Università degli Studi di Padova
Dipartimento di Scienze Statistiche
Corso di Laurea Triennale in
Statistica e Tecnologie Informatiche

Relazione finale

Confronto tra Data Mining e Information Retrieval

Relatore: Professore Massimo Melucci
Dipartimento di Ingegneria dell'Informazione

Laureando: Boyuan Zhang
Matricola N. 1067641

Anno Accademico: 2016-2017

Confronto tra Data Mining e Information Retrieval

Indice

Prefazione

L'*Informazione* è ciò che modifica la conoscenza, cioè l'insieme coordinato di fatti identificati, scelti e acquisiti da una persona nel corso della propria vita in relazione a uno o più argomenti, in occasione della risoluzione di problemi o dello svolgimento di compiti. A seguito dell'aumento rapido delle innovazioni scientifiche, lo sviluppo di siti web, le tecnologie informatiche, la velocità dell'accumulo dei dati e delle informazioni è diventato esponenziale, tali quali vengono poi salvati nei grandi archivi come i *data warehouse*. Di conseguenza per ricavare informazioni utili, gli strumenti tradizionali falliscono, pertanto vengono poi introdotti nuovi metodi per effettuare le analisi dei dati. L'informazione assume una veste concreta mediante i dati, ovvero simboli che, una volta elaborati, formano la conoscenza di chi apprende l'informazione in essi contenuta e la interpreta. Per molti anni, i ricercatori del campo di *data science*, per l'estrazione di grandi masse di informazioni, hanno reso una strada più semplice e disponibile. Questo è dovuto in primo luogo alla realizzazione di metodi automatici per la raccolta e miglioramenti di dati in sistemi elettronici, i quali permettono di immagazzinare informazioni e ridurre i principali di costi. La scienza (e non solo), come molte altre discipline pertanto si è affacciata ad un nuovo approccio nel mondo del lavoro e della ricerca, dalla tecnologia alle imprese, così come nella vita di tutti i giorni.

Lo scopo di questa relazione è mostrare alcuni confronti tra il **data mining** (DM) ed l'**information retrieval** (IR) da vari punti di vista. Inoltre, sono stati riportati alcuni algoritmi di apprendimento di base. Questa tesi, tuttavia, copre solamente alcuni argomenti più importanti di questi due ambiti disciplinari.

Capitolo 1

Introduzione

La crescente disponibilità dei dati nell'attuale società dell'informazione ha evidenziato la necessità di disporre di strumenti adeguati per la loro analisi. Per poterne estrarre informazioni utili e fare decisioni basandosi sulle informazioni ricavate, in *data science* vi sono varie tipologie di strumenti applicativi, i quali si distinguono a seconda delle esigenze informative. Il data science è un settore multi-disciplinare che coinvolge *data mining*, *information retrieval*, *machine learning*, *statistica*, *basi di dati*, *intelligenza artificiale* ecc. In molti contesti applicativi ci si confondono facilmente le idee sulle differenze tra questi. Sebbene in entrambi essi si cercano di ottenere qualcosa dai dati per raggiungere uno scopo fissato a priori. L'**information retrieval** (**IR**) è un'insieme di applicazioni che utilizza gli algoritmi di **data mining** (**DM**) per ottimizzare il reperimento delle informazioni, dove le informazioni sono state generate da un insieme di dati a seguito di un'analisi dell'apprendimento automatico (*machine learning*). Tuttavia, al livello concettuale e applicativo l'IR e DM vi mostrano delle differenze significative.

Successivamente verranno descritte alcune caratteristiche principali di data mining e information retrieval. In questa relazione, per descrivere l'information retrieval, per la maggior ragione come contesto di applicazione, viene preso in riferimento soprattutto l'ambito del web. Benché l'information retrieval possa essere applicato anche in molti altri contesti.

Capitolo 2 - Data Mining. In questo capitolo è stata fatta una prima descrizione sugli argomenti principali che compongono il data

mining, i quali sono: *regole di associazione*, *classificazione*, *segmentazione* e *apprendimento parzialmente supervisionato*.

Capitolo 3 - Information Retrieval. In questo capitolo sono stati riportati gli argomenti principali di information retrieval, nel cui sono stati descritti i concetti di *analisi lessicale*, *indicizzazione*, metodi di *reperimento* e *ordinamento*, e una descrizione su *ricerca sul web*. Inoltre sono stati allegati alcuni algoritmi tipo quelli di indicizzazione, di ricerca, e implementazioni di reperimento.

Capitolo 4 - Applicazioni di DM in IR. Nel capitolo 4 sono state descritte in maniera più approfondita dalla base matematica-statistica le relative applicazioni di data mining utilizzate in information retrieval. I quali sono *regole di associazione*, *alberi di classificazione*, *classificazione di naïve Bayes*, *support vector machines*, *algoritmi di segmentazione* e *learning parzialmente supervisionato*. Dopodiché sono stati introdotti alcuni *modelli di reperimento* e il *feedback di rilevanza* quali fanno uso dei metodi di apprendimento di data mining.

Capitolo 5 - Misure di Valutazione. Nelle misure di valutazione in information retrieval sono riportati gli argomenti come *esaustività* e *specificità* dell'indice, il controllo statistico, e la misurazione dell'indice. Dopodiché è stata descritta la *valutazione dei modelli e algoritmi*.

Capitolo 6 - Analogie e Differenze tra DM e IR. Considerando gli aspetti tecnici di base, questo capitolo parte distintamente dalla descrizione dei vari compiti che si svolgono in information retrieval e data mining, o meglio vengono descritte alcune differenze e punti comuni tra questi due argomenti. Dopodiché vengono fatti dei cenni su *text mining*, *web mining* e *classificazione delle pagine web*.

Capitolo 7 - Conclusione.

Capitolo 2

Data Mining

Il *data mining* (DM) è un processo di estrazione di conoscenza da banche dati di grandi dimensioni, che avviene tramite l'applicazione di algoritmi che individuano le associazioni "nascoste" tra i dati e le rendono visibili. In altre parole è il processo di esplorazione e analisi, che ha una natura automatica o semi-automatica, al fine di scoprire dei modelli (*patterns*) e regole significative dai dati. Questo processo si consiste nell'estrazione complessa di informazioni implicite, precedentemente sconosciute e potenzialmente utili all'interno di grandi archivi di dati. Un pattern indica una struttura, che in generale è una rappresentazione sintetica dei dati. Si parte dall'assunto che ci sia più conoscenza nascosta nei dati di quella che si mostra in superficie e si usano tecniche statistiche, simboliche, subsimboliche e di visualizzazione. Le tecniche utilizzabili sono molteplici e, di conseguenza, così sono anche gli appositi algoritmi di apprendimento quali vengono implementati. La scelta dell'algoritmo dipende principalmente dall'obiettivo (di tipo descrittivo e/o predittivo) che si vuole raggiungere e dal tipo di dati da analizzare.

2.1 Nozioni Preliminari

Knowledge Base. Il *Knowledge Base* (o base di conoscenza) è un dominio di conoscenza utilizzato per la ricerca, o per valutare un modello d'interesse. Può includere il concetto di gerarchie utilizzato per organizzare attributi o attribuire valori in diversi livelli di astrazione. Può anche essere incluso *knowledge* come credenze

degli utenti, il quale può essere utilizzato per valutare un modello d'interesse basato sulla sua imprevedibilità.

Knowledge Discovery in Database. Il *Knowledge Discovery in Database* (KDD) è la scoperta di conoscenza dai depositi di dati, il processo d'estrazione di informazioni implicite, precedentemente sconosciute e potenzialmente utili dai basi di dati. Il proliferare di dati e la capacità di immagazzinarli in grossi basi di dati obbliga ad adattare le strategie e sviluppare metodi meccanici per filtrare, selezionare e interpretare i dati. A partire dai vari archivi di dati, l'utente può desiderare di sapere: dove si trovano i dati, quali dati ci sono, in che formato essi esistono, come questi sono in relazione con altri dati provenienti da altri depositi di dati, da dove arrivano e a chi appartengono ecc. È ragionevole suddividere il processo KDD in: *data pre-processing, data mining e post-processing.* Tutto il processo è quasi sempre iterativo, e potrebbe richiedere più giri per raggiungere i risultati soddisfacenti.

Data Pre-Processing. Durante il data pre-processing, i dati grezzi a priori, vengono ripuliti per rimuovere le anomalie. Poiché potrebbero essere di larga scala, e includere valori attributi irrilevanti, vengono eseguiti i seguenti passaggi: *integrazione dei dati* (combinazione di più fonti di dati), *selezione di dati rilevanti* e *trasformazione dei dati.*

Data Mining Processing. Al processo di DM, ovvero quando i dati processati sono ora pronti per essere usati dagli algoritmi di analisi e apprendimento, per poi costruire modelli (*pattern*) e ricavare conoscenze. Il DM è un passo essenziale, dove nel cui vengono applicati i metodi intelligenti. Questa è la fase che si interagisce con l'utente e/o il Knowledge base. I modelli interessanti vengono poi presentati all'utente, i quali possono essere memorizzate come nuove conoscenze nella Knowledge Base. Si potrebbe considerare il data mining come un passo di tutto il processo KDD.

Data Post-Processing. In molte applicazioni, non tutti i pattern trovati sono utili, nel passo di data post-processing si individua il

pattern desiderato. Vi sono molteplici tecniche che possono essere usate per la valutazione e visualizzazione. Questo ultimo passo si consiste in *valutazione del modello* e *presentazione della conoscenza*. Il primo identifica i modelli quali veramente interessanti che rappresentano la conoscenza basata su alcune misure interessanti; ed il secondo fa il lavoro di visualizzazione e rappresentazione del sapere per presentare la conoscenza estratta all'utente.

Funzioni e Modelli. Le funzioni di data mining specificano il tipo di modello che si trova nelle applicazioni di data mining. Il mining può essere ed suddiviso in: descrittivo e predittivo. Il primo è caratterizzato dalle proprietà generali dei dati nel database. Quello predittivo esegue l'analisi sui dati attuali, con lo scopo di fare previsioni. In alcuni casi, gli utenti possono non avere idea di quale tipologia di modello che può rappresentare i dati, e quindi parallelamente è anche possibile di trovare svariate tipologie di pattern differenti. Di conseguenza è importante disporsi un sistema di data mining capace di scoprire i vari tipi di modelli dai dati per accogliere le diverse aspettative degli utenti o applicazioni.

Il data mining offre numerosi strumenti per l'esplorazione e analisi dei dati: in letteratura si propongono numerosi algoritmi, spesso sono risultati di combinazioni di più tecniche diverse o varianti di algoritmi più noti al fine di ottimizzare determinati aspetti critici presenti nella formulazione originale. I principali problemi in data mining a cui si rispondono sono **regole di associazione (o correlazione)**, **classificazione** e **segmentazione (o clusterizzazione)**. Le tecniche di segmentazione consentono di effettuare operazioni di raggruppamento sui dati, cioè di individuare gruppi omogenei, o tipologie, che presentano delle regolarità al loro interno in grado di caratterizzarli e differenziarli dagli altri gruppi. Le operazioni di classificazione fanno uso della conoscenza acquisita in fase di addestramento per classificare nuovi oggetti o prevedere nuovi eventi. Le tecniche di analisi delle associazioni consentono di individuare delle regole nelle occorrenze concomitanti di due o più eventi.

Contesti Applicativi di Data Mining

- **Segmentazione della Clientela** (*database marketing*): applicazione delle tecniche di segmentazione per individuare i raggruppamenti impliciti nei dati, omogenei in termini di comportamento d'acquisto e di caratteristiche socio-demografiche.
- **Customer Redention**: applicazione di tecniche previsive per individuare i clienti in fase di abbandono.
- **Fraud Detection**: individuazione di comportamenti fraudolenti.
- **Analisi delle Associazioni** (*market basket analysis*): individuazione dei prodotti acquistati congiuntamente; individuazione di associazioni "anomale".
- **Competitive Intelligence** (*technology watch*): applicazione di tecniche di raggruppamento a documenti estratti da banche dati internazionali di tipo tecnico-scientifico volte ad individuare le tecnologie emergenti, le loro relazioni, l'evoluzione temporale e le aziende coinvolte.
- **Analisi Testuale** (*text mining*): individuazione degli argomenti trattati da un insieme di documenti di tipo testuale e delle relazioni tra argomenti.
- **Biologia**: in ambito biologico, per la previsione della struttura di proteine, ricerca di omologie, analisi di sequenze genomiche, individuazione e mappatura di geni, analisi di dati di espressione genetica (*microarrays*); per quanto riguarda l'analisi della letteratura scientifica, queste tecniche consentono di identificare e classificare specifici termini biologici (es: nomi di proteine), individuare parole chiave e concetti, raggruppare automaticamente, classificare ecc.

2.2 Problemi dei Modelli Regressione

Per effettuare le analisi e previsioni sui dati, la regressione è lo strumento statistico più classico che viene usato. Il concetto di base su cui poggia l'intera teoria è la possibilità di approssimare la vera funzione che ha generato le osservazioni. La previsione è verificata sugli scarti d'errore generati dal modello rispetto ai valori reali. La regressione si suddivide in approccio *parametrico* e *non parametrico*. Nei modelli lineari (e anche nei modelli lineari generalizzati)

si studiano le relazioni tra la variabile d'interesse e le variabili concomitanti. Tuttavia, questi ultimi si limitano ad analizzare insiemi di dati dati piccoli, tali che tendono a fallire quando il numero di osservazioni diventa significativamente grande, che di conseguenze dal punto di vista computazionale può diventare problematico il caricamento della matrice dei dati. Poiché con la quantità notevole dei dati è molto facile falsare qualsiasi modello parametrico, l'approccio non parametrico diventa particolarmente efficace.

2.3 Regole di Associazione

Regole di associazione (*association rule*) è una tecnica basata sulla ricerca di relazioni e dipendenze tra determinati attributi delle osservazioni. L'obiettivo non consiste nella previsione di un certo valore, ma si cerca di informare l'utente della presenza di particolari affinità tra determinati attributi valide per un sottoinsieme significativo dei dati disponibili. Spesso l'obiettivo dell'analisi consiste nella scoperta di relazioni *causa/effetto*. Pertanto lo scopo di regole di associazione è quello di trovare tutte le relazioni *associazioni* tra le osservazioni dei dati, ovvero le loro **co-occorrenze**. In ambito medico ad esempio, vengono usate tali tecniche per definire delle regole con cui assegnare la priorità di pronto soccorso ai pazienti sulla base delle loro caratteristiche fisiche e sintomatiche. Mentre l'esempio più classico è l'analisi del carrello di spesa, dove si vuole scoprire le varie associazioni sul modo in cui gli articoli vengono venduti. Inoltre anche i documenti di testo possono essere trattati come transazione di dati, dove si può considerare ciascun documento come una transazione, ed ogni distinta parola può essere considerata come un item. Oltre a ciò, la scoperta di regole può anche essere eseguita su tabelle relazionali.

2.4 Classificazione e Previsione

La *classificazione* è l'operazione che determina l'appartenenza di un'osservazione ad una classe sulla base dei valori degli attributi che caratterizzano l'osservazione stessa. In questo approccio si consiste nella etichettatura di ciascuna osservazione con un certo valore di-

screto tra un insieme ben definito di possibilità. Data una nuova osservazione il classificatore (ossia il modello costruito con tale tecnica) indicherà l'etichetta corretta sulla base dei valori attributi. La costruzione del classificatore si basa su un insieme di osservazioni di una matrice dei dati, che di cui è già noto il valore dell'etichetta. A partire da tali esempi, l'algoritmo di classificazione apprende le regole implicite che determinano l'appartenenza ad una certa classe. La classificazione nel mondo di *machine learning* è anche definito come *apprendimento supervisionato*, che come tipo di apprendimento, è analogo a quello dell'essere umano, ovvero si cerca di istruire il calcolatore conoscenze nuove dalle esperienza passate, in modo tale da migliorare le sue abilità durante gli svolgimenti dei compiti del mondo reale. Ne esistono svariati tipi di funzioni di classificazione, in questa relazione vi vengono mostrati solamente alcuni tra quelli più importanti.

Sia D una matrice dei dati. Sia $A = \{A_1, A_2, ..., A_{|A|}\}$ l'insieme di valori attributi (*attribute value*) che descrive il dataset D. Sia $C = \{c_1, c_2, ..., c_{|C|}\}$ l'insieme di attributi di classe (*attributi speciali, risposta o class value*) del dataset D. L'attributo speciale C viene considerato distintamente rispetto gli attributi in A dovuto al suo stato speciale, ovvero si assume che $C \notin A$. La classe attributi C possiede un insieme di valori discreti, dove il numero di classi $|C|$ è ≥ 2. Il valore della classe (*class value*) è anche definito come **etichetta classe** (*class label*). Ogni record di dati descrive un pezzo di "esperienza passata". Un record di dati è anche chiamato come **esempio**, **istanza**, **caso** o semplicemente **vettore**. Concretamente, dato un insieme di dati, l'obiettivo è produrre una funzione di **classificazione/previsione** che individua una relazione tra i valori degli attributi di A e la *class attribute* di C. Tale funzione può essere usata per predire l'etichetta classe (*class values* o di *risposta*) del futuro. Il risultato che questa funzione restituisce è definito come **modello di classificazione** (oppure **modello predittivo** o **classificatore**). Le tecniche di classificazione come applicazione di data mining in information retrieval che verranno descritte sono *alberi di decisione*, *naïve Bayes*, *support vector machines*.

Training set e Test set

Allo scopo delle analisi, spesso si suddivide la matrice dei dati in *training set* e *test set*. La partizione usata per l'apprendimento è **training set**. Una volta stimato il **modello** con i dati del training set mediante un **algoritmo di apprendimento**, si effettua poi la valutazione con l'insieme dei dati del **test set** per verificare l'accuratezza del modello. È importante tener presente che i dati del test set non vadano utilizzati durante l'apprendimento del modello di classificazione. I dati del test set usualmente possiedono anche le etichette di classe. Per verificare l'accuratezza, è pertanto possibile verificare se la classe predetta dal modello per ciascun test set sia uguale rispetto alle classi attuali del test set. L'accuratezza di un modello di classificazione su un test set è definita come segue:

$$Accuracy = \frac{\text{Numero di corrette classificazioni}}{\text{Numero totale di } \textit{test cases}} \tag{2.1}$$

dove per corretta classificazione si intende per modello appreso che predica la stessa cosa del test set.

Perciò, nell'apprendimento si assume l'uguaglianza tra la distribuzione del training set e quella del test set. Tuttavia, nelle applicazioni reali, tale assunzione è spesso violata ad un certo grado di libertà. Si può dire che il classificatore ha una buona accuratezza quando il training set si discosta poco dal test set. Se l'accuratezza risulta essere abbastanza soddisfacente, allora il modello può essere usato nelle applicazioni reali per predire le classi di dati nuovi (dove non ci sono ancora classi). Altrimenti occorre tornare dietro e scegliere un algoritmo di apprendimento alternativo. Spesso non si riesce a costruire un modello soddisfacente per il fatto di avere un'alta variabilità nei dati, o limitazioni di algoritmi di apprendimento presi in uso a quel determinate istante.

Apprendimento mediante un Sistema Informativo

Dato un dataset D che rappresenta un "esperienza passata", una funzione T e un misuratore di prestazione M, si dice che un sistema informativo apprende dai dati per rendere al meglio le funzionalità T in quanto misurato da M. In altre parole, il modello (o conoscenza) appreso(a) aiuta il sistema ad ottenere prestazioni migliori

rispetto a quanto avuto in precedenza. Pertanto l'apprendimento è un processo di costruzione del modello o estrazione di conoscenza. La domanda tecnica è: quanto meglio possiamo fare con l'apprendimento? Se il modello appreso può migliorare l'accuratezza allora si può dire che è **efficace.**

Alberi di decisione (o classificazione)

Il modello *Alberi di decisione* (o *di classificazione*), presenta la realtà dei dati come un albero, costituisce sia uno strumento per la rappresentazione di un problema decisionale, sia un supporto per l'analisi di una *sequenza di decisioni*. Come tecnica è nata nella teoria delle decisioni. La selezione tra le possibili alternative è valutata sulla massimizzazione del valore restituito da una funzione di rischio in corrispondenza di ciascuna scelta. In data mining, la sua applicazione naturale è relativa ai problemi di classificazione. Nel caso in cui la classificazione assumi variabili discrete, si dice albero di *decisione*, mentre quando le variabili sono continue è detto albero di *regressione*. L'idea si consiste nella suddivisione (*splitting*) dell'insieme dei dati in partizioni dove si assume che le osservazioni abbiano valori attributi simili: i punti di divisione dell'albero si chiamano *nodi* che corrispondono ad affermazioni di tipo logico sui valori assunti da uno o più attributi dell'osservazione.

Classificazione di Naïve Bayes

La formulazione corretta per impostare il problema della classificazione consiste nel considerare una variabile categoriale che rappresenti la classe di appartenenza dell'osservazione e una variabile p-dimensionale X come l'insieme di attributi dell'osservazione stessa. Dato il test set d, la funzione di classificazione può essere considerata come una stima di probabilità a posteriori

$$Pr(C = c_j|d). \tag{2.2}$$

Quindi possiamo vedere quale classe c_j è più probabile. La classe con con la probabilità più alta è pertanto assegnata all'esempio d.

Support Vector Machines (SVM)

Il *Support Vector Machines* è un tipo di sistema di apprendimento che si vanta di una fondazione teorica solida. Risponde ad una quantità considerevole di dati. È tra i quali che effettua le classificazioni anche in maniera più accurata rispetto a altri applicazioni di classificazione, in particolare quando i dati sono su una scala talmente vasta. L'SVM è un sistema di apprendimento lineare, costruita su due classi, dove ogni istanza di dati viene chiamato vettore d'input.

K-Nearest Neighbor Learning

In questo metodo, l'apprendimento si verifica solamente quando un campione test ha bisogno di essere classificato. L'idea di kNN è estremamente semplice ed è abbastanza effettiva in molte applicazioni, come ad esempio la classificazione dei testi. I metodi di apprendimento come alberi di decisione (o classificazione), regole di associazione, le probabilità a posteriori e iperpiani apprendono un insieme finito di modelli dal training data. Dal fatto che si apprendono i modelli dei dati prima del test, questi metodi di apprendimento sono stati chiamati **eager learning** (*apprendimento ansioso*). In contrasto, si dice che k-nearest neighbor (kNN) è un metodo di apprendimento chiamato **lazy learning**, nel senso che non ci sono modelli che apprendono dai dati.

Misure di Valutazione

Nei contesti di apprendimento supervisionato, alla fase successiva della costruzione del modello, avviene la valutazione dell'accuratezza del classificatore. Vi esistono tanti modi per valutare un classificatore, ed anche molte misure. La misura principale è la classificazione di **accuratezza** I test statistici di significatività possono essere usati per verificare se un classificatore è significativamente migliore di un altro dataset contenente training set e test set. In alcuni campi di ricerca si usa anche il **tasso di errore** tale che è $1-accuracy$. Se vi sono usati diversi classificatori, si preferisce quello con maggiore accuratezza.

2.5 Segmentazione

La segmentazione (*clusterizzazione* e/o in machine learning si dice *apprendimento non supervisionato*) mira alla separazione delle osservazioni dei dati senza attributi in gruppi (*clusters*) o classi significative, in cui tutti i membri di ciascun gruppo debbano avere caratteristiche simili. A differenza della classificazione, i gruppi non sono noti a priori. Si basa su un apprendimento non supervisionato, l'utente non deve fornire nessun esempio precedentemente classificato. Oggigiorno molte applicazioni utilizzano la tecnica della segmentazione: analisi della spesa (*market basket analysis*), analisi della vulnerabilità dei clienti (*risk analysis*), l'individuazione delle truffe (*fraud detection*), ecc. In molte applicazioni i dati non hanno i class attributes, pertanto occorre trovare una struttura intrinseca in essi. Il clustering è una tecnica che organizza i dati in gruppi di somiglianza, chiamato **cluster**. Un cluster è dunque una collezione di dati che siano simili o dissimili gli uni agli altri rispetto agli altri clusters. Esistono due tipi di clusterizzazione: *divisivo* e *gerarchico*.

2.6 Apprendimento parzialmente supervisionato

Nell'apprendimento supervisionato, per generare una funzione di classificazione, l'algoritmo di apprendimento utilizza i dati del training set, dove cui da tutte le classi possiedono etichette. Per il fatto che l'etichettatura sia spesso manuale, che di conseguenza potrebbe rendere il lavoro molto dispendioso, una delle inconvenienti è riuscire ad apprendere accuratamente una grande quantità di esempi. Pertanto, a dispetto di questi svantaggi sono stati introdotti due tipi di apprendimento parzialmente supervisionato: **LU learning** (*Labeled and Unlabeled learning* o *apprendimento da esempi etichettati ed esempi non etichettati*), e **PU learning** (*positive and unlabeled learning* o *apprendimento da esempi positivi ed esempi non etichettati*).

Capitolo 3

Information Retrieval

L'*information retrieval* (IR) è uno studio che si focalizza a trovare informazioni che soddisfino le **esigenze informative** degli utenti. Concretamente, si concentra sulla *struttura, analisi*, *rappresentazione*, *memorizzazione*, *organizzazione*, *distribuzione* e *reperimento* dell'informazione. Per cui lo scopo di un sistema di IR è di trovare informazioni che potrebbero essere *utili* e/o *rilevanti* per l'utente finale. L'enfasi non è sulla ricerca dei dati ma sulla ricerca delle informazioni. Un sistema di information retrieval (IRS) è realizzato per svolgere in modo automatico i compiti di reperimento delle informazioni. Le componenti dell'IRS rappresentano le strutture dei dati, e mediante gli algoritmi, automatizza i processi di **indicizzazione** dei dati e di **reperimento** delle informazioni. L'IRS, a parità di ogni interrogazione in arrivo, calcola la **rilevanza** per ciascun documento indicizzato. Quest'ultimo svolge anche la **collezione dei documenti** di qualsiasi dimensione, assicurando la descrizione del contenuto informativo e, reperimento veloce dei documenti che rappresentano informazioni rilevanti per rispondere alle esigenze informative.

Per il fatto della convenienza e dell'abbondanza delle informazioni disponibili sul web, l'IR nella ricerca sul web ha assunto un ruolo dominante. In information retrieval, si utilizza la parola **documento** per indicare un contenitore persistente di dati e identificabile in modo univoco. Alcuni esempi di documento possono essere libri, capitoli di un libro, articoli scientifici, quelli quotidiani o periodici, video, discorsi, immagini ecc. Un documento ha il ruolo di "documentare", poiché le informazioni rappresentate da quei dati contribuiscono alla formazione della conoscenza di una persona.

Pertanto in IR, una *collezione di documenti* è l'insieme di documenti da rappresentare, descrivere, memorizzare e da gestire in modo automatico.

3.1 Architettura di un Sistema IR

L'*architettura* di un sistema IR potrebbe essere descritta partendo da un'interrogazione utente (*query*) che arriva al sistema di reperimento tramite gli appositi comandi di ricerca. Il modulo di reperimento usa gli **indici del documento** per reperire documenti che contengano qualche termine di interrogazione (i documenti devono essere verosimilmente rilevanti all'interrogazione data in input), poi calcola il punteggio di rilevanza mediante appositi algoritmi di ottimizzazione, e quindi classifica i documenti a seconda dei punteggi di rango restituito dall'algoritmo di massimizzazione. Le collezioni di documenti vengono poi indicizzati da un indicizzatore. Le funzioni di reperimento danno un **ordinamento** ai documenti. La misura di rilevanza fornita dall'algoritmo di reperimento è infatti un numero reale che consente un ordinamento almeno parziale. Uno schema particolare di indicizzazione è l'**indice invertito**, tale che è molto usato nei sistemi IR, e rende facilmente una ricerca efficiente. Il **sistema di reperimento** calcola il punteggio di rilevanza per ogni documento indicizzato per la query. Così i documenti, in base al punteggio di rilevanza vengono illustrati agli utenti. Nel trovare la migliore configurazione in base alla quantità di documenti rilevanti o non rilevanti, la velocità, la quantità di risorse utilizzate, e alla funzionalità della configurazione, per ogni reperimento effettuato, è necessariamente fare una valutazione di efficacia ed efficienza. Il modulo delle **operazioni di interrogazione** può variare dal più semplice al più complesso. Quello più semplice può essere quella della rimozione degli **stop words**, mentre il caso più complesso è la trasformazione del linguaggio di interrogazione naturale in interrogazione eseguibile. Dopotutto, vengono realizzati i feedback utente, ed usarli per raffinare le interrogazioni originali. Questo ultimo procedimento si chiama **feedback di rilevanza**.

