ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA
SECONDA FACOLTÀ DI INGEGNERIA CON SEDE DI CESENA
CORSO DI LAUREA MAGISTRALE IN INGEGNERIA
INFORMATICA

Sviluppo di tecniche di Text Mining per la classificazione semantica di documenti

Tesi in:

Tecnologie e Sistemi per il Data Mining

Relatore:
Prof. Gianluca Moro
Correlatore:
Ing. Roberto Pasolini

Presentata da:
Fabio Magnani

Sessione: Terza
Anno Accademico: 2010/2011

Indice

Introduzione

Si può affermare con assoluta certezza che tra le grandi rivoluzioni nel mondo informatico vi è Internet. Inizialmente sia le persone che vi accedevano sia le pagine contenute all'interno della rete erano un numero limitato, adesso con i ripetuti abbassamenti dei costi dei calcolatori, delle connessioni internet e grazie al protocollo *http* (HyperText Transfer Protocol) definito da Tim Berners-Lee nel 1991 si è reso possibile far consultare a milioni di persone una grandissima quantità di documenti.

Gli utenti sono in parte aiutati a destreggiarsi con una tale mole di dati dai motori di ricerca. Tuttavia il fattore umano è ancora indispensabile, poiché i motori di ricerca sono soltanto in grado di indicare la strada e non di portare direttamente a destinazione. Infatti, l'incremento esponenziale lo ha avuto il numero di documenti presenti sul web, mentre la capacità di lettura e di analisi dell'utente è rimasta sostanzialmente invariata nel tempo.

La conseguenza è l'introduzione del termine *Information Overload* (sovraccarico cognitivo) e con essa si vuole rappresentare la difficoltà nel recuperare informazioni desiderate a causa della immensa quantità di contenuti da elaborare. Questo perché le informazioni presenti sul Web, sono rappresentate in maniera comprensibile dagli umani, ma non dalle macchine. Quindi si ottiene che una buona parte dei documenti presenti nel World Wide Web è sottoforma di testo libero scritto in linguaggio naturale non classificato, né strutturato, per il quale si è obbligati ad una categorizzazione automatica per rendere più rapida ed efficace la ricerca delle informazioni.

L'obiettivo di questa tesi è lo sviluppo di tecniche di Text Mining per la classificazione semantica di documenti non strutturati in categorie organizzate in modo gerarchico (e.g., DMOZ, Yahoo). La tecnica usata indaga, attraverso un dizionario (WordNet), sulle relazioni semantiche tra le parole presenti all'interno dei documenti mentre il processo di classificazione fino ad ora usato dai ricercatori è identificato come approccio *bag of words* per rappresentare i singoli documenti testuali.

La sostanziale differenza sta nel fatto di considerare coppie di parole che possono essere omogenee o eterogenee invece del singolo elemento *(bag of words).* Viene abbandonato l'uso dei termini come *features* per descrivere i contenuti dei documenti e vengono introdotti nuovi attributi definiti dalla combinazione tra dati ottenuti dal dizionario WordNet e quelli presenti nella collezione. Questa innovazione garantisce l'indipendenza dal contesto nella classificazione, cosa che negli approcci precedenti non è presente.

Riassumendo, le tecnologie/strumenti utilizzati sono:

1. Text Mining: tecnologia linguistico/matematica per lo studio di una enorme quantità di testi che non sono trattabili con le tecniche attuali se non a costi insostenibili. La sua particolarità sta nel fatto di essere in grado di estrarre informazioni ignote, nascoste. Si compie un'estrazione non banale d'informazioni implicite, precedentemente sconosciute e potenzialmente utili dei dati.

2. DMOZ: Open Directory Project (noto come DMOZ): è la più grande e la più esauriente directory del Web curata da uomini. E' costruita e mantenuta da una vasta comunità globale di editori volontari ed è caratterizzata dall'organizzazione delle categorie in una struttura gerarchica e rappresenta la fonte dei documenti e delle categorie utilizzati in questa tesi [19].

3. WordNet: database lessicale e semantico per la lingua inglese elaborato dal linguista George Miller, che si propone di organizzare,

definire e descrivere i concetti espressi dai vocaboli. Nomi, verbi, aggettivi e avverbi sono raggruppati in gruppi con significato affine (synsets), ciascun gruppo esprime un concetto distinto. I synsets sono interconnessi mediante la semantica concettuale e relazioni lessicali [8].

Come già affermato precedentemente, il progetto si basa sul confronto tra documenti quindi nella prima fase si imposta un procedimento il quale tenta di rappresentare una categoria attraverso un documento prototipo in modo tale che successivamente si possa applicare il processo di Text Mining seguendo le relazioni presenti tra documenti e documenti prototipi. In quest'ultima fase un fattore molto importante per la classificazione è la scelta sul criterio da applicare per determinare la generazione delle *features*. In studi precedenti si è già affermato che il criterio migliore è rappresentato dal fattore *tfidf* (i.e., Term Frequency Inverse Document Frequency) il quale si basa sulla rilevanza di un preciso termine rispetto all'insieme dei documenti implicati nell'analisi.

L'organizzazione del lavoro svolto ha la seguente struttura dei capitoli:

- Nel *primo capitolo* si descrive un quadro generale sulle tecniche di Text Mining e i rapporti con il Data Mining;
- Il *secondo capitolo* espone DMOZ, il più famoso strumento di fonte dei documenti già classificati in una gerarchia realizzata da umani;
- Nel *terzo capitolo* viene presentato lo strumento WordNet e le sue principali caratteristiche, soffermandosi maggiormente sulle relazioni utilizzate per definire il rapporto che esiste fra due concetti;
- Nel *quarto capitolo* viene illustrato il software Weka, descrivendone l'utilizzo e le funzionalità che mette a disposizione;

- Il *quinto capitolo* contiene un'analisi degli strumenti/tecnologie presenti nello stato dell'arte, sulla classificazione di documenti secondo una struttura gerarchia di categorie;
- Il *sesto capitolo* presenta i nuovi metodi creati per la classificazione semantica stabilita attraverso le relazioni tra coppie di documenti e categorie di documenti.

Capitolo 1

Text Mining

Il concetto che sta alla base del Text Mining è *"ottenere piccole pepite delle informazioni desiderate su montagne di dati testuali senza dover leggere tutto"*, quindi il concetto fondamentale è quello di riuscire ad estrarre pochissime informazioni rilevanti dall'immensa quantità di dati a disposizione senza doverli elaborare tutti.

In questo capitolo si fornisce un quadro generale delle potenzialità del Text Mining e si descrivono gli aspetti più importanti.

1.1 Inizi del Text Mining

I primi segnali furono dati da Hans Peter Luhn (1896 – 1964) che in un suo famoso studio osservò che: *"il potere risolutivo sta nelle parole significative"* [17]. Egli, infatti, definì che l'informazione statistica deriva dalla frequenza e dalla distribuzione delle parole e la utilizzò per calcolare una misura relativa del "significato", prima per singole parole e poi per frasi. Le frasi con punteggio più alto nel "significato" erano estratte per far parte di quella che lui chiamò l'*"auto-abstract"*. Luhn è stato il primo, o tra i primi, ad elaborare molte delle tecniche di base che sono ormai comuni in scienza dell'informazione. Tali tecniche comprendevano *"full-text processing"*, codici hash, auto-indicizzazione, astrazione automatica e il concetto di diffusione selettiva d'informazioni (SDI).

In seguito nel 1961 Lauren B. Doyle [6], denotò il concetto del text mining quando dichiarò che *"l'organizzazione e la caratterizzazione dell'informazione può discendere dall'analisi delle frequenze e della distribuzione delle parole in una collezione di documenti."*.

Nel 1988 Don R. Swanson [29], ha invitato gli scienziati ad essere più come analisti di *intelligence*, di prendere sul serio l'idea che la nuova conoscenza deve essere acquisita dalle collezioni di documenti.

Lui disse:

"[...] New knowledge, or finished intelligence, is seen as emerging from large numbers of individually unimportant but carefully hoarded fragments that were not necessarily recognized as related to one another at the time they were acquired. Use of stored data is intensively interactive; "information retrieval" is an inadequate and even misleading metaphor. The analyst is continually interacting with units of stored data as though they were pieces selected from a thousand scrambled jigsaw puzzles. Relevant patterns, not relevant documents, are sought. [...]"

Definizione che esprime che la nuova conoscenza deriva da un numero elevato di frammenti ammassati e poco rilevanti se considerati singolarmente e non obbligatoriamente riconosciuti come legati tra loro nel momento in cui sono stati appresi. Per l'analista l'uso di collezioni di dati rappresenta il mezzo con cui interagire per individuare pattern piuttosto che documenti importanti.

Swanson in un sistema aveva già messo in pratica questa idea di sviluppo per scoprire nuove conoscenze significative nella letteratura biomedica (vedi riferimenti di Swanson & Smalheiser, 1999 [30]).

Il software, ora chiamato ARROWSMITH [28], aiuta a rilevare interessanti co-occorrenze grazie a comuni *keywords* e frasi in *"complementary and noninteractive"* (complementari e non-interattive) serie di articoli o in *"letterature"* (letterature). Due letterature sono *complementari* se insieme possono rivelare informazioni utili non evidenti nei due insiemi considerati separatamente. Due letterature sono *non-*

interattive se i loro articoli non sono citati tra loro e non sono co-citate altrove nella letteratura.

Sempre nel 1999 sono state usate importanti definizioni (metafore) per il Text Mining tra le quali: *"il mining implica l'estrazione di preziose pepite di minerali dalla roccia senza valore"* (Hearst, 1999 [13]) oppure *"l'oro nascosto in ... montagne di dati testuali"* (Dorre, Gerstl & Seiffert, 1999 [5]).

Hearst non si limitò a tentare di spiegare cos'è il Text Mining ma espose una serie di distinzioni che vi sono a confronto con *"information retrivial"* (i.e., IR). Precisò che l'IR si occupa soprattutto dell'estrazione di documenti importanti, lasciando a carico dell'utente la totale comprensione del contenuto semantico. Invece nel Text Mining (specializzazione del data mining) si tenta di derivare o scoprire nuova conoscenza dai dati. Quindi in generale, quando si vuole adoperare il Text Mining si deve tentare di soddisfare le seguenti caratteristiche:

- Essa deve operare su collezioni di testi di grandi dimensioni scritti in linguaggio naturale;
- Utilizzare algoritmi piuttosto che filtri manuali o euristici;
- Generare nuove conoscenze.

1.2 Differenze tra Text Mining e Data Mining

Il data mining può essere liberamente definito come alla ricerca di *pattern* nei dati, il text mining è in cerca di *pattern* nel testo. Tuttavia, la somiglianza tra le due nasconde reali differenze.

Il data mining può essere meglio caratterizzato come l'estrazione dai dati di informazioni precedentemente sconosciute e potenzialmente utili. L'informazione è implicita, non poteva essere estratta senza ricorrere a tecniche automatiche di data mining.

Col Text Mining, tuttavia, le informazioni da estrarre sono esplicitamente e chiaramente indicate nel testo. Il problema, naturalmente,

è che le informazioni non sono formulate in un modo strutturato e quindi non è possibile utilizzare elaborazioni automatiche. Con l'uso di tecniche di text mining si tenta di estrarre il testo in un formato adatto per l'uso degli algoritmi di data mining.

Quindi risulta che nel Data Mining le informazioni in ingresso sono archivi di dati strutturati, mentre il text mining dispone di un insieme di documenti testuali liberi. Quindi la prima operazione da eseguire è la trasformazione di questi documenti attraverso la fase *pre-processing*.

1.3 Principi del Text Mining

L'esponenziale crescita del web e dei suoi utilizzatori verificatosi negli ultimi anni hanno contribuito ad incrementare l'aggiunta di nuove fonti di conoscenza formando un gigantesco patrimonio informativo. Un altro fattore che ha determinato lo sviluppo del text mining, è l'enorme passo avanti che si è fatto nel settore linguistico, cioè di come è cambiata la visione della logica della "lingua". Si è passati da una logica di tipo "linguistico" ad una "lessico-testuale". Questo ha condotto ad importanti cambiamenti e alla realizzazione di nuove tecniche d'analisi dei testi.

Sempre più persone si sono "buttate" sullo studio del Text Mining perché si crede che il suo potenziale commerciale sia superiore a quello dell'estrazione dati, visto che la forma più naturale di memorizzazione delle informazioni è il testo. In effetti, basta pensare che la maggior parte della produzione d'informazioni di una società/azienda proviene da chat, e-mail, blog e forum e quindi testo scritto in linguaggio naturale, solo una piccola percentuale è espressa in dati numerici.

Con tali prospettive di sviluppo solo grazie al text mining si riesce a dare maggiore efficacia e robustezza all'analisi automatica dei documenti perché è l'unica tecnologia che esegue uno studio approfondito del "significato" del testo.

1.4 Classificazione testuale

I documenti testuali caratterizzano la maggior parte delle informazioni degli esperimenti sul Text Mining, quindi è necessario trasformarli dal formato destrutturalizzato in cui si trovano ad un formato strutturalizzato adatto per poter applicare gli algoritmi di data-mining. Per completare questa conversione bisogna applicare correttamente tutte le fasi elencate nei prossimi paragrafi, altrimenti si possono ottenere risultati insoddisfacenti.

1.4.1 Selezione dati

I motori di ricerca sono senza i sistemi più utilizzati per il recupero dei documenti da utilizzare negli esperimenti. Questa fase è molto importante perché se viene eseguita in modo approssimativo e non relativo alla descrizione del problema, si rischia si ottenere risultati non esatti. In molti casi, i documenti rilevanti possono essere già disponibili. Ormai con l'aumento esponenziale dei documenti digitali è spesso necessario selezionare un sottoinsieme dei dati presenti nelle biblioteche tradizionali.

Lo sviluppo tecnologico ha caratterizzato un miglioramento della qualità, numerosità ed eterogenità delle risorse. Sono nate specifiche organizzazioni che hanno investito mezzi e tempi per la raccolta / l'organizzazione / la selezione dei documenti.

1.4.2 Tokenizzazione

Una volta completata la fase di selezione, il passo successivo consiste nel manipolare i dati estraendo un insieme di elementi rilevanti quali parole, frasi o addirittura lettere. La tecnica più usata per eseguire questa operazione è detta *tokenization* (Tokenizzazione), la quale si occupa di suddividere il testo contenuto in un documento in *token* (i.e., blocco di testo atomico, tipicamente formato da caratteri indivisibili).

Il *token* è definibile come una qualsiasi sequenza di caratteri circondata dai delimitatori. Il problema sta proprio nel definire questi limitatori, infatti, come primo approccio si potrebbe pensare che gli spazi, le tabulazioni e i caratteri di ritorno a capo possono bastare per una corretta suddivisione. In realtà non è così perché ci sono moltissime eccezioni, basta pensare ai segni di punteggiatura presenti vicino alle parole: esempio, l'apostrofo di norma è situato in mezzo a due termini diversi che, basandoci sulla definizione data, sarebbero erroneamente rappresentati come un unico *token*.

Altri caratteri che, a seconda del contesto, possono essere o meno dei delimitatori sono: <> () . , : - ". Infatti, si possono creare delle ambiguità, per esempio:

- Il carattere "." usato come fine di una frase è diverso da quello che separa le cifre di un numero decimale;

- Il carattere "-" usato come operazione di differenza tra due numeri è diverso da quello usato come trattino in un testo;

Visto le difficoltà descritte sopra, è importante creare degli strumenti di tokenizzazione ad *hoc* in base al contesto in cui ci si trova. Le problematiche crescono maggiormente se s'intende identificare non le singole parole ma intere frasi.

L'unica certezza è che la tokenizzazione risulta facile per un umano ma come visto sopra può diventare difficile per un calcolatore.

Sempre in questa fase, è buona norma fare uso di una lista di *stopwords*, che tradotta letteralmente significa "parole da fermare". Questa lista contiene un elenco di termini che non devono essere considerati perché non sono rilevanti per i fini che s'intendono realizzare. Infatti, al suo interno presenta numeri, articoli, congiunzioni, preposizioni, avverbi, caratteri speciali e tutti quei vocaboli che, dopo una scrupolosa analisi delle frequenze, risultano essere comuni. I benefici di tale lista sono:

- Riduzione delle quantità di parole, abbassando così le risorse computazionali del processo;
- Aumento della qualità dei dati.

Come ultima attività conviene convertire i termini alla forma minuscola applicando così la funzionalità detta *casefolding*.

1.4.3 Stemming

Si definisce *stemming*, il processo di riduzione della forma flessa (i.e., una qualsiasi variazione morfologica) delle parole, alla forma base detta radice o meglio "tema".

Questo procedimento non è obbligatorio per gli obiettivi di classificazione dei documenti ma, in alcuni casi, può offrire un piccolo beneficio. Infatti si può affermare che, un effetto favorevole, è la diminuzione dei numeri distinti di tipi nel testo e l'incremento delle frequenze delle occorrenze in alcuni tipi. Un altro aspetto positivo, ma abbastanza discutibile, sta nel fatto che unendo termini simili nella stessa radice potrebbe garantire risultati migliori in fase di classificazione.

La creazione di un algoritmo di stemming è stato da sempre uno dei problemi più complicati dell'informatica, in quanto estremamente dipendente dal contesto in cui si sta lavorando e soprattutto dalla lingua utilizzata. Infatti per la lingua inglese bisogna occuparsi anche delle diverse forme irregolari presenti nelle regole grammaticali.

Esiste anche una "sotto-categoria" chiamata *Inflectional Stemming*, dove si applicano le seguenti conversioni:

1. Forme plurali convertite in quelle singolari. Esempio relativo alla lingua inglese: "cars" $\rightarrow$ "car";
2. Forme coniugate dei verbi trasformate nella relativa forma base. Esempio relativo alla lingua inglese: "fished" $\rightarrow$ "fish".

In generale, tutti gli algoritmi che tentano di sviluppare questa tecnica senza tener conto delle regole grammaticali e del contesto (*part*

of speech, parti del discorso), possono generare un immenso numero di errori causati dall'ambiguità derivata dal linguaggio. Esempio: il vocabolo "bored" può essere considerato come aggettivo o come verbo, in quest'ultimo caso può essere ricondotto a "bore" oppure a "bear". Per essere sicuro del troncamento che si deve eseguire, bisogna valutare il contesto in cui è inserita la parola.

In letteratura sono stati creati diversi tipi d'algoritmi di stemming [1] che differiscono tra loro rispetto all'accuratezza, prestazioni e su come risolvono i problemi che questa tecnica presenta. Sono elencati e descritti brevemente di seguito:

- "Lookup algorithms" (i.e., algoritmi di ricerca o forza bruta): un semplice *stemmer* cerca la forma flessa in una tabella di ricerca. Vantaggi: semplice, veloce e presenta una facile gestione delle eccezioni. Svantaggi: la tabella può essere molto grande perché tutte le forme flesse devono essere esplicitamente elencate: parole nuove o non familiari, non sono gestite anche se perfettamente regolari (ad esempio *"iPads"*);

- "Suffix-stripping algorithms" (i.e, algoritmi rimozione suffissi): contengono tipicamente un elenco di "regole" che forniscono un percorso per l'algoritmo, infatti dato in ingresso una forma di un termine, viene restituita la sua forma radice. Alcuni esempi di regole per la lingua inglese:

  - se la parola termina in "ed', rimuovere il "ed";
  - se la parola termina in "ing", rimuovere il "ing".

  Vantaggi: sono semplici da gestire rispetto agli algoritmi di forza bruta, supponendo che il manutentore è sufficientemente esperto della linguistica e della morfologia per creare le regole di codifica. Svantaggi: scarso rendimento quando si tratta di rapporti eccezionali (come "ran" e "run"). Problema: non tutte le parti di un

discorso sono formulate con un insieme di regole. Un esempio è descritto da Porter [22];

- "Lemmatisation algorithms" (i.e., algoritmi di lemmatizzazione): approccio abbastanza complesso. Si basa sulla determinazione della parte del discorso di una parola, per poi applicare le diverse regole di normalizzazione (che sono, più o meno, le stesse regole degli algoritmi di rimozione dei suffissi) per ogni parte del discorso;

- "Stochastic algorithms" (i.e., algoritmi stocastici): si creano un modello probabilistico attraverso una tabella contenente le relazioni tra le forme radici e quelle flesse. Questo modello è solitamente espresso con complesse regole linguistiche. Dopo di che, passando in ingresso una forma flessa, si ottiene in uscita la sua forma base con più alta probabilità di essere corretta;

- "Hybrid approaches" (i.e, approcci ibridi): utilizzano due o più dei metodi sopra descritti. Un semplice esempio è un algoritmo *suffix tree algorithm* il quale prima consulta una tabella di ricerca usando la forza bruta, poi se la parola non è stata trovata si applica o *Suffix-stripping algorithms* o *Lemmatisation algorithms*. Tuttavia, invece di cercare di conservare l'intera serie di relazioni tra le parole in una determinata lingua, la tabella di ricerca è mantenuta solo per memorizzare una quantità minima delle "frequenti eccezioni" come "ran" $\rightarrow$ "run";

- "Matching algorithms" (i.e, algoritmi di matching): usano un database di *stem*. Questi *stem*, non sono necessariamente parole ma piuttosto comuni sotto-stringhe. L'algoritmo prova a confrontare le parole con gli *stem* presenti nel database, applicando vari vincoli, come la lunghezza relativa dello *stem* all'interno della parola.

1.4.4 Features and vectors generation

Arrivati a questo punto, tutte le *words* (i.e., parole) estratte dai documenti sono state uniformate ma ancora non possono essere esaminate dagli algoritmi perché devono prima essere trasformate in un formato strutturato, come vettori e matrici.

Tipicamente un documento I è rappresentato come un vettore di pesi $P = \langle W_1, W_2, ..., W_{|A|} \rangle$ dove $|A|$ è la cardinalità dell'insieme degli attributi con cui si vuole descrivere I. La scelta di quali attributi adoperare deve essere fatta prendendo solo quelli indispensabili e quindi più rilevanti.

