortemente associati di quanto lo siano con documenti di altri gruppi, quindi un processo di clustering si può vedere come l'organizzazione degli oggetti in gruppi i cui membri sono simili. Prima di decrivere la tecnica di K-Means Clustering [15] è opportuno citare l'enunciato che sta alla base dell'utilizzo delle tecniche di clustering e che in letteratura viene indicato come *Cluster Hypothesis*: i documenti che sono simili l'uno con l'altro tendono ad essere rilevanti per le medesime query. Nelle tecniche di clustering viene assegnato ad ogni cluster un elemento, detto *centroide*, che rappresenta in qualche modo la media dei documenti appartenenti a un determinato cluster. Il centroide può essere considerato come il documento più "tipico" tra quelli appartenenti al suo stesso cluster, ovvero quello che esprime meglio la caratterizzazione del gruppo di documenti cui appartiene. Invece di confrontare la query con tutti i documenti della collezione la si confronta con i vari centroidi e come risultato si ha il cluster od i cluster che risultano essere i più simili alla query. Per organizzare i documenti in clustering bisogna definire una misura di similarità tra i documenti e definire i metodi per suddividere in classi quest'ultimi.

Una tecnica di clustering è *K-Means Clustering* che utilizza l'algoritmo *Bisecting-Spherical K-Means*. L'algoritmo Bisecting-Spherical K-Means cerca di trovare k cluster diversi $\{\pi_j\}_{j=1}$ dalla collezione dei documenti espressa mediante la matrice M in modo da massimizzare la seguente funzione

obiettivo:

$$f\left(\{\pi_j\}_{j=1}^{k}\right) = \sum_{j=1}^{k} \sum_{m \in \pi_j} m^t c_j$$

dove c_j è il centroide normalizzato o il vettore dei concetti del cluster π_j, che è calcolato dalla seguente espressione

$$t_j = \frac{1}{n_j} \sum_{m \in \pi_j} m; \qquad c_j = \frac{t_j}{\|t_j\|} \qquad (1)$$

dove n_j è il numero dei documenti nel cluster π_j. L'algoritmo Bisecting-Spherical K-Means è costituito da quattro passi:

1. Calcola il cluster iniziale k con l'algoritmo di Bisection K-Means: $\left\{\pi_j^{(0)}\right\}_{j=1}^{k}$ e i vettori dei concetti $\left\{c_j^{(0)}\right\}_{j=1}^{k}$. Inizializza $t = 0$.

2. Calcola la nuova partizione $\left\{\pi_j^{(t+1)}\right\}_{j=1}^{k}$ indottta dal vettore dei concetti $\left\{c_j^{(t)}\right\}_{j=1}^{k}$: $\pi_j^{t+1} = \left\{m \in \{m_j\}_{i=1}^{n} : m^T c_j^{(t)} > m^T c_l^{(t)}, 1 \leq l \leq n, l \neq j\right\}$, con $1 \leq j \leq k$.

3. Calcola i vettori dei concetti associati ai nuovi clusters $\left\{c_j^{(t+1)}\right\}_{j=1}^{k}$, usando l'espressione (1).

4. Quando il criterio di stop è soddisfatto, ritornare $\left\{\pi_j^{(t+1)}\right\}_{j=1}^{k}$ e $\left\{c_j^{(t+1)}\right\}_{j=1}^{k}$. Altrimenti incrementa t $(t = t + 1)$ e torna al passo 2.

Il criterio di stop (errore relativo) usato per l'algoritmo è il seguente:

$$\frac{\left| f\left(\left\{\pi_j^{(t+1)}\right\}_{j=1}^{k}\right) - f\left(\left\{\pi_j^{(t+1)}\right\}_{j=1}^{k}\right)\right|}{\left| f\left(\left\{\pi_j^{(t+1)}\right\}_{j=1}^{k}\right)\right|} \leq \epsilon = 1 \times 10^{-2}$$

2.2 Preprocessing del testo

Il preprocessing consiste di un processo, che precede la fase elaborazione delle queries, per ottimizzare la lista dei termini, che identificano una collezione [1]. Questa ottimizzazione può essere ottenuta eliminando quei termini che sono poveri di informazione. Durante questa fase possono essere eseguite tutte o parte delle seguenti operazioni sul testo:

1. analisi lessicale del testo;
2. eliminazione delle *stopwords*;
3. *stemming* delle parole rimanenti;
4. selezione dei termini indice.

2.2.1 Analisi lessicale del testo

L'analisi lessicale è il processo di convertire la serie di caratteri formanti il testo del documento in una serie di parole candidate per essere termini indice. Così uno dei maggior obiettivi della fase dell'analisi lessicale è l'identificazione delle parole nel testo. Questo avviene attraverso il trattamento della punteggiatura, delle cifre, e delle lettere maiuscole e minuscole.

2.2.2 Eliminazione delle Stopwords

L'eliminazione delle Stopwords ha come obiettivo la selezione delle parole, attraverso la rimozione di quelle considerate inutili per il recupero. Di solito le parole candidate ad essere Stopwords sono gli articoli, le preposizioni e le congiunzioni. Un vantaggio dell'eliminazione delle Stopwords è che consente di ridurre la dimensione della struttura di indicizzazione. La lista di

stopwords può essere estesa includendo, oltre agli articoli, alle preposizioni e congiunzioni, anche alcuni verbi, avverbi ed aggettivi di uso frequente, anche se questo potrebbe ridurre il Recall. Per esempio, pensiamo ad un utente che voglia cercare i documenti contenenti la frase *'to be or not to be'*: l'eliminazione delle Stopwords porta a considerare solo il termine *be*, rendendo quasi impossibile il riconoscimento dei documenti che contengono la frase esatta. Questo è uno dei motivi per cui alcuni motori di ricerca preferiscono inserire tutte le parole del documento come termine indice.

2.2.3 Stemming

Al fine di considerare parole che hanno varianti morfologiche diverse come una unica parola, ma con simile interpretazione semantica, è utile l'operazione di stemming. Questa sostituisce ad una parola il suo rispettivo *stem*. Uno stem è la porzione di parola ottenuta rimovendo prefissi e sufissi. Un esempio tipico è la parola *connect* che è lo stem per le sue varianti *connected, connecting, connection* e *connections*, o *mang* che è lo stem per *mangiare, mangio, mangiata, mangi* etc. Si distinguono quattro tipi di strategie di stemming: *affix removal*, *table lookup*, *successor variety* e *n-grams*. *Table lookup* consiste nel cercare lo stem di una parola in una tavola, questa è una semplice procedura, ma dipende pesantemente dalle statistiche collezionate per gli stem di un certo linguaggio. Dal momento che i dati possono non essere disponibili subito e possono richiedere uno spazio considerevole di memoria, questo tipo di stemming non risulta molto pratico. *Successor variety stemming* è basato sulla determinazione dei limiti dei morfemi (stem, suffissi e prefissi, ovvero le parti di parola che non possono essere ulteriormente scomposte, vengono dette *morfemi*) ed usa la conoscenza delle strutture linguistiche.

N-grams stemming è basato sull'identificazione dei bigrammi e trigrammi (ovvero sequenze di due o tre parole). *Affix removal stemming* è invece il tipo di stemming più semplice ed intuitivo e può essere implementato facilmente. In affix removal prende importanza la rimozione dei suffissi perchè la maggior parte delle varianti di una parola sono costituite dall'introduzione di suffissi invece di prefissi. Esistono dei algoritmi conosciuti per la rimozione dei suffissi tra cui il più popolare per la sua semlicità e per la sua eleganza è l'algoritmo di Porter. L'algoritmo di Porter[14] si basa sull'idea di applicare una serie di regole ai suffissi delle parole nel testo. Per esempio la regola $s \longrightarrow \phi$ viene usata per convertire le forme plurali nelle rispettive forme singolari togliendo la lettera s. Queste regole sono separate in cinque distinte fasi numerate da uno a cinque, e vengono applicate al testo sequenzialmente una dopo l'altra attraverso l'uso di comandi. Lo stemming è utile per migliorare la performance di un sistema di Information Retrieval, in quanto riduce le varianti di una stessa parola-radice ad un concetto comune. Inoltre questa operazione presenta il vantaggio di ridurre il numero dei termini indici che identificano una collezione, con la conseguente riduzione della dimensione della struttura di indicizzazione. Nonostante questi vantaggi, in Information Retrieval ci sono state delle dispute riguardo ai benefici dello stemming sulle performance del recupero, risultati di diversi studi hanno portato conclusioni contrastanti sull'uso dello stemming [1], anche se attualmente sembra essere quello che dà i risultati migliori.

2.2.4 Selezione dei termini indice

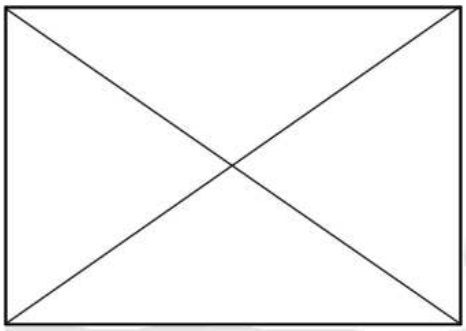
Dato un insieme di termini per un documento, non tutti i termini sono ugualmente utili per descrivere il contenuto del documento. Infatti vi sono dei

termini che possono meno rappresentativi di altri. Decidere sull'importanza di un termine per riassumere il contenuto di un documento non è un problema banale. Una tecnica molto semplice consiste nel selezionare le parole più frequenti all'interno del documento, anche se il contrario, ovvero che le parole meno frequenti siano meno importanti, non è sempre vero: infatti, una parola fondamentale per il contenuto del documento potrebbe comparire esclusivamente nel titolo e non essere più ripetuta all'interno del documento. Ad esempio, nell'articolo mostrato in fig. 2.6, intitolato *When did WMD deals between Islamabad and Pyongyang begin?*, una parola fondamentale (*WMD*, acronimo di *Weapons of Mass Destruction*, tristemente salito alla ribalta a causa della guerra in Iraq) per identificare il contenuto compare unicamente nel titolo.

When Did WMD Deals between Pyongyang and Islamabad Begin?

By Daniel A. Pinkston

According to press reports, the North Korean-Pakistani trade of missiles for highly enriched uranium (HEU) technology occurred around 1997,[2] which would coincide with Pyongyang's shipment of Nodong missiles to Pakistan. The *Washington Post* quoted "Washington sources" as having said that the first signs of Pyongyang's HEU program emerged in 2000, but that "the 'dots weren't connected' by intelligence analysts until the summer of that year."[3] Such reports have led many to speculate that the "missiles for HEU equipment" deal occurred years after North Korea signed the Agreed Framework in 1994, but we cannot be certain this is the case. Nuclear development programs have long time horizons because of their complexity and technical difficulty. Political decisions must precede the engineering work; and in the case of a "missiles for HEU equipment" transaction, the negotiations and agreement over the complex contractual details would have to precede the transaction.



North Korea's Academy of Sciences in Pyongyang

Figura 2.6: Un articolo estratto dal Web.

2.2.5 Lemmatizzazione e Lemmatizzazione Semantica

Il processo di lemmatizzazione consiste nel ricondurre una parola in una forma flessa o coniugata nel *lemma* corrispondente: il *lemma* è una forma "canonica" della parola, che può essere trovata in un dizionario. I lemmi corrispondenti ai nomi hanno la forma singolare (ad esempio, il lemma di *sedie* è *sedia*), quelli corrispondenti agli aggettivi hanno la forma singolare maschile (ad esempio, il lemma di *belle* è *bello*), e quelli corrispondenti ai verbi hanno la forma dell'infinito del verbo (ad esempio, il lemma di *mangio* è *mangiare*).

Se il dizionario utilizzato contiene anche informazioni semantiche (come l'ontologia WordNet descritta nel capitolo seguente), è possibile passare dal lemma ad una rappresentazione semantica, che in WordNet viene chiamata *Synset*. Il passaggio dal lemma alla semantica non è semplice, per via del problema della *polisemia*. Per esempio la parola *calcio* può avere più significati: lo sport, l'elemento chimico, la parte di un'arma o un'azione compiuta calciando. L'unico modo per consentire a ciascuna parola un unico significato consiste nell'effettuare la *disambiguazione semantica*, od, in inglese, *word sense disambiguation*.