3.2 Pre-Processo dei dati testuali e delle pagine web

Prima che i documenti di una collezione vengano usati per il reperimento, viene eseguito il processo preliminare dei testi. Sui documenti di testo tradizionali (senza tag HTML), i compiti da eseguire sono rimozione di *stop word*, manipolazione dei *numeri*, i segni di *punteggiatura* ecc. Le pagine web sono differenti dai documenti di testo tradizionali, per le quali occorrono dei processi preliminari delle pagine. Tra i quali sono identificazione dei vari campi di testo, identificazione delle ancore di testo, rimozione dei tag HTML, e identificazione dei blocchi di contenuti principali.

3.2.1 Analisi lessicale

L'*Analisi lessicale* è il processo che attraverso il quale un testo viene scandito per rilevare ed estrarre le stringhe che sono potenzialmente parole chiave o termini. Le parole vengono rilevate ad ogni carattere di separazione come spazi e simboli di punteggiatura. Lo strumento per la scansione del testo si chiama *analizzatore lessicale*. Quest'ultimo dipende dalla lingua, perché le parole sono scritte con alfabeti diversi e possono esserci diversi caratteri di separazione. Il riconoscimento delle parole è una condizione necessaria, ma non è sufficiente per riconoscere i descrittori a causa della varietà di modi per scrivere le parole anche quando rappresentano la stessa informazione.

Stop word. Dal fatto che alcune parole sono presenti in quasi tutti i documenti, mediante l'analisi lessicale alcune parole vengono rimosse, queste parole sono stop word. Nel caso in cui queste parole venissero lasciate, si determinerebbe un'indicizzazione eccessivamente esaustiva. Le stop word possono essere come dei segni di punteggiatura che separano (*stop*) le altre parole, che siano poco o non specifiche. Le stop word sono solitamente preposizioni e articoli, che genericamente sono tutte queste parole molto frequenti nella collezione. Un insieme di stop word viene chiamato **stop list**.

Descrittore. In un testo, si assume che un *descrittore* sia un dato che esprime gli aspetti salienti del contenuto informativo di un documento, possiede inoltre un significato analogo di *parola chiave* (*keyword*) o *termine indice* (*index pattern*).

Parola chiave. L'accezione *parola chiave* si riferisce a singole parole con un ruolo chiave nella descrizione del contenuto. Ogni stringa "sopravvissuta" alla rimozione delle stop word è un potenziale descrittore o un suo componente. Tuttavia, le parole diverse possono avere un significato simile e la stessa radice linguistica come "informazione" e "informativo".

Posting. Un *posting* è l'assegnazione di un descrittore ad un documento, tale nozione è simile a quelle di etichetta o tag comunemente impiegate per descrivere il contenuto informativo dei documenti disponibili solitamente mediante il web. Una **posting list** è una struttura informativa, residente nella memoria dell'IRS, che raccoglie il posting list è la lista degli identificatori dei documenti in cui è presente una certa parola chiave.

Stemming. Lo *Stemming* è l'identificazione della radice di una parola: di fatto "*stem*" vuol dire stelo, e in questo senso lo stem di una parola è lo stelo su cui si applicano le varianti che danno luogo alle derivazioni linguistiche della parola. Pertanto l'IRS deve riconoscere le varianti e ottenere lo stelo, ovverosia la radice. Mediante lo stemming, per ciascuna di queste parole viene costruita una stringa, detta **stem**, che ne rappresenta la radice linguistica. In questo modo le parole che provengono dalla stessa radice linguistica sono ricondotte allo stesso stem.

Tokenizzazione. La *tokenizzazione* è una funzione, tale che data una sequenza di caratteri e unità di documenti, lo si taglia in pezzi chiamati **token**, e allo stesso tempo elimina certi caratteri. Quando il gruppo di token che costituisce un termine è sostenuto da una struttura grammaticale, la costruzione dei termini passa attraverso la cosidetta etichettatura (*tagging*) delle parti del discorso (*Part-of-Speech* (POS)) mediante cui ogni parola è etichettata da una cate-

goria lessicale: quest'ultima è il ruolo svolto da una parola viene etichettata da una categoria lessicale: quest'ultima è il ruolo svolto da una parola in determinati frammenti di testo.

Termine. Un *termine* è definito come un gruppo o parole (in generale *token*). Nel caso di gruppo ordinato, un termine è una semplice lista di token oppure è una frase sostenuta da una struttura grammaticale in cui ogni token ha un ruolo. Nel caso di gruppo disordinato, i gruppi di token sono trattati come se fossero lo stesso gruppo indipendentemente dall'ordine. L'accezione **termine indice** si riferisce a gruppi di due o più parole chiave o a frasi legate da strutture grammaticali. Una volta etichettati i token, i termini sono costruiti associando un aggettivo a un sostantivo oppure un sostantivo a un sostantivo; l'ordine dei token è essenziale ad esempio "information retrieval" ha un significato diverso da quello di "retrieval information". L'etichettatura delle parti del discorso può raggiungere buoni livelli di precisione, almeno per le lingue più diffuse; resta però il problema di trovare l'etichetta giusta per le parole meno note, isolate dal resto del testo o poste all'interno di testi molto brevi come i titoli.

3.2.2 Pre-Processo delle pagine web

Il processo preliminare delle pagine web vengono trattati in maniera differente rispetto ai documenti tradizionali. Il quale necessita del processo addizionale. Alcune tipologie di pre-processo più importanti sono riportati sotto:

- **Identificazione dei vari campi di testo:** In HTML vi sono vari campi di testo, come per esempio titolo, metadati e corpo. La loro identificazione permette al sistema di reperimento a trattare i termini dei vari campi in modo differente. Per esempio nei motori di ricerca i termini che appaiono nel campo del titolo di una pagina vengono considerati più importanti dei termini che appaiono in altri campi e vengono assegnati pesi maggiori perché il titolo è solitamente una descrizione concisa della pagina. Nel testo del corpo ai termini enfatizzati vengono dati un peso maggiore (i tags <h1>, <h2>,..., <b>, ecc.).

- **Identificazione dei testi ancorati:** I testi ancorati associati all'iperlink vengono trattati con una particolare importanza nei motori di ricerca perché il testo ancorato spesso rappresenta una descrizione più accurata di un informazione contenuta nella pagina a cui suo link punta. Il caso che l'iperlink punti ad una pagina esterna (ovvero non punta sui contenuti della pagina corrente), è una situazione particolarmente considerevole perché fornisce una descrizione di riepilogo riguardo alla pagina data dalle altre persone invece di autore/proprietario della pagina corrente, ed è pertanto un fatto più affidabile.
- **Rimozione dei tags HTML:** La rimozione dei tags può essere affrontata mediante il calcolo del punteggio di similarità. L'HTML è una presentazione visuale del linguaggio. Nella tipica pagina commerciale, l'informazione è presentata in molti blocchi rettangolari. Rimuovendo semplicemente i tags HTML potrebbero causare problemi di accoppiamento che potrebbe non essere congiunto, i quali possono causare problemi per le frasi di interrogazione e la somiglianza all'interrogazione.
- **Identificazione dei blocchi principali nei contenuti:** Una pagina web tipica, in particolare quella commerciale, contiene una grande quantità di informazioni che non fa parte del contenuto principale della pagina. È stato dimostrato che i risultati di ricerca e di data mining possono essere migliorate notevolmente se vengono utilizzati solamente i blocchi principali nei contenuti.

Tree matching: Il metodo *tree matching* si basa sul fatto che la maggior parte dei siti web commerciali sono stati generati facendo uso di alcuni modelli prefissati. Pertanto il suo obiettivo è cercare tali modelli nascosti. Dato che HTML possiede una struttura annidata, è pertanto facile costruire un albero di tag per ciascuna pagina. L'associazione all'albero di pagine multiple può essere svolta facendo ricerca di tali modelli.

3.3 Indicizzazione e Reperimento

In un sistema IR, qualsiasi richiesta di informazione viene elaborata facendo uso dell'indice. L'*indice* è una struttura informativa che

raccoglie le posting list relative ad un insieme dato di descrittori. L'*indicizzazione* è l'unico modo per rendere il reperimento efficiente e allo stesso tempo efficace. Se un documento non è indicizzato allora non può essere reperito. Gli strumenti utilizzati per l'indicizzazione hanno un impatto diretto sui modelli e metodi di reperimento, che di conseguenza anche sugli esiti della valutazione di un sistema IR. I problemi dell'indicizzazione riguardanti le pagine web sono eterogenei per la sua vastità di contenuti, struttura, media e lingue. I metodi di indicizzazione sono metodi di base per la rappresentazione del contenuto informativo dei documenti di una collezione e delle interrogazioni poste dagli utenti finali, sono le soluzioni principali per la realizzazione degli indici. Infine, spiega i meccanismi essenziali di un agente di ricerca per il web. Mediante i descrittori, un sistema IR rappresenta il contenuto informativo di ciascun documento della collezione. L'insieme dei descrittori è l'unico insieme a disposizione dell'IRS per descrivere il contenuto informativo. L'*alimentazione* di una collezione, ovvero l'aggiunta dei documenti, può essere manuale, semi-automatica o automatica a seconda del grado di controllo effettuato dal personale specializzato.

3.3.1 Indici Invertiti

In IR, il metodo base è quello di trovare documenti che contengono termini indicati nell'interrogazione utente.

Dato un insieme di documenti $d = \{d_1, d_2, ..., d_N\}$, e ciascun documento possiede un identificatore unico (ID). L'indice invertito si consiste in due parti: sia V un vocabolario, tale che contiene tutti i termini distinti nell'insieme di documenti, e per ciascun termine t_i distinto vi è una **lista invertita** di postings. Ogni posting è associato ad un'ID (id_j) del documento d_j che contiene termine t_j ed altri pezzi di informazioni riguardo al termine t_i nel documento d_j. In base alle condizioni necessarie per il reperimento, possono essere incluse varie tipologie d'informazione. In generale un posting per il termine t_i si consiste nella forma seguente:

$$\langle id_j, f_{ij}, [o_1, o_2, ..., o_{f_{ij}}] \rangle \tag{3.1}$$

dove id_j è l'ID del documento d_j che contiene il termine t_i, f_{ij} è il conteggio della frequenza di t_i in d_j, e o_k è la posizione del termine

t_i in d_{ij}. I postings di un termine è ordinato in modo crescente, l'ordine è basata su l'id_j, ed è così disposta anche ciascun posting.

Esempio: Abbiamo a disposizione tre documenti: id_1, id_2, *e* id_3

$\mathbf{id}_1 : \underset{1}{Web}\ \underset{2}{mining}\ \underset{3}{is}\ \underset{4}{useful}$

$\mathbf{id}_2 : \underset{1}{Usage}\ \underset{2}{mining}\ \underset{3}{applications}$

$\mathbf{id}_3 : \underset{1}{Web}\ \underset{2}{structure}\ \underset{3}{mining}\ \underset{4}{studies}\ \underset{5}{the}\ \underset{6}{Web}\ \underset{7}{hyperlink}\ \underset{8}{structure}$

I numeri sotto ciascun documento è l'offset di posizione di ciascuna parola.
Il vocabolario è l'insieme:
{Web, mining, useful, applications, usage, structure, studies, hyperlink}
Dove gli stopwords "*is*" e "*the*" sono stati rimossi, ma non è applicata lo stemming.
La fig. 3.3.1 mostra due indici invertiti.

Applications:	id_2	Applications:	<id_2, 1, [3]>
Hyperlink:	id_3	Hyperlink:	<id_3, 1, [7]>
Mining:	id_1, id_2, id_3	Mining:	<id_1, 1, [2]>, <id_2, 1, [2]>, <id_3, 1, [3]>
Structure:	id_3	Structure:	<id_3, 2, [2, 8]>
Studies:	id_3	Studies:	<id_3, 1, [4]>
Usage:	id_2	Usage:	<id_2, 1, [1]>
Useful:	id_1	Useful:	<id_1, 1, [4]>
Web:	id_1, id_3	Web:	<id_1, 1, [1]>, <id_3, 2, [1, 6]>

(A) (B)

Fig. (3.3.1: Due versioni di indici invertiti, una semplice e una complessa)

La fig. 3.3.1(A) è una versione semplice, dove ad ogni termine è allegato a solamente l'inverted list di $\mathrm{ID_s}$ del documento che contiene il termine. Mentre ciascun inverted list nella fig. 3.3.1(B) contiene informazioni aggiuntivi come la frequenza e le posizioni del termine in ogni documento. È usata la notazione id_i come documento $\mathrm{ID_s}$ per distinguere da altri offsets.

Ricerca mediante l'uso di indici invertiti. In primo luogo le interrogazioni vengono valutate afferrando le liste invertite dei termini delle interrogazioni. Dopodiché avviene il processo di ricerca dei documenti, i quali contengono tutti i termini. Date le interrogazioni di termini, la ricerca dei documenti in indici invertiti si consiste in seguenti principali passi:

1. **Vocabulary search**: In questo passo si cerca il termine dell'interrogazione, tale che restituisce la lista invertita di ciascun termine. Per accelerare la ricerca, il vocabolario generalmente risiede nella memoria principale. Vari metodi di indicizzazione come *hashing* e *B-tree* possono essere applicati per finalizzare la ricerca. Può anche essere impiegato l'ordine lessicografico per la sua efficienza di spazio, pertanto può essere applicato il metodo della ricerca binaria. La complessità è $O(\log|V|)$, dove $|V|$ è la taglia del vocabolario. Nel caso in cui l'interrogazione contiene solamente un termine singolo, restituisce tutti i documenti rilevanti e l'algoritmo passa al passo 3. Se invece l'interrogazione contiene query multipli, si procede al passo 2.
2. **Fusione dei risultati**: Dal momento in cui sono state trovate le liste invertite di ciascun termine la fusione delle liste viene eseguita per trovare un'intersezione. Per la fusione si naviga attraverso tutte le liste in sincronizzazione per verificare se ciascun documento contiene tutte le interrogazioni termini. Una euristica è fare uso della lista più corta come base per poi fondere con le altre più lunghe. Per ciascun posting nella lista più corta è possibile applicare la ricerca binaria.
3. **Punteggio di rango**: In questo passo si calcola il punteggio di rango per ogni documento basando su una funzione di rilevanza (per esempio il metodo della similarità di *coseno*), tale che può anche considerare la frase ed vicinanza dell'informazione del termine.

3.3.2 Algoritmi di Indicizzazione

L'accesso all'indice consiste essenzialmente di due passi: prima si accede al dizionario per reperire l'elemento cercato e poi si accede al file invertito. Per ogni descrittore di un indice, un indirizzo "punta" ad una *posting list* costituita da una lista di URL. Nel caso in cui la frequenza del descrittore nel documento sia uguale ad uno, l'indirizzo della posting list "punta" direttamente all'identificatore del documento; altrimenti punta ad un record di informazioni, come la frequenza del descrittore nel documento.

Algoritmi 1 e 2. Gli algoritmi che scandiscono ogni documento della collezione e aggiorna la *posting list* ad ogni descrittore del documento scandito.

Algorithm 1 Aggiunta di un descrittore ad un indice I

Require: descrittore w, documento j
1: aggiungi(w, j, I)
2: **if** $w \notin I$ **then**
3: aggiungi w al dizionario di I
4: **end if**
5: aggiungi j alla posting list di w
6: **return**

Algorithm 2 Aggiunta di un documento ad un indice I

Require: documento j, indice I
1: aggiungi(j, I)
2: **for all** descrittore $w \in j$ **do**
3: aggiunti(w, j, I)
4: **end for**
5: **return**

Algoritmo 3. Un semplice algoritmo d'indicizzazione. Per evitare i colli di bottiglia causati dalla dimensione del file invertito, si costruiscono diversi indici invertiti che scaricano su disco quando la memoria a disposizione per ciascuno di essi è esaurita; si scrive il file su disco lo si svuota e si va avanti a riempirlo fino a quando lo si deve riscrivere su disco.

Algorithm 3 Indicizzazione di una collezione (1)

Require: collezione J, indice I
1: indicizza(J, I)
2: **while** esiste un $j \in J$ **do**
3: **while** esiste un $j \in J$ e I non è pieno **do**
4: aggiungi(j, I)
5: **end while**
6: scarica I su disco
7: svuota I
8: **end while**
9: **for all** indice I' scaricato su disco **do**
10: $I \leftarrow$ fondi(I, I')
11: **end for**
12: **return**

Algoritmo 4. Con questo algoritmo si mantengono indici diversi in parallelo che vengono riempiti in modo pseudo-casuale, come passando da un indice all'altro ad ogni documento. Al termine, si hanno diversi indici la cui dimensione è fissata in modo che la fusione di due di essi possa procedere sempre in memoria centrale e con pochi accessi al disco.

Algorithm 4 Indicizzazione di una collezione (2)

Require: collezione J, indici $I, I_0, ..., I_{m-1}$
1: indicizza(J, I)
2: $i \leftarrow 0$
3: **for all** documento $j \in J$ **do**
4: $r \leftarrow$ resto della divisione i/m
5: aggiungi(j, I_r)
6: $i \leftarrow i + 1$
7: **end for**
8: $I \leftarrow$ fondi$(I_0, ..., I_{m-1})$
9: **return**

3.3.3 Agente di Ricerca

Un agente di ricerca è un programma che, dato un URL, legge la pagina dal *web server* e passa ai programmi d'indicizzazione i dati contenuti nella pagina. L'agente estrae gli URL contenuti nella pagina e continua a leggere le pagine collegate, naviga il web seguendo i links verso altre pagine. Per riconoscere i link e quindi navigare da una pagina all'altra, l'agente implementa un analizzatore lessicale, in modo che sia in grado di stabilire se una stringa di testo è quello di un URL o meno.

Il punto di partenza (*pagina seme)* è una pagina da cui iniziare la scansione del web, tale che la sua scelta è determinare il funzionamento complessivo dell'agente. La strategia di ricerca di un agente stabilisce l'ordine di reperimento e trattamento delle pagine. Il metodo di determinazione del punto di partenza può essere manuale, semi-automatico o automatico. I metodi manuali sono efficaci, ma anche dispendiosi per l'investimento intellettuale umano necessario. I metodi automatici sono meno dispendiosi in termini di costo dell'intellettuale umano, però sono più imprecisi, sebbene possano essere ripetuti in modo efficiente fino al ritrovamento delle pagine migliori. Essi possono essere basati sulla generazione ca-

suale di URL, oppure sulla scoperta automatica di pagine che sono probabilmente dei buoni punti di partenza da cui iniziare la raccolta. Si suppone di inviare una serie di interrogazioni relative ad una tematica ad un IRS che restituisce degli insiemi di pagine probabilmente rilevanti che costituiscano un'insieme radice. Per ciascuna pagina dell'insieme p si inseriscono nello stesso insieme quelle che hanno un link verso p. Il risultato è un insieme base che contiene pagine probabilmente rilevanti alle interrogazioni e, pagine che sono collegate a quelle rilevanti.

Esplorazione in Ampiezza. Sul web, non tutti i link portano a pagine meritevoli d'essere indicizzate e, inoltre possono esistere più di un link da seguire. La decisione di seguire un link è basata su euristiche per esclusione. Nel corso di visita in ampiezza, data una pagina, le pagine collegate direttamente sono elaborate prima di quelle collegate indirettamente. L'agente mette in coda le pagine da elaborare secondo un criterio *First-In First-Out (FIFO)* e le pagine sono elaborate nello stesso ordine in cui sono stati trovati i link ad esse. Quando una pagina è stata elaborata, esce dalla coda.

Algoritmo 5. Il programma che implementa l'algoritmo mantiene una propria coda Q. Dopo aver accodato il punto di partenza, l'agente ripete le operazioni (ciclo 5-12) che consistono nel prelevare il primo URL della coda, estrarre gli URL presenti nella pagina indirizzata e accodarli in Q. L'algoritmo termina quando Q si svuota, ciò accade quando si eseguono tanti POP quante sono le pagine in coda, senza alcun accodamento. Per evitare che l'algoritmo vada avanti all'infinito quando trova un "cappio" dato da un link da u a v ed un altro da v a u , è necessario che il programma tenga traccia in una lista V delle pagine già elaborate, controllando che la pagina non sia in V ad ogni chiamata di URLs.

Visita in profondità. Con la visita in profondità, data una pagina, le pagine sono elaborate non appena si trova il loro URL nella pagina corrente: l'agente mette in pila le pagine da elaborare secondo un criterio *Last-In Firt-Out* (LIFO) e le pagine sono elaborate

Algorithm 5 Algoritmo d'esplorazione in ampiezza

Require: s: URL di una pagina "punto di partenza"
Require: Q: coda di URL di pagine da elaborare
Require: URLS(u): insieme di URL in u
Require: POP(Q): restituisce il primo di Q
Require: QUEUE(u, Q): accoda u a Q

```
1: Q ← 0
2: for all s do
3:     QUEUE(s, Q)
4: end for
5: while Q ≠ 0 do
6:     u ← POP(Q)
7:     U ← URLS(u)
8:     qui si elabora il resto del contenuto di u
9:     for all r ∈ U do
10:        QUEUE(r, Q)
11:    end for
12: end while
```

in ordine inverso rispetto a quello in cui sono stati trovati i link ad esse.

Algoritmo 6. L'implementazione è simile a quella di visita in ampiezza. Il programma che implementa l'algoritmo mantiene una propria pila S. Dopo aver inserito il punto di partenza nella pila, l'agente esegue un ciclo di operazioni che consiste nel prelevare il primo URL, estrarre gli URL presente nella pagina indirizzata ed aggiungere gli URL a S. Quando si esegue una serie di estrazioni POP senza inserire alcun URL, S si svuota. Anche questo algoritmo può andare avanti all'infinito e ciò accade quando esiste un "cappio" tra due pagine. Per evitare ciò anche in questo caso è necessario tenere traccia in una lista V delle pagine già elaborate e controllare che la pagina non sia già in V a ogni chiamata di URLs.

L'eventualità che l'agente segua un numero elevato di link e aumenti così il carico computazionale rende necessario un criterio di fermata. Quest'ultimo è basato sull'ampiezza massima o sul numero di pagine raccolte, che consiste banalmente nel porre un limite al numero di link da seguire a partire dalla pagina corrente in modo analogo funziona il criterio basato sulla profondità.

Algorithm 6 Algoritmo d'esplorazione in profondità

Require: s: URL di una pagina "punto di partenza"
Require: S: pila di URL di pagine da elaborare
Require: URL(u): insieme di URL in u
Require: POP(S): restituisce il primo di S
Require: PUSH(u, S);impila u a S

```
1: S ← 0
2: for all s do
3:     PUSH(s, S)
4: end for
5: while S ≠ 0 do
6:     u ← POP(S)
7:     U ← URLS(u)
8:     qui si elabora il resto del contenuto di u
9:     for r ∈ U do
10:        PUSH(r, S)
11:    end for
12: end while
```

3.3.4 Livello di Coordinamento

Il *livello di coordinamento* è una misura di quanto l'interrogazione sia vera per un documento. In generale, se si ha un'interrogazione costituita da una *forma normale disgiuntiva* (FND), ossia, una disgiunzione di m proposizioni ciascuna delle quali è costituita dalla congiunzione di un certo numero di proposizioni atomiche $x_{i,j}$, pesate con $w(x_{i,j})$, e scritta come:

$$\underbrace{\underbrace{(x_{1,1} \wedge \ldots \wedge x_{1,\mathrm{n}_1})}_{\text{congiunzione di } \mathrm{n}_1 \text{ proposizioni}} \quad \vee \ldots \vee \quad \underbrace{(x_{\mathrm{m},1} \wedge \ldots \wedge x_{\mathrm{m},\mathrm{n}_\mathrm{m}})}_{\text{congiunzione di } \mathrm{n}_\mathrm{m} \text{ proposizioni}}}_{m \text{ disgiunzioni}}$$

il livello di coordinamento pesato di un documento è

$$max\left\{\sum_{j=1}^{n_1} w(x_{1,\mathrm{j}}), ..., \sum_{j=1}^{n_m} w(x_{\mathrm{m,j}})\right\}$$

dove $x_{i,j}$ è il j-esimo descrittore dell'i-esima congiunzione e $w(x_{i,j})$ è il peso del descrittore presente nel documento. Nel caso in cui un'interrogazione non sia FND, l'IRS procede con una trasformazione sfruttando la proprietà associativa degli operatori logici, l'interrogazione:

(sistema OR rilevanza) AND (sistema OR NOT rango)

viene trasformata nella proposizione:

$$\text{sistema OR (rilevanza AND NOT rango)} \tag{3.2}$$

e il livello di coordinamento pesato dell'interrogazione (3.3) diventa

$$\max\{w(\text{sistema}),\ w(\text{rilevanza}) + w(\text{NOT rango})\}$$

In termini sperimentali, l'efficacia del livello di coordinamento è accettabile, specialmente con collezioni di record catalografici o documenti descritti con meta-dati, ma è meno soddisfacente per collezioni eterogenee o molto grandi. La ragione fondamentale i questi limiti è dovuta al fatto che la nozione di livello di coordinamento è euristica e non permette d'affrontare gli errori del sistema ad un livello di astrazione sufficientemente alto senza fare a meno di provare, uno dopo l'altro, diversi pesi per ciascun descrittore.

3.3.5 Algoritmi di Reperimento

Uno schema di reperimento può essere quello definito nell'Algoritmo 7, dove ad ogni interrogazione data in ingresso, si esegue questo algoritmo, il passo cruciale è il terzo, che per tale vi sono due possibili approcci: *Term At A Time* (*TAAT*) e *Document at time* (*DAAT*).

Algorithm 7 Schema di un Algoritmo di reperimento

Require: interrogazione m descrittori $\{q_1, ..., q_m\}$
1: recupera le *posting list* di ciascun descrittore q_j
2: calcola la misura di rilevanza per ciascun documento
3: **return** documenti con la misura maggiore

Term At A Time (TAAT). Si accede alla posting list di ciascun descrittore dell'interrogazione alla volta, dopodiché si elaborano tutti i posting list di tutti i descrittori e si aggiornano i punteggi dati ai documenti. L'algoritmo di questo tipo di reperimento, è caratterizzato dalla presenza di operatori logici, i quali sono *intersezione, unione e differenza* tra gli insiemi. Per la loro implementazione si usano le funzioni POTA(L, n) e FONDI(L_{q1}, L_{q2}). La prima è la funzione che rimuove gli elementi che appaiono n volte in una posting list, mentre la seconda è la funzione che fonde le posting list date in input, ripetendo gli elementi che appaiono in ambedue le liste.

Algoritmo 8. Con questo algoritmo si sceglie il descrittore meno frequente affinché si arrivi prima all'eventualità in cui $C = 0$ e la funzione POTA possa lavorare su un C più piccolo.

Algorithm 8 TAAT con congiunzioni

1: **for** $j = 1, ..., m$ **do**
2: $L_j \leftarrow$ posting list di q_j
3: $n_j \leftarrow$ taglia di L_j
4: **end for**
5: $j' \leftarrow \arg_{j=1,...,m} \min_{n_j}$
6: $C \leftarrow L_{j'}$
7: **for** $j = 1, ..., m, j \neq j'$ **do**
8: $C \leftarrow \text{FONDI}(C, L_j)$
9: $C \leftarrow \text{POTA}(C, 1)$
10: **if** $C = \varnothing$ **then**
11: **return**
12: **end if**
13: **end for**
14: **return** C

Algoritmo 9. In questa implementazione invece si sceglie il descrittore più frequente affinché la funzione FONDI lavori su posting list non più lunghe di C.

