ALMA MATER STUDIORUM
UNIVERSITÀ DEGLI STUDI DI BOLOGNA

Seconda Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica

TEXT MINING GERARCHICO: CLASSIFICAZIONE SEMANTICA DI DOCUMENTI IN TASSONOMIE DI ARGOMENTI

Elaborata nel corso di: Tecnologie E Sistemi Per Il Data Mining Lm

Tesi di Laurea di:
GIACOMO DOMENICONI

Relatore:
Prof. GIANLUCA MORO

Co-relatore:
Dott. Ing. ROBERTO PASOLINI

ANNO ACCADEMICO 2011-2012
SESSIONE I

PAROLE CHIAVE

data mining

text mining gerarchico

classificazione semantica

hierarchical text classification

tassonomie web

Indice

Introduzione

La classificazione è una funzione eseguita naturalmente dagli esseri umani. Essa non è una disciplina inventata dall'uomo, ma è piuttosto una capacità umana innata. Inoltre è legata alla nostra memoria che è organizzata in modo da rendere le informazioni ottenute dalle esperienze passate a disposizione per le situazioni attuali.
L'essere umano sperimenta solo singoli eventi, li ricorda e li identifica come istanze di classi o categorie. Di conseguenza, l'essenza della memoria sono l'organizzazione e la classificazione. Quanto difficile sarebbe la nostra vita se non avessimo capacità di classificare? Quante diverse attività umane hanno una classificazione come fondamento?
Anche gli atti più banali come distinguere un'entità come un cane e un gatto richiedono la capacità di classificare sulla base delle nostre percezioni. Come già detto da Estes [46] la classificazione è infatti fondamentale per tutte le nostre capacità intellettuali.

L'era dell'informazione, in particolare con l'avvento del web, ha portato a una grandissima quantità di dati, generalmente però disorganizzati. Il numero di informazioni presenti sul web ha registrato un incremento esponenziale, mentre la capacità di analisi e lettura di essi da parte degli utenti è rimasta pressoché invariata. La conseguenza è un fenomeno detto Information Overload (sovraccarico cognitivo) e con esso si intende la difficoltà di recupero di informazioni rilevanti a causa della mole

di dati da consultare.

La classificazione di documenti digitali, siano essi documenti testuali, siti web, ecc. è un processo che richiede tempo e, a volte, è persino frustrante se fatto manualmente. Classificare i nuovi elementi manualmente ha alcuni svantaggi:

1. Per le specifiche zone di interesse, sono necessari specialisti di ogni zona per l'assegnazione di nuovi elementi (ad esempio banche dati mediche, banche dati giuridiche) a categorie predefinite.

2. Assegnare manualmente i nuovi elementi è un compito soggetto a errori, perché la decisione è fondata sulla conoscenza e la motivazione di un dipendente.

3. Le decisioni di due esperti umani possono essere in disaccordo (inter-indicizzazione, incoerenza).

L'uso comune della classificazione in categorie di argomenti consinste nella creazione di una tassonomia di argomenti, organizzando gerarchicamente le informazioni si rende più semplice la consultazione dall'utilizzatore, si facilita il reperimento. Inoltre, nell'ambito del web, si rende possibile l'indicizzazione sui motori di ricerca, i quali sono in grado di interpretare il tema di una certa categoria e dai siti in essa contenuti.

L'obiettivo di questa tesi è lo sviluppo di un metodo di classificazione semantica di documenti testuali non strutturati, mediante tecniche di Text Mining, in categorie organizzate gerarchicamente. In particolare l'idea alla base del sistema che si vuole costruire differisce dai normali metodi proposti in letteratura, che rappresentano i documenti tramite un dizionario composto dalle loro una bag-of-words, in quanto si vuole creare un processo che si basi non sulle singole parole, ma sulle relazioni semantiche tra coppie di termini. Si passa così dalla creazione di un modello basato sulle singole parole di ogni documento a uno creato

sulla base di coppie di elementi, valutando la relazione fra essi: si relazionano coppie di termini appartenenti a un documento e una categoria.
Viene quindi abbandonato l'utilizzo di singoli termini utilizzati come features per classificare i documenti, introducendo nuovi attributi generati in base alla presenza di relazioni semantiche, ricercate tramite il dizionario WordNet, tra coppie di parole; si crea in questa maniera un meta-classificatore, indipendente dal dizionario di parole contenute nell'insieme di documenti di training.

La tesi si organizza nel modo seguente:

- Nel *capitolo 1* viene illustrata una panoramica del text mining, fornendo un dettagliato studio dei più recenti metodi di classificazione e clustering testuale, focalizzandosi principalmente sulle tecniche che prevedono una categorizzazione all'interno di una gerarchia di argomenti.

- Il *capitolo 2* descrive le sorgenti dati utilizzate per testare il sistema, inoltre presenta i principali strumenti utilizzati nel sistema: WordNet, per la ricerca di relazioni semantiche, e Weka, strumento utilizzato per la creazione di algoritmi e modelli di data mining.

- Nel *capitolo 3* viene presentata l'idea alla base della tesi, riguardante tutto il processo di classificazione gerarchica, partendo dal preprocessamento, proseguendo nella dimensionality reduction e concludendo con la classificazione gerararchica utilizzando il nuovo metodo basato sulle relazioni semantiche tra documenti e categorie.

- Il *capitolo 4* descrive l'implementazione pratica dei metodi proposti nel capitolo precedente, mostrando l'architettura del sistema e le interazioni tra le varie parti dello stesso.

- Il *capitolo 5* mostra i risultati ottenuti nei vari test del sistema, suddivisi per classificazione semantica di coppie documento-categoria e classificazione gerarchica di documenti.

Capitolo 1

Text mining: Stato dell'Arte

In questo capitolo verrà presentata una panoramica generica del data mining e del text mining, fornendo le definizioni formali dei modelli che si andranno a creare e delle entità in essi utilizzate. Verrà inoltre fornito un dettagliato studio dello stato dell'arte riguardante il clustering e la classificazione testuale, con una particolare focalizzazione sulle tecniche che prevedono strutture di argomenti gerarchiche.

1.1 Introduzione al text mining

Il text mining, o alternativamente data mining testuale, si riferisce al processo di derivazione di informazioni di alta qualità ricavate da un testo. Queste informazioni sono tipicamente derivate attraverso l'elaborazione di modelli statistici.
Il text mining è una specializzazione del data mining, una disciplina ampiamente consolidata e studiata. Data mining significa *scavare* nei dati per trasformare il patrimonio di informazioni in indicazioni strategiche, consiste quindi nell'insieme di tecniche e metodologie che hanno per oggetto l'estrazione di un sapere o di una conoscenza a partire da grandi quantità di dati (attraverso metodi automatici o semi-automatici) e l'utilizzo scientifico, operativo o industriale di questa conoscenza. Queste tecniche

fanno emergere le tendenze, le relazioni tra i dati, le ragioni legate al manifestarsi dei fenomeni; evidenziano non solo cosa sta accadendo, ma anche perchè.
Le numerose tecniche impiegate nel data mining possono essere ricondotte a cinque aree di applicazione:

- Previsione: si utilizzano valori noti per la previsione di quantità non note (es. stima del fatturato di un punto vendita sulla base delle sue caratteristiche).
- Classificazione: individuazione delle caratteristiche che indicano a quale gruppo un certo caso appartiene (es. discriminazione tra comportamenti ordinari e fraudolenti).
- Segmentazione: individuazione di gruppi con elementi omogenei all'interno del gruppo e diversi da gruppo a gruppo (es. individuazione di gruppi di consumatori con comportamenti simili).
- Associazione: individuazione di elementi che compaiono spesso assieme in un determinato evento (es. prodotti che frequentemente entrano nello stesso carrello della spesa).
- Sequenze: individuazione di una cronologia di associazioni (es. percorsi di visita di un sito web).

Il text mining, come detto, è una forma particolare di data mining dove i dati consistono in testi in lingua naturale, in altre parole, documenti "destrutturati". Questa mancanza di una struttura predefinita, caratteristica dei documenti testuali, crea una differenza significativa nelle metodologie di analisi dei dati rispetto al data mining. È necessaria una fase di preprocessamento dell'informazione che possa portare il testo libero in un formato strutturato, rendendo possibile in questa maniera l'utilizzo degli algoritmi di data mining.
Attribuire una struttura ai dati è necessario per l'applicazione

di tecniche di data mining. L'elemento base di una struttura è definito *record* e consiste in un insieme predefinito di informazioni (*attributi*), che potrebbero a loro volta essere strutturati (es. un punto tridimensionale è definito nei tre attributi x,y e z, un numero complesso dall'attributo reale e da quello immaginario). L'idea stessa di data mining è legata al concetto di analisi dei diversi *record*, che normalmente coincidono con tuple di database.
Per l'applicazione delle tecniche di data mining a documenti testuali è necessaria quindi una trasformazione della collezione di documenti a una collezione di record, trasformare quindi ogni documento in una rappresentazione strutturata. Il metodo più immediato per l'applicazione di questa trasformazione è l'utilizzo di tutte le parole presenti nella collezione (dizionario) come attributi dei record rappresentanti i documenti, assegnando a ogni attributo, in ogni record, la presenza o l'assenza (o eventualmente l'occorrenza) del termine all'interno del documento relativo.
In altre parole, il text mining unisce la tecnologia della lingua con gli algoritmi del data mining.

1.2 Applicazioni del text mining

la classificazione è un processo naturale e fondamentale in molte attività umane. Per questo motivo possono essere trovate applicazioni di categorizzazione testuale in molti problemi pratici.

Di seguito saranno descritte alcune delle applicazioni più comuni per la categorizzazione del testo [40].

Document indexing

Il document indexing è il processo che descrive il contenuto di un documento tramite l'assegnazione di un insieme finito di parole (o frasi) chiave; questo insieme di parole chiave

è definito *vocabolario controllato*. Questo compito è stato tradizionalmente eseguito manualmente dai bibliotecari per facilitare il reperimento dei libri in una biblioteca. Dagli anni '60 però [57] è stata avanzata una ricerca per rendere questo processo automatico.

L'indicizzazione automatica può essere vista come una classificazione testuale se ogni keyword è trattata come una categoria, pertanto possono essere utilizzate tutte le tecniche di text classification note in letteratura.

Document organization

L'indicizzazione tramite un vocabolario controllato è un'istanza del problema più generale riguardante la document organization. L'organizzazione dei documenti utilizza tecniche di classificazione testuale per catalogare documenti in una predefinita struttura di classi.

Document filtering

Nel document filtering esistono solo due classi disgiunte: *rilevante* o *irrilevante*. I documenti irrilevanti vengono eliminati mentre quelli rilevanti sono catalogati in una specifica destinazione.

Un classico esempio di questa tecnica è il filtraggio delle E-mail catalogate come spam.

Word sense disambiguation

Le tecniche di word sense disambiguation (WSD) precisano il senso semantico di una parola ambigua in un documento, osservando il contesto in cui la parola è utilizzata (es. la parola 'bank' può essere utilizzata con significati diversi in 'river bank' e 'financial bank'). La WSD può essere utilizzata per aumentare l'efficacia del document indexing.

Classificazione di pagine web

Recentemente, con la crescita del web, la text categoriza-

tion ha riscosso molto interesse per la possibile applicazione della classificazione automatica dei documenti, riferita alle pagine web, catalogate in gerarchie di classi. Questa catalogazione gerarchica rende estremamente più semplice la navigazione e la ricerca dei contenuti web di interesse per gli utenti.

La catalogazione di pagine web ha due aspetti peculiari: la natura ipertestuale del contenuto dei documenti e la struttura gerarchica dell'insieme delle categorie.

1.3 Definizioni

Di seguito verranno introdotte le definizioni e le notazioni matematiche utilizzate in questa tesi.

1.3.1 Gerarchia (H)

Una gerarchia $H = (N, E)$ è definita come un grafo direzionato aciclico consistente in un insieme di nodi N e un insieme di coppie ordinate chiamate archi $(N_p, N_c) \in E \subseteq \{N \times N\}$. La direzione di un arco (N_p, N_c) è definita dal nodo padre N_p al nodo figlio diretto N_c, detta anche percorso diretto da N_p a N_c ($N_p \rightarrow N_c$).

Un percorso $N_p \rightarrow N_c$ di lunghezza n è quindi un insieme ordinato di nodi $\{N_1 \rightarrow N_2 ... \rightarrow N_n\} \subseteq N$ dove ogni nodo è il padre del nodo successivo. In una gerarchia H aciclica, dato il percorso $N_p \rightarrow N_c$ non esiste il percorso inverso $N_c \rightarrow N_p$.

Esiste inoltre un nodo denominato *root* (N_r) di un grafo H, che non ha padri. I nodi che non hanno figli sono chiamati *foglie*. Un esempio di gerarchia è in Fig. 1.1.

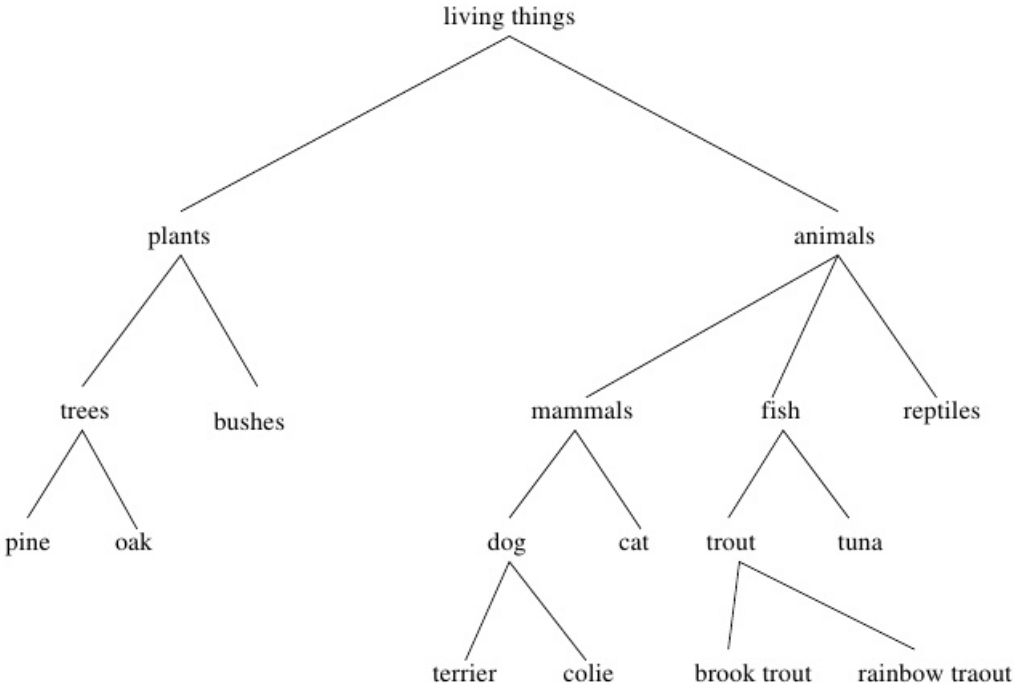

Figura 1.1: Esempio di gerarchia.

1.3.2 Categoria (C)

Ogni nodo N_i all'interno di una gerachia H equivale a una categoria C_i ($C \equiv N$). Ogni categoria C_i consisiste in un insieme di documenti $D \in C_i$.

1.3.3 Documento (D)

I documenti di una gerarchia H contengono il contenuto testuale e sono assegnati a una o più categorie.

Ogni documento è rappresentato come un vettore di termini ottenuti dalla fase di preprocessing.

1.3.4 Text classification

La classificazione testuale è il processo per cui si cerca una approssimazione per la *funzione target*:

$$\Phi : D \times C \rightarrow \{T, F\}$$

dove D è un insieme di documenti e C è un insieme di categorie predefinite. La funzione target ha valore T se un documento $D_j \in D$ viene assegnato alla categoria $C_i \in C$, mentre il caso contrario comporta il valore F. Φ descrive come i documenti possono esseere classificati.

La funzione approssimata $\widetilde{\Phi} : D \times C \rightarrow \{T, F\}$ è denominata *classificatore* e deve coincidere il più possibile con Φ. Tanto più $\widetilde{\Phi}$ sarà simile a Φ, tanto più sarà efficace la classificazione.

Per l'applicazione considerata e creata in questa tesi si sono fatte queste assunzioni:

- La funzione target Φ è descritta da una collezione di documenti. Tale collezione è definita attraverso un insieme di categorie C e di documenti D e dall'assegnamento di tali categorie ai vari documenti. Nessuna altra informazione è fornita per descrivere Φ. La collezione di documenti è anche chiamata classification scheme.

- I documenti D sono rappresentati da un contenuto testuale che ne descrive la semantica.

- Le categorie C sono label simboliche per i documenti, che non forniscono informazione aggiuntiva al contenuto degli stessi.

- I documenti $D_j \in D$ a seconda dell'organizzazione della collezione, possono essere assegnabili a una e una sola categoria (single-label classification) o ad un numero arbitrario

di categorie (multi-label classification). In quest'ultimo caso, il problema si può scomporre in classificazioni binarie multiple, una per categoria.

1.3.5 Hierarchical text classification

Per l'applicazione gerarchica viene aggiunto il concetto di gerarchia H alla definizione della funzione target Φ:

$$\Phi(D, C_i) = T \Rightarrow \Phi(D, C_j) = T$$

se

$$C_j \rightarrow C_i; C_j, C_i \in H$$

H è una struttura gerarchica che definisce le relazioni tra le categorie, l'assunzione alla base di questo vincolo è che $C_i \rightarrow C_j$ definisce una relazione per cui C_i riferisce a un argomento più ampio rispetto a C_j. La relazione $C_i \rightarrow C_j$ è asimmetrica (es. ogni cane è un animale, ma non tutti gli animali sono cani) e transitiva (es. il labrador è un cane e i cani sono animali, quindi il labrador è un animale). Lo scopo della classificazione è approssimare la funzione target sconosciuta utilizzando la collezione di documenti, rispettando i vincoli imposti dalla struttura gerarchica H.

Possono essere distinte alcune proprietà della classificazione gerarchica:

- Struttura della gerarchia: data la definizione della gerarchia H, possono essere distinti due casi: una struttura ad *albero*, in cui ogni categoria (esclusa la root) ha esattamente un nodo padre; un *grafo diretto aciclico*, in cui una categoria può avere più di un padre.

- Categorie contenenti documenti: una proprietà di base è il livello del grafo in cui i documenti sono assegnati alle categorie, possono essere previsti il caso in cui i documenti

siano assegnati solo alle foglie dell'albero (gerarchia virtuale), oppure che possano essere assegnati anche ai nodi interni.

- Assegnamento dei documenti: come nel caso di classificazione flat, possono essere distinti i casi di classificazione multi-label o single-label.

1.4 Text Clustering

Il clustering è una tecnica di data mining che cerca di posizionare gli elementi in gruppi di dati correlati, senza conoscenze pregresse sulle caratteristiche dei diversi gruppi. La somiglianza dei dati è ottenuta tramite una funzione di similarità. I metodi tradizionali di clustering venivano storicamente applicati su dati numerici, da qualche tempo il problema è stato sottoposto a dati categorici. Esempi di applicazioni pratiche di clustering su dati categorici possono essere:

- Organizzazione e browsing di documenti, organizzazione gerarchica dei documenti coerententemente con il contenuto degli stessi.
- Organizzazione dei documenti in capitoli coerenti.
- Classificazione dei documenti.

I metodi prevalenti di clustering possono essere categorizzati in differenti maniere:

- Gerarchici o non gerarchici: si differsificono in base ai cluster prodotti, nel primo caso viene prodotta una gerarchia, nel secondo tutti i cluster sono allo stesso livello.
- Partitivi o aggromerativi: nei primi si parte con un unico grande cluster e lo si divide in cluster più piccoli iterativamente, nel secondo ogni documento è un cluster e man

mano si combinano i più simili unendoli fino ad ottenere il numero di cluster desiderato.

- deterministici o probabilistici: i primi assegnano ogni documento a un cluster (assegnamento hard) mentre i secondi producono la probabilità di appartenenza di un documento a ogni cluster(soft).

- Incrementali o batch: gli incrementali impiegano uno o pochi passi sul dataset per decidere a che cluster assegnare un documento, mentre i batch iterano molte volte il dataset e gradualmente cambiano gli assegnamenti migliorando la funzione obiettivo.

Un documento di testo può essere rappresentato in forma di dati binari segnando la presenza o meno di una parola nel testo creando così un vettore binario, utilizzando tale approccio si rendono applicabili algoritmi già noti in letteratura creati per dati quantitativi, come ad esempio il k-means. Un ulteriore accorgimento potrebbe essere quello di notificare per ogni parola non solo la presenza, ma la frequenza della stessa nel documento.

I dati testuali sono distinti da alcune caratteristiche:

- Alta dimensionalità della rappresentazione del lessico (ovvero le diverse parole contenute nell'insieme dei documenti) ma dati utilizzati effettivamente da ogni documento molto sparsi. Mediamente si può avere un lessico di 10^5 parole ma solo qualche centinaia effettivamente contenute in ogni documento.

- Molte parole in ogni documento sono correlate fra loro: il numero di concetti è molto minore rispetto al numero di parole.

- il numero di parole contenute nei diversi documenti è altamente eterogeneo.

Per ottenere un processo di clustering efficace, la frequenza delle parole deve essere normalizzata in termini di frequenza nel documento e frequenza nell'intera collezione. Una comune rappresentazione è la *vector-space based TF-IDF rapresentation* [2]: la frequenza di un termine è normalizzata in base alla frequenza dello stesso negli altri documenti; questo metodo minimizza l'importanza dei termini più comuni ai documenti, dando maggiore rilevanza nella classificazione ai termini più discriminativi e meno frequenti. Inoltre viene applicata una trasformazione sub-lineare per evitare che un termine molto frequente in un documento domini gli altri.

1.4.1 Feature selection

La qualità di ogni processo di data-mining dipende molto dalla rumorosità delle feature, ad esempio l'utilizzo degli articoli non è molto utile. Alcuni metodi noti in letteratura per la selezione non supervisionata delle feature sono:

- **Document frequency-based selection:** è il metodo più semplice di feature selection, utilizza la frequenza nel documento di una parola per decidere se eliminarla. In particolare vengono denotate con il termine stop words tutte quelle parole (come ad esempio gli articoli) che non sono in alcun modo discriminative; questi termini vengono elimanati sistematicamente, basandosi su una lista.

- Una tecnica più agressiva è il **term-strength** [58]: l'idea di base è estendere le tecniche utilizzate nel learning supervisionato per i casi non supervisionati. E' essenzialmente usato per capire quanto una parola è significativa per identificare due documenti correlati. Bisogna chiaramente stabilire come definire i documenti correlati. Un semplice approccio sarebbe l'utilizzo di un feedback utente, ma è chiaramente

meglio utilizzare un metodo non supervisionato. Ad esempio è possibile stabilire la relazionalità di due documenti tramite la funzione coseno [2]. La funzione s(t) di term strength è essenzialmente usata per misurare quanto una parola è utile nell'identificare due documenti correlati:

$$s(t) = \frac{\text{numero di coppie in cui t è in entrambe}}{\text{numero di coppie in cui t è solo nel primo}}$$

Per eseguire il pruning si confronta la *s(t)* con la strength attesa per un termine distribuito casualmente nella collezione. se la s(t) non è almeno 2 volte quella casuale, il termine viene rimosso dalla collezione. Il vantaggio di questa tecnica è che non necessita di una supervisione (o training) iniziale; questo approccio è particolarmente adatto per clustering basato su similarità.

- **Ranking basato su entropia[9]:** la qualità di un termine è misurata come riduzione dell'entropia nel caso in cui si eliminasse il termine stesso. La complessità di questo approccio è $O(n^2)$, troppo elevata per la mole di dati dei problemi di text-clustering, in [9] sono proposte alcune varianti più efficienti.

- **Term contribution:** questo approccio si basa sul fatto che il text-clustering è altamente dipendente dalla similarità dei documenti, quindi il contributo di un termine è visto come il contributo alla similarità dei documenti. Il problema di questo metodo è che favorisce le parole più frequenti rispetto a quelle più specifiche e discriminative.

Oltre alla funzione di selezione si aggiungono funzioni di trasformazione che migliorano la qualità della rappresentazione del documento, ad esempio modificando la correlazione tra parole influenzandola in modo da creare feature, che corrispondano ai concetti espressi. Queste tecniche sono spesso usate prima

del processo di clustering. Alcuni di questi metodi di *feature transformation* sono:

- **Metodi basati su LSI (latent semantic indexing)[13]:** Il metodo più comune consiste nel trasformare i dati in dataset di dimensioni più piccole rimuovendo il rumore. LSI utilizza una matrice contenente la frequenza dei termini *i-esimi* nel documento *j*. Un'eccellente caratteristica è che la riduzione della dimensione riduce anche il rumore dovuto a sinonimi, dando più valore ai concetti semantici dei dati. Un metodo simile è il PCA, che riduce la dimensionalità utilizzando una matrice di covarianza. Una variante probabilistica è il PLSA [25].

- **Fattorizzazione in matrici non negative:** il documento viene rappresentato in un nuovo sistema di coordinate basato sull'analisi della matrice dei termini del documento, il NMF è un numero indicante le differenze critiche dallo schema LSI al punto di vista concettuale. In LSI il nuovo sistema consiste in un insieme di vettori ortonormali, in NMF i vettori nel sistema di base corrispondono direttamente agli argomenti dei cluster. In questo modo l'appartenenza ad un cluster di un documento può essere determinata esaminando la componente maggiore lungo uno dei vettori. Si può esprimere il vettore del documento *a* come un'approssimazione della combinazione lineare (non negativa) dei vettori di base. Si può dimostrare [49] che la funzione obiettivo, continuamente migliorata dalle regole di aggiornamento, converge a una soluzione ottima. Esiste anche una tecnica analoga, chiamata concept factorization [45].

1.4.2 Distance-based partitioning Algorithms

Gli algoritmi di clustering distance-based sono creati usando una funzione di similarità che misuri la vicinanza tra oggetti testuali. La funzione più classica si similarità è il coseno. Dati i vettori, smorzati e normalizzati, relativi alle frequenze dei termini appartenenti a due documenti U e V, la funzione di similarità è definita:

$$cosine(U, V) = \frac{\sum_{i=1}^{k} f(u_i) \cdot f(v_i)}{\sqrt{\sum_{i=1}^{k} f(u_i)^2} \cdot \sqrt{\sum_{i=1}^{k} f(v_i)^2}} \tag{1.1}$$

Un metodo per assegnare un peso ai termini è il BM25 [10] dove il TF normalizzato non affronta solo la lunghezza normalizzata, ma anche un upper bound che aumenta la robustezza, in quanto evita il matching troppo facile tra ogni termine. Un documento può essere anche rappresentato con una distribuzione di probabilità sulle parole, e la somiglianza può essere quindi misurata tramite l'entropia incrociata.

Per clusterizzare piccoli segmenti di testo il keyword matching esatto non funziona molto bene, una strategia generale per risolvere questo problema è di espandere la rappresentazione del testo sfruttando i documenti correlati [44].

I più classici ed utilizzati esempi di algoritmi distance-based sono:

- **K-metoid clustering:** si usa un insieme di punti dai dati originali come ancore e attorno a essi si creano i cluster; l'aspetto chiave è determinare l'insieme ottimo per rappresentare il documento. L'algoritmo lavora iterativamente, il set dei k rappresentativi viene migliorato con interscambi randomizzati, a ogni iterazione si cambia casualmente un valore dal set di ancoraggio e lo si sostituisce da uno preso a caso nella collezione, se questo scambio aumenta la

funzione obiettivo. I due grossi svantaggi sono che servono molte iterazioni per arrivare alla convergenza (dovuto ai k-metoid in generale) e non è molto efficace con i dati sparsi tipici dei dati testuali

- **k-means clustering:** anch'esso utilizza un insieme di k rappresentativi, ma non devono necessariamente appartenere ai dati originali.

1.4.3 Clustering basato su parole o frasi

Un problema del text-mining è che se si visualizza il corpo come n documenti contenenti d termini, quindi una matrice $n \cdot d$ dove una entry individua la frequenza del *j-esimo* termine nell'*i-esimo* documento, questa matrice è molto sparsa. Clusterizzare le righe significa clusterizzare i documenti, mentre con le colonne si clusterizzano le parole; i due problemi sono strettamente correlati.