2.3 Indicizzazione: Inverted Files

Per indicizzazione s'intende l'attività di individuazione di keyword. Le principali tecniche di indicizzazione sono *inverted file*, *suffix array* e *signature files*. In questa sezione viene descritta la tecnica *inverted file*, in quanto si tratta di quella più diffusa e più utilizzata nell'ambito di IR. *Un inverted file* o *inverted index* è un meccanismo orientato alle parole per indicizzare

una collezione di testo con lo scopo di velocizzare l'operazione di ricerca. La struttura dell'inverted file è composta da due elementi: il *vocabolario* e le *occorenze*. Il vocabolario è l'insieme di tutte le parole nel testo e per ognuna di queste viene conservata una lista di tutte le posizioni nel testo dove essa compare. L'insieme di tutte queste liste viene chiamato occorenze. Le posizioni possono far riferimento a caratteri o a parole. Le posizioni relative alle parole semplificano le query, mentre quelle relative ai caratteri facilitano l'accesso diretto alle posizioni corrispondenti nel testo. La figura 2.7 mostra un esempio di inverted index, ed in particolare la differenza tra la descrizione delle occorrenze come posizione della parola nella frase e posizione del primo carattere della parola nella stringa. In questo secondo caso l'informazione può essere utilizzata per accedere più velocemente alle parole.

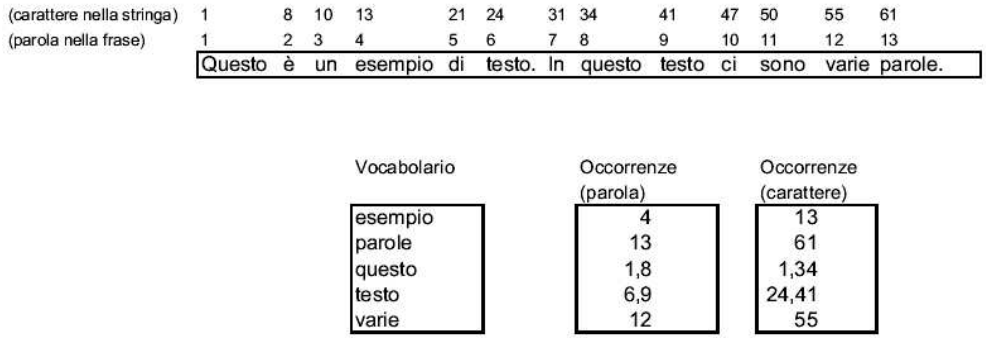


Figura 2.7: Un esempio di inverted index [1].

Capitolo 3

Valutazione dei sistemi di IR

Un sistema di IR viene solitamente valutato in funzione della compatibilità dell'insieme delle risposte con le richieste dell'utente (*efficacia*). Questo è sostanzialmente diverso da quello che accade nel caso della valutazione dei sistemi di Data Retrieval, dove l'enfasi è posta sull'*efficienza*, ovvero la velocità nel restituire le risposte e la quantità di spazio occupato. Valutare l'efficacia non è un problema semplice, in quanto include diversi aspetti soggettivi, per esempio due utenti con diversi livelli di conoscenza, formulando la stessa richiesta, potrebbero ottenere due diverse valutazioni sull'insieme di documenti reperiti dal sistema.

Una valutazione solitamente è basata su una collezione di test, che consiste in una collezione di documenti, un insieme di esempi di richieste di informazione e per ciascuna di esse una lista di documenti rilevanti, fornita da appositi specialisti. La misura dell'efficacia quantifica, per ogni esempio di richiesta di informazione, la somiglianza tra l'insieme di documenti recuperati dal sistema e la lista di documenti rilevanti. Questo permette di dare una stima della bontà di un sistema e quindi della sua strategia utilizzata

per il recupero di informazioni.

3.1 Misure di efficacia: Precisione e Recall

Si consideri una richiesta di informazione I e il suo insieme R di documenti rilevanti. Sia $|R|$ il numero di documenti in questo insieme. Si assuma che una data strategia di recupero (che si sta valutando) elabori la richiesta di informazioni I e generi un insieme di documenti di risposta A. Sia $|A|$ il numero di documenti di A. Inoltre, sia $|R_a|$ il numero dei documenti nella intersezione degli insiemi R ed A (fig.3.1).

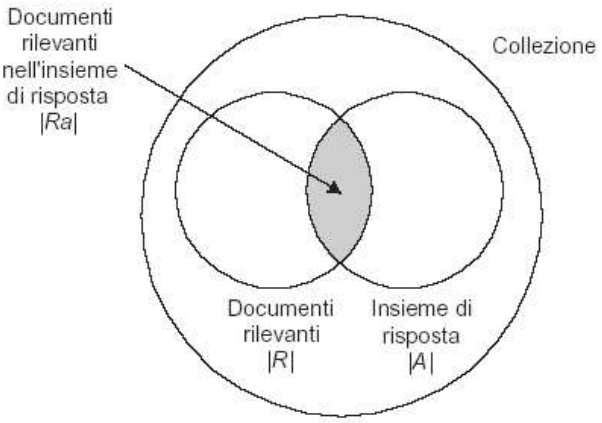


Figura 3.1: Precisione e Recall per una data richiesta di informazione

Precisione e Recall sono definite nel modo seguente:

- *Precisione* è la frazione di documenti recuperati (insieme A) che è rilevante cioè

$$Precisione = \frac{|R_a|}{A}$$

- *Recall* è la frazione di documenti rilevanti che è stata recuperata cioè

$$Recall = \frac{|R_a|}{R}$$

Ha senso tenere in considerazione il Recall solo quando la collezione di documenti è finita; infatti nel caso di IR su web è impossibile sapere il numero totale di documenti rilevanti, pertanto la valutazione dei web search engines si basa più sulla Precisione (in particolare per via del ranking, data l'importanza di presentare i risultati rilevanti in cima alla lista).

3.2 Il grafico Precisione-Recall

Precisione e Recall assumono che siano stati esaminati tutti i documenti nell'insieme di risposta A. Tuttavia all'utente non viene solitamente mostrato tutto l'insieme dei documenti in una sola volta. I documenti in A vengono prima ordinati in base al grado di rilevanza, quindi l'utente esamina la lista ordinata iniziando dall'alto. In questa situazione le misure Precisione e Recall cambiano a mano a mano che l'utente procede con l'esame dell'insieme delle risposte A. Pertanto, per una corretta valutazione è necessario disegnare il *grafico Precisione-Recall*, ottenuto, appunto, scorrendo la lista ordinata dei documenti ritornati dal sistema.

Per esempio, si assuma che l'insieme, detto R_q, dei documenti rilevanti per la query q sia il seguente:

$$R_q = \{d_3, d_5, d_9, d_{25}, d_{39}, d_{44}, d_{56}, d_{71}, d_{89}, d_{123}\}$$

e che il sistema da valutare ritorni, per la query q, il seguente elenco ordinato di documenti:

1. d_{123}♣	6. d_9♣	11. d_{38}
2. d_{84}	7. d_{511}	12. d_{48}
3. d_{56}♣	8. d_{129}	13. d_{250}
4. d_6	9. d_{187}	14. d_{113}
5. d_8	10. d_{25}♣	15. d_3♣

I documenti rilevanti alla query q sono evidenziati con il simbolo ♣.
Il documento d_{123} occupa la prima posizione nella lista dei documenti recuperati, è rilevante ed inoltre corrisponde al 10% di tutti i documenti rilevanti. Pertanto si avrà una Precisione del 100% ad un Recall del 10%. Il successivo documento rilevante è il d_{56}, che si trova in terza posizione. A questo punto si ha una Precisione del 66% (due documenti su tre esaminati sono rilevanti) circa ad un Recall del 20% (sono stati considerati due dei dieci documenti rilevanti). Procedendo in questo modo si ottiene il grafico Precisione-Recall mostrato in figura (fig.3.2). La Precisione a livelli di Recall più alti del 50% scende a 0 perchè non tutti i documenti rilevanti sono stati recuperati. Il diagramma Precisione-Recall di solito è basato su 11 (invece che 10) livelli *standard* di Recall, ovvero: 0%, 10%, 20%,.....,100%.

Nell'esempio sopra riportato il diagramma Precisione-Recall è relativo a una singola query, tuttavia i sistemi di IR vengono generalmente valutati su un insieme di queries. Per valutare un sistema di IR su tutte le queries di test, si calcola la media dei valori di Precisione ad ogni livello di Recall attraverso la seguente formula:

$$\overline{P}(r) = \sum_{i=1}^{N_q} \frac{P_i(r)}{N_q}$$

dove $\overline{P}(r)$ è la Precisione media al livello r di Recall, N_q è il numero di query utilizzate e $P_i(r)$ è la Precisione al livello r di Recall per la i-esima query.

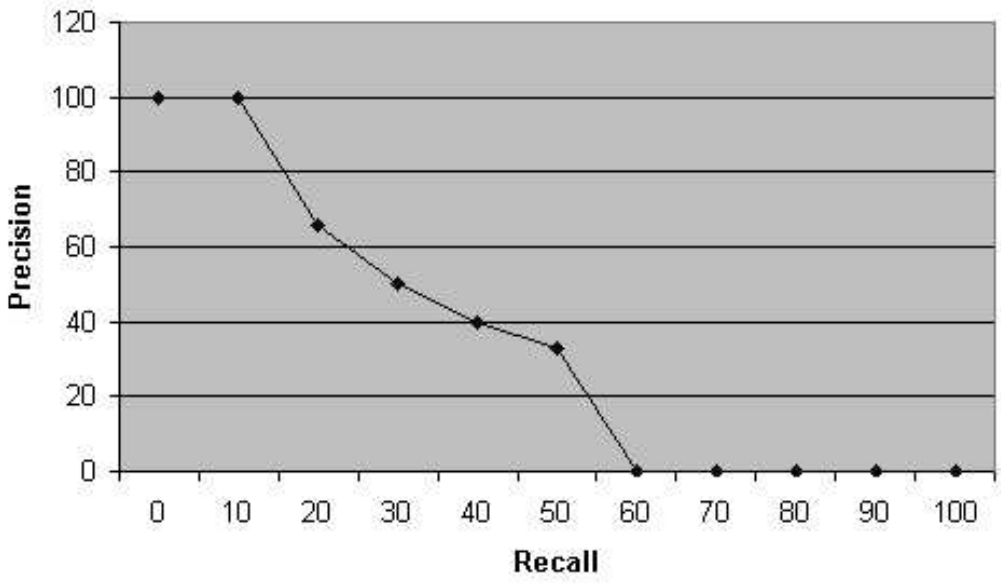


Figura 3.2: Grafico Precisione/Recall risultante dall'esempio

Al livello di Recall 0% la Precisione è ottenuta attraverso una procedura di interpolazione. Questa è utilizzata anche per quei livelli di Recall per cui non è possibile valutare la Precisione (ad esempio, se il primo documento rilevante della lista è il terzo, non si hanno informazioni di Precisione per livelli di Recall inferiori al 33.3%, è necessario usare l'interpolazione per assegnare i valori anche ai livelli di Recall precedenti: 0, 10, 20 e 30%). La procedura di interpolazione consiste semplicemente nell'assegnare al livello di Recall j la precisione massima tra il livello di Recall j e $j+1$:

$$P(r_j) = \max_{r_j <= r <= r_{j+1}} P(r)$$

3.3 Collezioni di test

Negli anni sono state usate parecchie collezioni per la valutazione di sistemi di IR, quelle che vale la pena menzionare per loro importanza storica sono *CACM* e *ISI*, mentre quella che oggi è considerata di riferimento per l'IR e che verrà presentata in questo capitolo, perchè è stata oggetto di esperimenti per questa tesi, è la collezione *TREC*. L'acronimo TREC sta per *Text REtrieval Conference* e consiste di una serie di conferenze sponsorizzate dal *National Institute of Standards and Technology (NIST)*[1] e dal *Information Technology Office of the Defence Advanced Research Projects Agency (DARPA)*[21]. Lo scopo di queste conferenze sono i seguenti:

- incorraggiare la ricerca al text retrieval basati su grandi collezioni;
- incrementare la comunicazione tra le industrie, l'università e il governo, creando un forum aperto per lo scambio di idee;
- velocizzare il trasferimento della tecnologia dai laboratori di ricerca in prodotti commerciali, dimostrando sostanzialmente i miglioramenti delle metodologie di fronte a problemi del mondo reale;
- incrementare la disponibilità di appropriate tecniche di valutazione, usate dall'industria e l' università, sviluppandone delle nuove applicabili ai sistemi moderni.