Algorithm 9 TAAT con disgiunzioni

1: **for** $j = 1, ..., m$ **do**
2: $L_j \leftarrow$ posting list di q_j
3: $n_j \leftarrow$ taglia di L_j
4: **end for**
5: $j' \leftarrow \arg_{j=1,...,m} \max_{n_j}$
6: $C \leftarrow L_{j'}$
7: **for** $j = 1, ..., m, j \neq j'$ **do**
8: $C \leftarrow \text{FONDI}(C, L_j)$
9: **end for**
10: **return** C

Algoritmo 10. I descrittori sono pesati con dati registrati durante l'indicizzazione; il costo maggiore dell'algoritmo è dato dal ciclo 7-12 che rischia di richiedere tanti passi quanti sono i *postings* della lista più lunga se non c'è nessun criterio di terminazione anticipata.

Document At A Time (DAAT). Si accede alla posting list di tutti i descrittori dell'interrogazione in una volta sola e si scorro-

Algorithm 10 TAAT con pesatura

Require: $A = a_1, ..., a_n$ accumulatori
1: **for** $j = 1, ..., m$ **do**
2: $L_j \leftarrow$ posting list di q_j
3: $n_j \leftarrow$ taglia di L_j
4: $w(q_j) \leftarrow$ taglia di L_j
5: **end for**
6: **for** $j = 1, ..., m, j \neq j'$ **do**
7: **for all** posting i presente in L_j **do**
8: **if** $a_i \notin A$ **then**
9: $A \leftarrow A \cup \{a_i\}$
10: **end if**
11: $a_i \leftarrow a_i + w(q_j) f(i, j)$
12: **end for**
13: **end for**
14: normalizzazione di A
15: **return** documenti con i maggiori accumulatori di A

no i documenti delle posting list in parallelo. L'algoritmo si basa sull'allineamento di due o più posting list facendo coincidere gli identificatori di documento ordinati in modo crescente in ciascuna posting list. Per poter sfruttare la prossimità delle parole, i postings registrano anche la posizione, e le posizioni all'interno di ciascun documento sono ordinate in modo crescente. Nell'algoritmo vengono definite le seguenti funzioni: INIZIA(L), ATTUALE(L) e PROSSIMO(L). INIZIA(L) posiziona un cursore C all'inizio della posting list L prima del primo elemento. ATTUALE(L) ritorna il posting indicato dal cursore di L. PROSSIMO(l) sposta il cursore al posting successivo.

Algoritmo 11. Si assume che un posting sia un identificatore del documento, per cui PROSSIMO(L) ritorna l'identificatore di documento successivo a quello indicato dal cursore di L. Il ciclo 1-4 recupera le posting list e pone un cursore all'inizio di ciascuna di esse. Nella riga 5 si determina la posting list in cui il primo documento ha l'identificatore più piccolo. Tenendo presente che una posting list è ordinata per identificatore di documento, nel caso in cui si sia alla fine di tutte le posting list, c non esiste e l'elaborazione termina. Si riprende poi il primo identificatore di documento disponibile nella c-esima posting list (riga 7). Dopodiché aggiorna l'accumulatore del documento con identificatore i (riga 8) e deter-

mina la posting list in cui il prossimo documento ha l'identificatore più piccolo (riga 9). Sarà certamente una posting list diversa da quella appena elaborata, ma l'identificatore di documento potrebbe essere uguale al valore attuale di i se il documento contiene due o più descrittori.

Algorithm 11 Reperimento DAAT

Require: Reperimento m accumulatori
1: **for** $j = 1, ..., m$ **do**
2: $L_j \leftarrow$ posting list di q_j
3: INIZIA(L_j)
4: **end for**
5: $c \leftarrow \arg_{j=1,...,m} min$ PROSSIMO(L_j)
6: **while** c esiste **do**
7: $i \leftarrow$ ATTUALE(L_c)
8: $a_i \leftarrow a_i + w(c)f(i,c)$
9: $c \leftarrow \arg_{j=1,...m} \min$ PROSSIMO(L_j)
10: **end while**

Gli algoritmi di reperimento di tipo TAAT sono più comunemente utilizzati negli IRS tradizionali. Per piccole collezioni di documenti, le implementazioni di questi algoritmi sono eleganti e hanno buone prestazioni. Per collezioni di grandi dimensioni invece gli algoritmi di reperimento di tipo DAAT richiedono meno memoria durante l'esecuzione, perché non c'è bisogno di mantenere gli accumulatori. E inoltre sfruttano l'*input/output* dei dati in modo più efficiente grazie al naturale parallelismo della scansione di due o più posting lists. Il reperimento sia di tipo TAAT che di tipo DAAT possono essere ottimizzate in modo significativo se si accettano delle approssimazioni delle misure di rilevanza e si rinuncia ad avere un valore esatto. Ciò che è d'interesse è l'ordinamento dei documenti piuttosto che il valore della funzione di reperimento, spesso incomprensibile per l'utente finale. Una strategia di ottimizzazione degli algoritmi di reperimento di tipo DAAT è max-score, dove la misura di rilevanza di un documento non viene più aggiornata quando è chiaro che il documento non comparirà tra i primi della lista. Un'altra strategia introduce un criterio di terminazione anticipata ed è mostrata in algoritmo 12, questa strategia si basa sul caso in cui il documento posto al numero di rango $K + 1$ non potrebbe migliorare il proprio numero di rango, ossia passare a un numero di rango inferiore, anche se tutti i descrittori dell'interrogazione che restano da elaborare

fossero presenti nel documento.

Algorithm 12 Reperimento TAAT con terminazione anticipata

Require: interrogazione di m descrittori $q_1, ..., 1_m$
Require: insieme A di accumulatori a_i per gli n documenti
Require: numero K di documenti da reperire
Require: Schiera T di $K+1$ documenti con gli accumulatori più alti, mantenuta ordinata in senso non crescente

```
1: for j = 1, ..., m do
2:     L_j ← posting list di q_j
3:     n_j ← taglia di L_j
4:     w(q_j) ← peso di q_j
5: end for
6: Ordinare i descrittori per w decrescente per velocizzare la terminazione anticipata
7: k ← 1
8: for j = 1, ..., m do
9:     for all posting i presente in L_j do
10:        if ai ∉ A then
11:            A ← A ∪ {a_i}
12:        end if
13:        a_i ← a_i + w(q_j) f(i, j)
14:        if k ≤ K ∨ a_T[k] < a_i then
15:            if i non è tra gli elementi di T then
16:                T[k] ← i
17:                k ← k + 1
18:            end if
19:            riordina T in senso non crescente
20:        end if
21:    end for
22:    if k ≥ K + 1 then
23:        s_T[K + 1] ≤ a_T[K+1] ottenibile con i descrittori ancora da elaborare
24:        if s_T[K+1] ≤ a_T[K+1] then
25:            return T[1],...,T[K]
26:        end if
27:    end if
28: end for
```

Algoritmo 12 I due passi cruciali dell'algoritmo sono le righe 14-20 e 22-27, dove si inserisce un nuovo documento in lista e quello in cui si calcola il massimo possibile del documento $T[K+1]$; il calcolo dipende dalla ponderazione ed informazioni raccolte nelle posting lists.

3.4 Ricerca sul Web

Per descrivere un motore di ricerca si inizia partendo dalla scansione delle pagine sul web, dove le pagine scansionate vengono convertite, indicizzate e memorizzate. Poi va usato un convertitore per convertire la pagina HTML in entrata, tale che produce un flusso di token o termini per l'indicizzazione. L'**indicizzazione** è un passo in cui si elabora l'*indice invertito*. Per l'efficienza del reperimento, con un motore di ricerca è possibile costruire indici invertiti multipli. I titoli e testi ancorati sono spesso delle descrizioni molto accurate delle pagine. L'indice invertito di dimensione piccola potrebbe essere costruito basandosi su termini apparsi in essi solitariamente. È da notare che qui il testo ancorato è per l'indicizzazione delle pagine, le direzioni per le quali i link puntano, e non la pagina che la contiene. Un indice completo è allora costruito basando su tutti i testi in ciascuna pagina, includendo i testi ancorati. Nella ricerca, l'algoritmo potrebbe cercare per primo l'indice di cardinalità minore e poi in quello completo. Se un numero sufficiente di pagine rilevanti sono stati trovati negli indici parziali allora il sistema potrebbe non effettuare la ricerca nell'indice completo. Data un'interrogazione utente, la ricerca coinvolge i seguenti passi:

1. Pre-processare l'interrogazione termini (rimozione *stopword* e *stemming*);
2. ricerca di pagine che contengano tutte (o la maggior parte delle) interrogazioni di termini negli indici invertiti;
3. classificare le pagine e restituirli all'utente.

L'**algoritmo di classificazione** è il cuore di un motore di ricerca. Tuttavia in generale si sanno poco di algoritmi di ricerca usati nei motori di ricerca commerciali. L'IR tradizionale utilizza il **coseno di similarità** (o qualche altra misura di similarità) per la classificazione dei documenti. Queste misure considerano solamente il contenuto di ciascuna pagina. Nell'ambito del web, tali metodi basati sul contenuto non sufficienti. Il problema che si riscontra su quello del web è che vi sono troppi documenti rilevanti per quasi tutte le interrogazioni. Chiaramente non vi sono modi che qualunque utente possa fare ricerca su questa quantità di numero di pagine. Allora il problema diventa il modo con cui classificare le

pagine e presentare la pagina migliore al top. Uno dei fattori più importanti sul web è la qualità delle pagine, tale per cui potrebbe essere difficilmente studiata in IR tradizionali, perché la maggior parte dei documenti usati in valutazioni IR provengono da risorse affidabili. Tuttavia, sul web, chiunque può pubblicare qualsiasi cosa, e non vi è un controllo di qualità. Sebbene una pagina possa essere rilevante al 100%, ma potrebbe non essere di qualità a causa di diverse ragioni che concorrono. Una pagina web può essere valutata basando sui contenuti dei fattori e la sua reputazione. La pagina web può essere valutata basando su entrambi i fattori del contenuto e la sua reputazione. La valutazione basata sul contenuto può dipendere da tre tipologie di informazioni:

- **Occurrence type:** Vi sono diversi tipi di occorrenze di interrogazioni di termini in una pagina, i quali sono Titolo, testi ancorati, URL e Corpo.
- **Conteggio:** Il numero di occorrenze di termine di ciascun tipo. Come per esempio un termine query potrebbe apparire per due volte. Allora il conteggio per il termine è 2.
- **Posizione:** La posizione di ciascun termine in ogni tipo di occorrenza. L'informazione è usata in prossimità di valutazione coinvolgendo interrogazioni termini multipli. I termini query quelli uni vicini altri sono meglio rispetto a quelli che sono distanti. Inoltre, i gruppi query apparendo nella pagina nella stessa sequenza come essi sono nell'interrogazione sono anche meglio.

Per il calcolo basato sul punteggio del contenuto (chiamato anche *information retrieval score*), ad ogni tipo di occorrenza è dato un peso associato. Tutti i vettori peso formano un vettore fissato. Ogni conteggio di termine grezzo viene convertito al conteggio peso, e tutti i pesi conteggio formano un vettore. La qualità o reputazione di una pagina è in genere calcolata basando su strutture di collegamento delle pagine web.

3.4.1 Motore di Ricerca

In un sistema IR tradizionale si assume che le unità di informazioni siano i documenti, e l'immensa *collezione di documenti* sia disponibile per formare i databases di testi. Sul web, analogamente, le **pagine**

web sono viste come dei documenti. Reperire informazioni significa trovare un'insieme di documenti che sono rilevanti per l'interrogazione utente. In senso generale, la classificazione degli insiemi di documenti è strutturata basandosi sul loro punteggio di rilevanza. Il formato di interrogazione più comunemente usato è la lista di **parole chiavi** (o **termini**). L'IR è differente dal reperimento dei dati in basi di dati usando i comandi *query*, dove i dati sono altamente strutturati e immagazzinati nelle tabelle relazionali, mentre le informazioni in testi non sono strutturati. Non vi è un linguaggio di interrogazione per il reperimento dei testi. Il *motore di ricerca* per il web è un caso particolare e semplificato di un sistema di reperimento (IRS). Un IRS come il motore di ricerca conosce in ogni istante la propria collezione di documenti. È lecito affermare che la ricerca sul web è l'applicazione più importante dell'IR. I motori di ricerca presentano alcuni aspetti particolari che si distinguono dagli IRS tradizionalmente intesi.

I documenti sono le pagine web e non necessariamente prodotti di un processo editoriale come i libri o gli articoli di giornale. I motori di ricerca conservano solo gli URL e non l'intero contenuto delle pagine. Infatti, le pagine rimangono memorizzate e gestite dal server e in cui sono state trovate. Di conseguenza una pagina può essere cambiata nel tempo o, addirittura, cancellata senza che il motore di ricerca né l'utente se ne accorgano. Le pagine web sono abbastanza diverse dai documenti di testi convenzionali usati nei sistemi IR tradizionali. Innanzitutto, le pagine web possiedono *collegamenti ipertestuali* e *testi ancorati*, i quali non esistono nei documenti tradizionali. Anche i testi ancorati sono cruciali, perché vi possono esserci pezzi di testi ancorati che spesso riportano descrizioni più accurate per la pagine cui si intende puntare. I collegamenti ipertestuali sono estremamente importanti per il ruolo di ricerca e dell'algoritmo di classificazione.

Sebbene i motori di ricerca non conservino di norma il contenuto dei documenti, mantengono in una *memoria cache* le ultime versioni delle pagine indicizzate le quali sono tuttavia solo una delle versioni di quelle conservate nei server web, e non è detto che siano le versioni più recenti. Contrariamente a quanto si possa pensare, i motori di ricerca permettono all'accesso solo ad una piccola parte dell'in-

tero web, ovvero essi non indicizzano tutte le pagine del web, sia per propri limiti tecnologici, sia per scelte organizzative fatte dall'azienda che lo gestisce, sia per vincoli normativi imposti dal paese che ospita i data center. Dopo tutto lo *spamming* è tra i problemi principali che ci sono sul web, che influisce le valutazioni. Tuttavia non è una questione che concernono gli IR tradizionali. Ma classifiche delle posizioni delle pagine fornite dai motori di ricerca è estremamente importante.

3.4.2 Tipologie di Query

Un'interrogazione utente rappresenta i bisogni informativi dell'utente, che possono essere in forme come seguono:

- **Keyword queries:** L'utente esprime i suoi bisogni informativi mediante una lista di (almeno una) parole chiavi (o termini) con lo scopo di cercare documenti che contengano qualche (almeno uno) o tutti i termini di interrogazione. I termini nella lista sono assunte come essere connesse con una versione "soft" del AND logico. Per esempio se l'utente si fosse interessato nella ricerca di informazioni riguardo a web mining, potrebbe inserire l'interrogazione 'web mining' ad un sistema IR, 'web mining' potrebbe essere ritrattato come 'web AND mining'. Dopodiché l'IRS cerca i documenti rilevanti e li classifica in modo appropriato all'utente. Pertanto, un documento reperito non ha bisogno di contenere tutti i termini nell'interrogazione.
- **Query booleani:** l'utente può usare gli operatori *booleani*, AND, OR e NOT per costruire i query complessi; pertanto questa tipologia di query si consiste in termini e operatori *booleani*.
- **Query frasali:** tali query si consistono in una sequenza di parole che compongono una frase; ogni documento deve contenere almeno una istanza della frase.
- **Query di prossimità:** una versione rilassata di query frasale e può essere una combinazione di termini e frasi.
- **Query dell'intero documento:** quando la query è un documento completo, l'utente vuole cercare altri documenti che siano simili al documento di query.

- **Domanda di linguaggio naturale:** l'utente esprime il bisogno l'informativo come una domanda di linguaggio naturale, e il sistema cerca la risposta.

Capitolo 4

Applicazioni di DM in IR

4.1 Algoritmi di Associazione

Sia $I = \{i_1, i_2, ..., i_m\}$ un insieme di **item**. Sia $T = \{t_1, t_2, ..., t_m\}$ un insieme di **transazioni** (*le basi di dati*), dove ogni transazione t_i è un insieme di items tale che $t_i \subseteq I$. Una **regola di associazione** è quando $X \rightarrow Y$, tale che $X \subset I, Y \subset I$, e $X \cap Y = \varnothing$, e X (o Y) è un insieme di item, o meglio chiamato **item set**.

In una transazione potrebbe esserci un insieme del tipo $\{A, B, C\}$, dove gli elementi di questo insieme possono essere oggetti diversi di qualsiasi tipo, ed una possibile regola di associazione potrebbero essere $A, B \rightarrow C$ dove $\{A, B\}$ è X e C è Y. Il numero di items in un item set è chiamato **taglia** (*size*), e un item set di taglia k è chiamato *k-itemset* . Per esempio, l'itemset $\{A, B, C\}$ è a taglia *3-itemset*. Si dice che una transazione $t_i \in T$ contiene un itemset X se X è un sottoinsieme di t_i. Il **support count** di X in T ($X.count$) è il numero di transazioni in T che contiene X, tale che la robustezza di una regola è misurata dal suo **supporto**. Il supporto di una regola $X \rightarrow Y$ è la percentuale di transazioni in T che contiene $X \cup Y$, e può essere visto come una stima di probabilità, $Pr(X \cup Y)$. La regola di supporto perciò calcola quanto è frequente per essere applicabile nella transazione dell'insieme T.
Sia n il numero di transazioni in T. Il supporto della regola $X \rightarrow Y$ è definita come segue:

$$support = \frac{(X \cup Y).count}{n} \tag{4.1}$$

Il *supporto* è una misura utile quando il suo valore è molto basso, perché la regola potrebbe essere verificata dovuta ad un evento specifico. Se una regola copre poche transazioni, allora potrebbe dare poche informazioni. La *confidenza* di una regola $X \to Y$, è la percentuale di transazioni in T che contiene X ed anche Y, tale che può essere vista come una stima di probabilità condizionata, $Pr(Y|X)$, ed è definita come segue:

$$confidence = \frac{(X \cup Y).count}{X.count}, \tag{4.2}$$

pertanto la confidenza determina la **prevedibilità** delle regole. Se la confidenza di una regola è troppo bassa, allora non è possibile inferire o prevedere in modo efficiente Y da X. Dato un insieme di transazioni T, l'obiettivo è trovare tutte le regole di associazione in T che abbia supporto e confidenza maggiore rispetto al **supporto minimo** (*minsup*) specificato dall'utente e **confidenza minima** (*minconf*). Analogamente, delle volte è possibile notare che un documento di testo o una frase in un documento trattato come una transazione senza considerare la sequenza di parole ed il numero di occorrenze di ciascuna parola. Pertanto dato un set di documenti o un set di frasi, possiamo trovare relazioni co-occorrenze delle parole. Gli algoritmi per trattare una grande quantità di regole di associazioni possono riportare delle differenze in efficienza. Tuttavia, i risultati ottenuti sono basati sulla stessa definizione di regole di associazione. Il miglior algoritmo conosciuto finora è l'**algoritmo a priori**.

4.1.1 Algoritmo Apriori

L'algoritmo *Apriori* si consiste in due passi: **Generare tutti gli itemsets frequenti**, e poi **generare tutte le associazioni di confidenza dagli itemsets frequenti**. Al primo passo si definisce che un itemset frequente è quello che ha il supporto di transazione superiore a minsup. Al secondo passo è ottenere una regola tale che abbia la confidenza maggiore di minconf. Se un itemset ha il supporto minimo, allora ogni sottoinsieme non vuoto di questo itemset possiede un supporto minimo. Inoltre si suppone di avere una transazione che contiene un insieme di items X, allora deve contenere qualche

sottoinsieme non vuoto di X, pertanto la soglia minsup pota un gran numero di itemsets che possono essere non frequenti. Per rassicurare una generazione di itemsets efficienti, si assume che gli items in I siano disposte in *ordine lessicografico*, tale che viene fatto in tutto l'algoritmo. La notazione $\{w[1], w[2], ..., w[k]\}$ rappresenta un k-itemset $\mathbf{w}$, tale che $w[1] < w[2] < ... < w[k]$. Questo algoritmo è basato su **level-wise search**, che genera tutte le itemsets frequenti da facendo passi multipli sui dati.

Algoritmo 13. Al primo passo calcola il supporto di tutti gli items individuali (riga 1), e determina se ciascuno di essi sia frequente (riga 2). F_1 è l'insieme di 1-itemsets frequenti. In ogni sottosequenza k, ci sono tre passi:

1. Esso inizia con il seme dell'insieme di itemsets trovati frequenti F_{k-1} al (k-1)-esimo passo, dopodiché il seme viene usato dalla funzione **candidate-gen()** per generare gli **itemsets candidati** C_k (riga 4), i cui sono i possibili itemsets frequenti.
2. Viene scansionato il database delle transazioni e poi calcolato il supporto corrente di ciascun itemset candidato. Non occorre caricare tutto il dataset prima del processo. In ogni istante, solamente una transazione si risiede nella memoria. Questo modo di procedere rende l'algoritmo scalabile ai datasets di dimensione significativamente grandi, tali che non si possono essere caricati nella memoria.
3. Alla fine del processo, l'algoritmo determina gli itemsets candidati che sono attualmente frequenti.

L'output finale è l'insieme F di tutti gli itemsets frequenti (riga 13).

Algoritmo 14. La funzione candidate-gen si consiste in 2 passi, *join step* e *pruning step*.

- **Join step** (righe 2-6): In questo passo si uniscono due (k-1)-itemsets frequenti per produrre un possibile candidato c (riga 6). I due itemsets f_1 e f_2 possiedono esattamente gli stessi item eccetto l'ultimo (righe 2-5). Poi c viene aggiunto all'insieme di candidati C_k (riga 7).

Algoritmo 13 Apriori(T). Generazione di itemsets frequenti

```
1: C_1 ← init-pass(T)                                   ▷ the first pass over T
2: F_1 ← {f|f ∈ C_1, f.count/n ≥ minsup};               ▷ n is the number of transactions in T
3: for (k = 2; F_{k-1} ≠ ∅; k++) do                     ▷ subsequent passes over T
4:     C_k ← candidate-gen(F_{k-1});
5:     for each transaction t ∈ T do                    ▷ scan the data once
6:         for each candidate c ∈ C_k do
7:             if c is contained in t then
8:                 c.count++;
9:         end for
10:    end for
11:    F_k ← {c ∈ C_k|c.count/n ≥ minsup}
12: end for
13: return F ← ⋃_k F_k;
```

- **Pruning step** (righe 8-11): Il candidato c dal *join step* può essere un candidato finale. Questo passo determina se tutti i k-1 sottoinsiemi (ci sono k di essi) di c sono nel F_{k-1}.

Se alcuni di essi non sono in F_{k-1}, c non può essere frequente secondo la proprietà della **chiusura verso il ribasso** (*downward closure*) e quindi viene rimosso da C_k.

Algorithm 14 Function candidate-gen(F_{k-1})

```
1: C_k ← ∅                                              ▷ initialize the set of candidates
2: for all f_1, f_2 ∈ F_{k-1}                           ▷ find all pairs of frequent itemsets
3:     with f_1 = {i_1, ..., i_{k-2}, i_{k-1}}          ▷ that differ only in the last item
4:     and f_1 = {i_1, ..., i_{k-2}, i_{k'-1}}
5:     and i_{k-1} < i'_{k-1} do                        ▷ according to the lexicographic order
6:     c ← {i_1, ..., i_{k-1}, i'_{k-1}};               ▷ join the two itemsets f_1 and f_2
7:     C_k ← C_k ∪ {c}                                  ▷ add the new itemset c to the candidates
8:     for each (k − 1)-subset s of c do
9:         if (s ∉ F_{k-1}) then
10:            delete c from C_k                        ▷ delete c from the candidates
11:    end for
12: end for
13: return C_k;                                         ▷ return the generated candidates
```

Dal punto di vista teorico, questo è un algoritmo di tipo esponenziale. Sia m il il numero di items in I. Lo spazio di tutti gli itemsets è di ordine $O(2^m)$ perché ognuno di essi può o non può essere in un itemset. Tuttavia, l'algoritmo sfrutta la scarsità dei dati e l'alto valore di supporto minimo per avere un mining possibile ed efficiente. L'algoritmo può scalare fino ai dataset di dimensioni molto grandi per il fatto che non carica tutti i dati nella memoria prin-

cipale. Esso scansiona i dati solamente per K volte, dove K è la taglia del itemset più grande. Si basa su level-wise search, tale che ha la flessibilità di fermare a qualsiasi livello.

4.1.2 Generazione di Regole di Associazione

In molte applicazioni, gli itemsets frequenti sono già funzionali e sufficienti. Pertanto, non occorre generare le regole di associazione. Nelle applicazioni dove le regole sono desiderate, si usano gli itemsets frequenti per generare tutte le regole di associazione. Per generare regole per ogni itemset f frequente, occorre prendere in uso tutti i sotto-insiemi di non-vuoti di f. Ovvero per ciascun dei sottoinsiemi α, si genera in uscita una regola della forma $(f - \alpha) \to \alpha$ se

$$confidence = \frac{f.count}{(f-\alpha).conf} \geq minconf, \tag{4.3}$$

dove $f.count$ (o $(f - \alpha).count$) è il support count di f (o $(f - \alpha)$). Il supporto della regola è $f.count/n$, dove n è il numero di transazioni della transazione dell'insieme T. Per il calcolo della confidenza sono necessari tutti i supporti, perché se f è frequente, allora anche un suo sottoinsieme qualunque può essere frequente. Mentre che tutti i calcoli del supporto vengono registrati nel processo di data mining. Pertanto non occorre la scansione intera dei dati. Questo modo di procedere è esaustivo, ed è un algoritmo poco efficiente. Per designare un algoritmo efficiente, si osserva che il numero dei supporti di f nel calcolo di confidenza sopra, non cambia come quanto cambia α. Segue che se si mantiene una regola $(f - \alpha) \to \alpha$, allora si mantengono tutte le regole della forma $(f - \alpha_{sub}) \to \alpha_{sub}$, dove α_{sub} è un sottoinsieme di α.

Algoritmo 15. È simile alla proprietà *downward closure* che, dove se un itemset è frequente, allora sono frequenti anche tutti i suoi sottoinsiemi. Pertanto da un itemset frequente f, si generano tutte le regole con un item nelle regole successive, dopodiché si usano le conseguenti delle regole e la funzione *candidate-gen*() per generare tutte le possibili conseguenti con due items che compaiono in una regola, e così via. Dall'algoritmo si nota che tutte le conseguenti

dell'1-item (regola con un item nella conseguente) sono generate per primo nella riga 2 delle funzione genRules().