La scelta e il numero di elementi da adottare per descrivere ogni documento dipendono dai parametri che sono ritenuti più utili e significativi. Dopo di che si deve decidere come si vogliono identificare i termini in base agli attributi scelti, e sono possibili due soluzioni:

1. *bag of words*: i pesi $\langle W_1, W_2, ..., W_{|A|} \rangle$ vengono scritti in modo binario (e.g., "0" se il termine è presente nel documento, "1" altrimenti);

2. I pesi $\langle W_1, W_2, ..., W_{|A|} \rangle$ non vengono scritti in modo binario ma ad esempio si può considerare la frequenza della parola nel documento [14].

1.4.5 Classificazione

Finalmente il testo iniziale è stato strutturalizzato in record o vettori, pronti per potergli applicare algoritmi di data mining specifici per il *machine learning* (i.e., apprendimento automatico). Quest'ultimi, attraverso un processo induttivo, hanno l'obiettivo di trovare delle regole per discriminare i dati iniziali in base alle classi di appartenenza, quindi i testi devono già essere pre-classificati. Tali regole sono poi utilizzate per assegnare la classe appropriata a nuovi dati.

Definizione formale del problema: dato un set iniziale di documenti $\mathcal{D} = \{d_1, d_2, ..., d_{|\mathcal{D}|}\}$ pre-classificati in $\mathcal{C} = \{c_1, c_2, ..., c_{|\mathcal{C}|}\}$. Si dovrebbe ottenere una funzione ϕ : $\mathcal{D}$ x $\mathcal{C}$ $\rightarrow$ $\{T, F\}$ la quale indica se un documento d $\in \mathcal{D}$ deve appartenere alla categoria c $\in \mathcal{C}$ ($\phi\,(d, c) = \mathrm{T}$) o non deve appartenere ($\phi\,(d, c) = \mathrm{F}$). Tale funzione deve essere il più simile possibile alla funzione "target"(obiettivo) ϕ' : $\mathcal{D}$ x $\mathcal{C} \rightarrow \{T, F\}$, la quale è considerata come la giusta funzione classificatrice.

Dopo di che per valutare l'efficienza di un classificatore, è possibile utilizzare due metodi:

1. *Training and test set*: il set iniziale viene diviso in due sottoinsiemi T_R e T_S, non necessariamente devono avere la stessa cardinalità:

   - Training set $T_R = \{d_1, d_2, ..., d_{|T_R|}\}$ usato per costruire le regole del classificatore ϕ';
   - Test set $T_S = \{d_{|T_R|+1}, d_{|T_R|+2}, ..., d_{|\mathcal{D}|}\}$: usato per verificate l'accuratezza del classificatore. Per far questo, tutti i risultati ottenuti da $\phi'\,(d_i, c)$ dove $d_i \in T_S$, vengono confrontati con $\phi\,(d_i, c)$.

2. *k-fold cross validation*: consiste di k classificatori $\phi_1, \phi_2, ..., \phi_k$ diversi, creati dividendo il set iniziale in k sotto-insiemi disgiunti $T_{S_1}, T_{S_2}, ..., T_{S_k}$ che formeranno i *test set*. In seguito, si applica k volte il metodo *training and test set* aventi $T_{R_i} = \mathcal{D} - T_{S_i}$. Per ricavare l'accuratezza totale, bisogna far la media di quelle ricavate dai singoli classificatori.

Capitolo 2

Stato dell'arte

Visto l'esponenziale aumento della quantità e dell'uguaglianza dei documenti, molti ricercatori sono passati da una visione *flat classification* (classificazione "piatta") ad una *hierarchical classification* (classificazione gerarchica). In quest'ultima, le classi sono organizzate in modo gerarchico sviluppando così delle specifiche relazioni tra loro.

Recentissimi lavori [14, 4, 31, 10] utilizzano delle collezioni di documenti già opportunamente classificate come Reuters Corpus Volume I (RCV1) [16], Reuters-21578 [15], web directory DMOZ [19] e Yahoo! Directory.

Il problema più complesso della classificazione sta nel rappresentare i documenti, infatti il formato testuale così com'è non può essere impiegato dagli algoritmi se prima non ne viene trasformato il contenuto in una forma numerica.

In questo capitolo sono descritte le più importanti caratteristiche che differenziano la classificazione dalla clusterizzazione. Entrambe sono tecniche relative rispettivamente a *supervised learning* (apprendimento supervisionato) e *unsupervised learning* (apprendimento non supervisionato).

In seguito si definiscono i vantaggi che si possono ottenere, per la risoluzione del problema di classificazione di documenti in una gerarchia di categorie, adoperando una classificazione di tipo gerarchica piuttosto che quella *flat*. Sempre per lo stesso problema, si analizzano

dettagliatamente alcuni dei recenti classificatori utilizzati.

Infine, si approfondiscono due lavori presenti in letteratura per avere un'idea dei risultati e degli strumenti utilizzati.

2.1 Classificazione vs. clusterizzazione

La classificazione è una tecnica supervisionata, infatti i dati iniziali (training set) devono essere etichettati, cioè pre-calcolati. L'obiettivo è quello di trovare delle regole per discriminare le varie classi, in modo tale da assegnare la classe appropriata a nuovi dati. La classificazione è un processo costituito di due fasi:

1. Costruzione del modello a partire dal training set dato;
2. Mediante il modello si assegna l'etichetta di classe a nuovi dati (test set) non etichettati;

Il modello può essere realizzato mediante:

- Alberi di decisione;
- reti neurali;
- Regole di associazione;
- ...

Il *clustering* (clusterizzazione) è una tecnica non supervisionata (perché manca l'etichetta di classe). Il suo scopo è quello di suddividere una collezione d'oggetti in gruppi, definiti *cluster*, tali che:

- Gli oggetti nello stesso cluster sono simili, quindi si massimizza la similarità intra-cluster;
- Gli oggetti in cluster diversi sono dissimili, quindi si minimizza la similarità inter-cluster.

Il clustering ci aiuta ad individuare dati omogenei permettendoci di capire se esistono dei gruppi non casuali, cioè che mostrano delle regolarità e che condividono certe caratteristiche. Il clustering può essere utilizzato anche al posto della classificazione, facendo corrispondere ai cluster individuati le classi prestabilite, ma in genere porta a risultati meno accurati.

2.2 Classificazione flat vs. classificazione gerarchica

Gli ultimi studi ci forniscono delle soluzioni riguardanti al noto problema definito come "classificazione di documenti in gerarchie di categorie".

Alcuni di questi utilizzano la classificazione *flat* (piatta) [2, 7], la quale assegna ai documenti la categoria in base ai risultati ottenuti da uno o più classificatori, senza considerare una struttura gerarchica.

In altri casi invece, si adopera la classificazione *hierarchical* (gerarchica) [7, 2, 27], in cui si costruiscono classificatori per ogni livello della struttura gerarchica dove sono posizionate le categorie, quindi l'assegnamento della classe di appartenenza è effettuato sulla base dei risultati ottenuti ad ogni livello.

Con quest'ultima tecnica si ha la possibilità di scomporre il problema in sotto-problemi più semplici da risolvere, ottenendo così risultati migliori. Con questa visione ci si adegua alla struttura gerarchica dell'albero aprendo nuove prospettive. Ad esempio: costruzione di un criterio di terminazione quando si esegue la ricerca della categoria di un documento nei nodi interni.

2.3 Classificatori

I classificatori non fanno altro che tentare di risolvere il problema di classificazione (già ampiamente affrontato nel paragrafo 1.4.5).

Nell'elenco sottostante descriviamo le proprietà desiderate da un algoritmo di questo tipo:

- "computational efficiency" (i.e., efficienza computazionale): questa misura descrive quanto un algoritmo è relativamente adatto ad affrontare i diversi problemi;
- "robustness" (i.e., robustezza): tolleranza ai "rumori" presenti nei dati d'input;
- "statistical stability" (i.e., stabilità statistica): i modelli creati dall'algoritmo non devono essere basati su induzioni casuali ma generalizzare i concetti presenti nei dati in modo tale che, applicando due dataset diversi relativi alla stessa area tematica, l'algoritmo generi lo stesso modello.

Nei sotto-paragrafi saranno descritti alcuni dei classificatori sia utilizzati negli ultimi tempi sia creati molti anni fa [24]. Una parte delle informazioni presentate sono state estratte dall'articolo [14] il quale ne esegue un'analisi più approfondita.

2.3.1 Naïve Bayes

Offre un modello probabilistico avente una dichiarazione esplicita delle ipotesi semplificative adottate. Tipo:

- I documenti possono essere modellati come distribuzione di probabilità di eventi indipendenti;
- Ogni evento(termine) deve essere indipendente dal contento all'interno del documento.

Esso dispone della tipologia *Bernoulli* e di quella *multinomial* (i.e., multinomiale) per rappresentare i documenti. La prima rappresenta la presenza o assenza di una parola in un documento, attraverso un valore binomiale. La seconda è basata sul calcolo della probabilità del

numero di occorrenze di una parola in un documento. Entrambe sono relative agli aspetti descritti nel paragrafo 1.4.4.

Si è già dimostrato che la tecnica *multinomial* è superiore a quella di *Bernoulli* [18].

L'aspetto principale del Naïve Bayes è l'uso del teorema di Bayes, il quale "viene impiegato per calcolare la probabilità di una causa che ha scatenato l'evento verificato" [1].

Formalmente, si definisce classificatore quella funzione f che mappa i vettori delle *feature* $x \in \mathcal{X}$ in ingresso con le uscite $y \in \{1, 2, ..., C\}$ rappresentanti le etichette delle classi, dove $\mathcal{X}$ identifica lo spazio delle parole.

I classificatori probabilistici, in questo caso Naïve Bayes, utilizzano la probabilità $p(y|x)$. Per far questo si calcola per ogni valore di y la *class-conditional density* $p(x|y)$ e la *class priors* $p(y)$. Poi viene applicata le regola di *Bayes* per computare la parte posteriore:

$$p(y|x) = \frac{p(x,y)}{p(x)} = \frac{p(x|y)\,p(y)}{\sum_{y'=1}^{C} p(x|y')\,p(y')}$$

2.3.2 K-Nearest Neighbour

L'algoritmo K-Nearest Neighbour (k-NN) è molto semplice da implementare ed è utilizzato spesso come valore di confronto con gli algoritmi più sofisticati.

Gli approcci per questo tipo di classificatori sono diversi, quello più adoperato è *majority vote method* (i.e., metodo di voto della maggioranza) dove la classe da assegnare ad un documento è scelta guardando quella più frequente nei K vicini.

Un altro approccio calcola dei pesi per ogni vicino attraverso la distanza proporzionalmente inversa dal punto di test e anche in base alle loro capacità di classificare.

Supponendo che il calcolo della distanza sia poco costoso (problemi espressi a bassa dimensionalità), la complessità è lineare con il numero di vettori utilizzati. Negli altri casi, vengono apportate delle modifiche per migliorare i tempi di esecuzione. Esempio: rimozione dei vettori che non contribuiscono a prendere le decisioni sui dati.

Questo algoritmo è un'approssimazione locale della funzione obiettivo, senza eseguire dei tentativi per formare la funzione obiettivo sull'intero spazio delle istanze. Questo porta dei vantaggi quando la funzione *target* è complicata.

Formalmente: dato un documento x, viene calcolata la classe y d'appartenenza di x attraverso l'insieme dei vicini $N(x)$. La funzione f associata al classificatore (il significato di questa funzione è descritto nel paragrafo precedente) viene calcolata:

$$f(x) \;=\; argmax_{y \in Y} \sum_{x' \in N(x)} \phi(y, y')$$

Dove y' è la classe associata a x e la funzione ϕ è descritta:

$$\phi(y, y') \;=\; \begin{cases} 0, & se\ y \neq y' \\ 1, & se\ y = y' \end{cases}$$

2.3.3 Metodo di Rocchio e Centroid based

Il metodo di Rocchio si basa sulla famosissima formula di Rocchio [24] basata sul meccanismo conosciuto come *relevance feedback* relativo al *Vector Space Model* (dove ogni documento è rappresentato come un vettore in uno spazio comune). L'idea alla base del *relevance feedback* è quella di prendere i risultati che sono inizialmente restituiti da una data query e di utilizzare le informazioni indipendentemente dal fatto che i risultati siano o meno rilevanti per eseguire una nuova query. Questo è l'unico metodo del *Text categorization* che affonda le sue

radici nelle tradizioni dell'*Information Retrieval* (IR) piuttosto che in quelle del *Machine learning* (ML).

In termini matematici, un classificatore $\vec{c_i} = \langle w_{1i}, w_{2i}, ..., w_{|\mathcal{T}|i} \rangle$ si calcola per la categoria c_i attraverso la formula [26]:

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

dove w_{kj} è il peso di t_k nel documento d_j, $POS_i = \{d_j \in T_r | \Phi(d_j, c_i) = T\}$ e $NEG_i = \{d_j \in T_r | \Phi(d_j, c_i) = F\}$. In questa formula, β e γ sono parametri di controllo che consentono di settare l'importanza degli esempi positivi e negativi.

Vantaggi: è abbastanza semplice da implementare, ed è anche molto efficiente dal momento che l'apprendimento del classificatore è relativo ad una media di pesi. Svantaggio: un classificatore costruito con il metodo Rocchio, come tutti i classificatori lineari, ha lo svantaggio che divide lo spazio di documenti linearmente.

Il metodo di Rocchio è una generalizzazione del classificatore basato su centroide: si basa sugli stessi principi ma tiene conto solo dei documenti che fanno parte della categoria (i.e., gli esempi positivi) e non considera quelli che non ve ne fanno parte (i.e., esempi negativi).

Più precisamente, si costruisce un documento-vettore-prototipo per ogni categoria e si classifica ciascun altro documento in base alla distanza misurata con *cosine similarity* da questi prototipi. Il valore di *cosine similarity* è calcolato con la seguente formula:

$$\cos(d_i, d_j) = \frac{d_i \cdot d_j}{||d_i||_2 * ||d_j||_2}$$

La funzione mappa la similarità dei documenti in un valore compreso tra 0 ed 1.

2.4 Classificazione gerarchica tramite SVM

Viene illustrato uno studio eseguito da A. K Pulijala e S. Gauch [23], il quale analizza alcuni aspetti importanti sulla classificazione di documenti web in una gerarchia di categorie.

Si classificano i documenti durante la loro indicizzazione in modo che poi possano essere recuperati da una combinazione di parole chiave e da una corrispondenza concettuale.

La maggior parte degli approcci utilizzano classificatori *flat* che ignorano la struttura gerarchica, trattando ogni argomento come una classe separata. Sebbene questi classificatori *flat* sono computazionalmente semplici, non riescono a sfruttare le informazioni inerenti alle relazioni strutturali tra gli argomenti. Questo articolo esplora l'utilizzo della struttura gerarchica per classificare enormi ed eterogenei contenuti web.

2.4.1 Introduzione

Piuttosto che costruire un unico grande classificatore, si è deciso di costruire una gerarchia di classificatori che incrementano l'accuratezza, concentrandosi solo su un piccolo insieme di classi (e.g., le più rilevanti) ad ogni livello.

Il problema di classificazione può essere scomposto in un insieme di problemi più piccoli corrispondenti alla divisione gerarchica nell'albero.

L'approccio utilizzato è *top-down* basato a livelli, la classificazione è realizzata con la collaborazione di classificatori costruiti a ciascun livello dell'albero. Il documento da testare inizia alla radice dell'albero per poi essere confrontato con le classi del primo livello. Il documento è assegnato alla classe più appropriata e viene quindi confrontato con tutte le sotto-classi di tale classe. Questo processo continua fino a quando il documento raggiunge una foglia o una classe interna al di sot-

to della quale il documento non può essere ulteriormente classificato. I classificatori usati sono basati su Vector Space Model (VSM).

2.4.2 Architettura

Il processo di classificazione comprende due fasi:

- Addestramento del classificatore: sono raccolti ed uniti un numero fissato di documenti per ogni concetto, ottenendo un super-documento D. Quest'ultimo è poi pre-elaborato (rimozione *stopwords* e HTML tag e l'utilizzo dello *stemmer Porter* [9]) e indicizzato usando il metodo $tf * idf$. Questo in sostanza rappresenta il baricentro per ogni concetto.
- Classificazione di nuovi documenti: i documenti sono indicizzati utilizzando un VSM che viene poi confrontato con i centroidi di ogni concetto per poter classificare tale documento.

VSM: un concetto j è rappresentato con un vettore c_j contenente una voce per ogni termine nel vocabolario. Il peso tc_{ij} per un dato termine i è un fattore di frequenza del termine nel super-documento per il concetto, tf_{ij} e la rarità della parola in un altro concetto, idf_i.

$$tc_{ij} \ = \ tf_{ij} \ * \ idf_i$$

Dove:

1. tf_{ij} = numero di occorrenze della parola i-esimo nel j-esimo super-documento;
2. $idf_i = \text{Log}(\frac{Numero\ documenti\ in\ D}{Numero\ di\ documenti\ in\ D\ che\ contengono\ t_i})$

Poiché non tutti i documenti sono della stessa lunghezza, i pesi relativi ad un termine i di ciascun vettore concetto j sono normalizzati ntc_{ij} per l'ampiezza del vettore.

$$ntc_{ij} \ = \ \left(\frac{tc_{ij}}{\sum_i \ tc_{ij}} \right)$$

2.4.3 Esperimenti

I documenti utilizzati per gli esperimenti sono stati presi da DMOZ [19], prendendo come training data i primi quattro livelli dell'albero e come test set solo i primi tre. Inoltre, si sono considerate solo le classi con almeno 20 documenti. Si ottiene così:

Livello	**Classi**
Primo	15
Secondo	356
Terzo	2.812
Quarto	3.895

Tabella 2.1: Tabella dataset Pulijala e Gauch

Ogni classificatore di un livello è addestrato con i documenti associati a quella classe e a tutte le sotto-classi del livello. Dopo diversi esperimenti si è concluso che i miglior risultati si ottenevano addestrando i classificatori con le seguenti medie:

Livello	**Documenti**
Primo	7078
Secondo	7063
Terzo	6707

Tabella 2.2: Training set Pulijala e Gauch

Per misurare l'accuratezza del classificatore si è adoperato un test set contenente 750 documenti selezionati casualmente nel terzo livello dell'albero. Questi documenti sono stati esclusi dal processo di addestramento e sono stati selezionati da 750 diverse classi.

Il primo test (figure 2.1) mostra l'accuratezza della classificazione eseguita adoperando un numero variabile di parole per documento (da 1 a 35). Il miglior risultato, 70% di accuratezza, si ottiene quando le parole usate per i documenti sono 19. Con l'aumentare di questo valore, l'accuratezza tende a scendere.

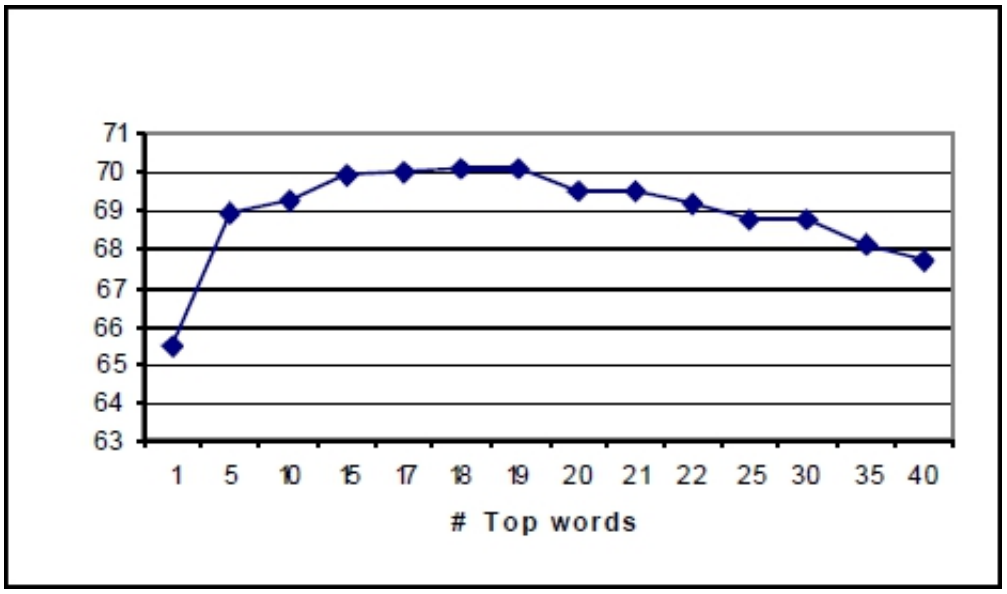


Figura 2.1: Risultati test Pulijala e Gauch

2.5 Framework per la classificazione gerarchica

Viene illustrato un recente studio eseguito da M. Ceci e D. Malerba [4], il quale analizza alcuni aspetti importanti della classificazione di documenti web in una gerarchia di categorie. Viene presentato il loro framework per la classificazione gerarchica chiamato WebClassIII.

I documenti vengono rappresentati mediante la selezione di un insieme di parole (features) e definito *bag-of-words*. E' possibile definire diverse rappresentazioni per ogni documento, e ognuna è utile per la classificazione del documento ad un livello della gerarchia. Per esempio, i documenti del tema generale "Math" possono essere ben rappresentati da termini generali come "mathematics", mentre documenti relativi a temi più specifici (e.g., "geometry" cioè geometria) sono meglio rappresentati da termini specifici come "parallelepiped" (i.e., parallelepipedo).

2.5.1 Strategia

L'approccio utilizzato per la classificazione dei documenti è quello *top-down* che consiste nel discendere l'albero partendo dalla radice fino ad

arrivare alle categorie foglia adoperando strategie di tipo *greedy*.

Quando il documento arriva ad una categoria interna "C", viene rappresentato sulla base del set di funzioni associate in "C". Il classificatore di categoria "C" restituisce uno *score* (i.e., punteggio) per ogni sottocategoria diretta. A questo punto, viene scelta la sottocategoria con il punteggio più alto superiore ad un valore che ne definisce la soglia. La ricerca procede ricorsivamente da quella sotto-categoria, fino a quando si raggiunge una categoria foglia oppure i punteggi non superano la soglia. Se la ricerca si ferma alla radice, allora il documento è considerato non classificato.