La diversità di gruppi di partecipanti alla conferenza fa sì che ci sia una grande varietà di tecniche ed approcci all'IR, tutti valutati attraverso un metodo comune, in modo da permettere una comparazione. La prima conferenza del TREC è stata tenuta nel Novembre del 1992. La collezione TREC

[1] http://trec.nist.gov

è composta da tre parti: i documenti, i *topics* ed i *giudizi di rilevanza*). I documenti sono in formato SGML per agevolare il parsing (vedi fig.3.3). Tutti

```
<DOC>
<DOCNO> FT911-3 </DOCNO>
<DATE> 910514 </DATE>
<HEADLINE\>
FT 14 MAY 91 / International Company News: Contiguas plans DM900m east German project
</HEADLINE>
<BYLINE> By David Goodhart </BYLINE>
<DATELINE> Bonn </DATELINE>
<TEXT>
 Contigas, the German gas group 81 per cent owned by the utility Bayernwerk, said yesterday that it
intends to invest DM900m (Dollars 522m) in the next four yers to buid a new gas distribution
system in the east German state of Thuringia. .....
</TEXT>
</DOC>
```

Figura 3.3: Un esempio di documento del TREC

i documenti iniziano con il tag radice `<DOC>`, hanno il tag `<DOCNO>` che indica il numero di documento ed infine documento vero e proprio è contenuto all'interno dei tag `<TEXT>` e `</TEXT>`. La filosofia del NIST che sta dietro a questa formattazione dei documenti è di lasciare i dati il più possibile vicini a quelli originali. I topics rappresentano le richieste di informazione dell'utente, vengono scritti da utenti esperti ed hanno una struttura semplice comprendente tre campi che sono: *Title*, *Description* e *Narrative*. Il campo Title consiste di tre parole che meglio descrivono il topic ed è stato progettato per fare esperimenti con query molto corte, il campo Description è una frase che descrive l'argomento del topic ed infine il campo Narrative dà una concisa descrizione di ciò che rende un documento rilevante. La figura 3.4 mostra un esempio di topic. I giudizi di rilevanza sono di importanza critica per il test collection e rappresentano per ogni topic la lista di documenti rilevanti. Il TREC usa un metodo chiamato *pooling method* per mettere insieme i giudizi di rilevanza. Questo metodo consiste nel creare un gruppo di possibili docu-

<num> Number: 409
<title> legal, Pan Am,103
<desc>Description:
What legal actions have resulted from the destruction of Pan Am Flight 103 over Lockerbie, Scotland, on December 21, 1988
<narr> Narrative:
Documents describing any charges, claims, or fines presented to or imposed by any court or tribunal are rilevant, but documents that discuss charges made in diplomatic jousting are not rilevant.

Figura 3.4: Un esempio di topic del TREC

menti rilevanti, prendendo un campione di documenti selezionati dai sistemi partecipanti al TREC. Il campione è costituito dai primi K documenti (solitamente $K = 100$) della graduatoria ottenuta dalla combinazione dei risultati ottenuti dai diversi sistemi. Questo gruppo viene poi mostrato a dei giudici, che stabiliscono la rilevanza di ogni documento.

Capitolo 4

Risorse e Tools

In questo capitolo verranno esaminate le risorse ed i tools utilizzati nell'ambito di questa tesi, passando brevemente per Smart, il sistema che abbiamo provato ad utilizzare inizialmente, ma che in seguito è stato abbandonato in favore di Furbo. Questa scelta è dipesa dal fatto che avremmo dovuto cambiare parte dell'implementazione per poter efettuare gli esperimenti con le diverse tecniche di indicizzazione, cosa che non è risultata per niente facile vista l'obsolescenza di Smart. In seguito verranno descritte le risorse utilizzate per il text processing, in particolare WordNet, un POS Tagger, ed il disambiguatore semantico basato su densità concettuale e frequenza.

4.1 Smart

Smart è un sistema di IR sviluppato originalmente all'università di Harvard tra il 1962 ed il 1965 [19]. Questo sistema è una implementazione del modello vettoriale proposto da Salton nel 1960. Lo scopo principale per cui Smart è stato creato è quello di fornire un framework in cui si potesse condurre la

ricerca in IR, ed è stato concepito in particolare per tre tipologie di utenti: gli esperimentatori, che necessitano dell'abilità di cambiare facilmente i parametri ed aggiungere o modificare i moduli del sistema, gli amministratori di database che hanno bisogno di creare e gestire una collezione di documenti senza lavorare sulle peculiarietà di una particolare collezione e gli utenti finali che devono inserire delle query e vedere i risultati ottenuti dal sistema senza conoscere il funzionamento interno del sistema. Il nucleo di Smart è costituito da una serie di procedure progettate per permettere un efficiente accesso alla collezione ed un efficiente recupero. La versione standard è fornita di procedure di indexing, di recupero e di valutazione.

4.2 Furbo

Furbo[1] è stato sviluppato all'Università Politecnica di Valencia (UPV) nel 2003, ed è nato con lo scopo di realizzare un'implementazione più semplice e più veloce del sistema Smart, seguendo le linee guida dell'*agile programming*. Il sistema di Information Retrieval Furbo è costituito da una collezione di moduli scritti in Python[2]. In figura 4.1 è mostrata l'architettura del sistema. Un primo modulo di parsing si occupa di esaminare una collezione di documenti (in formato XML) ed estrarre per ciascun documento una lista di termini con un peso (generalmente la frequenza). I termini possono essere le parole, gli stem, i lemmi o gli offset dell'ontologia di WordNet, che descriveremo in questo stesso capitolo. Queste ultime tre caratteristiche sono state aggiunte al sistema originale, utilizzando per lo stemming la libreria *PyStemmer.py*, che implementa uno stemmer basato sull'algoritmo di Porter, e per

[1] da Matteo Dell'Amico (studente del DISI, stage Erasmus presso la UPV)

[2] http://www.python.org

l'accesso a WordNet la libreria PyWordnet.py, un porting in python della libreria in C fornita con WordNet. La lista dei termini viene salvata in un file temporaneo, mentre le stopwords vengono salvate in un file nominato common_words. Il file temporaneo è formattato nel modo seguente:

```
NomeDoc1
termine peso
NomeDoc2
```

etc.

Questo file temporaneo viene preso come input dal modulo *create.py* che si occupa di creare un indice incompleto, il cui completamento avviene attraverso il successivo modulo *strategize.py*, che permette di utilizzare i vari modelli (sono implementati quello booleano, vettoriale e probabilistico) con il database creato al passo precedente. Strategize.py quindi permette di scegliere la strategia per i pesi dei dati e per indicizzare le keywords. Le strategie sono le stesse che sono presenti in Smart e sono eseguite in un processo a tre passi. Il primo passo si occupa di normalizzare la frequenza dei termini, il secondo cambia i pesi delle parole nei documenti secondo la loro frequenza in tutta la collezione ed infine il terzo normalizza l'intero vettore. Ogni passo è codificato da tre diverse lettere (si riportano i casi più utilizzati):

1. n (nessuna): nessuna conversione, $tf_{new} = tf$;

   b (booleana): passa dalla frequenza del termine a 1 (perchè la parola è stata trovata), $tf_{new} = 1$;

   a (normalizzazione aumentata): $tf_{new} = 0.5 + 0.5 * (tf/tf_{max})$, in questo modo si ha sempre $0.5 < tf_{new} \leq 1$;

   s (quadrata): $tf_{new} = tf^2$.

2. n (nessuna): nessuna conversione, $wt_{new} = tf_{new}$;

t (tf-idf): $wt_{new} = tf_{new} \log(ndocs/tf_{collection})$;

3. n (nessuna): nessuna conversione, $wt_{normalized} = wt_{new}$;

   c (coseno): si divide ciascun wt_{new} per la radice della somma di tutti i nuovi wt.

Dal punto di vista dell'utente, la ricerca viene eseguita passando direttamente una query al modulo *get.py*, nel caso non sia necessario effettuare preventivamente operazioni testuali sulla query. In caso contrario, il modulo *Query Processor* si occupa di varie operazioni come stemming, lemmatizzazione, "synsetizzazione" ed espansione della query.

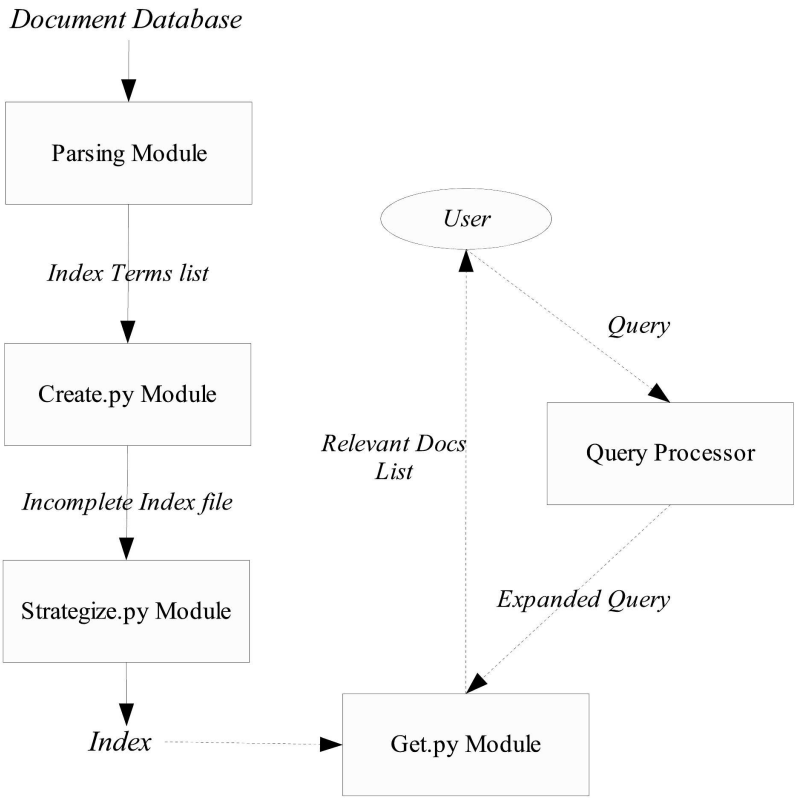


Figura 4.1: Architettura di Furbo IR system

4.3 Risorse per il text processing

Per poter effettuare la lemmatizzazione e la lemmatizzazione semantica è necessario fare ricorso ad alcune risorse utilizzate nel campo dell'elaborazione del linguaggio naturale. In particolare, il Part-Of-Speech tagger permette di stabilire la categoria delle parole in modo da rendere possibile la ricerca della parola in un apposito thesauro, per estrarre i sinonimi, od in una ontologia, per estrarre una rappresentazione semantica della parola e le sue relazioni con altri concetti. Queste relazioni, infine, vengono utilizzate sia per effettuare alcuni tipi di elaborazioni sul testo come l'espansione delle query, sia nell'ambito di sistemi di disambiguazione non supervisionati, come il disambiguatore semantico basato su densità concettuale descritto alla fine di questa sezione.