Algorithm 15 genRules(F)

1: **for** each frequent k-itemset f_k in $F, k \geq 2$ **do** ▷ F is the set of all frequent itemsets
2: output every 1-item consequent rule of f_k with confidence $\geq minconf$ and
3: $support \leftarrow f_k.cont/n$ ▷ n is the total number of transactions in T
4: $H_1 \leftarrow$ {consequents of all 1-item consequent rules derived from f_k above};
5: **end for**

Procedure ap-genRules(f_k, H_m)

1: **if** $(k > m + 1)$ AND $(H_m \neq \varnothing)$ **then**
2: $H_{m+1} \leftarrow$ candidate-gen(H_m);
3: **for** eachh_{m+1}inH_{m+1} **do**
4: $conf \leftarrow f_k.count/(f_k - h_{m+1}).count$;
5: **if** $(conf \geq minconf)$ **then**
6: output the rule$(f_k - h_{m+1}) \rightarrow h_{m+1}$ with confidence = $conf$ and support = $f_k.count/n$; ▷ n is the total number of transactions in T
7: **else**
8: delete h_{m+1} from H_{m+1};
9: **end for**
10: ap-genRules(f_k, H_{m+1});

4.1.3 Mining con supporto minimo multipla

Problema di item raro. L'elemento chiave che rende le regole di associazione più pratico è la soglia di *minsup*, tale che è usato per ridurre lo spazio di ricerca e limitare il numero di itemset frequenti e regole generate. Tuttavia, utilizzando un minsup unico si assume implicitamente che tutti gli items nei data siano di stessa natura e/o hanno frequenza simili nella base di dati. In molte applicazioni, alcuni items risultano ad essere molto frequenti nei dati, mentre alcuni altri items compaiono raramente. Se le frequenze degli items hanno una variabilità molto alta, possono comportare delle inconvenienze. Nel caso in cui il minsup è impostato ad un valore troppo elevato, non si riesce a rilevare gli **items rari** nei dati. Inoltre per fare in modo che si rilevino entrambi gli items frequenti e rari, occorre impostare il minsup ad un valore molto basso, però, in tale modo potrebbe comportare una esplosione combinatoria, rendendo quindi il mining impossibile, perché quegli items frequenti possono essere associati in tutti i possibili modi. Poiché alcuni items di natura sono più frequenti, di conseguenza non si può catturare le

differenze inerenti alla natura e/o la frequenza degli items. Allora è importante catturare regole di associazione coinvolgendo meno items frequenti. Tuttavia occorre procedere impedendo di produrre tante regole non significative, con dei supporti molto bassi che causano l'esplosione combinatorio. La soluzione comune potrebbe essere quella di partizionare i dati in diversi piccoli blocchi, ciascuno dei quali contiene solamente gli items di frequenze simili. L'estrazione delle informazioni può essere fatta per ciascun blocco usando un minsup differente. Malgrado questo approccio non può essere soddisfacente perché gli itemsets o regole in blocchi diversi possono non essere rintracciati. Una soluzione migliore è permettere all'utente di specificare il **minimo supporto di item** a ciascun itemsets. Il modello così definito ci permette di trovare itemsets coinvolgendo gli items rari senza generare gli itemsets non significativi. Questo metodo ci permette di risolvere il problema di f_1. Per trattare f_2 si in introduce un vincolo che previene gli itemsets che contengono entrambi gli items frequenti e items rari per essere generati.

Estensione. Per poter permettere all'uso di minimo supporto multiplo, il modello originale ha bisogno di essere esteso. Il supporto deve essere espresso in termini di **minimo supporto di items (MIS)** degli items che compaiono nelle regole. Per Fornire diversi valori MIS agli items differenti, l'utente deve esprimere i requisiti supporto differenti per le regole diverse.

Sia $MIS(i)$ il valore MIS dell'item i. Il **supporto minimo** di una regola R,

$$i_1, i_2, ..., i_k \rightarrow i_{k+1}, ..., i_r, \tag{4.4}$$

che soddisfa il suo supporto minimo se le regole attuali che supportano i dati è maggiore o uguale a:

$$min(MIS(i_1), MIS(i_2), ..., MIS(i_r)). \tag{4.5}$$

Il MIS ci permette così raggiungere l'obiettivo di avere un supporto minimo di regole minore che coinvolge tutte gli items frequenti. Nel modello esteso se si usa l'algoritmo Apriori per trovare tutti gli itemsets frequenti, si potrebbero violare le proprietà di downward closure. Per evitare il problema, l'idea essenziale dell'algoritmo è

ordinare gli items in base al loro valore di MIS in ordine ascendente. Per prevenire che gli items troppo frequenti e rari che compaiano nello stesso itemset, si introduce il vincolo di supporti differenziali. Sia $sup(i)$ il supporto attuale dell'item i nei dati. per ogni itemset s, il vincolo del supporto differenziale è come segue:

$$max_{i \epsilon s}\{sup(i)\} - min_{i \epsilon s}\{sup(i)\} \leq \varphi \tag{4.6}$$

dove $0 \leq \varphi \leq 1$ è il **massimo supporto delle differenze** specificato dall'utente, che può essere uguale per tutti gli itemsets. Il vincolo fondamentalmente si limita alla differenza tra il massimo supporto ed il minimo supporto di items nell'itemset s rispetto a φ. Questo vincolo permette di ridurre il numero di itemsets generati drammaticamente, e non viola la proprietà di downward closure.

4.1.4 Minimo Supporto - Apriori

L'algoritmo *Minimo Supporto - Apriori* si riduce all'algoritmo Apriori quando c'è solamente un valore di MIS (per tutti gli items). Come Apriori, MS-Apriori è basata su level-wise search, tale che genera tutti gli itemsets frequenti facendo passi molteplici sui dati. Tuttavia vi è un'eccezione al secondo passo. L'operazione chiave è l'ordine in ascendenza dei valori MIS. Questo ordine è fissato ed è usato in tutte le operazioni delle sottosequenze dell'algoritmo. Gli items in ciascun itemset seguono quest'ordine.

Sia F_k denota l'insieme di k-items frequenti, ogni itemset w assume la forma $\{w[1], w[2], ..., w[k]\}$ dove $MIS(w[1]) \leq MIS(w[2]) \leq ... \leq MIS(w[k])$.

Algoritmo 16. La prima riga esegue l'ordinamento su I in base al valore MIS di ciascun item (memorizzato in MS). La seconda riga fa passare i dati usando la funzione int-pass(), tale che prende due argomenti, il data set T e gli items ordinati M, produce i semi L per generare gli itemsets candidati di lunghezza 2, es. C_2.

Funzione init-pass(): Per primo si scansionano una volta i dati per registrare i conteggi di supporto di ciascuno di essi. Dopodiché

si cerca il primo item i in M che incontra $MIN(i)$, tale che i è inserito in L. Per ciascun item sottosequenze j in M dopo i, se $j.count/n \geq MIS(i)$ allora anche j viene inserito in L, dove $j.count$ è il conteggio di supporto di j, e n è il numero totale di transazioni in T. Gli 1-itemset(F_1) frequenti sono ottenuti da l (riga 3). Si intuisce che è facile mostrare che tutti gli 1-itemsets frequenti sono in F_1. Per ogni passo successivo si registra il passo k, l'algoritmo esegue i seguenti tre passi:

1. Gli itemsets frequente in F_{k-1} trovato nel $(k-1)$-esimo passo sono usati per generare i candidati C_k usando la funzione $MScandidate\text{-}gen()$ (riga 7). Comunque vi è un caso speciale, come quando $k = 2$ (riga 6), per la quale la funzione di generazione del candidato è differente, (per es. $level2\text{-}candidate\text{-}gen()$).
2. Scansiona i dati e aggiorna i vari punteggi di supporto dei candidati in C_k (riga 9-16). Per ogni candidato c senza il primo item ci occorre aggiornare il suo punteggio di supporto (righe 11-12) e il suo conteggio di supporto di c senza il primo item (righe 13-14), Se la generazione non fosse richiesta, righe 13 e 14 possono essere eliminate.
3. Gli itemsets frequenti F_k per il passo sono identificati nella riga 17.

Funzione Level2-candidate-gen: Si passa l'argomento L, e restituisce al superset del l'insieme di tutti i 2-itemset frequenti. Sulla riga 5, è stato usato $|sup(h) - sup(l)| \leq \varphi$ perché $sup(l)$ non può essere più piccolo di $sup(h)$ anche se MIS(l) ≤MIN(l). Si usa L piuttosto di F_1 perché F_1 non contiene quegli items che possono soddisfare MIS dell'item precedente (in modo ordinato).

Funzione MScandicate-gen: Questa funzione è simile alla funzione $candidate\text{-}gen()$ dell'algoritmo Apriori, e anche esso è composto da due passi: *join step* e *pruning step*. Il **join step** (righe 2-6) lo stesso come quello della funzione $candidate\text{-}gen()$, il **pruning step** (righe 8-12) invece è leggermente differente. Per ogni (k-1)-subset vi è un'eccezione, tale che quando s c[1] (è presente solamente uno come s). Il primo item di c che possiede valore MIS più basso, e non

Algorithm 16 MS-Apriori(T,MS,φ)

▷ MS memorizza tutti i valori di MIS in memoria
1: $M \leftarrow \text{sort}(I, MS)$; ▷ in base al valore di MIS(i) memorizzato in MS
2: $L \leftarrow \text{init-pass}(M, T)$; ▷ fare il primo passo su T
3: $F_1 \leftarrow \{\{l\} | l \in L, l.count/n \geq \text{MIS}(l)\}$; ▷ n è la taglia di T
4: **for** $(k = 2; F_{k-1} \neq \varnothing; k{+}{+})$ **do**
5: **if** $k = 2$ **then**
6: $C_k \leftarrow \text{level2-candidate}(L, \varphi)$ ▷ k=2
7: **else**
8: $C_k \leftarrow MScandidate - gen(F_{k-1}, \varphi)$
9: **for** each transaction $t \in T$ **do**
10: **for** each candidate $c \in C_k$ **do**
11: **if** $c$is contained in t **then** ▷ c è un sottoinsieme di t
12: $c{+}{+}$
13: **if** $c = \{c[1]\}$is contained in$t$ **then** ▷ c senza il primo item
14: $(c - \{c[1]\}).count{+}{+}$
15: **end for**
16: **end for**
17: $F_k \leftarrow \{c \in C_k | c.count/n \geq MIS(c[1])\}$
18: **end for**
19: **return** $F \leftarrow \bigcup_k F_k$

Function 17 level2-candidate-gen(L, φ)

1: $C_2 \leftarrow \varnothing$; ▷ inizializza l'insieme dei candidati
2: **for** each item l in L nello stesso ordine **do**
3: **if** $l.count/n \leq \text{MIS}(l)$ **then**
4: **for** each item h in L that is after l **do**
5: **if** $h.count/n \geq \text{MIS}(l)$ and $|sup(h) - sup(l)| \leq \varphi$ **then**
6: $C_2 \leftarrow C_2 \cup \{l, h\}$; ▷ inserire il candidato $\{l, h\}$ in C_2

è in s. Anche se s non è in F_{k-1}, non è possibile eliminare c perché non si è sicuri che s non soddisfi MIS(c[1]). Benché sapessimo non soddisfa MIS(c[2]), a meno che MIS(c[2])=MIS(c[1]) (riga 9).

Function 18 MScandidate-gen(F_{k-1}, φ)

```
1:  C_k ← ∅                                          ▷ inizializza l'insieme di candidati
2:  for all f_1, f_2 ∈ F_k                           ▷ trovare tutte le coppie di itemsets frequenti
3:      with f_1 = {i_1, ..., i_{k-2}, i_{k-1}}      ▷ si differenzia solamente nell'ultimo item
4:      and f_2 = {i_1, ..., i_{k-2}, i'_{k-1}}
5:      and i_{k-1} < i'_{k-1} and |sup(i_{k-1}) − sup(i'_{k-1})| ≤ φ do
6:      c ← {i_1, ..., i_{k-1}, i'_{k-1}};           ▷ Unire i due itemsets f_1 e f_2
7:      C_k ← C_k ∪ {c};                             ▷ inserire l'itemset candidato c in C_k
8:      for each (k-1)-subset s of c do
9:          if (c[1] ∈ s) or (MINS(c[2]) = MIS(c[1])) then
10:             if (s ∉ F_{k-1}) then
11:                 delete c from C_k                ▷ eliminare c dall'insieme di candidati
12:     end for
13: end for
14: return C_k                                       ▷ restituisce i candidati generati
```

4.1.5 Pattern Sequenziali

L'estrazione delle informazioni in *regole di associazione* non considera l'ordine delle transazioni. Tuttavia tali ordinamenti sono importanti. Si definisce che una lista ordinata di itemsets è una **sequenza**.

Sia $I = \{i_1, i_2, ..., i_m\}$ un insieme di items, sia un itemset $X \neq \varnothing$, tale che $X \subseteq I$, e sia $s = \langle a_1, a_2, ..., a_r \rangle$ una sequenza dove a_i è un itemset oppure elementi di s. Sia l'insieme $\{x_1, x_2, ..., x_k\}$ un elemento (o un itemset) di una sequenza, dove $x_j \in I$ è un item. Si assume senza perdere la generalità che items in un elemento di una sequenza è disposta in ordine lessicografica. Un item può presentarsi solamente una volta in un elemento della sequenza, ma può presentarsi diverse volte in elementi differenti. La taglia di una sequenza è il numero di elementi (*itemsets*) nelle sequenze. La lunghezza di una sequenza è il numero di items in nella sequenza. Una sequenza di lunghezza k è chiamata *k-sequenza*. Se un item nella sequenza si presenta diverse volte in elementi diversi della sequenza, allora ogni occorrenza contribuisce al valore di k. Una sequenza $s_1 = \langle a_1, a_2, ..., a_r \rangle$ è una **sottosequenza** di un'altra sequenza $s_2 = \langle b_1, b_2, ..., b_v \rangle$, oppure si dice che s_2 è una **supersequenza** di s_1, se esistessero i numeri interi

$1 \leq j_1 < j_2 < ... < j_r \leq v$ tale che $a_1 \subseteq b_{j1}, a_2 \subseteq b_{j2}, ..., a_r \subseteq b_{jr}$. È anche possibile dire che s_2 **contiene** s_1.

L'obiettivo è pertanto, dato un insieme S della sequenza di dati in input, si cercano tutte le sequenze che abbiano un supporto minimo specificato dall'utente. Ogni sequenza è chiamata **sequenza frequente** o **pattern sequenziali**. Il **supporto** per una sequenza è la frazione di sequenza di dati totali in S che contiene questa sequenza.

4.2 Alberi di Classificazione

Gli alberi di classificazione possiedono tre tipi di nodi:

- **Nodi di decisione** (*domanda*), è associato ad una variabile decisionale. Gli archi uscenti da un nodo decisione rappresentano i differenti valori che la variabile può assumere.
- **Nodi evento** (*opzioni*), è associato ad un evento casuale. Gli archi uscenti rappresentano i possibili stati di natura per l'evento di riferimento.
- **Nodi terminale** (nodi foglia), ad ogni nodo terminale è il guadagno determinato dallo scenario ad esso associato, ovvero dalla sequenza di decisioni ed eventi verificatisi lungo il cammino che va dal nodo radice al nodo terminale.

Il primo nodo è anche detto *radice*: in base all'esito della proposizione logica del nodo si scende ai nodi sottostanti, dalla parte sinistra se l'esito è positivo o dalla parte destra se l'esito è negativo; i nodi foglie e contengono la funzione approssimante della classe di appartenenza. Un nodo di decisione specifica qualche verifica (per es. fare domande) su un attributo singolo e, un nodo foglia che indica la classe.

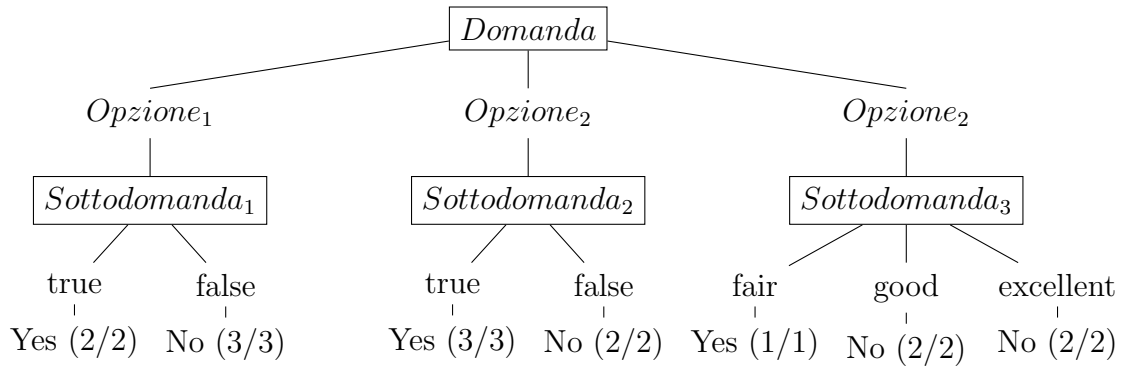

Fig. 4.2 Un esempio di albero di decisione

Alla radice dell'albero decisionale c'è il nodo di decisione, e su cui si pone una domanda di tipo a scelta multipla (con $n \geq 2$ possibili risposte, e in base al contesto si decide il numero di risposte istanziare). Dopodiché alla radice vengono associati i nodi foglia, per ciascun nodo foglia si attribuisce la specifica risposta della domanda (a scelte multiple) iniziale. Successivamente alla luce di ciascuna risposta (nodo foglia) si pone un'altra domanda, e via così, fino ad arrivare ai nodi foglia finché non si pongono più ulteriori domande. Per valutare l'albero decisionale, si considera l'albero dall'alto verso il basso, e secondo il valore attributo fornito del test set fino al nodo foglia. Dove la classe foglia è la classe predetta del test set. L'albero decisionale è costruita partendo da partizionare il training set in modo che i sottoinsiemi possano essere puri possibili. Il sottoinsieme puro contiene solamente la singola classe di training set. Per varie ragioni nelle applicazioni pratiche, si vuole ottenere un albero piccolo ed accurato. Un albero piccolo è più generico, e tende ad essere più accurato. In generale non è soddisfacente avere una semplice classificazione dando per esclusione la decisione fatta secondo il contesto.

Nell'esempio della Fig.4.2 i dati che raggiungono ciascun nodo foglia, appartengono tutti quanti alla stessa vedendo i valori (x/y) ad ogni nodo foglia). In genere nei data sets reali non è così. I valori (x/y) tale che $x \leq y$ sono i valori di confidenza usate nelle regole di associazione dove x è il conteggio supporto. Pertanto si evince che un albero di decisione può essere convertito in regole di tipo if-$then$. Un albero di decisione cerca solamente il sottoinsieme delle regole che esistono nei dati, che è sufficiente per la classificazione. I risultati dell'insieme delle regole sono mutuamente esclusive ed esaustive, ciò implica che ogni istanza di dati è coperta da una regola singola.

Funzione di Impurità

Le funzioni di impurità più note per l'apprendimento dell'albero di decisione sono **information gain** e **information gain ratio**.

L'information gain si basa sulla funzione di **entropia**:

$$entropy(D) = -\sum_{j=1}^{|C|} Pr(c_j) \log_2 Pr(c_j), \text{ t.c. } \sum_{j=1}^{|C|} Pr(c_j) = 1, \qquad (4.7)$$

dove $Pr(c_j)$ è la probabilità di classe c_j nel data set D, tale che è il numero di esempi della classe c_j in D diviso per il numero totale di esempi in D. Da ciò si osserva che al crescere della purezza, l'entropia tende ad annullarsi. L'entropia è massima quando entrambi i gruppi hanno la probabilità empirica pari a 0.5.

4.2.1 Algoritmo di Apprendimento ad Alberi

L'albero di apprendimento è tipicamente fatto usando la strategia **divide et impera** che partiziona ricorsivamente i dati. Alla fase iniziale tutti i dati sono sulla radice. Man mano che cresce l'albero i dati vengono suddivisi ricorsivamente.

Algoritmo 19. Secondo il **criterio dell'arresto** della ricorsione (righe 1-4), l'algoritmo si ferma quando tutti i dati training nei dati correnti sono della stessa classe, o quando ogni attributo è stato usato lungo il cammino dell'albero corrente. Nell'apprendimento dell'albero, ogni ricorsione successiva sceglie l'**attributo migliore** per partizionare i dati al nodo corrente in base al valore dell'attributo. L'attributo migliore è selezionato basando su una funzione che mira a minimizzare l'impurità dopo il partizionamento. In altre parole esso massimizza l'impurità. La chiave in apprendimento dell'albero decisionale è comunque la scelta della **funzione dell'impurità** (righe 7,9 e 11). La chiamata ricorsiva dell'algoritmo prende i dati del training al nodo per ulteriori partizioni per estendere l'albero. Questo è un algoritmo avido che non permette di far ritornare sui propri passi. Una volta creato il nodo non viene revisionato.

Information Gain

Sia calcolato il valore di impurtà del data set D (la funzione **impurityEval-1**, riga 7). Per sapere quale attributo riduce l'impurità, viene valutato ogni attributo (righe 8-10). Sia v il possibile valore dell'attributo A_i, dopodiché si usa A_i per partizionare i dati D, dividendo D in

Algorithm 19 decisionTree(D, A, T)

if D contains only training examples of the same class $c_j \in C$ **then**
  make T a leaf node labeled with class c_j;
else if $A = \varnothing$ **then**
  make T a leaf node labeled with c_j, which is the most frequent class in D
else ▷ D contains examples belonging to a mixture of classes. We select a single
▷ attribute to partition D into subsets so that each subset is purer
  P_0=impurityEval-1(D)
  for each attribute $A_i \in A(= \{A_1, A_2, ..., A_k\})$ **do**
    P_i=impurityEval-2(A_i, D)
  end for
  Select $A_g \in \{A_1, A_2, ..., A_k\}$ that gives the biggest impurity reduction, computed using $p_0 - p_i$;
  if $p_0 - p_g < threshold$ **then** ▷ A_g does not significantly reduce impurity p_0
    make T a leaf node labeled with c_j, the most frequent class in D.
  else ▷ A_g is able to reduce impurity p_0
    Make T a decision node on A_g;
    Let the possible values of A_g be $v, v_2, ..., v_m$. Partitions D into m disjoint subsets $D_1, D_2, ..., D_m$ based on the m values of A_g.
    for each D_j in $D_1, D_2, ..., D_m$ **do**
      if $D_j \neq \varnothing$ **then**
        create a branch (edge) node T_j for v_j as a child node of T;
        decisionTree($D_j, A - A_g, T_j$) ▷ A_g is removed
      end if
    end for
  end if
end if

v sottoinsiemi $D_1, D_2, ..., D_v$. L'entropia dopo la partizione diventa (funzione di **impurityEval-2**, riga 9)

$$entropy_{A_i}(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times entropy(D_j). \tag{4.8}$$

L'information gain dell'attributo A_i viene così calcolato,

$$gain(D, A) = entropy(D) - entropy_{A_i}(D). \tag{4.9}$$

La misura di *gain* (riga 11) sceglie l'attributo A_g che risulta essere la più grande riduzione in impurità. Se il gain di A_g è troppo piccolo, l'algoritmo si ferma (riga 12), dove normalmente viene usata una soglia. Se scegliendo A_g, è possibile ridurre significativamente l'impurità, A_g viene impiegato per partizionare i dati per estendere l'albero ulteriormente (righe 15-21). Il processo va avanti ricorsivamente costruendo sottoalberi utilizzando $D_1, D_2, ..., D_m$ (riga 20). Per l'estensione della sottosequenza dell'albero, non è più necessario A_g, come in tutto il training set di ciascun ramo possiede lo stesso valore A_g.

Rapporto di Information Gain

Il criterio di information gain tende a favorire gli attributi con più valori possibili. Una situazione estrema è che i dati contengono un attributo ID per partizionare i dati, ogni training set formulerà un sottoinsieme e possiede solamente una classe, che risulta in $entropy_{ID}(D) = 0$. Così le informazioni che si ottengono dall'uso di questo attributo può essere massimo. Dal punto di vista di predizione, tale partizione è inutile. Il rapporto di accumulo (4.10) rimedia questo scostamento normalizzando l'accumulo usando l'entropia dei dati rispetto al valore dell'attributo:

$$gainRatio(D, A_i) = \frac{gain(D, A)}{-\sum_{j=1}^{s} (\frac{|D_i|}{|D|} \times \log_2 \frac{|D_j|}{|D|})} \tag{4.10}$$

dove s è il numero di valori possibili di A_i e D_j è il sottoinsieme dei dati che possiede il j-esimo valore di A_i. $|D_j|/|D|$ corrisponde alla probabilità dell'espressione (4.7). Usando l'espressione (4.10), si sceglie l'attributo con il valore del rapporto di accumulo più alto per estendere l'albero. Questo metodo funziona perché se A_i possiede troppi valori, il denominatore potrà essere grande.

4.3 Classificazione di Naïve Bayes

Sia $A_1, A_2, ..., A_{|A|}$ l'insieme di attributi con valori discreti nel dataset D. Sia C la classe attributo con $|C|$ valori, $c_1, c_2, ..., c_{|C|}$. Dato un test set d insieme al valore attributo osservato da a_1 a $a_{|C|}$, dove a_i è un possibile valore di A_i, e

$$d = \langle A_1 = a_1, ..., A_{|A|} = a_{|A|} \rangle . \tag{4.11}$$

La classe c_j è la previsione tale che la probabilità $Pr(C = c_j | A_1 = a_1, ..., A_{|A|} = a_{|A|})$ sia massima, o meglio definito ipotesi di **massimo a posteriori**. Dove c_j è sotto l'ipotesi di massimo a posteriori. Dalla legge di Bayes, l'espressione (2.2) può essere espressa come segue

$$\begin{aligned} Pr(C = c_j | A_1 = a_1, ..., A_{|A|} = a_{|A|}) &= \frac{Pr(A_1 = a_1, ..., A_{|A|} = a_{|A|} | C = c_j) Pr(C = c_j)}{Pr(A_1 = a_1, ..., A_{|A|} = a_{|A|})} \\ &= \frac{Pr(A_1 = a_1, ..., A_{|A|} = a_{|A|} | C = c_j) Pr(C = c_j)}{\sum_{k=1}^{|C|} Pr(A_1 = a_1, ..., A_{|A|} = a_{|A|}) Pr(C = c_k)} \end{aligned} \tag{4.12}$$

$Pr(C = c_j)$ è la probabilità della classe a priori di c_j, tale che può essere stimato dal training set, il quale è semplicemente la frazione di dati in D della classe c_j. Se siamo interessati solamente nella classificazione, $Pr(A_1 = a_1, ..., A_{|A|} = a_{|A|})$ è irrilevante per fare decisioni perché è lo stesso per ogni classe. Pertanto solamente $Pr(A_1 = a_1, \wedge ... \wedge, A_{|A|} = a_{|A|} | C = c_j) Pr(C = c_j)$ ha bisogno di essere calcolato, quindi può essere riscritto come

$$\begin{aligned} &Pr(C = c_j | A_1 = a_1, ..., A_{|A|} = a_{|A|} | C = c_j) \\ &= Pr(A_1 = a_1 | A_2 = a_2, ..., A_{|A|} = a_{|A|}, C = c_j) \times Pr(A_2 = a_2, ..., A_{|A|} = a_{|A|} | C = c_j) \end{aligned} \tag{4.13}$$

Ricorsivamente, il secondo termine sopra $Pr(A_2 = a_2, ..., A_{|A|} = a_{|A|} | C = c_j)$, può essere riscritto basando sullo stesso modo di procedere.

Assunzione di Indipendenza Condizionata

Si assume che data la classe $C = c_j$ tutti gli attributi siano condizionatamente indipendenti.

$$Pr(A_1 = a_1 | A_2 = a_2, ..., A_{|A|} = a_{|A|}, C = c_j) = Pr(A_1 = a_1 | C = c_j) \tag{4.14}$$

e analogamente per gli attributi da A_2 fino a $A_{|A|}$. Otterremo pertanto

$$Pr(A_1 = a_1, ..., A_{|A|} = a_{|A|} | C = c_j) = \prod_{i=1}^{|A|} Pr(A_i = a_i | C = c_j) \tag{4.15}$$

e

$$Pr(C = c_j|A_1 = a_1, ..., A_{|A|} = a_{|A|}) = \frac{Pr(C = c_j)\prod_{i=1}^{|A|} Pr(A_i = a_i|C = c_j)}{\sum_{k=1}^{|C|} Pr(C = c_k)\prod_{i=1}^{|A|} Pr(A_i = a_i|C = c_j)} \quad (4.16)$$

Successivamente occorre stimare le probabilità a priori $Pr(C = c_j)$ e le probabilità condizionate $Pr(A_i = a_i|C = c_j)$ dal training set, che sia chiaro:

$$Pr(C = c_j) = \frac{\text{numero di esempi della classe } c_j}{\text{numero totale degli esempi nel dataset}} \quad (4.17)$$

$$Pr(A_i = a_i|C = c_j) = \frac{\text{numero di esempi con } A_i = a_i \text{ e classe } c_j}{\text{numero di esempi della classe } c_j} \quad (4.18)$$

Se abbiamo bisogno di una decisione sulla classe più probabile per ogni test set, avremo bisogno solamente del numeratore dell'espressione (4.16) dal fatto che il denominatore risulta essere analogo per ogni classe. Pertanto dato un test set, calcoliamo la classe più probabile per il test set:

$$c = \arg\min_{c_j} Pr(c = c_j)\prod_{i=1}^{|A|} Pr(A_i = a_i|C = c_j) \quad (4.19)$$

È facile vedere che le probabilità (come $Pr(C = c_j)$ e $Pr(A_i = a_i|C = c_j)$) richieste per costruire un classificatore di naïve Bayes può essere trovata in una scansione dei dati. Perciò l'algoritmo è lineare in numero di esempi di training, che è uno dei punti forti di naïve Bayes. Per apprendere i classificatori pratici di Naïve Bayes, occorre indirizzare ad alcuni problemi addizionali: su come maneggiare gli attributi numerici, conteggi nulli, e valori mancanti.