Un aspetto molto importante di questo procedimento è il calcolo delle soglie perché determina se il documento deve essere trasmesso ad una categoria di livello inferiore. Questo passaggio non ha senso effettuarlo nel caso in cui:

1. Il documento non tratti di argomenti specifici ma generali;

2. Il documento da classificare appartenga ad una specifica categoria che non è presente nella gerarchia e quindi ha più senso classificare il documento nella categoria generale piuttosto che in una sbagliata.

L'algoritmo per la determinazione della soglia automatica si fonda su una strategia bottom-up e cerca di minimizzare una misura *minimal score* (punteggio minimo) che si basa su una *tree distance* (distanza nell'albero). In una *tree distance*, la diversità tra due categorie è riprodotto come somma dei pesi di tutti gli archi dell'unico percorso che collega le due categorie nella gerarchia. Quando un arco pesa una unità (come nel WebClassIII), la dissomiglianza è la lunghezza del percorso.

Nel framework di Ceci e Malerba, il set di *feature* è unico per ogni categoria interna e viene automaticamente determinato per mezzo di una serie di esempi di addestramento positivi e negati-

vi. Più specificamente, i documenti di training subiscono le seguenti operazioni:

1. Tokenizzazione;
2. Applicazione dei filtri per rimozione di tag HTML, parole con meno di tre lettere e segni di punteggiatura;
3. Rimozione di parole non significative, come articoli, avverbi e preposizioni tramite le cosidette *stopwords*;
4. Uso di tecniche di *stemming*.

Per la creazione dell'insieme delle *fetaures* ci si basa sul prodotto TF x DF^2 x ICF, dove:

- TF (term frequency): frequenza relativa di un termine in un documento;
- DF (document frequency): percentuale di documenti di una categoria in cui è presente una determinata parola;
- ICF (inverse category frequency): inversa di CF (category frequency). CF è il numero di sotto-categorie relative ad una categoria in cui è presente una determinata parola;

Per maggiori dettagli implementativi e matematici sull'estrazione delle *features* si può far riferimento all'articolo di Ceci e Malerba [4].

2.5.2 Sistema di apprendimento

Visto l'uso dello stesso set di *features* per una categoria e le sue sottocategorie permette l'uso di algoritmi di apprendimento multi-classe. Gli approcci di apprendimento considerati sono:

1. Naïve Bayes [20]] basato sul modello multinomiale, il quale specifica che un documento è rappresentato dalla serie delle occorrenze dei termine del documento e queste occorrenze influiscono il calcolo della probabilità a posteriori;

2. Un metodo basato sul centroide [12] dove ogni centroide è il centro di un cluster di documenti appartenenti alla stessa categoria;

3. L'SVM (Support Vector Machine) utilizzato è una versione modificata del classificatore Sequential Minimal Optimization (SMO) [21]. Il metodo è molto veloce ed è basato sull'idea di spezzare un grosso problema QP (quadratic programming) in una serie di problemi QP più piccoli che possono essere risolti analiticamente. Il metodo originale è definito per solo due classi, mentre Ceci e Malerba lo hanno ricondotto per una classificazione multi-classe.

La classificazione di un nuovo documento in una categoria "C" si ottiene come segue:

- Stimando la probabilità bayesiana a posteriori per tale categoria (Naïve Bayes).

- Calcolando la somiglianza tra il documento e il baricentro di quella categoria (Centroid-based).

- Stimando la probabilità a posteriori per la categoria in base ad un classificatore SVM probabilistico.

2.5.3 Esperimenti

I dataset adottati per gli esperimenti effettuati nell'articolo di Ceci e Malerba, sono:

- DMOZ: contiene 5.612 documenti organizzati in 221 categorie secondo una struttura gerarchica foramta da 5 livelli;

- Yahoo: contiene 907 documenti disposti in 68 categorie secondo una struttura gerarchica foramta da 3 livelli;

- RCV1(Reuters Corpus Volume 1);

Si analizzano i risultati dei primi due dataset usati, riportandone i risultati ottenuti. Per maggiori informazioni sulle caratteristiche e sui contenuti presenti si può far riferimento al paragrafo 4.1.2.

Le modalità e le misure adottate sono:

- Validazione tramite five-folds cross-validation;
- Le cinque misure di valutazione considerate sono:
  1. *Accuracy* (accuratezza): rappresenta il numero di documenti di test classificati correttamente su tutti i documenti di test;
  2. *Misclassification error*: calcola la percentuale di documenti classificati in modo errato, in una categoria non parentata alla categoria corretta;
  3. *Generalization error*: misura la percentuale di documenti classificati in modo errato in una super-categoria della categoria corretta;
  4. *Specialization error*: calcola la percentuale di documenti classificati in modo errati in una sotto-categoria di quella corretta;
  5. *Unknown ratio*: misura la percentuale di documenti non classificati.
- Gli approcci utilizzati per la selezione delle *feature* si differenziano per dimensione e tipologia:
  - *Flat*;
  - *Hierarchical feature set*;
  - *Hierarchical with proper feature set.*

Nel grafico 2.2 è riportata la misura dell'accuratezza dei risultati ottenuti. Osservando attentamente la figura ci si rendo conto immediatamente che i migliori risultati si sono ottenuti adoperando il classificatore SVM flat. L'approccio *flat* ha un'accuratezza migliore

di quelli gerarchici, sia utilizzando il metodo SVM che quello basato sui centroidi. Queste osservazione valgono sia per i test effettuati su Yahoo che Dmoz.

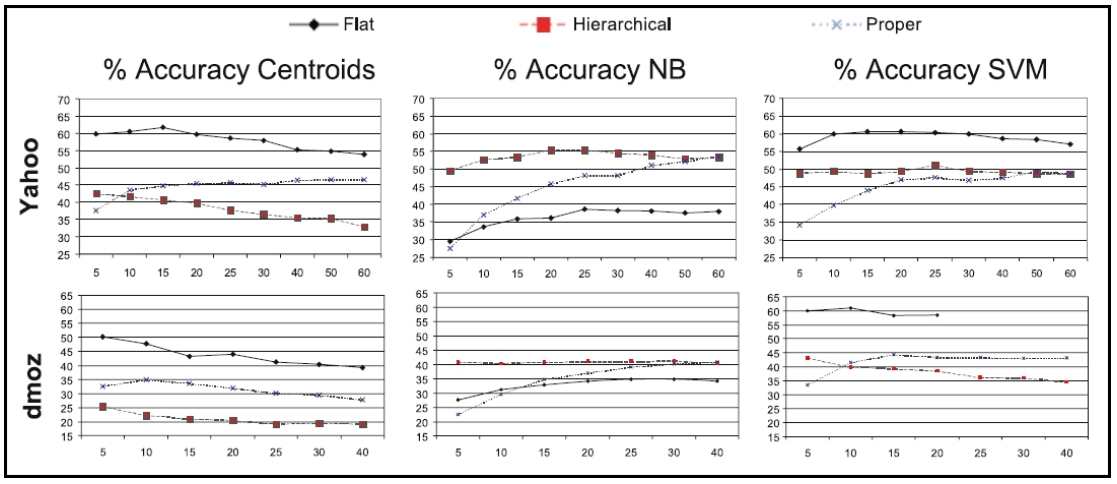


Figura 2.2: Risultati esperimenti Ceci e Malerba

I modelli gerarchici hanno maggior efficenza di quelli *flat* quando si fa uso del classificatore Naïve Bayes. Si nota anche come quello che utilizza l'insieme delle *features* corrette è meno accurato quando il numero di parole selezionate per i documenti è basso (i.e., 5,10,15,25), ma quando il numero di parole aumenta ha gli stessi risultati dell'altro metodo.

Capitolo 3

Strumenti utilizzati

Gli strumenti adoperati per lo sviluppo di questa tesi sono: DMOZ *directory web* impiegato per l'acquisizione dei dati su cui lavorare, WordNet database lessicale e semantico utilizzato per scoprire le relazioni esistenti tra diverse parole e Weka software open source impiegato per l'esecuzione di algoritmi di Data Mining. Nei paragrafi successivi saranno definiti più specificatamente.

3.1 DMOZ

DMOZ (Directory Mozilla) [19] conosciuto anche come ODP (Open Directory Project) è la più esauriente directory di siti e pagine web curata da uomini per le differenti lingue. Un insieme di editori volontari garantiscono il corretto inserimento dei siti Web per poter conservare la struttura gerarchica.

Nasce dal fatto che la crescita del Web è caratterizzata da ritmi sempre più elevati e quindi i motori di ricerca automatizzati hanno difficoltà a produrre risultati utili dai criteri di ricerca. I piccoli staff editoriali impiegati presso i siti di directory commerciali non riescono a fare fronte alle richieste, caratterizzando un abbassamento della qualità e della completezza delle loro directory. Essi non sono in grado di tenere il passo con la crescita di Internet a causa dell'obsolescenza dei collegamenti.

Invece di combattere l'esplosiva crescita di Internet, DMOZ fornisce un mezzo per organizzarsi autonomamente. La crescita di Internet corrisponde a quella del numero di cittadini presenti nella rete, ciascuno di questi può organizzare una piccola porzione del Web e proporla al resto della popolazione, selezionando il materiale migliore e tralasciando quello sgradevole o inutile. Gli obiettivi di DMOZ sono:

- completezza della directory: vi è una precisa descrizione che illustra cosa si deve e non si deve mettere nella proposta di un sito web. Questo è fatto per potere garantire una qualità e anche una certa quantità di contenuti unici (i.e., non presenti in siti già catalogati);

- organizzazione della directory: chiunque può richiedere l'inserimento di un nuovo sito web, ma la catalogazione spetta ad un editore. Quest'ultimo ne esegue una sua valutazione per deciderne la bontà e il posizionamento su una precisa categoria tematica che rappresenta i contenuti del sito.

Andando nello specifico e come si vede dalla figura 3.1, i dati presenti momentaneamente dentro DMOZ sono: 5.007.664 siti - 94.441 editori - 1.010.258 categorie. Questo solo per quanto riguarda la lingua inglese, mentre per tutte le altre lingue abbiamo 1.892.754 categorie.

Le categorie sono organizzate su una struttura ad albero che, partendo da una categoria situata ad un livello superiore dell'albero, permette di scendere più in profondità verso tematiche sempre più specifiche riferite alla categoria superiore, fino a raggiungere le foglie che referenziano le pagine web dai contenuti specifici.

Ogni categoria oltre ad avere le proprie sottocategorie, ha anche:

- categorie collegate: sono collegamenti trasversali identificati dal simbolo "@" che identificano categorie che possono essere sottocategorie ideali;

- categorie correlate: sono categorie rappresentate sotto la voce "vedi anche" e riguardano argomenti connessi però situate in altre aree tematiche;

- categorie analoghe: rappresentano la stessa categoria ma in altre differenti lingue.

Il numero di livelli dell'albero va da 0 a 14. Al livello "0" è disposta un'unica categoria denominata "Top" che identifica il nodo radice dell'albero e al suo interno vi sono presenti i documenti che trattano tematiche più generiche, mentre a livello "1" abbiamo gli argomenti raffigurati nella figura 3.1, i quali rappresentano gli "ancestor" per le discendenze.

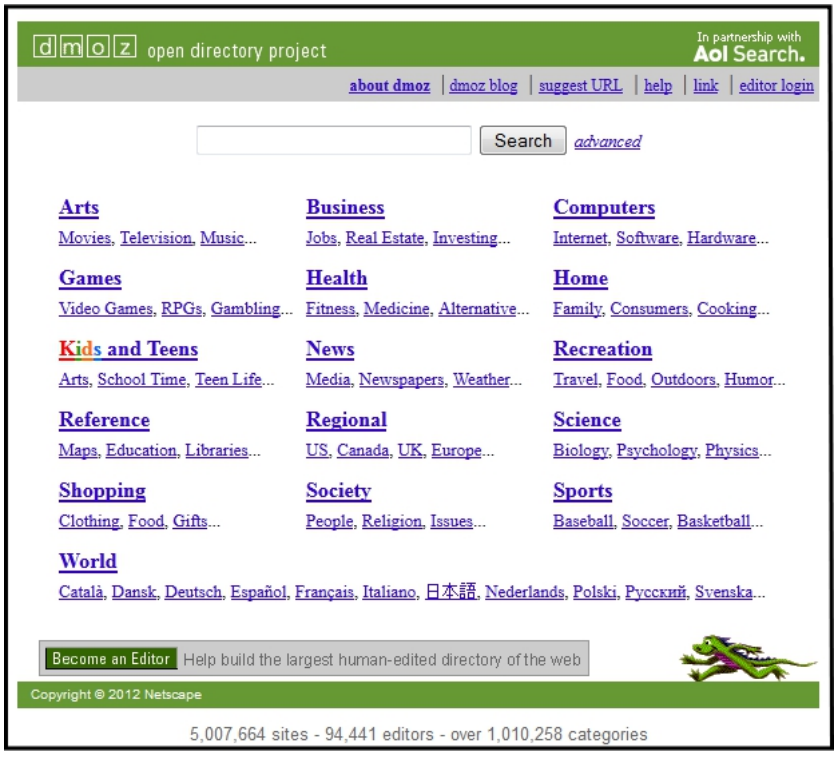


Figura 3.1: Sito Web di DMOZ

DMOZ dà la possibilità di scaricare i file in formato RDF (Resource Description Format) che contengono dati. La principale caratteristica di questo formato è la presenza di determinati "tags" che identificano una precisa informazione:

- "Topic": descrizione del percorso eseguito nell'albero per raggiungere la categoria (e.g., "Top/Arts/Animation/Production");
- "catid": identificativo numerico univoco per ogni categoria;
- "Title": titolo della categoria;
- "Description": una descrizione concisa degli argomenti trattati nella categoria (e.g., "Provides information resources to the international animation community. Features include searchable database archives, monthly magazine, web animation guide, the Animation Village, discussion forums and other useful resources.").

3.2 WordNet

WordNet è un database lessicale e semantico per la lingua inglese elaborato dal linguista George Miller presso l'Università di Princeton nel 1995. La sua caratteristica principale è quella di organizzare, definire e descrivere i concetti espressi dai vocaboli.

WordNet nel 1997 è divenuto un copyright dell'Università di Princeton la cui licenza d'uso ne prevede un utilizzo gratuito anche per fini al di fuori della ricerca con l'unico obbligo di citare gli autori e l'indirizzo del sito web. E' anche disponibile gratuitamente per il download nella pagina ufficiale del sito: `http://wordnet.princeton.edu`.

La sua struttura lo rende uno strumento utile per la linguistica computazionale e l'elaborazione del linguaggio naturale. Sostantivi, aggettivi, verbi e avverbi sono raggruppati in insiemi di sinonimi cognitivi (synset), ognuno dei quali esprime un concetto distinto. I synsets sono interconnessi mediante la semantica concettuale e relazioni lessicali. In più collega fra loro non solo la forma delle parole (stringhe di lettere), ma anche specifici sensi di parole.

L'obiettivo è stato quello di sviluppare un sistema che fosse coerente con le conoscenze acquisite nel corso degli anni su come elaborare il

linguaggio degli esseri umani. L'innovazione che è stata introdotta grazie a WordNet sta nel memorizzare le informazioni su base semantica, categoria sintattica di appartenenza e sulle relazioni che si formano tra le varie parole. Un progetto correlato, finanziato dalla Comunità europea, è *EuroWordNet* che realizza le stesse funzionalità e la struttura di WordNet ma aggiungendo alla lingua inglese anche quelle Europee.

La versione attuale 3.0 è stata rilasciata nel Dicembre del 2006, e contiene 147.278 parole (lemmi) distinti organizzati in 117.659 insiemi di synset.

Nella sua organizzazione vi sono presenti due proprietà fondamentali:

- Sinonimia: è relativa a parole che denotano lo stesso concetto e sono intercambiabili in molti contesti. Come già detto, i termini sono raggruppati in insiemi non ordinati (synset).
- Polisemia: è relativa alla proprietà che una parola ha di esprimere più significati.

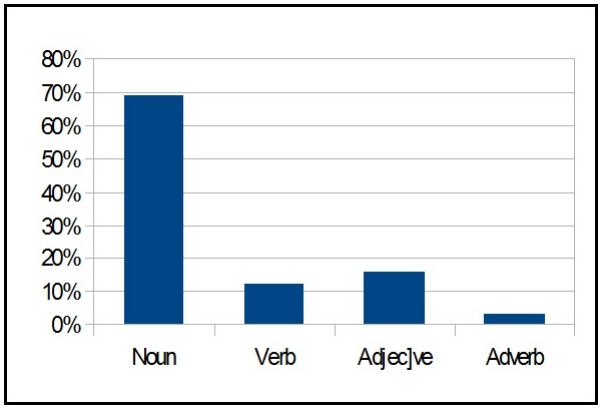


Figura 3.2: Percentuali di synset per ogni tipologia

3.2.1 Relazioni

Ci sono ben 28 tipi di relazioni e in più c'è la "sinonimia" che viene considerata, come visto in precedenza, più come una proprietà che una relazione su cui si basa WordNet.

Vengono di seguito elencate:

1. hyponym (iponimia): collega i synset più generali con quelli più specifici. Si è affermato che: un X sia iponimo di Y se si può definire il rapporto $X\ is-a\ Y$ il che significa che X *è un tipo di* Y. Questa relazione è transitiva, infatti può affermare che se una poltrona è un tipo di sedia, e se una sedia è un tipo di mobili, allora una poltrona è un tipo di mobili;

2. instance hyponym: indica una istanza di hyponym. Serve per distinguere i tipi da specifiche istanze (e.g, la poltrona è un tipo di sedia, Mario Rossi è un'istanza di un persona). Le istanze sono sempre i nodi foglia nelle gerarchie.

3. hypernym (iperonimia): collega i synset più specifici con quelli più generali;

4. instance hypernym: indica una istanza di hypernym;

5. meronym (meronimia): è il legame tra una parola che designa la parte di un tutto e la parola che rappresenta l'intero. Detto in modo formale si ha che X è meronimo di Y se X *è parte di* Y:

    6. a) part meronym: indica che un concetto (componente) è parte di un altro concetto (oggetto), ma può esistere anche in modo separato con una sua funzione (e.g.,. ruota/macchina);

    7. b) member meronym: definita da elemento/insieme e indica l'appartenenza (intesa come nell'omonima nozione matematica) di un concetto individuale (elemento) al concetto collettivo (insieme) (e.g.,ossigeno/aria);

    8. c) substance meronym: definita da materiale/oggetto e indica di che materiale è costituito l'oggetto. A differenza di *part meronym* le due parole sono una parte non separabile e integrante dell'altra (e.g., plastica/bottiglia);

9. holonym (olonimia): è l'inversa della meronimia:
    10. a) part holonym;
    11. b) member holonym;
    12. c) substance holonym;
13. entail (i.e., entailment): è riferita ai verbi. Infatti si dice che il verbo X è un'implicazione del verbo Y se nel fare Y uno deve per forza fare X (e.g., russare/dormire). Non necessariamente si ha che il contrario è valido;
14. cause (i.e., cause to): lega verbi causativi con il rispettivo verbo risultativo. Questa tipologia è di tipo implicazione, ne fanno parte ad esempio i verbi dare e avere;
15. antonym (antonimia): organizza gli aggettivi. Coppie di antonimia "diretta" come bagnato-asciutto e giovane-vecchio riflettono il contratto di forte semantica dei loro membri. Ognuno di questi aggettivi polari a sua volta è collegato a una serie di relazioni "semanticamente simili", per esempio: "secco" è legato a "arida", "essiccato" e ecc. Aggettivi semanticamente simili si dicono di antonimia "indiretta";
16. similar (similarità): valida per definire due aggettivi che stanno tra loro in relazione secondo l'espressione X *è simile a* Y;
17. also (i.e., see also): è di tipo lessicale e unisce parole che appartengono a synset diversi;
18. attribute: riferito ai "nomi" a cui gli applicano aggettivi. Un nome per il quale aggettivi esprimono valori. Il nome "peso" è un attributo, per cui gli aggettivi "leggeri" e "pesanti" indicano dei valori espressi;
19. verbgroup: raggruppa i verbi che si riferisco allo stesso significato;

20. participle: collega gli aggettivi ai verbi da cui sono derivati;
21. pertainym: viene applicata ad aggettivi relazionali. Aggettivi di questo tipo sono solitamente definiti da frasi come "di o pertinente a" e non hanno antonimi. Possono puntare a un nome o ad un altro aggettivo relazionale;
22. derivation: collega termini che sono tra di loro semanticamente appartenenti a diverse categorie sintattiche ma hanno la stessa forma di radice;
23. domain (dominio): relativa a synset definiti tramite legami di:
    24. a) domain category;
    25. b) domain region;
    26. c) domain usage;
27. member (membro): relativa a synset definiti tramite legami di:
    28. a) domain member category;
    29. b) domain member region;
    30. c) domain member usage.

Risultano 30 per un semplice fatto che la relazione *meronym* e *holonym* non sono da conteggiare, ma vengono prese in considerazione solo le loro sotto-classi. Mentre le relazioni *domain* e *member* oltre alle loro sotto-parti sono loro stesse presenti nel quadro generale di WordNet.

3.3 Weka

Weka (Waikato Environment for Knowledge Analysis) [11] è un software open source rilasciato sotto la licenza GNU (General Public License) nato presso l'università di Waikato in Nuova Zelanda e svilup-

pato completamente in Java quindi utilizzabile su sistemi operativi dotati di un ambiente d'esecuzione Java.

Weka è una collezione di algoritmi di *machine learning* (i.e., apprendimento automatico) per le attività di data mining. Quindi il suo scopo è quello di analizzare automaticamente una grande quantità di dati e decidere quali informazioni sono più rilevanti. Queste informazioni possono essere usate per fare previsioni automatiche o per aiutare gli utenti a prendere decisioni in modo più rapido e preciso.