Frequent word patterns clustering [3]: è una tecnica molto usata in letteratura per determinare il pattern più rilevante in dati transazionali. L'idea di base è di utilizzare un insieme di termini frequenti di bassa dimensione, ciò implica che l'insieme dei termini frequenti sia una descrizione del cluster, corrispondente a tutti i documenti che contengono questi termini frequenti.

Un algoritmo che utilizza i word patterns per il document clustering [11] consiste in due fasi:

1. Determinare i cluster di parole dai documenti in modo che un documento sia rappresentabile in termini di cluster di parole.

2. Si fa il clustering dei documenti utilizzando la rappresentazione dei documenti in cluster di parole creata in precedenza.

Si ha un miglioramento dell'efficienza in quanto viene eliminato il rumore. Per scovare cluster di parole interessanti si può pensare ai cluster che caratterizzano il senso delle parole o i concetti semantici [24]; un *n-gram class language model* può essere utilizzato per minimizzare la perdita di informazione reciproca tra parole adiacenti).

Una tecnica di clustering simultaneo è il *co-clustering* [26, 27], questo approccio può essere considerato equivalente al problema di ri-ordino delle matrici per creare un rettangolo denso di entries diverse da 0. È definito come una coppia di mappe da righe a indice di riga nel cluster e da colonna a indice di colonna nel cluster, queste mappe sono determinate simultaneamente.
Il concetto di co-clustering è un'applicazione naturale dell'idea di rappresentare il dominio dei dati come una matrice sparsa ad alta dimensionalità.

Un'ulteriore tecnica può essere il *clustering con frasi frequenti*; la principale differenza con gli altri metodi è che tratta un documento come una serie di parole e non una serie di caratteri. Si utilizza quindi un metodo di indicizzazione per organizzare le frasi nella collezione di documenti, per poi utilizzare questa nuova organizzazione durante la creazione i cluster. I passi dell'algoritmo sono i seguenti:

1. Si eliminano suffissi e prefissi delle parole e i plurali diventano singolari, si eliminano le *stop-words*.

2. Si identificano i *cluster base*, definiti come le frasi frequenti nella collezione; vengono rappresentati come un albero di suffissi, ovvero un albero che contiene ogni suffiso della collezione, ogni nodo del tree rappresenta un gruppo di documenti e una frase comune a tutti questi documenti.

3. I vari documenti compaiono più volte nel *suffix tree* indicizzati da varie frasi, il terzo step è di unire i cluster sulla

similarità dei propri insiemi di documenti. La formula per trovare la similarità BS per due cluster P e Q, dipende dal fatto che i cluster abbiano almeno il 50% di documenti in comune, ed è così definita:

$$BS(P,Q) = \lfloor \frac{|P \cap Q|}{max\{|P|,|Q|\}} + 0.5 \rfloor$$

1.4.4 Clustering dei contenuti web

Quando si applica il clustering a contenuti web bisogna prestare attenzione sia al contenuto testuale, sia ai link contenuti in quanto anche questi possono contenere conoscenza realativa al documento in esame.

Un approccio generale [5] per portare l'informazione di un link all'interno del processo di clustering, consiste nell'utilizzo di una combinazione tra k-means sugli attributi di testo e le probabilità Bayesiane per i side-attribute (link). L'idea è di identificare gli attributi più utili per il processo di clustering.

Altri metodi per combinare testo e link [28] sono di usare le informazioni dei link per poter inclinare il peso dei termini, il vantaggio è che si possono utilizzare le tecniche standard di clustering. Oppure avere un approccio graph-based dove si usano direttamente i link nel processo, in questo caso l'idea è di modellare la probabilità che un documento appartenga a un cluster per un particolare insieme di link e contenuto. Una tecnica *Markov random field (MFR)* viene utilizzata per implementare il clustering e una tecnica iterativa (*relaxion labeling*) è usata per produrre la massima verosimiglianza dei parametri di MRF.

1.5 Clustering Gerarchico

Gli algoritmi gerarchici incrementali di clustering per dati testuali sono importanti per organizzare i documenti di uno streaming

on-line di fonti (per esempio news o blog). I benefici dell'uso di un clustering gerarchico, soprattutto in casi di fonti web, sono molteplici [14]; esso descrive le relazioni fra i gruppi di e il giusto numero di cluster indocumenti permettendo di sfogliare gli specifici argomenti di interesse. Inoltre vi è anche una ragione tecnica: trovar un set di documenti mal formati e dove non si conosce l'informazione di interesse per l'utente non è un problema facile da risolvere, utilizzando un clustering gerarchico, l'utente può navigare nei livelli di gerarchia, circumnavigando così il problema.

La valutazione di un cluster è comunemente definita come:

$$p_k = \frac{\max_c CF_k(c)}{N_k} \tag{1.2}$$

Dove c è indice della classe, k indice del cluster, $CF_k(c)$ è il numero di documenti della classe c che sono in k (o frequenza di c in k) e N_k numero di elementi di classe k.

In letteratura manca però una definizione di valutazione per la gerarchia dei cluster; un possibile metodo per valutare la qualità della gerarchia consiste nell'osservazione di quanto spesso un cluster figlio, contenga una classe derivata da una assegnata al cluster padre. In altre parole, si valuta quanto la gerarchia di cluster rispetti la gerarchia delle classi.

1.5.1 Algoritmi agglomerativi gerarchici incrementali

L'idea generale del clustering agglomerativo è di unire documenti in cluster basati sulla similarità; si può visualizzare il risultato come un dendogramma avente come foglie i documenti e come nodi interni i merge tra i vari documenti. Alcuni metodi per il merging di gruppi di documenti sono:

- **single linkage clustering [31]:** si fa merge tra i due documenti che hanno più alta similarità rispetto tutte le altre

coppie; questo metodo è estremamente efficiente da applicare nella pratica. Il maggiore svantaggio è il fenomeno di chaining, ovvero portare nello stesso cluster documenti eterogenei (se A è simile a B e B a C, non è detto che A sia simile a C).

- **group-average linkage clustering:** la similarità tra due cluster è data dalla media tra le similarità delle coppie (nei cluster), è più inefficiente del single linkage clustering ma è molto più robusta in termini di qualità.

- **complete linkage clustering:** la similarità tra due cluster è data dal caso peggiore di similarità fra ogni coppia di documenti nei cluster; questa tecnica ha complessità $O(n^2)$ nello spazio e $O(n^3)$ nel tempo, nel caso di dati testuali il numero è però significativamente minore in quanto un vasto numero di coppie ha similitudine 0.

In letteratura sono già noti alcuni metodi di clustering gerarchico incrementali quali *Cobweb* e *Classit*, ma non sono adatti per il clustering testuale; per estendere tali algoritmi ai problemi testuali [14] si può pensare di modellare la frequenza della presenza di ogni parola con l'uso di una distribuzione *multi-poisson*. Si modificano gli algoritmi *Cobweb/Classit*, sostituendo l'utilizzo distribuzioni normali, standard per l'algoritmo *Classit*, con l'utilizzo di distribuzioni *Katz* [12], che assegnano la probabilità che una parola i occorra k volte in un documento tramite la formula:

$$P(k) = (1 - \alpha)\delta_k + \frac{\alpha}{\beta + 1}(\frac{\beta}{\beta + 1}) \tag{1.3}$$

dove $\beta = \frac{collectionFrequency - documentFrequency}{documentFrequency}$, $\alpha = \frac{1}{\beta} \times \frac{collectionFrequency}{N}$ e $\delta_k = 1$ se $k = 0$ o 0 altrimenti. È stato dimostrato tramite vari test in [14] che tale variante ha migliorato le prestazioni degli algoritmi originali.

Un contributo di grande utilità di tale modifica è la separazione dell'attributo di distribuzione e la stima di questo parametro nella struttura di controllo di *Classit*, in questo modo è possibile sostituire in seguito la distribuzione Katz (o quella normale) con una distribuzione più adatta ai dati di interesse.

1.5.2 Scatter-Gather method

Per quanto riguarda gli algoritmi partitivi/agglomerativi, i metodi gerarchici tendono ad essere robusti ma poco efficienti, $O(n^2)$, mentre i k-means sono molto efficienti ma non troppo efficaci. Il metodo *Scatter-Gather* [29] combina entrambi i metodi, i gerarchici su un campione di documenti per trovare un robusto set iniziale, utilizzando poi k-means partendo da questo set. Questo approccio prevede che i documenti siano rappresentati come una bag-of-words, comunemente utilizzata come vettore binario rappresentante l'assenza o la presenza di una parola nel documento, esistono però varianti utilizzanti le frequenze delle parole.

L'algoritmo prevede quindi tre fasi:

1. cerca k centroidi.
2. assegna ogni documento nella collezione a un centroide, algoritmo più semplice e efficiente è l'*assign-to-nearest*.
3. raffina le partizioni costruite in precedenza.

I k centroidi iniziali possono essere trovati in diverse maniere:

- **Buckshot:** dato k, numero di cluster da trovare e n numero di documenti nella collezione, si usano inizialmente $\sqrt{k \cdot n}$ centroidi random e ad essi viene applicato un algoritmo standard di clustering agglomerativo gerarchico. Se si usano algoritmi quadratici per questa fase la complessità è di $O(k \cdot n)$.

- **Fractionation:** viene divisa la collezione in n/m bucket di grandezza $m > k$; un algoritmo agglomerativo è applicato a ogni bucket per ridurre i termini di un fattore v, ottenendo così un totale di $v \cdot n$ punti agglomerati; il processo viene ripetuto iterativamente trattando ogni agglomerato come un singolo record e si termina quando si hanno k seeds. Anche questo algoritmo, considerando $v < 1$ e $m = O(k)$ ha complessità $O(k{\cdot}n)$. Fractionation è applicato a un gruppo random di documenti negli n/m bucket, una modifica potrebbe essere di ordinare i documenti sulla presenza della *j-esima* parola più comune (per esempio la terza, ovvero media frequenza).

L'algoritmo standard prevede, una volta ottenuto il set iniziale, di applicare il *k-means*. Alcune modifiche per migliorare l'algoritmo possono essere:

- **Split operation:** con questo approccio si vuole migliorare la granularità applicando la procedura *buckshot* a ogni documento in un cluster usando $k = 2$ e poi riclusterizzando attorno a questi centri, complessità $O(k{*}n_i)$, può non essere necesarrio dividere tutti i gruppi ma solo quelli incoerenti che contengono documenti di natura diversa. Per misurare la coerenza di un gruppo si calcola la self-similarity di un cluster, ovvero una misura indicante la coerenza tra i documenti contenuti. Un metodo di calcolo di questa misura è il *single-linkage clustering*.

- **Join operation:** l'idea di base è di unire i cluster simili; si valutano le topical words di ogni cluster e due cluster sono considerati simili se hanno un significante overlap di topical words.

L'approccio Scatter-Gather è usato per la navigazione gerarchica di documenti simili, in particolare l'utente ha la possibilità

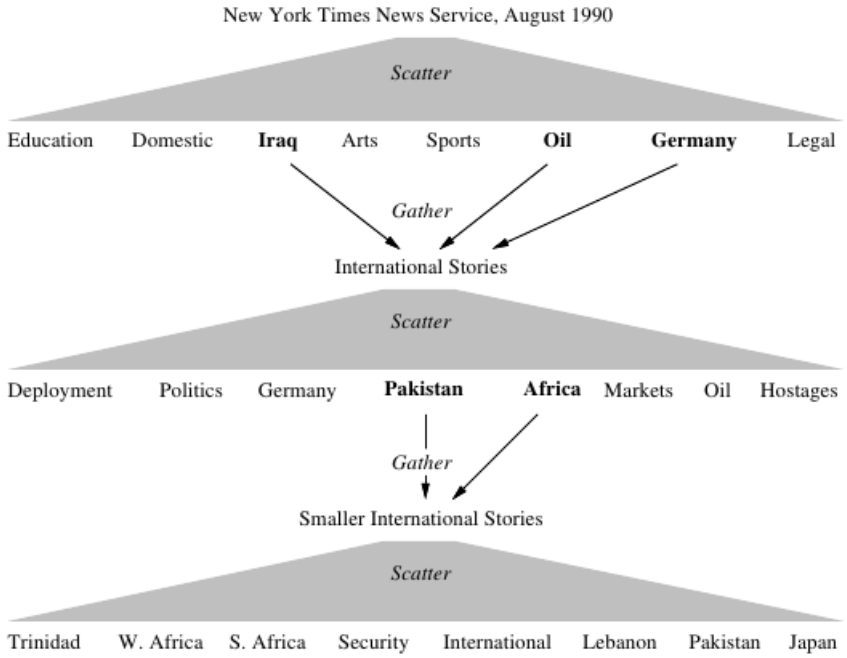

Figura 1.2: Navigazione gerarchica dei cluster tramite Scatter-Gather.

di navigare la gerarchia dei cluster interattivamente per capire gli argomenti dei vari livelli, Fig. 1.2

In [30] si propone un metodo per l'interazione nella navigazione: si hanno delle keywords, che corrispondono a uno o più cluster, tramite le quali l'utente può scegliere di aumentare la granularità on-line, unendo i cluster selezionati e poi riclusterizzando. Il presupposto per questo approccio è l'ipotesi di raffinatezza dei cluster, ovvero che i documenti appartenenti allo stesso cluster in una granularità più fine sono contenuti nello stesso cluster anche con granularità più grossolana; in pratica si possono vedere i nodi all'interno dell'albero della gerarchia come meta-documenti corrispondenti alla concatenazione dei documenti nelle foglie.

Un'altro algoritmo di on-line clustering è LAIR2 [47], più veloce dell'algoritmo buckshot.

Measure	Buckshot	LAIR2	Diff	t	Pr
Precision	0.603	0.651	0.047	0.695	0.489
Recall	0.061	0.082	0.012	1.373	0.173
F_1	0.172	0.223	0.021	1.367	0.175

Figura 1.3: Comparazione dei vari metodi utilizzati all'interno della clusterizzazione Scatter-Gather.

Un'ulteriore possibilità è usare un clustering gerarchico *a-priori*, con lo svantaggio che non si può unire o riclusterizzare i gruppi dell'albero della gerarchia al volo quando l'utente lo richiede.

Un problema del metodo Scatter-Gather è che il centroide di un cluster è formato da una concatenazione dei documenti che alloca, se questi sono in numero molto alto si hanno molti termini nel centroide e ciò comporta un rallentamento inevitabile nei calcoli critici, come la valutazione della similarità tra centroidi; una soluzione [4] propone di usare il concetto di proiezione per ridurre la dimensionalità della rappresentazione dei documenti, sono proposte 3 tipi di proiezione:

- **Proiezione globale:** la dimensionalità è ridotta rimuovendo il termine meno importante nella collezione dei dati, in ordine di frequenza.

- **Proiezione locale:** i termini di ogni centroide sono troncati separatamente, rimuovendo il termine meno importante di ogni centroide.

- **Latent semantic indexing:** il document-space è trasformato con tecniche LSI e il clustering è applicato al document-space trasformato, le tecniche LSI possono essere applicate sia globalmente sia localmente.

1.6 Text classification

La text classification (o text categorization) è il processo che automaticamente assegna una o più categorie a un documento di testo. Formalmente il problema è definito come segue: si ha un insieme di documenti di training $D = X_1, ..., X_n$ tali che ogni record è etichettato con il valore di una classe presa da un insieme di k valori discreti e differenti.

Il data-set di training è utilizzato in modo da costruire un modello di classificazione che relazioni le caratteristiche dei record a una delle etichette. Per una data istanza test, di classe sconosciuta, il modello di training è utilizzato per predire a quale classe appartiene l'istanza. Nelle versioni hard è assegnata una etichetta all'istanza, nelle soft è assegnato un valore di probabilità di appartenenza alle classi.

Alcuni esempi di domini applicativi per la text classification sono:

- Organizzazione e filtraggio delle news (text-filtering)
- Organizzazione e ricerca dei documenti, gerarchicamente organizzati in collezioni
- Opinion Mining
- Email/spam Classification

Esistono varie tecniche di implementazione per la text classification, in particolare è possibile utilizzare la maggior parte dei metodi creati per dati quantitativi, in quanto gli attributi dei documenti sono dati quantitativi basati sulla presenza e frequenza dei termini in essi. Alcuni metodi utilizzati comunemente per la classificazione testuale sono:

- **Decision tree:** creati con l'utilizzo di divisioni gerarchiche nello spazio dei dati utilizzando le proprietà del testo.

- **Rule-Based classifiers:** viene determinato il pattern di parole che sono più probabilmente correlate alle differenti classi; si costruisce un insieme di regole, utilizzate per mettere in pratica la classificazione.

- **SVM (Support Vector machine):** provano a partizionare lo spazio dei dati con l'uso di funzioni lineari o non, la chiave per la classificazione è di determinare il confine ottimo tra le diverse classi.

- **Neural network classifiers:** le reti neurali sono utilizzate in vari domini della classificazione, nel caso di text classification vengono adattati all'uso delle caratteristiche delle parole, spesso sono utilizzati congiuntamente agli SVM.

- **Bayesian (generative) classifiers:** si cerca di costruire un classificatore probabilistico, classificando il testo in base alla probabilità che un documento appartenga a una classe, in base alle parole presenti nello stesso.

1.6.1 Feature selection

Prima di ogni processo di classificazione alcuni dei passi fondamentali da eseguire sono le decisioni su come rappresentare un documento e su quali tecniche di feature selection applicare. La feature selection è particolarmente importante per i problemi di text classification sia per l'alta dimensionalità intrinseca dei dati, sia per l'esistenza di dati inutili al fine della caratterizzazione dei documenti (rumore).

In generale il testo di un documento può essere rappresentato in vari modi:

- Una *bag of word*, in cui un documento è rappresentato da un vettore di parole, in cui a ognuna è associata o la presenza/assenza, nel caso binomiale, o la frequenza dell'oc-

corenza, nel caso multinomiale, all'interno del documento stesso.

- Rappresentare il testo direttamente come stringhe, ogni documento è una sequenza di parole.

- Nei VSM (Vector Space Model) i documenti sono rappresentati da vettori composti da un elemento reale valutato per ogni termine nel documento; gli elementi del vettore sono generalmente pesati e normalizzati mediante modelli basati su *TFIDF*.

  I modelli *TFIDF* prevedono che a ogni termine t_i venga associato un peso tc_{ij} in base alla frequenza tf_{ij} del termine nel documento j-esimo e la rarità della parola negli altri documenti idf_i. Si definisce quindi:

  $$tc_{ij} = tf_{ij} \cdot idf_i \tag{1.4}$$

  con tf_{ij} numero delle occorrenze del termine t_i nel documento d_j e $idf_i = \log(\frac{\text{Numero Doc. in D}}{\text{Numero Doc. in D contenenti } t_i})$.

  La forza di questa funzione è data dal fatto che vengono favoriti i termini con maggiore presenza intra-categoria (tf) e che hanno però bassa frequenza extra-categoria (idf), ovvero i termini più discriminanti in base alla tematica delle categorie.

  Considerando che i documenti possono avere lunghezza variabile, i pesi relativi ad un termine t_i di ciascun vettore-concetto d_j sono normalizzati secondo l'ampiezza del vettore:

  $$ntc_{ij} = \frac{tc_{ij}}{\sum_i tc_{ij}} \tag{1.5}$$

  La valutazione di similarità tra due documenti avviene tramite il calcolo del coseno dell'angolo tra i vettori dei documenti.

Un'ulteriore decisione nella creazione di un feature-set per rappresentare i documenti è:

- Avere un feature-set globale, uguale per tutti i documenti appartenenti alle varie categorie del dominio.
- Avere feature-set diversi per le varie categorie, in particolare nel caso di classificazione gerarchica. Documenti di una categoria c e delle proprie sottocategorie possono essere rappresentati per mezzo dello stesso feature set, in modo da poter avere un classificatore diverso per ogni livello nella gerarchia, che assegni l'istanza da classificare a c o a una delle sue sottocategorie. Ogni sottocategoria però può avere un set di feature diverso rispetto le altre *sorelle*, in questo modo vengono definite diverse rappresentazioni per ogni documento in base al livello del'albero che si sta valutando.

Per la pulizia dei dati le strategie standard consistono nell'eliminare ogni segno di punteggiatura, numero, parola con meno di 3 lettere e tag html (se si considerano documenti web). Inoltre solo i token rilevanti sono utilizzati nel feature set, si applicano quindi i metodi standard di pre-processing:

- Rimozione stop words, come articoli, avverbi, preposizioni e tutte le parole troppo frequenti per poter dare un contributo alla categorizzazione.
- Determinare le parole derivate da altre, come sinonimi; utilizzando lo stesso token per gli stessi concetti, anche se espressi in parole diverse nei documenti.

La fase di pre-processing riduce il numero di token estratti, ciò è fondamentale per una efficiente e efficace fase di feature selection. In letteratura sono noti molti metodi di feature selection, che si basano su quattro possibili dipendenze tra un termine w e una categoria c_i:

1. (w, c_i): w e c_i co-occorrono.

2. $(w, \neg c_i)$: w occorre senza c_i ;

3. $(\neg w, c_i)$: c_i occorre senza w;

4. $(\neg w, \neg c_i)$: nè w nè c_i occorrono.

La caratteristica più importante [6] per la misura di bontà di una feature selection è data dal fatto che favorisca feature comuni e consideri le caratteristiche del dominio e dell'algoritmo, ciò focalizza principalmente l'interesse nelle prime due dipendenze.

Alcuni metodi di feature selection [32] per la text classification sono:

- **Gini index:** uno dei metodi più comuni per quantificare il livello di discriminazione di una feature è l'uso di una misura detta *gini-index*. Dati $p_1(w)...p_k(w)$ frazioni della presenza nelle k diverse classi per la parola w, ovvero $p_i(w)$ è la probabilità condizionata che un documento appartenga alla classe i considerato il fatto che contiene la parola w; notando quindi che:

$$\sum_{i=1}^{k} p_i(w) = 1$$

  Allora il gini-index per la parola w, denotato con $G(w)$ è definito come segue:

$$G(w) = \sum_{i=1}^{k} p_i^2(w) \tag{1.6}$$

  $G(w)$ indica il potere discriminativo della parola w: più è alto, maggiore è la discriminazione. Il problema di questo approccio è che inzialmente la distribuzione delle classi non

è accurata e può non riflettere correttamente la reale potenza di discriminazione delle parole. Una possibile modifica per ovviare parzialmente a questo problema si ha inserendo una normalizzazione nelle $p_i(w)$.

- **Information Gain:** un'altra misura comunemente usata per la selezione delle text-feature è l'information gain (o entropia). Data P_i la probabilità globale della classe i, e $p_i(w)$ la probabilità che il documento appartenga alla classe i considerato il fatto che contiene la parola w, si definisce $F(w)$ la frazione dei documenti contenenti la parola w. La misura dell'information gain è definita:

$$I(w) = -\sum_{i=1}^{k} P_i \cdot \log(P_i) + F(w) \cdot \sum_{i=1}^{k} p_i(w) \cdot \log(p_i(w)) + \\ +(1 - F(w)) \cdot \sum_{i=1}^{k} (1 - p_i(w)) \cdot \log(1 - p_i(w)) \tag{1.7}$$

  La formula sopra citata indica che maggiore è il valore di $I(w)$ maggiore è il potere discriminatorio di w.

- **Mutual Information:** è derivata dalla teoria dell'informazione. $M_i(w)$ è definito come il livello di co-occorrenza tra la classe i e la parola w:

$$Mi(w) = \log(\frac{p_i(w)}{P_i}) \tag{1.8}$$

  $M_i(w)$ è un valore specifico per ogni coppia $< i, w >$, si deve quindi definire un valore $M(w)$ che sia funzione del solo termine w. Questa funzione può essere scelta come media o come valore massimo tra gli $M_i(w)$:

$$M_{avg}(w) = \sum_{i=1}^{k} P_i \cdot M_i(w)$$

$$M_{max}(w) = \max_i \{M_i(w)\}$$

- X^2 **statistic:** si tratta di una via differente per carpire la dipenza tra una parola e una classe. Definito n come il numero totale di documenti nella collezione, X^2 statistic per tra parola w e una classe i è definito:

$$X_i^2(w) = \frac{n \cdot F(w)^2 \cdot (p_i(w) - P_i)^2}{F(w) \cdot (1 - F(w)) \cdot P_i \cdot (1 - P_i)} \tag{1.9}$$

  Il maggiore vantaggio di X^2 è che è un valore normalizzato, quindi maggiormente comparabile fra i termini di una categoria. Anche in questo caso si può calcolare il valore di X^2 in funzione del termine w tramite la media o il valore massimo.

- **tfdficf:** questa feature, proposta da Ceci e Malerba in [8], e riferise a un nuovo calcolo tramite il quale valutare l'importanza di un termine all'interno di una collezione di documenti. Per ogni categoria c_j, viene calcolata una lista di coppie $< v_i, w_i >$, in cui w_i è il termine i-esimo di un documento $d \in c_j$ e v_i è calcolato come:

$$v_{ij} = TF_{c_j}(w_i) \cdot DF_{c_j}^2(w_i) \cdot \frac{1}{CF_c(w_i)} \tag{1.10}$$

  In cui:

  - $TF_{c_j}(w) = \max_{d \in c_j} TF_d(w)$
  - $DF_{c_j}(w) = \frac{|\{d \in c_j| \text{ w occorre in d}\}|}{|c_j|}$
  - $CF_c(w)$ è il numero di sottocategorie $c_k \in DirectSubCategory(c)$ in cui occorre w in almeno un documento $d \in c_k$

- **LSI supervisionato:** si tratta di una feature transformation, ovvero si cerca di creare un nuovo e più piccolo insieme

di feature come funzione dell'insieme originale. Un tipico esempio è il *Latent Semantic Indexing* (LSI) [13] e la sua variante probabilistica PLSA [25]. Il metodo LSI trasforma lo spazio del testo (con molte centinaia di migliaia di parole) in un nuovo sistema di assi (poche centinaia) creati tramite combinazione lineare delle feature originali; per fare ciò, tramite tecniche di analisi dei compontenti [23], si individuano i sistemi di assi che contengono il maggior livello di informazione sulla variazione dei valori degli attributi.

- **Clustering supervisionato per la riduzione di dimensionalità:** è una tecnica che costruisce cluster dalle collezioni del testo sottostanti tramite l'uso di supervisione dalla classe distribuzione. Nel caso più semplice ogni classe può essere denotata come un singolo cluster, le parole più frequenti di questi cluster supervisionati possono essere utilizzate per creare il nuovo set di dimensioni. Un vantaggio di questo approccio è che viene conservata l'interpretabilità rispetto alle parole originali, mentre lo svantaggio è che non sempre la direzione ottimale di separazione può essere rappresentata dai cluster.

- **Analisi discriminante lineare:** questa tecnica prova esplicitamente a costruire direzioni nello spazio delle feature, lungo le quali si trova la separazione migliore fra le classi; un metodo comune è il *Fisher's linear discriminant* [22], che determina la direzione ottimale cercando di separare maggiormente i punti dello spazio, cercando iterativamente un vettore α nei dati tramite un'analisi discriminante, per semplicità consideriamo solo il caso di classe binaria:

$$\alpha = (\frac{C_1 + C - 2}{2})^{-1} \cdot (\mu_1 - \mu_2) \tag{1.11}$$

dove μ_1 e μ_2 sono le medie dei due dataset e C_1 e C_2 sono le matrici di covarianza. A ogni iterazione il vettore è sempre ortonormale al precedente.

Durante la feature selection si può decidere che tipo di vocabolario utilizzare, il metodo standard consiste nel considerare ogni singola parola (unigramma) e valutare in base all'occorrenza della parola stessa nei documenti il proprio apporto discriminitivo.

Un metodo diverso consiste invece nel valutare anche più parole occorrenti consecutivamente, portando ad avere vocabolari *n-gram*, dove n è il numero massimo di parole consecutive considerate. In questo modo si possono carpire semantiche discriminative impossibili da valutare con semplici unigrammi, ad esempio tramite un vocabolario 3-gram si può valutare l'occorrenza di "world wide web", parole che singolarmente non sarebbero molto discriminative, ma che insieme esprimono un concetto potenzialmente utile alla classificazione.