4.4 Support Vector Machines

Sia un insieme di training set $D = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), ..., (\mathbf{x_n}, y_n)\}$, dove $x_i = (x_{i1}, x_{i2}, ..., x_{ir})$ è un input di vettori r-dimensional nello spazio reale $X \subseteq \Re^r$, y_i è l'etichetta classe (valore output) e $y_i \in \{1, -1\}$, 1 denota la classe positiva e -1 denota la classe negativa.
Per sviluppare un classificatore, *support vector machines* (SVM) cerca una funzione lineare della forma seguente

$$f(x) = \langle \mathbf{w} \cdot \mathbf{x} \rangle + b \quad (4.20)$$

e

$$y_i = \begin{cases} 1 & se \quad \langle \mathbf{w} \cdot \mathbf{x_i} \rangle + b \geq 0 \\ -1 & se \quad \langle \mathbf{w} \cdot \mathbf{x_i} \rangle + b < 0 \end{cases} \tag{4.21}$$

dove $f(x)$ è una funzione reale $f : X \subseteq \Re^r \rightarrow \Re$, $\mathbf{w} = (w_1, w_2, ..., w_r) \in \Re^r$ è un **vettore peso**, e $b \in \Re$ è lo **scostamento**. Utilizzando la notazione matriciale, l'espressione (4.20) può essere scritta come una combinazione lineare

$$f(x_1, x_2, ..., x_r) = w_1 x_1 + w_2 x_2 + ... + w_r x_r + b,$$

dove x_i è la variabile che rappresenta l'i-esima coordinata del vettore $\mathbf{x}$. In sostanza, con SVM si cerca un iperpiano chiamato **confine di decisione** (o **superficie di decisione**)

$$\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0 \tag{4.22}$$

il quale separa i dati positivi del training set da quelli negativi. Geometricamente, l'iperpiano $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$ divide lo spazio dei dati in input in due spazi: metà per gli esempi positivi e l'altra metà per gli esempi negativi. In genere può esserci un numero infinito di linee (o iperpiano) che separa i valori positivi da quelli negativi. In senso concreto l'SVM sceglie l'iperpiano che massimizza il margine (*gap*) tra il data points positivi e quelli negativi. L'espressione (4.20) è la **regola di decisione** del classificatore SVM, usato per decisioni di classificazione sulle istanze di test. Quando invece i dati non hanno una distribuzione lineare, SVM in addizione si ricorre alle funzioni di **Kernel**.

4.4.1 SVM lineare

Dalla teoria di algebra lineare, nella forma $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$, $\mathbf{w}$ è il **vettore normale** che definisce una direzione perpendicolare verso l'iperpiano. Fissando i valori di $\mathbf{w}$, variando b si potrebbe muovere l'iperpiano parallelo verso se stesso. È da considerare che anche $\langle \mathbf{w} \cdot \mathbf{x} \rangle + \lambda b = 0$ i gradi di libertà inerenti. È possibile riscalare l'iperpiano rispetto a $\langle \lambda \mathbf{w} \cdot \mathbf{x} \rangle + \lambda b = 0$ per $\lambda \in \Re^+$, senza cambiare la funzione dell'iperpiano. L'SVM ha il compito di massimizzare il margine tra i data points positivi da quelli negativi. Sia d_+ (rispettivamente d_-) la distanza minima dall'iperpiano di separazione

($\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$) al data point positivo (negativo). Il margine dell'iperpiano di separazione è $d_+ + d_-$. In SVM si cerca il margine più ampio dell'iperpiano di separazione, ovvero l'**iperpiano marginale massimale**, come **vincolo di decisione**.

Considerando un data point positivo $(\mathbf{x}^+, \mathbf{1})$ e uno negativo $(\mathbf{x}^-, \mathbf{1})$ che sono i punti più vicini rispetto all'iperpiano $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$. Si definiscono H_+ e H_- che passano attraverso $\mathbf{x}^+$ e $\mathbf{x}^-$ rispettivamente. H_+ e H_- sono anche paralleli rispetto a $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$. È possibile riscalare $\mathbf{w}$ per ottenere:

$$H_+ : \langle \mathbf{w} \cdot \mathbf{x}^+ \rangle + b = 1 \qquad H_- : \langle \mathbf{w} \cdot \mathbf{x}^- \rangle + b = -1 \tag{4.23}$$

tali che

$$\begin{cases} \langle \mathbf{w} \cdot \mathbf{x_i} \rangle + b \geq 1 & \text{se } y_i = 1 \\ \langle \mathbf{w} \cdot \mathbf{x_i} \rangle + b \leq -1 & \text{se } y_i = -1 \end{cases} \tag{4.24}$$

i quali indicano che nessun training set cade tra gli iperpiani H_+ e H_-. La distanza tra H_+ e H_- è il margine $(d_+ + d_-)$. Ricordando che la distanza perpendicolare Euclidea dal punto $\mathbf{x_i}$ all'iperpiano $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$ è:

$$\frac{|\langle \mathbf{w} \cdot \mathbf{x} \rangle + b|}{\|\mathbf{w}\|}, \tag{4.25}$$

dove $\|\mathbf{w}\|$ è la norma Euclidea di $\mathbf{w}$,

$$\|\mathbf{w}\| = \sqrt{< \mathbf{w} \cdot \mathbf{w} >} = \sqrt{w_1^2 + w_2^2 + ... + w_r^2} \tag{4.26}$$

Per calcolare d_+, invece di misurarla da $\mathbf{x}^+$ all'iperpiano $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$, si prende qualsiasi punto $\mathbf{x_s}$ prese dalle osservazioni su $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$, e calcolare la distanza da $\mathbf{x_s}$ a $\langle \mathbf{w} \cdot \mathbf{x}^+ \rangle + b = 1$ applicando l'espressione (4.25) e considerando l'espressione $\langle \mathbf{w} \cdot \mathbf{x_s} \rangle + b = 0$,

$$d_+ = \frac{|\langle \mathbf{w} \cdot \mathbf{x_s} \rangle + b - 1|}{\|\mathbf{w}\|} = \frac{1}{\|w\|} \tag{4.27}$$

Analogamente si può calcolare la distanza da $\mathbf{x_s}$ a $\langle \mathbf{w} \cdot \mathbf{x}^+ \rangle + b = -1$ per ottenere $d_- = \frac{1}{\|w\|}$. Pertanto il vincolo di decisione cadrà nella via di mezzo tra H_+ e H_-. Il margine è pertanto

$$margin = d_+ + d_- = \frac{2}{\|w\|} \tag{4.28}$$

Dal fatto che SVM cerca di separare l'iperpiano che massimizza il margine, questo ci riconduce al problema di ottimizzazione. Massimizzare il margine come minimizzare $\|w\|^2/2 = \langle \mathbf{w} \cdot \mathbf{w} \rangle / 2$.

Dato un insieme di training set $D = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), ..., (\mathbf{x_n}, y_n)\}$, l'apprendimento si pone nella seguente risoluzione del problema di minimo vincolato,

$$\begin{aligned} &\text{minimo: } \frac{\langle \mathbf{w} \cdot \mathbf{w} \rangle}{2} + C \sum_{i=1}^{n} \xi_i \\ &\text{soggetto a: } y_i(\langle \mathbf{w} \cdot \mathbf{x_i} \rangle + b) \geq 1 - \xi_i \end{aligned} \tag{4.29}$$

per $i = 1, 2, ..., n$ e $\xi_i \geq 0$, dove ξ_i è la variabile di rilassamento, C è un parametro costante e $y_i \in \{-1, 1\}$. Poiché la funzione obiettivo è convessa e quadratica, i vincoli sono lineari in parametri **w** e b, è possibile applicare il metodo di moltiplicatore di Lagrange. Pertanto il problema primale standard è definito come segue:

$$L_p = \frac{1}{2} \langle \mathbf{w} \cdot \mathbf{w} \rangle + C \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i [y_i(\langle \mathbf{w} \cdot \mathbf{x_i} \rangle) - 1 + \xi_i] - \sum_{i=1}^{n} \mu_i \xi_i \tag{4.30}$$

dove per $\{\alpha_i, \mu_i\} \geq 0$, α_i e μ_i sono i moltiplicatori di **Lagrange**. Per minimizzare i parametri, si considerano le seguenti espressioni:

$$\frac{\partial L_p}{\partial w_j} = w_j - \sum_{i=1}^{n} y_i \alpha_i x_{ij} = 0, \text{ con } j = 1, 2, ..., r \tag{4.31}$$

$$\frac{\partial L_p}{\partial b} = -\sum_{i=1}^{n} y_i \alpha_i = 0, \text{ con } i = 1, 2, ..., n \tag{4.32}$$

$$\frac{\partial L_p}{\partial \xi_i} = C - \alpha_i - \mu_i = 0, \text{ con } i = 1, 2, ..., n \tag{4.33}$$

$$y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 + \xi_i \geq 0, \text{ con } i = 1, 2, ..., n \tag{4.34}$$

$$\sum_{i=1}^{n} \alpha_i [y_i(\langle \mathbf{w} \cdot \mathbf{x_i} \rangle) - 1 + \xi_i] = 0, \text{ con } i = 1, 2, ..., n \text{ e } \alpha_i \geq 0 \tag{4.35}$$

Vi è un moltiplicatore di Lagrange α_i per ogni punto del training set. L'espressione 4.35 mostra che solamente i data points sul margine degli iperpiani possiedono $\alpha_i \geq 0$. Questi data points sono **vettori di supporto**, mentre tutti gli altri data points possiedono $\alpha_i = 0$. Dall'espressione 4.33 si deduce che $\alpha_i \leq C$ dal fatto che $\mu_i \geq 0$. Facendo opportune sostituzioni si ottiene:

$$w_j = \sum_{i=1}^{n} y_i \alpha_i x_{ij}, \quad j = 1, 2, ..., r \quad \text{tale che} \quad \sum_{i=1}^{n} y_i \alpha_i = 0 \tag{4.36}$$

pertanto la funzione di ottimizzazione originale si trasforma nella forma seguente:

$$\max L_D = max \left\{ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle \right\}, \text{ soggetto a: } \sum_{i=1}^{n} y_i \alpha_i = 0 \quad (4.37)$$

dopo avere ottenuto i valori α_i tali che vengono poi usati per calcolare i vettori ponderati $\mathbf{w}$ e lo scostamento b. Da cui si accorge che i parametri ξ_i e i suoi moltiplicatori μ_i non sono più inclusi nella funzione di ottimizzazione. Dalle equazioni (4.34) e (4.35), e dal fatto che $\mu_i \xi_i = 0$ si deduce che è possibile utilizzare qualsiasi punto dei dati di training set. E con $\xi_i = 0$ è possibile calcolare il valore b:

$$b = \frac{1}{y_i} - \sum_{i=1}^{n} y_i \alpha_i \langle \mathbf{x_i} \cdot \mathbf{x_j} \rangle . \quad (4.38)$$

Dovuto ad errori numerici, è possibile calcolare tutti i possibili valori b. Inoltre:

$$\alpha_i = 0 \Rightarrow y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 \geq 0 \text{ e } \xi_i = 0 \quad (4.39a)$$

$$0 < \alpha_i < C \Rightarrow y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 = 0 \text{ e } \xi_i = 0 \quad (4.39b)$$

$$\alpha_i = 0 \Rightarrow y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 \leq 0 \text{ e } \xi_i \geq 0 \quad (4.39c)$$

Caso separabile: Per $C = 0$, si ha che i data points sono **separabili**. La disuguaglianza (4.34) è l'insieme dei vincoli originali. Si nota che nonostante vi sia un moltiplicatore α_i per ogni data point del training, la condizione complementare (4.35) mostra che solamente i data points sull'iperpiano (H_+ e H_-) possono avere $\alpha_i > 0$ quando $y_i(< \mathbf{w} \cdot x_i > +b) - 1 = 0$. Questi data points vengono chiamati **vettori di supporto**.

Caso non separabile: Le espressioni (4.39) mostrano le più importanti proprietà di SVM: avendo C diverso da zero, la soluzione è scarsa nei α_i. Si ha che la maggior parte dei data points al di fuori dell'area di margine ed il loro α_i nella soluzione sono 0. Soltanto quei data points che sono sul margine ($y_i(< \mathbf{w} \cdot \mathbf{x_i} > +b = 1$, tali che sono i vettori di supporto), o all'interno del margine ($\alpha_i = C$ e $y_i(< \mathbf{w} \cdot \mathbf{x_i} > +b < 1)$), o errori sono non-zero. Senza queste proprietà SVM potrebbe essere non praticabile per i datasets significativamente grandi.

Il vincolo di decisione finale (**Iperpiano di margine massimale**) è quindi

$$\langle \mathbf{w} \cdot \mathbf{w} \rangle + b = \sum_{i \epsilon sv} y_i \alpha_i \langle \mathbf{x} \cdot \mathbf{x} \rangle + b = 0 \tag{4.40}$$

dove sv è l'insieme di indici dei vettori di supporto del training set.

Testing: La regola di decisione per la classificazione da applicare è l'espressione (4.40). Dati gli esempi di test $\mathbf{z}$, il classificatore è pertanto definito come segue:

$$sign(\langle \mathbf{w} \cdot \mathbf{z} \rangle + b) = sign\left(\sum_{i \epsilon sv} y_i \alpha_i \langle \mathbf{x} \cdot \mathbf{z} \rangle + b \right). \tag{4.41}$$

Se (4.41) ritorna 1, allora l'esempio test $\mathbf{z}$ lo si classifica come positivo, altrimenti negativo.

4.4.2 SVM Non lineare

Le formulazioni SVM richiedono gli esempi positivi e negativi per essere linearmente separati. Il vincolo di decisione deve essere un iperpiano. Tuttavia per molti datasets reali, i vincoli di decisione non sono lineari. Per queste forme di dati, la stessa formulazione e tecniche di soluzione per il caso lineare sono ancora usate. Occorre solamente la trasformazione dei dati in input dal suo spazio originale allo spazio di trasformazione, tale che è chiamato **spazio caratteristica**. Lo spazio originale si chiama come **spazio input**. L'idea di base è mappare i dati dello spazio input X allo spazio caratteristica F via una mappatura non lineare ϕ,

$$\phi : X \to F \tag{4.42}$$

$$\mathbf{x} \mapsto \phi(\mathbf{x}). \tag{4.43}$$

Dopo la mappatura, il dataset originale $\{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), ..., (\mathbf{x_n}, y_n)\}$ diventa $\{(\phi(\mathbf{x_1}), \mathbf{y_1}), (\phi(\mathbf{x_2}), \mathbf{y_2}), ..., (\phi(\mathbf{x_n}), \mathbf{y_n})\}$. Dopodiché lo stesso metodo di soluzione SVM viene applicato ad F.

4.5 K-Nearest Neighbor Learning

Sia l'insieme D i dati del training set. Nessuna azione sarà effettuata sugli esempi di training. Quando un istanza test d è presentato, l'algoritmo d confronta d con ogni esempio di training in D per calcolare la similarità o la distanza tra essi. Vengono selezionati poi i k esempi più simili (vicini) i esempi in D. Questo insieme di esempi si chiama k **nearest neighbors** di d. d allora prende la classe più frequente in mezzo al k nearest neighbors. È da notare che $k = 1$ usualmente non è determinare la classe di d dovuto al rumore e valori anomali nei dati. Un insieme di nearest neighbors necessario ha bisogno di scegliere accuratamente la classe. L'algoritmo kNN in generale è dato in algoritmo 20

Algoritmo 20 $k\text{NN}(D, d, k)$

1: Calcolare la distanza tra d ed ogni esempio in D;
2: Scegliere i k esempi in D che sono più vicini a d, denotare l'insieme da $P(\subseteq D)$;
3: Assegnare d alla classe più frequente in P(o la classe di maggioranza).

La componente chiave di un algoritmo kNN è la *funzione distanza* (o *similarità*), tale che è scelta basando su applicazioni e la natura dei dati. Per i dati relazionali, la distanza Euclidea è comunemente usata. Per i documenti di testo, coseno di similarità è una scelta popolare. Il numero di nearest neighbors k è usualmente determinata facendo uso di un insieme di validazione, o attraverso la convalida incrociata sui dati di stima. Questo è un intervallo di k valori che vengono provati, ed i k valori che rendono la miglior accuratezza sul set di validazione (o convalida incrociata) viene selezionata. Nonostante la sua semplicità, si mostra che la classificazione accurata di kNN può essere abbastanza considerevole e in molti casi è tanto accurato quanto quei metodi accurati. kNN tuttavia rallenta il tempo di classificazione. Dovuto al fatto che non vi è la costruzione del modello. Non è pertanto applicabile se un modello comprensibile è richiesta nell'applicazione.

4.6 Algoritmi di Segmentazione

4.6.1 Metodo di k-medie

Sia D un insieme di data points $\{X_1, X_2, ..., X_n\}$, dove ogni X_i = $(X_{i1}, X_{i2},\ ...,\ X_{ir})$ è un vettore di nello spazio reale $X \subseteq R^r$, r è il numero di attributi nei dati. L'algoritmo k-medie partiziona i dati in k clusters. Ogni cluster possiede un punto centrale che si chiama **centroide**, tale che usualmente rappresenta il cluster ed è semplicemente la media di tutti i punti in un cluster. Al passo iniziale, l'algoritmo seleziona casualmente k data points come seme del centroide, dopodiché calcola la distanza tra il centroide di ciascun cluster ed ogni data point. I data points poi vengono assegnati al centroide più vicino. Una volta svolto l'assegnazione di tutti i data points al gruppo di appartenenza, iterativamente si ricalcola il centroide di tutti i cluster, e così anche la riassegnazione dei data points ai clusters. Questo processo si ripete fino alla convergenza del criterio. Le condizioni di convergenza sono le seguenti:

1. nessuna riassegnazione di data points a clusters differenti.
2. nessun cambiamento di centroide.
3. minima diminuzione nella somma degli errori quadratici (SSE).

dove

$$SSE = \sum_{j=1}^{k} \sum_{x \in C_j} dist(\mathbf{x}, \mathbf{m})^2 \tag{4.44}$$

tale che k è il numero di clusters richiesti. Siano poi C_j la *j-esima* cluster, $\mathbf{m_j}$ il centroide del cluster C_j (vettore medio di tutti i data points in C_j), e $dist(\mathbf{x}, \mathbf{m_j})$ è la distanza tra i data point $\mathbf{X}$ e centroide $\mathbf{m_j}$. Nello **spazio Euclideo**, la media di un cluster è così definita:

$$\mathbf{m_j} = \frac{1}{|C_j|} \sum_{x_i \in C_j} \mathbf{x}_i, \tag{4.45}$$

dove $|C_j|$ è il numero di data points in cluster C_j. La distanza da un data point $\mathbf{x_i}$ al cluster con la media $\mathbf{m_j}$ è definita come segue

$$\begin{aligned} dist(\mathbf{x_i}, m_j) &= \|\mathbf{x_i} - \mathbf{m_j}\| \\ &= \sqrt{(x_{i1} - m_{j1})^2 + (x_{i2} - m_{j2})^2 + ... + (x_{ir} - m_{jr})^2} \end{aligned} \tag{4.46}$$

Uno dei problemi con l'algoritmo di k-medie è quello che alcuni clusters potrebbero essere vuote durante il processo di clustering, per il

fatto dell'esaurimento dell'assegnazione di data points, i tali gruppi vengono chiamati **clusters vuoti**. Per processare un cluster vuoto, è possibile scegliere un data point come centroide di sostituzione.

Algoritmo *k-medie.*

Uno dei metodi divisivi è l'algoritmo di *k-medie* (*k-means* - Algoritmo 21). Per la semplicità ed efficienza, questo algoritmo è tra gli algoritmi più utilizzati. Dato un insieme di dati, dove a cui vengono richiesti k clusters (dove k è richiesto dall'utente), questo algoritmo iterativamente divide i dati in k partizioni di clusters basando su una funzione di distanza. L'algoritmo k-medie può essere applicata nei datasets dove la **media** può essere definita e calcolata.

Algorithm 21 k-means(k, D)

```
1: choose k data points as the initial centroids (cluster centers)
2: repeat
3:     for each data point x ∈ D do
4:         compute the distance from x to each centroid;
5:         assign X to the closest centroid;        ▷ Un centroide rappresenta un raggruppamento
6:     end for
7:     re-compute the centroid using the current cluster memberships
8: until the stopping criterio is met
```

Algoritmo *K*-medie (*disk-version*)

L'algoritmo k-medie può essere implementato in modo tale che non ci sia la necessità di caricare l'intero dataset nella memoria centrale, un fattore molto utile per per i dataset grandi. I centroidi per per i k clusters possono essere calcolati in modo incrementale in ciascuna iterazione perché al primo passo la somma dell'espressione ($\mathbf{m_j}$) può essere calcolata separatamente. E durante il processo di clustering, il numero di data points in ogni cluster può anche esso essere contato incrementando. Pertanto così ci rende un implementazione basata su un disco dell'algoritmo 21, che produce esattamente lo stesso risultato rispetto all'algoritmo in 22, però solamente i dati sul disco. In ogni ciclo, l'algoritmo semplicemente scansiona i dati una volta. In tutto il processo di clustering quindi scansiona i dati t volte, dove t è il numero di iterazioni prima della convergenza, che solitamente

non è molto grande (< 50). Nelle applicazioni è abbastanza comune impostare un limite il numero di iterazioni, poiché tipicamente nelle iterazioni successive risulteranno cambiamenti minori.

Algoritmo 22. Il primo passo dell'algoritmo è analogo alla versione classica. Dopodiché si istanzia il vettore $\mathbf{s_j}$ che è usato per calcolare incrementando la somma nell'equazione (4.45) (riga 8), e poi (riga 4) si inizializza n_j dove si registra il numero di data points assegnato al cluster j (riga 9). Nelle righe 6 e 7 si eseguono esattamente le stesse funzioni come nelle righe 4 e 5 nell'algoritmo originale. La riga 11 ricalcola i centroidi che è usato nell'iterazione successiva. Se viene applicata la somma degli errori quadratici, è possibile modificare leggermente l'algoritmo per calcolare la somma quadratica incrementale.

Algorithm 22 disk-k-means(k, D)

1: choose k data points as the initial centroids $\mathbf{m_j}$, $j = 1, ..., k$
2: **repeat**
3: initialize $\mathbf{s_j} \leftarrow \mathbf{0}, j = 1, ..., k$
4: initialize $n_j \leftarrow 0, j = 1, ..., k$
5: **for** each data point $\mathbf{x} \in D$ **do**
6: $j \leftarrow \arg\min_{i \in 1,2,...,k} dist(\mathbf{x}, \mathbf{m})$
7: assign $\mathbf{x}$ to the cluster j;
8: $\mathbf{s_j} \leftarrow \mathbf{s_j} + \mathbf{x}$
9: $n_j \leftarrow n_j + 1$
10: **end for**
11: $\mathbf{m_j} \leftarrow \mathbf{s_j}/n_j, j = 1, ..., k$;
12: **until** the stopping criterio is met

Vantaggi: La ragione principale dell'algoritmo k-medie è la sua semplicità ed efficienza, tale che è facile da implementare e comprendere. Il suo tempo di complessità è $O(tkn)$, dato n il numero di data points, k è il numero di clusters, e t è il numero di iterazioni. Si assume che k e t siano più piccoli di n. L'algoritmo k-medie in numero di data points è considerato un algoritmo lineare.

Svantaggi: L'algoritmo è applicabile solamente dove è possibile definire la **media**. Pertanto risulta essere più difficile applicare ai dati categoriali. Una variante di algoritmo k-medie è k-**modalità**,

il quale raggruppa i dati categoriali, dove si usa la modalità invece della media come centroide. Assumendo che le istanze di dati siano descritte da r attributi categoriali, la modalità del cluster C_j è una tupla $\mathbf{m_j} = (m_{j1}, m_{j2}, ..., m_{jr})$ dove m_{ji} è il valore più frequente del i-esimo attributo dei dati del gruppo C_j. La similarità tra un'istanza di dati la modalità è il numero di valori che accoppia o meno. Un altro inconveniente è che sia l'utente a specificare il numero di gruppi in anticipo. In senso concreto si dovrebbe provare diversi k valori, e così i risultati più auspicabili verrebbero forniti. Inoltre l'algoritmo è sensibile a valori anomali. Uno dei metodi più semplici per trattare i valori anomali è rimuovere qualche data point che si distanziano troppo lontano dal centroide. L'algoritmo è sensibile ai **semi iniziali**, con un seme iniziale diverso potrebbe risultare clusters diversi, pertanto se la somma dei quadrati è usata come criterio di fermata, l'algoritmo raggiunge solamente l'ottimo locale. Un altro difetto dell'algoritmo k-medie è che non è compatibile per la scoperta dei clusters quali non sono iper-ellissoidi.

4.6.2 Funzioni di Distanza

Le funzioni di distanza (o di similarità) giocano un ruolo centrale in tutti gli algoritmi di raggruppamento. Quelle comunemente più utilizzate sono la distanza **Euclidea** e quella di **di Manhattan**, entrambe le misure sono una generalizzazione della **distanza di Minkowski**. Sia $dist(\mathbf{x_i}, \mathbf{x_j})$ la funzione per denotare la distanza tra due punti di dati di r dimensioni. La distanza di Minkowski è definita come segue:

$$dist(\mathbf{x_i}, \mathbf{x_j}) = (|x_{i1} - x_{j1}|^h + |x_{i2} - x_{j2}|^h + ... + |x_{ir} - x_{jr}|^h)^{\frac{1}{h}} \tag{4.47}$$

dove h è un numero intero positivo. Se h=2, è la **distanza Euclidea**,

$$dist(\mathbf{x_i}, \mathbf{x_j}) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + ... + (x_{ir} - x_{jr})^2} \tag{4.48}$$

Se h=2 è la **distanza di Manhattan**

$$dist(\mathbf{x_i}, \mathbf{x_j}) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + ... + |x_{ir} - x_{jr}|. \tag{4.49}$$

Oltre a quelli citati sopra, a seconda del contesto, è anche possibile altre funzioni come la distanza *Euclidea pesata*, distanza *Euclidea quadratica*, distanza di *Chebychev* ecc.

Variabili dicotomici e nominali. Per le variabili dicotomiche esistono funzioni di distanza tali che sono basate sulla proporzione di valori che si accoppiano due data points. Un accoppiamento implica che per una particolare variabile, entrambi i data points possiedono lo stesso valore. Risulta esser molto facile fare uso matrice di confusione per introdurre a queste misure. Per dare le funzioni di distanza si dividono le variabili dicotomiche in simmetriche ed asimmetriche. In quella simmetrica, la funzione comunemente usata è la **distanza di accoppiamento semplice**, tale che è la proporzione di valori di corrispondenza errata. Mentre per le variabili asimmetriche, in genere si usa **la distanza di Jaccard**.