Contiene strumenti per la pre-elaborazione, la classificazione, la regressione, il clustering, le regole associative, visualizzazione ed estrazione dei dati. E' adatto anche per lo sviluppo di nuovi sistemi d'apprendimento automatico.

Le funzionalità di cui è dotato sono accessibili tramite una semplice interfaccia grafica o richiamabili dal proprio codice Java.

3.3.1 Interfaccia grafica di Weka

L'interfaccia grafica illustrata nella figura 3.3 è molto semplice e si presenta in quattro sezioni:

1. Explorer: permette l'analisi dei dati e l'applicazione di tecniche di Data Mining;

2. Experimenter: 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 (e.g., caricamento di file, applicare i filtri, etc);

4. Simple CLI: concede l'utilizzo di Weka da linea di comando.

Nei sotto-paragrafi seguenti si analizzano più dettagliatamente le varie sezioni appena elencate.

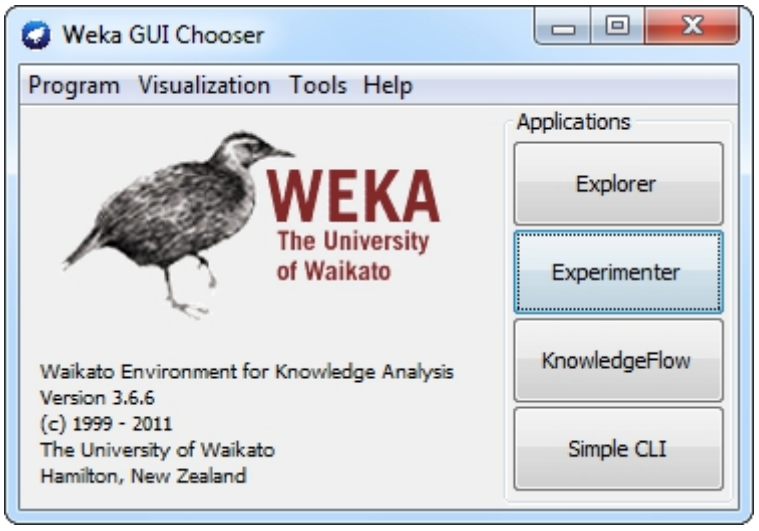


Figura 3.3: Interfaccia grafica Weka

Explorer

Come si può notare in figura 3.4, vi sono presenti diverse schede ognuna delle quali permette di usufruire di alcune funzionalità messe a disposizione dal software:

- Preprocess: selezionamento e modifiche apportabili ai dati su cui si vorranno lavorare;
- Classify: addestramento del sistema per la classificazione ed esecuzione dei test per verificarne la bontà;
- Cluster: insieme di *clusterers* capaci di trovare nel dataset gruppi d'istanze simili;
- Associate: contiene un'implementazione dell'algoritmo *"Apriori"* per imparare le regole di associazione;
- Select attributes: pannello che può essere usato per studiare quali (sottoinsiemi di) attributi sono quelli più predittivi;
- Visualize: visualizzazione molto utile nella pratica: ad esempio aiuta a determinare difficoltà nei problemi di apprendimento. Modalità di visualizzazione dei dati: 1D,2D e si sta lavorando per il 3D.

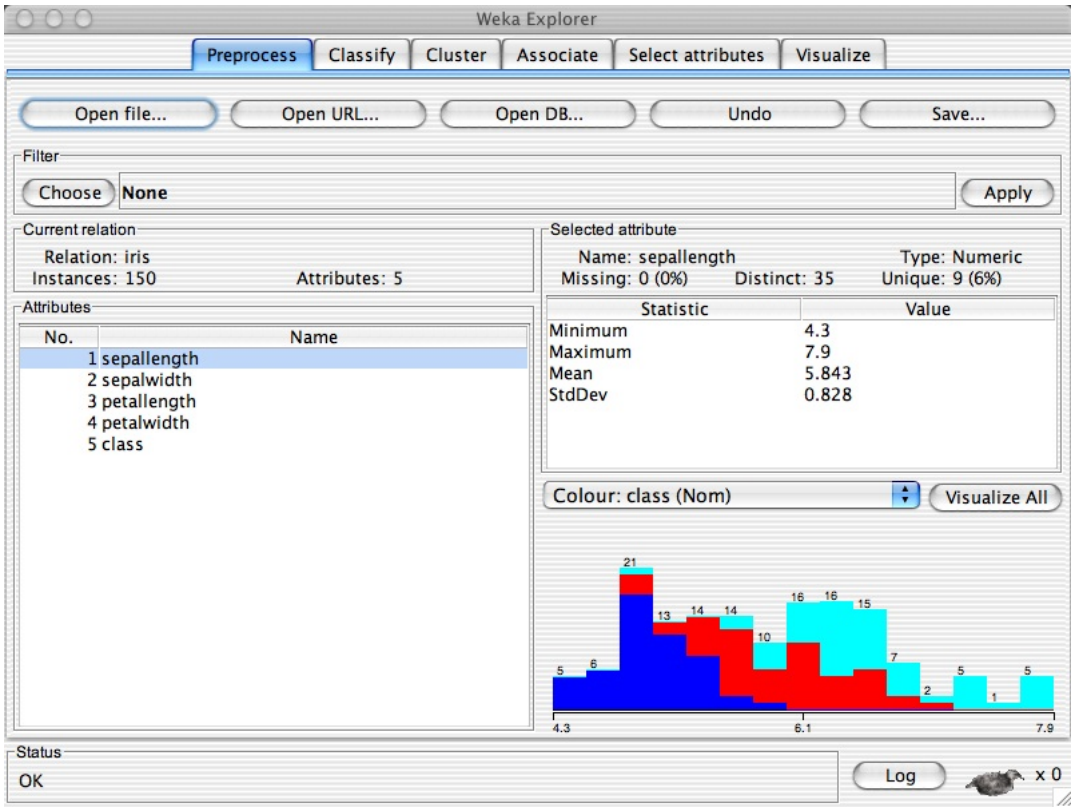


Figura 3.4: Interfaccia grafica Weka - Explorer

Inizialmente l'interfaccia si apre con la scheda di pre-precessamento, nella quale si ha la possibilità di importare i dati da un formato *flat*, ciò significa che ogni dato del dataset deve essere descritto da un numero fisso di attributi (numerici o nominale). I formati disponibili sono:

- ARFF;
- CSV;
- C4.5;
- binary.
- In aggiunta si possono leggere tramite un URL (Uniform Resource Locator) o da un database SQL (Structured Query Language) attraverso l'uso di JDBC (Java DataBase Connectivity).

Nella figura 3.4 è visibile come Weka mostra le informazioni riguardanti gli attributi e nel reparto denominato "filtri" ci sono dei

tools di pre-processamento per la discretizzazione, normalizzazione, ricampionamento, selezione, trasformazione e combinazione degli attributi.

La questione più interessante è relativa alla tematica della classificazione. Questo aspetto è discusso più approfonditamente nei paragrafi più avanti.

Experiment

Rende facile confrontare le prestazioni dei diversi sistemi di apprendimento e aiuta molto l'utente ad analizzare i problemi di classificazione e di regressione. Le opzioni di valutazione sono: *cross-validation*, *learning curve*, *hold-out*.

I risultati possono essere salvati in file o database, dopo di che li si può interrogare. Inoltre è possibile scorrere sulle impostazioni dei diversi parametri. Con questo strumento è possibile parallelizzare il lavoro su più macchine.

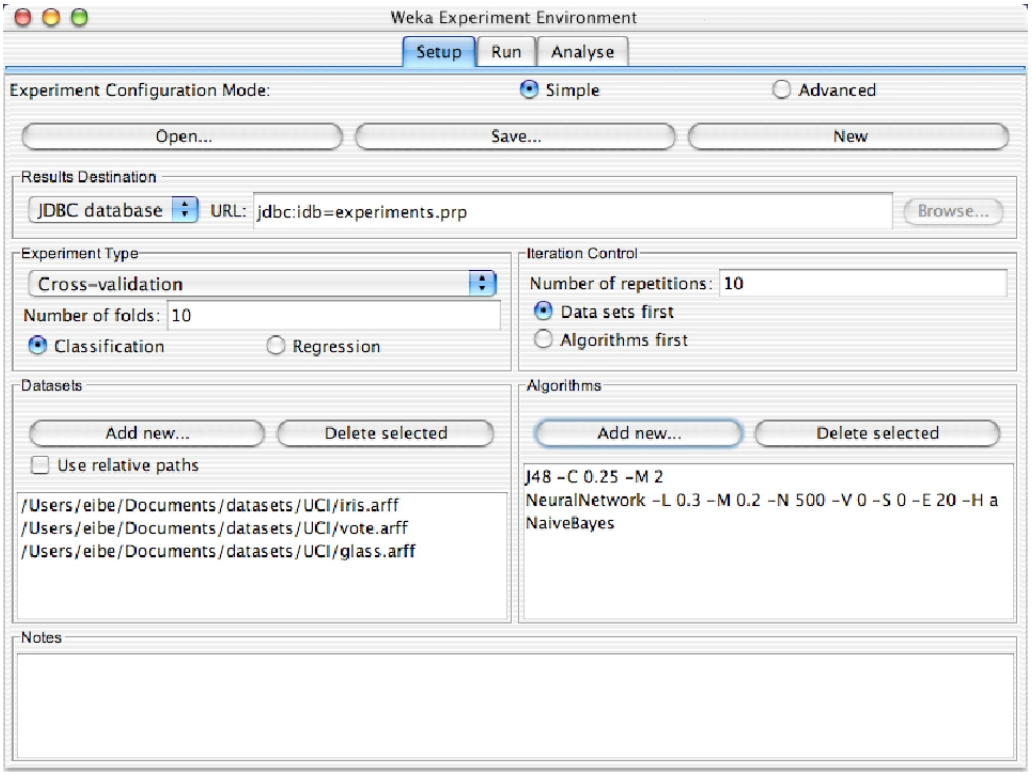


Figura 3.5: Interfaccia grafica Weka - Experiment

Knowledge Flow

Nuova interfaccia grafica basata su *Java-Beans* per la creazione e svolgimento di esperimenti per l'apprendimento automatico. Sorgenti di dati, filtri, algoritmi di classificazione sono detti beans i quali si possono collegare graficamente in un diagramma secondo gli obiettivi che s'intendono realizzare.

Quello che si ottiene sono i cosiddetti *"data-flow"*, cioè flussi di dati. Questi flussi possono essere richiamati da codice Java e vi è anche la possibilità di eseguirli in parallelo (ciascun flusso separato è eseguito in un *thread* separato).

Infine è disponibile l'aggiunta di plugin per consentire un facile inserimento di nuovi componenti per il *Knowledge Flow* e la possibilità di visualizzare le prestazioni dei classificatori durante la fase di elaborazione (elementi di scorrimento per la precisione di classificazione, errore RMS, previsioni, ecc).

Simple CLI

Simple CLI è una semplice shell divisa in due parti: una relativa all'output e l'altra dedicata all'input da riga di comando. Permette l'accesso alle classi Java di Weka come ad esempio filtri e classificatori. Comandi:

- java <classname> [<args>]: invoca una classe Java con gli argomenti dati (se presenti);
- break: interrompe il thread corrente in un modo amichevole, ad esempio un classificatore in esecuzione;
- kill: interrompe il thread corrente in modo ostile;
- cls: cancella l'area d'output;
- exit: esce dalla Simple CLI;

- help [<command>]: se presente, specifica maggiori informazioni sul comando in ingresso, altrimenti in uscita è fornita una panoramica dei comandi disponibili.

3.3.2 Classificazione

Sezione particolarmente importante visto che rappresenta lo "strumento" per l'attività di un classificatore. La scheda *Classify* è suddivisa in:

- classifier: lista degli algoritmi di classificazione divisi per tipologia. E' possibile per ognuno di questi settare i parametri d'esecuzione. Algoritmi:

  1. Bayesian Classifier: classificatori probabilistici basati sul teorema di Bayes;
  2. Trees: classificatori di supporto alle decisioni;
  3. Rules: classificatori che producono regole di decisione per differenziare le diverse classi;
  4. Functions: classificatori definiti su funzioni matematiche;
  5. Lazy classifier: eseguono l'apprendimento del classificatore sullo spazio definito dalle istanze;
  6. Miscellaneous: classificatori non che appartengono a nessuna classe precedente.

- test option: il risultato del classificatore prescelto sarà testato in base alle opzioni impostate facendo clic nella finestra di *TestOptions*. Ci sono quattro modalità di prova:

  1. Utilizzando il training set: il classificatore viene valutato su quanto predice bene la classe delle istanze su cui è stato addestrato;

2. Tramite test set: il classificatore viene valutato su quanto predice bene la classe di un insieme di istanze caricate da file;
3. N-Folds Cross-validation: il classificatore divide il dataset in N parti e fa N esperimenti: in ciascuno si utilizza una delle N parti come test set e tutto quello che rimane come training set e alla fine si calcola la media delle accuratezze ottenute. C'è un opzione che ti permette di decidere il *seed* da utilizzare per la divisione casuale (se non viene impostato, i dati sono divisi sempre nello stesso modo);
4. Percentage split: il classificatore divide il dataset prendendo una certa percentuale per il training set e l'altra per il test set. La quantità dei dati è decisa dall'utente impostando il campo "%".

- classifier output: area utilizzata per la visualizzazione dei risultati forniti dai test di classificazione. L'output è diviso in diverse sezioni (visibili nella figura 3.6):

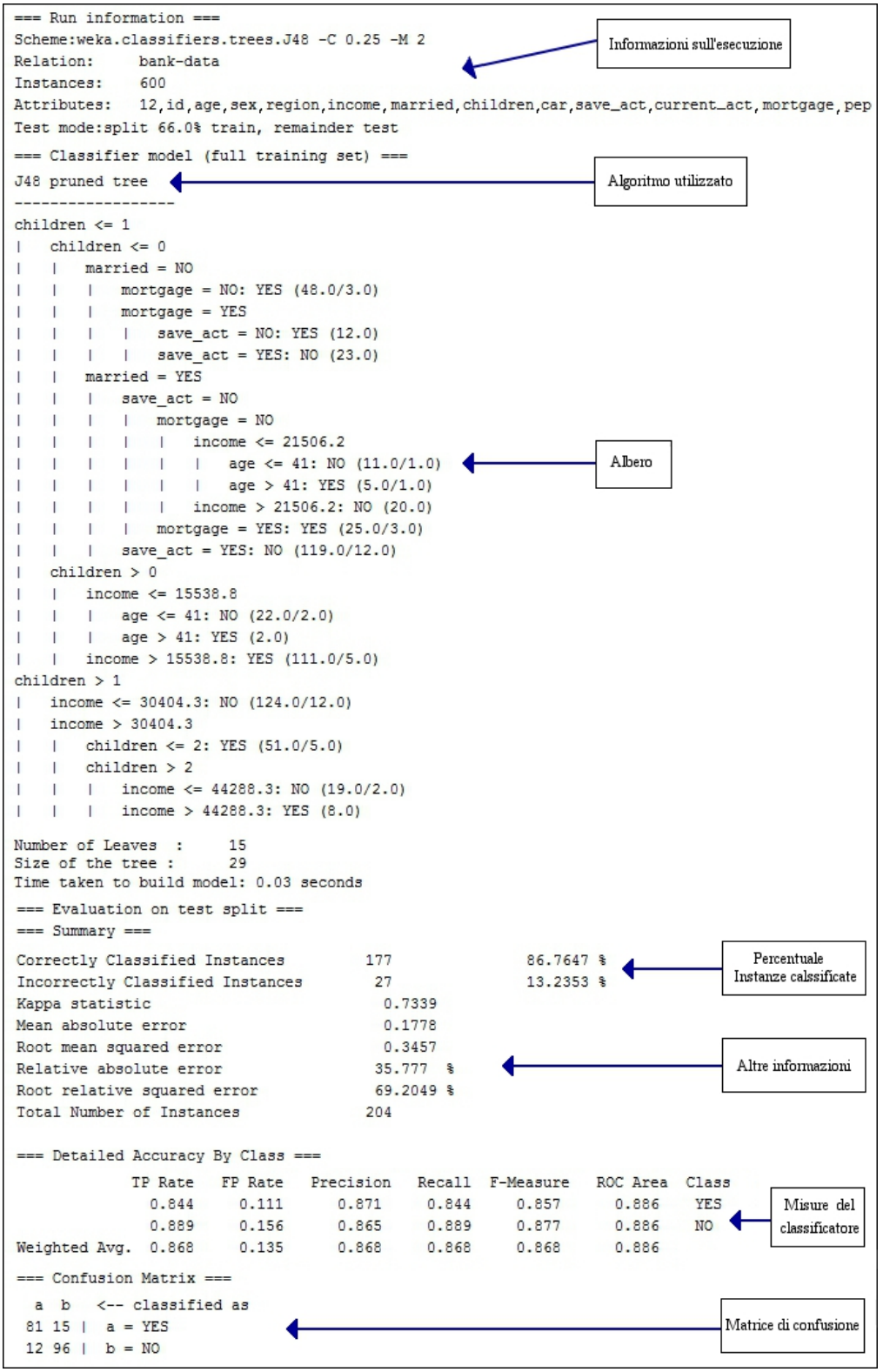
  1. Informazioni dell'esecuzione: una lista di informazioni che illustrano le opzioni dello schema di apprendimento, il nome della relazione, le istanze, gli attributi e le modalità di test che sono stati coinvolti nel processo;
  2. Modello di classificazione: descrive l'albero e le regole di associazione create, cioè una rappresentazione testuale del modello di classificazione che è stato realizzato sull'intero training set;
  3. I risultati della modalità di test scelto sono così ripartiti:
     - Riepilogo: un elenco di statistiche che riassumono il grado di precisione delle istanze classificate in modo corretto e non, che il classificatore è stato in grado di prevedere sotto la modalità di test scelta;

  - Dettagli: visualizzazione delle informazioni riguardanti alcune misure relative alla predizione del classificatore;
  - Matrice di confusione: descrive in modo semplice i risultati ottenuti dall'esecuzione dell'esperimento: le righe rappresentano le classi *actual* (reale) mentre le colonne descrivono le predizioni. Un dato molto rilevante è la diagonale principale dove sono descritte le predizioni eseguite correttamente e dette $TruePositive$ mentre nelle rimanenti celle sono segnalate le predizioni errate. In sostanza ci indica la distribuzione delle previsioni corrette ed errate;
  - Codice sorgente (opzionale): se è stata scelta l'opzione "codice sorgente di uscita" nel riquadro di dialogo "Altre opzioni", in questa sezione è riportato il del codice sorgente Java del classificatore.

- result list: elenco che contiene tutte le esecuzioni degli algoritmi utilizzati per la classificazione e consente di salvarle l'output testuale generato in semplice formato di testo, oppure di salvare un oggetto del modello in un file binario. Gli oggetti sono salvati in Java come *oggetti serializzati*. In più si ha anche la possibilità di:

  - Visualizzare gli errori di classificazione: compare una finestra di visualizzazione che traccia i risultati della classificazione. Casi correttamente classificati sono rappresentati da croci, mentre quelli erroneamente classificati appaiono come quadrati.
  - Visualizzare l'albero: mostra una rappresentazione grafica della struttura del modello del classificatore.

Nella figura sottostante viene illustrato un esempio di un risultato ottenuto dopo l'esecuzione di una classificazione.




```
=== Run information ===
Scheme:weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     bank-data
Instances:    600
Attributes:   12,id,age,sex,region,income,married,children,car,save_act,current_act,mortgage,pep
Test mode:split 66.0% train, remainder test

=== Classifier model (full training set) ===

J48 pruned tree
------------------

children <= 1
|   children <= 0
|   |   married = NO
|   |   |   mortgage = NO: YES (48.0/3.0)
|   |   |   mortgage = YES
|   |   |   |   save_act = NO: YES (12.0)
|   |   |   |   save_act = YES: NO (23.0)
|   |   married = YES
|   |   |   save_act = NO
|   |   |   |   mortgage = NO
|   |   |   |   |   income <= 21506.2
|   |   |   |   |   |   age <= 41: NO (11.0/1.0)
|   |   |   |   |   |   age > 41: YES (5.0/1.0)
|   |   |   |   |   income > 21506.2: NO (20.0)
|   |   |   |   mortgage = YES: YES (25.0/3.0)
|   |   |   save_act = YES: NO (119.0/12.0)
|   children > 0
|   |   income <= 15538.8
|   |   |   age <= 41: NO (22.0/2.0)
|   |   |   age > 41: YES (2.0)
|   |   income > 15538.8: YES (111.0/5.0)
children > 1
|   income <= 30404.3: NO (124.0/12.0)
|   income > 30404.3
|   |   children <= 2: YES (51.0/5.0)
|   |   children > 2
|   |   |   income <= 44288.3: NO (19.0/2.0)
|   |   |   income > 44288.3: YES (8.0)

Number of Leaves  :     15
Size of the tree :      29
Time taken to build model: 0.03 seconds

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances         177               86.7647 %
Incorrectly Classified Instances        27               13.2353 %
Kappa statistic                          0.7339
Mean absolute error                      0.1778
Root mean squared error                  0.3457
Relative absolute error                 35.777  %
Root relative squared error             69.2049 %
Total Number of Instances              204

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.844     0.111      0.871     0.844     0.857      0.886    YES
                 0.889     0.156      0.865     0.889     0.877      0.886    NO
Weighted Avg.    0.868     0.135      0.868     0.868     0.868      0.886

=== Confusion Matrix ===

  a  b   <-- classified as
 81 15 |  a = YES
 12 96 |  b = NO
```

Figura 3.6: Interfaccia grafica Weka - Scheda Classify, area di visualizzazione dei risultati

Qui di seguito si analizzano maggiormente le misure riguardanti il classificatore e visibili nella figura 3.6:

- Nelle prime due colonne si ha i *True Positive* (TP) e *False Positive* (FP) calcolati attraverso la matrice di confusione. TP è la frequenza della proporzione delle istanze che sono state classificate come classe X, fra quelle che hanno veramente la classe X. Invece FP è la frequenza della proporzione delle istanze che sono state classificate come classe X ma appartengono ad una classe diversa, tra tutte quelle che non appartengono a X. Calcolati matematicamente basta applicare le formule:

$$TP_{yes} = \frac{81}{81+15} = 0.844$$

$$TP_{no} = \frac{96}{96+12} = 0.889$$

$$FP_{yes} = \frac{12}{96+12} = 0.111$$

$$FP_{no} = \frac{15}{81+15} = 0.156$$

- Precision è la proporzione delle istanze che hanno veramente la classe X fra tutte quelle che sono state classificate nella classe X;

- Recall è la proporzione di istanze che sono state classificate come classe X, fra tutte le istanze che hanno veramente classe X. E' equivalente alla True Positive.