1.6.2 Decision Tree Classifiers

Un albero di decisione è essenzialmente una decomposizione gerarchica dello spazio dei dati (di training) dove un predicato, o condizione, sugli attributi è usato per dividere lo spazio gerarchicamente; nei problemi di text-mining generalmente questa condizione riguarda la presenza o meno di una o più parole nel documento.

I tipi di split con cui dividere lo spazio dei dati possono essere:

- **Single attribute splits:** in questo caso si usa la presenza o l'assenza di una parola in un particolare nodo dell'albero per effettuare lo split; a ogni livello, viene utilizzata la parola che discrimina maggiormente le classi, misurata ad esempio con il *Gini-index*.

- **Similarity-based multi-attribute split:** si usano le parole clusterizzate per la similarità tra documenti.

- **Discriminant-based multi-attribute split:** si sceglie un cluster di parole che discrimini maggiormente le diferenti classi.

Un'implementazione molto utilizzata in letteratura dei Decision Tree è il C4.5 [7] e C5 [19], che usa il single-attribute split. I Decision Tree sono spesso usati insieme a tecniche di *boosting*, una tecnica adattiva [20] che può essere usata per aumentare l'accuratezza della classificazione usando n classificatori, con l'*n-esimo* classificatore che viene costruito esaminando gli errori dell'*(n-1)-esimo*.

1.6.3 Rule-based Classifiers

I *Rule-based Classifiers* sono generalmente correlati ai *decision tree*. Lo spazio dei dati è modellato come un'insieme di regole, che partendo da condizioni sul feature-set indirizza una label; queste regole sono generalmente espresse come semplici congiunzioni di condizioni sulla presenza dei termini.

Le condizioni maggiormente usate nella generazione delle regole, dal training-set, sono:

- **Supporto:** quantifica il numero assoluto di istanze nel training-set rilevanti per la regola, in pratica quantifica il volume statistico che è associato alla regola. Data una associazione $A \rightarrow B$ è la proporzione di transazioni che contengono sia A e B.

- **Confidenza:** quantifica l'accuratezza dell'associazione. Consiste nella probabilità condizionata che le transazioni che contengono A, contengano anche B.

Nella fase di training vengono costruite tutte le regole, cercando per ogni istanza tutte le regole rilevanti. Un interessante classificatore rule-based per dati testuali [48] prevede l'utilizzo di una metodologia iterativa, tramite la quale viene determinata la singola regola migliore per ogni classe nel training-set in termini della confidenza della regola.

Un ulteriore tecnica è il *RIPPER* [33] che determina le combinazioni frequenti di parole che sono in relazione con una particolare classe.

1.6.4 Classificatori probabilistici e Naïve Bayes

I classificatori probabilistici sono progettati per creare implicitamente un modello *miscelato* per la generazione dei documenti, tipicamente assume che ogni classe è un componente della miscela.

Il modello di un classificatore probabilistico è un modello condizionale:

$$p(C|F_1, ..., F_n)$$

Dove C rappresenta le classi a cui un'istanza può appartenere e $F_1, ..., F_n$ rappresenta l'insieme delle feature.

Un esempio classico di questi classificatori sono le tecniche Naïve Bayes, che modellano la distribuzione dei documenti in ogni classe usando tecniche probabilistiche, basate sul teorema di Bayes:

$$p(C|F_1, ..., F_n) = \frac{p(C) \cdot p(F_1, ..., F_n|C)}{p(F_1, ..., F_n)} \tag{1.12}$$

Questi metodi assumono che la presenza o l'assenza di una particolare feature in un documento testuale non sia correlata alla presenza o assenza delle altre feature.

Normalmente per i classificatori Bayesiani sono utilizzati due tipi di modelli:

1. **Multivariate Bernoulli Model:** si usa la presenza/assenza delle parole nel testo come caratteristica per rappresentare il documento, non la frequenza. Nella costruzione del modello si usa generalmente l'assunzione di *Naïve independence* secondo cui si assume che la probabilità di occorrenza dei differenti termini sia indipendente fra essi. Senza questa assunzione i metodi Naïve non hanno molta efficacia.

2. **Multinomial Model:** si cattura la frequenza dei termini nel documento, rappresentandolo come una *bag-of-words*. Una gerarchia di categorie può essere usata per migliorare la stima dei parametri multinomiali nei classificatori Naïve Bayes per aumentarne significativamente l'accuratezza. L'idea chiave è di applicare tecniche di contrazione per levigare i parametri per le categorie dei figli con dati sparsi, con i propri nodi genitori. Il risultato è che i dati di training di categorie in relazione sono condivisi fra essi in modo ponderato, in modo da aumentare la robustezza e l'accuratezza della stima dei parametri quando i dati di training per i nodi figli sarebbero insufficienti.

Il processo di classificazione combina quindi uno di questi modelli con delle regole di selezione, ottenendo infine la seguente funzione per eseguire la classificazione:

$$classify(f_1, ..., f_n) = \arg\max_c p(C = c) \prod_{i=1}^{n} p(F_i = f_i | C = c) \tag{1.13}$$

1.6.5 Classificatori lineari

Un classificatore lineare raggiunge la classificazione dei dati tramite l'utilizzo di una combinazione lineare degli stessi. L'output di questi classificatori può essere definito come $p = \overline{A} \cdot \overline{X} + b$, dove

$\overline{X} = (x_1...x_n)$ è il vettore delle frequenze normalizzate delle parole, $\overline{A} = (a_1...a_n)$ è il vettore dei coefficienti lineari e b è uno scalare.

Un'interpretazione della funzione di predizione in scenari discreti può essere un iperpiano che separi le differenti classi. Un esempio di questo approccio sono le **SVM** (Support vector machine). Una caratteristica dei classificatori lineari è che sono strettamente relazionati a molti metodi di feature transformation, utilizzando i boundaries per trasformare lo spazio delle feature. Il modello a regressione è un metodo statistico classico per la classificazione del testo. Tuttavia è generalmente usato nei casi in cui le variabili target da imparare sono numerici piuttosto che categorici, esistono diverse varianti per l'applicazione alla text classification.

- **SVM:** L'idea di base consiste nel dividere nel miglior modo possibile le diverse classi, i dati testuali sono molto adatti ai classificatori SVM in quanto l'alta dimensionalità e sparsità dei dati sono spesso facilmente separabili linearmente. Un SVM costruisce un iperpiano, o insieme di iperpiani, in uno spazio ad alta dimensionalità, che può essere utilizzato per la classificazione, regressione, o altri compiti. Intuitivamente, una buona separazione si ottiene tramite l'iperpiano che ha la maggiore distanza dal punto più vicino dai dati di qualsiasi classe (margine funzionale), poichè, generalmente, maggiore è il margine e inferiore è l'errore del classificatore.

  Considerando un problema di classificazione binaria con dataset $(x_1, y_1), ..., (x_n, y_n)$, dove x_i è un vettore di input e $y_i \epsilon \{-1, +1\}$ sono le label binari corrispondenti. La risoluzione della classificazione utilizzando le SVM consiste, in termini formali, alla risoluzione del seguente problema quadratico

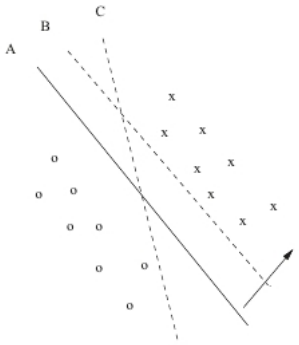

Figura 1.4: Ricerca del margine migliore per la separazione delle classi.

(QP):

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j K(x_i, x_j) \alpha_i \alpha_j \tag{1.14}$$

con: $0 \leq \alpha_i \leq C$, per $i = 1, 2, ..., n$ e $\sum_{i=1}^{n} y_i \alpha_i = 0$

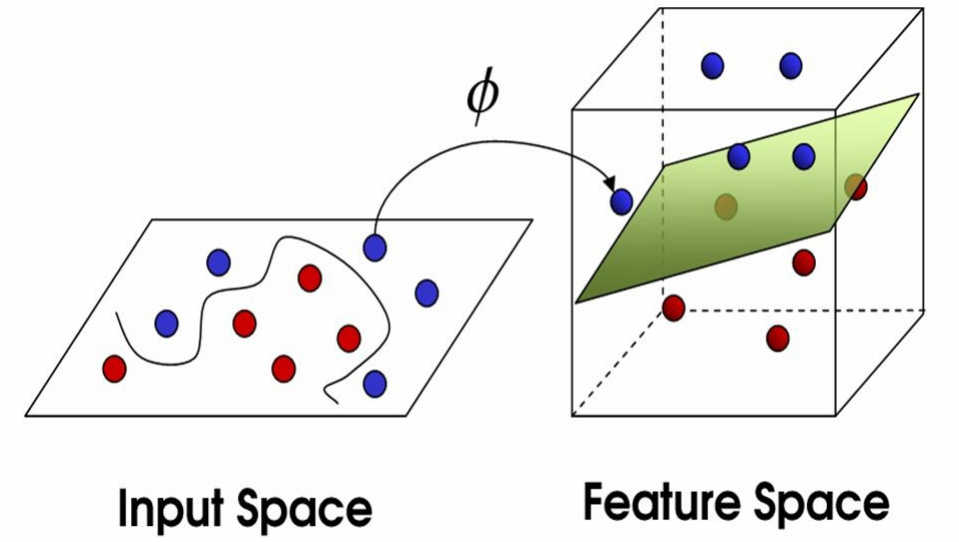

Figura 1.5: SVM, iperpiano ottimo costruito sullo spazio delle feature.

Un esempio di algoritmo SVM molto utilizzato in letteratura è stato proposto da Platt [41], chiamato *Sequential*

minimal optimization (SMO). L'idea di questo algoritmo è di risolvere i problemi di ottimizzazione presenti durante il processo di training degli SVM, cioè viene fatto dividendo i problemi in una serie di sottoproblemi più piccoli possibile, i quali vengono poi risolti in maniera analitica. I passi dell'algoritmo sono:

1. Seleziona una coppia di variabili α_1 e α_2.
2. Congela tutte le variabili eccetto α_1 e α_2.
3. Risolvi il problema QP considerando solo α_1 e α_2.
4. Ripeti fino alla convergenza.

Il numero di passi necessari per arrivare alla convergenza è fortemente dipendente dal metodo di selezione utilizzato nel passo 1.

- **Regression-Based Classifier:** Un esempio classico di classificatore regression-based è il *regression decision tree*, che analogamente ai *decision-tree* divide il data-set in sottoinsiemi più piccoli creando al tempo stesso un albero di decisione, in cui ad ogni split si cerca di minimizzare la varianza all'interno del nodo.

  Un metodo di regressione molto utilizzato è quello dei minimi quadrati: la funzione trovata deve essere quella che minimizza la somma dei quadrati delle distanze tra i dati osservati e quelli della curva che rappresenta la funzione stessa. Dati (x_i, y, i) con $i = 1, 2, .., n$ i punti rappresentanti i dati di training, si deve trovare la funzione $f(X)$ tale che approssimi la successione di punti data:

$$f(X) = \alpha + \beta X \tag{1.15}$$

  con:

$$\beta = \frac{\sum_{i=1}^{n}(x_i - \widehat{x})(y_i - \widehat{y})}{\sum_{i=1}^{n}(x_i - \widehat{x})^2}$$

$$\alpha = \widehat{y} - \beta\widehat{y}$$

Un altro esempio di funzione di regressione, creata appositamente per dati testuali, è proposta in [16] e chiamata *LLSF* (Linear Least Squares Fit).

- **Neural Network Classifiers:** Questi classificatori sono basati su una unità base detta neurone, che riceve un insieme di input X_i, frequenza dei termini nel documento *i-esimo*. Ogni neurone è associato a un insieme di pesi A, che viene utilizzato nella funzione di classificazione; un esempio di tipica funzione lineare è $p_i = A * X_i$. L'idea è quella di di partire con pesi random, o 0, e gradualmente aggiornarli ogni volta che si ha un errore, applicando la funzione corrente dell'esempio di training con una potenza di aggiornamento regolata da un parametro μ (learning-rate). La potenza di questa tecnica risiede nella possibilità di separare classi non separabili linearmente tramite l'utilizzo di strati multipli di neuroni; il prezzo da pagare è però la complessità del processo di training e che l'errore deve essere propagato a ritroso lungo gli strati.

  Alcune osservazioni e test [18] mostrano che i benefici di classificatori non lineari rispetto ai lineari, non pagano, in termini di efficienza ed efficacia, il prezzo computazionale fatto per tale implementazione.

1.6.6 Classificatori proximity-based (K-NN)

L'algoritmo *K-Nearest Neighbors* (k-NN) viene utilizzato nel riconoscimento di pattern per la classificazione di oggetti basandosi sulle caratteristiche degli oggetti vicini a quello considerato. Questo metodo ha complessità lineare rispetto al numero di vettori presenti nel problema, ipotizzando che il calcolo per la misura delle distanze sia poco oneroso.

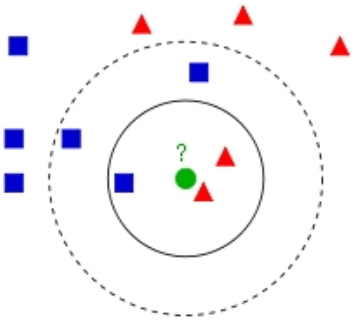

Figura 1.6: Classificazione con k-NN, con $k = 3$ e $k = 5$.

Dato un documento $\widehat{d}$, il metodo impiega una parte del training set $N_k(\widehat{d})$ per calcolare la classe $\widehat{c}$ associata ad $\widehat{d}$. $N_k(\widehat{d})$ rappresenta il neighbourhood di $\widehat{d}$, ossia l'insieme dei punti del training set più prossimi a $\widehat{d}$. Formalmente:

$$f(\widehat{d}) = argmax_{\widehat{c} \in C} \sum_{\widehat{d} \in N_k(\widehat{d})} \delta(\widehat{c}, c) \tag{1.16}$$

dove c è la classe associata a d e $\delta(\widehat{c}, c) = 1$ se $\widehat{c} = c$, 0 altrimenti.

Ai fini del calcolo della distanza gli oggetti sono rappresentati attraverso vettori di posizione in uno spazio multidimensionale. Normalmente viene impiegata la distanza *euclidea*, ma anche altri tipi di distanza sono ugualmente utilizzabili, ad esempio la distanza *Manhattan*. Nel caso di classificazione testuale possono essere utilizzate altre distanze quali ad esempio la distanza di *Hamming*.

Possono essere utilizzati 2 metodi per la classificazione di una data istanza di test:

- Un punto (che rappresenta un oggetto) è assegnato alla classe C se questa è la più frequente fra i k esempi di training più vicini.

- Organizzare durante la fase di learning il training-set in cluster e associare a ogni cluster un meta-documento rappresentativo, applicando poi l'approccio *k-nearest neighbors* su questi meta-documenti.

WHIRL [36] è un metodo per eseguire la classificazione nearest neighbor su dati testuali, essenzialmente crea unioni sulle basi della somiglianza tra attributi testuali.

1.6.7 Algoritmi Centroid-Based

Gli algoritmi di classificazione basati sul centroide utilizzano una rappresenrazione dei documenti mediante un *Vector Space Model*, in cui ognuno di essi consiste in un singolo vettore. Nella fase di learning viene creato un vettore prototipo (centroide) che rappresenti i documenti appartenenti a una categoria. Ogni centroide C_i associato alla categoria i-esima c_i è calcolato come valore medio di tutti i vettori rappresentati i documenti appartenenti a c_i.

$$C_i = \frac{1}{|S|} \sum_{d \in c_i} d$$

Una volta creati i vari centroidi, uno per ogni categoria, durante la classificazione di un'istanza test, rappresentata anch'essa come un vettore, viene assegnata la categoria avente centroide più simile al vettore rappresentativo. La similirarità è calcolata tramite la funzione coseno (1.1).

Una generalizzazione di questi algoritmi è data dal metodo di *Rocchio* [21], basato sulla formula omonima riguardante il meccanismo di *relevance feedback* nei sistemi *Vector Space Model*, creato nell'ambito dell'*information retrieval* circa negli anni '70. Il *relevance feedback* prevede di utilizzare le informazioni inizialmente restituite da una query indipendentemente dal fatto che siano rilevanti o meno per una nuova query.

Formalmente, il metodo di Rocchio crea un classificatore $\vec{c_i} = \langle w_{1i}, ..., w_{\tau i} \rangle$ per la categoria c_i, tramite la definizione dei pesi w_{kj} di t_k nel documento d_j secondo la formula:

$$w_{ki} = \beta \cdot \sum_{d_j \in POS_i} \frac{w_{kj}}{|POS_i|} - \gamma \sum_{d_j \in NEG_i} \frac{w_{kj}}{|NEG_i|} \tag{1.17}$$

Dove $POS_i = \{d_j \in Tr | \phi(d_j, c_i) = T\}$ e $NEG_i = \{d_j \in Tr | \phi(d_j, c_i) = F\}$.

I vantaggi di questo metodo sono l'efficienza, in quanto l'apprendimento è relativo a una media dei pesi, e che l'implementazione risulta semplice. Lo svantaggio è che però è in grado di dividere lo spazio dei documenti linearmente.

1.7 Hierarchical text classification

La maggior parte della ricerca sulla classificazione del testo è stata storicamente focalizzata sulla classificazione dei documenti in un insieme di categorie non relazionate strutturalmente fra esse (*flat classification*). In realtà in molte repository i documenti sono organizzati in gerarchie di categorie, per aiutare la navigazione degli argomenti d'interesse.

Nella classificazione tradizionale un documento è assegnato a una label tramite l'esecuzione di un classificatore, nell'approccio gerarchico, invece, l'output è dato da un insieme di classificatori che risiedono sui vari livelli dall'albero gerarchico, che distribuiscono i documenti in maniera top-down. In questa maniera si ha da una parte il vantaggio di scomporre la classificazione in sottoproblemi gestibili più efficaciemente, dall'altra l'aggiunta di problemi nello svolgimento della classificazione automatica, come ad esempio:

- Il fatto che i documenti possono essere associati sia a foglie dell'albero gerarchio sia a nodi interni.

- L'insieme delle feature selezionate dal classificatore possono essere specifiche per una categoria o le stesse utilizzate da più categorie.
- Il training-set associato a ogni categoria potrebbe includere documenti di training anche delle sottocategorie.
- Il classificatore potrebbe o no tener conto delle relazioni gerarchiche fra categorie.
- Si rende necessario un criterio di stop per la classificazione dei documenti non-foglia.
- I criteri di valutazione devono tener conto anche della gerarchia quando classificano l'errore.
- Gli errori di classificazione commessi in un nodo vengono propagati al figlio.
- La soglia (threshold) di decisione per una categoria ad alto livello è spesso non lineare.

Nel caso di una classificazione gerarchica si può decidere se associare un feature-set ad ogni categoria c, definendolo sulla base delle sue sottocategorie, più precisamente dall'unione dei dizionari delle dirette sottocategorie; oppure se definire un feature-set generale per tutta la gerarchia, definito anch'esso come unione dei dizionari di tutte le sottocategorie. Questa decisione è associata principalmente al tipo di approccio che si vuole avere per la classificazione:

- **Approccio big-bang:** lo stesso classificatore è utilizzato per ogni livello della gerarchia, la feature selection è unica è condivisa per ogni nodo dell'albero gerarchico.
- **Approccio top-down:** viene applicato un classificatore diverso ad ogni livello, in questo modo si ha un maggiore

sforzo computazionale in quanto si hanno diversi classificatori, col vantaggio però di poter applicare feature selection appropriate per ogni livello della gerarchia. L'approccio gerarchico procede nella classificazione di ogni istanza di documento in maniera top-down, partendo dal nodo radice fino alle foglie tramite un algoritmo greedy.

Quando il documento è in una categoria interna c della gerarchia, esso viene rappresentato in base al feature-set associato a c; il processo agisce itarativamente sui livelli della gerarchia, assegnando il documento alla categoria del livello sottostante con punteggio maggiore, valutando sempre che tale punteggio sia maggiore di un threshold definito, in caso contrario il documento viene assegnato alla categoria corrente. L'algoritmo itera finchè non esistono più sottocategorie con punteggio maggiore al treshold oppure se si è giunti a un nodo foglia.

Un caso particolare è rappresentato da una categoria c avente una sola sottocategoria diretta c', in questa situazione potrebbero sorgere dei problemi in caso di utilizzo di classificatori probabilistici, che definiscono la somma delle probabilità di appartenenza a tutte le sottocategorie uguale a 1; ogni documento sarebbe passato a c' senza un efficace controllo sul threshold. Per ovviare a questo problema si può decidere di inserire una categoria fittizia, corrispondente alla categoria padre c, che alteri la somma dei punteggi senza però alterare il processo di classificazione.

Viene dunque creato un classificatore per ogni categoria interna c nella gerarchia; tali classificatori sono utilizzati per decidere durante il processo di classificazione di un nuovo documento, quale sottocategoria c' è più appropriata per il documento, oppure, si deve anche essere in grado di stabilire se non è necessario passare a una sottocategoria. Ciò può avvenire perchè:

1. Il documento è di tipo generale e non di uno specifico argomento.

2. Il documento apparterrebbe a una specifica categoria non esistente nella gerarchia e quindi è più corretto classificarlo più generalmente.

Il problema della propagazione dell'errore da nodi genitori a figli può essere risolto [34] attraverso l'utilizzo di cross-validation per ottenere un training-set per il nodo figlio il più simile possibile al testo corrente. Mentre per quanto riguarda il piano di decisione non lineare, si può risolvere tale problema utilizzando classi di label sperimentali ottenute dai nodi figli come feature da utilizzare nei nodi padri.

La strategia di ricerca della classe a cui assegnare un documento può essere [15]:

- **Document based:** compara, tramite ad esempio la funzione coseno (1.1) la similarità tra il documento da classificare e uno nel training-set, si trovano gli N documenti più simili e viene assegnata la categoria dominante tra questi. I documenti in questo approccio sono rappresentati come bag-of-words.

- **Category based:** viene creata un categoria fittizia contente le feature del documento in esame e viene confrontata, sempre tramite una funzione di similarità, con le feature delle altre categorie, assegnando il documento alla categoria più simile.

1.7.1 Naïve Bayes gerarchici

I metodi Bayes sono molto indicati per la classificazione gerarchica; quando i dati di training sono strutturati in tassonomie di argomenti, questa struttura può essere sfruttata per raffinare

la classificazione in quanto è stato osservato [35] che una feature selection sensibile al contesto è in grado di fornire risultati di classificazione migliore.

La feature selection ha un apporto molto diverso nel caso gerarchico rispetto a quello flat, ad esempio in una gerarchia con due categorie top-level, *Sport* e *Computer*, e diverse sottocategorie per entrambi i rami, considerando tutte le sottocategorie in modalità flat, la parola "computer"non sarebbe molto discriminante, in quanto sarebbe comune a tutte le sottocategorie di *Computer*; mentre in un approccio gerarchico, la stessa parola, valutata al top-level, sarebbe molto discriminante.

Diventa quindi necessaria, oltre la scelta della feature selection in sè, l'introduzione di un metodo che modifichi le feature delle categorie a seconda del livello occupato nella gerarchia. Due metodi sono proposti per questo scopo:

- Un approccio derivato dalla teoria dell'informazione [42], che tiene conto delle dipendenze fra attributi, consiste in un algoritmo che elimina le feature una alla volta. L'approccio proposto può essere scomposto in tre fasi:

  1. Una fase di apprendimento non supervisionata, in cui viene utilizzato un *pLSA* per derivare i concetti specifici del dominio e creare su essi una rappresenzione semantica.
  2. Si costruiscono *weak hypoteses* basate sia sulle feature dei termini che dei concetti; le feature dei concetti sono derivate da diversi modelli pLSA aventi differenti granularità sui concetti.
  3. L'ultima fase è il combination stage, che utilizza *AdaBoost* per combinare efficientemente le weak hypoteses e integrare le informazioni term-based e concept-based. Nell'articolo vengono proposte due diverse varianti utilizzanti *AdaBoost.MH* e *AdaBoost.MR*, che si

differenziano principalmente per il modo con cui minimizzare l'upper bound, la prima tramite l'Hamming loss mentre la seconda con il ranking loss [43].

- Le specifiche feature dei nodi sono indicate come firme (signatures) e vengono utilizzate per il calcolo del rapporto tra la varianza intra-classe e inter-classe per ogni parola a ogni nodo dell'albero [35], tale calcolo può essere efficacemente effettuato tramite il *Fisher's discriminant*.

1.7.2 SVM gerarchici

L'approccio SVM può essere usato con successo nel contesto dell'organizzazione gerarchica delle classi. Come per la maggior parte degli approci gerarchici si può pensare [17] di applicare un classificatore a ogni livello della gerarchia, mantenendo un valore di soglia per decidere se proseguire nell'albero gerarchico o se ritenere conclusa la classificazione.

Il vantaggio dell'utilizzo degli SVM è che si prestano molto bene a feature organizzate come *bag-of-words*. Inoltre è semplice settare i parametri del SVM come ad esempio la penalità C imposta agli esempi di training classificati erroneamente, il cui valore classico è $C = 0,01$, e la soglia di decisione p, che è utilizzata per controllare la *precision* e la *recall*, impostata in [17] a $p = 0.2$.

1.8 Metodi di valutazione

In letteratura vengono utilizzati svariati metodi per la valutazione di un algoritmo di text-mining. I metodi più classici consistono nella *precision*, che può essere vista come una misura di esattezza o fedeltà, e la *recall*, che è una misura di completezza. Data la seguente tabella, per una classificazione binaria:

	YES atteso	NO atteso
classificato YES	tp	fp
classificato NO	fn	tn

Si possono definire le seguenti misure:

$$Precision = \frac{tp}{tp + fp} \tag{1.18}$$

$$Recall = \frac{tp}{tp + fn} \tag{1.19}$$

$$Accuracy = \frac{tp + tn}{tp + fp + fn + tn} \tag{1.20}$$

$$F_1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{1.21}$$

Un ulteriore metodo di valutazione per una classificazione tassonomica multilabel è stata introdotta da Cesa-Bianchi et al. in [39] e chiamata *H-loss*. Si tratta di una funzione che misura la bontà di una classificazione in base all'informazione persa nelle classificazioni errate lungo tutto il percorso gerarchico. Dato $\widehat{y} = (\widehat{y_1}, ..., \widehat{y_n})$ vettore delle predizioni multilabel per un'istanza e $v = v_1, ..., v_n$ le corrette classi di appartenenza, si definisce il valore di *H-loss* in questo modo:

$$l_H(\widehat{y}, v) = \sum_{i=1}^{n} \{\widehat{y_i} \neq v_1 \wedge \widehat{y_j} = v_j, j \in ANC(i)\} \tag{1.22}$$

1.9 Riepilogo

Un confronto tra vari algoritmi di classificazione testuale flat, ottenuto tramite quattro diversi Reuters data-set è stato fatto da Yang et al. in [38], con il risultato finale espresso dalla tabella 1.7:

System	Reuters version 1	Reuters version 2	Reuters version 3	Reuters version 4
WORD	-	.15 (Scut)	.31 (Pcut)	.29 (Pcut)
kNN	-	.69 (Scut)	.85 (Scut)	.82 (Scut)
LLSF	-	-	.85 (Scut)	.81 (Scut)
NNets.PARC (perceptron)	-	-	-	.82 (Pcut)
CLASSI (perceptron)	-	-	.80	-
RIPPER (DNF)	-	.72 (Scut)	.80 (Scut)	-
SWAP-1 (DNF)	-	-	.79	-
DTree IND	-	.67 (Pcut)	-	-
DTree C4.5	-	-	.79 (F_1)	-
CHARADE (DNF)	-	-	.78	-
EXPERTS (n-gram)	-	.75 (Scut)	.76 (Scut)	-
Rocchio	-	.66 (Scut)	.75 (Scut)	-
NaiveBayes	-	.65 (Pcut)	.71	-
CONSTRUE (Exp. Sys.)	.90	-	-	-

Tabella 1.1: Confronto di alcuni algoritmi noti in letteratura per la classificazione testuale flat.

Per quanto riguarda la classificazione strettamente gerarchica, la tabella 1.7 mostra un riepilogo degli algoritmi reperiti dai vari lavori proposti negli articoli studiati.