4.6.3 Rappresentazione Gerarchica

Il metodo gerarchico formula i gruppi in sequenze annidate come **alberi** (o **dendrogramma**). I gruppi con un unico data point è chiamato come singoletto, i quali si sono situati in fondo dell'albero, e la radice in testa, che copre tutti i data points. Ogni nodo all'interno dell'albero contiene nodi figli. I fratelli si raggruppano i data points coperti da un loro nodo genitore comune. Ci sono due tipi di raggruppamenti gerarchici: **Metodo agglomerativo** e **Metodo divisivo**. Il primo metodo costruisce il dendrogramma dal livello basso, e ad ogni livello unisce i gruppi simili. Il processo continua fino ad arrivare un gruppo singolo, ovvero la radice. Il secondo metodo inizia a partire da un unico gruppo che comprende tutti i data points, divide la radice in un insieme di gruppi figli. Ogni gruppo figlio viene ricorsivamente diviso fino ad arrivare al cluster singoletto di ciascun data point rimasto. Il metodo agglomerativo è più usato rispetto al metodo divisivo. A differenza dell'algoritmo k-medie, che usa solamente i centroidi sul calcolo delle distanze, il raggruppamento gerarchico può qualsiasi metodo per determinare la distanza tra i due clusters.

Il vantaggio della rappresentazione gerarchica rispetto all'algoritmo k-medie e altri metodi di raggruppamento è che il quale è capace di trattare qualsiasi forma di distanza o funzioni di similarità. Inoltre, diversamente dall'algoritmo k-medie che restituisce k clusters alla fine, le gerarchie dei clusters permette all'utente di esplorare i dettagli dei cluster a qualsiasi livello. Alcuni svantaggi del rag-

gruppamento gerarchico è che nel legame singolo subisce l'effetto catena, ed il metodo del legame completo è sensibile ai valori anomali. Inoltre il raggruppamento gerarchico è computazionalmente più costoso, tali che sono almeno di ordine quadratico, che messo in confronto con l'algoritmo k-medie, tale quale è molto inefficiente e poco pratico per i data sets troppo grandi. È possible campionare i dati per evitare il problema della dimensionalità. Dopo il raggruppamento, i data points rimanenti si possono essere assegnati a ciascun gruppo, oppure basando sul confronto delle distanze, od anche per apprendimento supervisionato. Inoltre possono esser applicati alcuni metodi di tipo **scale-up** per trovare tanti gruppi piccoli dai dataset grandi.

Algoritmo 23 Agglomerative(D). Clustering gerarchico

1: Make each data point in the data set D a cluster
2: Compute all pair-wise distances of $x_1, x_2, ..., x_n \in D$;
3: **repeat**
4: find two clusters that are nearest to each other;
5: merge the two clusters form a new cluster c;
6: compute the distance from c to all other clusters;
7: **until** there is only one cluster left

Metodo del legame singolo. Nel raggruppamento gerarchico a legame singolo, la distanza fra i due gruppi è la distanza tra i due data points (di due clusters diversi) più vicini. Il metodo a legame singolo è adatto per trovare le forme di clusters non ellittici. Tuttavia, può essere sensibile sulla dispersione nei dati, che può causare l'effetto catena producendo dei clusters ruvidi. Con una struttura dati adeguata, il raggruppamento gerarchico a legame singolo, in tempi computazionali può essere fatta in $O(n^7)$, dove n è il numero di data points. Tale quale è molto più lento che l'algoritmo a k-medie, che raggruppa i cluster in tempi lineari.

Metodo del legame completo. In questo tipo di raggruppamento, la distanza tra i due gruppi è la massima distanza di tutti i data points. In altre parole, a ciascun passo si unisce due gruppi, basando sulla più piccola tra le distanze massime. Sebbene il metodo del legame completo non ha problemi di effetti a catena, esso può essere

sensibile a valori anomali. Nonostante questa limitazione, si osserva che il metodo del legame completo solitamente produce gruppi migliori. Il caso peggiore è la complessità di tempo del metodo a legame completo, tale che è $O(n^2 \log n)$, dove n è il numero di data points.

Metodo del legame intermedio. Come impatto, questo metodo è una compromessa tra la sensitività del metodo di legame completo, i valori anomali e la tendenza del metodo di legame singolo di formare catene non corrispondono alla nozione intuitiva dei clusters. In questo metodo, la distanza tra due gruppi è la distanza media di tutte le distanze tra i data points in due gruppi. Analogamente anche questo metodo possiede un tempo di complessità pari a $O(n^2 \log n)$.

4.6.4 Rappresentazione dei Gruppi

Una volta trovato un insieme di gruppi, il passo successivo è quello di trovare il modo per rappresentarli. Nelle applicazioni che coinvolgono i processi decisionali, i gruppi risultanti devono essere rappresentati in modo compatto e comprensibile, tali quali devono anche facilitare la loro valutazione. Ci sono tre vie principali per rappresentare i gruppi:

- Usare il centroide di ciascun cluster a rappresentare il gruppo è il modo più popolare. È anche possibile calcolare calcolare il raggio e la deviazione standard del cluster per determinare la distribuzione in ciascuna dimensione. Tramite il centroide si può rappresentare molto bene i cluster con forma iper-sferica, quelli con forme allungate invece i centroidi sono poco compatibili.
- Uso di modelli di classificazione per rappresentare i clusters, ovvero trattare ogni cluster come una classe.
- Uso di valori frequenti in ciascun cluster è un metodo usato principalmente per il raggruppamento di dati categoriali. Inoltre è anche considerato come metodo chiave in raggruppamento di testi, dove un insieme piccolo di parole frequenti di ciascun gruppo viene selezionato per la rappresentazione.

In Generale i gruppi dalle forme iper-ellittiche e iper-sferiche sono più facili da rappresentare, usando i loro centroidi insieme alla distribuzione, regole o combinazione di entrambi, per la forma del cluster invece l'algoritmo k-medie non sono capaci di discriminare arbitrariamente.

4.6.5 Valutazione dei Raggruppamenti

Ogni algoritmo di rappresentazione di cluster ha dei limiti e si adatta bene solo con certe distribuzioni dei dati. Tuttavia non è facile sapere a priori la distribuzione ideale da assumere per i dati. Nel peggio l'algoritmo potrebbe non avere nessuna distribuzione ideale. Oltre nella scelta dell'algoritmo di raggruppamento, si decide il modo per standardizzare i dati, scelta di una funzione di distanza adatta e la selezione dei parametri. Dovuta a queste complessità, la pratica comune è provare diversi algoritmi, usando funzioni di distanza diverse, e poi analizzare e confrontare accuratamente i risultati. L'interpretazione dei risultati deve essere basata sull'intuizione dei dati originali assieme alle conoscenze degli algoritmi. Dal momento cui è stato elaborato un in insieme di clusters, occorre valutare la bontà di adattamento. Diversamente dalla classificazione, dove l'accuratezza si misura basando i dati aventi attributi, al clustering non è stato fornito alcun indizio sul raggruppamento corretto. La qualità del raggruppamento rispetto al caso della classificazione è più difficile da valutare. Alcuni strumenti di valutazione proposti sono: *ispezione utente* e *ground truth*.

Ispezione Utente. Agli esperti del campo vengono chiesti di ispezionare i clusters risultanti per poi valutarli. Poiché questo è un processo soggettivo, si prende in considerazione il punteggio medio di tutti gli esperti come come punteggio finale di clustering. Questa è una ispezione manuale, è palese che è un lavoro intensivo e che richiede molto tempo. L'ispezione è necessaria in alcuni livelli di applicazioni perché non vi sono metodi di valutazione adeguati che garantiscono la qualità dei clusters finali.

Ground Truth. In questo metodo i dati di classificazione vengono utilizzati per valutare gli algoritmi di clustering, assumendo che ciascuna classe corrisponda ad un cluster. Se un dataset avesse b classi allora si assume di avere b cluster in corrispondenza. Dopodiché si confrontano i membri. Le misure per la valutazione possono essere gli indici usati in apprendimento supervisionato, i quali come *entropia*, *purità*, *richiamo* e *precisione*. Per facilitare la valutazione, una matrice di confusione può essere costruita basando i clusters risultanti. Così varie misure possono essere calcolate.

4.7 LU e PU Learning

4.7.1 LU Learning

In *LU learning*, si assume che vi sia solamente un piccolo insieme di esempi etichettati. L'obiettivo è fare uso di esempi non etichettati per migliorare l'apprendimento. Il principale collo di bottiglia di un classificatore è quando che un numero considerevole di etichette dei documenti di training set (spesso proibitive) necessitano di una classificazione accurata. Nella classificazione dei testi, l'etichettatura è spesso manuale leggendo i testi, tale che è un compito che richiede tempo. Tuttavia, non è possibile eliminare completamente l'etichettatura perché in assenza di quello, l'algoritmo di apprendimento non potrà sapere in cosa l'utente sia interessato. Un piccolo insieme di esempi di etichette non è sufficiente per costruire un classificatore accurato, mentre una grande quantità di esempi non etichettati possono essere di supporto. Nel contesto di classificazione dei testi, la ragione per cui i dati non etichettati possono essere d'aiuto è che forniscono informazioni sulla probabilità congiunta della distribuzione delle parole.

Classificazione EM Naïve Bayesiano

L'algoritmo *EM* (*Expectation Maximization*) si consiste in **passo di aspettazione (o E-step)** e **passo di massimizzazione (o M-step)**. L'E-step riempie i dati mancanti basando sulla stima dei parametri correntemente presenti. L'M-step massimizza la verosimiglianza, ristima i parametri. Di conseguenza questo ci porta all'iterazione

successiva dell'algoritmo, e così via. Quando i parametri del modello si stabilizzano, l'EM si converge in un minimo locale. L'abilità dell'EM è lavorare con i dati mancanti tali che è necessario per l'apprendimento dagli esempi etichettati e non quelli non etichettati. I documenti nell'insieme nell'insieme etichettato (denotato da L) possiedono tutti etichette (o valori). I documenti nell'insieme non etichettato (denotato da U) può essere considerato come avere etichette classi mancanti. È possibile usare EM per stimarli basando sul modello corrente, per assegnare un'etichetta classe a ciascun documento d_i in U, $Pr(c_j|d_i)$. Dopo un certo numero di iterazioni, tutte le probabilità si convertono. L'algoritmo naïve Bayes (NB) come algoritmo di base esegue iterativamente. I parametri che l'EM stima in questo caso sono le probabilità di ciascuna parola data una classe e le classi di probabilità a priori. Specificatamente, si costruisce per primo un classificatore f usando esempi etichettati in L. Si usa pertanto f per classificare gli esempi non etichettati in U, più accuratamente ad assegnare una probabilità a ciascuna classe per ogni esempio non etichettato, come quanto $Pr(c_j|d_i)$ che prende i valori da [0,1] invece di {0,1}.

Sia $C = \{c_1, c_2, ..., c_{|C|}\}$ un insieme di classi. Ogni iterazione di EM assegnerà ciascun esempio d_i in U una distribuzione di probabilità sulle classi che potrebbe appartenere. Questo è assegnare a d_i la classe di probabilità di $Pr(c_1|d_i), Pr(c_2|d_i), ..., Pr(c_{|C|}|d_i)$. È differente dall'esempio in insiemi etichettati L, dove ogni documento appartiene solamente ad una classe singola c_k come $Pr(c_k|d_i) = 1$ e $Pr(c_j|d_i) = 0$ per $j \neq k$.

Basandosi sull'assegnazione di probabilità $Pr(c_j|d_i)$ ad ogni documento in U, quindi un nuovo classificatore NB può essere costruito. Questo nuovo classificatore può usare entrambi gli insiemi etichettati L e gli insiemi non etichettati U come se gli esempi in U avessero le etichette di probabilistiche, $Pr(c_j|d_i)$. Questo ci porta all'iterazione successiva. Il processo continua affinché ai parametri dei classificatori ($Pr(w_t|c_j)$ e $Pr(c_j)$) non vi sono più ulteriori cambiamenti (o cambiamenti minimi). L'algoritmo EM con classificazione NB è proposto per LU learning. Questo algoritmo può essere visto come un metodo di clusterizzazione con qualche seme iniziale (dati etichettati) in ciascun raggruppamento. Le classi etichette dei semi

indicano le etichette di classi di ciascun gruppo risultante. Per la derivazione dell'algoritmo EM, si fanno due assunzioni: il primo è che i dati siano generati da un modello misto; mentre il secondo è che vi sia una corrispondenza uno a uno tra la mistura di componenti e classi. L'algoritmo EM funziona bene se vengono soddisfatte le assunzioni di due modelli bene per un particolare dataset.

4.7.2 PU Learning

In *PU learning* (*apprendimento da esempi positivi ed esempi non etichettati*) si fa l'assunzione di avere due classi per la classificazione, il training set possiede solamente un insieme di esempi etichettati positivi e un insieme di esempi non etichettati. L'obiettivo è costruire un classificatore accurato senza etichettare alcun esempio negativo. Si studiano questi due problemi nei contesti di classificazione dei testi e la classificazione delle pagine web. Le idee ed algoritmi in generale sono anche applicabili ad altri tipi di classificazione. In alcune applicazioni, il problema è identificare una particolare classe P dei documenti da un insieme di documenti misti. Le classi di documenti a cui si interessano nei documenti si chiamano **classi positive**, o semplicemente **documenti positivi**. Il resto dei documenti vengono chiamati classe di **documenti negativi** (o semplicemente **documenti negativi**). Questo può essere visto come un problema di classificazione con due classi, positiva e negativa. Tuttavia, non vi sono documenti etichettati negativi per la stima.

Problema di formulazione: Dato l'insieme P dei documenti positivi a cui si è interessati, ed un insieme U di **documenti non etichettati** (l'insieme misto), tale che contiene entrambi documenti positivi e negativi, si vuole costruire un classificatore usando P e U che identificano documenti positivi in U o in un test set separato - in altre parole, si vuole classificare accuratamente documenti positivi e negativi in U o nel data set test (o data set del futuro). È da notare che l'insieme U può essere usato in entrambi training e testing perché U non è etichettato.

La caratteristica chiave di questo approccio è che non vi siano etichette negative per l'apprendimento. Gli algoritmi di apprendi-

mento tradizionali, pertanto, non sono applicabili direttamente perché tutti essi richiedono documenti etichettati positivi e negativi. Questo è anche valido per il caso di LU learning, sebbene l'insieme etichettato per ciascuna classe potrebbe essere molto piccolo.

4.8 Modelli di Reperimento

Un *modello* di IR è un insieme di costrutti di rappresentazione del contenuto informativo dei documenti e dell'**esigenza informativa** di un'interrogazione. Ogni modello IR definisce una *funzione* di reperimento coerentemente ai costrutti usati per rappresentare esigenze e contenuti informativi. I modelli più usati sono: *modello booleano*, *modello di vettoriale*, *modello linguistico probabilistico*. I modelli di reperimento rappresentano i documenti ed effettuano interrogazioni in vie differenti, entrambi di essi utilizzano lo stesso piattaforma, trattano ciascun documento come un insieme di parole o termini, dove ogni termine è associato ad un peso. La selezione dei documenti avviene tramite una funzione di reperimento calcolabile sui descrittori dei documenti. L'IRS però si limita a predire la rilevanza del documento e a consegnarlo all'utente senza spiegare la causa per la quale il documento soddisfi l'esigenza informativa, perché si presume che sia l'utente a dare l'ultimo giudizio sulla rilevanza. Le funzioni di reperimento danno un *ordinamento* ai documenti, la misura di rilevanza fornita è un numero reale che consente un ordinamento almeno parziale. Pertanto l'ordine della lista dei documenti è data dal loro numero di rango.

Data una collezione di documenti D, sia $V = \{t_1, t_2, ..., t_{|V|}\}$ l'insieme distintivo della collezione, dove t_i è un termine. L'insieme V solitamente è noto come **vocabolario** della collezione, e $|V|$ è la sua taglia, il numero di termini in V. Un peso $w_{ij} > 0$ è associato con ogni termine t_i del documento $\mathbf{d_j} \in D$, per il termine che non compare in documento $\mathbf{d_j}$, $w_{ij} = 0$. Pertanto ogni documento $\mathbf{d_j}$ come un vettore i termini, $\mathbf{d_j} = \{w_{1j}, w_{2j}, ..., w_{|V|j}\}$.

4.8.1 Modello Booleano

Nel modello booleano i descrittori indicano insiemi di documenti e le interrogazioni sono proposizioni logiche definite su insiemi di documenti; gli operatori sono AND, OR, NOT e gli operandi sono i descrittori. La *funzione di reperimento* del modello booleano associa ad ogni documento della collezione un numero positivo, se il documento rende vera l'interrogazione, 0 altrimenti. Pertanto sia:

$$w_{\text{ij}} = \begin{cases} 1 & \text{se } t_i \text{ appare in } d_j \\ 0 & \text{altrimenti} \end{cases}$$

dove w_{ij} è il peso del termine t_i del documento d_i. Il modello logico richiede utenti addestrati all'uso della *logica booleana* ed esperti dell'ambito per il quale è stata preparata la collezione di documenti. Dato un query booleano, il sistema reperisce ogni documento che rende l'interrogazione logicamente vera. Pertanto il reperimento si basa sul criterio della decisione binaria. Intuitivamente si dice accoppiamento esatto. Una delle inconvenienti d'uso del modello logico è riconducibile al sovraccarico cognitivo necessario all'utente per eliminare la confusione tra gli operatori logici AND, OR e NOT e le preposizioni "e", "o" e "non" della lingua naturale.

4.8.2 Modello Vettoriale

Un documento nel modello vettoriale si presenta come un vettore ponderato, nel cui ogni componente ha il peso che è definito da qualche variazione dello schema di **Frequenza di Termini** (TF-IDF o TF). Nel metodo di schema di TF il peso w_{ij} del termine t_i del documento d_i è il numero volte che t_i compare nel documento $\mathbf{d_j}$, al quale è anche possibile applicare la normalizzazione. Lo svantaggio dello schema di TF è che non considera la situazione dove un termine può comparire in molti documenti della collezione, come fosse un termine poco discriminativo.

Sia N il numero totale di documenti nel sistema o nella collezione, e sia df_i il numero di documenti in cui il termine t_i compare almeno una volta. Sia f_{ij} la frequenza empirica di termine t_i del documento $\mathbf{d_j}$. Allora la **frequenza di termini normalizzata** tf_{ij} in $\mathbf{d_j}$ è data da

$$tf_{ij} = \frac{f_{ji}}{max\{f_{1j}, f_{2j}, ..., f_{|V|j}\}} \tag{4.50}$$

dove il massimo è calcolato su tutti i termini dove compaiono in documento $\mathbf{d_j}$, se il termine $\mathbf{d_j}$ allora tf_{ij}=0. L'inverso della frequenza di documenti (idf_i) del termine t_i è data da:

$$idf_i = \log \frac{N}{df_i}. \tag{4.51}$$

L'intuizione è se un termine compare in molti documenti della collezione, e probabilmente non importante o non discriminante. Pertanto il temine TF-IDF pesato è dato da:

$$w_{ij} = tf_{ij} \times idf_i \tag{4.52}$$

Una query $\mathbf{q}$ potrebbe essere rappresentata esattamente allo stesso modo come un documento in collezione di documenti. I pesi dei termini w_{iq} di ciascun termine t_i in $\mathbf{q}$ può anche essere calcolato in un documento normale o leggermente differente.

Reperimento e Rilevanza

Nel modello vettoriale, a differenza del modello booleano non fa la decisione binaria. I documenti sono classificati in base al grado di rilevanza dell'interrogazione. Un modo per calcolare il grado di rilevanza è calcolare l'indice di **similarità** dell'interrogazione $\mathbf{q}$ per ciascun documento $\mathbf{d_j}$ della collezione di documenti D. Dove seconda del contesto, si potrebbe utilizzare qualunque tipo di indice di similarità per la calcolare il punteggio di rilevanza. La misura di similarità potrebbe essere misurata la legge di similarità di coseno, ovvero il *coseno* dell'angolo tra il vettore d'interrogazione $\mathbf{q}$ ed il vettore di documento $\mathbf{d_j}$,

$$cos\,(\mathbf{d_j}, \mathbf{q}) = \frac{\langle \mathbf{d_j} \bullet \mathbf{q} \rangle}{\|\mathbf{d_j}\| \times \|\mathbf{q}\|} = \frac{\sum_{i=1}^{|V|} w_{ij} \times w_{iq}}{\sqrt{\sum_{i=1}^{|V|} w_{ij}^2} \times \sqrt{\sum_{i=1}^{|V|} w_{iq}^2}} \tag{4.53}$$

La classificazione dei documenti viene pertanto fatto usando i loro valori di similarità. Quelli classificati con un rango superiore sono considerati come documenti più rilevanti per l'interrogazione.

Algoritmo 24. L'iterazione 5-13 espande il vettore dell'interrogazione calcolando un nuovo vettore il cui elemento è il peso ottenuto sommando i prodotti dei pesi per la correlazione. L'uso di una struttura ad accesso calcolato permette d'accedere direttamente agli elementi peri quali si ha $b_h \neq 0$, ciò vale per il ciclo 8-11 e per quello 14-22 dell'algoritmo. L'iterazione 14-22 consiste nel reperimento vero e proprio. Nell'algoritmo si utilizzano solo le righe di R in corrispondenza dei descrittori per cui i b_i sono non nulli. Per ciascuno di essi si sommano i valori di correlazione moltiplicati per il peso del descrittore correlato al rispettivo peso b'_j dell'interrogazione espansa.

Algorithm 24 Funzione di IR con il modello vettoriale

1: $b = (b_1, ..., b_n)$ ▷ vettore dei pesi di un'interrogazione di n descrittori
2: R=$[r_{ij}]$ ▷ matrice di correlazione dei descrittori
3: $b = (b'_1, ..., b'_n)$ ▷ vettore dei pesi dell'interrogazione di espansa mediante R
4: $s[1, ..., N]$ ▷ misure di rilevanza di N documenti
5: **for** $j = 1, ..., n$ **do**
6: **if** $b_j \neq 0$ **then**
7: $R_j = \{r_{j1}, ..., r_{jn_j}\}$ ▷ lista di n_j gradi di correlazione con j di altrettanti descrittori
8: **for** $i = 1..., n$ **do**
9: r_{ij} ▷ grado di correlazione tra i descrittori i e j
10: $b'_i \leftarrow b'_i + b_i r_{ij}$
11: **end for**
12: **end if**
13: **end for**
14: **for** $j = 1, ..., n$ **do**
15: **if** $b'_j \neq 0$ **then**
16: $L_j = \{l_{j1}, ..., l_{jn_j}\}$ ▷ lista di n_j identificatori associati a j
17: **for** $i = 1, ..., n$ **do**
18: $w(j, l_{ji})$ ▷ peso di j nel documento l_{ji}
19: $s[l_{ji}] \leftarrow s[l_{ji}] + w(j, l_{ji})$
20: **end for**
21: **end if**
22: **end for**

4.8.3 Modello Indicizzazione Semantica Latente

Molti concetti od oggetti possono essere descritti in vari modi basando sul contesto l'abitudine linguistico delle persone. Nel caso in cui l'utente nell'interrogazione usasse parole diverse rispetto alle parole usate nel documento, il documento non potrà essere reperito, anche nel caso dovesse essere rilevante, pertanto anche nel caso si

trattassero di **sinonimi**. Si assume che nei dati esistano strutture semantiche latenti, tali che sono nascosti dalla casualità della scelta di parola.

Il modello *indicizzazione semantica latente* (LSI) mira a risolvere questo problema attraverso l'identificazione di associazioni statistiche dei termini, dove si utilizza la tecnica statistica per stimare la struttura latente e rimuovere il rumore nei è la **decomposizione a valori singolari**. Il risultati di questa decomposizione sono descrizioni di termini e documenti basata sulla struttura semantica latente derivata da SVD. Questa struttura si chiama anche **spazio di concetto nascosto**, tale che associa sintatticamente i termini e documenti differenti ma semanticamente simili.

Sia D una collezione di testi, sia m il numero di parole diverse nel D. L'LSI inizia con la matrice A di dimensione $m \times n$. Ogni riga di A rappresenta un termine ed ogni colonna un documento. SVD fattorizza la matrice $A_{m \times n}$ in un prodotto di tre matrici,

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathbf{T}}, \text{ dove} \tag{4.54}$$

U è una matrice $m \times r$ e, le colonne sono vettori singolari di sinistra, inoltre sono gli autovettori associati con n autovalori non nulli di AA^T. Le colonne di U sono le unità di vettori ortogonali, tale che $U^T U = I$.

V è una matrice $n \times r$ e, le colonne sono vettori singolari di sinistra, inoltre sono gli autovettori associati con n autovalori non nulli di $A^T A$. Le colonne di V sono le unità di vettori ortogonali, tale che $V^T V = I$.

Σ è una matrice diagonale $r \times r$, $\mathbf{\Sigma} = diag(\sigma_1, \sigma_2, ..., \sigma_r)$, $\sigma_1, \sigma_2, ...,$ e σ_r sono valori singolari, ovvero la radice quadratica di di r autovalori non nulli di AA^T. Essi sono disposti nell'ordine decrescente come segue $\sigma_1 \geq \sigma_2 \geq, ..., \geq \sigma_r > 0$.

Da ciò si evince che m è il numero di righe (termini) in **A**, n è il numero di colonne (documenti) in **A**, e r è il **rango** di **A**, tale che $r \leq min(m, n)$. Una caratteristica importate di SVD è che si può eliminare qualche dimensione insignificante dello spazio trasformando la matrice A approssimata a seguito della ottimizzazione. La significatività delle dimensioni è indicata dai magnitudi di valori singolari in Σ, tale che è già stato ordinato. In information retrieval, le dimensioni non significanti possono rappresentare i "rumori" nei dati

e possono essere rimossi. Si usano k valori singolari più grandi in Σ ed impostare i valori piccoli rimanenti a zero. Di conseguenza è anche possibile ridurre la dimensione della matrice Σ, si ottiene così

$$\mathbf{A_k} = \mathbf{U_k}\mathbf{\Sigma_k}\mathbf{V_k^T}, \tag{4.55}$$

usando k dimensioni per approssimare la matrice di termini-documenti.

Interrogazione e Reperimento

Data un'interrogazione utente $\mathbf{q}$ (rappresentata da un vettore colonna come quelle in A), in primo luogo viene convertito in un documento in spazio di k-concetti, denotato da $\mathbf{q_k}$. Questa trasformazione è necessaria perché SVD trasforma i documenti originali in spazio di k-concetti e li immagazzini in V_k. L'idea è che $\mathbf{q}$ viene trattato come un nuovo documento nello spazio originale rappresentato come colonna in A, e poi mappato a $\mathbf{q_k}$ come un documento addizionale (o colonna) in V_k^T. Dall'equazione (4.55) è facile vedere che

$$\mathbf{q} = \mathbf{U_k}\mathbf{\Sigma_k}\mathbf{q_k^T}. \tag{4.56}$$

Le colonne in U sono unità di vettori ortogonali, $\mathbf{U_k^T U_k} = \mathbf{I}$. Pertanto,

$$U_k^T q = \Sigma_k q_k^T \Longrightarrow \Sigma_k^{-1} U_k^T q = q_k^T \Longrightarrow \mathbf{q_k} = \mathbf{q^T U_k \Sigma_k^{-1}} \tag{4.57}$$

Per il reperimento, confrontiamo semplicemente $\mathbf{q_k}$ con ciascun documento (riga) in V_k usando una misura di similarità (coseno di similarità).

Come risultato, con LSI è possibile adempiere meglio rispetto i metodi basati sugli *keywords*. L'unico inconveniente importante è il tempo di complessità dell'SVD, tale che è $O(m^2 n)$. Pertanto è difficile usare per una grande quantità di collezione di documenti come quello del web. Un altro inconveniente è che il concetto spazio non è interpretabile come la sua descrizione che si consiste in tutti i numeri con un scarso significato semantico. A determinare il numero ottimale di dimensioni k del concetto spazio è anche una difficoltà rilevante. Non vi è un consenso generale per un numero ottimale di dimensioni. È possibile immaginare che le regole di associazione potrebbero approssimare i risultati dell'LSI ed evitare tale difetto.