- F-measure media armonica tra *precision* e *recall*, questa tende al valore più basso tra i due, quindi più questo valore è alto migliore è la classificazione. E' calcolata secondo la seguente formula:

$$F - measure = 2 * \frac{precision * recall}{precision + recall}$$

- ROC-Area: rappresenta l'area sotto la ROC(Receiver operating characteristic), ed è calcolata esaminando tutto il range dei valori. Se tale valore è pari a 0.5 significa che la bontà del classificatore è uguale a quella del lancio di una moneta.

3.3.3 Utilizzo di Weka da codice Java

Il potere della manipolazione dei dati con Weka può anche essere sfruttato direttamente da codice Java. Ciò consente lo sviluppo di applicazioni di data mining (per i sistemi di supporto alle decisioni) senza scrivere alcun codice *machine learning*. Classi di base sono:

- Attribute: è la classe per la gestione degli attributi. Sono supportati quattro tipi: Numeric, Nominal (un fissato insieme di valori), String e Date. Esempio della definizione di un attributo Nominal:

  Attribute temperatura = new Attribute("Temperatura");

- Instance: è la classe per la gestione di una singola istanza. Creazione di un'istanza vuota con tre attributi:

  Instance inst = new Instance(3);

- Instances: è la classe per la gestione di un set di istanze. Può essere configurato attraverso l'importazione di un file $arff$:

  FileReader reader = new FileReader("myDataset.arff");

  Instances set = new Instances(reader);

  Mentre per impostare l'attributo di classe basta scrivere:

  int classAttributeIndex = set.numAttributes() - 1;

  set.setClassIndex (classAttributeIndex);

- Classifier: è una classe astratta che deve essere implementata specificando l'algoritmo che si vuole utilizzare, quindi al suo interno abbiamo il modello del nostro sistema. Un esempio di creazione e inizializzazione di un classificatore J48 è:

  Classifier myClassifier = new J48();

  myClassifier.buildClassifier(set);

  Poi per classificare una sola istanza:

  double class = myClassifier.classifyInstance(inst);

Per usufruire delle funzionalità di Weka da codice Java bisogna per prima cosa scaricarsi il "jar" di Weka attraverso la pagina web `http://www.cs.waikato.ac.nz/ml/weka/`, dopo di che in Eclipse cliccare su Project $\longrightarrow$ Properties, andare nella sezione "Libraries" nella scheda "Java Build Path" e cliccare su "Add JARs.." o "Add External JARs.." a seconda se il "jar" di Weka da inserire si trova rispettivamente all'interno o all'esterno del progetto.

Capitolo 4

Classificazione semantica di documenti in categorie

In questi ultimi tempi molti ricercatori stanno tentando di automatizzare il processo di classificazione per poter fronteggiare l'eccessivo aumento dei dati testuali. L'obiettivo di questa tesi, come indicato anche nel titolo, è di sviluppare nuove tecniche di Text Mining per la classificazione semantica dei documenti. Attualmente, per assegnare documenti ad un preciso argomento, l'idea di base è rimasta quella di rappresentare i documenti tramite l'approccio definito *bag of words*.

La nuova proposta definita all'interno di questa tesi tenta di classificare documenti all'interno di categorie tramite il loro confronto basandosi sulle relazioni semantiche presenti fra le parole contenute nei documenti. Questo metodo viene utilizzato con l'intento di riuscire a generare delle regole che aiutino la classificazione.

Per affermare che un determinato documento può essere assegnato ad una specifica categoria, si potrebbe pensare di confrontarlo con altri documenti che sono appartenenti a tale categoria.

Tuttavia, questo approccio richiederebbe un consistente numero di confronti per avere risultati adeguatamente accurati. Si può superare questo problema sviluppando un metodo che rappresenti una categoria (insieme di documenti) attraverso un unico documento.

Per rendere una categoria paragonabile con i documenti, dobbiamo

estrarre da essa un prototipo, cioè generare un documento astratto che contenga i termini più rappresentativi per la categoria.

Visto l'elevata complessità di questo problema, il processo è diviso in varie fasi dove ognuna di queste consente l'analisi delle risorse computazionali e temporali utilizzate. La suddivisione è eseguita assicurandosi che le diverse parti rimangano distinte e grazie ad un modello a spirale è possibile il costante ed incrementale raffinamento. Si ha così che, ad ogni risultato prodotto dei test effettuati, è importante rivedere gli step del processo per ricercare eventuali errori e trarre precise conclusioni in modo tale da proporre nuove soluzioni.

Fasi del processo:

1. Selezione dati: selezionamento di categorie e documenti;

2. Filtraggio dati: settaggio di vincoli che eliminano dati che non si intendono utilizzare;

3. Pre-processing: fase molto importante del processo perché si convertono i dati acquisiti sotto uno specifico standard;

4. Creazione nuove informazioni: si aggiungono nuove variabili attraverso le proprietà e le misure disponibili;

5. Generazione training e test set: generazione dei dati utilizzati per l'addestramento e il test del classificatore;

6. Classificazione: applicazione degli algoritmi di Data Mining.

Riportiamo di seguito i software e gli strumenti adoperati per effettuare tutte le fasi del processo:

- PostgreSQL: è un sistema database relazionale ad oggetti, rilasciato con licenza *open-source*. E' utilizzato per creare un grande numero di tabelle, che in alcuni casi racchiudono un elevato numero di record. Per ottimizzare la gestione dei tempi, cioè ottenere

maggiore velocità nell'accedere ai dati, sono stati adoperati gli indici di tipo B-tree e Hash. Per la creazione, selezione, eliminazione, l'inserimento e l'aggiornamento dei dati presenti nelle tabelle sono state realizzate generiche funzioni riutilizzabili e facilmente aggiornabili grazie all'utilizzo di parametri in ingresso;

- DMOZ: già analizzato nei capitoli precedenti. Impiegato per l'acquisizione dei documenti;

- Navicat Premium: è uno strumento di amministrazione per connessioni a più database, consente di connettersi simultaneamente, all'interno di una singola applicazione, a SQL Server, MySQL, SQLite, PostgreSQL e Oracle. Questo rende l'amministrazione di più tipi di database molto più facile. Come vedremo più avanti, questo software sarà utile per eseguire l'importazione di tabelle presenti nel formato ".mdb" a quello di PostgreSQL.

- WordNet: già analizzato nei capitoli precedenti. Viene adoperato per scoprire le relazioni esistenti tra diverse parole;

- Weka: già analizzato nei capitoli precedenti. E' impiegato per l'esecuzione di algoritmi di Data Mining;

- Eclipse: Integrated Development Environment (IDE), cioè un ambiente di sviluppo integrato multi-linguaggio e multipiattaforma.

I linguaggi adoperati per lo svolgimento della tesi sono:

- Structured Query Language (SQL): è un linguaggio usato per interrogare i database. Infatti è progettato per leggere, scrivere e gestire dati memorizzati in un sistema basato sul modello relazionale (RDBMS). Le interrogazioni vengono eseguite tramite dei costrutti chiamati query (interrogazioni).';

- SQL Procedural Language (PL/pgSQL): è un linguaggio procedurale definito per il sistema di database PostgreSQL. Aggiunge

strutture di controllo al linguaggio SQL, in modo da eseguire calcoli più complessi. E' facile da usare, e la cosa più importante è che permette di creare funzioni e procedure di trigger;

- Java: linguaggio di programmazione orientato agli oggetti.

I dispositivi hardware adottati:

- AMD Opteron (tm) Octa-core 6128: 2 GHz con 16 GB di RAM e sistema operativo Linux Ubuntu 10.04.
- Intel Core Dual-core T9300: 2.5 GHz con 4 GB di RAM e sistema operativo Windows 7.

4.1 Selezione dati

Le sorgenti documentali utilizzati per la collezione dei testi utili sono recuperati da DMOZ e WebClassIII [3], entrambi in versione inglese. La scelta di ricorrere a tale lingua sta nel fatto che le informazioni disponibili sono più complete e in maggiore quantità, soprattutto perché DMOZ è nato proprio per questa lingua.

In tutti i test effettuati vi è presente una struttura gerarchica delle categorie utilizzate, avendo così le seguenti tipologie di relazioni semantiche:

1. Similar (i.e., simili): il documento appartiene alla stessa categoria di quella riferita dal documento-prototipo;
2. Dissimilar unrelated (i.e., dissimili imparentati): il documento appartiene ad una categoria diversa da quella riferita dal documento-prototipo;
3. Dissimilar related (i.e., dissimili parentati): il documento appartiene ad una categoria è discendente da quella riferita dal documento-prototipo. Si possono distinguere nelle seguenti tipologie:

- Dissimilar related hyponym (i.e., dissimili parentati iponime): dove il documento D_1 appartiene ad una categoria figlia di quella rappresentata dal documento-prototipo D_2;
- Dissimilar related hypernym (i.e., dissimili parentati iperonime): dove il documento D_1 appartiene ad una categoria padre di quella rappresentata dal documento-prototipo D_2.

4.1.1 Dataset DMOZ

Quello che si è fatto è l'importazione dei dati all'interno del sistema informativo su cui si è eseguito il processo di Text Mining. Per i test effettuati su DMOZ si sono scaricati, tramite un web crawler, le pagine web presenti nei link delle tabelle di tale strumento. Il dataset, attraverso un filtraggio, contiene 11.644 documenti in 245 categorie organizzate in una gerarchia specificata nella tabella 4.1.

Livello	**Categorie**	**Documenti**
Primo	3	156
Secondo	22	932
Terzo	150	7.505
Quarto	64	2.868
Quinto	4	111
Sesto	2	72

Tabella 4.1: Dataset DMOZ

Questo dataset da adesso in avanti lo si definisce "Dataset DMOZ".

4.1.2 Dataset WebClassIII

Per quel che riguarda WebClassIII i dati, sono stati scaricati dal sito Web, i documenti riferiti al test DMOZ ed a quello Yahoo.Il primo, che da adesso in avanti definiamo "Dataset DMOZ Ceci", è stato estratto nel 2004 da DMOZ e contiene tutti i documenti Web presenti ai primi cinque livelli della directory Web

"Top/Health/Conditions_and_Diseases/". Non sono stati considerati quelli contenenti solo gli script e quelli vuoti. Il dataset ottenuto contiene 5.612 documenti in 221 categorie organizzate in una gerarchia specificata nella tabella 4.2.

Livello	**Categorie**	**Documenti**
Primo	21	340
Secondo	81	1.514
Terzo	85	2.604
Quarto	32	1.099
Quinto	2	55

Tabella 4.2: Dataset DMOZ usato da Ceci e Malerba

Il secondo, che da adesso in avanti definiamo "Dataset Yahoo Ceci", è stato estratto nel 2003 da Yahoo! Directory e contiene tutti i documenti Web presenti ai primi tre livelli della directory "http://dir.yahoo.com/Science". Anche in questo caso non sono stati considerati quelli contenenti solo gli script e quelli vuoti. Il dataset ottenuto contiene 907 documenti in 68 categorie organizzate in una gerarchia definita nella tabella 4.3.

Livello	**Categorie**	**Documenti**
Primo	6	98
Secondo	27	352
Terzo	35	457

Tabella 4.3: Dataset Yahoo usato da Ceci e Malerba

4.2 Trasformazione database usati da WebClassIII

Questa fase viene applicata solo per i dati relativi a WebClassIII perché per poter associare ogni documento alla categoria corretta e per adoperare le stesse *stopwords* adoperate nei test fatti da Ceci e Malerba,

si sono dovute analizzare le tabelle presenti nei database scaricabili sempre dal sito [3].

4.2.1 Importazione database con Navicat Premium

Al fine di utilizzare il database PostgreSQL descritto precedentemente è stato necessario effettuare una conversione da un database Microsoft Access. Questa operazione ha previsto la conversione automatica del database attraverso il programma Navicat Premium (figura 4.1). Il risultato di questa esecuzione si è dimostrato molto valido, infatti non si sono riscontrati problemi di perdita di informazioni e, come descritto sotto, i passaggi sono molto semplici.

Per prima cosa, una volta aperto il programma, bisogna creare una nuova connessione al database PostGres in cui si vogliono inserire i nuovi dati. Per far ciò, cliccare su File $\longrightarrow$ New Connection $\longrightarrow$ PostgreSQL e nella finestra che compare, inserire le informazioni necessarie (e.g., HostName/IP Address, Port, User Name, Password) per concludere tale operazione.

Non rimane altro da fare che spostarsi nello *schema* opportuno, cliccare sul pulsante *Import Wizard* e seguire i passi:

1. Selezionare dalla lista "MS Access Database (*.mdb,*.accdb)";
2. Selezionare il file contenente il database da trasformare. Vi compariranno una lista di tutte le tabelle che sono incluse nel file. Scegliere quelle di interesse;
3. Decidere i nomi delle tabelle di destinazione. Vi è anche l'opportunità di creare o meno una nuova tabella per ognuna di esse;
4. Controllare che ogni campo di ogni tabella sia definito in modo corretto;
5. Determinare quale sia la miglior modalità di importazione che si intende eseguire tra:

- Append: aggiunge i record alla tabella della destinazione;
- Update: aggiorna i record di destinazione con i record corrispondenti alla sorgente;
- Append/Update: se nella destinazione il record esiste allora lo si aggiorna. Altrimenti lo si aggiunge;
- Delete: cancella i record della destinazione che corrispondono con quelli della sorgente;
- Copy: cancella tutti i record della destinazione e la ripopola con quelli della sorgente.

6. Iniziare il trasferimento.

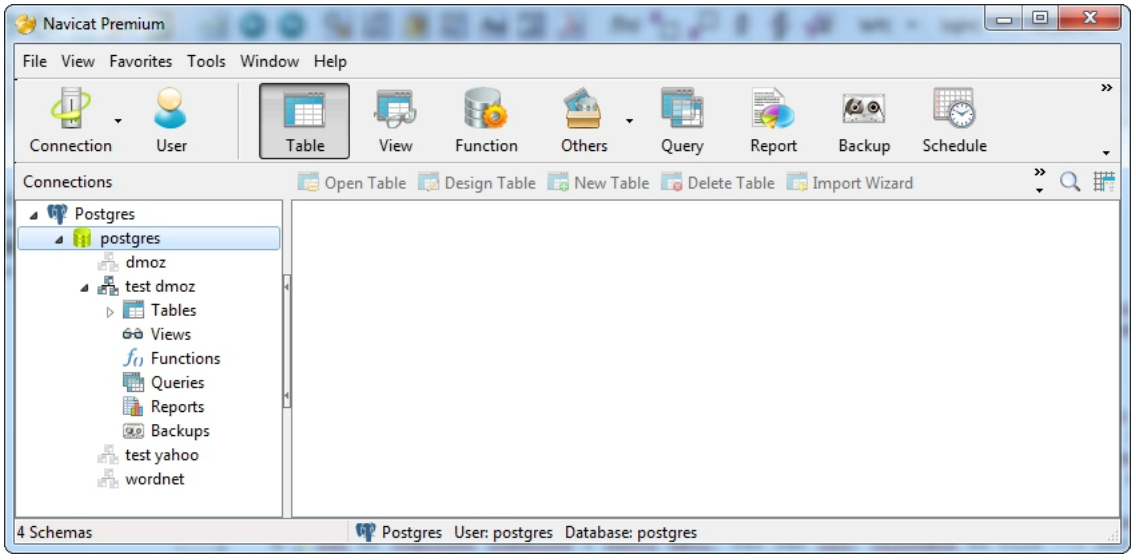


Figura 4.1: Programma Navicat Premium

4.2.2 Selezione tabelle

Non tutte le tabelle presenti nei database utilizzati da WebClassIII sono da importare. Infatti solo quelle elencate qui sotto servono per effettuare la corretta trasformazione:

- Appartenenza: rappresenta l'associazione tra i documenti e le categorie;

- Categoria: contiene il nome e il padre di ogni categoria;
- Stopword: parole che non devono essere considerate in fase di analisi dei documenti.

4.3 Filtraggio dati

Come detto in precedenza Ceci e Malerba consentono il download dei file da loro utilizzati. Ma osservando bene i dati messi a disposizione ci si rende conto che superano le reali caratteristiche adoperate. Infatti per il dataset Yahoo viene specificato che il totale del numero di documenti utilizzati è 907, mentre la cartella scaricabile ne contiene 909. Stessa cosa per il test DMOZ che contiene 5.916 documenti invece di 5.612. Quindi per poter utilizzare quelli corretti, si deve verificare il campo "URL" della tabella "documento" presente nel database ".mdb" fornito.

Invece il dataset estratto direttamente da DMOZ, cioè non quello preso da WebClassIII, ha maggiori problematiche perché il catalogo internazionale comprende un grande quantità di dati e, anche se molto stimato e di buona reputazione, potrebbe contenere qualche documento scritto in lingua diversa da quella inglese.

Per assicurarsi di rendere efficace l'esecuzione del classificatore e per evitare di raccogliere documenti scritti in diverse lingue, è stata fatta un'accurata analisi dell'organizzazione di tutte le categorie. Per cui per prima cosa si sono identificate le aree tematiche principali, i cosidetti "ancestor":

- "Top/Arts";
- "Top/Business";
- "Top/Computers";
- "Top/Games";

- "Top/Health";
- "Top/Home";
- "Top/Kids_and_Teens";
- "Top/News";
- "Top/Recreation";
- "Top/Reference";
- "Top/Regional";
- "Top/Science";
- "Top/Shopping";
- "Top/Society";
- "Top/Sports";
- "Top/World".

Dopo di che si nota immediatamente che vi sono due nodi che sicuramente hanno al loro interno documenti eterogenei linguisticamente: "Top/Regional" e "Top/World". Infatti le categorie disposte ai livelli inferiori di queste sezioni contengono nell'attributo "Topic" nomi di Continenti e Stati, il che significa che i loro contenuti sono tipicamente scritti in lingua originale. Quindi il primo filtro applicato è scartare i nodi di queste due ultime categorie elencate.

Facendo una seconda analisi più approfondita si è notato che alcune categorie hanno al loro interno trattazioni con riferimenti locali pertinenti all'argomento di quella categoria. Per capire meglio questo importante discorso viene presentato un esempio: guardando il "Topic" della categoria "Top/Science/Environment/Biodiversity/Regional" ci si accorge che il padre "Top/Science/Environment/Biodiversity", ha un ancestor "Science", e quindi non scartata dal filtro precedente, ma

nella sua sottocategoria "Regional" comprende documenti che trattano dello stesso argomento ma di diversa tipologia linguistica.

Dunque il primo filtraggio è necessario, ma non sufficiente per eliminare il problema. Per poterlo risolvere, si adotta un secondo filtro (sempre per quel che riguarda la selezione delle categorie), agendo sull'attributo 'Topic" fornito da DMOZ. Infatti non si considerano le categorie che contengono la parola "Regional" nel proprio "Topic".

Per sicurezza la stessa analisi è stata fatta anche per "World" ma come ci si aspettava non si sono riscontrate categorie che includono tale parola nel "Topic", fatta eccezione per la categoria "Top/World".

Altre operazioni eseguite prima di caricare le pagine sul database sono:

- Parsing;
- Eliminazione dei "tag";
- Eliminazione di punteggiatura e caratteri speciali;
- Eliminazione di numeri e parole non rilevanti per gli obiettivi predisposti. Esempio: parole contenenti un basso numero di lettere.

Questa attività esegue un filtraggio che consente un notevole risparmio di memoria e un'importante riduzione dei dati considerati come "rumore" (e.g., termini maggiormente ricorrenti presenti nel linguaggio naturale che sono poco significativi). Ecco perché si applica dalle pagine la rimozione delle liste nominate *stop words*, le quali contengono le parole più usate in un determinato linguaggio (e.g., articoli, congiunzioni, ecc..).

Come già detto nel capitolo 3.1, DMOZ dispone di oltre 750.000 categorie posizionate su una struttura gerarchica ad albero avente ben 15 livelli. E' riportata questa osservazione perché c'è da tenere presente che per ottenere un buon classificatore bisogna selezionare sia una grande quantità sia un'elevata qualità dei dati. Con qualità, si

assume che la distribuzione dei documenti presenti nelle categorie sia omogenea e uniforme. In altre parole, ogni categoria deve comprendere un adatto numero di documenti per fare in modo che il classificatore possa avere abbastanza informazioni per riconoscere se un documento appartiene o meno alla categoria.

Sicuramente l'aspetto della quantità è soddisfatto pienamente, mentre per quel che riguarda la qualità bisogna fare una piccola indagine. Non prendendo in considerazione il livello "0" (nodo della radice), si può constatare che a partire dal livello 5 il valor medio di documenti è già molto basso e man mano che si scende nell'albero tale valore decresce.

Questo dato è spiegato dal fatto che i documenti di più alto livello trattano degli argomenti di maggiore specificità, mentre quelli di più basso livello sono riferiti a concetti specifici. Detto ciò, i livelli disponibili su cui devono essere selezionate le categorie sono stati limitati tra 1 e 6.

Infine, come ultimo studio, si è condotto anche per i documenti lo stesso ragionamento appena descritto per le categorie. L'unica differenza tra le due analisi, sta nel fatto che l'aspetto relativo alla qualità viene espresso attraverso il numero di parole presenti all'interno del documento.