4.3.1 Part-Of-Speech Tagger

Nei sistemi di elaborazione del linguaggio naturale esiste una fase chiamata *Part-Of-Speech tagging* o *POS-tagging* (in italiano la traduzione sarebbe "etichettatura delle parti del discorso"). In questa fase si etichettano le parole individuate nella fase di analisi lessicale con il POS corrispondente, senza eseguire una vera e propria analisi sintattica, ma ricorrendo in genere ad informazioni statistiche (ad esempio il TnT tagger [2], basato su trigrammi) od a regole. Le etichette, o *tags*, sono reperite attraverso *tagset*, mappe da categorie lessicali in tags, che oltre a contenere i tag per le otto categorie lessicali di base, includono delle specializzazioni o raffinamenti, che distinguono delle sottocategorie, spesso in base al significato della parola. Nella tabella 4.1 è riportato un estratto dal tagset utilizzato nel Penn Treebank corpus [12], il primo corpus etichettato sintatticamente. Un esempio di POS-tagging è il seguente: data la frase *book that flight* (prenota quel volo), le etichettature

Tag	Descrizione	Esempio	Tag	Descrizione	Esempio
CC	congiunzione	*and*, *or*	PP	pronome personale	*I*, *you*
CD	numero	*one*	PP$	pronome possessivo	*my*, *your*
DT	articolo	*the*, *a*	RB	avverbio	*fully*
FW	parole straniere	*coup*	SYM	simbolo	+, &
IN	preposizione	*of*, *by*	TO	*to* finale	*to*
JJ	aggettivo	*white*, *big*	UH	interiezione	*oops*!
JJR	aggettivo comparativo	*bigger*	VB	verbo, forma base	*eat*
JJS	superlativo	*biggest*	VBD	verbo, passato	*ate*
MD	verbo modale	*can*	VBG	verbo, *-ing* form	*eating*
NN	nome	*cat*	VBZ	verbo, 3^a persona pres.	*eats*
NNS	nome,plurale	*cats*	WDT	pronome determinativo	*which*, *that*
NNP	nome proprio	*George*	WP	pronome *Wh-*	*what*, *who*
POS	genitivo sassone	*'s*	WRB	avverbio *Wh-*	*how*, *where*

Tabella 4.1: Un sottoinsieme del Penn Treebank Tagset

possibili, considerando il tagset del Penn Treebank corpus, sono: *book:NN that:WDT flight:NN* oppure *book:VB that:WDT flight:NN*, per via dell' ambiguità della parola *book*. In questo caso l'etichettatura corretta è quella che assegna a book il tag *VB*. Gli approcci al POS-tagging sono principalmente due:

- *rule-based*: si usa un dizionario per assegnare a ciascuna parola una lista di possibili tag. Quindi si applicano regole (ad esempio "se una parola ambigua segue un articolo determinativo, allora è un nome") per eliminare i tag non consistenti.

- *statistico*: si usa un corpus per calcolare la probabilità di un tag in una certa sequenza di parole, scegliendo il tag con probabilità massima

Un esempio di POS-tagger statistico è il tagger basato su bigrammi: data una parola ambigua w_i con un insieme J di possibili tag, il tagger seleziona come tag t_i quello più probabile considerando il tag precedente (t_{i-1}):

$$t_i = \arg\max_{jinJ} P(t_j|t_{i-1}, w_i)$$

La precisione dei migliori pos-tagger statistici si aggira intorno al 95%, cosa che ha permesso, in alcune applicazioni, lo snellimento della fase di analisi sintattica, se non addirittura l'eliminazione della fase stessa (ad esempio, per determinare se un documento parla di sport o di economia, può essere sufficiente considerare solamente i nomi che compaiono nel documento, senza bisogno di analizzare sintatticamente le frasi che lo compongono).

Nel corso di questo lavoro, è stato utilizzato il POS-tagger basato su regole realizzato da Eric Brill [3].

4.3.2 WordNet

WordNet è una risorsa lessicale, sviluppata all'Università di Princeton [13]. Più precisamente si tratta di un'ontologia, ovvero di un particolare dizionario che fornisce un elenco di concetti, relazioni ed assiomi che formalizzano un certo dominio o campo di interesse. Di per sè la parola *ontologia* significa "una particolare teoria della natura dell'essere o dell'esistenza, il motivo per cui le ontologie spesso non si limitano a dare un mero elenco di concetti o assiomi associati ad un particolare dominio, ma descrivono anche una struttura gerarchica in cui questi concetti sono organizzati.

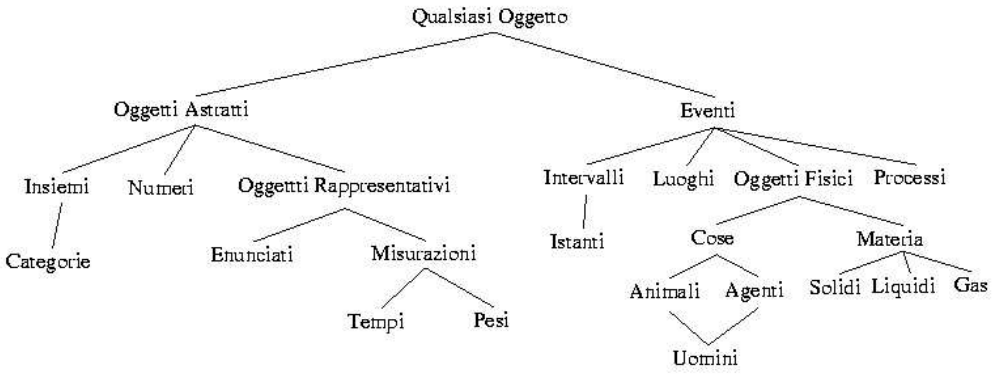


Figura 4.2: Esempio di ontologia generale proposta da Russell e Norvig[18]

Un'ontologia generale come quella in fig. 4.2 si limita a concetti che sono meta, generici, astratti o filosofici, e pertanto sufficientemente generali per essere utilizzati in un ampio spettro di applicazioni. Le ontologie generali non includono concetti specifici a particolari domini, ma tali ontologie forniscono una struttura su cui ontologie per domini specifici (come ad esempio medicina, finanza, scienza) possono essere realizzate.

WordNet è un'ontologia che include anche la funzionalità di thesaurus, ovvero un dizionario che associa ad ogni parola le parole con il medesimo

significato, ovvero i sinonimi. E' l'unica risorsa lessicale machine-readable di questo tipo disponibile gratuitamente, e copre, nella sua ultima versione (2.0), più di 150000 parole tra nomi, verbi, aggettivi e avverbi della lingua inglese. Altre versioni di WordNet, dedicate ad altre lingue europee, sono state sviluppate sotto il nome di EuroWordNet[3].

L'elemento fondamentale di WordNet è il *synset*, ovvero un "insieme di sinonimi" (dall'inglese *set of synonyms*) che descrivono il medesimo concetto. Un synset può essere descritto elencando le parole che lo compongono (ad esempio {*movie, film, picture, moving picture, moving-picture show, motion picture, motion-picture show, picture show, pic, flick*}), o attraverso una coppia *(POS, offset)*, che identifica univocamente la posizione del synset all'interno dei file di dati di WordNet per il POS corrispondente (in questo caso il synset dell'esempio precedente è il *(NOUN, 6205452)*). Si può pensare ad un synset come alla rappresentazione astratta di un significato di una parola. Pertanto a una parola monosemica corrisponde un singolo synset, mentre ad una parola polisemica corrispondono più synset. Pertanto un altro modo per identificare i synset, se non ci sono dubbi sul POS, è di indicare una parola che compare nel synset unitamente al numero di significato della parola corrispondente al synset (nel nostro caso *film(1)*).

I synset di WordNet contengono a volte anche parole composite, composte da due o più parole, ma che sono trattate come le parole singole. Ciascun synset ha una definizione associata ad esso, chiamata *gloss*. Esso consite di una breve descrizione che esplicita il significato del concetto rappresentato dal synset. Per esempio il gloss del synset {sample} è: "a small part of something intended as representative of the whole"(una piccola parte di

[3]http://nipadio.lsi.upc.es/cgi-bin/wei/public/wei.consult.perl

qualcosa intesa come rappresentativa del resto), quello di {sample distribution, sample, sampling} è: "items selected at random from a population and used to test hypotheses about the population". Molti gloss (non tutti, però) contengono anche frasi di esempio che mostrano in che modo le parole del synset possono essere incontrate in un testo (per esempio: {fly, fell, vanish} – (pass away rapidly; "Time flies like an arrow").

In WordNet sono memorizzate, oltre ai synset, anche un certo numero di relazioni semantiche tra essi. Le uniche relazioni valide per tutte le categorie lessicali sono quella di *antonimia* (un synset è detto antonimo di un'altro se esprime un concetto opposto, ad esempio: *big, large* (grande) e *small, little* (piccolo), *love(1)* (amore) e *hate, hatred* (odio), *open, open up* (aprire) e *close, shut* (chiudere) e quella di *sinonimia* (che però riguarda le parole e non i synset): due parole sono sinonime se sono presenti nello stesso synset, ad esempio: *big* e *large* (entrambe significano "grande").

Una delle relazioni più importanti, definita però solo per nomi e verbi, è quella di *iperonimia*, nota anche come *is-a* relationship. Si dice che un synset è iperonimo di un'altro se rappresenta un concetto più generico. Ad esempio: *vehicle* è iperonimo di *car* (una macchina *è un* veicolo). La relazione opposta è detta *iponimia* (nel caso dei verbi, in WordNet si parla di *troponimia*); quindi *car* è iponimo di *vehicle*. La relazione è una relazione transitiva: se A è iperonimo di B e B è iperonimo di C, allora A è iperonimo di C. Di conseguenza, tutti i synset dei nomi e dei verbi sono organizzati in gerarchie, ciascuna delle quali ha come radice un synset detto *unique beginner*. Gli *unique beginner* per i nomi in WordNet 2.0 sono i seguenti: *entity, physical_thing*, *psychological_feature*, *abstraction*, *state*, *event*, *act, human_action, human_activity*, *group, grouping*, *possession*, *phenomenon*. La maggior ricchezza di dettagli nella definizione dei nomi è evidente anche dal-

la complessità della gerarchia dei nomi rispetto a quella dei verbi, molto più sparsa e meno profonda. In [5] la sequenza di synset ottenuta prendendo un certo synset e tutti gli iperonimi fino a uno unique beginner viene chiamata *classpath*. La lunghezza media dei classpath per i nomi è più che doppia rispetto a quella dei verbi.

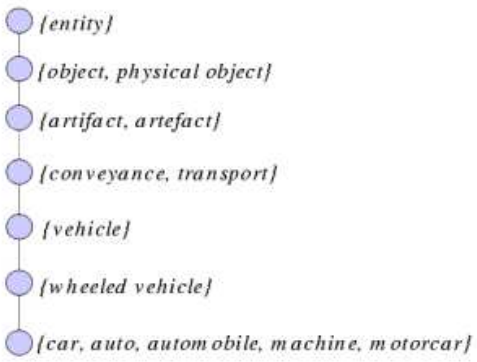


Figura 4.3: Rappresentazione del classpath che connette il synset di *car* con lo unique beginner *entity*

Poi vi è la relazione di *meronimia*, definita solo su nomi, nota anche come *has-part* relationship. Un synset A è meronimo di un synset B se A è una parte di B (ad esempio: *finger(1)* (dito) e *hand(1)* (mano)). La relazione inversa si chiama *olonimia*.

La relazione chiamata *entailment* (implicazione) coinvolge solo i verbi; A *entails* (implica) B se e solo se il fatto che un soggetto compia l'azione A implica che anche l'azione B è compiuta dallo stesso soggetto. Per esempio *snore* (russare) implica *sleep* (dormire). Un'altra relazione definita solo per i verbi è la relazione di causa (*cause*): il synset A causa il synset B, ad esempio *give* and *have* (dare qualcosa a qualcuno causa il possesso della cosa da parte di quel qualcuno).

Le relazioni di *pertainymy* (pertinenza) ed *attribute* (attributo) collegano aggettivi con nomi. Un aggettivo A è pertinente ad un certo nome B se l'aggettivo denota una proprietà del nome (ad esempio: *eletrical*, elettrico ed *electricity*, elettricità). Un aggettivo è legato dalla relazione di *attribute* ad un nome se il nome rappresenta un attributo (ad esempio *size*, dimensione) e l'aggettivo un valore per quell'attributo (ad esempio *small*, piccolo o *large*, grande).