Work	Doc Rapresentation	Feature selection	Learning	Classification	Data-set	Results
Koller et al. [42]	Bag of words con frequenza	Cross-entropy	Naïve Bayes o KDB, un classificatore per ogni nodo interno	Ricerca greedy di un singolo percorso, assegnamento di una singola categoria	Reuters-22173 (3 diverse gerarchie considerate)	A=0,9
McCallum et al. [50]	Bag of words con frequenza	Mutual information, un feature set per tutta la gerarchia	Naïve Bayes + algoritmo di shrinkage	Ricerca greedy ed estensiva di un singolo percorso, assegnamento di una singola categoria	Yahoo "science" hierarchy (14.831 doc in 264 classi), Industry sector hierarchy (6.440 webpage in 71 classi), Newsgroup by Ken Lang (20.000 articoli)	A=0,76
D'Alessio et al. [51]	1-2-gram vocabolary, con frequenza nel doc	Variante dell'algoritmo ACTION, pos & neg featur.	Stima dei feature weight	Ricerca greedy ed estensiva con pruning, assegnamento di una o più categorie	Reuters-21578 (21578 doc, 70/30 training/test)	F1avg=0,8, Pavg=0,84, Ravg=0,77
Dumais et al. [17]	Bag of words con frequenza	mutual information, un feature-set per ogni categoria	SVM binario	Ricerca estensiva con pruning, assegnamento di una o più categorie	LookSmart's web directory (training: 50.078 doc, test: 10.024 doc; in 173 categorie su 2 livelli)	Top-level: F1avg=0,572, Sub-level: F1avg=0,495, senza errori nel top-level: F1avg=0,711
Ng et al. [52]	Bag of words con frequenza	Coefficiente di correlazione, X^2 e frequenza	Classificatore perceptron-based binario	Ricerca estensiva, assegnamento di una o più categorie	Reuters-22173 (939 doc in 9 classi su 2 livelli)	F1=0,72, P=0,73, R=0,71
Ruiz et al. [53]	Bag of words binaria	Coefficiente di correlazione, mutual information, odds ratio	Neural network	Ricerca estensiva, assegnamento di una o più categorie	Ohsumed (training: 183.229 doc, test: 50.216 doc; in 109 categorie su 5 livelli)	F1avg=0,57
Weigend et al. [54]	Bag of words binaria	LSI + X^2, feature-set globale o locale	Neural network	Ricerca estensiva, assegnamento di una o più categorie	Reuters-22173 (training: 9.610 doc, test: 3.662 doc)	PavgTerm=0,85 PavgLSI=0,74
Ceci & Malerba [8]	Bag of words con frequenza	Max{TF * DF^2 * ICF}	Naïve Bayes, centroid-based, SVM	Ricerca greedy di un singolo percorso, assegnamento di una singola categoria	DOMZ (5.612 doc in 221 categorie), RCV1 (150.765 doc), yahoo	Ayahoo=0,54 Arcv1=0,70 Admoz=0,45
Cesa-Bianchi et al. [55]	Vettore di parole creato utilizzando le BOW library, log(1 + TF) log(IDF) encoding	Parole occorrenti almeno 3 volte nel documento	Basato su pesi e aggiornamento di essi	Ricerca estensiva, assegnamento di una o più categorie	Reuters corpus volume 1 (100.000 newswire, in 101 classi su 3 livelli), Ohsumed (55.503 doc in 94 categorie)	RCV1 / Oshumed: Hloss-SH-LRS=0,74 / 1,2 Hloss-Hperc=1,22 /1,56 Hloss-Hsvm=0,71 / 1,17
Cesa-Bianchi et al. [39]	standard bag-of-words vectorization con tf-idf e poi normalizzata	-	Naïve Bayes + SVM	Ricerca greedy bottom up, si eliminano man mano i percorsi meno app ropriati	Reuters corpus volume 1 (100.000 newswire, in 101 classi su 3 livelli), Ohsumed (55.503 doc in 94 categorie)	RCV1 / Oshumed: Hloss-Bsvm=0,71 / 1,16 Hloss-Hsvm=0,71 / 1,17
Cai et al. [56]	Matrice della frequenza dei termini nei doc, vettore di concetti	pLSA	Sono costruite weak-hypotheses basate sia sulle feature dei termini che dei concetti	Ricerca greedy con AdaBoost (MH o MR)	Reuters-21578 (training: 9.603 doc, test: 3299 doc), Oshumed (54.708 doc, 90/10 training/test)	Reuters / Oshumed: PavgTerm=0,89 / 0,57 PavgMix=0,90 / 0,62 RavgTerm=0,77 / 0,33 RavgMix=0,82 / 0,35 F1avgTerm=0,83/0,42 F1avgMix=0,86 / 0,45
Xue et al. [15]	3-gram vocabolary, con frequenza nel doc	X^2	3-gram NB o SVM, Category o Document Based	Flat, top-down pruned, anchestor-assistant.	DMOZ (4.800.870 Web pages in 712.548 categorie su 12 livelli)	Profondità1/5/9nel tree: F1sb=0,82 / 0,3 / 0,25 F1Hsvm=0,86 / 0,35 / 0,21 F1deep=0,89 / 0,52 / 0,4

Figura 1.7: Riassunto di alcuni algoritmi gerarchici noti in letteratura.

Capitolo 2

Strumenti

2.1 Sorgenti dati

Tutti i test effetuati sul sistema sono stati effettuati su tre diverse collezioni di documenti. Due di esse, denominate *DMoz* e *Yahoo* (descritte nel dettaglio in seguito) sono costituite da siti web, ricavate dal lavoro di Ceci e Malerba [62], costituiscono una base di conoscenza ricavata appunto dal contenuto web di vari siti, organizzate in tassonomie di argomenti.
La terza sorgente di dati è invece stata ricavata dalla nota sezione del database medico americano: *Ohsumed*, consiste in una collezione di documenti ricavati da riviste mediche, organizzati secondo la tassonomia MeSH.

2.1.1 Yahoo

Il primo insieme di dati utilizzati nella sperimentazione del sistema è stata ottenuto tramite il lavoro di Ceci e Malerba e scaricabile da [62]. La collezione di dati è stata ricavata dai documenti di riferimento nella *Yahoo! Search Directory*. Sono stati estratti i 907 documenti Web effettivi cui si fa riferimento ai primi tre livelli della directory Web http://dir.yahoo.com/Science. Documenti vuoti e documenti contenenti solamente scripts sono stati rimossi.

Il data-set è formato da 901 documenti catalogati in 69 categorie (App. A.1) disposte in una tassonomia a 4 livelli (Tab. A.1).

2.1.2 DMoz

DMOZ [61] (abbreviazione di Directory Mozilla), anche noto come *Open Directory Project* (ODP), nato nel 1998, è il più grande catalogo di siti web esistente su Internet. Questo catalogo è organizzato e curato da volontari, chiamati editor, che non percepiscono alcun compenso per tali attività e che mantengono la correttezza della catalogazione dei siti web e quindi di tutta la struttura gerarchica. ODP utilizza un gerarchica ontologia per organizzare elenchi di siti web: pagine relative ad argomenti simili sono raggruppate in categorie, le quali possono quindi includere sia direttamente siti sia sotto-categorie più specifiche.

La parola Open nel nome del progetto, indica un approccio assai diverso rispetto ad altre directory esistenti di natura strettamente commerciale: gli elenchi di siti classificati da DMOZ vengono infatti resi disponibili gratuitamente a chiunque ne voglia fare uso. Questa scelta rende le informazioni raccolte da DMOZ aperte a tutti e ha favorito notevolmente la diffusione, e la raccolta, dei dati della directory sul web.

A differenza delle directory di tipo commerciale, ODP non richiede denaro per l'iscrizione di un sito all'interno del catalogo. I criteri seguiti per decidere se inserire o meno un sito in catalogo non sono influenzati da pressioni commerciali e si basano esclusivamente sui contenuti dei siti stessi. È proprio questo approccio e questa filosofia che ha costituito il presupposto per ottenere una directory il più possibile obiettiva, in grado di offrire informazioni utili e ben organizzate, tralasciando quelle non desiderate.

Dal punto di vista della catalogazione dei siti, i principali obiettivi di DMOZ sono due:

1. Completezza della directory: i siti web catalogati devono fornire uno stralcio rappresentativo di quanto si trova sul web. Ogni sito web contenuto nella directory deve offrire una consistente qualità e quantità di contenuti unici, ovvero non presenti in altri siti già catalogati.

2. Organizzazione della directory: ogni sito web catalogato va inserito esclusivamente nella categoria che rappresenta maggiormente i contenuti del sito stesso. Si cerca di evitare, rare eccezioni a parte, l'inserimento di un sito in più di una categoria.

I siti web inseriti in DMOZ spaziano su tutti i campi possibili del sapere e offrono informazioni di ogni genere. Inoltre la directory viene costantemente aggiornata e implementata così che gli studenti, i direttori di imprese, le impiegate, le casalinghe, i professori ed il semplice ricercatore vi possano trovare documentazioni veramente utili e che possono far risparmiare tempo rispetto ai tradizionali motori di ricerca.

Al 3 Maggio 2012 i siti, in lingua inglese, contenuti nella directory sono 5.028.612, catalogati in 1.011.042 categorie da 95.394 editors. Le categorie sono disposte in una gerarchia a 14 livelli, il cui livello radice (livello 0), è composto da un'unica categoria denominata *top*, da esso discendono 16 categorie di livello 1 (Fig. 2.1), che rappresentano gli *ancestor* per le discendenze nella tassonomia:

- *Top/Arts*
- *Top/Business*
- *Top/Computers*
- *Top/Games*
- *Top/Health*

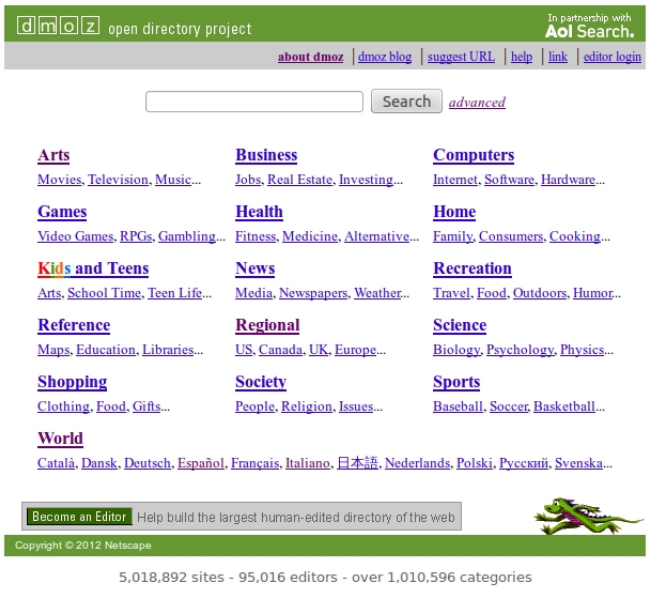

Figura 2.1: Home-Page di DMOZ, categorie di livello 1.

- *Top/Home*
- *Top/Kids and Teens*
- *Top/News*
- *Top/Recreation*
- *Top/Reference*
- *Top/Regional*
- *Top/Science*
- *Top/Shopping*
- *Top/Society*
- *Top/Sports*
- *Top/World*

DMOZ rende pubblico il download dei dati contenuti nella directory tramite file in formato RDF (Resource Description Format) che contengono le informazioni identificate da svariati tags, tra cui:

- topic: sigla rappresentativa della collocazione del tema trattato nell'albero delle categorie (es. Top/Arts/Art_History).
- catid: id univoco per identificare una categoria.
- title: sigla che rappresenta in sintesi il contenuto della categoria (es. Art_History).
- description: breve descrizione degli argomenti trattati all'interno della categoria (es. The Art History category covers all types of art: painting, sculpture, architecture, etc. All time periods and movements are obviously covered as well).
- fatherid: id della categoria padre, ovvero del livello direttamente superiore da cui discende.
- livello: indica il livello della categoria nella struttura ad albero.

Anche in questo caso, il data-set è stato ottenuto dal lavoro di Ceci e Malerba [62], che comprende solo una porzione dei documenti contenuti in tutta la *ODP*.
Si sono estratti i documenti Web effettivi cui si fa riferimento nei primi cinque livelli della directory Web, radicati nel ramo _Top/Health/ConditionsandDiseases/. Documenti vuoti e documenti contenenti solo script sono stati rimossi.
Il data-set è formato da 222 categorie (App. A.2) disposte in una tassonomia a 6 livelli, contenente 5836 documenti (Tab. A.2).

2.1.3 Ohsumed

La raccolta di documenti *Ohsumed* è stato creata da Hersh e colleghi del Oregon Health Sciences University [37]. Si tratta di un sottoinsieme del database MEDLINE ed è composto da 348.566 documenti ricavati da 270 riviste mediche nel periodo dal 1987 al 1991. Solo 233.445 documenti contengono titolo e abstract (i restanti documenti contengono solo un titolo). I documenti sono stati catalogati manualmente utilizzando la tassonomia MeSH (Medical Subject Headings), che consiste in 18.000 categorie, di cui 14.321 utilizzate per catalogare i documenti di questa sorgente di dati.
L'utilizzo di tutti i documenti porterebbe ad un insieme di dati computazionalmente intrattabile, la scelta utilizzata da tutta la letteratura consiste nella considerazione dei soli articoli facenti parte della categoria "Health Diseases"e del relativo sottoalbero nella tassonomia MeSH. Questa sottogerarchia dispone di 119 categorie (App. A.3), di cui solo 102 contententi dati utilizzabili, contenenti 23.669 documenti. La costruzione iniziale di questa raccolta prevede una classificazione multi-label per ogni documento, per poter utilizzare il dataset all'interno del sistema creato sono stati eliminati tutti i documenti classificati con più di una categoria. Inoltre, per motivi di efficienza e tempistica, sono stati considerati solamente 1000 documenti, scelti in maniera casuale, all'interno della collezione di dati (Tab. A.3).

2.2 WordNet

WordNet è un database semantico-lessicale per la lingua inglese [63], un ontologia linguistica che rappresenta in modo esplicito la conoscenza linguistica umana, in grado di organizzare, definire e descrivere i concetti semantici espressi dai vocaboli. Il progetto è nato presso l'università Princeton (1985) dal gruppo di

linguistica e psicolinguistica, in particolare dal linguista George Armitage Miller.

L'innovazione introdotta da WordNet sta nell'idea di base di memorizzare le informazioni su base semantica, categoria sintattica di appartenenza e sulle relazioni che si formano tra le varie parole. Il lessico è disponibile gratuitamente on-line e il medesimo progetto è stato ampliato anche alle varie lingue europee nell'ambito del progetto *EuroWordNet*, finanziato dalla Comunità europea.

Nella comunità di Text Processing WordNet viente utilizzato per aggiungere semantica. Semantizzare un testo significa collegarlo in modo appropriato con il resto della base di conoscenza posseduta: la lingua è un reticolo di collegamenti.

WordNet suddivide il lessico in quattro categorie lessicali: sostantivi (o nomi), verbi, aggettivi e avverbi. Tale suddivisione deriva da analisi psicolinguistiche secondo cui queste categorie vengono trattate indipendente l'una dall'altra nella mente umana.

Il concetto di parola viene definito come associazione tra due elementi:

1. Word-form (forma-parola): stringa di caratteri (lettere) che definiscono l'espressione fisica di una parola

2. Word-meaning (significato-parola): concetto lessicale espresso dalla parola, ogni parola veicola, anche in modo sottinteso, un senso.

Sostantivi, aggettivi, verbi e avverbi sono raggruppati in *synset* (insiemi di sinonimi), ognuno dei quali esprime un concetto distinto, ovvero word-form diverse che sono associate alla stessa word-meaning. Ogni *synset* è in pratica un insieme di word-form e rappresenta un word-meaning. Le word-form mappate su più word-meaning sono definite *polysemous*. Le due proprietà

fondamentali delle parole nell'organizzazione di WordNet sono quindi:

1. *Sinonimia*: proprietà relativa a Word-form (parole) che denotano lo stesso concetto e sostituendo l'una con l'altra non si cambia il valore di verità di una frase. Come detto in precedenza i termini sono raggruppati in insiemi non ordinati (synset).

2. *Polisemia*: proprietà di una parola di avere più significati, ovvero appartenere a più word-meaning.

L'ultima versione del progetto è stata rilasciata nel Dicembre del 2006 e contiene 155.287 parole (lemmi) distinte organizzate in 117.659 insiemi di *synset* e divise secondo le quattro categorie lessicali (Fig. 2.2).

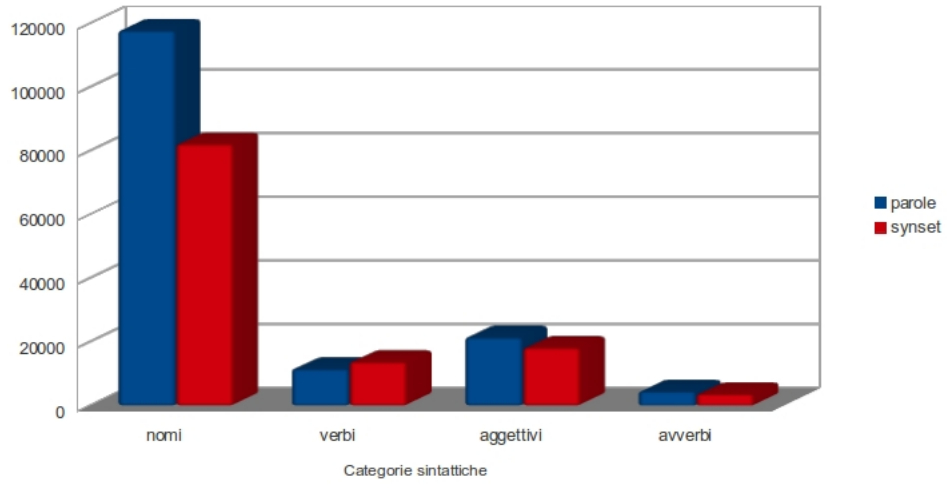

Figura 2.2: Distribuzione di parole e synset per categoria lessicale.

2.2.1 Relazioni

Oltre alla sinonimia, che può essere considerata una vera e propria proprietà, WordNet prevede 28 tipi di relazioni, che possono essere:

- Relazioni lessicali: si instaurano tra word-forms, sia tra forme contenute nello stesso synset sia esterne.
- Relazioni semantiche: si instaurano tra word-meaning.

Vediamo ora nel dettaglio le varie relazioni previste in nel database:

1. Iponimia (*hyponym* [∼]): è la relazione "is a", ovvero di sottoinsieme, collega synset più specifici a quelli più generali. Un synset A è *hyponym* di un synset B se A è del tipo di B (es. abete ∼ albero). Questa relazione è transitiva, infatti affermando che un abete ∼ albero e albero ∼ pianta, allora abete ∼ pianta.
2. Istanza di iponimia (*instance hyponym* [∼i]): indica una singola istanza di iponimia, è utilizzata per distinguere i tipi dalle istanze specifiche, quest'ultime sono sempre i nodi di foglia delle gerarchie. (es. Mario Rossi è un'istanza di persona).
3. Iperonimia (*hypernym* [@]): relazione inversa all'iponimia, relaziona i *synset* più generali a quelli più specifici. (es. felino @ gatto)
4. Istanza di iperonimia (*instance hypernym* [@i]): duale dell'*instance hyponym* per l'iperonimia.
5. Meronimia (*meronym* [%]): è la relazione "part of", componente di, è la relazione tra la parte e l'intero. Un *synset* A è *meronym* di un synset B se A è un componente di B.

(a) *Part meronym* [%p]: indica la relazione tra un concetto (componente) che è parte di un altro concetto (oggetto), ma che può avere una propria funzione anche singolarmente (es. ruota/macchina).

(b) *Member meronym* [%m]: indica l'appartenenza di un concetto individuale a un concetto collettivo (es. ossigeno/aria).

(c) *Substance meronym* [%s]: relaziona un oggetto con il materiale con cui è composto. A differenza della *part meronym*, le due parti sono integranti l'una dell'altra, non sono separabili. (es. plastica/bottiglia).

6. Olonimia (*holonymy* [#]): relazione inversa di *meronym*. Un synset A è *holonym* di un synset B se B "è componente di"A. (es. computer # CPU).

   (a) *Part holonymy* [#p]: duale del *part meronym*.

   (b) *Member holonymy* [#m]: duale del *member meronym*.

   (c) *Substance holonymy* [#s]: duale del *substance meronym*.

7. Implicazione (*entailment* [*]): relazione riferita ai verbi. Si dice che il verbo A è un'implicazione del verbo B se per compiere B è necessario compiere A (es. partecipare * vincere). Non necessariamente è valido il contrario.

8. Causa (*cause to* [>]): è di tipo implicazione, ne fanno parte ad esempio i verbi dare e avere. Lega verbi causativi con il rispettivo verbo risultativo (es. bere > dissetarsi).

9. Antonimia (*antonym* [!]): organizza gli aggettivi. Coppie di aggettivi si dicono in antonimia diretta quando hanno significati semanticamente inversi (es. bagnato/asciutto o giovane/vecchio). Al contrario, gli aggettivi semanticamente simili (es. secco/arido) si dicono in antonimia indiretta.

10. Similarità (*similar* [&]): definisce due aggettivi che sono in relazione secondo l'espressione A è simile a B.

11. Vedi anche (*see also* [^]): tipo lessicale e unisce parole appartenenti a *synset* diversi.

12. Attributo (*attribute* [=]): relaziona i nomi a cui vengono applicati aggettivi, un nome per il quale aggettivi esprimono valori. (es. peso è un attributo, per cui gli aggettivi leggeri e pesanti indicano dei valori espressi.

13. Gruppo di verbi (*verbgroup* [$]): raggruppa i verbi che riferiscono lo stesso significato.

14. Participio (*participle* [<]): relaziona gli aggettivi ai verbi da cui sono derivati.

15. Riferirisce (*pertainym*): relazione applicata ad aggettivi relazionali. Aggettivi di questo tipo sono solitamente definiti da frasi come di o pertinente a e non hanno antonimi. Possono essere riferiti a un nome o ad un altro aggettivo relazionale.

16. Derivazione (*derivation* [\]): collega parole semanticamente appartenenti a categorie sintattiche diverse, ma che hanno la stessa forma di radice.

17. Dominio (*domain* [;]): relazione relativa a *synset* definiti tramite legami di:

    (a) *Domain category* [;c].

    (b) *Domain region* [;r].

    (c) *Domain usage* [;u].

18. Membro (*member* [−]): relazione relativa a *synset* definiti tramite legami di:

(a) *Domain member category* [−c].

(b) *Domain member region* [−r].

(c) *Domain member usage* [−u].

L'insieme di queste relazioni rappresenta il cardine del sistema di WordNet. Il numero delle relazioni sopracitate è 30, ma la meronimia e l'olonimia vengono prese in considerazione solo nelle proprie sotto-sezioni; *member* e *domain* invece sono valutate anche singolarmente, oltre alle proprie sotto-classi. Si ha così un totale di 28 relazioni più la proprietà di sinonimia.

2.3 Weka

Weka (Waikato Environment for Knowledge Analysis) [60] è un software open source rilasciato con licenza GNU (General Public License) sviluppato presso l'università di Waikato in Nuova Zelanda. Essendo sviluppato completamente in Java questo software è utilizzabile su qualsiasi sistema operativo dotato di una Java Virtual Machine. Weka consiste in una collezione di algoritmi di machine learning (ovvero apprendimento automatico) nell'ambito di data mining, in particolare fornisce tramite interfaccia grafica vari tools per l'analisi dei dati e la creazione di modelli predittivi.

Questo sistema fornisce diversi strumenti di data mining come pre-processamento, clustering, classificazione, regressione, visualizzazione e selezione delle features. Tutte queste tecniche sono applicabili a dati in formato flat, ovvero ogni dato del dataset è descritto da un numero fisso di attributi, numerici o categorici.

Le funzionalità rese disponibili da Weka sono utilizzabili sia tramite interfaccia grafica, sia richiamabili in progetti esterni java tramite l'utilizzo del file .jar

Weka consente inoltre, grazie al JDBC (Java DataBase Connectivity), l'interfacciamento a database SQL, permettendo il processamento del risultato di una query su un database.

2.3.1 Ambienti operativi

Come si può notare dall'interfaccia grafica di base di Weka (Fig. 2.3), sono disponibili quattro diversi ambienti operativi grafici:

1. Explorer: permette l'analisi dei dati e l'applicazione di tecniche di Data Mining.
2. Experimenter: versione batch di esplorer, offre la possibilità di eseguire esperimenti e test per l'analisi statistica.
3. Knowledge Flow: offre la possibilità di automatizzare i processi di mining, definendo un determinato workflow per l'esecuzione di alcune funzionalità (es. caricamento di file, applicazione di filtri, etc).
4. Simple CLI: utilizzo di Weka da linea di comando.

Figure 1: Graphical User Interface Chooser

Figura 2.3: Interfaccia grafica base di weka.

Capitolo 3

Framework Gerarchico

In questo capitolo verranno descritti i metodi e le scelte fatte per la creazione dell'algoritmo per la classificazione gerarchica dei documenti, in particolare sarà ampiamente descritta l'idea alla base di questa tesi, ovvero l'impostazione di tutto il modello di classificazione sulle relazioni semantiche tra parole derivate da WordNet.

Come decritto nel capitolo 1.3.4 e 1.6 la classificazione testuale è il processo che approssima la funzione target Φ attraverso la costruzione induttiva di un classificatore di un dato dataset. Fatto ciò, si assegnano documenti non ancora visti utilizzando la funzione approssimata $\widetilde{\Phi}$. La prima fase è chiamata *apprendimento*, la seconda *classificazione*.

Come di consueto nei processi di classificazione, la fase di apprendimento e classificazione può essere suddivisa in due fasi:

1. Pre-processing: viene creato un mapping del contenuto di ogni documento in una *logical view*, ovvero una rappresentazione degli stessi, che poi può essere utilizzata nell'algoritmo di classificazione. Varie operazioni testuali e statistiche sono utilizzate per estrarre il contenuto più importante di ogni documento.

2. Apprendimento/Classificazione: basato sulla rappresentazione dei documenti, rappresenta il vero algoritmo di ap-

prendimento tramite un insieme di training di documenti e la successiva classificazione di documenti test.

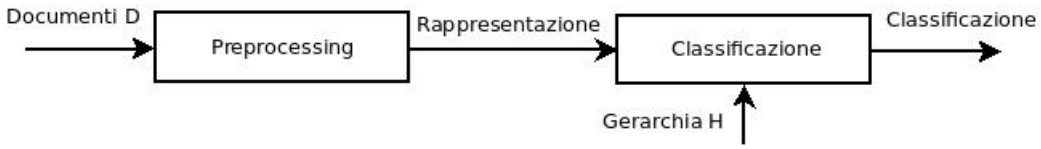

Figura 3.1: Le due fasi della classificazione di documenti.

Il processo di classificazione implementato nello svolgimento di questa tesi è stato reso parametrico in ogni suo particolare, in modo da rendere confrontabili le varie tecniche o scelte implementative così da poter valutare il più oggettivamente possibile ogni aspetto studiato.

3.1 Pre-processing

Come detto in precedenza il preprocessing è il processo che mappa un contenuto testuale di un documento in una logical view, o rappresentazione, che possa essere utilizzato nella fase di classificazione. In particolare in questa fase vengono applicate varie tecniche di manipolazione testuale per applicare la *term extraction*, ovvero l'estrazione di tutti i termini potenzialmente utili ai fini della rappresentazione semantica del documento. Verranno descritti ora i vari filtri testuali utilizzati nel sistema per questo scopo:

1. Rimozione dei tag HTML: lavorando in questa tesi su sorgenti dati ricavati da siti web (es. DMoz e Yahoo!) la prima operazione da effettuare è l'eliminazione di tutti i tag HTML che evidentemente non portano alcuna semantica al contenuto del documento. Questo filtro è stato ottenuto tramite la classe Java `javax.swing.text.html.parser.ParserDelegator`.

2. Rimozione punteggiatura e numeri: in questa fase viene eliminato ogni tipo di segno di punteggiatura e cifre numeriche contenute nei documenti. In particolare viene utilizzato il codice ASCII di ogni termine, eliminando tutti quelli non compresi fra [a-z,A-Z].