Le regole di associazione ha due vantaggi. Il primo, il fatto che l'algoritmo di mining è molto efficiente per scopi pratici, l'algoritmo mining ha solamente bisogno di scandire la collezione dei documenti per 2-3 volte. Mentre il secondo è che le regole sono facili da comprendere.

4.8.4 Modello Linguistico Probabilistico

Il modello linguistico probabilistico si basa sulla teoria di probabilità e statistica. Con tale si stima per primo il modello di ogni di ciascun documento e poi la si classifica basando la verosimiglianza del query dato dal modello.

Sia q una sequenza di termini, $q = \{q_1, q_2, ..., q_m\}$, e sia la collezione di documenti D un'insieme di documenti, $D = \{d_1, d_2, ..., d_N\}$. Si considera che la probabilità di una query **q** sia generata da una distribuzione di probabilità basata su un documento d_j, $Pr(q|d_j)$. Per classificare i documenti, siamo interessati a stimare la probabilità a posteriori $Pr(d_j|q)$, utilizzando la legge di Bayes:

$$Pr(d_j|q) = \frac{Pr(q|d_j)Pr(d_j)}{Pr(q)} \tag{4.58}$$

Allo scopo della classificazione, $Pr(q)$ non ha bisogno di essere equiprobabile per ogni documento. Usualmente $Pr(d_j)$ è considerata uniforme e quindi non andrà ad influenzare la classificazione. Dobbiamo solamente calcolare $Pr(q|d_j)$. In molte opere esistenti, il modello linguistico è basata su unigram, ovvero solamente i termini individuali vengono considerati. Il modello assume che ogni termine sia generata indipendentemente, che è essenzialmente una distribuzione multinomiale sulle parole. Il caso generale è il modello n-gram.

$$Pr(q = q_1, 1_2, ..., q_m|d_j) = \prod_{i=1}^{m} Pr(q_i|d_j) = \prod_{i=1}^{|V|} Pr(t_i|d_j)^{f_{iq}}, \tag{4.59}$$

f_{iq} è la frequenza assoluta del termine t_i si verifica in q, e si assume che $\sum_{i=1}^{|V|} Pr(t_i|d_j) = 1$. Il problema di reperimento si riduce alla stima di $Pr(t_i|d_j)$, tale che può essere una frequenza relativa,

$$Pr(t_i|d_j) = \frac{f_{ij}}{|d_j|}, \tag{4.60}$$

per ogni f_{ij} il numero di volte in cui il termine t_i si verifica in documento d_j, e $|d_j|$ denota il numero totale di parole in d_j.

Classificazione dei Testi

Nel metodo di apprendimento di Naïve Bayesiano per la classificazione dei testi, si assume che ogni documento sia generata da una distribuzione parametrica. Tali parametri sono ignoti. Per stimare questi ultimi, occorre prendere in uso i dati del *training set*. Dopodiché i quali vengono usati per classificare i documenti di testo usando le leggi di Bayes, e calcolando le **probabilità a posteriori** tale che la distribuzione associata ad una classe potrebbe aver generato il documento dato. Nella stima del modello si assume che i dati siano generati da un modello misto, ed esiste una corrispondenza univoca tra la mistura di componenti e classi di documenti. Un modello misto modella i dati con un certo numero di distribuzioni statistiche. Intuitivamente, ogni distribuzione corrisponde ad un cluster di dati e parametri. Ogni distribuzione in un modello misto è anche chiamato **componente misto**.

Sia K il numero di componenti misti in un modello misto, e la j-esima distribuzione che possiede parametri θ_j. Sia Θ l'insieme di parametri di tutti componenti, $\Theta = \{\varphi_1, \varphi_2, ..., \varphi_K, \theta_1, \theta_2, ..., \theta_K\}$, dove φ_j è il peso misto (probabilità mista) della componente mista j e θ_j è l'insieme di parametri della componente j. I pesi misti sono soggetti al vincolo $\sum_{j=1}^{K} \varphi_j$. Si assume la corrispondenza biunivoca tra i componenti misti e le classi, dove ogni classe corrisponde ad una componente mista. Pertanto $|C| = K$ tale che $C = \{c_1, c_1, ..., c_{|C|}\}$ sono le classi, quindi la j-esima componente mista corrisponde alla class c_j ed parametrizzata da θ_j. I pesi misti sono le classi di probabilità a priori, tale che, $\varphi_j = Pr(c_j|\Theta)$. Il modello misto genera ciascun documento d_j da:

1. selezione di un componente misto (o classe) basando su probabilità a priori (pesi misti), $\varphi_j = Pr(c_j|\Theta)$;

2. generare un documenti d_j in base ai i suoi parametri, con distribuzione $Pr(d_j|c_j;\theta_j)$.

La distribuzione di probabilità che genera un documento d_j è la seguente:

$$Pr(d_j|\Theta) = \sum_{j=1}^{|C|} Pr(c_j|\Theta) Pr(d_r|c_j;\Theta). \tag{4.61}$$

Dal momento in cui ad ogni classe è stata assegnata la propria etichetta classe, è possibile derivare il modello di naïve Bayes. Il parametro Θ dell'espressione sopra indica l'assunzione dell'indipendenza delle classi.

Formulazione del Modello

Un documento di testo si consiste in una sequenza di frasi, e ogni frase si consiste in una sequenza di parole. In base alla complessità della modellazione della sequenza di parole, diverse assunzioni sono derivate dalle assunzioni del classificatore Bayesiano. Il classificatore Bayesiano tratta ciascun documento come un insieme di parole. Le assunzioni sono simili a quelle del naïve Bayes citato in precedenza. Con tali assunzioni ciascun documento può essere considerato come essere generato da una **distribuzione multinomiale**. Le parole sono date dal vocabolario $V = \{w_1, w_2, ..., w_{|v|}\}$, tale che $|V|$ è il numero di parole nel vocabolario. Ora si assume che n sia il la lunghezza di un documento, e k risultati corrispondano a tutte le parole nel vocabolario $V(k = |V|)$, e $(p_1, p_2, ..., p_k)$ corrispondono alle probabilità di occorrenze delle parole in V di un documento, i quali sono $Pr(w_t|c_j;\Theta)$. Pertanto sia X_t la variabile casuale che rappresenta il numero di volte che la parola w_t appaia in un documento. È pertanto possibile applicare la funzione di probabilità di una distribuzione multinomiale per trovare le probabilità di un data la classe di un documento (includendo la probabilità della lunghezza del documento, $Pr(|d_i|)$) è assunto come indipendente dalla classe:

$$Pr(d_i|c_j;) = Pr(|d_i|)|d_j|! \prod_{t=1}^{|V|} \frac{Pr(w_t|c_j;\Theta)^{N_{ti}}}{N_{ti}!} \tag{4.62}$$

dove N_{ti} è il numero volte che si verifica w_t nel documento d_i t.c.

$$\sum_{t=1}^{|V|} N_{ti} = |d_i|, \ e \ \sum_{t=1}^{|V|} Pr(w_t|c_j;\Theta) = 1 \tag{4.63}$$

Il parametro θ_j delle componenti generative per ciascuna classe sono le probabilità di tutte le parole w_t in V, denotato come $Pr(w_t|c_j;\Theta)$, e le probabilità delle lunghezze dei documenti, tali che sono le stesse per tutte le classi dovute alle assunzioni.

Stima del parametro: I parametri possono essere stimati dai dati del *training set* $D = \{D_1, D_2, ..., D_{|C|}\}$, basando sui calcoli empirici, dove D_j è il sottoinsieme di dati per la classe c_j. La probabilità stimata della parola w_t data dalla c_j, è il numero volte che si verifica w nel training set D_j diviso per il numero totale delle occorrenze:

$$Pr(w_t|c_j;\hat{\Theta}) = \frac{\sum_{i=1}^{|D|} N_{ti} Pr(c_j|d_i)}{\sum_{s=1}^{|V|}\sum_{i=1}^{|D|} N_{si} Pr(c_j|d_i)} \tag{4.64}$$

Nell'equazione (4.64) non viene usato D_j esplicitamente, si include invece $Pr(c_j|d_i)$ per raggiungere lo stesso effetto perché $Pr(c_j|d_i) = 1$ per ogni documento in D_j e $Pr(c_j|d_i) = 0$ per i documenti di altre classi. Di nuovo N_{ti} è il numero volte che si verificano le parole w_t in d_i. Al fine di gestire i conteggi 0 delle parole non frequenti che appaiono nel training set, che potrebbe apparire nel test set, occorre lisciare la probabilità per evitare le probabilità 0 o 1. La via standard è aggiungere il conteggio di ciascuna parola distinta una piccola quantità $\lambda(0 \leq \lambda \leq 1)$. Pertanto ogni parola potrà avere una piccola probabilità di occorrenza. Si definisce allora il **Lisciamento di Lidstone**:

$$Pr(w_t|c_j;\hat{\Theta}) = \frac{\lambda + \sum_{i=1}^{|D|} N_{ti} Pr(c_j|d_i)}{\lambda|V| + \sum_{s=1}^{|V|}\sum_{i=1}^{|D|} N_{si} Pr(c_j|d_i)} \tag{4.65}$$

Infine anche le classi di probabilità, tali che sono i pesi misti φ_j sono stimati usando i dati del training set.

$$Pr(c_j|\hat{\Theta}) = \frac{\sum_{i=1} |D| Pr(c_j|d_i)}{|D|}. \tag{4.66}$$

Classificazione

Dopo avere stimato i parametri del modello, occorre stimare le probabilità di ciascuna classe c_j per il documento di testo d_i. Tale che

si calcola la probabilità dei componenti misti c_j generato dal dato documento d_i:

$$Pr(c_j|d_j;\hat{\Theta}) = \frac{Pr(c_j|\hat{\Theta})Pr(d_i|c_j;\hat{\Theta})}{Pr(d_i|\hat{\Theta})} = \frac{Pr(c_j|\hat{\Theta})\prod k = 1^{|d_i|}Pr(w_{d_i,k}|c_j;\hat{\Theta})}{\sum_{r=1}^{|C|}Pr(c_r|\hat{\Theta})\prod_{k=1}^{|d_i|}Pr(w_{d_i,k}|c_r;\hat{\Theta})} \quad (4.67)$$

dove $w_{d_i,k}$ è la parola nella posizione k del documento d_i. Se il classificatore finale è a classificare ogni documento in una classe singola, la classe con la più alta classe a posteriori viene selezionata:

$$\underset{c_j \in C}{\operatorname{argmax}} Pr(c_j|d_i;\hat{\Theta}). \quad (4.68)$$

Un *documento* di testo si consiste in una sequenza di sequenze, e ciascuna sequenza si consiste in una sequenza di parole. Tuttavia dovuta alla complessità della modellazione di sequenza di parole, e le loro relazioni, diverse assunzioni sono fatto dalla derivazione del classificatore Bayesiano. Specificatamente, la classificazione di naïve Bayesiana tratta ciascun documento come una "borsa" di parole.

4.9 Feedback di Rilevanza

Il *feedback* di rilevanza è un metodo di per migliorare l'efficacia di reperimento. Tale che è un processo dove l'utente identifica alcuni documenti rilevanti ed altri irrilevanti nelle liste iniziali di reperimento di documenti. Pertanto il sistema crea un interrogazione ampliato, estraendo alcuni termini addizionali dai documenti rilevanti e non irrilevanti. Inoltre il sistema produce anche un modello di classificazione usando documenti rilevanti e non rilevanti nelle collezioni di documenti. Il processo di feedback di rilevanza può essere ripetuta più volte finché l'utente sia soddisfatto del risultato reperito.

4.9.1 Metodi di Machine Learning

Dal momento in cui abbiamo in possesso un insieme di documenti rilevanti e irrilevanti, è possibile costruire un modello di classificazione da essi. Allora il problema di feedback di rilevanza diventa un problema di apprendimento automatico. Qualsiasi metodo di

apprendimento supervisionato può essere usato, come per esempio la classificazione di Naïve Bayes e SVM. Le misure di similarità non vengono più usati.

Metodo di Rocchio

Il metodo di Rocchio usa l'approccio di rilevanza dell'utente, la nuova interrogazione ampliata non è altro che ancora un reperimento. Sia $\mathbf{q}$ il vettore dell'interrogazione originale e, siano D_r e D_{ir} gli insiemi di documenti rispettivamente rilevanti e irrilevanti. L'interrogazione ampliata $\mathbf{q}_e$ è definita come segue:

$$\mathbf{q}_e = \alpha\mathbf{q} + \frac{\beta}{|D_r|}\sum_{\mathbf{d}_{ir}\epsilon D_r}\mathbf{d}_r - \frac{\gamma}{|D_{ir}|}\sum_{\mathbf{d}_{ir}\epsilon D_{ir}}\mathbf{d_{ir}}, \tag{4.69}$$

dove α, β e γ sono parametri. L'equazione (4.69) semplicemente aggiunge all'interrogazione originale $\mathbf{q}$ dei termini addizionali dai documenti rilevanti. Si mantiene la query originale $\mathbf{q}$ perché riflette direttamente all'informazione che l'utente richiede. La sottrazione viene usata per ridurre l'influenza di quei termini che sono poco discriminanti, e i termini che appaiono solamente nei documenti irrilevanti. I tre parametri sono assegnati empiricamente.

Classificazione di Rocchio

Una variante del metodo di Rocchio è la classificazione di *Rocchio*. Il quale si consiste in un vettore prototipo $\mathbf{c_j}$ per ogni classe i, tali quali possono essere rilevanti o irrilevanti:

$$\mathbf{c}_i = \frac{\alpha}{|D_i|}\sum_{\mathbf{d}\epsilon D_i}\frac{\mathbf{d}}{\|\mathbf{d}\|} - \frac{\beta}{\|D - D_i\|}\sum_{\mathbf{d}\epsilon D - D_i}\frac{\mathbf{d}}{\|\mathbf{d}\|}, \tag{4.70}$$

dove D_i è l'insieme di documenti di classi i e, α e β sono parametri. Usando lo schema di ponderazione termini TF-IDF, per α=16 e β=4 solitamente si potrebbe lavorare bene. Nella classificazione, viene applicata la similarità del *coseno*. Ogni documento test $\mathbf{d}_t$ viene confrontato con ogni prototipo $\mathbf{c}_i$ basando su similarità di coseno. $\mathbf{d}_t$ viene assegnato alla classe con il valore di similarità più alta.

Algoritmo 25. In pratica, una volta che si ha in possesso i documenti rilevanti e irrilevanti, è possibile costruire modelli di classificazione da essi. Pertanto da cui si diventa un approccio di apprendimento, dove qualsiasi metodo supervisionato può essere applicato.

Algorithm 25 Apprendimento di Rocchio

1: **for** each class i **do**
2: construct its prototype vector $\mathbf{c_i}$ using equation (4.70)
3: **end for**
4: **for** each text document $\mathbf{d_t}$ **do**
5: the class of $\mathbf{d_t}$ is argmax_i $cosine(\underline{\mathrm{d}}_t, c_i)$
6: **end for**

Oltre i metodi di classificazione citati finora, sono applicabili anche le tecniche seguenti:

LU Learning. Dal momento in cui la quantità dei documenti rilevanti e non rilevanti da selezionare possano essere piccoli, potrebbe essere difficile creare un classificatore accurato. Il LU learning (*apprendimento da esempi etichettati e non etichettati*) serve a quando il numero di documenti rilevanti e irrilevanti possano essere piccoli, diventa difficile costruire un classificatore accurato. Tuttavia, gli esempi non etichettati come quei documenti che non sono stati selezionati dagli utenti, possono essere utilizzati per migliorare l'apprendimento a produrre un classificatore più accurato. I documenti rilevanti e irrilevanti selezionati formano il training set etichettato piccolo.

PU Learning. (*apprendimento da esempi positivi ed esempi non etichettati*). I due metodi menzionati sul feedback di rilevanza (Rocchio e Apprendimento), assumono che l'utente può confidenzialmente identificare entrambi i documenti rilevanti ed irrilevanti. D'altronde, in alcuni casi, l'utente seleziona solamente i documenti che a loro sembrino rilevanti basandosi sul titolo o le informazioni di riepilogo (come i frammenti di ricerca in ricerca sul web), che sono più verosimilmente veri documenti rilevanti. I documenti che non sono selezionati dall'utente possono non essere trattati come irrilevanti perché esso non li vede. Così, essi possono essere considerati come

documenti non etichettati. Questo è chiamato **feedback implicito**. In questo caso, allo scopo di apprendere, è possibile usare PU learning, come apprendimento da esempi positivi e non etichettati. Si considerano i documenti selezionati da utenti come esempi positivi, e documenti non selezionati come esempi non selezionati. Dove i ricercatori hanno sperimentato con questo approccio sul contesto della ricerca sul web ed hanno ottenuto un buon risultato.

Modelli SVM e Linguistico. Nell'impostazione del feedback implicito, è proposta la classificazione SVM per classificare i documenti non selezionati sulla base dei documenti non selezionati. Dopodiché è da considerare anche l'approccio basato sul modello linguistico.

4.9.2 Feedback di Pseudo-Rilevanza

Il feedback pseudo-rilevanza è un'altra tecnica usata per migliorare l'efficacia. L'idea di base è estrarre alcuni termini (solitamente termini frequenti), dai documenti classificati per primi ed aggiungerli alle interrogazioni originali per formare una nuova interrogazione per un secondo giro di reperimento. Il processo può essere ripetuto di nuovo affinché l'utente sia soddisfatto con i risultati finali. La differenza principale tra questo metodo ed il feedback di rilevanza è che in questo metodo, l'utente non è coinvolto nel processo. L'approccio assume semplicemente che i documenti classificati per primi siano verosimilmente rilevanti. Tramite l'espansione delle interrogazioni, alcuni documenti rilevanti mancati nel giro iniziale può essere reperito per migliorare la prestazione complessiva. Chiaramente, l'efficacia di questo metodo fa affidamento sulla qualità dei termini di espansione selezionati.

Capitolo 5

Misure di Valutazione

5.1 Esaustività e Specificità dell'Indicizzazione in IR

I descrittori si distribuiscono tra i documenti con diverse frequenze dando luogo ad una indicizzazione più o meno complete e precise, l'*esaustività* dell'indicizzazione in information retrieval (IR) è la qualità dei descrittori di un documento di descriverne in modo compiuto il contenuto informativo. L'esaustività è misurata dal numero di descrittori associati a un documento. Si verifica un'indicizzazione esaustiva quando il contenuto informativo di un documento possiede parecchi descrittori, e un descrittore tende a essere assegnato a tanti documenti. Un'elevata esaustività dell'indicizzazione si aumenta indirettamente il numero di documenti associati a ciascun descrittore, in questo modo anche la possibilità che ad un descrittore utilizzato nell'interrogazione siano associati documenti reperiti rilevanti. Viceversa un'indicizzazione specifica assegna ai documenti un numero basso di descrittori al fine di descrivere gli elementi salienti del contenuto nei documenti. Un'indicizzazione esaustiva aumenta il richiamo, ma diminuisce la precisione.

La *Specificità* dell'indicizzazione in IR è la capacità di un descrittore di discriminare i documenti sulla base del loro contenuto informativo, è misurata dal numero di documenti a cui è stato assegnato a pochi documenti e il contenuto informativo di un documento tende ad essere descritto con un numero ragionevole di descrittori. Un'elevata specificità dell'indicizzazione diminuisce indirettamente il numero dei documenti associati a ciascun descrittore, riducendo così la possibilità che ad un descrittore utilizzato nell'interrogazio-

ne siano associati documenti reperiti non rilevanti. Di conseguenza, un'indicizzazione specifica aumenta la precisione, ma diminuisce il richiamo. Il grado di specificità ed esaustività dell'indicizzazione spiega come vi sia una relazione inversa tra richiamo e precisione: e ad elevati tassi di richiamo corrispondono bassi tassi di precisione, e un'elevata precisione corrisponde ad un basso richiamo. La relazione inversa tra richiamo e precisione è osservata da sempre in tutti gli esperimenti condotti da tutti i ricercatori del settore dell'IR: si tratta quindi di una legge empirica accettata.

5.2 Controllo Statistico dell'Indice in IR

Il *controllo statistico dell'indice* consiste nell'uso di modelli statistici di stima e previsione della distribuzione dei descrittori e dei documenti all'interno di un indice. Il controllo è effettuato sia per misurare e organizzare l'indice sia per mantenere esaustività e specificità dell'indicizzazione a livelli tali da massimizzare l'efficacia del reperimento mediante l'individuazione di stop word, parole sbagliate e termini. Uno dei modelli statistici utilizzati è la *power law*

$$p : N \to R \quad p(r) = \frac{k}{r} \tag{5.1}$$

A titolo esemplificativo, considerando una lista di parole ordinata per la frequenza con cui le parole si distribuiscono su un insieme di documenti, *Zipf* (1949) ha osservato che, tra la posizione r della parola nella lista ordinata e la sua frequenza relativa p, vale la *power law* $p(r) = \frac{k}{r}$, dove k è una costante dipendente dal testo e dalla lingua.

La relazione tra parole, o lettere, e la loro frequenza si consiste il principio del *minimo sforzo*. Nelle comunicazioni le persone tendono a usare pochi e semplici simboli per massimizzare l'informazione e, al tempo stesso, come nello scambio di messaggi di tipo SMS o tweet. Di conseguenza poche parole di poche lettere sono utilizzate molto di frequente per esprimere quasi tutti i concetti di un testo. In termini di probabilità, le poche parole o lettere molto usate sono facilmente prevedibili o molto probabili, mentre le tante parole o lettere molto usate sono facilmente prevedibili o molto probabili, mentre le tante parole o lettere raramente usate rappresentano, se

osservate, un evento sorprendente. Queste ultime apportano una quantità d'informazione maggiore delle parole molto frequenti dal momento che la sorpresa con cui sono osservate può essere fonte di informazioni. Un fatto sorprendente aumenta infatti la conoscenza proprio perché non ce lo si aspettava e quindi non faceva parte ancora della conoscenza fino al momento della sua manifestazione. In sintesi, le parole poco frequenti modificano la conoscenza più significativamente, ovvero apportano più informazione, di quelle parole che, manifestandosi frequentemente, hanno già contribuito a consolidare la conoscenza.

5.3 Misurazione dell'Indice in IR

Una delle misure dell'indice è basata sulla *legge di Heap*, che descrive la crescita del dizionario nel tempo. Secondo questa legge, il numero di parole uniche dipende in misura sublineare dalla dimensione della collezione, come descritto quanto segue:

$$v = kn^{\beta} \qquad 10 \leq k \leq 100 \qquad \beta \approx \frac{1}{2} \tag{5.2}$$

dove n è il numero di parole (non uniche) e v è il numero di parole uniche (dimensione del dizionario). La legge di *Heap* è utile per prevedere la crescita e quindi le risorse computazionali consumate dagli indici con il passare del tempo. La misurazione delle **posting list** permette di stimare il costo computazionale delle posting list lunghe. Poiché la gestione delle posting list diventa particolarmente onerosa, in termini computazionali, per il lungo tempo d'accesso necessario a trasferire i posting dal disco alla memoria centrale. La **dimensione delle posting list** è cruciale quando le interrogazioni contengono degli operatori che prendono delle intersezioni tra i corrispondenti insiemi di documenti. La posting list associata ad un descrittore implementa un insieme, per cui l'intersezione richiede il calcolo dei descrittori in comune a due posting list. Il calcolo è particolarmente dispendioso quando si trattano gli insiemi documenti sono molto grandi, in questi casi, è utile ricorrere a delle stime della loro dimensione. La stima della **dimensione del risultato** consiste nel calcolo del numero approssimato di documenti che saranno dati in risposta ad un'interrogazione dal sistema di information retrie-

val: è un numero approssimato perché il calcolo del numero esatto per interrogazioni costituite da più di un descrittore ha un costo computazionale eccessivo rispetto al requisito di fornire i risultati in tempo reale.

Esempio: 5.1. Siano A, B, C tre descrittori (tre insiemi di documenti), la probabilità di osservare un con tutti e tre i descrittori è:

$$P(A \cap B \cap C) = P(A|B \cap C)P(B \cap C) \tag{5.3}$$

Se N è il numero di documenti, si hanno le seguenti stime:

$$P(X \cap Y) = \frac{X \cap Y}{N} \qquad P(X|Y) = \frac{X \cap Y}{Y} \tag{5.4}$$

Di conseguenza,

$$P(A \cap B \cap C) = P(A|B \cap C)P(B|C)P(C) \tag{5.5}$$

Poiché le combinazioni per trovare almeno un documento è $2^3 - 1$, pertanto la stima della distribuzione di probabilità d'osservare un documento in uno dei possibili sottoinsiemi determinati da tre descrittori richiede 7 stime. Per pervenire a una stima in tempi ragionevoli, si assume l'*indipendenza stocastica*:

$$P(A|B) = P(A) \qquad \text{per cui} \qquad P(A \cap B) = P(A)P(B) \tag{5.6}$$

Tuttavia, questa l'assunzione fornisce spesso un'approssimazione imprecisa, per il fatto che ci siano molte coppie di parole dipendenti. Un'assunzione meno forte dell'*indipendenza stocastica condizionata* è la seguente:

$$P(A \cup B|C) = P(A|C)P(B|C). \tag{5.7}$$

5.4 Valutazione dei Modelli e Algoritmi

I dati disponibili D sono divise in due sottoinsiemi disgiunti, il **training set** D_{train} e il **test set** D_{test}, $D = D_{train} \cup D_{test} = \varnothing$. Questo metodo è particolarmente usato quando il dataset D è notevolmente grande. Tutti gli esempi nei dataset originali sono etichettati per valutare il classificatore. Il training set non dovrebbe essere usato nella valutazione come quanto il classificatore è polarizzato, ovvero il classificatore potrebbe sovradattare i dati training, che renderebbe un'accuratezza molto alta sul training set ma un'accuratezza

molto bassa sul test set. Normalmente le percentuali di training set e test test dipendono dipendono dalla taglia del data set. 50-50, e due terzi per il training set e un terzo per il testing set è comunemente usato. Per partizionare D in training e test sets, è possibile utilizzare le seguenti due approcci:

- Viene campionato casualmente un set di esempi di training da D per learning ed usare il resto per il test.
- Se i dati sono stati collezionati col tempo, allora è possibile la prima parte dei dati per *training/learning* e l'ultima parte dei dati per il testing. In molte applicazioni, questo è un approccio più adatto perché quando il classificatore è usato nel mondo reale, i dati provengono dal futuro. Questo approccio pertanto riflette meglio gli aspetti dinamici delle applicazioni.

Campionamento Casuale Multipla. Quando i dataset disponibili sono piccoli, utilizzare i metodi citati sopra può essere inaffidabile, perché il test set potrebbe essere troppo piccolo per essere rappresentativo. Un approccio per trattare il problema è eseguire i campionamenti casuali su metodi per n volte. Ogni volta viene prodotto un training set e un test set differente. Questo produce n indicatori di accuratezza. L'indicatore di accuratezza sui dati è la media di indicatori di accuratezza.

Convalida Incrociata. Quando il dataset è piccolo, l'**n-fold convalida-incrociata** è molto comunemente usato. In questo metodo, i dati disponibili vengono partizionati in n taglie egualmente disgiunte. Ogni sottoinsieme è quindi usato come il test set ed il $n-1$ sottoinsieme rimanente è combinata come training set per apprendere un classificatore. Questa procedura iterativamente gira per n volte, tale che fornisce n indicatori di accuratezza. La stima di accuratezza finale dei apprendimento da questi dati è la media di n indicatori di accuratezza. 10-pieghe e 5-pieghe sono spesso usate. Un caso particolare della convalida incrociata è la **convalida-incrociata leave-one-out**. In questo metodo, ogni piega della convalida incrociata possiede un test singolo e tutto il resto dei dati è usato in training. Ovvero se i dati originali possiedono m esempi, allora si dice che la convalida incrociata è a m-fold. Questo metodo è normalmente

usato quando i dati disponibili sono molto piccoli. Non è efficiente per un data set molto grande come m classificatori necessari per essere costruiti.