La relazione più frequente definita sugli aggettivi è quella di *similar to* (similarità). Si tratta di una relazione che collega due synset di aggettivi simili nel significato, ma non abbastanza da essere uniti in un singolo synset, ad esempio *far* (lontano) e *distant* (distante).

Nella versione 2.0 di WordNet sono state aggiunte inoltre alcune nuove relazioni basate sul dominio: *category*, *use* e *region*. Queste relazioni collegano synset di qualsiasi categoria lessicale. La relazione di *category* collega un synset A con un altro synset B che rappresenta il dominio a cui appartiene A. Per esempio *sample*, inteso come campione statistico, è in relazione con *statistics*. La relazione di *region* collega un synset A con un synset B che rappresenta la regione geografica associata ad A (ad esempio *French Revolution* e *France*). Infine, *use* collega un synset A con un synset B che identifica un particolare uso di A, per esempio *gone* (andato) con *euphemism* (eufemismo).

La realizzazione di risorse lessicali come dizionari ed ontologie come WordNet è lunga e costosa, e ciò ha due conseguenze principali: le aziende che si occupano di realizzare queste risorse sono restie a renderle pubbliche per non vanificare l'investimento fatto, e la richiesta di includere nuove caratteristiche nelle risorse lessicali disponibili può richiedere lungo tempo, ostacolando il lavoro di ricerca. Per questo nell'ambito della ricerca sul linguaggio naturale si cercano di usare ampie collezioni testuali, *corpora*, appunto, come risorse

lessicali. Sfruttando le tecniche di apprendimento automatico (supervisionato o meno), da un corpus possono essere estratte relazioni sintattiche o semantiche tra le parole. I corpora possono essere etichettati sintatticamente, se le parole nel corpus sono etichettate con la propria categoria sintattica, o semanticamente, se alle parole viene assegnato il significato corretto (per esempio indicando il corrispondente significato in WordNet), oppure non essere etichettati affatto. I corpora etichettati sintatticamente e semanticamente più noti sono il SemCor corpus[4] ed il Senseval[5].

4.3.3 Disambiguatore basato su Densità Concettuale e Frequenza

Per effettuare gli esperimenti con i synset di WordNet è stato utilizzato il disambiguatore basato su Densità Concettuale e Frequenza presentato in [17]. Il problema della disambiguazione si ha in caso di polisemia della parola, ed è necessario per poter attribuire alla parola il significato corretto nel *contesto* in cui si colloca. Il contesto è costituito dalle parole che si trovano intorno alla parola che vogliamo disambiguare. Per esempio, nelle frasi *The president of Turkey* e *Recipes with turkey*, "turkey" ha due significati diversi, che possono essere distinti solo esaminando le parole intorno. Solo allora sarà possibile distinguere il paese eurosiatico con capitale Ankara da un grasso uccello molto gustoso.

Il disambiguatore utilizzato effettua la disambiguazione dei nomi sfruttando le gerarchie di WordNet determinate da iponimi ed iperonimi. Come si può osservare in fig. 4.4, i significati di una parola polisemica determinano

[4]http://www.cs.unt.edu/∼rada/downloads/semcor

[5]www.senseval.org

un partizionamento della gerarchia di WordNet, dal momento che a ciascun significato corrisponde un unico synset.

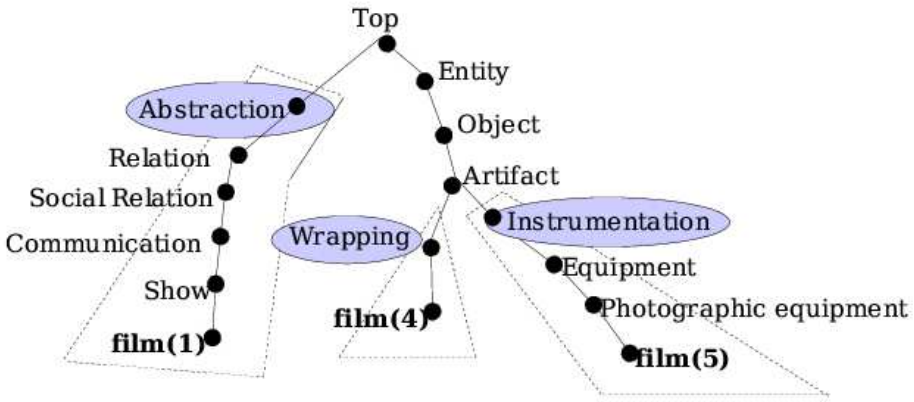


Figura 4.4: Sottogerarchie e rispettive radici determinate da alcuni sensi della parola *film*

All'interno di ciascuna sottogerarchia possono comparire uno o più synsets corrispondenti ai significati delle parole del contesto. La selezione del significato da assegnare a una parola da disambiguare w avente N significati, a cui corrispondono N sottogerarchie, avviene scegliendo il significato corrispondente alla sottogerarchia con densità massima:

$$CD(M, nh, f) = M^{\alpha} \left(\frac{M}{nh}\right)^{\log f}$$

dove M indica il numero di synset delle parole del contesto contenute in una data sottogerarchia ed nh il numero di synset appartenenti alle porzioni di classpath all'interno della sottogerarchia. Con f si indica l'ordine di frequenza del significato correlato alla sottogerarchia (1 indica il più frequente), od, equivalentemente, il numero di significato associato alla sottogerarchia: in questo modo la densità per la sottogerarchia relativa al primo significato è sempre uguale a 1; di conseguenza i significati meno frequenti vengono

selezionati solo se la loro sottogerarchia è particolarmente "densa", ovvero quando $(M/nh) > 1$. $\alpha = 0.10$ è una costante, determinata dopo una serie di esperimenti condotti sul SemCor al fine di distinguere i casi in cui il rapporto $(M/nh) = 1$ e $M = nh = 1$ dai casi in cui $(M/nh) = 1$ con $M = nh \wedge M > 1$. Nel caso esistano più sottogerarchie alla stessa densità, non viene assegnato alcun significato alla parola da disambiguare (viene ritornato 0).

Il disambiguatore seleziona il contesto in maniera simmetrica rispetto al nome da disambiguare, ove possibile; nei casi in cui questo non era possibile è stata aggiunta una parola dal lato opposto. Per esempio si supponga di disambiguare la parola *measure* in un paragrafo in cui compaiono, nell'ordine, le seguenti parole: *density measure correlation sense word context*, e si consideri una finestra di dimensione 4 centrata in *measure*; allora le parole che compongono il contesto di *measure* sono: *density*, *correlation*, *sense* e *word*. La precisione migliore è stata ottenuta con la finestra di dimensione minima (2 nomi).

Questo disambiguatore ha ottenuto l'81.5% di precisione sul SemCor e circa il 73.4% sul Senseval-3 All-Words Task corpus.

Capitolo 5

Esperimenti e Valutazione del Sistema e delle Varianti Realizzate

La fase preliminare all'implementazione di un sistema di IR che facesse affidamento sulla semantica, utilizzando Wordnet, è consistita nella valutazione del sistema di base utilizzato, cioè il Furbo IR System. Lo scopo di questa fase era di verificare l'efficacia di tale sistema, in particolare rispetto ad un sistema similare (cioè molto semplice e che non utilizza tecniche di analisi del testo), il sistema JIRS sviluppato anch'esso all'Università Politecnica di Valencia [9]. JIRS è un sistema di Information Retrieval basato su passaggi. Ogni passaggio è costituito da una frase contenuta nei documenti della collezione (idea ispirata da Llopis 2002 [11]). Il sistema è basato sull'idea di Witten (Witten et. al 99 [22]) di dividere i documenti in passaggi durante la fase di indicizzazione. Il sistema è indipendente dal metodo utilizzato per dividere in passaggi i documenti. Inoltre i passaggi possono essere gran-

di quanto i documenti (ed in questo caso il sistema corrisponde ad un IRS standard). JIRS, come Furbo, fa uso del modello vettoriale, ma i termini nell'inverted index non si riferiscono ai documenti, ma ai passaggi che questi documenti contengono.

Per permettere la valutazione automatica del Furbo IRS è stata implementata una collezione di programmi in Java, utilizzando come test set la collezione di documenti del TREC-8.

Successivamente, sono state implementate diverse tecniche di indicizzazione e ne è stato verificato l'impatto sulle prestazioni del sistema. A causa della grande mole di dati e la complessità computazionale delle operazioni, per questi esperimenti è stato utilizzato un test set estratto sempre dal TREC-8 ma di dimensioni minori.

Infine, sono stati effettuati alcuni esperimenti con l'espansione delle query, in particolare utilizzando sinonimi e meronimi di alcuni termini geografici, estratti da WordNet. Per l'espansione delle query è stata sviluppata un'applicazione separata scritta in Java.

5.1 Valutazione di Furbo

Per il confronto con JIRS è stato utilizzato il test set completo del TREC-8 adhoc task. Più precisamente sono stati utilizzati i documenti del disk 4 e 5 del TREC-8. I dettagli di tale collezione sono illustrati nella tabella seguente. Perchè questa collezione potesse essere utilizzata da Furbo, è stato necessario prima indicizzarla, ed a tale scopo è stato utilizzato il modello vettoriale e la classica funzione *tf-idf* per il calcolo dei pesi dei termini. Il numero totale dei documenti che sono stati indicizzati e analizzati è risultato pari a 520.155. Per valutare la performance del sistema sono state utilizzate

Origine documenti	# doc	Media parole/doc
Finacinal Times, 1991-1994 (FT)	210.158	316
Federal Register, 1994 (FR94)	55.630	588
Foreign Broadcast Information Service (FBIS)	130.471	322
LA Times (LA)	131.896	351

Tabella 5.1: Statistiche della collezione di documenti del TREC-8 adhoc task.

cinquanta queries, estratte dal TREC-8 adhoc task, numerate da 401 a 450. Ogni query è stata ottenuta utilizzando unicamente il campo *title* dei topics del TREC-8. Per ciascuna di esse il sistema ha dato come output i primi 1000 documenti ordinati in base al grado di somiglianza alla query. Per ogni query del TREC-8 è stata usata una lista di documenti rilevanti, estratta dai giudizi di rilevanza del TREC-8 escludendo quelli che si riferivano ai documenti non indicizzati da Furbo. I documenti che non sono stati giudicati e quindi non erano nella lista dei giudizi di rilevanza sono stati considerati non rilevanti. La tabella seguente riassume i dati relativi ai documenti utilizzati per questo test: il campo *Recuperati* rappresenta il numero totale dei documenti recuperati da Furbo (1000 per ciascuna delle 50 query), *Rilevanti* è il numero totale di documenti rilevanti esistenti nella collezione e *Rilevanti-recuperati* rappresenta il numero totale di documenti rilevanti e recuperati da Furbo per tutte le query.

Una volta ottenuti i risultati da Furbo e da JIRS, è stato disegnato il grafico Precisione-Recall relativo ai due sistemi, dando i valori di Precisione relativi ai 11 livelli di Recall fig 5.1. Dal grafico si deduce che i due sistemi hanno un comportamento analogo, anche se Furbo permette di ottenere una

	Numero di documenti
Recuperati:	50.000
Rilevanti:	4.728
Rilevanti-recuperati:	1.832

Tabella 5.2: Statistiche dei documenti recuperati da furbo per il TREC-8 adhoc task.

Precisione maggiore ai livelli di Recall più alti.