3. Case-folding: vista la disambiguazione dei caratteri scritti in minuscolo o maiuscolo: la stessa parola scritta "dog"o "DOG"non farebbe matching in quanto vengono utilizzati diversi caratteri ASCII, ogni carattere di ogni termine viene convertito in minuscolo.

4. Conversione Plurale→Singolare: Per aumentare il matching tra le varie parole e aumentare anche le relazioni ottenute in WordNet (dato che contiene solo termini in forma singolare), si è creato un metodo basato sulle regole grammaticali inglesi per convertire i termini in forma plurale in quella singolare.

   La creazione di questo metodo non si è limitata al caso base del plurale inglese che aggiunge una "s"al sostantivo in forma singolare. Il metodo proposto cerca sempre inizialmente, se presente, di eliminare la "s"finale di un termine. Fatto ciò si cerca la parola risultante su WordNet, in caso la ricerca abbia esito positivo il nuovo termine viene aggiunto alla rappresentazione del documento, in caso contrario si procede con una delle regole create osservando i vari casi delle forme regolari e irregolari si sono create le seguenti regole, schematizzate in tabella 3.1:

5. Rimozione stop words: come descritto nel capitolo 1, la rimozione delle parole più comuni nel linguaggio naturale è un importante mezzo per la riduzione della dimensionalità, in quanto spesso sono presenti in grande quantità nel testo ma non danno alcun significato semantico. Nel sistema si è

Regola	**Singolare**	**Plurale**
Nomi che termininano in *"s"*, *"ss"*, *"ch"*, *"sh"*, *"x"* aggiungono *"es"*	Touch, Kiss	Touches, Kisses
Nomi che termininano in *"y"*, se la *"y"* non è preceduta da vocale, si trasformano in *"ies"*	Lady, Boy	Ladies, Boys
Nomi che termininano in *"o"* preceduta da vocale, o parole di origine straniera, o parole abbreviate, aggiungono la *"s"*	Video, Canto, Piano	Videos, Cantos, Pianos
Alcuni nomi che termininano in *"f"* o *"fe"*si trasformano in *"ves"*	Life, Knife, Roof	Lives, Knives, Roofs

Tabella 3.1: Regole grammaticali

utilizzata una lista di 671 parole comuni nel vocabolario inglese. Inoltre, ogni termine consistente di 3 o meno caratteri viene eliminato dalla rappresentazione del documento.

Un ulteriore filtro classico di preprocessamento è lo *Stemming*. Si tratta di un processo utilizzato per ridurre parole flesse al loro tema, il tema non deve necessariamente coincidere con la radice morfologica della parola: l'importante è che parole con una semantica strettamente correlata vengano mappate sullo stesso tema. Le tecniche di stemming sono note in informatica dagli anni '60, Porter nel 1980 creò un raffinato metodo che si impose come metodo standard per lo stemming in inglese.
Nonostante l'utilità dello stemming, in questa tesi non viene utilizzata questa tecnica in quanto la semantica dei termini viene attribuita tramite le relazioni WordNet, inoltre data la costituzione di WordNet da parole nella forma base e non stammizzate impone il non utilizzo di questo metodo.

3.2 Dimensionality reduction

Nella categorizzazione testuale, come già ampiamente esposto nei capitoli precedenti, uno dei maggiori problemi è l'alta dimensionalità delle feature tramite le quali viene creata la classificazione, ovvero tutte le differenti parole occorrenti nella collezione di documenti. Una riduzione di questa dimensionalità è necesarria per varie ragioni, la prima e più ovvia delle quali è per le performance del classificatore, in quanto un così elavato numero di feature all'interno del modello renderebbe il processo impraticabile in termini di complessità sia temporale che spaziale. Inoltre una buona riduzione della dimensionalità risulta vantaggiosa anche in termini di riduzione dell'*overfitting* [40], ovvero il fenomeno per il quale il classificatore viene sintonizzato più sui documenti specifici di training che sulle caratteristiche semantiche reali delle categorie.

La riduzione della dimensionalità comporta, dopo la fase di preprocessing e term extraction una fase definita di weighting dei termini, in cui si associa tramite una formula di feature selction un peso a ciascun termine, e infine vengono selzionati k termini significativi per ogni documento e categoria in modo da crearne la rappresentazione definitiva che sarà utilizzata nel modello di classificazione (Fig. 3.2).

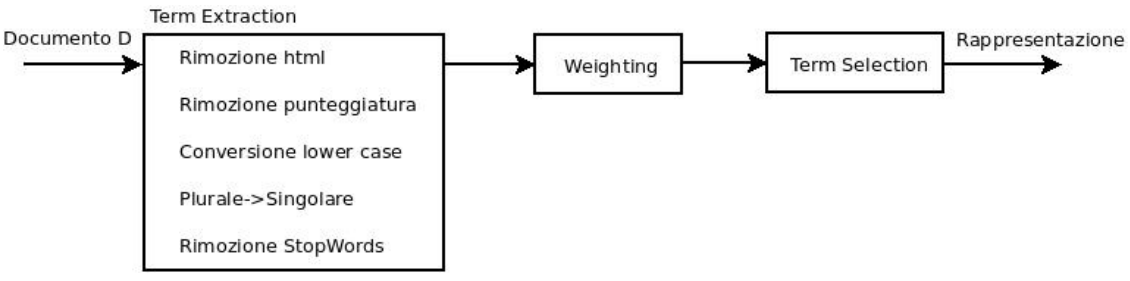

Figura 3.2: Fasi principali della dimensionality reduction.

I metodi specifici proposti in questa tesi per la riduzione della

dimensionalità e la creazione quindi della rappresentazione di documenti e categorie verrà descritta nei paragrafi seguenti.

3.2.1 Feature selection

Per la feature selection ci si è basati sia sulle tecniche proposte dalla letteratura, sia su una variante del tfidf proposta in questa tesi. Il calcolo del peso di ogni termine è stato valutato secondo varie feature selction, in modo da poter valutare l'efficacia dell'algoritmo con vari parametri.

Il calcolo più importante, e che si è rivelato più performante (Cap 5), è una variante del *tfidf* [2], il cui calcolo standard (denominato d'ora in avanti di *tipo 0*) è già stato descritto in 1.6.1. In questa tesi viene proposto il calcolo del tfidf in due varianti, chiamate *tipo 1* e *tipo 2*.

Il $tfidf^{type1}$ rappresenta il calcolo del tfidf in cui però la parte *idf*, che consiste nel logaritmo del rapporto tra il numero di documenti totali e quelli in cui occorre la parola, viene modificata in modo da ignorare tra i documenti conteggiati tutti quelli appartenenti alla stessa categoria di quello in esame; in questa maniera l'intento è di diminuire la rilevanza solo per le parole che compaiono in documenti di categorie diverse. La formula ottenuta è la seguente:

$$tc_{ij}^{type1} = tf_{ij} \cdot \log\left(\frac{\text{Numero Doc. in D - Numero Doc. in } C_j + 1}{\text{Numero Doc. in } \{D \setminus C_j\} \text{ contenenti } t_i + 1}\right)$$

Si può notare che nella formula è stato aggiunto +1 sia al numeratore che al denominatore, ciò è dovuto al caso in cui il termine non sia presente in alcun altro documento appartenente a cate-gorie differenti, per evitare quindi la divisione per 0 è stata applicata questa modifica.

La seconda variante proposta, $tfidf^{type2}$, prosegue sull'idea generale proposta nella prima, escludendo però dal calcolo dell'*idf*, oltre ai documenti appartenenti alla categoria del documen-

to in esame, tutti quelli appartenenti a categorie contenute nel sottoalbero di quella in esame. Questa variante è stata pensata appositamente per l'algoritmo gerarchico, in quanto i termini contenuti con maggiore frequenza nel sottoalbero di C_j e non nel resto delle categorie hanno un maggiore peso nella ricerca top-down quando ci si trova al livello l-esimo cui appartiene C_j. La formula ottenuta è:

$$tc_{ij}^{type2} = tf_{ij} \cdot \log\left(\frac{\text{Num Doc. in D - Num Doc. in } SubTree(C_j) + 1}{\text{Num Doc. in } \{D \setminus SubTree(C_j)\} \text{ contenenti } t_i + 1}\right)$$

Dove $SubTree(C_j)$ indica ogni categoria appartenente al sottoalbero avente C_j come radice. Entrambe le misure sono poi state normalizzate tramite la stessa formula (1.5) utilizzata per i tfidf standard.

Oltre a questi tre *tfidf* per ogni termine è stata calcolata come misura la sola frequenza relativa di occorrenza nel documento e la feature $tfdficf$ proposta da Ceci e Malerba [8] descritta in 1.6.1.

3.2.2 Rappresentazione dei documenti

L'ultimo step della dimensionality reduction consiste nella generazione vera e propria della rappresentazione dei documenti (e categorie). Ovvero, una volta assegnato a ogni termine un peso, calcolato tramite una delle formule precedentemente descritte, vengono selezionati per ogni documento i k_d termini che hanno peso maggiore.
Va notato che i documenti appartenenti al training-set possono essere ordinati per tutti i tipi di *tfidf*, mentre per quelli appartenenti al test-set non possono essere utilizzati i *tfidf* introdotti nella tesi (*tipo1* e *tipo2*) in quanto richiederebbero la conoscenza della categoria di appartenenza, vincolo inammissibile per definizione dei documenti test.

Come per la scelta della feature selection con cui vengono ordinati, anche il numero di termini k_d rappresentativi di un documento è una variabile modificabile all'interno dell'algoritmo.

3.2.3 Rappresentazione delle categorie

Per quanto riguarda la rappresentazione delle categorie, nello sviluppo del sistema, si è scelto di utilizzare due diverse rappresentazioni delle categorie tramite i termini più discriminitavi.

La prima rappresentazione, denominata *singola*, modella la categoria utilizzando i k_c termini più discriminativi appartenenti ai documenti contenuti nella categoria stessa; per ottenere la corretta discriminazione dei vari termini contenuti nei documenti, viene utilizzata la media aritmetica fra tutti i valori della feature selction scelta di ogni parola nei vari documenti. Questa rappresentazione è utilizzata in particolare durante la categorizzazione gerarchica per definire la similarità di un documento con la singola categoria.

Il secondo modello invece rappresenta la categoria in maniera *bottom-up*, ovvero utilizza i termini significativi appartenenti alla categoria rappresentata, più tutte le categorie appartenenti al sottoalbero della stessa. Per non rendere l'aumento esponenziale della dimensione della rappresentazione bottom-up delle categorie ai livelli più alti, si è creato un parametro che controllasse il numero di livelli del sottoalbero da cui prelevare i termini contenuti. La selezione dei termini per questa modalità è stata fatta selezionando k_c termini per ogni categoria utilizzata nel processo di risalita delle feature, per cui il massimo numero totale di termini rappresentativi per la categoria C_j sono $k_{C_j} \leq k_c \cdot |SubTree(C_j)|$. Si può notare come questo valore sia maggiore per categorie poste su livelli alti dell'albero gerarchico, abbassandosi man mano che la categoria in esame sia posta su livelli inferiori della gerarchia. Questa particolare selezione

dei termini rappresentativi di una categoria è stata pensata per aumentare l'efficacia del processo gerarchico nella selezione del nodo figlio in cui continuare la discesa della classificazione, in questa maniera infatti il potere discriminativo di ogni categoria non si basa solamente sui termini contenuta in essa, ma su tutti quelli contenuti nel sottoalbero, aiutando così la classificazione di documenti test appartenenti a livelli inferiori della gerarchia.

3.3 Categorizzazione tramite relazioni semantiche

L'innovazione dell'algoritmo proposto in questa tesi, al contrario di ogni altra tecnica proposta in letteratura, che per la discriminazione di un documento utilizza l'occorrenza di un determinato set di feature nel corpo dello stesso, consiste nella creazione di un classificatore basato sulle relazioni semantiche messe a disposizione da WordNet.

L'aspetto rivoluzionario di un classificatore creato in questa maniera è la possibilità di classificare istanze in categorie non utilizzate durante la fase di training del classificatore. Creando la funzione approssimata $\widetilde{\Phi}$ in base alle sole relazioni semantiche dei termini contenuti in documenti simili, dissimili o iponimi di una categoria, il classificatore così creato può essere utilizzato per categorizzare categorie e feature non utilizzate durante la fase di training; è inoltre possibile utilizzare un modello, istruito tramite un certo dataset, su un dataset completamente diverso da quello di training. Questa possibilità rappresenta una caratteristica assolutamente innovativa in questo settore e di estrema importanza, in quanto si crea un meta-classificatore, poichè questo è indipendente dal dizionario (insieme delle feature) dei documenti e delle categorie su cui è stato addestrato.

L'idea di base sta nel creare un classificatore che come attributi abbia funzioni delle relazioni semantiche interne a WordNet, tale classificatore sarà addestrato con un training-set composto da istanze generate da coppie documento-categoria $< D_i, C_j >$, ogni coppia sarà classificata tramite un attributo nominale che può essere *SIMILAR*, *HYPONYM* o *DISSIMILAR*, rispettivamente a seconda che D_i apparttenga a C_j ($i \equiv j$), che C_j sia una categoria iponima (padre, padre del padre, ecc) della categoria C_i cui appartiene D_i, oppure che non ci sia alcuna relazione tra la categoria C_i di appartenenza di D_i e C_j.

Per ogni coppia $< D_i, C_j >$ si è quindi creata l'istanza del set di dati cercando le relazioni semantiche tra D_i e C_j, questo procedimento prevede la generazione, tramite prodotto cartesiano, di tutte le coppie di termini rappresentativi di D_i e C_j, creando così una lista di coppie di termini $< t_m^d, t_n^c >$, con $t^d \in Term(D_i)$, $t^c \in Term(C_j)$, $m = 1...k_d$ e $n = 1...k_{C_j}$.

Ogni coppia di termini $< t_m^d, t_n^c >$ viene processata in WordNet valutando e memorizzando la presenza di una o più relazioni semantiche. Una volta processate tutte le coppie di termini ottenuti dal prodotto cartesiano, per ogni relazione semantica R appartenente alle 28 presenti in WordNet (vedi cap. 3.2.1) viene calcolato:

- la somma F_R di tutte le frequenze relative dei termini delle coppie interessate in R:

$$F_R = \sum_{<t^d,t^c>\in R} freqRel(t^d) + freqRel(t^c)$$

- La somma P_R di tutte le posizioni, ovvero l'importanza all'interno della rappresentazione del documento o categoria, dei termini delle coppie interessate in R:

$$P_R = \sum_{<t^d,t^c>\in R} (k_d - rank(t^d)) + (k_c - rank(t^c))$$

- Il rapporto R_R tra il numero di coppie interessate in R e il totale del numero di coppie:

$$R_R = \frac{|R|}{|k_d \times k_{C_j}|}$$

- La somma $Measure_R$ dei valori della misura di feature selection scelta e tramite la quale sono stati selezionati i termini rappresentativi per D_i e C_j, per le coppie interessate in R:

$$Measure_R = \sum_{<t^d,t^c>\in R} measure(t^d) + measure(t^c)$$

  Si noti che measure può assumere i valori dei $tfidf$ o $tfidf^{type1}$ ecc.

Ogni istanza del data-set, ovvero ogni coppia $< D_i, C_j >$, sarà caratterizzata da 116 attributi:

- Un attributo nominale, che rappresenta la classe della classificazione, indicante la relazione tra D_i e C_j (*SIMILAR*, *DISSIMILAR* o *HYPONYM*).

- Due attributi, non utilizzati nel processo di training e di categorizzazione ma utili per le statistiche, indicanti l'id del documento D_i e quello di C_j.

- Quattro attributi: F_R, P_R, R_R, $Measure_R$, descritti precedentemente, per ognuna delle 28 relazioni semantiche di WordNet.

- Un attributo R_{tot} rappresentante la somma di tutte le relazioni trovate:

$$R_{tot} = \sum_R R_R$$

3.3.1 Generazione data-set

Ogni processo di classificazione prevede l'utilizzo di un set di training, in cui è nota la categoria di appartenenza di ogni documento e tramite il quale viene addestrato il classificatore, e uno di test, con il quale si testa l'efficacia del modello generato.

Come descritto in precedenza, il training set viene creato tramite una serie di coppie documento-categoria, distinte in base alla similarità, iponimia o dissimilarità fra essi. Nel sistema, questo insieme di training, viene creato tramite alcuni parametri modificabili ad ogni run. Dato un numero $n_{docTraining}$ di documenti facenti parte del training set, per ogni documento $D_i \in TrainingDoc(D)$ vengono create:

- Una coppia, etichettata *SIMILAR*, ottenuta accoppiando a D_i la categoria di appartenenza C_i.
- $n_{hyponymCouples}$ coppie etichettate *HYPONYM*, ottenute da D_i e le categorie progenitori di C_i (padre, padre del padre, ecc..). In caso $|ancestor(C_i)| < n_{hyponymCouples}$, vengono create coppie fino all'arrivo della categoria *root*.
- $n_{dissimilarCouples}$ coppie etichettate *DISSIMILAR*. In questo caso possono essere previsti 3 tipi di selezione della categoria dissimile:
  1. Coppie create in maniera casuale tra D_i e una qualsiasi categoria dissimile $C_j \notin \{C_i \cup ancestor(C_i) \cup subTree(C_i)\}$.
  2. Coppie create utilizzando le categorie sorelle (figlie dello stesso nodo padre) alla categoria C_i di appartenenza di D_i, ovvero $C_j \in brother(C_i)$.
  3. Coppie create in maniera casuale, ma limitando la scelta della categoria C_j alle sorelle di C_i o alle sorelle dei nodi padri: $C_j \in brother(C_i \cup ancestor(C_i))$.

3.4 Classificazione gerarchica

La classificazione gerarchica riguarda il fulcro di tutto il sistema creato. Questa fase prevede la categorizzazione dei documenti appartenenti al test-set tramite una discesa top-down della gerarchia di categorie. Per ogni documento test $D_i \in TestDoc(D)$ è prevista una procedura ricorsiva che, partendo dalla *root*, esplorerà in maniera top-down i vari livelli della tassonomia fino ad ottenere la classificazione di D_i in una certa categoria C_j, nel caso in cui $i \equiv j$ la categorizzazione avrà portato al risultato corretto.

La procedura ricorsiva, arrivati al nodo C_j di livello $l-esimo$ nella tassonomia, prevede due principali funzioni:

1. Scelta di quale è il figlio di C_j migliore, su cui continuare la discesa della gerarchia a livello $l + 1$.

2. Stabilire se C_j è la categoria cui appartiene D_i, in caso positivo fermare la procedura iterativa e assegnare C_j come categoria di appartenenza di D_i

Per lo sviluppo di queste due funzioni si è utilizzato il metodo di classificazione proposto in precedenza. Viene inizialmente creata la rappresentazione del documento di test D_i tramite k_d termini più rilevanti, scelti esclusivamente con $tfidf^{type0}$ in quanto i tipi 1 e 2 prevedono la conoscenza pregressa della categoria di appartenenza di D_i, dato che va invece ignorato trattandosi di un documento di test. Ottenuta la rappresentazione di D_i, l'algoritmo ricorsivo prevede, al nodo C_j:

- Creazione di una coppia $< D_i, C_j >$, utilizzando la *rappresentazione singola* di C_j, per valutare l'effettiva similarità ($similarity(D_i, C_j)$) tra il documento e la categoria attuale.

- Creazione di una coppia $< D_i, C_{C_j} >$, per ogni $C_{C_j} \in Children(C_j)$ categoria figlia di C_j, rappresentate in maniera

bottom-up. In questa fase vengono valutati tutti i figli di C_j in base alla probabilità di predizione del classificatore che D_i e C_{C_j} siano *SIMILAR* o *HYPONYM*. Viene selezionato come nodo su cui proseguire la classificazione a livello inferiore, il nodo C_{C_j} avente maggiore somma di probabilità di essere *SIMILAR* e *HYPONYM* con il documento di test, definiamo questa somma $hyponymy(D_i, C_{C_j})$.

- Nel caso in cui $hyponymy_{best}(D_i, C_{C_j})$, ovvero il valore maggiore tra tutti gli $hyponymy(D_i, C_{C_j})$ di tutti i figli di C_j, sia minore di un valore di soglia, definita empiricamente in 0.2, nessuna categoria figlia è selezionata per la discesa gerarchica; si controllerà quindi se anche $similarity(D_i, C_j)$ sia minore o meno del valore di soglia. In caso anche $similarity(D_i, C_j) < 0.2$ la classificazione termina e il documento viene definito *unclassified*, in caso contrario viene assegnata C_j come categoria di appartenenza di D_i.

- Vengono confrontati i risultati ottenuti nei due punti precedenti, ovvero la similarità tra D_i e C_j e la somma tra la similarità e l'iponimia tra D_i e il migliore figlio C_{C_j}; nel caso $similarity(D_i, C_j)$ sia maggiore di $hyponymy_{best}(D_i, C_{C_j})$, la classificazione viene considerata conclusa, in quanto la categoria attuale è più simile rispetto all'iponima del miglior figlio, che, in quanto rappresentato in maniera bottom-up, rappresenta anche le categorie del sottoalbero sottostante, viene quindi assegnato il documento D_i a C_j.
  Se $hyponymy_{best}(D_i, C_{C_j}) > similarity(D_i, C_j)$ l'algoritmo prosegue in una nuova iterazione con la nuova categoria C_{C_j}. In caso C_{C_j} sia un nodo foglia, l'algoritmo termina assegnando C_{C_j} come categoria di appartenenza di D_i.

  Un'ulteriore personalizzazione del sistema può prevedere la scelta di più di un nodo da esplorare al ciclo successivo dell'algoritmo, ovvero si hanno $n_{openNode}$ categorie *attuali* ad

ogni ciclo, dalle quali verranno scelti gli $n_{openNode}$ figli con maggiore $hyponymy(D_i, C_{C_j})$. Questa funzione è stata creata in modo da ovviare eventuali errori di classificazione, soprattutto negli alti livelli della gerarchia, per documenti appartenenti a nodi più bassi in H, man mano che viene scorsa la tassonomia la classificazione diventa più efficace. Tenendo quindi aperte più strade viene ridotta la probabilità di eliminare il percorso corretto per il documento di test. Si noti che con $n_{openNode} = 1$ il procedimento risulta essere il caso classico descritto inizialmente.

3.4.1 Un classificatore generale

Data la costituzione intrinseca del metodo di costruzione del modello di classificazione, basata esclusivamente sulle relazioni semantiche, il nostro sistema può garantire l'utilizzo di un solo classificatore generale, utilizzato per la valutazione delle probabilità di similarità, dissimilarità e iponimia per ogni categoria presente in H.

In questo caso quindi il classificatore verrà creato tramite l'accoppiamento di tutti i documenti appartenenti al training set con le varie categorie così come descritto in 3.3.1. Tramite le API *Weka* verrà creato il classificatore che sarà utilizzato in ogni fase dell'algoritmo descritto in precedenza, per ogni confronto del documento test e una categoria $< D_i, C_j >$, verrà creata l'istanza generata dall'accoppiamento dei termini rilevanti di D_i e C_j, interrogando poi il classificatore sulla probabilità di appartenenza della coppia alle 3 classi di classificazione (*SIMILAR*, *DISSIMILAR* o *HYPONYM*).

3.4.2 Un classificatore per ogni categoria

Nel sistema è prevista anche la possibilità, come nella maggior parte dei metodi proposti in letteratura, di effettuare la classifi-

cazione gerarchica tramite n classificatori, uno per ogni categoria di H.
In questa modalità ogni classificatore si riferisce a una sola categoria; per ognuno di essi sono utilizzati i documenti di training, per ogni documento D_i è creata la coppia $< D_i, C_{C_j} >$ con la categoria C_j in esame, assegnando la relativa classe di relazione. In questo modo tutti i classificatori saranno creati con lo stesso numero di istanze di training, mentre varieranno per ogni categoria il numero di istanze di documenti simili, iponimi e dissimili.

Una volta creati gli n classificatori si procede con la procedura iterativa descritta in precedenza, la differenza con il caso di un classificatore unico è che, ad ogni ciclo dell'algoritmo, viene utilizzato, per ottenere le probabilità di similarità e iponimia del documento di test, il classificatore relativo alla categoria C_j corrente.

3.4.3 Parametri del sistema

All'interno del framework è stato previsto un massiccio utilizzo di parametri, tramite i quali modificare i comportamenti del sistema. Tali parametri sono stati inseriti in un file di configurazione XML, in modo che ogni simulazione di categorizzazione potesse essere facilmente configurata.
Verranno elencati in seguito i principali parametri previsti:

- Utilizzazione di un solo classificatore generale o di un classificatore per ogni categoria.
- $n_{openNode}$ nodi da esplorare ad ogni ciclo iterativo.
- Tipo di classificatore Weka da utilizzare.
- k_d e k_c termini rappresentativi di documenti e categorie.

- Misura di ordinamento e selezione dei termini rappresentativi per documenti e categorie in rappresentazione singola e bottom-up (es. $tfidf^{type2}$).
- Sottolivelli del $subTree(C_j)$ utilizzati nella rappresentazione bottom-up di C_j.
- Modalità di scelta delle categorie dissimili accoppiate ad ogni documento di training.
- Numero di categorie dissimili e iponime accoppiate ad ogni documento di training.
- Numero documenti di training e test.
- Profondità della ricerca delle relazioni tra due termini in WordNet.
- Seed random, in modo da rendere riproducibili le simulazioni: a parità di tutti gli altri parametri, i risultati possono cambiare solo se cambia il seed.

Capitolo 4

Architettura del sistema

In questo capitolo verrà descritta l'implementazione pratica del sistema, la quale ha permesso di effettuare le simulazioni e i test che verranno elencati nel capitolo successivo.

Un diagramma riassuntivo del comportamento del sistema, descritto nel capitolo precedente, è dato in Fig. 4.1.

4.1 Architettura del sistema

Il sistema Java è stato implementato utilizzando Eclipse [65], interfacciato ad un database PostgreSQL. Le classi del progetto sono state suddivise in package; seguirà l'elenco di tali classi con una breve descrizione di ognuna di esse:

Package textcat: Questo package generale contiene le classi eseguibili del sistema:

- `DataFromSourceToDataBaseMain`: classe eseguibile per la fase di preprocessamento dei dati.
- `HierarchicalClassificationMain`: contiene il main del sistema di classificazione gerarchico vero e proprio.