5.4.1 Valutazione del classificatore in DM

Le classi che siamo interessati comunemente sono la **classe positiva** e la **classe negativa**, che però non è una misura molto adatta in questo caso, perché potremmo raggiungere un'accuratezza molto alta. **Precisione** e **richiamo** invece sono più adeguati perché misurano la precisione e completezza di classificazione sulle classi positive. Per realizzarli è conveniente utilizzare la matrice di confusione.
In riferimento alla tabella 5.1 dove:

- TP è il numero di corrette classificazioni del **vero positivo**;
- FN è il numero di errate classificazioni del **falso negativo**;
- FP è il numero di errate classificazioni del **falso positivo**;
- TN è il numero di corrette classificazioni del **vero negativo**.

Basando sulla matrice di confusione, la precisione (p) ed il richiamo (r) della positive class è definita come quanto segue:

$$p = \frac{TP}{TP + FP} \qquad r = \frac{TP}{TP + FN} \tag{5.8}$$

p è il numero di true positive classificati correttamente rapportato per il numero totale di dati che sono stati classificati come positivi. r è il numero di corrette classificazioni apportato per il numero totale di dati positivi veri. Sebbene in teoria la precisione e il richiamo non sono relazionate, ma concretamente un'alta precisione può essere sempre raggiunta allo dispendio di richiamo, e viceversa. Se abbiamo bisogno di una misura singola per fare confronto su classificatori differenti, l'**F-score** viene spesso usato:

$$F = \frac{2pr}{p + r} \quad \Longrightarrow \quad F = \frac{2}{1/p + 1/r} \tag{5.9}$$

La F-score è la media armonica di precisione e recall, la media armonica dei due numeri tende a essere più vicino al più piccolo delle due statistiche, anche se la F-score è alto allora entrami p e r dovrebbero alti.

Tabella 5.1: Matrice di confusione di un classificatore

	Classified positive	Classified negative
Actual positive	TP	FN
Actual negative	FP	TN

5.4.2 Misure di valutazione in IR

Sia D la collezione dei documenti nel database, e N il numero di documenti in D. Dato un'interrogazione utente $\mathbf{q}$, l'algoritmo di reperimento calcola per primo il punteggio di rilevanza per tutti i documenti in D e fornisce un classifica R_q dei documenti basato sul punteggio di rilevanza,

$$R_q : \langle \mathbf{d}_1^q, \mathbf{d}_2^q, ..., \mathbf{d}_N^q \rangle , \tag{5.10}$$

dove $\mathbf{d}_1^q \in D$ è il documento più rilevante per l'interrogazione $\mathbf{q}$ e $\mathbf{d}_n^q \in D$ è il documento meno rilevante per l'interrogazione $\mathbf{q}$. Sia $D_q (\subseteq D)$ l'insieme di documenti più rilevanti della query $\mathbf{q}$ in D, è possibile calcolare i valori di richiamo e precisione ad ogni $\mathbf{d}_i^q$.

Il **Richiamo** alla classificazione della posizione i, o documento $\mathbf{d}_i^q$ $(r(i))$ è la frazione di documenti rilevanti da $\mathbf{d}_1^q$ a $\mathbf{d}_i^q$ in R_q. Sia $s_i (\leq |D_q|)$ (dove $|D_q|$ è la taglia di D_q), il numero di documenti rilevanti da $\mathbf{d}_1^q$ a $\mathbf{d}_i^q$ in R_q. Allora,

$$r(i) = \frac{s_i}{|D_q|}. \tag{5.11}$$

La **Precisione** alla classificazione della posizione i o documento $\mathbf{d}_i^q$ $(p(i))$, è la frazione di documenti rilevanti da $\mathbf{d}_1^q$ a $\mathbf{d}_i^q$ in R_q:

$$p(i) = \frac{s_i}{i}. \tag{5.12}$$

Precisione media: Talvolta si vuole avere una precisione singola per fare confronto tra diversi algoritmi di reperimento sull'interrogazione $\mathbf{q}$. Una precisione media (p_{avg}) può essere calcolata basando sulla precisione ad ogni documento nella classificazione,

$$p_{avg} = \frac{\sum_{d_i^q \in D_q} p(i)}{D_q}. \tag{5.13}$$

Curva di Precisione e Richiamo: Basando su valori di precisione e richiamo ad ogni posizione di classificazione, è possibile disegnare

una curva di precisione-richiamo dove sull'asse delle x viene riportato il richiama e sull'asse delle y viene riportato la precisione. Con il quale è possibile fare confronto tra due tra i risultati di due tipi di algoritmi di reperimento diversi. Non è possibile ottenere i livelli di richiamo esatto nella classificazione, l'interpolazione è necessaria a questi livelli di richiamo. Sia r_i il livello di richiamo, $i \in 0, 1, 2, ..., 10$, e $p(r_i)$ la precisione al livello di richiamo r_i. $p(r_i)$ è calcolato come segue

$$p(r_i) = \max_{r_i \leq r \leq r_{10}} p(r), \tag{5.14}$$

per interpolare la precisione ad un livello particolare di r_i, e si prende la precisione massima di tutti i richiami tra il livello r_i e r_{10}.

Valutazione usando interrogazioni multiple. In molte valutazioni di reperimento, siamo interessati nella prestazione di un algoritmo su una grande dimensione di interrogazioni. La precisione complessiva ($\overline{p}(r_i)$) ad ogni livello di richiamo r_i viene calcolato come la media della precisione individuale che richiama il livello.

$$\overline{p}(r_i) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} p_j(r_i), \tag{5.15}$$

dove Q è l'insieme di tutte le interrogazioni e $p_j(r_i)$ è la precisione di query j al richiamo del livello r_i. Utilizzando la precisione media si può anche disegnare la curva di precisione-richiamo.

F-score. Un'altra misura di valutazione utilizzata è la F-score, tale per cui calcoliamo il punteggio F ad ogni posizione di rango i. Infine la precisione e il richiamo **break even point** è comunemente utilizzato pure.

5.5 Alcune considerazioni sui Metodi di Apprendimento

Regole di Associazione in IR. Nell'ambiente web, le regole di associazione sono molto utili perché molte tipologie di dati web sono in forma di transazione, come le interrogazioni di ricerca degli utenti e pagine cliccate da visitatori. Pertanto si vuole sapere se in che modo le attività utenti vengano associate alle pubblicità che

ad essi potrebbero piacere di vedere, tali quali potrebbero influire problemi di classificazione o di predizione. Tenendo presente che le transazioni non avvengono allo stesso tempo, ma uno dopo l'altro, il cammino è utile nel *web usage mining* per analizzare i dati di clickstream in server logs. Dopodiché sono anche utili per trovare modelli linguistici dai testi di natura linguistica.

LSI vs AR. È mostrato che LSI (*indicizzazione semantica latente*) svolge meglio rispetto i tradizionali metodi basati su parole chiavi. L'inconveniente principale è il tempo di complessità di SVD, tale che è $O(m^2n)$. È pertanto difficile usarlo per una grande collezione di documenti come il web. Un altra inconvenienza è il concetto di spazio che non è interpretabile come la sua descrizione che si consiste in tutti i numeri con un piccolo significato semantico. Determinare il numero ottimale di dimensioni k del concetto spazio è anche una difficoltà maggiore. Non vi sono consensi generali per un numero ottimale di dimensioni. Normalmente si suggeriscono 50-350 dimensioni. Nella pratica, il valore di k ha bisogno di essere determinato basando sulla collezione di documenti specifici mediante esperimenti ed errori, che è un processo quale richiede molto tempo, dovuto all'alta complessità del tempo di SVD. È possibile pensare che le regole di associazione (AR) possano essere capaci di approssimare i risultati di LSI e prevenire i difetti. Le regole AR elaborano le correlazioni dei termini. Inoltre possiede due vantaggi. Primo, il suo algoritmo di estrazione è molto efficiente. Dal momento in cui ha solamente bisogno di regole con 2-3 termini, tale che è sufficiente per scopi pratici, l'algoritmo di estrazione ha solamente bisogno di scansionare la collezione di documenti per 2-3 volte. Secondo, le regole sono facili da capire.

LSI vs modello vettoriale. Nella ricerca sul web, dovuto ai questioni di efficienza, l'indicizzazione a semantica latente probabilmente non è ancora stata applicata nella ricerca sul web. Attualmente l'algoritmo di ricerca quello più usato è ancora quello basato sul modello di spazio vettoriale e la combinazione di termini. Un motore di ricerca inizia con la scansione delle pagine sul web, dove le pagine scansionate

Applicazioni di Bayes. Abbiamo visto che la distribuzione di probabilità di Bayes è stata applicata in entrambi i casi di data mining e information retrieval. In DM il modello di Naïve Bayes viene usato per ottenere una classificazione e sulla appartenenza dalle osservazioni dei dati. Mentre in IR viene usato per dare un punteggio di rango di rilevanza a ciascun documento.

Applicazioni PU learning. Nelle ricerche su sistemi informativi, a causa del fatto che per la maggior parte delle volte, l'utente è interessato solamente in pagine web o documenti di testo di un argomento particolare. I problemi di PU learning si verificano molto frequentemente nel ambito del web. Per esempio, uno potrebbe essere interessato solamente in pagine relative ai viaggi (pagine positive), mentre tutte le altre pagine reperite vengono considerate pagine negative.

Segmentazione. La segmentazione è una tecnica molto comunemente usata in tecniche di analisi dei dati. Ha anche una storia molto lunga, e viene applicato in quasi tutti i contesti, come ad esempio in medicina, psicologia, botanica, sociologia, biologia, archeologia, marketing e molti altri. Negli anni recenti con la rapida crescita di documenti online e l'espansione del web, anche la clusterizzazione dei documenti è diventata un compito molto importante. Ha anche un ruolo molto importante in *web usage mining*.

Misure di valutazione. Nell'apprendimento supervisionato le misure di valutazione vengono usate per dare un giudizio sull'accuratezza del classificatore, dove le istanze delle osservazioni possono provenire da qualsiasi contesto. Nell'ambito di information retrieval, dove diversamente dal data mining, solitamente non si fanno valutazioni sull'accuratezza della classificazione di rilevanza 3/o non rilevanza dei documenti, è sufficiente fornire una classificazione dei documenti per l'utente. In altre parola la classificazione dei documenti generalmente si fa per soddisfare l'interesse dell'utente.

Capitolo 6

Analogie e differenze tra IR e DM

Quando si parla di data mining (DM) allora l'idea di base è quella di ottenere informazioni utili dai dati a disposizione per poi prendere le decisioni o fare previsioni in base ai propri obiettivi. Ad esempio, si potrebbe voler sapere se certi cognomi sono più comuni in certe regioni che in altre, oppure ottenere i documenti restituiti da un motore di ricerca e pertinenti alle informazioni di contesto e utili all'utente.

Il DM facilita la scoperta di conoscenza da basi di dati mediante un processo di estrazione delle informazioni implicite e sconosciute, mentre con l'information retrieval (IR) si vogliono reperire informazioni necessarie per raggiungere obiettivi precedentemente fissati. L'IR, come ad esempio l'accesso a pagine web, cataloghi online ed oggetti multimediali, è applicata quasi esclusivamente in ambito informatico. Mentre i dati che vengono trattati in DM sono di tipo numerico, in IR, invece, i dati trattati sono prevalentemente di tipo testuale. A prescindere dallo scopo di fare le decisioni, è facile intuire che le tecniche di DM, con o senza il supporto di strumenti informatici, può essere applicabile ovunque, sia nella ricerca scientifica sia nella ricerca del mercato.

Sia DM che IR sono ampiamente applicati nell'ambito informatico. Inoltre, è da tenere presente che diverse teorie matematiche e statistiche alla base di DM ed IR sono uguali o al più simili, tante idee formulate nell'uno sono analoghe a quelle formulate nell'altro, tuttavia, si differenziano dal modo di interpretare i problemi. Inoltre, tecnicamente è possibile affermare che diverse tecniche usate in IR sono identiche a quelle del DM, ma la tipologia di informazione

che si vuole ottenere è diversa. Dal punto di vista dell'apprendimento automatico, l'elaborazione preliminare dei testi e delle pagine può essere considerato, infatti, come apprendimento non supervisionato. Il processo di reperimento e quello di feedback di rilevanza possono essere considerati come apprendimento supervisionato.

Sebbene sia il DM che l'IR hanno uno scopo simile, ovvero, quello di ricavare informazioni utili dai dati, tuttavia, vi sono differenze notevoli sulla tipologia di informazione che restituiscono.

6.1 Text Mining

Con il termine *text mining* (TM) s'intende l'applicazione di tecniche di DM su documenti testuali, cioè, l'analisi di dati di tipo testuale. In altri termini, il TM è l'estensione del DM tradizionale su dati testuali non strutturati. L'obiettivo principale del TM è l'estrazione di informazione implicitamente contenuta in un insieme di documenti e la successiva visualizzazione di grossi insiemi di testi. Il TM è un campo più complicato del DM, perché opera su testi che non sono strutturati. È un campo multidisciplinare che impiega sia tecniche proprie dell'IR che tecniche proprie del DM. Le tecniche di TM sono applicabili a qualsiasi ambito di indagine. In generale, trovano applicazione tutte le volte che si è di fronte a grandi quantità di dati e si ha l'esigenza di conoscerne il contenuto. Le applicazioni possono essere tecniche di segmentazione al fine di (i) individuare gruppi omogenei di documenti in termini di argomento trattato, (ii) accedere più velocemente all'argomento di interesse e (iii) individuarne i legami con altri argomenti. Il TM ed il DM, dal punto di vista di apprendimento automatico, sono in contraddizione. I modelli sono costruiti da training set di campioni dei documenti non strutturati, e i risultati sono verificati in fase di test. L'obiettivo chiave di preparazione dei dati è la trasformazione dei testi in un formato numerico, eventualmente viene fatta la rappresentazione comune con il DM numerico. Anche in text mining come nel DM, vi sono introdotti strutture di predizione, tra le quali classificazione dei documenti, segmentazione dei documenti, estrazione delle informazioni, e valutazione delle prestazioni.

Le fonti dei dati di TM possono essere:

- **Dati web:** Internet sta diventando il principale "media" attraverso cui è possibile ottenere dati. I siti web liberamente raggiungibili via Internet sono una delle fonti principali della documentazione da analizzare (filtraggio informazioni).
- **Banche dati online:** Le banche dati online costituiscono collezioni di informazioni specializzate, generalmente accessibili via Internet tramite abbonamento. Esempi tipici di queste banche dati sono quelle dedicate alle pubblicazioni, ai brevetti o agli articoli scientifici (di chimica, fisica o matematica) rese disponibili in modo diretto.
- **Sorgenti informative private:** Una banca dati privata di documenti elettronici (costruita negli anni) può essere resa disponibile ed essere opportunamente usate insieme alle altre sorgenti informative. Il formato e contenuti dei documenti di una banca dati sono generalmente completamente differenti da quelli dei documenti ottenuti attraverso le banche dati online.
- **E-mail:** Le e-mail sono la forma più ricca dal punto di vista informativo e più semplice da analizzare. È il mezzo attraverso cui le persone comunicano all'interno ed all'esterno di aziende ed organizzazioni. Possono essere analizzate sia le e-mail interne ad un'organizzazione sia quelle ricevute dall'esterno od inviate all'esterno dell'organizzazione.
- **Opinion surveys:** Spesso le opinion surveys accurate nella parte codificata, dove è prevista la risposta: SI, NO, o numerica. Sono invece analizzate con un certo grado d'incertezza nella parte testuale, ove si raccolgono le risposte in testo libero alle domande aperte.
- **Newsgroups, Chatlines, Mailing Lists:** Importanti e ricche fonti di informazione dato che riguardano i temi più disparati, dai consumi alla politica. Il problema con questo tipo di informazione è che l'informazione pertinente è all'interno di frasi e/o affermazioni di scarsa importanza, espresse con linguaggio spesso gergale. Grazie al text mining queste affermazioni o opinioni possono essere analizzate e filtrate al fine di conoscere quali sono le opinioni di chi scrive.

Documenti di testo. Un documento di testo consiste di una sequenza di sequenze e, ciascuna sequenza in una sequenza di parole. Usualmente, in clusterizzazione, un documento è considerato come un insieme di parole. Le informazioni sulla sequenza e sulla posizione delle parole, in linea generale vengono ignorate. Pertanto, un documento può essere rappresentato come un vettore esattamente come un normale data point. Inoltre, vengono usate le misure di similarità per mettere in confronto due documenti piuttosto della funzione di distanza. La funzione di similarità più comunemente usata è la funzione similarità di *coseno*.

6.1.1 IR come una forma di TM

In un modo o nell'altro, l'IR può essere descritto come in termini di previsione text mining. I metodi possono essere considerati varianti di misure di similarità basate sul metodo di *nearest-neighbor*. Come in IR, in risposta a un'interrogazione, si vanno a reperire documenti rilevanti, allo scopo di esaminare la collezione di documenti, apprendere criteri per la classificazione, ed applicare questi criteri ai nuovi documenti. I problemi della predizione non vengono risolti direttamente a seguito della ricerca dei pattern nella collezione di documenti, ma si vanno a reperire piuttosto i documenti simili. Dal fatto che siamo interessati nella classificazione, si tengono conto le etichette per vedere quale etichetta dovrebbe essere assegnato al documento nuovo e non etichettato.

6.1.2 Segmentazione e Predizione dei Testi

Frequentemente, le collezioni di documento vengono preparate senza etichette. Le etichette possono essere determinate dalla segmentazione dei documenti. Una ragione importante per la segmentazione dei documenti è il calcolo delle misure di similarità. I metodi di raggruppamento più noti sono: clusterizzazione a k-medie, clusterizzazione gerarchica, algoritmo EM. Una volta che i testi sono stati trasformati in vettori numerici, si possono allora applicare metodi automatizzati di predizione. La predizione dei testi è descritta in termini di analisi empirici che possono essere relazionate a patterns delle parole, in particolare per la classificazione dei documenti. Le

tecniche fondamentali di apprendimento dagli esempi di dati vengono delineati includendo i metodi basati sulla similarità, regole di decisione e alberi per la classificazione, metodi probabilistici e metodi lineari. Le tecniche di valutazione sono esaminate per stimare la prestazione futura e massimizzare i risultati empirici.

6.2 Web Mining

Il *web mining* (WM) mira a trovare informazioni o conoscenze da **strutture ipertestuali del web**, **contenuto della pagina**, e **dati di utilizzo**. In questi ultimi, vengono applicati entrambi gli algoritmi di DM basandosi sulle teorie di IR per scoprire le regolarità nell'ambito del web. Dovuta alla natura eterogenea, semi-strutturata e non strutturata dei dati del web, il WM utilizza le tecniche di DM, però non è puramente un'applicazione di DM.

In WM, inoltre, si affrontano i problemi causati dall'accesso di informazioni disponibili su Internet, tra i quali vi è quello della classificazione automatica di pagine web in base al loro contenuto testuale. I classificatori sono costruiti a partire da pagine rappresentative fornite da un singolo utente o da un gruppo durante l'attività di ricerca in rete.

Il processo di WM è simile al processo di DM e si differenzia per la collezione dei dati. Con gli algoritmi DM tradizionali, i dati sono già raccolti e salvati in una basi di dati. In WM, si considera che la collezione dei dati sia un passo fondamentale. In particolare in *web structure mining* ed *web content mining*, dove viene trattata la scansione di un grande numero di pagine web. Una volta collezionati i dati, i passaggi successivi perlomeno sono analoghi rispetto a quello del DM tradizionale.

L'obiettivo di WM trova giustificazione nell'opinione diffusa che l'informazione presente nel web è sufficientemente strutturata da consentire una efficace applicazione di tecniche statistiche e di apprendimento automatico. In base al tipo di reperimento che si vuole effettuare, il WM può essere suddiviso in seguenti tre categorie:*web structure mining*, *web content mining*, *web usage mining*.

Web structure Mining. Il *web structure mining* (*o generalizzazione*) trova conoscenze utili da collegamenti ipertestuali (*links*), che rappresentano strutture del web. Per esempio dai links possiamo trovare delle pagine web importanti, che per inciso, è una tecnologia chiave usata in motori di ricerca. Può anche essere usato per esplorare le comunità di utenti che condividono argomenti comuni. Nel DM tradizionale non si eseguono funzioni come queste perché solitamente non ci sono collegamenti in tabelle relazionali.

Web content Mining. Il *web content Mining* (*o estrazione di informazioni*) estrae o scava informazioni o conoscenze utili dai contenuti delle pagine web. Per esempio possiamo automaticamente classificare e raggruppare le pagine web in base al loro argomento. Queste funzioni sono simili rispetto a quelle nei DM tradizionali. Tutt'al più possiamo anche trovare modelli nelle pagine web per estrarre dati utili come descrizioni di prodotti, posting dei forum, ecc. Inoltre possiamo rilevare recensioni e postings dei forum, per conoscere i pareri dei clienti. Questi di fatto non sono funzioni di DM tradizionali.

Web usage Mining. Per *web usage Mining* (*o scoperta di risorse*) si riferisce alla scoperta del modello degli accessi di utenti dal *web usage logs*, che registra ogni click fatto da ciascun utente. Il web usage logs applica molti algoritmi di DM. Una delle questioni chiave in questa categoria di WM è il pre-processing dei flussi di dati di click nel usage logs in modo da produrre i dati giusti per il mining.

6.2.1 Classificazione delle pagine Web

La costruzione di un profilo di interessi relativo a pagine web passa per la soluzione di due problemi: decidere quale informazione è interessante e determinare le sue modalità di estrazione dalle stesse pagine web. Per quanto riguarda il primo problema si assume che la rilevanza di una pagina web dipenda essenzialmente dal suo contenuto testuale, quindi criteri di giudizio "esterni", come il carattere di novità del documento o l'affidabilità dell'informazione, ed eventuali contenuti di tipo non testuale della pagina, non vengono

considerati. Inoltre, si assume che l'utente sia in grado di specificare un insieme di classi corrispondenti ai vari argomenti di interesse, e di fornire un insieme di esempi significativi per ciascuna delle classi (insieme di addestramento). Per il secondo problema è importante definire sia il linguaggio di rappresentazione da utilizzare per descrivere le parti testuali delle pagine HTML e sia le tecniche di DM in grado di generare la conoscenza classificatoria, cioè il profilo di interessi, da utilizzare in un processo di classificazione automatica di pagine web.

Metodi di apprendimento utilizzati

L'interazione dell'utente o del gruppo di utenti con il sistema informativo avviene in due passi: inizialmente l'utente naviga sul web e colleziona i riferimenti a pagine significative delle classi di interesse (pagine di addestramento). Al secondo passo il sistema assiste l'utente nella navigazione classificando autonomamente le pagine web. La decisione di sospendere la raccolta delle pagine di addestramento e di invocare i servizi di assistenza è responsabilità dell'utente o dell'amministratore del gruppo di utenti. La predizione della classe di appartenenza di una pagina web è realizzata attraverso tre modalità alternative:

1. **alberi di decisione**, costruiti per induzione a partire dalle pagine di addestramento;
2. **distanza dei prototipi** di classe costruiti a partire dalle pagine di addestramento;
3. **algoritmi di nearest-neighbor**, basati sulla valutazione della similarità rispetto alle pagine collezionate.

Le prime due modalità prevedono la costruzione dei classificatori (alberi di decisione o prototipi) in una fase di addestramento (training del sistema). Invece, la terza modalità non precostruisce classificatori espliciti per le classi, bensì usa le pagine selezionate direttamente durante la fase predittiva. Gli alberi di decisione sono generati utilizzando metodi di apprendimento automatico, particolarmente adatti ai domini con forte prevalenza di attributi numerici. Infatti, il sistema è anche in grado di indurre alberi decisionali multivariati, benché questa capacità non sia stata sfruttata

nel nostro caso. Il sistema di apprendimento genera un albero decisionale univariato per ogni classe di addestramento, considerando come esempi positivi tutte le pagine rappresentative della classe e come esempi negativi le rimanenti. In questo modo il risultato della classificazione di una nuova pagina potrà essere:

1. **nessuna classificazione**, ovvero la pagina è rigettata poiché non riconosciuta come appartenente a qualcuna delle classi definite dall'utente;
2. **classificazione singola**, la pagina è assegnata ad una sola classe fra quelle predefinite; di classe costruiti a partire dalle pagine di addestramento;
3. **classificazione multipla**, la pagina è attribuita a più classi fra quelle predefinite.

Per quanto riguarda la seconda modalità di predizione, il sistema di apprendimento è in grado di classificare una pagina web sulla base della similarità tra il vettore rappresentativo della pagina e un prototipo calcolato per ogni classe. Il prototipo di ciascuna classe è definito come il vettore centroide fra tutti i vettori relativi alle pagine web di addestramento della classe. Una nuova pagina viene attribuita alla classe il cui prototipo risulta essere meno distante da essa. La distanza è calcolata in funzione del coseno dell'angolo formato tra il vettore prototipo e il vettore rappresentativo della pagina. Relativamente alla terza modalità, è stato implementato l'algoritmo di k-nearest neighbor (k-NN). Anche in questo caso la distanza è calcolata in funzione del coseno dell'angolo formato tra i vettori all'interno dello spazio vettoriale. Il k-NN attribuisce la classe di appartenenza della pagina web da classificare analizzando la distribuzione delle classi di appartenenza dei k esempi di addestramento meno distanti. Come noto, uno dei limiti del k-NN è l'**intolleranza** agli attributi irrilevanti. Tuttavia, nel nostro caso il problema è meno evidente poiché la selezione degli attributi è già stata effettuata in precedenza, in modo da ridurre la presenza di attributi poco significativi.

Capitolo 7

Conclusione

Nella tesi è stato fatto un confronto tra le caratteristiche principali di information retrieval (IR) e data mining (DM). Questi due nello stesso dominio di applicazione, ciascuno gioca un ruolo ben preciso, ognuno ha i compiti diversi e specifici da svolgere. Per mettere in pratica l'IR e DM, abbiamo visto che a entrambi essi risultano immancabili i campi di studio che fanno da supporto come *machine learning*, *data warehouse*, *basi di dati*, *statistica*, *matematica* e molti altri ambiti inerenti al contesto di applicazione. Inoltre occorre tenere presente che tutto ciò nasce dalla continua crescita dei dati. È stato citato che i compiti principali di DM sono quelli di estrazione di conoscenza da bande di dati che sono in continua crescita, mentre i compiti di IR sono quelli di estrarre esigenze informative agli utenti.

In altre parole, in DM si applicano varie tecniche matematiche-statistiche per l'estrazione delle informazioni dai dati, dove i dati possono essere di qualunque tipo. Tuttavia in questo elaborato sono state descritte soltanto alcune tecniche tra quelle comunemente usate. L'idea di base di IR è quella di reperire informazioni interessanti dai dati. Sulla base dei dati, sono stati formulati diversi modelli di reperimento, e in base al contesto, per reperire informazioni interessanti si applicano le opportune tecniche di data mining. Le tecniche del DM possono essere applicate in qualunque ambito, da quello umanistico a quello economico, oppure da quello scientifico a quello ingegneristico. Information retrieval invece tende a dominare soprattutto il mondo informatico, in particolare nelle tecnologie web. Per la vastità dei dati web, e dati testuali a disposizione salvati nei grandi databases, nel giorno d'oggi, il data mining

insieme all'information retrieval hanno portano a formalizzare altri campi di studio di data science, come il text mining e web mining.

Bibliografia

[1] Melucci M. (2013). *Information Retrieval. Metodi e modelli per i motori di ricerca.*

[2] Liu B. (2007). *Web Data Mining. Exploring Hyperlinks, Contents, and Usage Data.* 1-264.

[3] Scarpa B. Azzalini A. (2004). *Analisi dei dati e data mining.*

[4] Manning Christopher D. Raghavan P. Schütze H. (2008). *Introduction to Information Retrieval.*

[5] Weiss Sholom M. Indurkhya N. Zhang T. (2010). *Fundamentals of Predictive Text Mining.* 75-90.

[6] Melucci M. (2013). Basi di Dati. 177-237.