La prima cosa che si può pensare è che utilizzando più parole la classificazione potrebbe migliorare, ma questo non è assolutamente vero visto che il testo presenta sempre parole comuni che sono poco importanti. Al contrario, prendendo solo poche parole (e.g., 5,10,15) non si hanno abbastanza informazioni per una corretta interpretazione del documento. L'ultima operazione di questa fase limita il numero minimo di termini necessari per la selezione dei documenti.

Concluse queste azioni si ottengono dei dati simili a quelli raffigurati nella tabella 4.4 dove:

- "docid": identificativo numerico relativo al documento;

- "wordnumber": contatore che precisa la posizione della parola nel documento, in base al campo "abs_freq";
- "word": parola contenuta nel documento;
- "abs_freq": numero di occorrenze della parola nel documento (frequenza assoluta). Nel paragrafo successivo viene indicato come sarà aggiornata questa variabile.

docid	**wordnumber**	**word**	**abs_freq**
100	1	webmaster	9
100	2	world	4
...	...	...	...
101	1	real	22
...	...	...	...

Tabella 4.4: Fase di filtraggio

4.4 Pre-processing

I testi scaricati sono generalmente scritti seguendo delle precise regole che prevedono l'inserimento della punteggiatura e di parole che contengono caratteri maiuscoli. Mentre per il primo problema basta ricercare i simboli non desiderati (e.g., virgola,due punti, ...) e applicare una semplice rimozione (già eseguita nella fase precedente), per la seconda questione si ha una situazione più complicata ma sempre risolubile.

Il problema che si intende risolvere è conosciuto come "frammentazione dell'informazioni" ed è derivato dal fatto che una stessa parola può essere scritta in più forme, dove ad ognuna di queste sono state associate diverse informazioni (come riportato nella tabella 4.4). Esempio: il vocabolo "car" potrebbe essere scritto in modi diversi: "CAR" o "Car". Per il lettore queste rappresentazioni hanno lo stesso significato ma per l'elaboratore le lettere "C" e "c" sono due caratteri diversi in termini di codice "ASCII".

A seguito di questa questione, è stato necessario creare un criterio in grado di unificare le moltiplicità dei dati appartenenti alla stessa parola allo scopo di ottenere una maggiore coerenza e quindi correttezza delle informazioni acquisite.

Quindi le possibilità sono due: o ricondurre tutte i vocaboli alla forma base "CAR" cioè tutto maiuscolo oppure alla forma "car" cioè tutto minuscolo. Visto che nel database WordNet le parole sono definite tutte minuscole, la scelta è caduta sulla seconda forma. A questo punto si ha che più record contengono lo stesso termine quindi per unificare le altre variabili (tabella 4.4) si è deciso:

- "wordnumber": scelto come il minimo tra i "wordnumber" di tutte le rappresentazioni;
- "abs_freq": scelta come somma delle "abs_freq" di tutte le rappresentazioni.

In termini matematici si ha che, dati:

- Un documento D_z;
- Due parole $w_{z,i}\,, w_{z,j} \in D_z$;
- Una funzione $lower(w_{z,i}\,)$ la quale restitusce una stringa con caratteri tutti minuscoli della parola in ingresso. Applica la fase nominata "casefolding" (e.g., $lower(''CaR'') ='' car''$);
- ${w_z}^{\text{down}} = lower(w_{z,i}\,) = lower(w_{z,j}\,)$.

Le variabili precedentemente descritte si calcolano:

- $\text{abs\_freq}({w_z}^{\text{down}}) = \text{abs\_freq}(w_{z,i}) + \text{abs\_freq}(w_{z,j})$;
- $\text{wordnumber}({w_z}^{\text{down}})=\min(\text{wordnumber}(w_{z,_i}),\text{wordnumber}(w_{z,j}))$

Per comprendere meglio vengono riportate nelle tabelle 4.5 i passaggi che sono eseguiti per il *casefolding*: eliminazione di record (quindi

docid	wordnumber	word	abs_freq
100	7	webmaster	9
100	13	WEBMASTER	4
...	...	...	...

⇓

docid	wordnumber	word	abs_freq
100	7	webmaster	13
100	7	WEBMASTER	13
...	...	...	...

⇓

docid	wordnumber	word	abs_freq
100	7	webmaster	13
...	...	...	...

Tabella 4.5: Tabelle create per la fase di casefolding

abbassamento del numero di vocaboli) e l'incremento del frequenze associate alle parole che avevano più rappresentazioni.

A questo punto, abbiamo ridotto il testo ad una lista di parole significative, ma per poterle confrontare tra loro, bisogna ridurre i termini derivati alla loro forma base. Questo include:

- Trasformazioni dei sostantivi in forma plurale, nella corrispondente forma singolare;

- Trasformazioni dei verbi coniugati nella forma base.

Questa ricostruzione, è svolta attraverso la conoscenza delle regole grammaticali generali (e.g., l'aggiunta di -es ai sostantivi per formare il loro plurale) e le forme irregolari (e.g., il plurale di "mouse" è "mice"). WordNet ci aiuta solo per quest'ultima classe perché contiene molte di queste forme, mentre per la prima regola sono stati adoperati i casi descritti nelle tabelle 4.6 derivati dalla lingua inglese. In più, nel caso di nomi regolari terminanti in "s" si procede con il controllo della

presenza in WordNet del vocabolo privato della lettera "s". Possiamo capitare in due casi:

- Termine presente: la modifica viene applicata;
- Termine assente: la modifica non viene applicare.

Esempi:

- Parola "cars" privata della "s" da luogo a "car", la quale è contenuta in WordNet e quindi si procede con la sostituzione di "cars" con "car";
- Parola "kiss" privata della "s" da luogo a "kis", la quale non è contenuta in WordNet e quindi non si procede con la sostituzione.

Regola	**Singolare**	**Plurale**
I nomi che terminano per: "-s", "-ss", "-ch", "-sh", "-x" aggiungono "-es".	Touch (toccato) Kiss (baciato)	Touches Kisses
I nomi terminanti in "-y"; eccetto quando la "-y" è preceduta da vocale, si trasformano in "-ies".	Lady (signora) Boy (ragazzo)	Ladies Boys
I nomi terminanti in "-o" preceduta da vocale; oppure se sono parole di origine straniera; oppure se sono parole abbreviate, aggiungono la "-s".	Video Canto Piano	Videos Cantos Pianos
I nome terminanti in "-f" o "-fe", si trasformano "-ves". Eccezioni:	Life (vita) Knife (coltello) Roof (tetto)	Lives Knives Roofs

Tabella 4.6: Regole grammaticali

Applicando queste semplici modifiche si ottiene un immenso vantaggio: infatti permettono di raggruppare ulteriormente i vocaboli riducendone il numero, ma senza perderne il contenuto semantico. Come per il caso del *casefolding*, viene eseguito l'aggiornamento delle variabili "wordnumber" e "abs_freq" per le parole che sono presenti sia in forma singolare che plurale.

4.5 Frequenze dei termini e generazione coppie

Questa è la fase più importante del processo perché comprende la realizzazione e creazione di nuove variabili che caratterizzano i successivi passi.

Fino ad ora si è lavorato con valori assoluti (i.e., "abs_freq" frequenza assoluta) che non sono sufficientemente adatti per gli obiettivi prefissati perché non si adeguono alle diverse lunghezze dei documenti (i.e., quantità totale di parole che formano il documento). Infatti, come prima operazione, è stato necessario sostituire queste variabili con le rispettive misure relative.

In termini matematici si ha che, dati:

- Un documento D_z;
- Una parola $w_{z,i} \in D_z$;

La frequenza relativa associata alla parola $w_{z,i}$ è calcolata come segue:

$$rel\_freq(w_{z,i}) = \frac{abs\_freq(w_{z,i})}{sum\_freq_z}$$

dove:

- $w_{z,i}$ rappresenta la parola i-esima nel documento z-esimo;
- abs_freq($w_{z,i}$) è la frequenza assoluta del termine i-esimo nel documento z-esimo;
- sum_freq$_z$ è la somma delle frequenze assolute di tutte le parole che sono contenute nel documento z-esimo.

Sempre basandoci sui dati descritti sopra, è stata calcolata una nuova variabile "tfidf" (term frequency inverse document frequency)[25] secondo la formula:

$$\text{tfidf}(w_{z,i}) = \text{rel\_freq}(w_{z,i}) * \log\left(\frac{N}{n_i}\right)$$

dove:

- rel_freq(w_z,$_i$) è la frequenza relativa del termine i-esimo nel documento z-esimo;
- N rappresenta il numero di documenti presenti nell'intero sistema;
- n_i definisce il numero di documenti in cui la parola i-esima è contenuta.

Si è in grado di poter scegliere quale di queste due misure di frequenza considerare e si può anche stabilire il numero k di parole da considerare per ciascun documento, scegliendo quelle per cui la misura di frequenza è più alta. Quindi inizialmente, nella fase di filtraggio dei dati, da un documento D generico, viene costruito un insieme $W_D^{'}$ $= \{w_1, w_2, ..., w_n\}$ che contiene tutte le sue parole estratte. Invece in questo step eseguiamo un altro filtraggio, il quale produce l'insieme $W_D \subseteq W_D^{'}$ che include i k termini considerati più "importanti", cioè quelli che hanno un valore maggiore nella variabile "rel_freq" o "tfidf" a seconda della misura scelta.

Viene anche assegnato per ogni parola w $\in W_D$ una nuova variabile definita "position", la quale rappresenta il "ranking" (posizione, cioè importanza) dentro la lista. Il termine con la maggiore frequenza (relativa o tfidf) ha posizione uguale a uno, la seconda avrà posizione due, e così via.

Per assicurare l'estrazione precisa di k termini nel caso in cui il k-esimo e il (k+1)-esimo vocabolo abbiano lo stesso valore della misura selezionata, si seleziona quella parola con il minor numero di word-number. Nelle tabelle 4.7 riportiamo un esempio di selezione di k=2 parole in base alla frequenza della misura "tfidf".

Per affermare che un determinato documento può essere assegnato ad una specifica categoria, dobbiamo estrarre da quest'ultima un prototipo, cioè generare un documento astratto che contenga i termini più rappresentativi per tale categoria.

docid	wordnumber	abs_freq	rel_freq	tfidf	position
100	4	7	0.42	0.46	1
100	3	5	0.29	0.12	2
100	6	4	0.11	0.51	3
100	11	2	0.09	0.30	4
...	...	...	...	...	...

⇓

docid	wordnumber	abs_freq	rel_freq	tfidf	position
100	6	4	0.11	0.51	1
100	4	7	0.42	0.46	2
100	11	2	0.09	0.30	3
100	3	5	0.29	0.12	4
...	...	...	...	...	...

⇓

docid	wordnumber	abs_freq	rel_freq	tfidf	position
100	6	4	0.11	0.51	1
100	4	7	0.42	0.46	2
...	...	...	...	...	...

Tabella 4.7: Selezione "k" parole in base alla misura "tfidf"

Come possiamo estrarre un elenco di parole importanti da un documento unico, così possiamo anche generalizzare la procedura descritta per ottenere un elenco di termini da un insieme C di documenti che, in questo caso, rappresentano tutti i documenti di una categoria I.

Dopo l'estrazione dell'elenco $W_D^{'}$ delle parole di ogni documento $D \in C$ si ha una lista $W_C^{'} = \bigcup_{\mathrm{d}\in\mathrm{C}} W_d^{'}$ la quale contiene tutti i termini che appaiono in almeno un documento. Il calcolo delle misure di frequenza per ogni parola $w_{I,j}$ relativa alla categoria I sono realizzate:

- Frequenza relativa:

$$freq\_rel(w_{I,j}) = \sum_{i \in I} freq\_rel(w_{i,j})$$

quindi la frequenza relativa di un parola j-esima è data dalla sommatoria di tutte le altre frequenze relative delle parole appartenenti ai documenti della categoria I-esima;

- *tfidf*:

$$\text{tfidf}(w_{I,j}) = \text{freq\_rel}(w_{I,j}) * \log\left(\frac{N_c}{n_j}\right)$$

dove N_c rappresenta il numero di categorie presenti nell'intero sistema e n_j definisce il numero di categorie in cui la parola j-esima è contenuta.

E' bene precisare alcune importanti osservazioni riguardo al calcolo del *tfidf*. Come prima cosa si può notare che si è data maggiore importanza alle parole più frequenti nella categoria I e poco presenti nelle categorie I' $\neq$ I. Infatti, se per esempio avessimo una parola che fosse contenuta in tutte le categorie del sistema, si otterrebbe che $N_c = n_j$ per cui il valore $\log\left(\frac{N_c}{n_j}\right)$ darebbe come risultato "0" e quindi anche il *tfidf* sarebbe "0".

Il vantaggio che si ottiene dall'uso di questa "proprietà" è notevole soprattutto quando si ha termini che sono presenti in documenti appartenenti a due aree tematiche molto distanti.

Riassumento, la costruzione dei documenti-prototipi si basa su parole appartenenti ai documenti presenti nella categoria che si vuole rappresentare, dove ognuno dei quali è identificato con le seguenti misure:

- Frequenza relativa calcolata sugli elementi intra categoria;
- tfidf ottenuto da due componenti già ampiamente spiegati e qui riportati in termini matematici:
  1. Intra categoria: $\text{freq\_rel}(w_{I,j})$;
  2. Extra categoria: $\log\left(\frac{N_c}{n_j}\right)$;

Una volta calcolate queste variabili, e deciso quale delle due utilizzare per definire l'importanza dei vocaboli, anche per i documenti prototipi viene eseguita la selezione delle k parole più rilevanti. E' lo stesso procedimento usato per tutti gli altri documenti con la differenza che per assicurare l'estrazione precisa di k termini, nel caso in cui il k-esimo e il (k+1)-esimo vocabolo abbiano l'identico valore della misura selezionata, si preleva la parola in base all'ordinamento alfabetico invece di basarsi sul campo *wordnumber*. Questo per il semplice motivo che sono creati senza *wordnumber*.

L'attività svolta successivamente riguarda generazione delle coppie (documenti, documenti prototipi). Per mostrare la versatilità delle tecniche create si è deciso di non implementare questa operazione adoperando sempre le stesse modalità. Questa scelta è stata pensata anche per fare in modo che le coppie generate si basino sulle caratteristiche, e quindi sulle informazioni, definite nei database utilizzati per gli esperimenti. Di seguito verranno analizzati i diversi principi generali adoperati per i differenti database.

Ricordando che la varie tipologie prese in cosiderazione sono già state spiegate nel paragrafo 4.1 per il "Dataset DMOZ" si ha che:

- Coppie simili: per ogni categoria del dataset se ne generano $\lfloor \frac{D}{N_c} \rfloor$;
- Coppie dissimili: per ogni categoria del dataset se ne generano $\lfloor \frac{D}{N_c} \rfloor$;
- Coppie iponime: per ogni categoria del dataset che ha un padre (i.e, si escludono le categorie del primo livello) si generano $\lfloor \frac{D}{N_c - N_{c\_no\_level\_1}} \rfloor$;

Dove N_c sono il numero di categorie presenti nel dataset, D rappresenta il valore delle coppie desiderate e $N_{c\_no\_level\_1}$ indica il numero di tutte le categorie tranne quelle del primo livello. Vista la possibilità che le coppie generate siano inferiori a quelle richieste, si inseriscono casualmente altre coppie per colmare questa differenza.

Per il "Dataset Yahoo Ceci" si hanno tre tipi di modalità aventi $L = \{L_1, L_2, ..., L_T\}$, che è l'insieme di tutti i livelli dell'albero, con L_1 che rappresenta i figli del nodo radice e L_i è il livello della categoria nella quale è presente un documento generico D_i:

1. Non si adopera il metodo training-test set ma si crea un unico dataset a cui si applica la tecnica five-folds cross-validation. Sono stati considerati il 40% dei documenti presenti nel dataset e per ognuno di questi si sono create le coppie seguenti:
   - Una coppia simile formata da D_i e il documento prototipo relativo alla categoria di appartenenza di D_i;
   - Quattro coppie divise in:
     - Due coppie iponime prese a caso tra i livelli disponibili $L - L_i$;
     - Due coppie ipernonime presi a caso tra i "padri" di D_i;
   - Di base tre coppie dissimili, in casi speciali se ne aggiungono altre per fare in modo che ogni documento abbia in totate sempre 8 istanze. Esempi di casi speciali: documenti con $L_i = 1$ non possono avere coppie iperonime perché non hanno padri, mentre quelli con $L_i = T$ non possono avere coppie iponime perché non hanno figli.
2. Rispettando la separazione effettuata negli originali esperimenti dell'articolo di Ceci e Malerba [4], si sono creati i dataset di training e test considerando tutti i documenti. Questa separazione viene realizzata a livello di documento e non a livello di categorie come si esegue nella terzo caso proposto. Per ogni documento D_i sono state prese le seguenti coppie:
   - Una coppia simile formata da D_i e il documento prototipo relativo alla categoria di appartenenza di D_i;
   - Una o più coppie dissimili basandosi sul numero di coppie desiderate meno quelle simili;

- Una coppia iponima per ogni livello dell'albero disponibile, quindi in totale $L - L_i$;
- Una coppia iperonima per ogni "padre" del nodo che contiene D_i, quindi in totale L_i-1 (e.g., se $L_i = 3$ allora le coppie di iperonimia generate per D_i possono essere la massimo due, infatti si possono avere solo due "padri" per D_i: uno è il padre diretto e l'altro è il padre del padre di D_i).

3. Per superare le costrinzioni imposte negli esperimenti di Ceci e Malerba, si è deciso di suddividere il training dal test set a livello di categorie, nel senso che il 50% del totale delle categorie si sono usate per l'addestramento del classificatore e l'altro 50% per la fase di test. Anche in questo caso, come nel secondo, si sono utilizzati tutti i documenti e per ognuno di questi si sono generate le seguenti coppie:
   - Una coppia simile formata da D_i e il documento prototipo relativo alla categoria di appartenenza di D_i;
   - Una o più coppie dissimili basandosi sul numero di coppie desiderate meno quelle simili;

Per il "Dataset DMOZ Ceci", valgono le stesse affermazioni fatte per il secondo caso del "Dataset Yahoo Ceci" con la differenza che si sono estratti il 40% di documenti di training e il 30% di documenti di test per il semplice motivo che il numero totale di documenti nel "Dataset DMOZ Ceci" è molto maggiore rispetto a quello del "Dataset Yahoo Ceci".

Inoltre, si ricorda che nel paragrafo 4.1 sono descritte le definizioni per le tipologie simili, dissimili, iponime e iperonime.

Grazie ad un parametro di casualità detto *seed*, la selezione dei documenti per la creazione delle coppie è effettuata in modo casuale e quindi ripetibile. Un risultato che si potrebbe ottenere a seguito di questa operazione è mostrato nella tabella 4.8.

catid1	**catid2**	**docid1**	**docid2**	**tipo**
95	168	1	cat168	dissimilar_unrelated
95	95	1	cat95	similar
188	131	905	cat131	dissimilar_unrelated
188	188	1002	188	similar
188	2	55	cat2	dissimilar_unrelated
...	...	...	...	...

Tabella 4.8: Generazione delle coppie documento, documento prototipo

Arrivati a questo punto, sono state create due tabelle:

- Una contenente una serie di dati W_D che rappresenta i termini più pertinenti contenuti o nel documento D_i o nel documento-prototipo D_j^P relativo ad una specifica categoria j (tabella 4.7);
- L'altra relativa ad un insieme di coppie di(documento,documento-prototipo) C = { D_i, D_j^P} che dovranno essere confrontati (tabella 4.8).

Ora bisogna ottenere, per ogni coppia $\{D_i, D_j^P\} \in C$, un unico insieme di misure prese dal confronto delle parole(W_i, W_j) dove W_i e W_j sono rispettivamente, l'insieme dei termini contenuti in D_i, D_j^P.

La decisione presa è stata quella di applicare il prodotto cartesiano dei due insiemi W_i x W_j, dove ogni coppia di parole $(w_i, w_j \in)$ W_i x W_j possono essere riconosciute attraverso WordNet. Per prima cosa si costruisce una tabella, come quella definita nell'esempio 4.9, che contiene il prodotto cartesiano. Tale tabella è composta da un numero di tuple pari a:

$$CartesianProduct_{(\mathrm{W_i\ x\ W_j})} = |W_i| \text{ x } |W_j|$$

con $|W_i|$ x $|W_j|$ che rappresentano la cardinalità dei documenti D_i, D_j^P.

Questo valore dipende dal numero di parole selezionate per definire ogni documento. Infatti ponendo tale parametro k=50 si dovrebbe avere che ogni documento sia rappresentato con 100 termini, ed il calcolo del numero totale di coppie di parole per una determinata

coppia di documenti dovrebbe essere $CartesianProduct_{(\mathrm{W_i\ x\ W_j})} = 50$ x 50 = 2.500. L'uso del condizionale è legato al fatto che un documento può contenere meno di k=50 vocaboli, e quindi si è costretti, dopo ogni esecuzione del prodotto cartesiano, a calcolarsi l'effettivo numero di coppie create che verrà poi utilizzato negli step successivi.