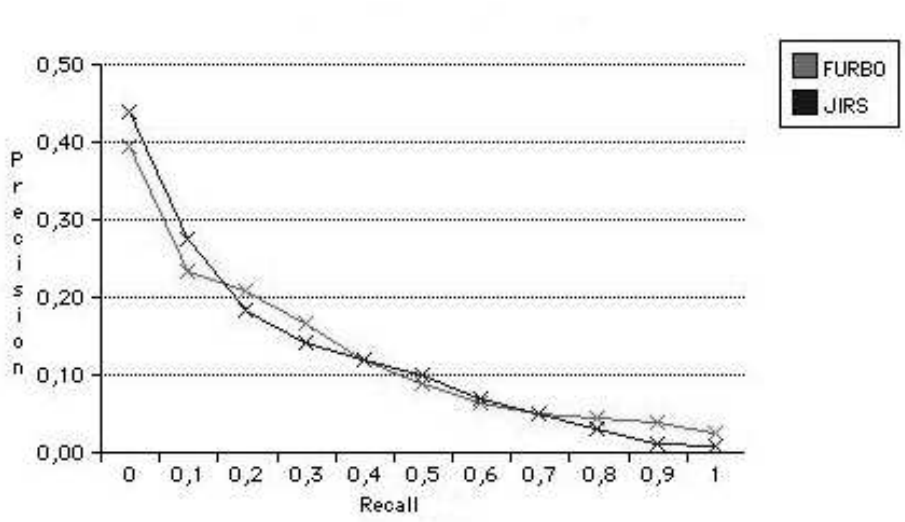


Figura 5.1: Grafico Precisione-Recall per Furbo e JIRS.

5.2 Tecniche di Indicizzazione

Il secondo passo è stato quello usare il sistema Furbo per valutare l'efficienza delle tecniche per indicizzare i termini mediante stem, lemmi e synset rispetto al sistema che non utilizza tali operazioni sul testo. L'indicizzazione con gli stem è stata ottenuta sostituendo ad ogni termine non-stopword il relativo

stem. L'esperimento è stato condotto utilizzando una implementazione in Phython del Porter stemmer descritto nel capitolo 2.

L'indicizzazione mediante i lemmi è stata ottenuta sostituendo ad ogni termine il relativo lemma ottenuto dall'ontologia Wordnet. WordNet fornisce alcune funzioni di lemmatizzazione che permettono di stabilire il lemma corrispondente ad una certa parola, purchè sia noto il POS di tale parola. Pertanto, prima di effettuare questa operazione di lemmatizzazione vera e propria, è stato necessario etichettare sintatticamente tutti i termini nella collezione con il rispettivo POS utilizzando il POS-tagger di Brill [3]. Nei casi in cui un termine non appariva in Wordnet, come indice è stato utilizzato il termine stesso.

L'etichettatura POS è stata impiegata anche per l'indicizzazione mediante synset, dal momento che anche in questo caso è necessario passare al lemma prima di poter assegnare un significato. La disambiguazione dei termini è stata ottenuta utilizzando il disambiguatore basato su densità concettuale descritto in [16]. Come termini indice sono stati utilizzati gli *offset* dei synset (un offset è un codice numerico presente in WordNet che identifica univocamente un synset), quando il disambiguatore ha potuto fornire una risposta, mentre in tutti gli altri casi (densità uguali per diversi significati, oppure POS diverso da nome) è stata mantenuta la parola originale.

Tutte le tecniche di indicizzazione sono state valutate usando lo schema TF-IDF (Term Frequency - Inverse Document Frequency) per i pesi, mentre alcuni esperimenti sono stati condotti modificando i pesi basandosi sulla frequenza e sulla distribuzione dei termini nel testo, come proposto in [10]. Questa tecnica considera come termini rilevanti quei termini che hanno grande frequenza e grande deviazione, ovvero un'ampia distribuzione nel testo. Per far ciò abbiamo definito una formula di correzione dei pesi, in

modo da poter cambiare il numero delle occorrenze di un termine in modo porporzionale alla sua distribuzione nel testo 5.1, permettendoci di tenere la formula TF-IDF per i pesi dei termini.

$$w_d(t) = 1 + f_d(t)\left(\alpha - \frac{\alpha}{(s_d(t)+1)}\right) \tag{5.1}$$

dove $f_d(t)$ è la frequenza del termine t nel documento d, $s_d(t)$ è la deviazione standard del termine t nel documento d e α è un intero > 0 che determina l'ammontare dell'incremento del peso per i termini con alta deviazione.

D'altra parte i termini che hanno alta frequenza e bassa deviazione standard possono essere considerati come keywords di paragrafo. Con questa ipotesi, è stata definita una formula per assegnare un peso maggiore ai termini con bassa deviazione standard e alta frequenza, aventi una posizione media entro il primo 10% del documento:

$$w_d(t) = \delta(t)\left(\frac{f_d(t)}{s_d(t)}\right) \tag{5.2}$$

dove $\delta(t)$ è definito come:

$$\delta(t) = \begin{cases} 2 & \texttt{if } p_d(t) < n_d * 0.10 \\ 1 & \texttt{else} \end{cases} \tag{5.3}$$

dove $p_d(t)$ è la posizione del termine t nel documento d, e n_d è il numero totale di parole nel documento d. Lo scopo di questa formula è di dare più peso alle parole frequenti nelle parti iniziali dei documenti, che spesso contengono più informazioni sul contenuto delle parti intermedie o conclusive di ciascun documento.

5.3 Esperimenti con le diverse Tecniche di Indicizzazione

Il primo esperimento è stato quello di utilizzare il sistema Furbo per valutare la tecnica di indicizzazione mediante lemmi. Per questo esperimento è stato fatto inizialmente un primo tentativo di utilizzare come test set la medesima collezione del TREC-8 usata per la valutazione di Furbo, ma il tempo occorrente per l'elaborazione di un singolo file era talmente alto che l'esperimento sarebbe durato per più di un mese. Questa lentezza era principalmente dovuta all'accesso a WordNet, che rallenatava pesantemente il sistema. Quindi, per poter portare avanti l'esperimento in un tempo "ragionevole", è stato selezionato un sottoinsieme della collezione del TREC-8. La porzione di collezione scelta è stata quella del FBIS (*Foreign Broadcast Information Service*), contenente 130471 documenti, in quanto era quella che possedeva il maggior numero di documenti rilevanti, pari a 1667. Quindi per avere un termine di paragone è stato ripetuto l'esperimento iniziale, che utilizzava come termini indice le parole originali dei documenti, sulla sottocollezione FBIS.

Indicizzazione con	Recall
Word	52.85%
Lemma	55.55%
Synset	52.73%
Stem	65.63%

Tabella 5.3: Valori di Recall ottenuti usando le quattro tecniche di indicizzazione.

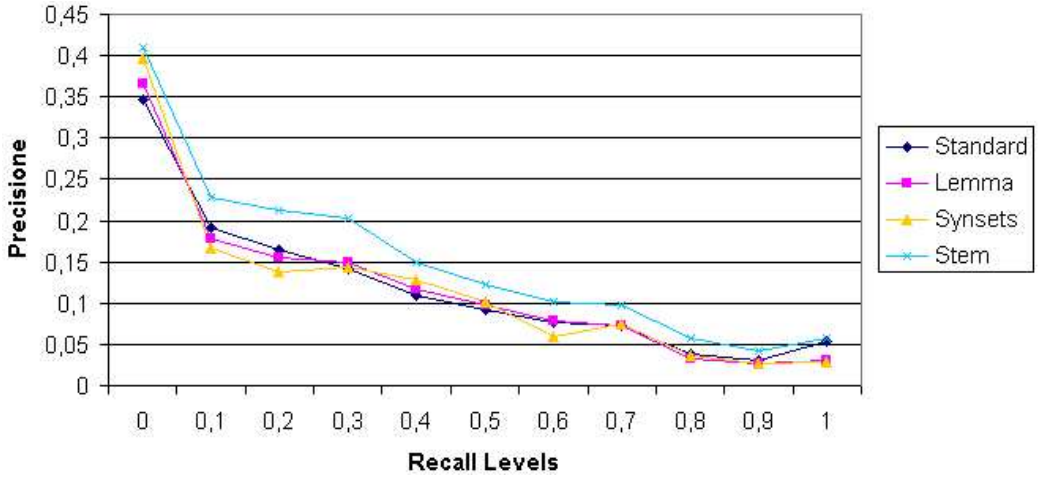


Figura 5.2: Grafico Precisione-Recall per le 4 tecniche di indicizzazione testate.

I risultati dimostrano che i risultati ottenuti usando gli stem superano di gran lunga quelli ottenuti utilizzando gli altri tipi di indice, arrivando ad ottenere un recall maggiore del 10% rispetto agli altri. Come si può notare, indicizzare mediante synset dà risultati simili a quelli ottenuti senza preprocessing del testo. Questo non corrisponde a quanto da noi atteso, dal momento che l'indicizzazione su synset permette di aggregare l'informazione delle parole sinonime tra loro, e quindi ottenere un recall maggiore. Come si può osservare dalla tabella 5.3, l'aggregazione di tale informazione permette di aumentare il recall, come dimostrano i risultati ottenuti proprio con gli stem, e con i lemmi, che unificano la stessa parola quando compare in forme diverse.

Per indagare su questo comportamento non previsto, abbiamo estratto i primi 1000 termini più frequenti dalla collezione, ed abbiamo osservato (vedi tab.5.4) come almeno un terzo di essi fossero nomi propri ed acronimi (ad esempio *oblast*, *kim* o *impd*), che non si trovavano in WordNet, spesso a causa della loro origine non inglese. Un altro terzo circa era composto da

Tipo del termine	Percentuale
Nomi propri	25.63%
Acronimi	9.15%
Parole non disambiguate	35.05%
Synset offsets	30.17%

Tabella 5.4: Composizione dei termini più frequenti nella collezione dopo l'indicizzazione mediante synsets.

parole non disambiguate dall'algoritmo. Perciò, dato che una gran parte dei termini nonostante il processing si trovano nella forma originale, i risultati non si discostano di molto da quelli ottenuti senza preprocessing, ed i vantaggi derivanti dall'aver aggregato le parole sinonime vengono vanificati dagli errori di disambiguazione.

5.4 Formulazione delle Query

La collezione dei topic del TREC-8 adhoc task consiste in 50 topics, la cui lunghezza media è di circa tre parole. Le query sono state trattate secondo la tecnica di indicizzazione adottata (per esempio se i documenti sono stati indicizzati mediante lemmi, anche le queries devono essere lemmatizzate). L'osservazione di alcune caratteristiche della collezione di test e dei topics, cioè che possono essere in gran parte considerati nell'ambito "geo-politico", ha dato spunto ad ulteriori esperimenti che sono consistiti nell'espandere i termini geografici delle query con sinonimi e meronimi. L'espansione delle query è stata ottenuta seguendo il seguente algoritmo:

1. Selezionare le parole (w) della query che sono state etichettate come

nomi propri;

2. Controllare in Wordnet se w ha il synset {*country, state, land*} tra i suoi iperonimi. Se il controllo è positivo aggiungere alla query tutti i sinonimi, ad eccezione di w e delle stopwords, se presenti, e procedere al passo 3, altrimenti tornare a 1.

3. Recuperare i meronimi di w ed aggiungere alla query tutte le parole del synset che contengono la parola *capital* nel suo gloss o nel suo synset, ad accezione della parola *capital* stessa, delle stopwords e di w. Se ci sono più parole nella query ritornare al passo 1, altrimenti fine.