Package textcat.control: In questo package sono contenute le classi incaricate del controllo nelle varie fasi del processo:

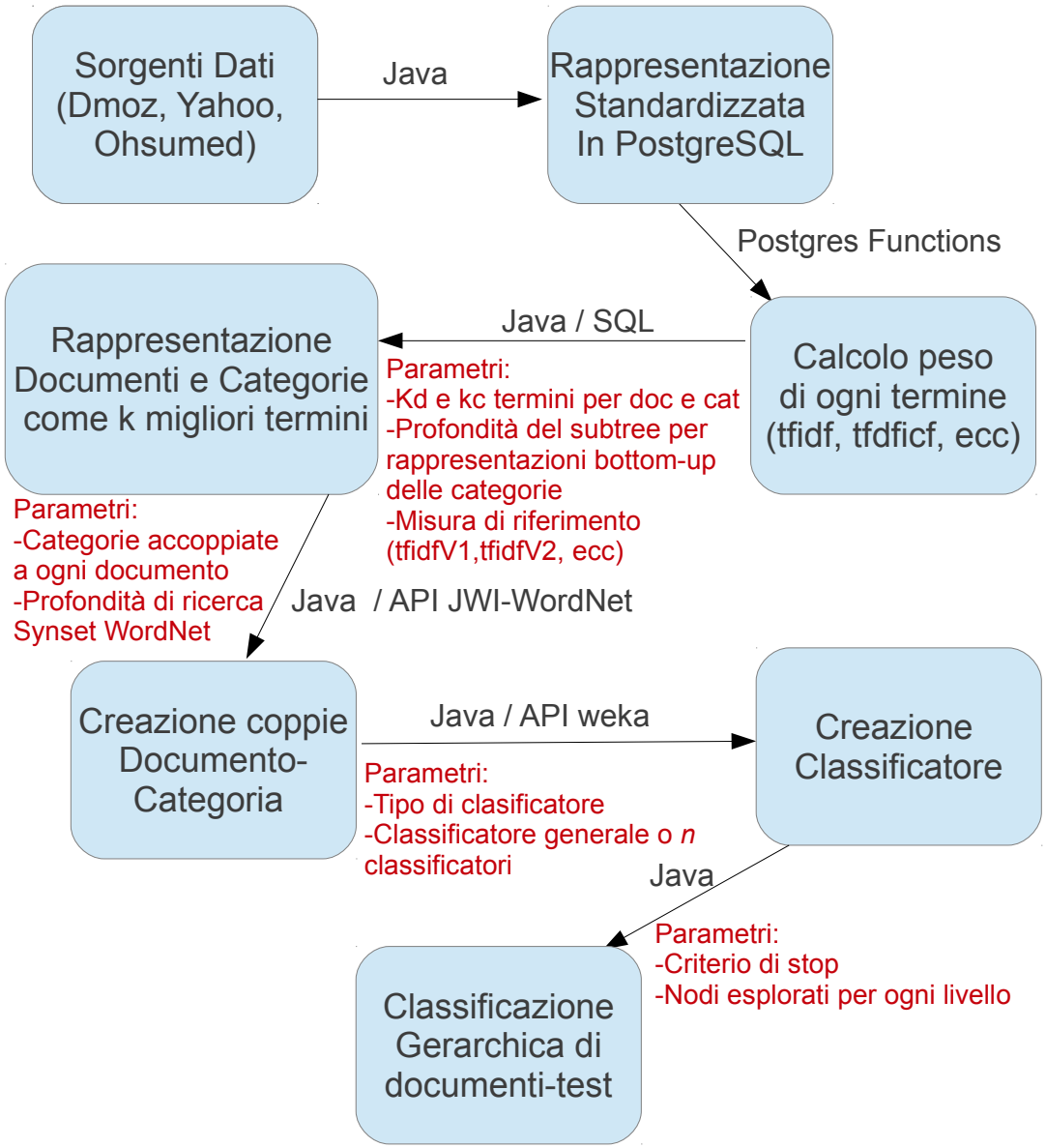

Figura 4.1: Diagramma di flusso riassuntivo del comportamento del sistema.

- `DataToDataBase`: incaricata, tramite il metodo pubblico `writeData` di scrivere i dati nel database Postgres.
- `HFAbstract`: classe astratta contentete i metodi comuni nei casi di un classificatore generale e n classificatori. Contiene nel costruttore la sezione di codice che crea la rappresentazione dei documenti e delle categorie nel sistema; un parser XML per il file di configurazione e il creatore di coppie documento-categoria con la relativa ricerca di relazioni WordNet. (nota: HF sta per Hierarchical Framework).
- `HFNClassifier`: estende `HFAbstract`, aggiungendo i metodi per la creazione di training e test-set, la classificazione semantica e la classificazione gerarchica vera e propria a un classificatore generale.
- `HFOneClassifier`: estende `HFAbstract`, aggiungendo il metodo per la creazione di training e test-set e la classificazione gerarchica a n classificatori.
- `RepresentationFromDataBase`: classe che crea la rapprestazione dei documenti e delle categorie a partire dai dati presenti nel database.
- `ResultToDataBase`: utilizzata per scrivere una tupla riguardante il risultato di ogni singola simulazione.
- `WordNetController`: contiene il metodo `match()` tramite il quale vengono ricercate le relazioni semantiche tra due parole in WordNet.

Package textcat.dataSource: Contiene le classi realizzate ad-hoc per ogni sorgente di dati, tramite le quali ricavare una rappresentazione standardizzata in Java partendo da sorgenti eterogenee, utilizzando anche i vari filtri per la fase di term extraction.

- `DataSetReaderAbstract`: Classe astratta generale, contenente i metodi comuni a tutte le sorgenti dati, come il calcolo dei parametri per ogni termine.
- `OhsumedReader`: lettura della sorgente Ohsumed, partendo da documenti testuali contenenti titolo e contenuto della pubblicazione.
- `WebClassReader`: classe utilizzata per la lettura delle sorgenti dati ottenute da WebClass (DMoz e Yahoo!), i documenti consistono in siti web.

Package textcat.filters: Contiene i filtri utilizzati nella fase di term extraction:

- `IFilter`: interfaccia generica implementata da tutti filtri, il metodo principale è `filter(String word)`.
- `Casefolder`: applica la conversione a lowercase di tutti i caratteri.
- `NumberRemover`: elimina ogni eventuale numero nella parola.
- `PunctuationRemover`: elimina ogni segno di punteggiatura dalla parola, per ogni segno di punteggiatura rilevato vengono generati due nuovi termini (es: *oppress'd* si tramuta in *oppress* e *d*, che verrà poi identificata come stopword ed eliminata).
- `StopWordsRemover`: confronta ogni termine con una lista di parole comuni e non discriminative nel linguaggio inglese, elimina inoltre ogni termine composto da meno di tre caratteri.
- `WordFilters`: Classe generale che implementa il vero filtro utilizzato nel term extraction, consiste nell'applicazione consecutiva dei filtri precedentemente descritti.

Package textcat.filters.en: In questo package sono contenuti i filtri specifici per la lingua inglese, si noti che tutti dataset utilizzati in questa tesi consistono in documenti in lingua inglese.

- `PluralToSingularFilter`: applica le formule ricavate dalla grammatica inglese per portare ogni parola plurale al proprio singolare.
- `PorterStemmer`: implementa lo stemmer inventato da Porter. Non è utilizzata nel sistema.

Package textcat.model: Contiene le classi riguardanti il modello dei dati del sistema, utilizzate dalla parte di controllo del framework.

- `Category`: modella una categoria della gerarchia.
- `CategoryRepresentation`: estende il modello della categoria inserendo la rappresentazione tramite una lista termini rilevanti.
- `Taxonomy`: rappresenta la tassonomia delle categorie del data-set. Contiene i metodi per la ricerca di una specifica categoria o il suo sottoalbero.
- `CoupleAttributes`: modella tutti gli attributi relativi a una coppia documento-categoria.
- `CoupleRelation`: rappresenta i quattro parametri descrittivi per una relazione di WordNet per una coppia documento-categoria.
- `Document`: modella un documento della collezione.
- `DocumentRepresentation`: estende il normale modello di un documento aggiungendo una lista di termini rilevanti e rappresentativi del documento stesso.

- `DocumentCollection`: rappresenta la collezione dei documenti.
- `Term`: rappresenta un termine in un documento. Contiene oltre la parola, l'identificativo del documento di appartenenza e i valori di occorrenza, frequenza relativa ecc. del termine.
- `RelevantTerm`: estende `Term`, aggiungendo il valore della misura che è stata scelta per ordinare e selezionare i termini durante la fase di term selection.
- `WordNetRelation`: modella una singola relazione WordNet inserendo il concetto di profondità della relazione stessa.

Package tectcat.util:

- `LogFile`: crea un file di log nel quale vengono memorizzate tutte le azioni del sistema durante ogni simulazione.
- `ParserDelegatorUtil`: implementa un parser che elimini ogni tag HTML da una stringa.
- `Util`: contiene alcuni metodi statici di utilità comune.
- `WekaUtil`: contiene metodi statici utilizzati per l'interfacciamento a Weka, in particolare la creazione di dataset e file arff.

4.2 Preprocessamento dei dati

La prima fase prevista nel framework riguarda il preprocessamento dei dati, che dalle proprie sorgenti devono essere standardizzati e inseriti in database PostgreSQL. Essendo le sorgenti dei dati di natura eterogenea (siti web o estratti di riviste scentifiche), e in previsione di nuovi studi su eventuali altre sorgenti,

si è creata la classe astratta `AbstractDataSetReader`, che contiene un campo riguardante la collezione dei documenti e uno per la tassionomia di categorie.
Per ogni sorgente di dati è stata creata una classe ad-hoc estendente `AbstractDataSetReader`; tale classe deve contenere i metodi per poter ricavare il contenuto di ogni documento e da esso estrapolare ogni singola parola, inoltre viene memorizzata la completa gerarchia di categorie.

Nella fase appena descritta avviene il preprocessamento dei dati. Il corpo di ogni documento viene processato inizialmente tramite `ParserDelegatorUtil`, che elimina ogni tag html presente nel documento. Ottenuta la lista di termini all'interno del documento, ogni parola viene processata tramite la classe `WordFilter`, che comprende una lista di filtri tutti implementanti l'interfaccia `IFilter`. `WordFilter` ha il preciso compito descritto in 3.1 di *Term Extraction*, ovvero vengono applicati i filtri di: rimozione punteggiatura, rimozione numeri, conversione al lowercase, conversione dal plurale al singolare, rimozione stop words.
Tutti i termini ottenuti alla fine del preprocessamento sono inseriti nella rappresentazione del documento, aumentando l'occorrenza di una particolare parola in caso fosse già contenuta nella rappresentazione. Per ogni termine vengono memorizzate occorrenza, posizione, posizione media, deviazione della posizione e frequenza relativa, calcolata come:

$$freqRel(t_i^d) = \frac{occurrence(t_i^d)}{\sum_{t^d \in d} occurrence(t^d)}$$

.

Una volta terminato il preprocessamento - o Term Extraction - dei dati, questi vengono standardizzati e memorizzati in un database PostgreSQL; si avrà in questo modo un database per ogni sorgente di dati. Ogni database è composto dalle seguenti tabelle:

- ***Categories***(*idcategory* integer *PK*,*path* varchar(20), *name* varchar(50), *idfathercategory* integer *FK*): contiene una tupla per ogni categoria, con le relative informazioni di base.

- ***Documents***(*iddocument* integer *PK*, *totword* integer, *idcategory* integer *FK*): contiene una tupla per ogni documento, con le relative informazioni basilari.

- ***Terms***(*iddocument* integer *PK FK*, *word* varchar(35) *PK*, *occurrence* integer, *freqrel* real, *tfidf* real, *tfidfexcat* real, *tfidfexsubcat* real, *tfdficf* real, *positionfirst* real, *positionavg* real, *positionstandarddev* real, *positionfirstnorm* real, *positionavgnorm* real, *positionstandarddevnorm* real, *intitle* boolean, *rank* integer): contiene tutti i termini di ogni documento della collezione, per ogni termine sono memorizzati tutti i diversi valori di weighting dello stesso, come ad esempio i tre diversi tfidf.

- ***Taxonomy***(*id* serial *PK*, *idfathercategory* integer *FK*, *idchildcategory* integer *FK*): rappresenta la tassonomia di categorie della sorgente, per ogni categoria sono espressi i figli diretti.

- ***Result***(...): tabella utilizzata per mantenere salvati i risultati di ogni simulazione, contiene una serie di campi corrispondenti ai vari parametri del sistema.

Tutto il data-set ottenuto tramite la lettura delle sorgenti in Java viene quindi memorizzato sul database appena descritto, tramite la classe `DataToDataBase`. La standardizzazione e la memorizzazione in un database si è resa necessaria in quanto la fase di weighting dei termini consiste in un processo lento e computazionalmente dispendioso, che deve essere fatto una sola volta per ogni sorgente. In questa maniera si hanno tutte le sorgenti in un formato standard, tramite il quale è possibile

con stored procedure - più efficienti dei corrispondenti metodi in Java - calcolare i diversi valori delle feature selection utilizzate nel sistema.

4.3 Dimensionality reduction

Il procedimento precedentemente descritto è stato effettuato una sola volta per ogni sorgente. Mentre a partire dalla decisione della feature selection da utilizzare per la dimensionality reduction il sistema torna ad essere implementato in Java e diventa parametrico, in modo da poter variare ad ogni simulazione il comportamento del sistema.
In particolare, la creazione delle rappresentazioni di documenti e categorie attraverso i termini più significativi, viene fatta in Java tramite la classe `RepresentationFromDataBase`.
La rappresentazione di ogni documento (`DocumentRepresentation`) estende la classe `Document`, aggiungendo ai normali campi una lista di `RelevantTerm` rappresentanti i termini rilevanti, ognuno dei quali ha specificato il valore della frequenza relativa all'interno del documento, un valore *rank* indicante la posizione in termini discriminativi del termine nel documento e il valore della misura utilizzata per ordinare i termini (es. $tfidf^{type2}$).
Il metodo `createDocumentsRepresentation()` crea la rappresentazione di tutti i documenti, restituendo la collezione di documenti `DocumentCollection`; vengono ordinanati inizialmente tutti i termini di ogni documento in base al parametro specificato, selezionando poi da tale elenco i primi k_d termini tramite i quali verrà rappresentato il documento.

Un procedimento analogo viene effettuato per la rappresentazione delle categorie, alle quali però è associata una lista di termini rilevanti ottenuta dall'insieme dei termini contenuti in tutti i documenti appartenenti ad ogni categoria; in questo caso per la frequenza relativa e il valore della misura, per termini

occorrenti in più documenti nella categoria, vengono utilizzati i valori medi.
I metodi `createCategoriesRepresentation` e `createCategoriesRepresentationBottomUp` creano le rappresentazioni per le categorie in maniera singola e bottom-up; il procedimento è simile per entrambi i metodi, con la differenza che l'insieme di k_c termini nel primo caso è ottenuto tra i soli appartenenti ai documenti nella categoria, mentre nel secondo l'insieme è esteso a tutti i documenti nel sottoalbero della categoria, selezionando k_c termini per ogni categoria appartenente al sottoalbero.

4.4 Creazione classificatore

Ottenute le rappresentazioni di documenti e categorie viene creato il training-set, tramite il quale addestrare il classificatore. Il training set, così come descritto in 3.3.1, viene creato tramite le relazioni tra parole riconosciute da WordNet.
L'utilizzo in Java di WordNet è stato reso possibile tramite le API *JWI 2.2.3* [64], utilizzate all'interno della classe `WordNetController`, questa libreria permette la navigazione dei synset tra le varie relazioni (`Pointer`) in WordNet. La ricerca delle relazioni tra due parole viene effetuata tramite il metodo `match()`, nel quale per ognuno dei 28 `Pointer` presenti in JWI, viene valutata la presenza della seconda parola all'interno dei synset della prima, in caso negativo si scende di livello per la relazione, ovvero si valutano tutti i nuovi synset creati a partire da ogni parola contenuta nel synset del livello precedente; si procede in questa maniera fino a un livello massimo, specificato in configurazione.
Il training-set viene quindi creato, per il sistema a un classificatore unico, accoppiando ad ogni documento una serie di categorie (cap 3.3.1) e contenente 116 attributi, 112 dei quali relativi

al riconoscimento delle relazioni su WordNet. Una volta ottenuta la lista di istanze delle varie coppie viene creato un file arff tramite il metodo statico `createArffFile()` di `WekaUtil`.

La fase seguente prevede la creazione del classificatore sulla base del training-set appena creato. Il classificatore viene creato utilizzando le API Weka e creando un'istanza di `Classifier` corrispondente al classificatore richiesto in fase di configurazione. Tale classificatore viene poi addestrato utilizzando il file arff creato in precedenza, a tal punto si è quindi pronti per passare alla fase di test del sistema.

4.5 Classificazione gerarchica

Tutto il processo a valle del Term Extraction viene controllato tramite le classi `HFOneClassifier` e `HFNClassifier`, a seconda che si sia scelto di utilizzare un unico classificatore generale o uno per ogni categoria, entrambi estendenti `HFAbstract` che contiene i metodi comuni a entrambe le classi, come il parser XML per la lettura del file di configurazione, la creazione della rappresentazione di documenti e categorie, la creazione del training e test-set.
La classificazione gerarchica vera e propria dei documenti test è effettuata all'interno del metodo `testSetClassification()` di tali classi, questo metodo prevede l'implementazione del processo iterativo di classificazione per ogni documento di test descritto in 3.4. La classificazione di ogni documento di test avviene valutando la similarità tra un categoria *current* e l'iponimia con ognuna delle categorie figlie di quella corrente; partendo dal nodo radice si valutano quindi le coppie formate dal documento e le categorie in esame, valutando a ogni iterazione il figlio migliore su cui proseguire la classificazione e la possibilità di terminare la classificazione assegnando la categoria attuale al documento. In caso di classificazione a n classificatori, ogni confronto tra

documento e categoria, corrente e figlie, viene effettuato utilizzando il classificatore relativo alla categoria in esame.
La possibilità di scelta di esplorazione di più di un figlio per il livello successivo a ogni iterazione, comporta l'inserimento di $n_{openNode}$ categorie *current* a ogni livello, vengono valutati indistintamente tutti i figli di queste categorie, scegliendo gli $n_{openNode}$ migliori su cui proseguire la ricerca.

Nel caso in cui sia il migliore dei figli, sia il migliore dei nodi correnti, hanno similarità minore del valore di soglia 0.2, il processo termina e si etichetta il documento in esame come *unclassified*. La classificazione termina su una determinata categoria in due occasioni:

1. Si è giunti a un nodo foglia, viene assegnata la migliore categoria figlia al documento.

2. La categoria corrente ha similarità, in rappresentazione singola, migliore dell'iponimia del migliore dei figli, in rappresentazione bottom-up. In questo caso si etichetta il documento come appartenente alla categoria corrente.

La bontà del processo di classificazione viene valutata attravarso i principali metodi di misura dell'efficacia nel text mining, così come descritto in 1.8, ovvero si sono utilizzati i valori di Precision, Recall e F-measure. Terminata la categorizzazione di ogni documento test si verifica l'esito di tale classificazione, valutando, in caso di categorizzazione errata, la tipologia dell'errore, discriminando in quattro tipi [8]:

- *Misclassification error*: documenti classificati in una categoria che non ha alcuna relazione con quella corretta.

- *Specialization error*: documenti classificati in una categoria più specifica - iponima - rispetto a quella corretta. In pratica ciò significa che l'algoritmo non è terminato abbastanza presto.

- *Generalization error*: documenti classificati in una categoria più generale - iperonima - di quella corretta. Ovvero l'algoritmo è terminato troppo presto.

- *Unclassified ratio*: documenti non classificati, scartati in quanto sotto il livello di soglia nel processo iterativo.

Un ulteriore metodo di misura della qualità dell'errore nella categorizzazione di un documento di test è stata effettuata tramite la creazione di una nuova misura, simile all'*H-loss* descritto in in [39] ma adattato per la classificazione single-label. Questa misura, denominata *tree distance error*, quantifica la distanza tra la categoria assegnata a un documento dall'algoritmo e quella a cui realmente il documento apparterrebbe. Tale distanza è stata specificata in modo che, data la categoria corretta C_c e quella assegnata dal classificatore C_a, si ha che:

$$td_{error}(C_a, C_c) = \begin{cases} 0 & C_c \equiv C_a \\ 1 & C_c \in \{brother(C_a) \cup father(C_a) \cup \\ & children(C_a)\} \\ 2 & C_c \in \{brother(father(C_a)) \cup father \\ & (father(C_a)) \cup children(children(C_a))\} \\ 3 & .... \end{cases} \tag{4.1}$$

Un esempio dei valori assunti dal *tree distance error* è situato in figura 4.2. Il significato pratico di questa misura può essere definito in quante classificazioni vengono errate nel processo gerarchico, quando infatti *tree distance error* vale 1, signica che l'ultima classificazione è stata errata, e così via all'aumentare del valore di questa misura.
Il *tree distance error* è stato utilizzato in questa tesi come metodo di valutazione degli errori, utilizzando il valore medio delle

sole istanze classificate erroneamente:

$$td_{errorAvg} = \frac{\sum_{C_a \neq C_c} td_{error}(C_a, C_c)}{|Error|}$$

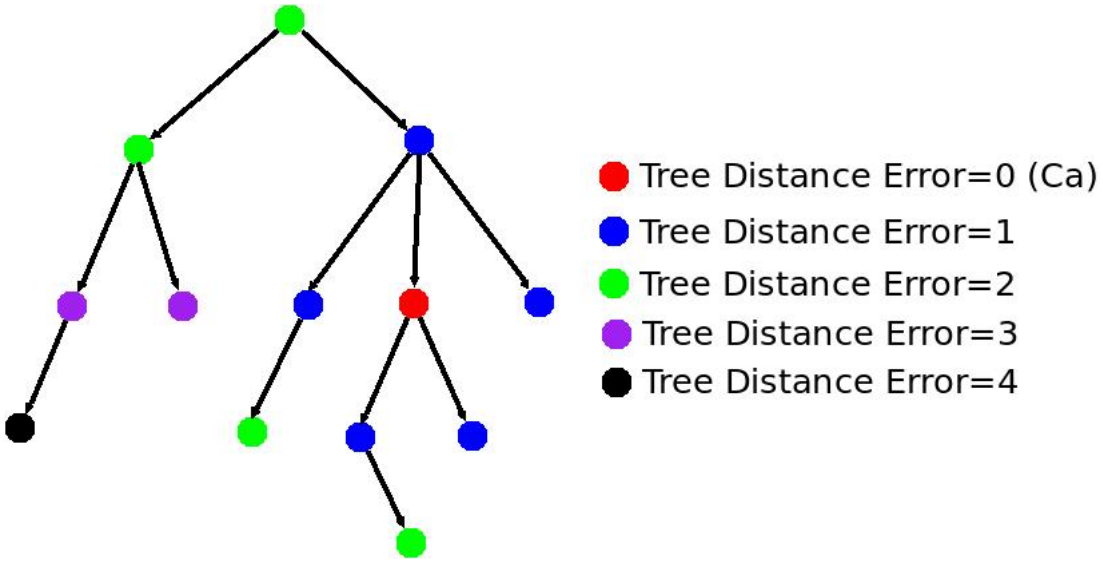

Figura 4.2: Valori assunti dall'*tree distance error* per una data C_a (in rosso).

Capitolo 5

Esperimenti e Risultati

Avendo creato un sistema altamente parametrico, si sono resi necessari una grande quantità di test per poter valutare l'apporto di ogni singolo parametro per la classificazione. In questo capitolo verranno descritti i risultati ottenuti dalle varie simulazioni del sistema al variare dei parametri, mostrando il risultato per ognuna delle tre sorgenti dati considerate.
Verrà inizialmente valutata l'efficacia della classificazione semantica di coppie documento-categoria. Sarà poi valutata la classificazione gerarchica, che assegna, tramite un algoritmo di discesa dell'albero, una categoria a ogni documento test. Mostrando in seguito la possibilità di creare un meta-classificatore utilizzabile su categorie, e conseguentemente anche sorgenti dati, non noti in fase di training nel modello.
Si effettuerà infine un confronto dei risultati ottenuti con quelli reperiti in letteratura.

5.1 Classificazione semantica

In questa sezione vengono proposti i diversi test effettuati per valutare l'effettiva abilità del classificatore creato nel giudicare la relazione semantica tra un documento e una categoria. Questi test si sono poi confrontati con quelli ottenuti dall'ing. Magnani

nella propria tesi [59] per ottenere un riscontro effettivo dei risultati ottenuti.
La valutazione semantica degli algoritmi è stata ottenuta creando il training-set nella medesima maniera della classificazione gerarchica (5.2), mentre il test-set è stato ottenuto creando un insieme di coppie documento-categoria, utilizzando i documenti di test e accoppiandoli seguendo le stesse metodologie utilizzate per la creazione del training-set. Si testerà in questa maniera il classificatore con una serie di istanze per ogni documento di test, relative a categorie simili, dissimili e iponime.

DataSet	**Accuracy**	**Precision**	$\mathbf{F_1}$
Yahoo!	0.948	0.871	0.897
DMoz	0.864	0.445	0.576
Ohsumed	0.927	0.895	0.673

Tabella 5.1: Classificazione semantica di coppie documento-categoria, classificate secondo la relazione di similarità, dissimilarità e iponimia.

Dalla tabella 5.1 risulta chiaro l'ottimo risultato conseguito da questa classificazione, i risultati si aggirano attorno al 90%, indicando che il modello creato basato sulle relazioni semantiche ricavate da WordNet è in grado di fornire una predizione della relazione semantica corrente fra un documento e una categoria più che eccellente.

5.2 Classificazione gerarchica

La valutazione del metodo proposto per la classificazione gerarchica è stata effettuata tramite diverse simulazioni effettuate sulle collezioni di dati.
Dato l'alto numero di parametri presenti nel sistema, la valutazione dell'apporto di ognuno di essi è stata effettuata a partire

da una configurazione standard, ovvero la parametrizzazione che su ogni dataset garantiva il miglior compromesso tra accuratezza e complessità computazionale (che può essere ricondotta al valore di k_d, k_c e $n_{openNode}$). Tale configurazione, denominata "standard", si è rivelata essere:

- Classificatore di Weka selezionato: *RandomForest*.
- Training-set formato dalla categoria di appartenenza di ogni documento ($SIMILAR$), tutte le categorie iperonime a quella di appartenenza fino al nodo radice dell'albero ($HYPONYM$) e 20 categorie prelevate in maniera casuale tra le categorie sorelle della categoria o dalle sorelle degli antenati ($DISSIMILAR$).
- Feature selection utilizzata per l'estrazione dei termini rilevanti di documenti e categorie: $tfidf^{type2}$.
- Rappresentazione bottom-up delle categorie relativa a 4 sottolivelli nell'albero gerarchico.
- Numero di termini rilevanti selezionati per la rappresentazione di ogni documento e categoria: per *Yahoo!* $k_d = 5$ e $k_c = 60$, per *DMoz* $k_d = 15$ e $k_c = 200$, per *Ohsumed* $k_d = 10$ e $k_c = 100$.
- Numero di nodi esplorati ad ogni livello: 1 per *Yahoo!*, 5 per *DMoz* e *Ohsumed*.
- Profondità massima per la ricerca delle relazioni tra due termini in WordNet $depthMax = 3$.

Ogni collezione di documenti è stata suddivisa in modo che il training-set, tramite il quale addestrare il classificatore, consiste nel 40% dei documenti della collezione selezionati in maniera casuale, mentre il restante 60% compone il test-set.

Per ogni simulazione verrà mostrato, oltre all'accuratezza del classificatore (percentuale di istanze correttamente classificate), la *Precision* e la F_1 della classificazione. Inoltre, per la qualificazione degli errori, verrà mostrato anche il valore della distanza media gerarchica degli errori.

Un primo test effettuato ha riguardato l'apporto della feature selection utilizzata, variando dalla configurazione precedentemente descritta il metodo di ordinamento ed estrazione dei termini per la creazione delle rappresentazioni di documenti e categorie. In queste simulazioni è stato testata quindi l'accuratezza del classificatore al variare delle rappresentazioni. I risultati sono mostrati in tabella 5.4.

DataSet	**Feature selection**	**Accuracy**	**Precision**	$\mathbf{F_1}$	$\mathbf{td}_{error}$
Yahoo!	$freqRel$	0.467	0.739	0.572	**2.158**
	$tfidf^{type0}$	0.777	0.849	0.812	2.310
	$tfidf^{type1}$	**0.810**	**0.868**	**0.838**	2.484
	$tfidf^{type2}$	0.796	0.861	0.827	2.179
	$tfdficf$	0.250	0.539	0.342	2.419
DMoz	$freqRel$	0.152	0.194	0.170	**3.764**
	$tfidf^{type0}$	0.276	0.307	0.291	3.895
	$tfidf^{type1}$	0.270	0.301	0.284	3.892
	$tfidf^{type2}$	**0.367**	**0.369**	**0.368**	3.986
Ohsumed	$freqRel$	0.345	0.367	0.355	1.44
	$tfidf^{type0}$	0.466	0.478	0.472	**1.503**
	$tfidf^{type1}$	0.47	0.483	0.476	1.522
	$tfidf^{type2}$	**0.48**	**0.531**	**0.504**	**1.503**

Tabella 5.2: Risultati test sulla feature selection.

Si può notare come la scelta del $tfidf^{type1}$ e $tfidf^{type2}$ dia i risultati migliori. Questo risultato ha una particolare importanza data la novità di questo metodo di feature selection. Il punto di forza di questa funzione è che vengono favoriti i termini maggiormente discriminativi tra quelli presenti all'interno della sottogerarchia della categoria (o del documento) rappresentata.

Questo fattore ha un particolare impatto nella discesa gerarchica dell'algoritmo di classificazione in quanto i termini discriminano non solo la semantica della categoria al livello corrente, ma di tutto il sottoalbero sottostante la categoria.