Nelle tabelle 4.9 si riporta un semplice esempio di come viene effettuata l'operazione del prodotto cartesiano.

docid	**word**	**rel_freq**	**tfidf**	**position**
100	car	0.11	0.51	1
100	test	0.42	0.46	2
100	lady	0.09	0.30	3
101	place	0.33	0.44	1
101	continue	0.25	0.33	2
101	circle	0.28	0.30	3
101	castle	0.44	0.27	4

⇓

id1	**id2**	**word1**	**word2**	**freq1**	**freq2**	**tfidf1**	**tfidf2**	**pos1**	**pos2**
100	101	car	place	0.11	0.33	0.51	0.44	1	1
100	101	car	cart	0.11	0.25	0.51	0.33	1	2
100	101	car	circle	0.11	0.28	0.51	0.30	1	3
100	101	car	castle	0.11	0.44	0.51	0.27	1	4
100	101	test	place	0.42	0.33	0.46	0.44	2	1
100	101	test	cart	0.42	0.25	0.46	0.33	2	2
100	101	test	circle	0.42	0.28	0.46	0.30	2	3
100	101	test	castle	0.42	0.4	0.46	0.27	2	4
100	101	lady	place	0.09	0.33	0.30	0.44	3	1
100	101	lady	cart	0.09	0.25	0.30	0.33	3	2
100	101	lady	circle	0.09	0.28	0.30	0.30	3	3
100	101	lady	castle	0.09	0.44	0.30	0.27	3	4

Tabella 4.9: Prodotto cartesiano basato sulla misura *tfidf*

Per problemi di visualizzazione, i nomi di alcuni campi sono stati abbreviati e corrispondono a:

- id1 e id2: *docid1* e *docid2*, corrispondenti rispettivamente agli identificativi dei documenti D_i, D_j^P;

- word1 e word2: *word_doc1* e *word_doc2*, parole corrispondenti rispettivamente per i documenti D_i, D_j^P;
- freq1 e freq2: *rel_freq1* e *rel_freq2*, corrispondenti rispettivamente alle frequenze relative dei termini per i documenti D_i, D_j^P;
- tfidf1 e tfidf2: corrispondenti rispettivamente ai valori della variabile "tfidf" dei termini per i documenti D_i, D_j^P;
- pos1 e pos2: *position1* e *position2*, corrispondenti rispettivamente alle posizioni (importanza) assegnata ai termini per i documenti D_i, D_j^P.

Dalla tabella ottenuta dal prodotto cartesiano si eseguono i confronti tra tutte le coppie di parole generate basandosi su WordNet, in modo tale da catturare le eventuali relazioni attraverso i formalismi lessicali e semantici. Tutte le relazioni trovate sono inserite in un unica tabella T globale realizzata cone segue:

- Due campi per la memorizzazione delle parole;
- Un campo per ogni possibile relazione presentate nel paragrafo 3.2.1. Al suo interno conterrà "1" se le parole salvate nei primi due campi hanno tale relazione semantica, altrimenti "0".

Questa tabella T servirà a velocizzare tutti i test successivi al primo. Infatti, per trovare le relazioni tra una coppia di parole si eseguono i seguenti passi:

- Si cerca la coppia all'interno di T. Possiamo avere due risultati:
  1. La coppia è presente: allora si restituiscono tutte le relazioni i cui campi hanno valore "1";
  2. La coppia non è presente: si ricercano le relazioni attraverso WordNet e poi le si inseriscono nella tabella T in modo tale che nei test successivi non si dovrà più interrogare le tabelle relative a WordNet.

Questa fase è conclusa, però come ultima cosa, si descrive una puntualizzazione sulla relazione di sinonimia adoperata.

Il concetto di *sinonimia* è realizzato su queste considerazioni:

- Data l'assenza da parte di WordNet, di rilevare la *sinonimia* per i sostantivi di forme irregolari, si decide necessario considerare questo aspetto perché tale relazione è fondamentale per ottenere un corretto confronto tra le parole. Come già detto WordNet aiuta a identificare le forme irregolari però non gli associa la relazione di *sinonimia* perché in effetti si tratterebbe di quella morfologica, ma secondo un punto di vista semantico la si può identificare come tale.

- WordNet non contiene tutti i vocaboli presenti in un dizionario cartaceo, ma solo una parte di esso. Quindi per evitare di non prendere in considerazione le relazioni di *sinonimia* tra parole non catturate da WordNet si reputano sinonime tutte quelle coppie formate da termini identici.

4.6 Generazione dataset

Questa fase genera i dataset sui quali è possibile eseguire gli algoritmi di Data Mining. Per far ciò bisogna che i dati siano riportati in un formato di tipo strutturato stabilendo per prima cosa la natura e la dimensione dell'architettura su cui dovranno essere memorizzate le informazioni dei risultati ottenuti dal confronto delle coppie di documenti.

Si è deciso che la tabella su cui si salvano i dati utili per l'addestramento e il *test* della bontà dei classificatori contiene per ogni coppia le seguenti informazioni:

1. Identificative: dettagli sulla coppia (D_i, D_j^P) formata dal documento, documento-prototipo . Si ricorda che questi attributi non

possono essere utilizzati in fase di classificazione, ma sono utili per poter rilevare eventuali errori nel dataset creato.

- catid1: categoria in cui appartiene il documento D_i;
- catid2: categoria riferita al documento prototipo D_j^P;
- docid1: identificativo numerico del documento D_i;
- docid2: rappresenta il documento prototipo D_j^P.

2. Semantiche: vera e propria sorgente di conoscenza per la classificazione. Ipotizzando che W_i e W_j^P siano gli insiemi contenenti le parole significative (w_i, w_j^P) del documento/documento prototipo D_i e D_j^P, per ogni relazione $\mathcal{R}$ delle 29 riconosciute da WordNet, abbiamo che $\mathcal{R} \subseteq W_i$ x W_j^P e si considerano le seguenti quattro misure relative:

   (a) La somma $\mathcal{F}_\mathcal{R}$ di tutte le frequenze relative dei termini delle coppie interessate in R:

$$\mathcal{F}_\mathcal{R} = \sum_{(w_i, w_j^P) \in \mathcal{R}} freq\_rel(w_i) + freq\_rel(w_j^P)$$

   (b) La somma $\mathcal{P}_\mathcal{R}$ di tutte le posizioni dei termini delle coppie interessate in R(le posizioni possono variare a seconda di quale misura si è adoperata):

$$\mathcal{P}_\mathcal{R} = \sum_{(w_i, w_j^P) \in \mathcal{R}} position(w_i) + position(w_j^P)$$

   (c) Il rapporto $\mathcal{R}_\mathcal{R}$ tra il numero di coppie interessate in R e il totale del numero di coppie:

$$\mathcal{R}_{\mathcal{R}} = \frac{|\mathcal{R}|}{|W_i \text{ x } W_j^P|}$$

(d) La somma $tfidf_{\mathcal{R}}$ di tutti i valori della misura *tfidf* dei termini delle coppie interessate in R:

$$tfidf_{\mathcal{R}} = \sum_{(w_i, w_j^P) \in \mathcal{R}} tfidf(w_i) + tfidf(w_j^P)$$

A queste informazioni si aggiunge un'altra misura $\mathcal{R}_{tot}$ calcolata come la somma di tutti i valori trovati per $\mathcal{R}_{\mathcal{R}}$. Si ottiene un indicazione sommaria di quanto siano correlate le parole dei due documenti. La formula è la seguente:

$$\mathcal{R}_{tot} = \sum_{R} \mathcal{R}_{\mathcal{R}}$$

3. Descrittive: etichetta che indica la tipologia della coppia di documenti. Valori:
   - similar;
   - dissimilar_unrelated;
   - dissimilar_related_hyponym;
   - dissimilar_related_hypernym;

Riassumento, in totale abbiamo 122 attributi che descrivono il rapporto tra una coppia di documenti. Sono divisi:

- 4 di tipo identificativo;

- 29(numero relazioni) x 4(misure adottate) + 1 (totale relazioni) = 117 di tipo semantico;
- 1 di tipo descrittivo.

4.7 Classificazione

Il tool usato per tutti gli esperimenti effettuati è Weka [11], già descritto nel paragrafo 3.3.

Ogni dataset generato nel processo di text mining, comprende un determinato numero di record dove ognuno dei quali contiene le informazioni su tutti i 122 attributi. I dataset creati sono di due tipi e hanno quindi due funzionalità diverse:

1. Training set: contiene tutte le istanze su cui viene eseguito l'addestramento del classificatore.
2. Test set: contiene le istanze su cui viene eseguita la validazione del classificatore.

A tutti i dataset viene applicato inizialmente un filtro non supervisionato "NominalToString" sull'attributo *docid2*. Per far ciò, si deve caricare il file nella parte Explorer e nella sezione "Preprocess" (pre-elaborazione) scegliere il filtro appena descritto nel reparto "Filter" (Filtri). Di default viene applicato sull'ultimo attributo, quindi per modificare le impostazioni basta cliccare sul nome del filtro e inserire l'indice dell'attributo desiderato. Questa operazione è obbligatoria se si intende eseguire la classificazione attraverso il meta-classificatore "FilteredClassifier" (classificatore filtrata).

Mentre i passi eseguiti per i test sono:

1. Caricamento del training set nella parte Explorer tramite la scheda pre-elaborazione;
2. Passare alla scheda "Classify" (classificazione);

3. Selezionare nel reparto definito "Classifier" (classificatore), il meta classificatore "FilteredClassifier";

4. Cliccare sul nome "FilteredClassifier" per impostare i parametri:

   - Nel campo "classifier" vengono scelti il classificatore J48 o Simple Cart;
   - Nel campo "filter" scegliamo il filtro "Remove" (rimuovere) non supervisionato sugli attributi identificativi, cioè i primi 4.

5. Nel reparto "Test Options" (opzioni per il test) selezionare il test tramite "Supplied test set";

6. Cliccare sul pulsante "Set" (settare) per impostare il test set;

7. Opzionale: in alcuni test, cliccando su "More options" (opzioni aggiuntive), si apre una finestra dove si possono settare alcune opzioni. Quella da impostare è "Output predictions" (stampa predizioni) che fa in modo che in uscita vengano restituite tutte le predizioni effettuate. Nel campo "Output additional attributes" si deve impostare l'identificativo numerico relativo all'attributo che si vuole visualizzare insieme alle predizioni;

8. Assicurarsi che l'attributo di classe sia impostato correttamente;

9. Cliccare sul pulsante "Start" per eseguire il test.

Capitolo 5

Esperimenti

I test effettuati hanno caratteristiche e tempistiche diverse in modo tale da poter valutare come si comporta il processo di classificazione elaborato. I dataset presi in considerazione sono descritti nel paragrafo 4.1, per ognuno di questi è riservata una sezione specifica dove sono spiegate le modalità adoperate e i risultati ottenuti.

5.1 Dataset DMOZ

Per estrarre i documenti secondo i criteri descritti nel paragrafo 4.3, si è eseguita un'analisi del catalogo DMOZ conclusasi con la determinazione dei seguenti vincoli per la creazione del "dataset I" utilizzato come base di appoggio per gli esperimenti:

1. Ogni categoria non deve contenere nel campo *topic* la stringa "World" e "Regional";
2. Ogni categoria deve avere come livello un numero compreso tra 1 e 6;
3. Ogni categoria deve contenere almeno 30 documenti;
4. Ogni categoria deve contenere almeno 3 sotto-categorie;
5. Ogni documento deve contenere almeno 60 termini;

Le categorie ottenute sono 245 aventi in totale nove ancestor distinti:

1. "Arts";
2. "Business";
3. "Computers";
4. "Health";
5. "Home";
6. "Kids_and_Teens";
7. "Science";
8. "Shopping";
9. "Society".

Per la creazione del "dataset II", si è partiti dal "dataset I" e gli si è escluso i discendenti dei nodi ancestor aventi una densità di documenti più bassa. Si ottiene che le categorie sono scese da 245 a 164, i documenti sono passati da 11644 a 8157 e gli ancestor distinti sono diminuiti da 9 a 5:

1. "Arts";
2. "Business";
3. "Home";
4. "Kids_and_Teens";
5. "Shopping".

Per la creazione del "dataset III", si è partiti dal "dataset I" e gli si è sottratto il "dataset II". Si ottiene che le categorie sono scese da 245 a 81, i documenti sono passati da 11644 a 3487 e gli ancestor distinti sono diminuiti da 9 a 4:

1. "Computers";

2. "Health";

3. "'Science";

4. "Society".

Abbiamo che il "dataset II", utilizzato come training set per gli esperimenti, non ha nessuna categoria e nessun documento in comune con il "dataset III" che invece viene utilizzato come test set. Questa particolarità ci permette di portare innovazione nel settore perché si è costruito un metodo che esegue una classificazione con un più alto livello di astrazione senza essere che sia vincolata dall'insieme di termini presenti nei documenti utilizzati come *training*. Questo è stato realizzato grazie al fatto che non si è adottato il comune approccio *bag of words* ma ci si è basati sulla presenza delle relazioni tra le varie coppie di parole.

La creazione dei documenti-prototipi viene eseguita per le 245 categorie, così da poter procedere con la generazione delle coppie (documento, documento-prototipo) per le tipologie:

- simile;
- dissimile;
- iponima.

I parametri inseriti nelle funzioni adottate per quest'ultima operazione descritta sono:

1. Fattore di casualità *seed*: 0,012;
2. Per il training set, cioè "dataset II", il numero di coppie per tipologia sono: 1640;
3. Per il test set, cioè "dataset III", il numero di coppie per tipologia sono: 400.

Nei documenti e documenti-prototipi si sono estratte le 120 parole più importanti in base alla variabile *tfidf*. Le coppie create vengono descritte dai 122 attributi già illustrati nel paragrafo 4.6, mentre i passaggi eseguiti per effettuare i test attraverso Weka, sono definiti nel paragrafo 4.7 nella sezione relativa all'uso di training e test set.

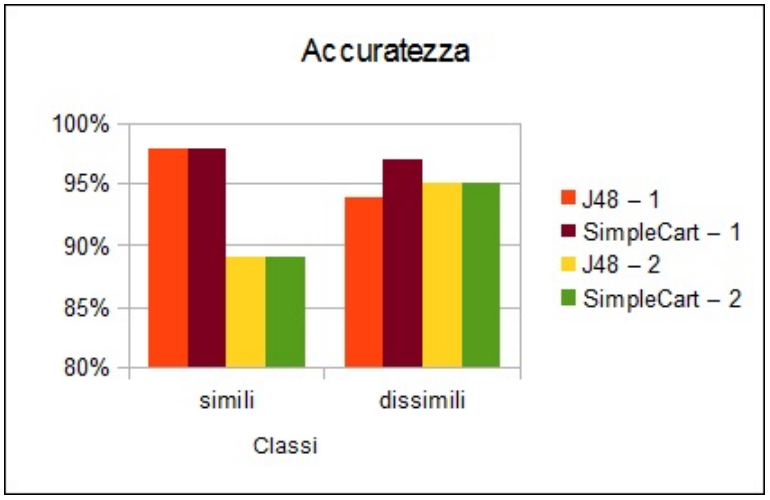


Figura 5.1: Accuratezza su coppie simili e dissimili nel "dataset DMOZ"

Dove:

1. J48 - 1 e SimpleCart - 1 sono valutati su training e test set aventi come numero di coppie per ogni tipologia ripettivamente 1640 e 400 raggiungendo così un numero di istanze pari a 3280 e 800;

2. J48 - 2 e SimpleCart - 2 scambiano il training con il test set usati in J48 - 1 e SimpleCart - 1.

Classificatore	**Precision**	**Recall**	**F-Measure**	**classe**
J48	0,94	0,98	0,96	simile
	0,98	0,94	0,96	dissimile
Media	0,96	0,96	0,96	
SimpleCart	0,97	0,98	0,98	simile
	0,98	0,97	0,98	dissimile
Media	0,975	0,975	0,98	

Tabella 5.1: Dettagli dei risultati su coppie simili e dissimili nel "dataset DMOZ"

Come si può notare dalla tabella i valori ottenuti sono molto buoni, quindi il lavoro svolto si può considerare ideale per riuscire a classificare istanze di tipo simili da quelle di tipo dissimili. In più, lo scambio tra training e test set non ha portato molte variazioni, si parla di qualche punto percentuale sulle solo istanze di tipo simili.

```
Classifier Model
J48 pruned tree
------------------

sum_synonym_tfidf <= 0.29264
|   synonym_perc <= 0.000192: dissimilar_unrelated (371.0/4.0)
|   synonym_perc > 0.000192
|   |   synonym_perc <= 0.000375
|   |   |   tot_rel_perc <= 0.000357: similar (2.0)
|   |   |   tot_rel_perc > 0.000357: dissimilar_unrelated (29.0/2.0)
|   |   synonym_perc > 0.000375: similar (4.0)
sum_synonym_tfidf > 0.29264
|   synonym_perc <= 0.000321
|   |   sum_synonym_frelwords <= 1.569599: similar (15.0/1.0)
|   |   sum_synonym_frelwords > 1.569599: dissimilar_unrelated (5.0)
|   synonym_perc > 0.000321: similar (374.0)

Number of Leaves  :     7

Size of the tree :  13

Time taken to build model: 0.15 seconds
```

Figura 5.2: Albero decisionale su coppie simili e dissimili nel "dataset DMOZ"

Lo scambio tra training e test set produce lo stesso effetto che si ottiene quando si aumenta il numero minimo di istanze che deve contenere una foglia dell'albero di decisione (questo è definito in Weka tramite il campo *minNumObj* di ogni classificatore). L'effetto è quello di ridurre notevolmente l'albero, cioé diminuire il numero di nodi e le foglie (come quello in figura 5.2), senza perdere sull'accuratezza.

Gli stessi esperimenti sono stati fatti aggiungendo alle precedenti istanze, quelle di classe iponime e nella figura 5.3 ne riportiamo i risultati ottenuti.

Dove:

1. J48 - 3 e SimpleCart - 3 sono valutati su training e test set aventi come numero di coppie per ogni tipologia ripettivamente 1640 e 400 raggiungendo così un numero di istanze pari a 4920 e 1200;

2. J48 - 4 e SimpleCart - 4 scambiano il training con il test set usati in J48 - 3 e SimpleCart - 3.

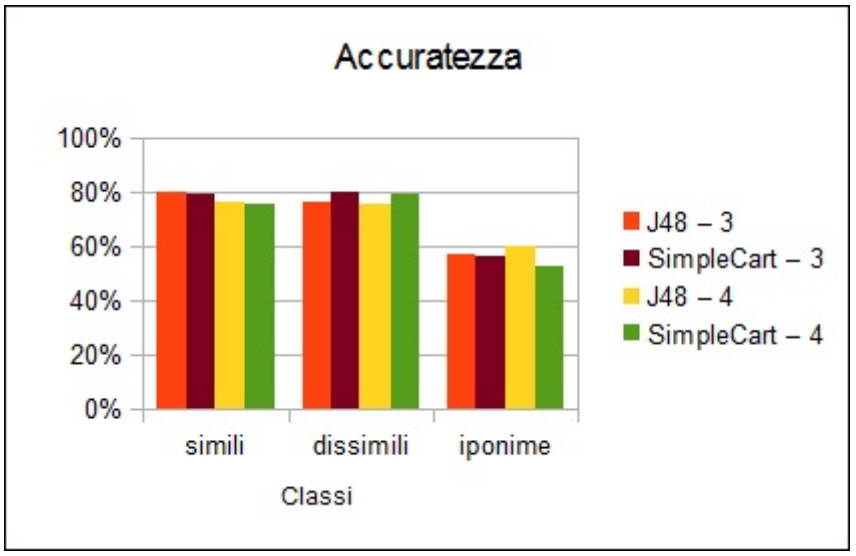


Figura 5.3: Accuratezza su coppie simili, dissimili e iponime nel "dataset DMOZ"

Come si poteva immaginare l'accuratezza scende sulla classificazione di classi simile e dissimile. Per quanto riguarda quella iponima, ci si aspettava che i risultati fossere più bassi rispetto alle altre due classi perché, ovviamente, la difficoltà a determinare se un documento è iponimo di una categoria è molto maggiore.

Questa difficoltà però potrebbe non essere sempre la stessa, infatti si può pensare che un documento figlio di una categoria sia più complicato individuarlo come iponimo di un documento nipote di una categoria. Nel primo caso abbiamo che l'iponima potrebbe essere confusa con la sinonimia mentre nel secondo caso potrebbe essere confusa con dissimilarità.

Per vedere se questa analisi è corretta si è deciso di eseguire un test specifico, nel quale si prevede l'aggiunta di un attributo in più ai 122 usati in precedenza. Questo attributo identifica la differenza di livello che c'è tra un documento e il suo documento-prototipo associato. Non

è utilizzato in fase di classificazione, ma se si inserisce il suo identificativo nelle opzioni aggiuntive in Weka (come spiegato nel paragrafo 4.7) si ha la possibilità di capire quali sono le predizione corrette/sbagliate in base a tale campo.

In aggiunta sono state generate anche coppie iperonime, il che significa che un documento appartiene ad una categoria padre rispetto a quella rappresentata dal documento-prototipo.