Ad esempio la query 401 *foreign minorities, Germany* viene etichettata dal POS-tagger nel seguente modo: JJ/foreign, NNS/minorities, NNP/Germany. Quindi viene selezionata la parola Germany come w. Il corrispondente Synset in wordnet è {*Germany, Federal Republic of Germany, Deustschland, FRG*} e siccome i suoi iperonimi includono il synset {*country, state, land*} i seguenti sinonimi vengono aggiunti alla query: Federal Republic, Deustschland, FRG. I seguenti meronimi contengono la parola "capital" nel synset o nel gloss:

- Berlin, German capital – (capital of Germany, located in eastern Germany)

- Bonn – (a city in western Germany on the Rhine River; was the capital of West Germany between 1949 and 1989)

- Munich, Muenchen – (the capital and largest city of Bavaria in southeastern Germany)

- Aachen, Aken, Aix-la-Chapelle – (a city in western Germany near the Dutch and Belgian borders; formerly it was Charlemagne's northern capital)

Quindi la risultante query espansa è *foreign minorities, Germany Federal Republic Deutschland FRG Berlin German Bonn Munich Muenchen Aachen Aken Aix-la-Chapelle.*

5.5 Esperimenti con Espansione delle Query

Gli esperimenti con l'espansione della query sono stati svolti processando i topics del TREC-8 con le procedure espresse nella precedente sezione. Le queries elaborate dall'algoritmo di espansione della query (Expanded Query, EQ), sono state in seguito sottoposte al sistema, utilizzando in un caso la collezione indicizzata con i lemma ed in un altro quella indicizzata con gli stem. I risultati ottenuti mostrano un miglioramento del Recall del 2% quando è stata usata la collezione indicizzata con gli stem, mentre un peggioramento di oltre del 3% quando è stata utilizzata la collezione indicizzata con i lemma (tabella 5.5). Un altro interessante risultato è che la Precisione nei

Indicizzazione con	Recall EQ	Recall ¬EQ
Lemma	51.95%	55.55%
Stem	67.79%	65.63%

Tabella 5.5: Valori di Recall ottenuti usando la frequenza dei termini e la distribuzione; Recall EQ: Recall ottenuti con l'espansione della query, Recall ¬EQ: ottenuti senza l'espansione della query.

due casi è più bassa di quella ottenuta con gli esperimenti senza l'espansione

della query, come si può notare dalla figura 5.3. La spiegazione è che vengono recuperati più documenti rilevanti, ma non vengono classificati nelle prime posizioni.

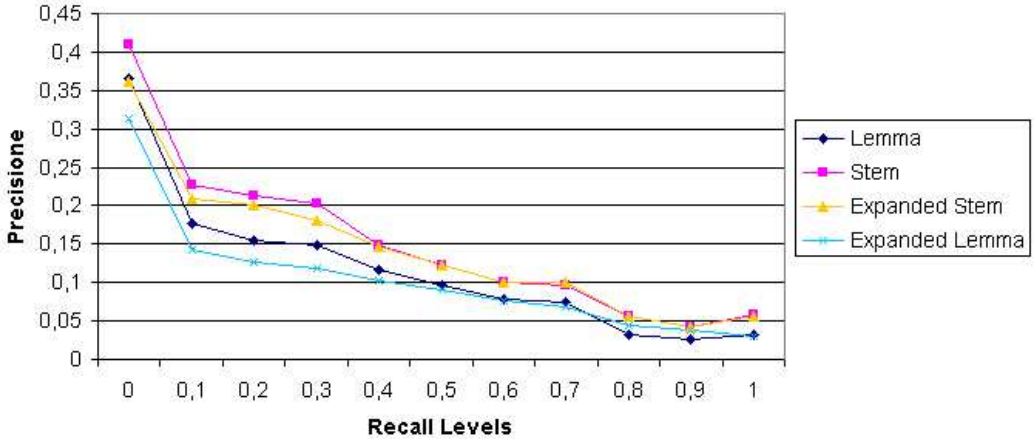


Figura 5.3: Grafico Precisione-Recall risultante dall'utilizzo di tecniche di espansione delle query in comparazione con le rispettive versioni senza espansione.

5.6 Esperimenti con la Frequenza dei Termini e la Distribuzione

In questi esperimenti i documenti sono stati indicizzati applicando la formula 5.1 con α uguale a 1 e 10 e la formula 2; in tutti questi casi sono stati usati gli stem come indici. Per quanto riguarda i valori assegnati al parametro α, bisogna sottolineare che inizialmente α doveva essere sempre uguale a 1. Tuttavia, osservando che con questo valore le frequenze dei termini differivano poco dalle frequenze ottenute senza tenere conto della deviazione, abbiamo provato ad amplificare il peso dovuto alla deviazione introducendo un valore

notevolmente più grande per α. Uno studio più dettagliato sul parametro α potrebbe essere uno degli sviluppi futuri di questo lavoro.

In seguito i risultati ottenuti sono stati comparati con quelli precedentemente ottenuti con gli stem. Dal grafico Precisione-Recall della figura 5.4 si può dedurre che un leggero miglioramento è stato ottenuto assegnando ai termini che hanno alta frequenza e grande deviazione standard pesi più grandi, ovvero i temini che hanno grande distribuzione nel testo. Invece, le keywords "di paragrafo", i termini con alta frequenza e bassa deviazione standard, deteriorano notevolmente la performance del sistema, anche se sono stati assegnati pesi più alti ai termini nel primo 10% di ogni documento. È invece importante il fatto che la stessa strategia di raddoppiare i pesi dei termini nel primo 10% applicata alle keywords "globali" dà un Recall migliore rispetto a quello ottenuto con i pesi normali (tabella 5.6).

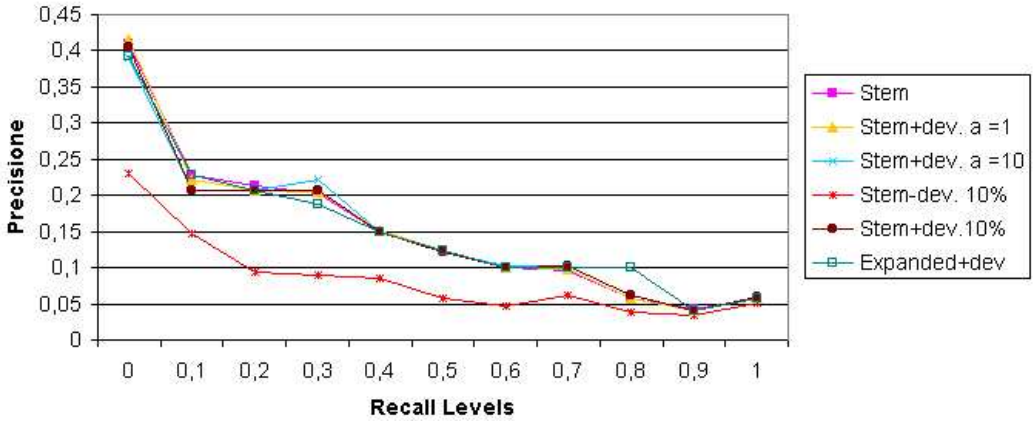


Figura 5.4: Grafico Precisione-Recall ottenuto utilizzando diversi pesi per la deviazione.

Un ultimo esperimento è stato condotto combinando la tecnica basata sulla frequenza dei termini (con $\alpha = 1$) e la deviazione con l'espansione delle query (i risultati sono indicati sotto l'etichetta "Expanded+dev" in figura

Indicizzazione con	Recall
Stem+dev, α=1	65,99%
Stem+dev, α=10	67,31%
Stem-dev, 10% doc	48,17%
Stem+dev, 10% doc	66,65%
Expanded+dev, α=1	66,23%
Stem	65.63%

Tabella 5.6: Valori di Recall ottenuti usando la frequenza dei termini e la distribuzione.

5.4 e tabella 5.6). Anche in questo caso l'espansione delle query ha permesso di incrementare il recall rispetto alla tecnica senza espansione.

Inizialmente era stato previsto di tentare di disambiguare solo i termini rappresentativi (quindi senza usare gli stem), ma due problemi principali ci hanno fatto desistere:

- Se una parola con grande frequenza e deviazione viene disambiguata, questa può risultare in significati diversi a seconda di dove si trova nel testo. In questo caso, è necessario ricalcolare la deviazione e non è detto che la parola continui ad risultare rappresentativa.

- In tal caso, risolvere il problema precedente comporta un uso massiccio di tempo e memoria e non è possibile indicizzare in tempi ragionevoli.

I risultati di questi esperimenti effettuati con le tecniche di indicizzazione basate su stem, lemma e synset verranno presentate al workshop "III Jornadas en Tecnología del Habla", che avrà luogo a Valencia il prossimo novembre (articolo "Comparison of Indexing Techniques based on Stems, Synsets, Lemmas and Term Frequency Distribution" vedi appendice).

Capitolo 6

Information Retrieval su Web

Oggigiorno, un numero via via sempre maggiore di informazioni si rende disponibile sul web: la ricerca di informazioni sul Web è una delle attività più comuni (se non la più diffusa) di chi utilizza Internet. L'information retrieval su Web allo stesso tempo presenta una sfida tecnica dovuta all'eterogeneità, alle dimensioni ed alla dinamicità del Web (Giugno 98: 350 milioni di pagine, Aprile '01: 4 miliardi di pagine). Per questo motivo i metodi di information retrieval tradizionali non sempre sono adatti alle applicazioni su internet, dal momento che sono stati sviluppati per lo più per essere utilizzati su collezioni statiche, omogenee e di dimensioni, per quanto ampie, sicuramente inferiori a quelle del Web.

Le principali differenze tra Information Retrieval classico e Web Information Retrieval possono essere riassunte nei seguenti punti [8]:

- Dimensione: la dimensione del Web era di 4 miliardi di pagine ad Aprile 2001, mentre una collezione testuale come quella del TREC contiene fino ad 1 milione di documenti [21], 1000 volte meno;

- Dinamicità: la dimensione del Web non è costante, a differenza delle collezioni testuali;
- Eterogeneità: Internet contiene un'ampia varietà di tipi di documenti: immagini, file audio, testuali, etc.;
- Linguaggi: sul Web è possibile incontrare quasi ogni linguaggio parlato sulla superficie del piantea, mentre le collezioni utilizzate nell'Information Retrieval tradizionale sono spesso scritte in una sola lingua;
- Ridondanza: si stima che sul Web il 30% delle pagine siano duplicate;
- Ipertesto: i documenti nel Web sono collegati tra loro: si stima che una pagina Web abbia in media più di 8 links ad altre pagine;
- Query formulation: le query sul Web sono in genere più corte e non particolarmente strutturate;
- Varietà degli utenti: gli utenti del Web variano sensibilmente in conoscenze, necessità ed aspettative, mentre, ad esempio,un utente di una biblioteca digitale avrà bisogni ed aspettative costanti;
- Pigrizia degli utenti: si stima che l'85% degli utenti del Web si fermano alla prima schermata di risultati ritornati dal motore di ricerca. Il 78% non riformula la query iniziale.

6.1 Motori di Ricerca per il Web

La qualità dei risultati ritornati è la caratteristica principale che un buon motore di ricerca per il Web dovrebbe avere. Il recall non è così importante

per questi sistemi come nei sistemi tradizionali di IR, dal momento che il numero totale di documenti rilevanti per una query non può essere determinato. Invece, avere tra le prime posizioni una maggioranza di documenti rilevanti (precisione) è fondamentale: una presenza eccessiva di "risultati spazzatura" (*Junk Results*) nelle prime posizioni rende la ricerca inutile anche se tra questi risultati se ne trova uno particolarmente valido. Inoltre, il concetto stesso di rilevanza va rivisto: è importante avere documenti *molto* rilevanti nelle prime posizioni che non documenti leggermente rilevanti.

6.1.1 Architettura di un Motore di Ricerca per il Web: Google

Le componenti basilari di un motore di ricerca sono tre: un indicizzatore, un *crawler*, un server per le query. Il *crawler* è un programma che naviga nel Web alla ricerca di pagine da collezionare. L'indicizzatore processa i documenti recuperati dal crawler e li rappresenta in una struttura dati efficiente. Il query server accetta le query dell'utente e ritorna i risultati attraverso l'esame delle strutture dati. Le prestazioni del query server sono critiche dal momento che un grande numero di utenti può essere collegato contemporaneamente e i tempi di attesa lunghi allontanano gli utenti dal sistema.