Il secondo test effettuato riguarda il numero $n_{openNode}$ di nodi esplorati a ogni livello: questo valore incide sulla complessità computazionale del sistema in quanto l'aumentare di questo valore implica un aumento proporzionale delle categorie confrontate per ogni livello dell'algoritmo. I risultati sono mostrati in tabella 5.3.

In questo caso risalta il fatto che nel dataset *Yahoo!* sia più

DataSet	$\mathbf{n}_{openNode}$	**Accuracy**	**Precision**	$\mathbf{F}_1$	$\mathbf{td}_{error}$
Yahoo!	1	**0.796**	**0.861**	**0.827**	2.179
	2	0.785	0.841	0.812	2.017
	3	0.791	0.846	0.817	1.963
	4	0.787	0.842	0.813	1.990
	5	0.783	0.838	0.809	1.964
	6	0.785	0.840	0.811	**1.955**
DMoz	1	0.202	0.212	0.207	4.45
	2	0.280	0.283	0.282	4.295
	3	0.325	0.327	0.326	4.172
	4	0.347	0.348	0.347	4.069
	5	0.367	0.369	0.368	3.986
	6	0.385	0.386	0.385	3.949
	7	0.388	0.389	0.388	3.927
	8	0.397	0.398	0.397	3.868
	9	**0.401**	**0.402**	**0.401**	3.879
	10	0.398	0.4	0.399	**3.824**
Ohsumed	1	0.45	0.533	0.488	1.675
	2	0.481	**0.541**	0.509	1.533
	3	0.483	0.537	0.508	1.509
	4	**0.486**	0.538	**0.511**	**1.496**
	5	0.48	0.531	0.504	1.509
	6	0.48	0.531	0.504	1.509

Tabella 5.3: Risultati test al variare di $n_{openNode}$.

efficace la classificazione con un solo nodo esplorato per volta, mentre nelle altre sorgenti dati, come auspicabile, aumenta l'accuratezza della classificazione con l'aumentare di $n_{openNode}$. Questo parametro è stato aggiunto in particolare per superare le difficoltà riscontrate nella scelta della categoria al livello top, che è dovuta a due motivi principali: l'alto numero di categorie top (sia su DMoz che su Ohsumed), e l'alto livello nella gerarchia, che può contenere argomenti molto più generali rispetto a quelli trattati in documenti appartenenti ai livelli inferiori. Esplorando più nodi vengono mantenute aperte più strade nella gerarchia, avvicinandosi man mano alla specializzazione degli argomenti contenuti nel documento test. Questo miglioramento è constatabile effettivamente osservando i valori di td_{error}, il quale diminuisce all'aumentare dei nodi esplorati ad ogni livello, ciò implica che gli errori commessi sono meno gravi, ovvero le categorie predette erroneamente sono meno distanti da quelle corrette.

Per testare la scalabilità del sistema, particolare importanza è riservata alla rappresentazione bottom-up delle categorie, in quanto l'utilizzo di tale rappresentazione in gerarchie di argomenti molto profonde porterebbe a un aumento della complessità computazionale. Si è dunque testato il comportamento del sistema variando il numero di sottolivelli considerati durante la creazione della rappresentazione di ogni categoria. Il risultato di questi test è mostrato nella tabella 5.4.

Come auspicabile, i risultati migliorano aumentando il numero di livelli considerati. Vengono commessi meno errori nelle scelte per le categorie a livelli più alti (figura 5.1) in quanto, incrementando il numero di livelli considerati, viene aumentato il potere rappresentativo della categoria per la propria gerarchia sottostante. È però importante notare che l'aumento dell'efficacia della classificazione non è esponenziale quanto l'aggiunta di

DataSet	sottolivelli	Accuracy	Precision	F_1	td_{error}
Yahoo!	0	0.208	0.214	0.211	3.806
	1	0.526	0.735	0.613	3.085
	2	**0.796**	**0.861**	**0.827**	**2.179**
DMoz	0	0.072	0.114	0.088	**3.417**
	1	0.242	0.268	0.254	3.915
	2	0.429	0.443	0.436	4.018
	3	**0.460**	**0.472**	**0.466**	4.002
	4	**0.460**	**0.472**	**0.466**	4.045
Ohsumed	0	0.328	0.358	0.342	2.101
	1	0.48	0.491	0.485	1.628
	2	0.466	0.481	0.474	**1.440**
	3	0.47	0.482	0.476	1.512
	4	0.48	**0.531**	0.504	1.509
	5	**0.5**	0.515	**0.507**	1.493

Tabella 5.4: Risultati test al variare dei sottolivelli considerati nelle rappresentazioni bottom-up delle categorie.

sottocategorie nella rappresentazione bottom-up al variare dei livelli considerati. In particolare sia su *Yahoo!* che su *Ohsumed*, già con un solo livello considerato si ha un'ottima accuratezza nella classificazione.

La creazione del training-set, più specificatamente l'accoppiamento di ogni documento di training con le varie categorie, può essere effettuata in varie maniere. In particolare la scelta delle categorie dissimili accoppiate con ogni documento può variare in tre modalità (3.3.1), nella tabella 5.5 viene mostrato l'effetto di questo paramentro sull'efficacia del sistema. Nei test sono valutati i tre tipi di scelta di categorie dissimili: *brother* indica la selzione di tutti i fratelli della categoria cui appartiene il documento, *random* la selezione casuale e *superiorTree* la scelta casuale di categorie appartenenti ai livelli superiori del documento; i numeri a fianco la selzione indicano la quantità di categorie selezionate.

Figura 5.1: Distribuzione degli errori di classificazione nei livelli della gerarchia al variare dei sottolivelli considerati nelle rappresentazioni bottom-up delle categorie. Si noti che il livello 0 consiste nella radice dell'albero, mentre il livello 1 corrisponde alle categorie top.

DataSet	Dissimili	Accuracy	Precision	$\mathbf{F}_1$	$\mathbf{td}_{error}$
Yahoo!	Random 10	**0.804**	0.869	0.835	2.323
	Random 20	0.793	0.821	0.807	2.703
	Random 40	0.789	0.815	0.802	2.763
	Brother	0.798	0.856	0.826	2.390
	SuperiorTree 10	0.796	0.861	0.827	**2.179**
	SuperiorTree 20	0.796	0.861	0.827	**2.179**
	SuperiorTree 40	0.800	**0.885**	**0.841**	2.278
DMoz	Random 10	**0.458**	0.472	0.465	4.048
	Random 20	0.456	**0.495**	**0.475**	**3.955**
	Brother	0.412	0.438	0.425	3.986
	SuperiorTree 10	0.325	0.339	0.332	4.040
	SuperiorTree 20	0.367	0.369	0.368	3.986
Ohsumed	Random 10	0.478	0.488	0.483	1.479
	Random 20	0.458	0.512	0.483	1.461
	Brother	0.491	**0.533**	**0.511**	**1.439**
	SuperiorTree 10	**0.5**	0.515	0.507	1.493
	SuperiorTree 20	0.48	0.531	0.504	1.503

Tabella 5.5: Risultati test per la scelta di categorie dissimili accoppiate ai documenti di training.

L'efficienza e l'efficacia del sistema si sono rivelate dipendere molto dai valori di k_d e k_c scelti in fase di configurazione. In particolare, aumentare k_c rispetto a k_d si è dimostrato cruciale per ottenere una buona classificazione, un k_d troppo elevato evidentemente porta ad avere problemi di abbondanza di termini che però non sono presenti nella rappresentazione della categoria di appartenenza. All'aumento di k_c si è rivelato corrispondere un netto aumento dell'accuratezza della classificazione, d'altro canto però l'efficienza in termini di tempo della classificazione tende a peggiorare in quanto si aumentano i confronti di termini richiesti per la classificazione.
I risultati di questi test sono riportati nella tabella 5.7.

$\mathbf{k}_c \setminus \mathbf{k}_d$	**5**	**10**	**15**	**20**	**40**	**60**	**80**	**100**
20	0.653	0.601	-	-	-	-	-	-
40	0.758	0.724	-	-	-	-	-	-
60	0.796	0.796	-	-	-	-	-	-
80	0.846	0.814	-	-	-	-	-	-
100	0.865	0.808	0.808	0.795	0.762	0.720	0.724	0.726

Tabella 5.6: Risultati test al variare di k_c e k_d su *Yahoo!*, i valori riportati indicano l'accuratezza della classificazione.

$\mathbf{k}_c \setminus \mathbf{k}_d$**=20**	**Accuracy**	**Precision**	$\mathbf{F}_1$	$\mathbf{td}_{error}$
20	0.3	0.314	0.307	1.730
40	0.358	0.377	0.367	1.675
60	0.431	0.451	0.441	1.521
80	0.453	0.471	0.462	1.545
100	0.48	0.531	0.504	1.503
150	0.52	0.532	0.526	1.381
200	0.54	0.550	0.544	1.402

Tabella 5.7: Risultati test al variare di k_c con $k_d = 20$ su *Ohsumed*.

Nella tabella 5.8 sono mostrati i risultati ottenuti utilizzando diversi tipi di classificatori messi a disposizione da Weka.

Si può notare come il classificatore RandomForest sia quello con risultati migliori, ciò è dovuto alla caratteristica intrinseca di questo classificatore di essere più efficace rispetto agli altri con classi sbilanciate nel training-set, infatti l'insieme di training creato per l'addestramento contiene molte più istanze di classe dissimile rispetto ai simili e gli iponimi.

DataSet	**Classificatore**	**Accuracy**	**Precision**	$\mathbf{F_1}$	$\mathbf{td}_{error}$
Yahoo!	RandomForest	**0.796**	**0.861**	**0.827**	2.179
	SimpleCart	0.758	0.846	0.8	2.420
	J48	0.718	0.848	0.778	2.326
	Ibk	0.521	0.809	0.634	**2.164**
	SMO	0.670	0.670	0.670	2.220
DMoz	RandomForest	**0.367**	**0.369**	**0.368**	3.986
	SimpleCart	0.210	0.211	0.210	4.738
	J48	0.112	0.137	0.123	4.460
	Ibk	0.279	0.279	0.279	**3.597**
	SMO	0.163	0.165	0.164	4.051

Tabella 5.8: Risultati test utilizzando diversi classificatori di Weka.

La possibilità di avere n classificatori comporta alcune modifiche al modello, specialmente nella creazione dei training-set utilizzati per addestrare ogni classificatore. Per questi test si è utilizzato quanto più possibile la configurazione "standard" descritta precedentemente, con la differenza che per ogni classificatore si sono utilizzati tutti i documenti appartenenti al training-set, accoppiandoli con la categoria relativa al classificatore creato. Utilizzando questo metodo di classificazione si perde il vantaggio della generalità di un solo classificatore, creato una sola volta all'inizio del sistema, ma si guadagna in termini di specificità dei modelli con cui sono creati i classificatori, rendendo così più efficace la decisione di quale nodo selezionare per il proseguimento della discesa gerarchica dell'algoritmo. Il confronto dei risultati ottenuti con questo metodo rispetto a quello

con un classificatore generale sono riportati nella tabella 5.9.

DataSet	classificatore	Accuracy	Precision	$\mathbf{F}_1$	$\mathbf{td}_{error}$
Yahoo!	One class.	0.796	0.861	0.827	2.179
	n class.	0.595	0.698	0.643	1.71
DMoz	One class.	0.367	0.369	0.368	3.986
	n class.	0.460	0.472	0.466	4.045
Ohsumed	One class.	0.48	0.531	0.504	1.503
	n class.	0.13	0.132	0.131	1.91188

Tabella 5.9: Confronto fra i risultati ottenuti utilizzando un unico classificatore generale o n classificatori, uno per categoria

La valutazione degli errori commessi durante la classificazione è stata fatta, oltre alla misura della distanza gerarchica tra le categorie assegnate rispetto quelle corrette, in due maniere. Si è considerata la tipologia di errore commesso, discriminando in errori di specializzazione, generalizzazione, errore generale e documenti non classificati. In particolare il tipo di errore considerato "peggiore"è sicuramente il *misclassified*, in quanto questo errore è dovuto ad una classificazione completamente sbagliata del documento di test, mentre la specializzazione e la generalizzazione indicano che l'argomento del documento era stato individuato, ma l'algoritmo ha riscontrato qualche problema nell'assegnare il giusto livello di specializzazione del documento. Alcuni esempi delle distribuzioni delle tipologie degli errori commessi dal classificatore sono mostrati in figura 5.2.

L'ultima valutazione degli errori commessi nella classificazione è stata la scomposizione degli errori in base ai livelli della gerarchia in cui sono stati commessi. Dal grafico 5.1 si può notare come la maggior parte degli errori viene commessa a livelli alti delle gerarchie, particolarmente nelle categorie top. Anche questo dato conferma le difficoltà dell'algoritmo nella scelta della prima categoria, ovvero di relazionare le categorie a livello

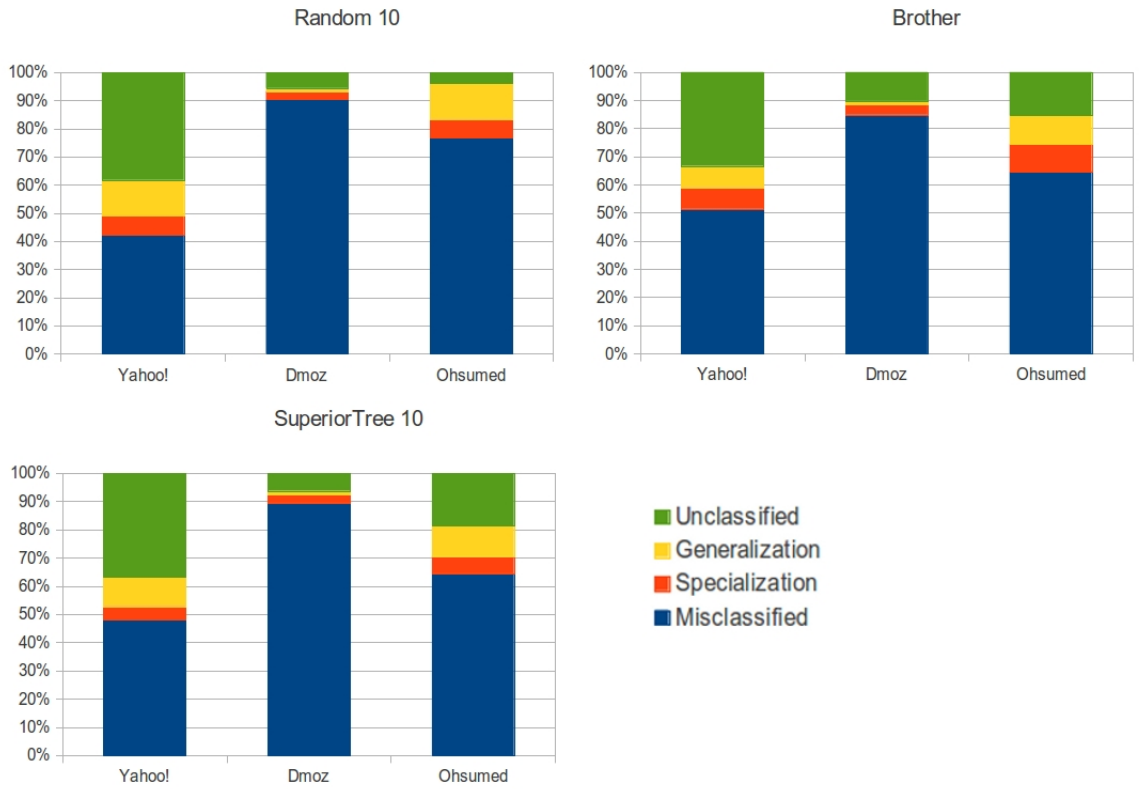

Figura 5.2: Distribuzione del tipo di errori di calssificazione commessi al variare del metodo di selezione dei *dissimilar* nei vari dataset.

alto con le discendenti ai livelli inferiori.
Inoltre, va notato dai risultati di tutti i test effettuati, che la distanza media nell'albero gerarchico tra la categoria corretta e quella assegnata è praticamente sempre compresa fra 1 e 3; ciò indica che il classificatore raramente assegna categorie completamente dissimili a un documento, ma, in media, il documento viene assegnato a una categoria sorella di un parente diretto (figlio o padre) di quella corretta.

5.3 Meta-classificazione: test su categorie non note al classificatore

Un aspetto assolutamente innovativo del metodo proposto in questa tesi è la possibilità di creare un classificatore che in realtà consiste in un meta-classificatore, ovvero che non è creato in base alle istanze delle categorie e dei documenti di training, ma dalle relazioni semantiche fra essi. Si crea in questo modo un classificatore indipendente dalle specifiche istanze di categorie e documenti utilizzati per la fase di learning, in grado quindi di classificare documenti di argomenti non noti in categorie non note.
Questo tipo di classificazione ha un impatto importantissimo per lo sviluppo di classificatori testuali, in quanto permette di creare un classificatore generale, creato su una certa collezione di documenti, e di esportarlo ed utilizzarlo su qualsiasi altra collezione.

La valutazione della meta-classificazione è stata fatta in due maniere. Per primo è stato effettuato un test nel quale il classificatore è istruito su una parte dell'albero e testato sulla restante; in pratica le categorie nel training e nel test-set hanno intersezione nulla, ciò significa che gli argomenti dei documenti di test sono completamente nuovi al classificatore. Un esempio

della modalità di divisione dell'albero è mostrato in figura 5.3. Tramite il dataset così formato si sono testati sia la classifi-

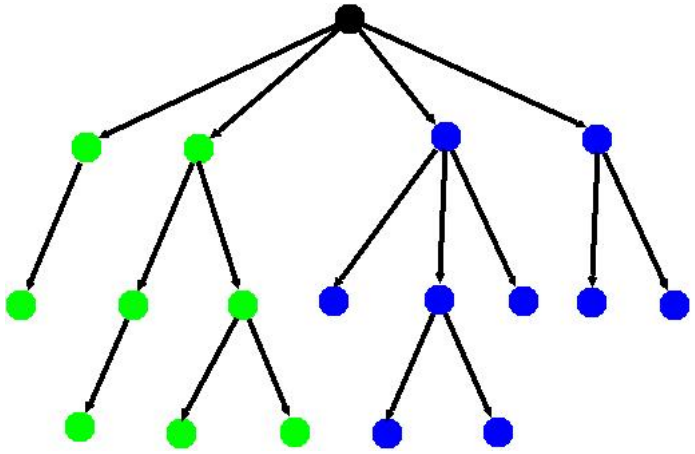

Figura 5.3: Divisione delle categorie di training (verdi) e test (blu) per la meta-classificazione interna a un dataset.

cazione semantica dei documenti, che valuta la relazione tra una coppia documento-categoria (cap. 5.2), sia l'algoritmo gerarchico per l'assegnazione di una categoria ai documenti test, escludendo, durante il processo di classificazione, la possibilità di assegnare una categoria appartenente all'albero di training ai documenti di test. I risultati sono mostrati in tabella 5.10.

Classificazione	**DataSet**	**Accuracy**	**Precision**	$\mathbf{F_1}$
Semantica	Yahoo!	0.935	0.892	0.905
	DMoz	0.907	0.775	0.698
	Ohsumed	0.926	0.809	0.658
Gerarchica	Yahoo!	0.824	0.885	0.854
	DMoz	0.379	0.390	0.385
	Ohsumed	0.570	0.777	0.658

Tabella 5.10: Meta-classificazione nei vari dataset, dividendo l'albero gerarchico delle categorie al livello top tra training e test.

Si può notare come i risultati siano assolutamente in linea con quelli ottenuti nei casi in cui le categorie di training e test fossero

le stesse, in alcuni casi addirittura migliori. Questo risultato indica chiaramente come il classificatore creato sia indipendente dalle specifiche istanze di categorie e documenti utilizzati per le fasi di training e di test.

Un ulteriore test, degno di particolare nota, ha previsto l'utilizzo di un classificatore addestrato con un un training-set appartenente a una collezione di dati e poi testato con il test-set di una diversa collezione. In pratica si è testato il classificatore di ognuna delle tre sorgenti dati a disposizione con le altre due. Anche in questo caso si è testato il sistema sia per quanto riguarda la classificazione semantica, sia per quanto riguarda la classificazione gerarchica dei documenti. I risultati sono mostrati nella tabella 5.11.

Classificaz.	**Training-Set**	**Test-Set**	**Accuracy**	**Precision**	$\mathbf{F_1}$
Semantica	Yahoo!	DMoz	0.772	0.65	0.015
		Ohsumed	0.766	0.802	0.298
	DMoz	Yahoo!	0.852	0.517	0.598
		Ohsumed	0.809	0.386	0.459
	Ohsumed	Yahoo!	0.829	0.570	0.450
		DMoz	0.878	0.791	0.162
Gerarchica	Yahoo!	DMoz	0.096	0.1	0.098
		Ohsumed	0.07	0.075	0.072
	DMoz	Yahoo!	0.637	0.713	0.673
		Ohsumed	0.375	0.375	0.375
	Ohsumed	Yahoo!	0.427	0.828	0.563
		DMoz	0.198	0.228	0.212

Tabella 5.11: Meta-classificazione con training e test set appartenenti a sorgenti dati diverse.

Si può notare dai risultati ottenuti che, per quanto riguarda la classificazione semantica, l'efficacia del sistema rimane a livelli eccellenti, nonostante la diversità sia dei contenuti delle diverse collezioni, sia dalle tipologie intrinseche delle sorgenti (articoli

di pubblicazioni mediche piuttosto che siti web dai contenuti più disparati).
Al contrario, la classificazione gerarchica, risente di più questa diversità dei contenuti e soprattutto delle diverse specializzazioni delle categorie nei diversi alberi. Infatti, utilizzando un classificatore addestrato sul training-set di *Yahoo!*, la sorgente più piccola e meno specializzata nei vari argomenti, la classificazione negli altri due dataset risulta essere molto poco accurata. Mentre addestrando i classificatori con i training-set di *DMoz* o *Ohsumed*, si riesce ad ottenere una classificazione accettabile su *Yahoo!*, in quanto la fase di learning viene effettuata tramite una sorgente più specifica rispetto a quella di test. Questo principio è dimostrato anche dal risultato dell'applicazione del modello addestrato su *DMoz* e testato su *Ohsumed*.
Confrontando i casi in cui i classificatori utilizzati per la classificazione semantica e quella gerarchica sono valutati sugli stessi test, si evince dalla tabella 5.11 la correlazione dei risultati ottenuti nelle due diverse categorizzazioni. Miglioramenti anche poco significativi nella classificazione semantica portano a miglioramenti molto più consistenti in quella gerarchica.

5.4 Confronto con i related work

La valutazione reale ed oggettiva del sistema creato consiste nel confronto dei risultati ottenuti con quelli reperiti in letteratura.

Per quanto riguarda la classificazione semantica, trattandosi di un metodo innovativo, il confronto tra una coppia documento-categoria, l'unico confronto possibile è dato dalla tesi dell'ing. Magnani [59], il quale, tra i vari test effettuati, nel confronto tra documenti e categorie classificate come simili, dissimili o iponime ha riscontrato come miglior risultato il 78% di accuratezza su *Yahoo!* e il 79% su *DMoz*. Dalla tabella 5.1 si può notare come il sistema creato in questa test ottenga il 94.8% di accuratez-

za su *Yahoo!*, mentre su *DMoz* ottenga l'86.4%, migliorando in maniera evidente l'efficacia della classificazione ottenuta su entrambi i dataset (tabella 5.12).

Sorgente	**Risultati Magnani [59]**	**Risultati tesi**
Yahoo!	0.78	0.948
DMoz	0.79	0.864

Tabella 5.12: Confronto dei risultati per la classificazione semantica tra quelli ottenuti in questa tesi e quelli di Magnani, valori indicanti l'accuratezza.

La classificazione gerarchica è stata confrontata con le pubblicazioni reperite in letteratura, mostrate in tabella 1.7, che utilizzassero nei propri test le sorgenti dati usate in questa tesi. Verrà ora fornito un riepilogo dei confronti con related work per ogni dataset utilizzato (tabella 5.13):

- *Yahoo!*: Questa sorgente dati è stata reperita direttamente dal progetto di Ceci e Malerba [62], corrisponde esattamente al dataset da loro utilizzato. In [8] il risultato migliore ottenuto su questa sorgente è circa un 54% di accuratezza ottenuto con Naïve Bayes, nel nostro sistema si è raggiunto un massimo di 86.5% di accuratezza, mentre con la configurazione standard si ha il 79.6%. Risultati che superano di gran lunga quelli precedenti ottenuti da Ceci e Malerba.

- *DMoz*: Anche per questa sorgente il confronto è stato fatto tramite i risultati ottenuti in [8]. I test di Ceci e Malerba si fermano sempre sotto la soglia del 45% di accuratezza, nel nostro sistema si è ottenuto un massimo del 46%. In questa sorgente i risultati si sono mostrati in linea con quelli reperiti in letteratura, ottenendo un piccolo miglioramento in termini di accuratezza, con il vantaggio però della creazione di un unico classificatore generale.

- *Ohsumed*: Il confronto di questa sorgente è stato effettuato con l'ausilio di diversi articoli che presentassero test e risultati su questo specifico dataset. In [53], Ruiz et al. ottengono un massimo della F_1 media pari circa al 57%, mentre in [56] (Cai et al.) si ha un massimo di $F_1 = 45\%$ media. il risutato migliore ottenuto nel nostro sistema è $F_1 = 54\%$, di poco inferiore ai risultati di Ruiz et al. ma superiore a quelli di Cai et al. Anche in questo caso i risultati sono simili a quelli noti in letteratura, con l'utilizzo però di un unico classificatore.

Sorgente	**Misura**	**Ceci[8]**	**Ruiz[53]**	**Cai[56]**	**Tesi**
Yahoo!	$Accuracy_{best}$	0.54	-	-	0.865
	F_{1best}	-	-	-	0.841
DMoz	$Accuracy_{best}$	0.45	-	-	0.46
	F_{1best}	-	-	-	0.466
Ohsumed	$Accuracy_{best}$	-	-	-	0.54
	F_{1best}	-	0.57	0.45	0.544

Tabella 5.13: Confronto dei risultati per la classificazione semantica tra quelli ottenuti in questa tesi e quelli di Magnani.

Conclusioni

In questa tesi si è sviluppato e implementato un sistema in grado di classificare documenti in una gerarchia di categorie tramite un modello creato sulle relazioni semantiche, un approccio totalmente diverso da quelli più comunemente utilizzati in letteratura che si basano su modelli legati direttamente alle categorie e ai termini dei documenti analizzati.

Il concetto innovativo su cui si basa questo sistema è la rappresentazione di feature basate sulle relazioni semantiche, rilevate attraverso il database semantico-lessicale WordNet, intercorrenti fra coppie di parole e non direttamente dai termini contenuti nei documenti. Il modello così creato risulta essere basato su istanze, ciascuna relativa all'accoppiamento di un documento con una categoria, classificate in base alla relazione semantica all'interno della coppia.

L'algoritmo di classificazione gerarchico è stato composto in varie fasi, la prima delle quali è stata la trasformazione in bag-of-words dei dati e contestualmente la riduzione della dimensionalità. Si è passati inzialmente dalle sorgenti dei dati eterogenee a una rappresentazione standardizzata, estraendo, dopo una fase di pre-elaborazione delle parole, i termini di ogni documento. Il termine di questa fase corrisponde con la selezione dei termini rilevanti per ogni documento e categoria in modo da creare rappresentazioni degli stessi. In questa fase si sono aggiunte metodologie innovative studiate appositamente per la classificazione gerarchica, sia per la fase weighting dei termini, creando

due nuove versioni del comune *tfidf*, sia per la fase di rappresentazione delle categorie, creando una variante bottom-up per la rappresentazione delle categorie.

I training-set utilizzati per l'addestramento dei classificatori sono stati creati attraverso l'accoppiamento di ogni documento con varie categorie, avendo a disposizione tre diverse tipologie di relazioni tra essi (simile, dissimile o iponimo). Gli attributi di ogni istanza sono quindi stati creati attraverso la ricerca di relazioni semantiche in WordNet tra tutte le possibili coppie di termini appartenenti alle rappresentazioni del documento e della categoria relativi all'istanza stessa.