I risultati ottenuti relativi all'esperimento sopra citato vengono mostrati nelle figure sottostanti:

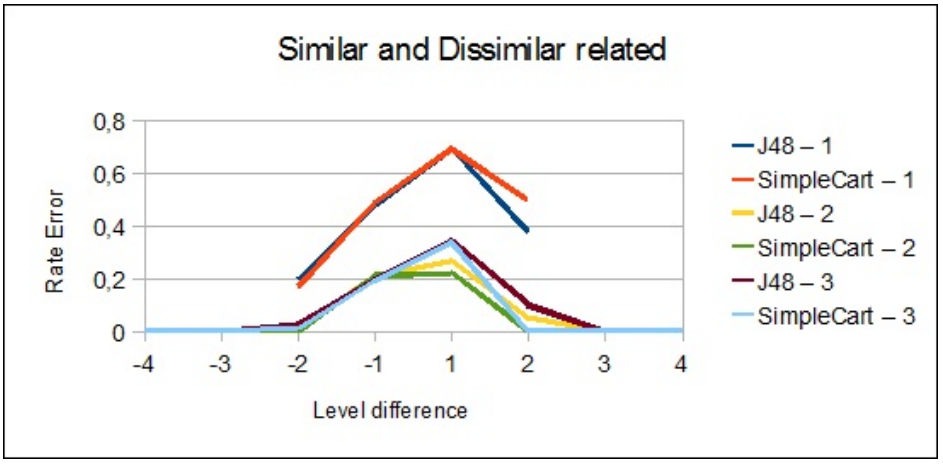


Figura 5.4: Confronto categorizzazioni su coppie similar e dissimilar related nel "dataset DMOZ"

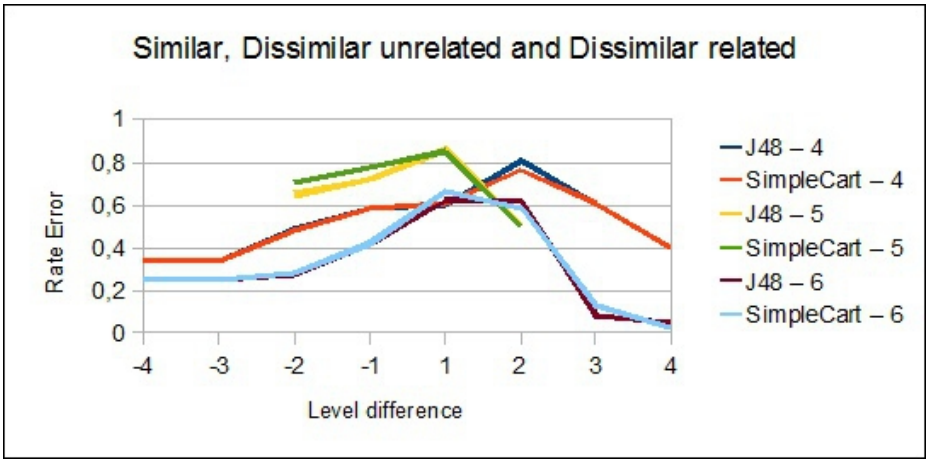


Figura 5.5: Confronto categorizzazioni documenti su coppie similar,dissimilar unrelated e dissimilar related nel "dataset DMOZ"

Prima di commentare queste immagini, illustro il loro significato e il loro scopo. Come riportato in figura 5.4 nell'asse "X" c'è segnalato il numero che identifica la differenza tra i documenti testati. Per esempio se una coppia (D_1,D_2) ha valore "1" significa che il documento D_1 appartiene ad una categoria padre rispetto a quella rappresentata dal documento prototipo D_2 mentre se ha valore "-1" si ha la relazione figlio-padre. Invece nell'asse "Y" viene riportato la percentuale d'errore di classificazione, cioè il numero di classificazione errate fratto il numero totale di classificazioni eseguite.

La classi usate sono:

1. *similar*: coppie di documenti simili;

2. *dissimilar unrelated*: coppie di documenti dissimili;

3. *dissimilar related*: coppie di documenti dissimili ma aventi lo stesso ancestor ed in relazione di parentela padre-figlio o figlio-padre.

La legenda è semplificata per occupare meno spazio, indica solo l'algoritmo di classificazione usato. Riporto di seguito maggiori dettagli, considerando sempre come dataset il "dataset II" e il "dataset III":

- J48 – 1 e SimpleCart – 1: eseguiti su un numero di coppie di documenti composti da 3280 (1640 per ogni tipo) di training set e 800 (400 per ogni tipo) di test set;

- J48 – 2 e SimpleCart – 2: eseguiti su un numero di coppie di documenti composti da 800 (400 per ogni tipo) di training set e 3280 (1640 per ogni tipo) di test set;

- J48 – 3 e SimpleCart – 3: eseguiti in modalità 10-folds cross-validation con un totale di coppie testate pari a 4080 (408 per ogni iterazione);

- J48 – 4 e SimpleCart – 4: eseguiti su un numero di coppie di documenti composti da 1200 (400 per ogni tipo) di training set e 4920 (1640 per ogni tipo) di test set;
- J48 – 5 e SimpleCart – 5: eseguiti su un numero di coppie di documenti composti da 4920 (1640 per ogni tipo) di training set e 1200 (400 per ogni tipo) di test set;
- J48 – 6 e SimpleCart – 6: eseguiti in modalità 10-folds cross-validation con un totale di coppie testate pari a 6120 (612 per ogni iterazione).

Questi risultati mostrano chiaramente che l'analisi fatta in precedenza è corretta, infatti gli andamenti delle linee hanno tutte una forma a campana il che significa che documenti che hanno una differenza di livello minore, hanno una maggiore percentuale di errore. Altre considerazioni che hanno un'importanza minore ma sempre interessanti sono:

- La percentuale di errore maggiore si trova con differenza di livelli uguale a "1";
- L'algoritmo migliore, cioè quello che ha minor "rate error" è 10-folds cross-validation con l'uso di SimpleCart;
- Gli esperimenti con test set relativo al "dataset III", hanno un range di differenza dei livelli che va da [-2,2] questo ovviamente perché si hanno a disposizione meno categorie. Come si può notare molto facilmente, l'errore è più elevato negli esperimenti in cui test e training sono scambiati;
- Un'ultima considerazione che dai grafici non si può sapere è che il numero di coppie testate è ovviamente minore quando il livello di differenza è alto. Questo è dato dal fatto che i livelli vanno da uno a sei quindi la differenza massima è cinque ma avendo che il dataset è diviso in due parti tale valore si abbassa a quattro.

5.2 Dataset WebClassIII

Per l'esecuzione dei test relativi a database creati da altri, si è deciso di adoperare quelli descritti nell'articolo di Ceci e Malerba [4] e tutti già descritti ampiamente nel paragrafo 4.1.2. Nei sotto-paragrafi riportiamo i risultati ottenuti per quanto riguarda il "dataset Ceci Yahoo" e "dataset Ceci DMOZ".

5.2.1 Dataset Ceci Yahoo

I test eseguiti estraggono dai documenti le parole in base alla variabile *tfidf*, e in quatità pari a: 20, 40 e 60.

Classificatore	**20**	**40**	**60**	**classe**
J48 - 1	0,90	0,93	0,94	simile
	0,98	0,99	0,99	dissimile
Media	0,94	0,96	0,965	
SimpleCart - 1	0,90	0,95	0,95	simile
	0,98	0,98	0,99	dissimile
Media	0,94	0,965	0,97	
J48 - 2	0,94	0,96	0,96	simile
	0,97	0,98	0,98	dissimile
Media	0,955	0,97	0,97	
SimpleCart - 2	0,94	0,97	0,96	simile
	0,97	0,98	0,98	dissimile
Media	0,955	0,975	0,97	
J48 - 3	0,95	0,98	0,97	simile
	0,98	0,97	0,97	dissimile
Media	0,965	0,975	0,97	
SimpleCart - 3	0,95	0,98	0,97	simile
	0,97	0,98	0,97	dissimile
Media	0,96	0,98	0,97	

Tabella 5.2: Dettagli dei risultati su coppie simili e dissimili nel "dataset Ceci Yahoo"

Per quanto riguarda la creazione di coppie (documento,documento-prototipo) sono state applicate le tre modalità descritte approfonditamente nel paragrafo 4.5:

- J48 – 1 e SimpleCart – 1: sone eseguite con la modalità "1";
- J48 – 2 e SimpleCart – 2: sone eseguita con la modalità "2";
- J48 – 3 e SimpleCart – 3: sone eseguita con la modalità "3".

Per tutte le altre informazioni non citate in questo paragrafo, fare riferimento a quelle scritte per i test relativi al "dataset DMOZ" (e.g., il parametro *seed* rimane impostato sul valore 0.012). Altra nota, per una più semplice comprensione si è deciso di riportare solo la misura relativa all'accuratezza.

Nella tabella 5.2 ci sono abbastanza dati per fare delle importanti considerazioni sulla bontà e precisione del metodo ideato in questa tesi. Per prima cosa si nota subito che si adatta bene a tutte le modalità utilizzate, infatti le percentuali sono altissime sia per quanto riguarda al caso di dataset sbilanciati (J48 – 1 e SimpleCart – 1), sia per il caso avente training e test set separati a livello di documenti (J48 – 2 e SimpleCart – 2), sia con dati di apprendimento relativi a tematiche completamente diverse da quelle di test (J48 – 3 e SimpleCart – 3). Addirittura i valori migliori si sono ottenuti proprio quando il classificatore non si addestra sulle categorie adoperate per la selezione delle coppie di test.

Grazie a questo, possiamo affermare che l'accuratezza del classificatore non scende anche se viene addestrato su categorie diverse da quelle utilizzate nei test. Questo dato è fondamentale perché ci da la possibilità di aprire moltissimi nuovi scenari. Ad esempio, se si vuole aggiungere una nuova categoria nella struttura, non bisogna addestrare nuovamente il classificatore come si dovvrebbe fare in tutti gli altri lavori presenti in letteratura ma basterebbe calcolarsi solamente il documento-prototipo associata alla nuova categoria.

In secondo luogo, si può osservare che i risultati ottenuti nel "dataset DMOZ" 5.1 si sono prodotti anche con i dati selezionati ed usati in altri lavori. Quindi si può constatare che la procedura proposta può

essere adoperata anche per qualsiasi dataset purché i documenti che essi contengono siano organizzati su una struttura gerarchica.

Di seguito si riportano altri esperimenti che illustrano l'accuratezza del classificatore all'aggiunta di nuove classi.

Classificatore	**20**	**40**	**60**	**Classifier**	**20**	**40**	**60**	**classe**
J48 - 1	0,81	0,82	0,86	Simple Cart - 1	0,82	0,84	0,82	simile
	0,96	0,95	0,94		0,96	0,95	0,94	dissimile
	0,30	0,37	0,45		0,30	0,36	0,44	iponime
Media	0,69	0,71	0,75		0,69	0,72	0,73	
J48 - 2	0,85	0,81	0,86	Simple Cart - 2	0,86	0,87	0,89	simile
	0,88	0,87	0,87		0,87	0,84	0,87	dissimile
	0,40	0,51	0,54		0,42	0,55	0,58	iponime
Media	0,71	0,73	0,76		0,72	0,75	0,78	

Tabella 5.3: Dettagli dei risultati su coppie simili, dissimili e iponime nel "dataset Ceci Yahoo"

Come primo impatto, l'aggiunta della classe iponima abbassa la bontà del classificatore, ma osservando meglio si nota che l'istanze di questa classe hanno bisogno di un maggior numero di informazioni per essere correttamente classificate. Infatti, aumentando il numero di parole per documento da 20 a 60, nel caso di SimpleCart - 2 l'accuratezza migliora quasi del 20%. Tutto questo è dato dal fatto che una coppia iponima è facilmente confondibile con una di tipo simile.

Classifier	**20**	**40**	**60**	**Classifier**	**20**	**40**	**60**	**classe**
J48 - 1	0,90	0,96	0,96	Simple Cart - 1	0,91	0,94	0,97	simile
	0,92	0,89	0,90		0,92	0,90	0,91	dissimile
	0,67	0,81	0,70		0,66	0,75	0,72	iponime
	0,40	0,55	0,69		0,37	0,52	0,63	iperonime
Media	0,72	0,80	0,81		0,72	0,78	0,81	

Tabella 5.4: Dettagli dei risultati su coppie simili, dissimili, iponime e iperonime nel "dataset Ceci Yahoo"

Inserendo tutte le possibili classi (i.e., simile, dissimile, iponima e iperonima), i risultati ottenuti sono abbastanza buoni. Abbiamo, come miglior valore, una media dell'81% nel caso SimpleCart - 1. Per

rivedere le caratteristiche di questo test, leggere attentamente le informazioni scritte all'inizio di questo paragrafo e di quelle presenti in 4.5.

Si ricorda che SimpleCart - 1 è creato provando a sbilanciare le istanze a favore di quelle appartenenti alla classe dissimile, più precisamente, i valori nella tabella 5.4 sono ottenuti con un dataset avente 361 istanze di tipo simile, 1664 dissimile, 472 iponime e 301 iperonime. Per gli esperimenti successivi e anche di quello precedente, per ottenere i dataset non si fa altro che togliere o aggiungere o raggruppare le verie istanze per ottenere nuove tipologie.

Un altro aspetto molto significativo è dato dal fatto che le percentuali di classificazione per le coppie di classe simile e dissimile restano sopra al 90%.

Classificatore	**20**	**40**	**60**	**classe**
J48 - 1	0,78	0,77	0,77	simile
	0,94	0,92	0,90	dissimile
	0,36	0,45	0,56	parentate
Media	0,69	0,71	0,74	
SimpleCart - 1	0,76	0,78	0,77	simile
	0,95	0,92	0,91	dissimile
	0,34	0,47	0,54	parentate
Media	0,68	0,72	0,74	
J48 - 2	0,79	0,77	0,83	simile
	0,88	0,85	0,80	dissimile
	0,43	0,52	0,62	parentate
Media	0,70	0,71	0,75	
SimpleCart - 2	0,84	0,82	0,82	simile
	0,80	0,84	0,85	dissimile
	0,47	0,56	0,61	parentate
Media	0,70	0,74	0,76	

Tabella 5.5: Dettagli dei risultati su coppie simili, dissimili e parentate nel "dataset Ceci Yahoo"

Il test relativo alla tabella 5.5 crea le coppie etichettate come "parentate" attraverso le classi di iponimia e iperonimia (i.e., sono state

unite dal test precedente le istanze appartenenti a queste due tipologie). Ovviamente i risultati sono più scarsi (si passa da un 80% visto in precedenza ad un 76%), perché l'unione eseguita non permette al classificatore di crearsi un modello con delle regole specifiche.

Aumenta notevolmente la difficoltà di dividere con precisione le coppie definite come "parentate" (quindi iponime o iperonime) con quelle simili. Infatti come si può notare nella tabella 1.4.5 l'accuratezza della sinonimia si abbassa di abbastanza, restando comunque sugli 80 punti percentuale.

Come ultima prova su questo dataset, si sono raggruppate sotto un'unica etichetta definita come "dissimili e imparentate" tutte le coppie appartenenti alle seguenti tre tipologie:

- dissimili;
- iponime;
- iperonime.

Classificatore	**20**	**40**	**60**	**classe**
J48 - 1	0,78	0,74	0,78	simile
	0,97	0,97	0,97	dissimili e imparentate
Media	0,88	0,86	0,88	
SimpleCart - 1	0,73	0,78	0,78	simile
	0,97	0,97	0,98	dissimili e imparentate
Media	0,85	0,88	0,88	
J48 - 2	0,80	0,77	0,80	simile
	0,95	0,96	0,96	dissimili e imparentate
Media	0,88	0,87	0,88	
SimpleCart - 2	0,85	0,80	0,78	simile
	0,94	0,96	0,97	dissimile
Media	0,90	0,88	0,88	

Tabella 5.6: Dettagli dei risultati su coppie simili, dissimili parentate nel "dataset Ceci Yahoo"

Questo test serve per confermare l'analisi fatta sui valori pervenuti nel test precedente. Infatti, il classificatore trova complicato classifi-

care correttamente le coppie di sinonima perchè, come nel caso dell'etichetta "imparentate", sono stati unite le tipologie: iponimia e iperonima.

Comunque i risultati sono ancora molto soddisfacenti visto che la media ottenuta si aggira sui 90 punti percentuale.

5.2.2 Dataset Ceci DMOZ

I test eseguiti estranno dai documenti le parole in base alla variabile *tfidf*, e in quatità pari a: 20, 30 e 40.

Mentre per quel che riguarda la generazione di coppie è stata applicata una sola modalità ben descritta nel paragrafo 4.5, e gli algoritmi usati sono:

- J48;
- SimpleCart.

Per tutte le altre informazioni non citate in questo paragrafo, fare riferimento a quelle scritte per i test relativi al "dataset DMOZ"(e.g., il parametro *seed* rimane impostato sul valore 0.012). Altra nota, per una più semplice comprensione si è deciso di riportare solo la misura relativa all'accuratezza.

Classificatore	**20**	**30**	**40**	**classe**
J48 - 1	0,90	0,93	0,96	simile
	0,99	0,99	0,97	dissimile
Media	0,945	0,955	0,965	
SimpleCart - 1	0,90	0,95	0,96	simile
	0,99	0,98	0,97	dissimile
Media	0,945	0,965	0,965	

Tabella 5.7: Dettagli dei risultati su coppie simili e dissimili nel "dataset Ceci DMOZ"

Le osservazioni fatte sui risultati ottenuti su coppie simili e dissimili nel "dataset Ceci Yahoo" valgono anche per questo test. I dati presenti

nella tabella 5.7 rafforzano ancora di più la nuova metodologia creata soprattutto per quel che riguarda la classificazione tra le due tipologie adoperate.

Classificatore	**20**	**30**	**40**	**classe**
J48 - 1	0,65	0,69	0,72	simile
	0,97	0,94	0,90	dissimile
	0,62	0,66	0,71	iponime
Media	0,75	0,76	0,78	
SimpleCart - 1	0,72	0,69	0,73	simile
	0,94	0,94	0,93	dissimile
	0,55	0,67	0,70	iponime
Media	0,74	0,77	0,79	

Tabella 5.8: Dettagli dei risultati su coppie simili, dissimili e iponime nel "dataset Ceci DMOZ"

I valori ottenuti sono più o meno gli stessi di quelli fatti sul nel "dataset Ceci Yahoo" (ovviamente con queste tre tipologie). Aumentano anche se di poco le percentuali sulla classificazione della classe di iponimia.

Capitolo 6

Conclusioni e sviluppi futuri

In questa tesi, si è sviluppata una tecnica di text mining per la classificazione semantica di documenti diversa dall'approccio *bag of words* comunemente utilizzato in letteratura. Il concetto innovativo si basa sulla capacità di trovare le relazioni semantiche presenti tra coppie di parole attraverso il supporto di WordNet. Un'analisi iniziale sulle proprietà di tale strumento ha reso necessario creare una fase di pre-elaborazione (*casefolding* e *stemming*[attraverso l'applicazione di regole grammaticali]) allo scopo di uniformare i dati per poter eseguire una corretta ricerca delle relazioni.

L'importanza della singola parola su un determinato documento è caratterizzata della misura *tfidf*. Le *feature* (attributi) per la classificazione sono calcolate in base all'unione dei dati numerici a loro associati con le informazioni semantiche trovate dal confronto tra documenti. Quest'ultime rappresentano la base su cui ricavare i modelli di conoscenza capaci di trovare relazioni semantiche in altri documenti. Grazie all'utilizzo delle sole relazioni definite in WordNet, le coppie formate da documenti e documenti-prototipi (i.e., rappresentazione di un'intera categoria) sono rappresentati con 122 attributi riducendo la dimensionalità dei dati e permettendo una notevole leggibilità delle regole estratte dal classificatore e un risparmio in termini di tempi di calcolo.

In più, col metodo presentato in questa tesi, è possibile generalizzare

la classificazione ottenendo un'indipendenza dal contesto. Tale operazione non è consentita negli approcci basati su *bag of words* perché si è vincolati ad usare i dati di test appartenenti alle stesse categorie usate per l'addestramento.

I dataset utilizzati per i vari test sono molteplici. Il primo è stato direttamente preso effettuando una selezione dei documenti di DMOZ, mentre gli altri due sono quelli utilizzati da Ceci e Malerba nel loro lavoro [4]. Questo permettere di dimostrare la versatilità del metodo creato. L'accuratezza ottenuta considerando coppie simili e dissimili è sempre superiore al 95% in tutti i dataset utilizzati, mentre quando s'inseriscono anche le tipologie iponime e iperonime il valore diminuisce all'80%.

Come sviluppo futuro si può pensare alla classificazione gerarchica dei documenti, partendo dalla radice dell'albero fino ad arrivare alle foglie, predisponendo un insieme di classificatori ad ogni livello dell'albero.

Bibliografia

[1] Wikipedia. `http://en.wikipedia.org/wiki/Main_Page`.

[2] L. Breiman. Bagging predictors. *Machine Learning*, 1996.

[3] M. Ceci and D. Malerba. WebClassIII. `http://www.di.uniba.it/~malerba/software/webclass`.

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

[5] J. Dorre, P. Gerstl, and R. Seiffert. Text mining: Finding nuggets in mountains of textual data, 1999.

[6] L. B. Doyle. Semantic road maps for literature searchers. *Journal of the Association for Computing Machinery*, pages 223–239, 1961.

[7] S. Dumais and H. Chen. Hierarchical classification of web content. *ACM SIGIR Conference*, 2000.

[8] C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[9] W.B Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures and Algorithms*. 1992.

[10] E. Gabrilovich and S. Markovitch. Feature generation for text categorization using world knowledge. *Computer Science Department, Technion, 32000 Haifa, Israel*.

[11] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The weka data mining software: An update, 2009.

[12] E.-H. Han and G. Karypis. Centroid-based document classification: Analysis and experimental results. In *PKDD '00: Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*. Berlin Heidelberg New York: Springer, 2000.

[13] M. Hearst. Untangling text data mining. `http://www.sims.berkeley.edu/~hearst/papers/acl99/acl99-tdm.html`, 1999.

[14] L. Henderson. *Automated Text Classification in the DMOZ Hierarchy*, 2009.

[15] D.D. Lewis. Reuters-21578 text categorization test collection. *AT&T Labs Research*, 1997.

[16] D.D. Lewis, Y. Yang, T.G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5:361–397, 2004.

[17] H. P. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, pages 159–165, 1958.

[18] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. 1998.

[19] G. Miller. Dmoz. `http://www.dmoz.org`.

[20] T. M. Mitchell. *Machine Learning*. New York: McGraw-Hill, 1997.

[21] J. Platt. Fast training of support vector machines using sequential minimal optimization. In *B. Scholkopf, C. Burges and A. Smola (Eds.), Advances in Kernel methods – support vector learning*. MIT Press, 1998.

[22] M.F. Porter. An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3):130–137, 1980.

[23] A. Pulijala and S. Gauch. Hierarchical text classification. In *International Conference on Cybernetics and Information Technologies*, Orlando, USA, 2004.

[24] J.J. Rocchio. Relevance feedback in information retrieval. 1971.

[25] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. In *INFORMATION PROCESSING AND MANAGEMENT*, pages 513–523, 1988.

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

[27] F. Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002.

[28] D. R. Swanson. Arrowsmith. `http://arrowsmith.psych.uic.edu`.

[29] D. R. Swanson. Historical note: Information retrieval and the future of an illusion. *Journal of the American Society for Information Science*, 1988.

[30] D. R. Swanson and N. R. Smalheiser. Implicit text linkages between medline records: Using arrowsmith as an aid to scientific discovery. *Library Trends*, pages 48–51, 1999.

[31] G.-R. Xue, D. Xing, Q. Yang, and Y. Yu. Deep classification in large-scale text hierarchies. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 2008.