In figura 6.1 è mostrata l'architettura del più noto motore di ricerca, Google[1]. Google è implementato in C o C++ per l'efficienza e può funzionare sia su Solaris che Linux. In Google, il crawling è distribuito su più crawlers. C'è un URL server che invia liste di URLs che devono essere esaminate dai crawlers. Le pagine web esaminate dai crawlers vengono im-

[1]http://www.google.com

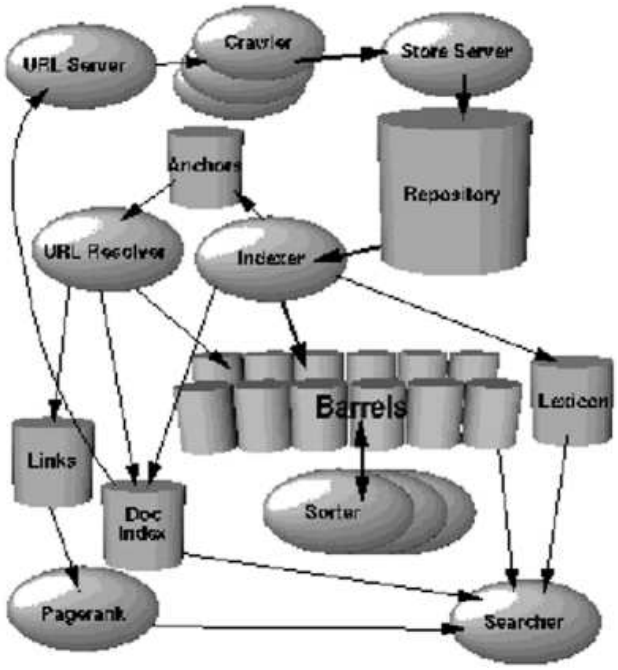


Figura 6.1: Architettura di Google [4].

magazzinate in uno "storeserver" che comprime ed immagazzina le pagine web in un database. Ciascuna pagina web ha un numero ID (detto *docID*) che viene assegnato nel momento in cui si incontra un link a quella pagina. La funzione di indicizzazione è eseguita da due moduli distinti detti *indexer* e *sorter*. L'indexer legge dal database, decomprime i documenti e ne fa il parsing. Ciascun documento viene convertito in un insieme di record (uno per ciascuna parola del documento) detti *hits*. Gli hits tengono traccia della parola, la sua posizione nel documento, la dimensione del font, ed altre caratteristiche. L'indexer distribuisce gli hits in un insieme di *barrels*, creando un indice parzialmente ordinato. L'altra funzione importante dell'indexer è di esaminare tutti i link in ciascuna pagina web e di immagazzinare le informazioni sui link (dove punta e da quale pagina, e il testo stesso del link).

L'*URL resolver* legge il file generato per i link e converte le URL relative in URL assolute e quindi nei corrispondenti docIDs. Il testo del link viene inserito nell'indice, associato con il rispettivo docID. Genera anche un database di links (coppie di docIDs), utilizzato per calcolare il ranking dei documenti attraverso l'algoritmo di *PageRank*.

Il sorter prende i barrels e li riordina in base alle wordIDs per generare l'inverted index. Un programma chiamato *DumpLexicon* prende la lista di wordIDs prodotta dal sorter ed il lessico prodotto dall'indexer per generare un lessico utilizzato dal *searcher* Il searcher è un programma in esecuzione su un web server che usa l'inverted index, il lessico generato dal DumpLexicon e l'algoritmo PageRanks per rispondere alle queries.

6.1.2 Algoritmo di Ranking PageRank

Questo algoritmo è stato proposto da Brin e Page [4] e viene utilizzato da Google. Si basa fondamentalmente sul conteggio dei backlink verso le pagine, però estende questa tecnica in quanto non assegna lo stesso valore a tutti i link: il valore di un link dipende dall'importanza della pagina (calcolata in base al numero di link che la indicano) da cui parte e dal numero di link che sono presenti in tale pagina; si tratta quindi di un algoritmo ricorsivo.

L'algoritmo è basato sulla seguente definizione: Si assuma che la pagina A abbia n pagine $T_1, ..., T_n$ che puntano ad essa. $C(A)$ sia definito come il numero di link che partono dalla pagina A. Il valore di PageRank per la pagina A è dato da:

$$PR(A) = (1-d) + d\left[\frac{PR(T_1)}{C(T_1)} + \ldots + \frac{PR(T_n)}{C(T_n)}\right]$$

I valori di PageRank vengono calcolati usando tecniche di tipo matriciale in cui si costruisce una matrice dei link presenti tra le pagine web che costi-

tuiscono il repository su cui si lavora; gli autori fanno notare che i valori di PageRank per un set di 25 milioni di pagine possono venire calcolati in poche ore su una workstation di media potenza (ovviamente il calcolo di tali valori viene eseguito off-line). Si tratta di valori generali: non sono riferiti ad una specifica query, ma all'importanza delle varie pagine in base ai link esistenti. La giustificazione intuitiva degli autori è che questo algoritmo può essere visto come un modello del comportamento di un certo utente ideale che naviga il web; si suppone infatti di avere un "random surfer" che parte da una pagina a caso e, cliccando sui vari link, naviga tra le pagine del web. Tale utente non esegue mai un "back" sulla pagina precedente, ma ad un certo punto può stufarsi e ricominciare nuovamente la navigazione da una pagina a caso (il parametro d è proprio la probabilità che il random surfer si annoi e richieda una nuova pagina di inizio casuale); la probabilità che un utente visiti una determinata pagina è proprio il valore di PageRank di quest'ultima.

Conclusioni e sviluppi futuri

Il lavoro svolto nell'ambito della tesi si proponeva inizialmente di verificare l'efficacia dell'indicizzazione mediante synset di WordNet, utilizzando in particolare il disambiguatore semantico basato su Densità Concettuale descritto in [17]. Successivamente è stata ampliata la portata del lavoro, effettuando dapprima uno studio più approfondito sulle diverse tecniche di indicizzazione, quindi testando una nuova tecnica di indicizzazione basata sulla distribuzione dei termini nel testo e non solo sulla frequenza. Infine sono stati effettuati alcuni test con l'espansione dei termini geografici nelle query.

I risultati ottenuti attraverso l'indicizzazione mediante synset rispecchiano quelli riscontrabili nella letteratura esistente, di conseguenza è stata riscontrata l'inferiorità di tale tecnica di indicizzazione rispetto alle altre, in particolare all'indicizzazione mediante stem. E' stata poi verificata l'efficacia della tecnica di indicizzazione basata sulla distribuzione degli stem nel testo, dando un peso maggiore a quegli stem che compaiono con grande frequenza e grande deviazione standard, che pertanto sono distribuiti in tutto il testo. Attraverso questa tecnica è stato dimostrato che è possibile incrementare il recall. Per contro, è stato osservato che dando un peso maggiore agli stem con grande frequenza e piccola deviazione le prestazioni del sistema decadono notevolmente, a dimostrazione che si tratta di "keywords di paragrafo" che

hanno poco a che fare con il contenuto generale del testo.

Gli esperimenti effettuati con l'espansione delle query, in particolare dell'espansione dei termini geografici, sono risultati aumentare il recall ma non la precisione. Il tipo di espansione è stato studiato in funzione della natura particolare della collezione utilizzata, che contiene una grande quantità di documenti a carattere geo-politico. Pertanto questi risultati non andrebbero generalizzati, ma bisognerebbe eseguire ulteriori esperimenti con altre query ed eventualmente diverse collezioni testuali.

L'analisi degli errori ha permesso di verificare che i vantaggi dell'indicizzazione attraverso synset, ovvero l'aggregazione di parole con il medesimo significato, vengono annullati dalla percentuale di errori commessi in fase di disambiguazione, confermando i risultati ottenuti in [6]. Sviluppi futuri in questo senso possono essere determinati solo dal progresso nella realizzazione di disambiguatori che forniscano una precisione elevata, anche se i vantaggi ottenibili attraverso un semplice stemming non permettono ottimismo riguardo al futuro delle tecniche basate su synset di WordNet.

Al fine di trovare un modo per impiegare WordNet in maniera più redditizia, potrebbe essere necessario un raffinamento delle tecniche di espansione delle query, specialmente con l'obiettivo di permettere l'utilizzo di queste tecniche su collezioni testuali più generiche di quanto non sia la collezione FBIS utilizzata per gli esperimenti condotti nell'ambito di questa tesi.

Una ulteriore direzione di ricerca, sicuramente più promettente, consiste nell'individuare una tecnica più efficace di quella utilizzata in questo lavoro per integrare l'informazione derivante dalla distribuzione dei termini. Potrebbe essere eseguito uno studio sul parametro α introdotto, per proseguire con la realizzazione di tecniche più sofisticate.

Bibliografia

[1] R.Baeza-Yates, B.Ribeiro-Neto, *Modern Information Retrieval*. Addison-Wesley, Cap. 1, 1999

[2] T.Brants, *TnT - A Statistical Part-Of-Speech Tagger*. www.coli.uni-sd.de/ thorsten/tnt

[3] E.Brill, *A simple rule-based Part-Of-Speech tagger*. In: Proceedings of the Third Conference on Applied Computational Linguistics, Trento, Italy, 1992

[4] S.Brin, L.Page, *The Anatomy of a Large-Scale Hypertextual Web Search Engine*. In: Proceedings of the Seventh International World Wide Conference, Brisbane, Australia, 1998

[5] P.Buitelaar et al., *An Unsupervised Semantic Tagger applied to German*. In: Proceedings of the International Conference on Recent Advances in NLP (RANLP), pp.52-57, Tzhigov Chark, Bulgaria, 2001

[6] J.Gonzalo, F.Verdejo, C.Peters, and N.Calzolari, *Applying EuroWordNet to Cross-language Text Retrieval*. Computers and Humanities, 32(2-3):185-207, 1998

[7] J.Gonzalo, F.Verdejo, I.Chugar, J.Cigarrán, *Indexing with WordNet Synsets can improve Text Retrieval*. In: Proceedings of Workshop on Usage of WordNet for NLP, 1998

[8] L.Huang, *A Survey On Web Information Retrieval Technologies*. Tech. Rep. ECSL, 2000

[9] J.M. Gómez, E.Sanchis, *JIRS: Information Retrieval System in Java*. Departamento de Sistemas Informaticós y Computación Universidad Politécnica de Valencia, Internal Report, agosto 2004.

[10] J.W. Lee, D.K. Baik *A model for Extracting Keywords of Document using Term Frequency and Distribuition*. In: Lecture Notes in Computer Science, Springer Verlag, pp.437-440, 2004

[11] F. Llopis, *Un sistema de Recuperación de Informatión basado su Pasajes*. In: Ph.D. Thesis, Universidad de Alicante, 2002

[12] M.Marcus, B.Santorini, M.A. Marcinkiewicz, *Building a Large Annotated Corpus of English: the Penn Treebank*. Computational Linguistics 19(2):313-330, 1993

[13] G.Miller, *WordNet: A Lexical Database for English*. In: Communications of the ACM, 38(11):39-41, 1995

[14] M.F. Porter, *An algorithm for suffix stripping*. In: Readings in Information Retrieval, pages 313-316. Morgan Kaufmann Publishers, Inc., 1997

[15] D.Jimnez, E.Ferretti, V.Vidal, P.Rosso, C.F.Enguix *The Influence of Semantics in IR using LSI and K-Means Clustering Techniques*. In: Work-

shop on Conceptual Information Retrieval and Clustering of Documents, pages 286-291, Dublin, 2003

[16] P.Rosso, F.Masulli, D.Buscaldi, F.Pla, A.Molina *Automatic Noun Disambiguation*. In: Lecture Notes in Computer Science (LNCS), Springer Verlag, (2588):273-276, 2003

[17] P.Rosso, F.Masulli, D.Buscaldi, *Word Sense Disambiguation with and without Supervision*. In: Proceedings of CIC2002, 2:531-540, Mexico City, D.F., Mexico, 2002

[18] S.J.Russell, P.Norvig, *Intelligenza Artificiale: Un Approccio Moderno*. Prentice Hall International, Capp. 22-23, 1998

[19] G.Salton, *Automatic Information Organization and Retrieval* . McGraw-Hill, Cap. 1, Gaithersburg, Virginia, 1968

[20] C.J.Van Rijsbergen, *Information Retrieval*. Dept.of Computer Science, University of Glasgow, 1981

[21] E.M.Voorhees, D.Harman, *Overview of the Eighth Text REtrieval Conference (TREC8)*. In: Proceedings of the 8th International Conference on Text REtrieval Conference at NIST, Cap. 1, 1981

[22] I.H. Witten, A. Moffat, T.C. Bell, *Managing Gigabytes*. Morgan Kaufmann Publishers, INC, 1999

Appendice
Pubblicazione realizzata nell'ambito del lavoro di tesi