La classificazione gerarchica è stata effettuata attraverso un'algoritmo iterativo top-down: partendo dalla categoria radice si scende all'interno dell'albero confrontando la similarità e l'iponimia delle categorie figlie del nodo corrente con il documento da classificare. Ad ogni livello viene valutata la scelta di terminare l'algoritmo nel nodo corrente, assegnando il documento ad esso, oppure di proseguire con una nuova iterazione scendendo di livello, selezionando il figlio più simile al documento.

Per la valutazione del modello creato, il sistema è stato reso quanto più possibile parametrico, in modo da valutare e confrontare i test attraverso diverse configurazioni del classificatore. Il sistema è stato testato su tre diverse sorgenti di dati: *Yahoo!*, *DMoz* e *Ohsumed*. Il processo di classificazione si è dimostrato particolarmente efficace su *Yahoo!*, superando di gran lunga i risultati reperiti in letteratura, mentre sulle altre due collezioni di dati si è giunti a risultati in linea con lo stato dell'arte, con il vantaggio però dell'utilizzo di un classificatore unico e generale.

Il punto di forza di questo metodo innovativo consiste nel fatto che il classificatore così creato è indipendente dalle istanze delle categorie con cui è addestrato, si crea quindi un meta-classificatore capace di predire l'appartenenza di documenti non noti in categorie non note. Questo risultato è stato dimostrato

attraverso i test svolti nella tesi, confermando la capacità del sistema di classificare documenti in categorie non note, sia facenti parte della stessa collezione del training-set, sia appartenente a sorgenti completamente diverse da quelle utilizzate nella fase di learning.

Come si nota anche dal consistente numero di parametri, l'implementazione del metodo generale proposto in questa tesi può essere alterata in molte sue parti. La revisione e il miglioramento delle diverse parti del processo potrebbero essere oggetto di sviluppi futuri.
Si potrebbero ad esempio studiare nuovi metodi per il weighting dei termini e nuove modalità di confronto semantico tra questi, con l'eventuale possibilità di migliorare l'efficacia e pure l'efficienza del sistema. Inoltre, si potrebbe intervenire sull'algoritmo di classificazione gerarchica, per provare a correggere i limiti che per ora ha presentato ad esempio nella selezione della categoria nei nodi di livello alto e con molte ramificazioni.

Appendice A

Sorgenti dati

A.1 Yahoo

Questa sezione descrive la struttura gerarchica dei dati Yahoo, inoltre in tabella A.1 viene mostrata la distribuzione dei documenti e delle categorie a seconda del livello all'interno dell'albero tassionomico.

0.1-Science
0.1.1-Chemistry
0.1.1.1-Education
0.1.1.2-Periodic Table of the Elements
0.1.1.3-Organic Chemistry
0.1.1.4-Spectroscopy
0.1.1.5-Polymers
0.1.1.5.1-Institutes
0.1.1.6-Biochemistry
0.1.1.6.1-Institutes
0.1.2-Alternative
0.1.2.1-Skeptics
0.1.2.2-Parapsychology
0.1.2.3-New Physics
0.1.2.4-Paranormal Phenomena
0.1.2.4.1-Extraterrestrial Life
0.1.2.4.2-Organizations
0.1.2.4.3-Cryptozoology
0.1.2.4.4-Ghosts
0.1.2.4.5-Magazines
0.1.2.4.6-Crop Circles
0.1.2.4.7-Near Death Experiences
0.1.2.5-Earth Changes
0.1.2.5.1-Pole Shift
0.1.3-Mathematics
0.1.3.1-Education
0.1.3.1.1-Teaching

0.1.3.2-Geometry
0.1.3.3-Problems, Puzzles, and Games
0.1.3.4-Statistics
0.1.4-Agriculture
0.1.4.1-Agricultural Pollution
0.1.4.2-Forestry
0.1.4.2.1-Organizations
0.1.4.3-Sustainable Agriculture
0.1.4.3.1-Permaculture
0.1.4.3.2-Organizations
0.1.4.4-Animal Science
0.1.4.4.1-Manure Management
0.1.4.4.2-Poultry
0.1.4.4.3-Sheep
0.1.4.4.4-College and University Departments and Programs
0.1.4.4.5-Cattle
0.1.5-Biology
0.1.5.1-Biodiversity
0.1.5.2-Biotechnology
0.1.5.2.1-Institutes
0.1.5.3-Botany
0.1.5.3.1-Phycology
0.1.5.3.2-Botanical Gardens
0.1.5.3.3-Herbaria
0.1.5.4-Anatomy
0.1.5.4.1-Cardiovascular System
0.1.6-Earth Sciences
0.1.6.1-Atmosphere
0.1.6.1.1-Ionosphere
0.1.6.1.2-Magnetosphere
0.1.6.1.3-Aeronomy
0.1.6.2-Oceanography
0.1.6.2.1-Artificial Reefs
0.1.6.2.2-Mid-Ocean Ridges
0.1.6.3-Geology and Geophysics
0.1.6.3.1-Organizations
0.1.6.3.2-Volcanology
0.1.6.4-Meteorology
0.1.6.4.1-Maps and Observations
0.1.6.4.2-Cryosphere
0.1.6.4.3-Climate Centers
0.1.6.4.4-Research

Livello	**Numero categorie**	**Numero Documenti**
1	1	0
2	6	98
3	27	349
4	35	454

Tabella A.1: Distribuzione categorie e documenti nel data-set *Yahoo!*.

A.2 DMoz

Viene ora elencata la tassionomia dei dei dati *DMoz*, in tabella A.2 viene mostrata la distribuzione dei documenti e delle categorie a seconda del livello all'interno dell'albero tassionomico.

0.1-Conditions and Diseases
0.1.1-Ear, Nose and Throat
0.1.1.1-Tinnitus
0.1.2-Allergies
0.1.2.1-Latex
0.1.2.2-Food
0.1.3-Genitourinary Disorders
0.1.3.1-Bladder
0.1.3.1.1-Incontinence
0.1.3.2-Kidney
0.1.3.2.1-Stones
0.1.3.2.2-End Stage Disease
0.1.4-Food and Water Borne
0.1.4.1-Toxins
0.1.5-Infectious Diseases
0.1.5.1-Mycobacterial
0.1.5.1.1-Tuberculosis
0.1.5.2-Zoonoses
0.1.5.3-Bacterial
0.1.5.3.1-E. coli
0.1.5.4-Viral
0.1.5.4.1-Influenza
0.1.5.4.2-Herpes
0.1.5.4.2.1-Herpes Zoster
0.1.5.4.3-West Nile Virus
0.1.5.5-Parasitic
0.1.5.5.1-Malaria
0.1.5.5.2-Nematodes
0.1.6-Sleep Disorders
0.1.6.1-Centers
0.1.6.2-Insomnia
0.1.6.3-Sleep Apnea
0.1.6.4-Narcolepsy
0.1.7-Musculoskeletal Disorders
0.1.7.1-Osteoporosis
0.1.7.2-Connective Tissue
0.1.7.2.1-Fibromyalgia
0.1.7.2.1.1-Support Groups
0.1.7.2.2-Marfan Syndrome
0.1.7.2.3-Lupus
0.1.7.2.3.1-Personal Pages
0.1.7.3-Arthritis
0.1.7.3.1-Rheumatoid
0.1.7.4-Back and Spine
0.1.7.5-Congenital Anomalies
0.1.7.5.1-Arthrogryposis
0.1.7.5.1.1-Personal Pages
0.1.7.6-Repetitive Strain Injuries
0.1.8-Endocrine Disorders
0.1.8.1-Thyroid

0.1.8.1.1-Hyperthyroid
0.1.8.1.1.1-Graves' Disease
0.1.8.2-Pancreas
0.1.8.2.1-Diabetes
0.1.8.2.1.1-Nutrition
0.1.8.2.1.2-Support Groups
0.1.8.2.1.3-Organizations
0.1.8.2.2-Hypoglycemia
0.1.8.3-Gonads
0.1.8.3.1-Polycystic Ovarian Syndrome
0.1.9-Blood Disorders
0.1.9.1-Anemia
0.1.10-Communication Disorders
0.1.10.1-Language and Speech
0.1.10.1.1-Aphasia
0.1.10.1.2-Spasmodic Dysphonia
0.1.10.2-Hearing
0.1.10.2.1-Deafness
0.1.10.2.1.1-Organizations
0.1.11-Cardiovascular Disorders
0.1.11.1-Heart Disease
0.1.11.1.1-Murmurs
0.1.11.1.2-Arrhythmia
0.1.11.1.3-Support Groups
0.1.11.1.4-Angina Pectoris
0.1.11.1.5-Heart Attack
0.1.11.1.6-Organizations
0.1.11.1.7-Resources
0.1.11.1.8-Valvular
0.1.11.1.8.1-Mitral Valve
0.1.11.1.9-Cardiomyopathy
0.1.11.1.10-Congestive Failure
0.1.11.1.11-Cholesterol
0.1.11.1.12-Congenital
0.1.11.2-Vascular Disorders
0.1.11.2.1-Thrombosis
0.1.11.2.2-Lymphedema
0.1.11.2.3-Varicose Veins
0.1.11.2.3.1-Treatment
0.1.11.2.4-Hypertension
0.1.11.2.5-Aneurysm
0.1.12-Neurological Disorders
0.1.12.1-Movement Disorders
0.1.12.1.1-Torticollis
0.1.12.2-Alzheimer's
0.1.12.2.1-Care Givers
0.1.12.2.2-Associations
0.1.12.3-Cranial Nerve Diseases
0.1.12.3.1-Bell's Palsy
0.1.12.4-Muscle Diseases
0.1.12.4.1-Reflex Sympathetic Dystrophy
0.1.12.5-Trauma and Injuries
0.1.12.5.1-Brain Injury
0.1.12.5.1.1-Treatment and Residential Programs
0.1.12.5.1.2-Organizations
0.1.12.5.1.3-Personal Pages
0.1.12.5.2-Spinal Cord Injury
0.1.12.5.2.1-Research
0.1.12.5.2.2-Personal Pages
0.1.12.6-Infections

0.1.17.2.1.2-Regional
0.1.17.2.1.2.1-Africa
0.1.17.2.1.3-Support Groups
0.1.17.2.1.4-Treatment and Therapies
0.1.17.2.1.5-Organizations
0.1.17.3-Auto-Immune
0.1.17.3.1-Sarcoidosis
0.1.18-Digestive Disorders
0.1.18.1-Intestinal
0.1.18.1.1-Irritable Bowel Syndrome
0.1.18.1.2-Diarrhea
0.1.18.1.3-Inflammatory Bowel Disease
0.1.18.1.3.1-Crohn's Disease
0.1.18.2-Anorectal
0.1.18.2.1-Hemorrhoids
0.1.18.3-Esophagus
0.1.18.3.1-Reflux Disease
0.1.18.4-Liver
0.1.18.4.1-Hepatitis
0.1.18.4.1.1-Hepatitis C
0.1.19-Cancer
0.1.19.1-Eye
0.1.19.1.1-Retinoblastoma
0.1.19.2-Gastrointestinal
0.1.19.2.1-Colorectal
0.1.19.2.2-Esophageal
0.1.19.3-Genitourinary
0.1.19.3.1-Prostate
0.1.19.3.1.1-Support Groups
0.1.19.4-Breast
0.1.19.4.1-Events
0.1.19.4.2-Organizations
0.1.19.4.2.1-Susan G. Komen Breast Cancer Foundation
0.1.19.4.3-Personal Pages
0.1.19.5-Journals
0.1.19.6-Centers
0.1.19.6.1-Breast
0.1.19.6.2-University
0.1.19.7-Melanoma
0.1.19.8-Organizations
0.1.19.9-Lung
0.1.19.10-Hematologic
0.1.19.10.1-Leukemia
0.1.19.10.1.1-Support Groups
0.1.19.11-Gynecologic
0.1.19.11.1-Ovarian
0.1.19.12-Skin
0.1.20-Respiratory Disorders
0.1.20.1-SARS
0.1.20.1.1-News and Media
0.1.20.2-Asthma
0.1.20.3-Chronic Obstructive Pulmonary Disease
0.1.20.3.1-Emphysema
0.1.20.4-Rhinitis
0.1.21-Wounds and Injuries
0.1.21.1-Burns
0.1.21.2-Hypothermia
0.1.21.3-Altitude Sickness

Livello	Numero categorie	Numero Documenti
1	1	0
2	21	350
3	81	1563
4	85	2703
5	32	1163
6	2	57

Tabella A.2: Distribuzione categorie e documenti nel data-set *DMoz*.

A.3 Ohsumed

Questa sezione descrive la struttura gerarchica dei dati *Ohsumed*, inoltre in tabella A.3 viene mostrata la distribuzione dei documenti e delle categorie a seconda del livello all'interno dell'albero tassionomico.

0.1-Heart Disease
0.1.1-Heart Valve Diseases
0.1.1.1-Heart Murmurs
0.1.1.2-Tricuspid Valve Stenosis
0.1.1.3-Pulmonary Valve Insufficiency
0.1.1.4-Tricuspid Valve Insufficiency
0.1.1.5-Mitral Valve Prolapse
0.1.1.6-Aortic Valve Stenosis
0.1.1.6.1-Cardiomyopathy, Hypertrophic
0.1.1.7-Tricuspid Valve Prolapse
0.1.1.8-Mitral Valve Stenosis
0.1.1.9-Aortic Valve Insufficiency
0.1.1.10-Mitral Valve Insufficiency
0.1.1.11-Pulmonary Valve Stenosis
0.1.1.11.1-Pulmonary Subvalvular Stenosis
0.1.1.12-Aortic Valve Prolapse
0.1.2-Pulmonary Heart Disease
0.1.3-Heart Defects, Congenital
0.1.3.1-Transposition of Great Vessels
0.1.3.1.1-Double Outlet Right Ventricle
0.1.3.2-Dextrocardia
0.1.3.3-Ductus Arteriosus,

Patent
0.1.3.4-Aortic Coarctation
0.1.3.5-Heart Septal Defects
0.1.3.5.1-Endocardial Cushion Defects
0.1.3.5.2-Aortopulmonary Septal Defect
0.1.3.5.3-Heart Septal Defects, Ventricular
0.1.3.5.4-Heart Septal Defects, Atrial
0.1.3.6-Cor Triatriatum
0.1.3.7-Truncus Arteriosus, Persistent
0.1.3.8-Levocardia
0.1.3.9-Crisscross Heart
0.1.3.10-Marfan Syndrome
0.1.3.11-Tetralogy of Fallot
0.1.3.12-Coronary Vessel Anomalies
0.1.3.13-Eisenmenger Complex
0.1.3.14-Ebstein's Anomaly
0.1.4-Cardiac Output, Low
0.1.5-Rheumatic Heart Disease
0.1.6-Arrhythmia
0.1.6.1-Sick Sinus Syndrome
0.1.6.2-Heart Block
0.1.6.2.1-Bundle-Branch Block
0.1.6.2.2-Adams-Stokes Syndrome
0.1.6.2.3-Sinoatrial Block
0.1.6.3-Atrial Fibrillation
0.1.6.4-Bradycardia
0.1.6.5-Atrial Flutter
0.1.6.6-Tachycardia
0.1.6.6.1-Tachycardia, Paroxysmal
0.1.6.6.2-Tachycardia, Supraventricular
0.1.6.6.2.1-Tachycardia, Atrioventricular Nodal Reentry
0.1.6.6.2.2-Tachycardia, Ectopic Junctional
0.1.6.6.2.3-Tachycardia, Sinoatrial Nodal Reentry
0.1.6.6.2.4-Tachycardia, Ectopic Atrial
0.1.6.6.2.5-Tachycardia, Sinus
0.1.6.7-Pre-Excitation Syndromes
0.1.6.7.1-Wolff-Parkinson-White Syndrome
0.1.6.7.2-Pre-Excitation, Mahaim-Type
0.1.6.8-Arrhythmia, Sinus
0.1.6.9-Long QT Syndrome
0.1.6.10-Ventricular Fibrillation
0.1.7-Myocardial Infarction
0.1.7.1-Shock, Cardiogenic
0.1.8-Heart Neoplasms
0.1.9-Coronary Disease
0.1.9.1-Coronary Thrombosis
0.1.9.2-Coronary Arteriosclerosis
0.1.9.3-Coronary Vasospasm

0.1.9.4-Coronary Aneurysm
0.1.9.5-Angina Pectoris
0.1.9.5.1-Angina, Unstable
0.1.9.5.1.1-Angina Pectoris, Variant
0.1.10-Pericarditis
0.1.10.1-Pericarditis, Tuberculous
0.1.10.2-Pericarditis, Constrictive
0.1.11-Postpericardiotomy Syndrome
0.1.12-Cardiomyopathy, Congestive
0.1.13-Myocardial Diseases
0.1.13.1-Myocardial Reperfusion Injury
0.1.13.2-Kearns Syndrome
0.1.13.3-Endocardial Fibroelastosis
0.1.13.4-Endomyocardial Fibrosis
0.1.13.5-Myocarditis
0.1.13.6-Cardiomyopathy, Alcoholic
0.1.13.7-Chagas Cardiomyopathy
0.1.13.8-Cardiomyopathy, Restrictive
0.1.14-Cardiac Tamponade
0.1.15-Heart Failure, Congestive
0.1.15.1-Dyspnea, Paroxysmal
0.1.15.2-Edema, Cardiac
0.1.16-Heart Aneurysm
0.1.17-Heart Arrest
0.1.18-Ventricular Outflow Obstruction
0.1.19-Endocarditis
0.1.19.1-Endocarditis, Bacterial
0.1.19.1.1-Endocarditis, Subacute Bacterial
0.1.20-Heart Rupture
0.1.20.1-Heart Rupture, Post-Infarction
0.1.21-Carcinoid Heart Disease
0.1.22-Pneumopericardium
0.1.23-Pericardial Effusion

Livello	**Numero categorie**	**Numero Documenti**
1	1	33
2	23	555
3	56	338
4	16	64
5	6	10
6	2	57

Tabella A.3: Distribuzione categorie e documenti nel data-set *Ohsumed*.

Bibliografia

[1] C. Aggarwal, C. Zhai. *Mining text data*, Kluwer Academic Publishers.

[2] G. Salton. *An Introduction to Modern Information Retrieval*, McGraw Hill, 1983.

[3] R. Agrawal, R. Srikant. *Fast Algorithms for Mining Association Rules in Large Databases*, VLDB Conference, 1994.

[4] H. Schutze, C. Silverstein. *Projections for Efficient Document Clustering*, ACM SIGIR Conference, 1997.

[5] C. C. Aggarwal, Y. Zhao, P. S. Yu. *On Text Clustering with Side Information*, ICDE Conference, 2012.

[6] D. Mladenic, M. Grobelnik, *Feature selection on hierarchy of web documents*. Decision Support Systems, 35(1), 45-87, 2003.

[7] J. R. Quinlan, *Induction of Decision Trees, Machine Learning*, 1(1), pp 81-106, 1986.

[8] M. Ceci, D. Malerba. *Classifying web documents in a hierarchy of categories: a comprehensive study*. Journal of Intelligent Information Systems, page 37-78, 2007.

[9] M. Dash, H. Liu. *Feature Selection for Clustering*, PAKDD Conference, pp. 110-121, 1997.

[10] S. E. Robertson, S. Walker. *Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval*. In SIGIR, pages 232-241, 1994.

[11] N. Slonim, N. Tishby. *Document Clustering using word clusters via the information bottleneck method*, ACM SIGIR Conference, 2000.

[12] Slava M. Katz. *Distribution of content words and phrases in text and language modelling*. Nat. Lang. Eng., 2(1):15-59, 1996.

[13] S. Deerwester, S. Dumais, T. Landauer, G. Furnas, R. Harshman. *Indexing by Latent Semantic Analysis*, JASIS, pp. 391-407, 1990.

[14] N. Sahoo, J. Callan, R. Krishnan, G. Duncan, R. Padman. *Incremental Hierarchical Clustering of Text Documents*, ACM CIKM Conference, 2006.

[15] G.-R. Xue, D. Xing, Q. Yang, Y. Yu. *Deep classification in large-scale text hierarchies*. ACM SIGIR Conference, 2008.

[16] Y. Yang, C.G. Chute. *An example-based mapping method for text categorization and retrieval*. ACM Transactions on Information Systems, 12(3), 1994.

[17] S. Dumais, H. Chen. *Hierarchical Classification of Web Content*. ACM SIGIR Conference, 2000.

[18]] H. Schutze, D. Hull, J. Pedersen. *A comparison of classifiers and document representations for the routing problem*. ACM SIGIR Conference, 1995.

[19] Y. Li, A. Jain. *Classification of text documents*. The Computer Journal, 41(8), pp. 537-546, 1998.

[20] Y. Freund, R. Schapire. *A decision-theoretic generalization of on-line learning and an application to boosting*. In Proc. Second European Conference on Computational Learning Theory, pp. 23-37, 1995.

[21] F. Sebastiani. *Machine learning in automated text categorization*. In Consiglio Nazionale delle Ricerche, Italy, 2001.

[22] R. Fisher. *The Use of Multiple Measurements in Taxonomic Problems*. Annals of Eugenics, 7, pp. 179-188, 1936.

[23] I. T. Jolliffee. *Principal Component Analysis*. Springer, 2002.

[24] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. Della Pietra, J/ C. Lai. *Class-based n-gram models of natural language*, Computational Linguistics, 18, 4 (December 1992), 467-479.

[25] T. Hofmann. *Probabilistic Latent Semantic Indexing*, ACM SIGIR Conference, 1999.

[26] I. Dhillon. *Co-clustering Documents and Words using bipartite spectral graph partitioning*, ACM KDD Conference, 2001.

[27] I. Dhillon, S. Mallela, D. Modha. *Information-theoretic Co-Clustering*, ACM KDD Conference, 2003.

[28]] R. Angelova, S. Siersdorfer. *A neighborhood-based approach for clustering of linked document collections*. CIKM Conference, 2006.

[29] D. Cutting, D. Karger, J. Pedersen, J. Tukey. *Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections*. ACM SIGIR Conference, 1992.

[30] D. Cutting, D. Karger, J. Pederson. *Constant Interaction-time Scatter/Gather Browsing of Large Document Collections*, ACM SIGIR Conference, 1993.

[31] E. M. Voorhees. *Implementing Agglomerative Hierarchical Clustering for use in Information Retrieval*,Technical Report TR86-765, Cornell University, Ithaca, NY, July 1986.

[32]] Y. Yang, J. O. Pederson. *A comparative study on feature selection in text categorization*, ACM SIGIR Conference, 1995.

[33] W. Cohen, Y. Singer. *Context-sensitive learning methods for text categorization*. ACM Transactions on Information Systems, 17(2), pp. 141-173, 1999.

[34] P. Bennett, N. Nguyen. *Refined experts: improving classification in large taxonomies*. ACM SIGIR Conference, 2009.

[35] S. Chakrabarti, B. Dom. R. Agrawal, P. Raghavan. *Using taxonomy, discriminants and signatures for navigating in text databases*, VLDB Conference, 1997.

[36] W. Cohen, H. Hirsh. *Joins that generalize: text classification using Whirl*. ACM KDD Conference, 1998.

[37] W. Hersh, C. Buckley, T. J. Leone, and D. Hickam. *Ohsumed: an interactive retrieval evaluation and new large test collection for research*. In Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval, pages 192-201. Springer-Verlag New York, Inc., 1994.

[38] Y. Yang. *An evaluation of statistical approaches to text categorization*. Information Retrieval, 1(1-2), 69-90, 1999.

[39] N. Cesa-Bianchi, C Gentile, L Zaniboni. *Hierarchical classification: combining Bayes with SVM.* In: Proc. of the 23rd Int. Conf. on Machine learning, pp 177-184, 2006.

[40] F. Sebastiani. *Machine learning in automated text categorization.* In Consiglio Nazionale delle Ricerche, Italy, 2001.

[41] J. Platt, *Fast training of support vector machines using sequential minimal optimization.* Advances in Kernel Methods-Support Vector Learning. B. Schölkopf, C. Burges, and A. Smola, eds., MIT Press, 1999.

[42]] D. Koller, M. Sahami. *Hierarchically classifying documents with very few words*, ICML Conference, 2007.

[43] R. E. Schapire and Y. Singer. *Improved boosting algorithms using confidence-rated predictions.* Machine Learning, 37(3):297-336, 1999.

[44] C. Zhai. *Statistical Language Models for Information Retrieval*, Morgan and Claypool Publishers, 2008.

[45] W. Xu, Y. Gong. *Document clustering by concept factorization*, ACM SIGIR Conference, 2004.

[46] W. K. Estes. *Classification and Cognition.* Oxford University Press, New York: NY, 1994.

[47] W. Ke, C. Sugimoto, J. Mostafa. *Dynamicity vs. effectiveness: studying online clustering for scatter/gather.* ACM SIGIR Conference, 2009.

[48] C. Apte, F. Damerau, S. Weiss. *Automated Learning of Decision Rules for Text Categorization*, ACM Transactions on Information Systems, 12(3), pp. 233-251, 1994.

[49] J. Kleinberg, *Bursty and hierarchical structure in streams*, ACM KDD Conference, pp. 91-101, 2002.

[50] A. McCallum, R. Rosenfeld, T. M. Mitchell, A. Y. Ng. *Improving text classification by shrinkage in a hierarchy of classes*. In Proceedings of the Fifteenth International Conference on Machine Learning (pp. 359-367), San Mateo, California, 1998.

[51] S. D'Alessio, K. Murray, R. Schiaffino, A. Kershenbau. *The effect of using hierarchical classifiers in text categorization*. In Proc. of the 6th Int. Conf. on "Recherche d'Information Assistée par Ordinateur"(RIAO), (pp. 302-313). Paris, France, 2000.

[52] H. T. Ng, W. B. Goh, K. L. Low. *Feature selection, perception learning, and a usability case study for text categorization*. SIGIR Forum, 31(SI), 67-73, 1997.

[53] M. E. Ruiz, P. Srinivasan. *Hierarchical text categorization using neural networks*. Information Retrieval, 5(1), 87-118, 2002.

[54] A. S. Weigend, E. D. Wiener, J. O. Pedersen. *Exploiting hierarchy in text categorization*. Information Retrieval, 1(3), 193-216, 1999.

[55] N. Cesa-Bianchi, C. Gentile, A. Tironi, L. Zaniboni. *Incremental algorithms for hierarchical classification*. Advances in Neural Information Processing Systems 17 (pp. 233-240). MIT Press, 2005.

[56] L. Cai, T. Hofmann. *Text categorization by boosting automatically extracted concepts*. ACM SIGIR Conference, 2003.

[57] M. Maron. *Automatic indexing: an experimental inquiry.* Journal of the Association for Computing Machinery, 8(3):404-417, 1961.

[58] J. Wilbur, K. Sirotkin. *The automatic identification of stopwords*,J. Inf. Sci., 1992.

[59] F. Magnani. *Sviluppo di tecniche di Text Mining per la classificazione semantica di documenti*, 2012.

[60] Machine Learning Group at University of Waikato. *Weka 3: Data Mining Software in Java*, URL: http://www.cs.waikato.ac.nz/ml/weka/.

[61] G. Miller. *Dmoz, Open Directory Project*, URL: http://www.dmoz.org.

[62] M. Ceci and D. Malerba. *WebClassIII*, URL: http://www.di.uniba.it/~malerba/software/webclass.

[63] Princeton University. *WordNet, a lexical database for English*, URL: http://wordnet.princeton.edu/.

[64] Princeton University. *JWI, the MIT Java Wordnet Interface*, URL: http://projects.csail.mit.edu/jwi/.

[65] The Eclipse Foundation. *Eclipse*, URL: http://www.eclipse.org/.

Ringraziamenti

Un primo doveroso ringraziamento va al mio relatore, il *Professor Moro*, per avermi dato la possibilità di sviluppare questa tesi. Insieme al professore vanno i miei più sentiti ringraziamenti all'*ing. Pasolini* per il tempo dedicatomi e per l'aiuto nello svolgimento della tesi.

Vorrei poi ringraziare in modo particolare la mia famiglia, che ha sempre costituito per me un punto di riferimento ed una costante fonte di supporto.

Un grazie sentito anche agli amici ed a tutti coloro che mi hanno sempre aiutato e supportato.
In particolare ringrazio i ragazzi del TSR con cui condivido la mia vita da anni. I Several Union, con cui esprimo l'altra mia grande passione, la musica. Infine quei compagni di facoltà, che mi hanno aiutato in questi anni universitari.